Geometry matching by multi-agent systems

Changing GFRP from an environmental hazard to a façade design solution

Geometry matching by multi-agent systems

Changing GFRP from an environmental hazard to a façade design solution

Student:
Lieuwe Thys Meekma

Student number:
4109333

Members of graduation committee:
M. Turrin - First Mentor
Architectural Engineering + Technology & Design Informatics

M. Bilow - Second Mentor
Architectural Engineering + Technology & Product Development

T. Jylhä - External Examiner

Delft University of Technology
Faculty of Architecture and the Built Environment
Master Architecture, Urbanism and Building Sciences
Building Technology Track

Cover art: Tommaso Casucci

Tuesday, June 26, 2018
Delft

Acknowledgement

'Geometry matching by multi-agent systems' is a graduation thesis for the master track Building Technology at the faculty of Architecture and the Built Environment of Delft University of Technology. This thesis is written by Lieuwe Thys Meekma with the help and input of others. The creation of this thesis has been an interesting challenge. Both in terms of the content creation and on a personal level.

However, this work would have not been laying before you if not for the help I've received. First of all I would hereby like to thank my mentors. Starting with Michela Turrin. She has continuously guided me, from the first ideas towards the finalisation of the thesis. Whenever I faced a tough decision or lost some of my motivation, the combination of both her knowledge and positive attitude towards my ideas helped me overcome many doubts. For the topic of computational design she was always able to help me to further explore the boundaries of this topic within my thesis. I would also like to thank Marcel Bilow. I have yet to see a moment where Marcel's energetic stance towards new ideas from student is wavering. His knowledge and experience of materials and fabrication techniques have helped tremendously to form a good idea of the overall building method around the computational part of this thesis. Finally I'd like to thank both Marcel and Michela for their understanding, patience and positive attitude during all the consults we've had together.

Apart from my mentors I would also hereby like to thank all my friends who have helped me during this process with either some motivation, understanding or just some fun and distraction during our breaks. Special thanks goes out to Lisette. She has helped me in so many ways during my thesis, her care, patience, understanding and ever lasting support have been and always will be an invaluable asset to me. And last but certainly not least a special thanks goes out to my parents, without them I would not be where I am today.

Abstract

The idea for Geometry matching by multi-agent systems as a subject originates at a previous course, where the J. Sturkenboom & author came up with an idea to use the material of old boat hulls as facade elements to provide a solution to an environmental problem. The approach then was rather practical, whereas the approach within this thesis is a theoretical one where a focus lies on solving the computational challenges this subject provides.

This thesis proposes a work-flow that includes an algorithm that finds matches between the material and design. The overall work-flow attempts to provide a solution for an environmental problem by creating a facade system that uses a discarded material as panelling. The ability of the algorithm to find matching shapes is provided by agent based modelling. This computational technique uses agents, which can be described as a computer system situated in some environment, that is capable of autonomous action in this environment in order to meet its design objectives. The outcome of the algorithm within this thesis are cutting positions on a 3D boat geometry, which provides a basis for the code that is able to instruct machinery to process and produce the design. This process is tested digitally on a case study design as a starting point for such a work-flow. The intention of this work-flow as a whole is not to provide architects with a solution that creates perfectly smooth facade designs using discarded material, but as a new direction into which designers can venture where the algorithm can be used to materialize a design idea as closely as possible with discarded materials. The outcome is a logic that defines the form finding process and a work-flow in which it is situated, both combined form a starting point which is to be developed in the future in order to move this idea from idea on paper to built in reality.

Table of Content

Introduction

'What is my job on the planet? What is it that needs doing, that I know something about, that probably won't happen unless I take responsibility for it?'

- Buckminster Fuller

1. Research framework

1.1 Introduction

Architecture and the spaces in which we reside are influenced and changed by the development of techniques, as Robin Evans states in his book, '*The Projective Cast*' (Evans, 1995). This refers back to the development of techniques and the influences it had on the built environment during the Gothic and early Renaissance times. After another change in production techniques during the industrial revolution, a similar effect was noticed. Change has taken place within architecture and the way we design them as new production technologies once again emerge and bring in a new view on architecture and provide a new approach to design (Kolarevic, 2003b). Nowadays the influx of computational techniques into architecture results in an even further diminished line between concept, shape and production.

The quick evolution of digital design in architecture together with new fabrication and construction technologies have provided architects with opportunities to move away from standardization and traditional design approaches that predicate themselves on symmetry and repetition (Gerber & Pantazis, 2016). We have gained the ability and computing power to move towards a fitting solution for non-standard designs.
All these changes offer possibilities and provide designers with a whole new spectrum of options. The renewal of the digital design process with both computer aided design (CAD) and computer aided manufacturing (CAM) tools enables the integration of material-, production- and assembly information into the early stages of design.

CAD and CAM play a pivotal role within architecture nowadays, yet a large part of the building industry seems to rely on using these techniques with a mind-set that is founded on a basis created by the industrial revolution and the repetitive nature of material production. This way of producing and dealing with design issues does not play into the strengths of the technology we have available. A seemingly small portion of the building industry is using the potential that CAD and CAM offer.

Nevertheless, to this day, materials still inform the design process. Yet we now are able to embed relevant information into the design generation process. Knowing what impact your material has, what it can and cannot do still impacts design and has significant relevance when making design decisions. This process is moving to the digital realm of designing with a goal of ultimately generating better performing designs in an earlier stage of the design process. The ability to include such information in a digital design optimisation process presents new challenges. The broader range of design objectives that have to be dealt with inevitably lead to more conflict between these objectives. With ever increasing abilities and digital tools available, the complexity of shapes found in design also increases. Double curved surfaces are barely a rarity anymore in architecture. Yet building such projects remains a challenge.

In architectural practice a multitude of possibilities are available to reach objectives through optimisation software. One of the techniques available for optimising and finding a solution is agent based modelling (ABM). Agents are a software based computer system where each agent system has an environment in which the agents are placed. The agent's behaviour is based on the ability to interact with the environment and other agents in the system. An agent can have an objective that influences it's behaviour, yet it does not only maximise that objective, rather it is used as information towards the decision making process(Macal & North, 2009).

A flourishing interest exists within architectural design in agent based modelling and simulation (ABMS). These and non-linear systems can be used as a means to not only improve the design process but also to improve design exploration through adaptation and emergence, facilitating the production of higher performing design outcomes without reducing geometric intricacy (Gerber, Pantazis, & Wang, 2017).

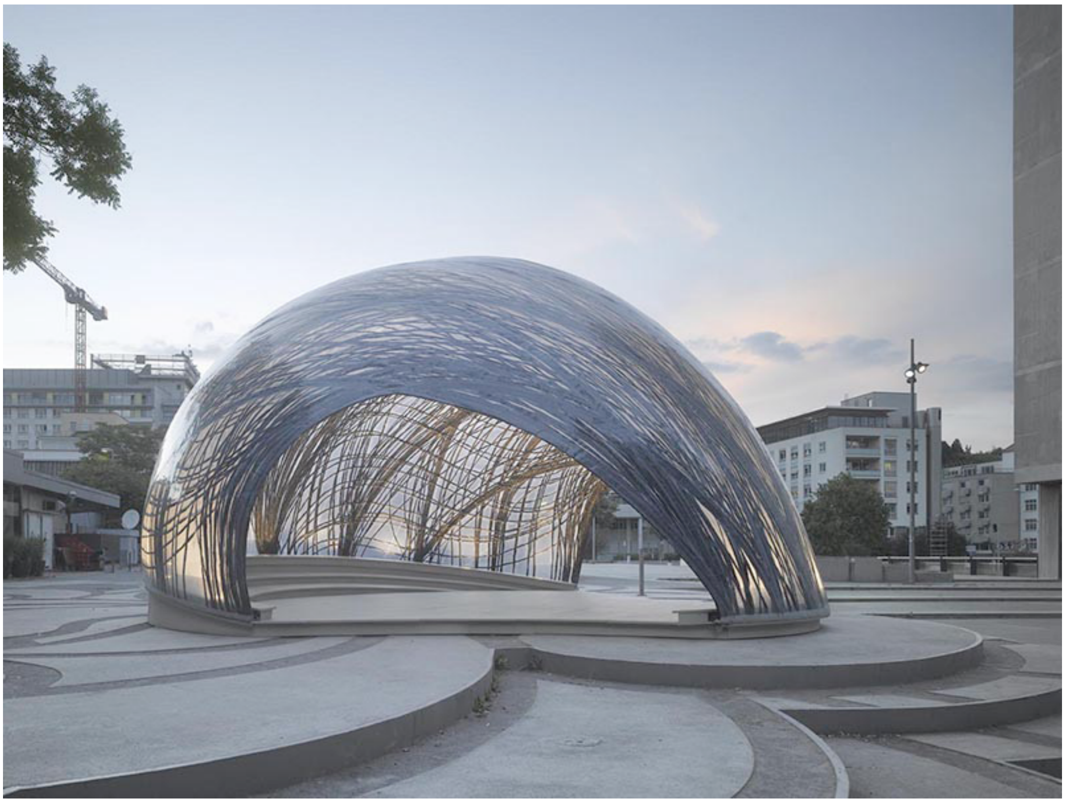


Figure 1 ICD/ITKE Research Pavilion 2014 - 2015, University of Stuttgart

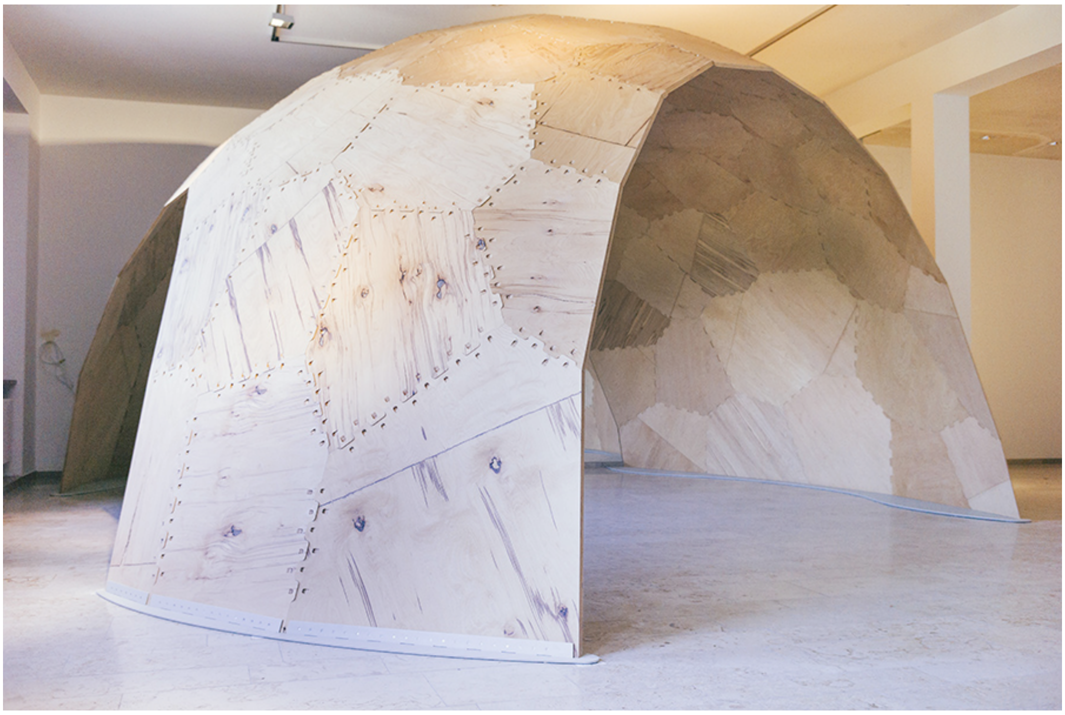


FIGURE 2 SEGMENTED TIMBER SHELL, BY THE INSTITUTE OF COMPUTATIONAL DESIGN AND CONSTRUCTION, UNIVERSITY OF STUTTGART

Such developments in techniques are sometimes showcased within pavilions. An example of such a research pavilion is the 2014-2015 research pavilion of the Institute of Computational Design and Construction at the University of Stuttgart. In the project, shown in Figure 1, an agent-based design method determines the fibre layout.

In the case of the research pavilion in Stuttgart, carbon fibre strands have been used as a material. Other materials, such as wood, lend itself well since incremental changes can easily be made in the material, as it is easily machined. Harder, denser and stronger materials, such as steel, become harder to machine and take a longer, thus being harder to process during the manufacturing process. Therefore it is not surprising to see other such research pavilions being made out of wood. Krieg, Schwinn and Menges describe an example where agents integrate requirements of fabrication, biomimetic principles and aesthetic criteria in a wooden shell structure(Krieg et al., 2015). The outcome of this research is displayed in Figure 2. When materials are getting harder to process, the material processing and the subsequent optimisation of it increases in significance. This is also the case for fibre-reinforced plastics, known as FRP. These materials are known to be tough, relatively strong material compared to their weight(Knippers, Cremers, Gabler, & Julian Lienhard, 2011). Properly processing such materials gains more significance when these materials reach the end of their initially intended life cycle and need to be recycled or reused. Unfortunately, the Netherlands, among other counties, is dealing with an environmental issue caused by such a FRP, namely glass fibre-reinforced plastics, GFRP.

This research aims to provide a starting point towards a solution for two main difficulties in the built environment, one the one hand the high cost and complexity of building the components necessary for curved shapes in architecture and on the other hand the surplus of materials that are difficult to re-use or recycle and thereby are an environmental hazard. The combination of these two topics converge in this research where we try to match curved surfaces found in both design and material together.

This starting point is an overview of how this process can work in practice. Starting at scavenging the materials to cutting it in the proper shape and size before actually placing the material on a architectural project. Yet, this research does not aim to achieve the perfect surface continuity or aesthetic qualities. Rather it tries to provide a basic understanding of what steps need to be taken to tackle such a problem. This is done through a case study design and a material that causes an environmental issue in the Netherlands. A software algorithm is created that tries to matches the curved shapes found in the case study design with the shapes of the material that causes environmental problems.

A generation that purchased a considerable amount of this material in the shape of boats during the 1970's and 1980's is growing older. As a result, the boats they bought are slowly but surely decommissioned. The amount of boats is approximately 25.000 in 2014 and is expected to grow up to 35.000 in 2030 (WA Yachting Consultants, 2015). Right now these boats weigh in at 1.400 tonnes of GFRP waste material a year (Ten Busschen, Bouwmeester, & Schreuder, 2016). Another 1.300 tonnes a year is created by decommissioned rotor blades from wind-turbines, all which is a part of a total 4.500 tonnes of GFRP waste material a year. As society's awareness of the urgent need to use renewable materials for building construction is raising, materiality and the impact it has on our environment has become an important parameter for architectural production (Weinand & Hudert, 2010).

This research aims to provide a solution for the glass fibre-reinforced polymer issue by using agent technology for the purpose of matching geometric properties of the a design input with that of leftover material, which in this case is glass fibre-reinforced plastic. Within the research the panelling of a case study design façade will be partly generated out of material shapes found in the GFRP.

The case study design is a pavilion designed by J. Sturkenboom, R. Wisse & L.T. Meekma for a previous study course. This design comes with an already determined panelling system in place. The matches that need to be made have a goal of finding a shape in a boat hull that most closely resembles that of the design. The material is represented by one type of boat, the Defender. The digital 3D shape of this boat represents the area that is searched for each panel to find a solution. The research uses the 3D model of one boat type and an already predetermined design in order to create an algorithm which is part of a broader work-flow, all of which can be considered as a starting point towards tackling the issues presented.

1.2 Problem statement

This research aims to provide a solution for a problem that is twofold. Firstly there is a material available that causes an environmental issue in the Netherlands and secondly the ever increasing objectives within architectural design optimisations.

Architecture can be considered to be an ever-lasting creative battle to find a certain language of form and shape. Some architects and their offices strive for a certain style to be continued throughout their projects, others try to diversify their project portfolio. Either way, the availability of current day computational tools continues to extend the typological design spectrum, resulting in ever more diverse and competing objectives (Gerber, Shiordia, Veetil, & Mahesh, 2014).

> *Aesthetically, it is the elegance of ordered complexity and the sense of seamless fluidity, akin to natural systems that constitute the hallmark of parametricism* (Schumacher, 2009).

Schumacher argues for a new design style, parametricism, as a way to capture all of the designs that are often marked by rounded edges, double curved surfaces and complexity in general. Whether this properly encompasses all of the output that parametric tools generate within architecture is debatable. It does however articulate the fact that the use of parametric tools is widespread within architecture and causes ever increasing complexity of shapes and data flows. In order to handle data generated by diverse, competing objectives in complex architecture, optimisation algorithms are employed to gain meaningful insight into the design processes. Complexity increases quickly by adding design objectives and the parameters that define the objectives.

To often building construction is a socially irresponsible process, that exhaust resources by unsustainably producing materials that are transporting globally (Gruber & Imhof, 2017). We build all the time, but very seldom account for the full lifetime of the materials that a building consists of and unfortunately the same goes for other industries.
As a result, among other materials, there is a great amount of non-deteriorating material in the Netherlands, in the form of glass fibre polyester hulls which currently knows no use (Ten Busschen et al., 2016; WA Yachting Consultants, 2015). Both the non-deteriorating character of the material and the fact that it is considered to be hard to process makes it an environmental issue. One of solutions for the re-use of the material

is chipping it, to later be used in fibre reinforced concrete. The other is reversing the chemical process that took place when the material was created. Both of these methods have the potential to be applied in an industrial setting and deal with large amounts of material. Yet they are also destructive methods that completely demolish the properties and values the material has such as the strength and complex geometric shapes.

Using material data and computational capacities are key aspects that offer earlier and easier insight into the design process of complex architectural geometries. In order to do so we have to close the gap between the shapes found in a design and in that of the material. The problem exists since there is no clear match between the two. A rectangular design can easily be imagined to be built out of straight building elements. Imagining this for a flowing, curved design is significantly more complex. In order to generate some understanding of this complexity and to deal with the problem these shapes have to be linked and the deviation between the two should be brought back to a minimum so designers, builders and other people involved will be able to easier understand and see the limitations and potentials when trying to re-use single or double curved material elements for an architectural application. We have the availability of necessary computing power, the ability to generate material data and technology to process it. Thus giving ourselves a possibility to take on both problems stated above.

1.3 Research objective

The objective of this research is to create a work-flow that has the ability to match geometry of the design intent with available geometry found in already existing material stock.

This is actually an objective that is twofold. The specific form finding process uses principles of agent technology in order to find a match of form between design and material. If done properly, the same logic used for the specific material and design case in this research can also be applied to other design goals or on other materials, therefore the logic behind such a process should be carefully explained to maintain its value for other potential applications. Also the use of agent technology attempts to generate some insight into the impact of multi-agent systems for form finding in architectural application.

Furthermore the context of the form finding algorithm is relevant since it will help to generate better understanding into the overall process into which it fits and how it might be developed further in the future. For this research specifically, the place of the form finding algorithm in architectural context is discussed, how can it be used and what other steps are necessary that need to be taken in order to use this work-flow in reality.

Moreover, the cross-reference between source material and a design shape will offer a solution to a prevalent environmental problem within the Netherlands. The objective of this research in relation to the environmental problem of the glass fibre-reinforced plastics (GFRP) boats should not be considered to be a final solution to the full extent of the environmental issue. The simple reasoning for this is that we are dealing with amounts of material that seemingly exceed an application such as façade cladding of even a relatively big building. It can however, add another option to the mix of solutions to this problem.

The outcome of this research should be considered a starting point of re-using materials as facade design system in architectural application. It does not intend to create an algorithm that will directly generate perfectly smooth, flowing architectural designs. Even tho the goal of this work-flow is to minimize deviation and get as close as possible to the previously mentioned smooth design, the outcome might also generate a new appreciation for the re-use of materials and thereby hopefully gaining more awareness for the environmental issues that my generation is dealing with. The outcome of the logic is strongly dependant on the input that is given by the source material. If that material has simple shapes, the outcome will also be simple. But the logic and algorithm still hold the same value, therefore it is of importance to generate a solid logic behind the algorithmic approach.

Overall the objective of this research is to show the value of emergent agent behaviour as a non-standard and geometrically intricate solution for design form finding in used materials. Thereby simultaneously generating more awareness for the way in which materials are used in architectural application. Possibly building the foundation for the logic to be applied in reality.

1.4 Research questions

Based upon the problem statement and the research objective, the following research questions have been formulated:

How can a multi-agent system match geometrical properties of curved surfaces?

To answer this question, the following sub-questions have been formulated:

What geometric properties define a curved surface?

How can geometry properties drive the behaviour of agents?

1.5 Relevance

The building industry and society as a whole are becoming more aware of the challenges that the environment provides us with as our current day reality. As we try to tackle them, multiple problems arise that we have not seen before. How do we properly demolish an object or building? Do we demolish it and with that it's structural integrity or do we disassemble, and if so, how? How do we create materials that can be given a second life? What do we do with materials that do not seem to have another life? These questions become more and more relevant and for one of them this research proposes a small step in the direction of a solution. A seemingly indestructible material that is hard to recycle is given another function, postponing its normal end-of-life.
The method used to achieve this objective uses an optimisation technology residing in a nascent state when it comes to architectural application. By using it we will try to add towards a body of technological knowledge where this agent technology belongs. The end result is a placement of façade panelling and the accompanying cutting lines onto already used material. If, by using this logic, any seemingly not recyclable material reaches an architectural application and raises a persons awareness about the significance of the environmental challenges, this research will have deemed itself relevant.

1.6 Methodology

In order to answer the questions posed by this research, the report is divided into main parts. These are the introduction, research, specification, application and evaluation.
After the introduction, the first step is research which will dive into the topic of agents in chapter 2. This provides an overview of what the agents are in theory and how they function. This step is necessary to generate a fundamental understanding of agents and how they can be applied in an architectural application that concerns geometric complexity.

Continuing with the research part of the report into geometric properties of surfaces and how they are defined, this is done in chapter 3, Surface geometry. This chapter clarifies how curvature is defined and what types of analysis can be done in order to generate data specifying surfaces types. This information is later needed as a component that can influence the agent's behaviour.
After the topic of surface geometry, the specification is made for this particular research. This explains what material is used and what kind of design this research deals with. A short summary of the material is given in chapter 4 and the case study design is explained in chapter 5.

In the application part of the report, the specified design case and material and the previously gained knowledge from research are applied on the specific setting of this research. The application part starts with chapter 6, Algorithm basic setup. This entails how geometric properties provide a metric that can influence the emergent behaviour of the agents, furthermore the right balance of behavioural rules is determined in this chapter.

In the next chapter of the application part in the report, the logic is explained which describes how the agents behave on the material surface. How they find the positions on the material to converge towards and how agents generate the accompanying cutting lines that are needed to extract the actual panel from the source material. This process is made as an algorithm in the 3D modelling program Rhino 3D and its accompanying plug-in Grasshopper (Robert McNeel & Associates, 2018). Furthermore a plug-in to simulate the agent behaviour is used, called Quelea (Fischer, 2015). This is a plug-in used in the Grasshopper visual coding environment.

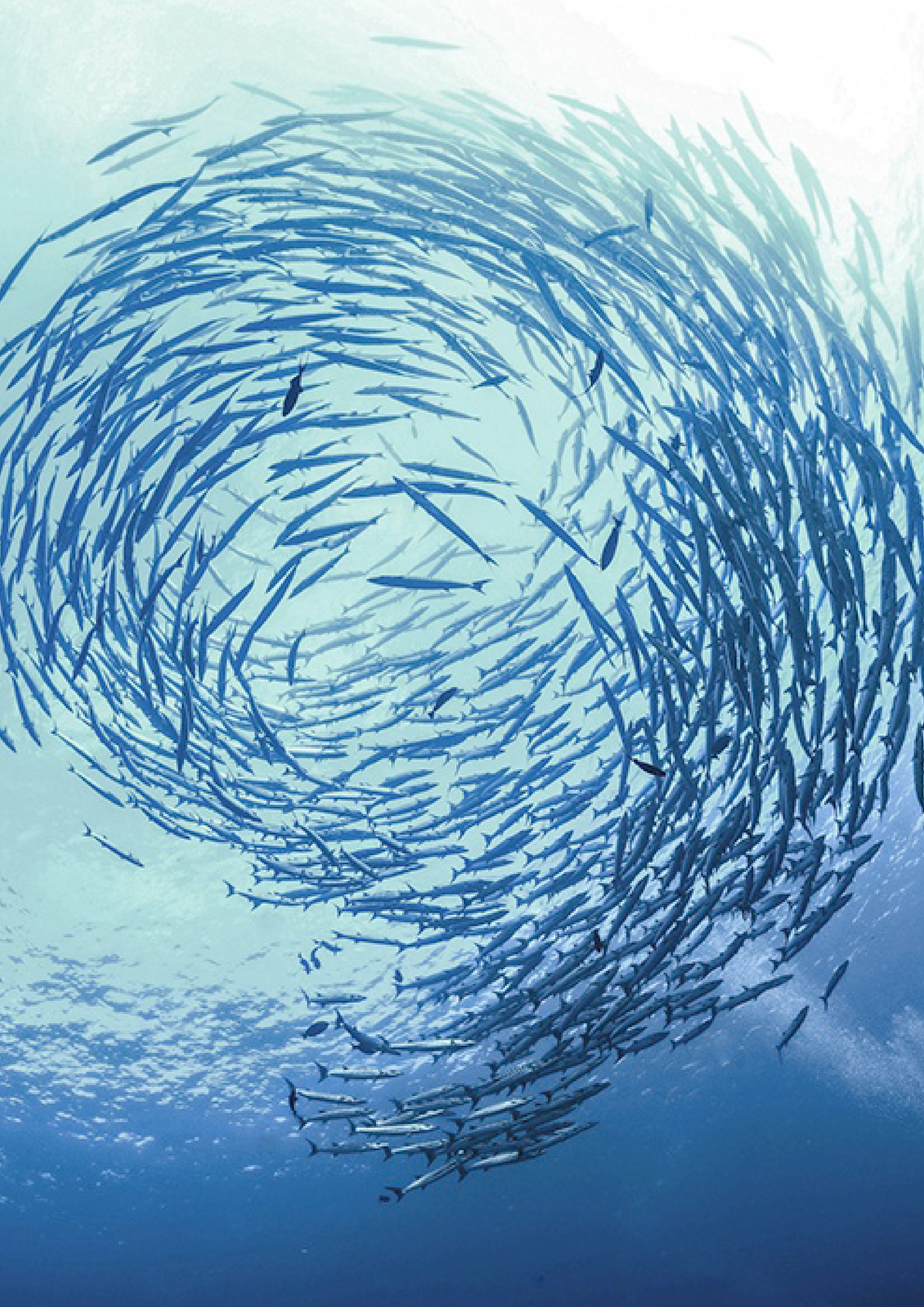
As preview of the abilities of the algorithm and the logic behind it, a case study design pavilion specified in chapter 5, with a predetermined panelling system is analysed and matches are generated out of a 3D model of a boat hull with the algorithm made in this research. Furthermore the application phase provides an overall building method into which the create algorithm fits. This provides some information for how this or a similar piece of software could be used in real architectural project.

The last part of this report is the evaluation where conclusions are being drawn from the overall research and what future research could be done. Also the bibliography and other relevant report information is found in the evaluation.

1.7 Research tools

Within this research certain tools have been used to generate the necessary 3D models. The main software used during this research is Rhino, a 3d modelling environment that includes Grasshopper, an algorithmic modelling plug-in(Robert McNeel & Associates, 2018). Quelea is a software program and plug-in for Grasshopper. This is software that offers the ability to simulate agent behaviour with direct visual feedback and is used within this research to generate insight into the functionality of agents and used to create the eventual algorithm and logic(Fischer, 2015).

Research



2. Agents

Design often poses a social problem when optimizing towards an objective. Do we quickly want to converge our data towards the best option possible? Or should there be a certain amount of differentiated preferences when trying to reach a certain objective?

The general logic of this discussion far precedes that of research into agent technology. A well known example of the questions posed above were also attempted to be answered by metaphor known as the Darwin Machine, named after Charles Darwin (1809-82) and another such machine, the Bernard Machine, based on the idea of French physiologist Claude Bernard (1813-78).
Architect's view of nature tends to rest on the assumption that it is inherently efficient in its use of materials and energy, and essentially ingenious and elegant in its solutions. This perception in science is one that has been propounded through Darwinism: the notion that refinement of 'design' is achieved through repetitive selection, variation and mutation. An argument is made by Scott Turner (Turner, 2012) that living design is based on the idea of homeostasis from Claude Bernard. Homeostasis is a tendency of living systems to regulate their environment as a stable state, bones being an example of this. When the strain on a bone is high it serves as an indication of the bone being to weak, thus it strengthens itself until strain is sufficiently reduced. If too little strain is sensed, it means that the bone is wastefully over mineralised, resulting in a reduction of bone structure until the sweet spot of strain is reached again. In short this is homeostasis, bones are well designed because they want to be, they know when their design falls short.

The same logic used to generate design optimisations, preventing optimisations from too quickly converging to a certain optimum might disregard other, relevant solutions. Agents provide another mean towards the goal of finding the most optimal solution. In relation to architecture and multiple conflicting objectives within optimisation, diverse teams composed of agents with different preferences maximize the number of optimal solutions, while uniform teams composed of multiple copies of the best agent are in general suboptimal(Marcolino et al., 2015).

2.1 Introduction

Optimising the data flow and relationship between objectives within design continues to be a field of great interest of research. Within the optimisation process and the search for solutions between conflicting objectives, agents are gaining interest.
Agent-based systems are part of a rule-based computational modelling method whose premise is that through the calculation of the local interactions between individual

entities (autonomous agents), which are based on pre-defined rule sets (behaviours), a global configuration emerges that meets higher-level goals. As such, agent-based modelling, ABM, is an example of behaviour-based artificial intelligence (Brooks, 1986). Furthermore it is used in several applications, ranging from simple email filters to large, complex, mission critical systems such as air-traffic control (Jennings, Sycara, & Wooldridge, 1998). In architecture agents are used for simulating pedestrian movement in buildings and urban environments (Aschwanden, Halatsch, & Schmitt, 2008), for design exploration with user feedback (Gerber et al., 2017) and panelling systems that include fabrication constraints (Baharlou & Menges, 2000; Krieg et al., 2015).

2.2 Agents

Agent-based design computation is in essence a behavioural system.
The knowledge distribution of an agent-system is based upon the viewing radius of the agents. This similar to the limited view humans and animals have. Thus knowledge distribution is done locally resulting in the avoidance of unnecessary computation.
Bottom-up knowledge distribution provides agent-based system with behaviour-based computation rather than knowledge-based computation. In behaviour-based computation, the topological space is explored with agents along with their specific behaviours to behave in this problem domain, rather than with a specific system that know about the problem domain (Maes, 1993).
Before we dive deeper into the definition and properties of agents, a definition should be described, since agents in software have been given a wide spectrum of definitions. One of the definitions used regularly, is the following:

> *'An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.'* (Wooldridge & Jennings, 1995)

Several classifications have been made for the use of agents, such as for social science. Furthermore the use of agents can be found in urban models, opinion dynamics, consumer behaviour, industrial networks, supply chain management and participative and companion modelling (Gilbert, 2008).

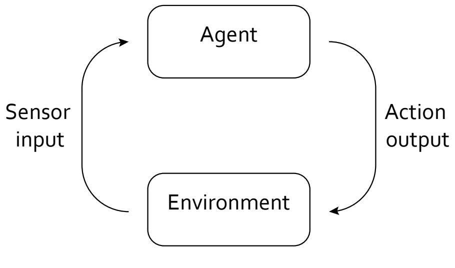


FIGURE 3 ON GOING AND NON-TERMINATING ACTION OF AN AGENT IN ITS ENVIRONMENT. (WOOLDRIDGE, 2002)

2.3 Agent intelligence

Agent intelligence can be defined by the following list(Wooldridge & Jennings, 1995):
Pro-activeness: Intelligent agents are able to exhibit goal-directed behaviour by taking initiative to satisfy their design objectives.
Reactivity: Smart agents can perceive and react to their environment in a timely fashion. They should be able to do so when changes occur.
Social ability: Capability to interact with other agents in order to satisfy their design objectives.
For non-functional systems, which do not terminate, goal directed programming is not acceptable since it makes important limiting assumptions. It assumes that the environment does not change while the procedure is executing.
Dynamic environments call for reactive agents in order to account for changes to the environment. Agents should attempt to achieve their goal systematically, but not blindly execute these procedures to achieve this goal when it is unclear if the procedure still works or if the goal is still valid. So a certain amount of reactivity is desired. Yet, continually reacting and never focussing on a goal long enough to achieve it is not desired either. Information exchange is not really social ability. Negotiating and cooperating is necessary to reason about and understand the goals of others, which is more complex then simple information exchange.
For many researches their idea of programming computer systems in terms of mental notions such as belief, desire and intention is a key component of agent-based systems(Wooldridge, 2002).

2.4 Multi-agent systems

A multi agent system is capable of independent action, on behalf of its user.
An agent can v out what to do in order to satisfy design objectives, instead of being told what to do. Multiple agents in a system interact with each other.
Interacting between agents consists of tasks such as cooperation, coordination and negotiation. These tasks are carried out in a similar way as humans do.
These tasks are a part of the agent's social ability. This social ability also embodies the ability of agents to communicate amongst themselves towards cooperation and negotiation.
An agent can act autonomously. An example for autonomous behaviour can be described as firstly recognizing its environment, figuring out what action to perform and execute the action. Then the results of the action are perceived and further action is selected to act accordingly.
Encounters among agents within a multi-agent system are economical encounters, meaning each agent in such an encounter has self-interest. This is a difference with

classic systems where all computing elements share a common goal to make the overall system function properly. Agents can have artificial intelligence (AI) abilities, such as learning, planning and understanding (Wooldridge, 2002). Thus they have the possible ability to read images, learn after each computation.

2.5 Environments

Both agents and multi-agent systems are situated in an environment. The characteristics of this environment influence the functionality of the agents within. Environments can have different characteristics that influence the computation that is being run by the agents and with it the time it takes to complete each computation.

Within our research the environment is the material we look through for matches to be made between the design and the material. Below is a short overview of these characteristics.

Accessible versus inaccessible

An environment is accessible when an agent can obtain complete, up-to-date, accurate information about the state of the environment.

Deterministic versus non-deterministic

When any action that has occurred has a single, guaranteed effect, the environment in which it is situated is called deterministic. This means there can be no uncertainty about the outcome of performing a single action. Unfortunately, if an environment is sufficiently complex, then the fact that it is actually deterministic is not much help(Russell & Norvig, 1995). To all intents and purposed, it may as well be non-deterministic. In practice almost all realistic environments must be regarded as non-deterministic from an agent's perspective (Wooldridge, 2002).

Agents have a limited 'sphere of influence' and thus they have only partial control over the environment at best. Actions performed by agents are typically done to bring about some desired state of affairs. Non-determinism captures the fact that actions can fail to have the desired outcome.

Static versus dynamic

A static environment is one that can be assumed to remain unchanged, except for the outcome of actions carried out by agents. A dynamic environment has other processes working on it, causing changes to the environment beyond the control of the agents. Examples of a dynamic environment are the physical world and the Internet.

Discrete versus continuous

An environment is discrete if there are a fixed (finite) amount of actions available for an agent to carry out. An example of a discrete environment would be a chessboard, having only a limited amount of options. A continuous environment with an infinite amount of actions could be exemplified by a taxi fare.

Overall, environments are often referred to being 'open' if they have the most complex combination of characteristics. These are inaccessible, non-deterministic, dynamic and continuous (Hewitt, 1986).

In this research we are focussing on accessible, non-deterministic, static, discrete environment. The information about the material environment is accurate and complete (accessible), it might not have the desired outcome (non-deterministic), it remains unchanged (static) and it has a fixed amount of options available (discrete).

2.5.1 Interaction agent & environment

The environment properties influence the complexity of agent design process and so does the interaction between the agent and its environment.

Whenever a system takes some input, performs some computation over the input and eventually produces some output, it is considered to be a functional system. Functional systems terminate based on preconditions and post conditions. Preconditions represent what is true about the environment so the program can terminate, or operate correctly. Post conditions state what true is about the environment after the program as terminated, assuming the preconditions were satisfied when the program executed.

The counterpart to a functional system is a reactive system. The role of reactive systems is to maintain interaction with their environment. Therefore they must be described and specified in terms of their on-going behaviour. It has the capability to respond to rapid changes in the environment. Reactive systems respond directly to the world, rather then reason explicitly about it.

Furthermore the concept of fairness in relation to agents has to be considered. Fairness (Francez, 1986) means that an entirely reasonable decision made locally can have undesirable effects in a global context. An example of this is an agent that controls a printer. The agent continually receives requests to have access to the printer, and is allowed to grant access to any agent that requests it, with the provision that it is only allowed to grant access to one agent at a time. At some time, the agent reasons that it will give control of the printer to process p1, rather than p2, but that it will grant p2 access at some later time point. This seems like a reasonable decision, when considered in isolation. But if the agent always reasons like this, it will never grant p2 access(Wooldridge, 2002).

In this research we are dealing with a functional system in which we can not properly deal with fairness. Since the outcome of the algorithm is the location of one design panel on the material, it does not make any relation towards other panels and their options. Thus taking a solution, a location on the material, for a panel that is a little below the optimal found solution, but still provides a solution within the given tolerance might enable more panels to be taken out of the material.

2.6 Swarm logic

The logic used within this agent application is a bottom-up approach of multiple entities, agents, which perform actions based on local stimuli. This is based upon a behavioural model known as swarm logic (Reynolds, 1987).

The aggregate motion of a flock of birds, a herd of land animals, or a school of fish is a beautiful and familiar part of the natural world. These are the phenomena that Reynolds describes and how they can be simulated in computer graphics. The simulated flock is an elaboration of a particle system, with the simulated birds being the particles. The aggregate motion of the simulated flock is created by a distributed behavioural model much like that at work in a natural flock; the birds choose their own course. Each simulated bird is implemented as an independent actor that navigates according to its local perception of the dynamic environment, the laws of simulated physics that rule its motion, and a set of behaviours programmed into it by the "animator." The aggregate motion of the simulated flock is the result of the dense interaction of the relatively simple behaviours of the individual simulated birds(Reynolds, 1987).

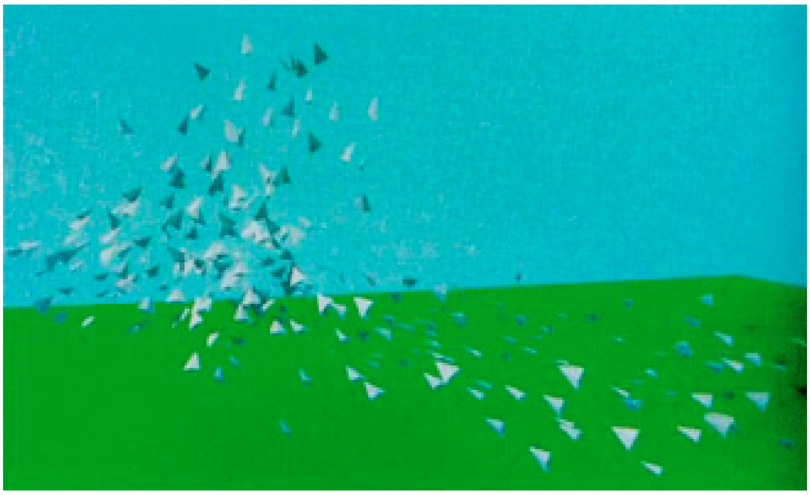


Figure 4 Flocking visual by Reynolds, 1987.

Swarm logic represents a model for the creation of systems that can generate geometry based upon local interactions, both between elements in a system and between these and their environment (Gerber et al., 2014).

This logic is also used for the distribution of agents in this research. Several weighting factors will be given to the rule set that each agent in the algorithm design have to follow.

2.6.1 Local and global optima

One of the characteristics of agent modelling is the fact that each agent has a viewing radius, similar to the particles that represent birds in Reynolds his research. With a viewing radius an agent has a limit to what it perceives and responds to. This generates a system where multiple agents find local optimal solutions, without taking into consideration the global level of their environment. The global level of the environment in which agents are situated means a full view of all other agents in the environment and, if applicable, other influences. The benefit of this is that the agents in the system do not require information about all other agents in the system, thereby reducing the computational power needed. A downside is that, without further additional input the agent will only concern itself with finding local solutions and have no regard as to what effect a decision locally has on the global level of the environment. This research attempts to solve this potential issue by generating an information layer that contains geometric characteristics of the environment, thereby leading the flocking agents towards the right position on the material.

Furthermore, translating local optima to global optima for a full structure can be achieved by coupling structural or climatic simulation output as drivers for the behaviour of the agents. This is not considered in this research but certainly is an aspect that could be used in future development of a similar logic as it has potential to increase the performance of the outcome generated by the agents (Gerber et al., 2017).

2.7 Conclusion

As previously stated in this chapter, agent-based systems are part of a rule-based computational modelling. It is a method whose premise is that through the calculation of the local interactions between individual entities (autonomous agents), which are based on pre-defined rule sets (behaviours), a global configuration emerges that meets higher-level goals. They are located in an environment that for a large part will be, in this research, a static and unchanged material surface, apart from the outcome that agents produce. The rule sets that are going to be used will have to be determined and will be largely based upon the swarm logic (Reynolds, 1987) and partly on geometric properties of the environment the agents are situated in. Currently the research only proposes a solution where a single agent group is responsible for reaching an objective of one panel. An increasing amount of agent groups, each responsible for it's own panel, could lead to a better system. As already shown in previous research, feedback such as from sources such as a structural simulation, could be used into the agent system which can further improve the agent behaviour and the consequential output the produce (Gerber, Pantazis, Marcolino, & Heydarian, 2015).

3. Surface geometry

When modelling a geometry, parametric or in any other way, there are certain aspects to keep in mind. A division can be made between two main aspects. The first being of importance to designers and architects, this is the aesthetic output of their design. How well does the created geometry match their design intent? Secondly there is an part to design that holds the mathematical descriptions of the properties that come with geometry. This part is the description of the geometric properties and what their relationship is to the actual form of the geometry.

When we consider surface, there is a classification to be made, since we are dealing with types that are ranging from flat, also known as planar, surfaces to doubly curved, saddle like surface.

Furthermore we are mainly concerned with two aspects of surfaces within this research. The first being surface curvature and the second is the continuity between surfaces. A starting point for the algorithm that is later developed in this research is the matching of curvature shapes and thus values. A second part, which is more an aesthetic analyses of the result that the algorithm generates, is concerned with the surface continuity.

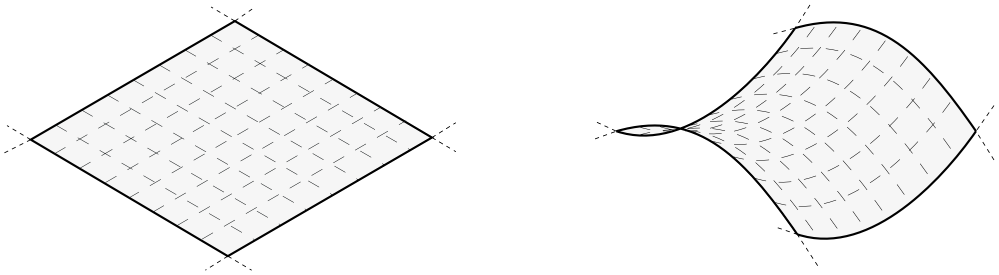


Figure 5 From flat, planar to saddle shaped, anticlastic surfaces, by author.

3.1 Classification

Before we dive into the geometric description of surface geometry, a distinction has to be made between several types of geometry. Below is a visualization of geometry types linked with Gaussian curvature.

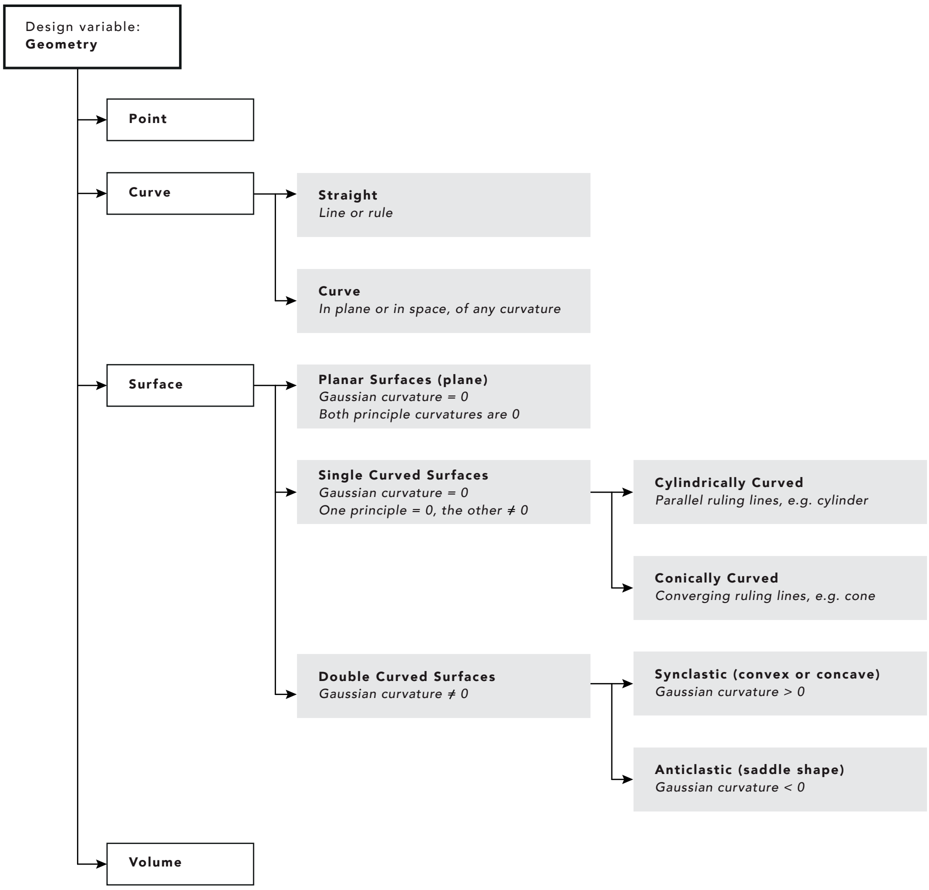


FIGURE 6 GEOMETRY TYPES. LECTURE A. BORGART, EDITED BY M. STOFFER.

3.2 Curvature

In order to explain the curvature types that appear on surfaces, firstly the curvature of a curve (2D) is explained before explaining it for situations in 3D.

3.2.1 *Curve curvature*

At any point on a curve in a plane, the line best approximating the curve that passes through this point is the tangent line. We can also find the best approximating circle that passes through this point and is tangent to the curve. The reciprocal of the radius of the circle is the curvature of the curve at this point.
The best approximating circle may lie either to the left of the curve, or to the right of the curve. If we care about this, then we establish a convention, such as giving the curvature a positive sign if the circle lies to the left and negative sign if the circle lies to the right of the curve. This is known as signed curvature.

3.2.2 *Surface curvature*

From a mathematical point of view, curvature is an amount of bending of a curve or surface at a point on this curve or surface. The value of curvature, κ, can have different values if the plane through the normal varies.

This generalization of curvature to surfaces is normal section curvature. Given a point on the surface and a direction lying in the tangent plane of the surface at that point, the normal section curvature is computed by intersecting the surface with the plane spanned by the point, the normal to the surface at that point and the direction. The normal section curvature is the signed curvature of this curve at the point of interest.

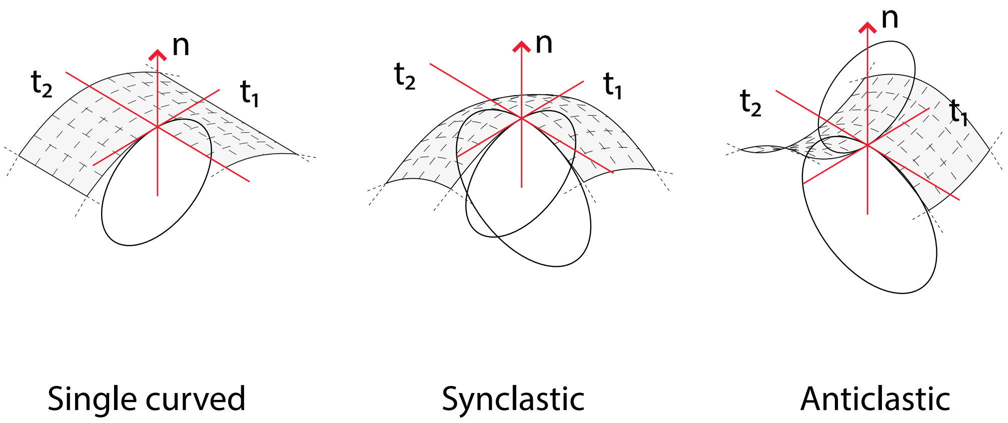


Figure 7 Curvature types. Lecture A. Borgart.

If we look at all directions in the tangent plane to the surface at our pint, and we compute the normal section curvature in all these directions, we get a range of values or normal curvature.

Between these variations a minimum (κ_2) and a maximum (κ_1) can be found in perpendicular direction, these minimum and maximum are the principal curvatures. The Gaussian curvature is the product of the principal curvatures. The tangent plane of any points with positive Gaussian curvature, known as synclastic surfaces, touches the surface at a single point. The tangent plane of any point with negative Gaussian curvature cuts the surface.

Another value to describe curvature at a point on a surface is mean curvature. This is one half of the sum of the principal curvatures of that point. Any point with zero mean curvature has negative or zero Gaussian curvature (“www.rhino3d.com,” 2018).

3.3 Surface continuity

When generating a design for a surface, such as a façade design, there is always a combination of surfaces caused by the division of the material that it consists of. Sometimes the goal of this surface combination is to create a continuous appearance between the parts. A curve or a surface can be described to have G^n continuity.

Positional continuity G^0: If curves or surfaces touch at the point of joining (position).
Tangential continuity G^1: The curves or surfaces share a common tangent direction at the point of joining (position and tangent).
Curvature continuity G^2: The curves or surfaces have end vectors of the same direction, length and curvature. These are considered to be visually smooth (position, tangent & curvature) (“www.rhino3d.com,” 2018).

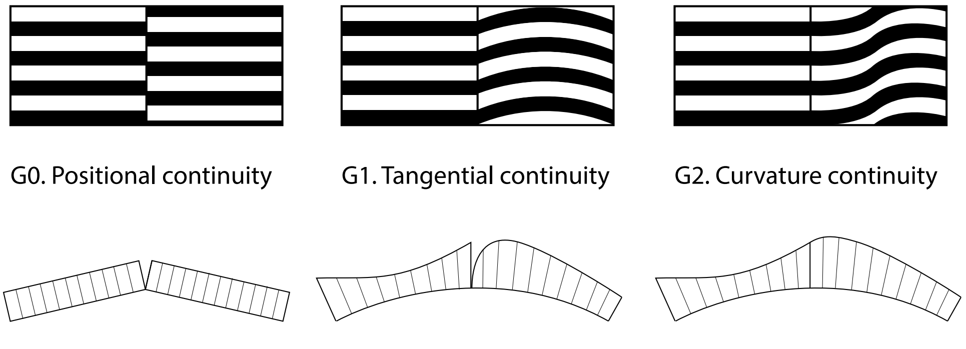


Figure 8 Surface continuity(“www.rhino3d.com,” 2018). Edited by author.

3.4 Conclusion

As stated in the research question, the goal is to find a solution for the matching of curved surfaces. The challenge presented by curved surfaces increases the complexity compared to planar surfaces, where there is little to no deviation in geometric properties and the exercise would be focussed on optimal material use on the material available, a process also known as nesting.

For curved surfaces this is also still a relevant consideration. There is potential for agent system to tackle such problems by either concerning a single agent for every panel that needs to be found or to let multiple agent groups search through the material at the same time, where each agent group is responsible for a different panel. A similar process has already been explored for panelling systems to help generate new designs(Baharlou & Menges, 2000). Yet the work-flow proposed in this research is also limited in an environment determined by the availability of the material, not by just the design intent.

Furthermore when applying such a process in reality, the expectations for the design outcome should be clearly stated. The goal is to find matches between design and material with as little deviation as possible, yet this will most likely not produce the curvature continuity that is achieved by designs such as the Nordpark Railway Stations by Zaha Hadid, seen in Figure 9. But rather have a similar outlook as seen in Figure 10, where a small mock-up shows what the result is on a smaller scale.

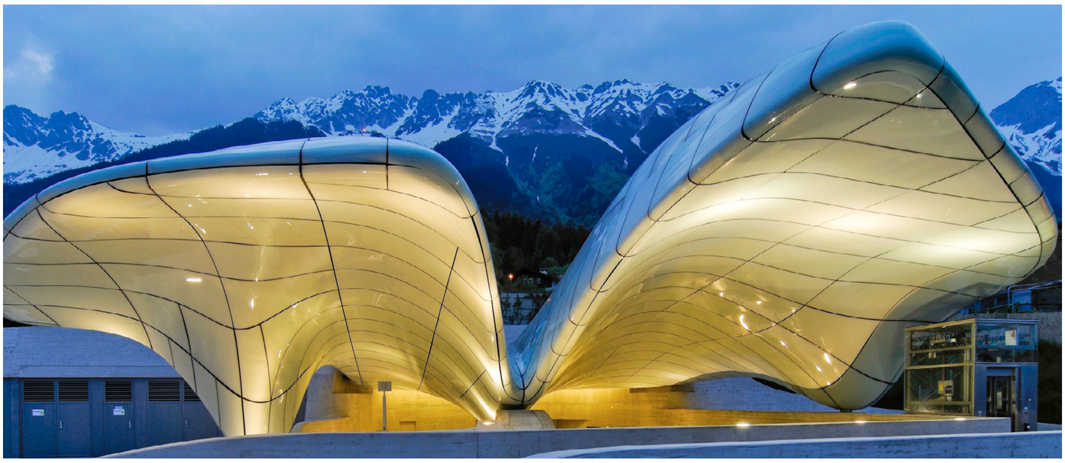


Figure 9 Nordpark Railway Station by Zaha Hadid Architects

Specification

4. Material

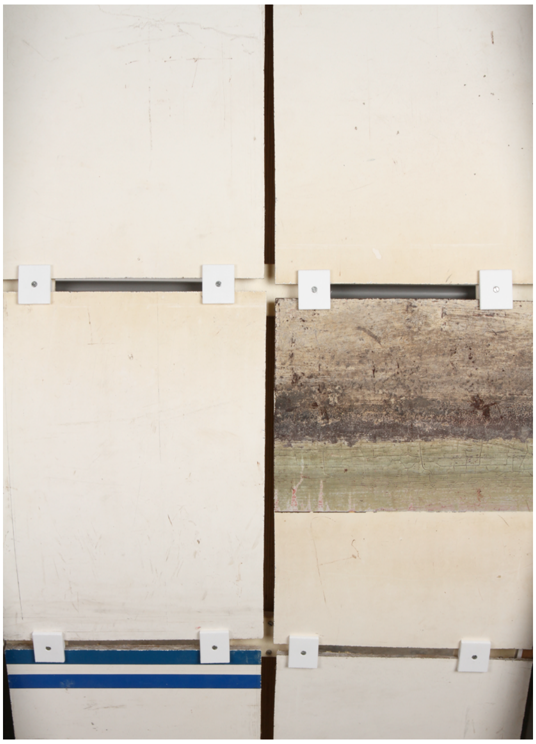


Figure 10 Boat hull panels in a mock-up of façade, Photo by M. Bilow, project by J. Sturkenboom, R. Wisse & author.

Since a period between 1970 and 1980 a significant amount of glass fibre reinforced polyester boats have been produced. As the generation using them is growing older, the amount of boats out of use is also growing. The amount of boats was estimated at 25.000 in 2014 for the Netherlands and this figure is expected to grow up to 35.000 in 2030. All of these boats, which are located in the Netherlands, are considered to be at the end of their life by 2030 (WA Yachting Consultants, 2015).

Apart from boat hulls made out of glass fibre reinforced plastic (GFRP), among other materials also rotor blades from windmills represent a significant part of the GFRP waste material flow within the Netherlands. The amount is quantified at 4.500 tonnes a year, out of which 1.400 tonnes are polyester boat hulls and 1.300 tonnes are rotor blades from windmills. Furthermore the amount of boat hulls is expected to grow up to 4.000 tones a year in 2030(Ten Busschen et al., 2016). This chapter gives a summary of what material glass fibre reinforced plastic is, along with some characteristics and how it can be processed.

4.1 Material

With decreasing value and usage of these boats, slowly but surely an environmental issue has occurred. Typical advantages of fibre-reinforced polymers *(FRP)* over other materials are that it does not rust, it is not affected by fungi and can have 4 times the strength of steel at equal weight (Knippers et al., 2011).

Glass fibre reinforced plastic is a material found under the composites category. Composites are solid materials created by combining tow or more different constituent material components, often with very different properties(Kolarevic, 2003a).

Combining the two main components, the reinforcement and the matrix together produces a composite material. Filler materials and other additives can be added to the mix. The matrix typically is a metallic, ceramic or polymer material into which multiple layers of reinforced fibres are placed. The fibres are made from glass, carbon, polyethylene or some other material. Sometimes lightweight fillers are added to increase the volume or thermal performance. Furthermore various chemical additives are used to create a certain colouring on the material or to improve the fire or thermal performance(Kolarevic, 2003a).

Glass fibre reinforced polyester material is a tough, hard material that is difficult to recycle because of the chemicals that are a part of the material. Furthermore it requires industrial machines to chip the material into smaller segments, thereby destroying the mechanical properties and shapes that can be found in the material.

4.2 Processing

Because of the energy intensive character of working with a material such GFRP, it is of importance to create an efficient way of processing it. This research attempts to do so be finding cutting patterns on existing material shapes that are similar to an architects design. Thereby using some of the mechanical strength and aesthetic qualities found in the material.

When cutting a material such as GFRP, a significant amount of dust is created during the cutting process. This dust consists of, among other materials, glass fibres. Dust generation in activities with cutting operations are considered extremely significant (Gangolells et al., 2009).

Because of the hazardous character the dust generates, it should carried out with caution. The cutting is preferably done by a machine within a closed off environment where the generated glass fibre dust is taken out of the air directly. This can be done by an air filtering system or with a lubricated cutting process, which prevents most of the dust particles to spread into the air. The advantage of using a lubricated cutting process, such as water-jet cutting, is that it might make the air filtration unnecessary.

Furthermore it is advised to take surface treatments into consideration. During production and maintenance coatings might have been applied onto the material or mixed into the resin used to create the material. The cutting process might leave these coatings damaged or in need of reapplication onto the material. Specifically the edges that are created by the cutting leave the resin and fibres open to the elements. In order to extend the usage of the materials and prevent de-lamination or deterioration of the material, edge treatment is an option.

A designer could be aiming for a clean look, which might result in a sanding of cutting edges and a subsequent treating of the edges with a new coating instead of simply cutting the panels and using them without further treatment.

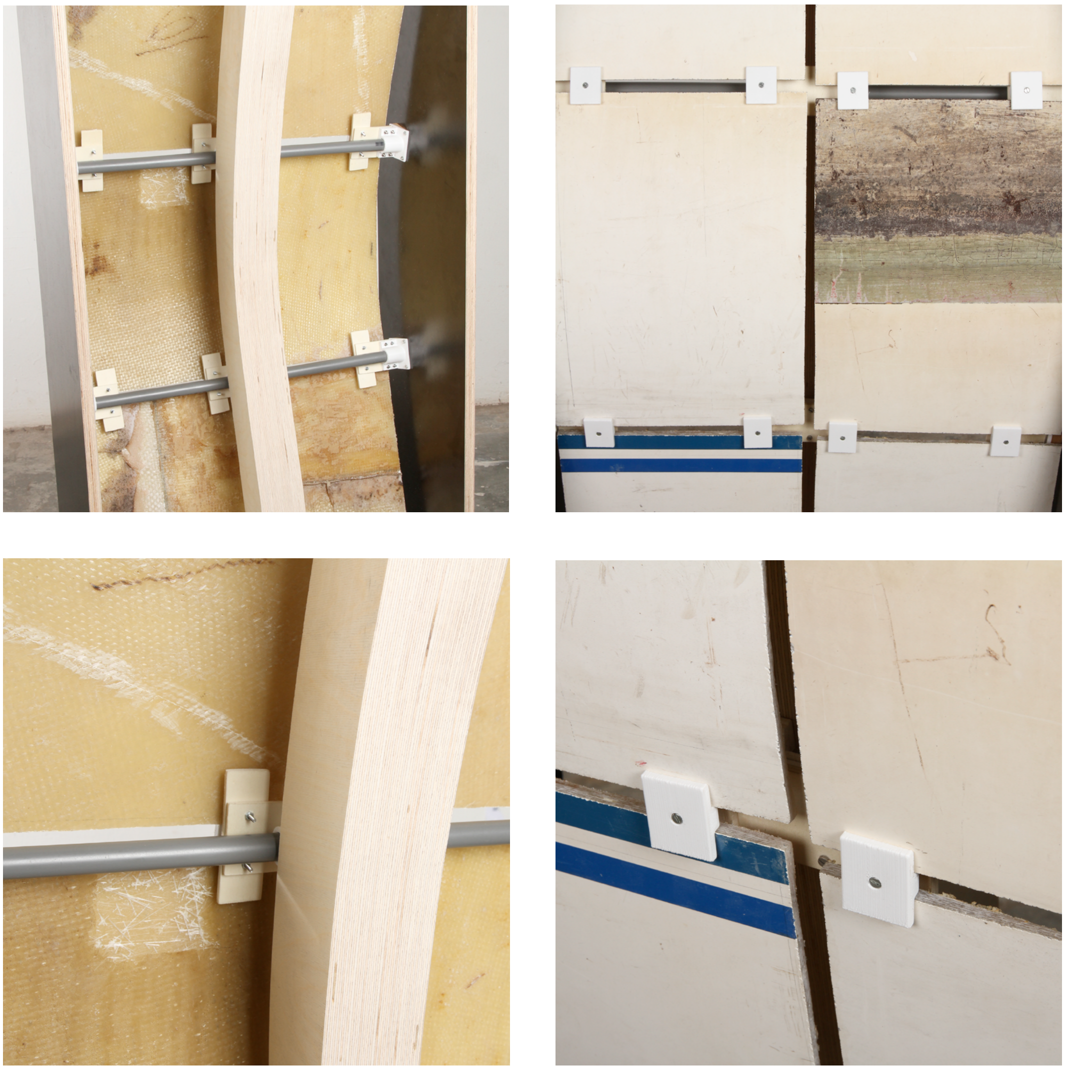


Figure 11 Pictures of a mock-up with GFRP panels extracted from a boat hull. Photo by M. Bilow, project by J. Sturkenboom, R. Wisse & author.

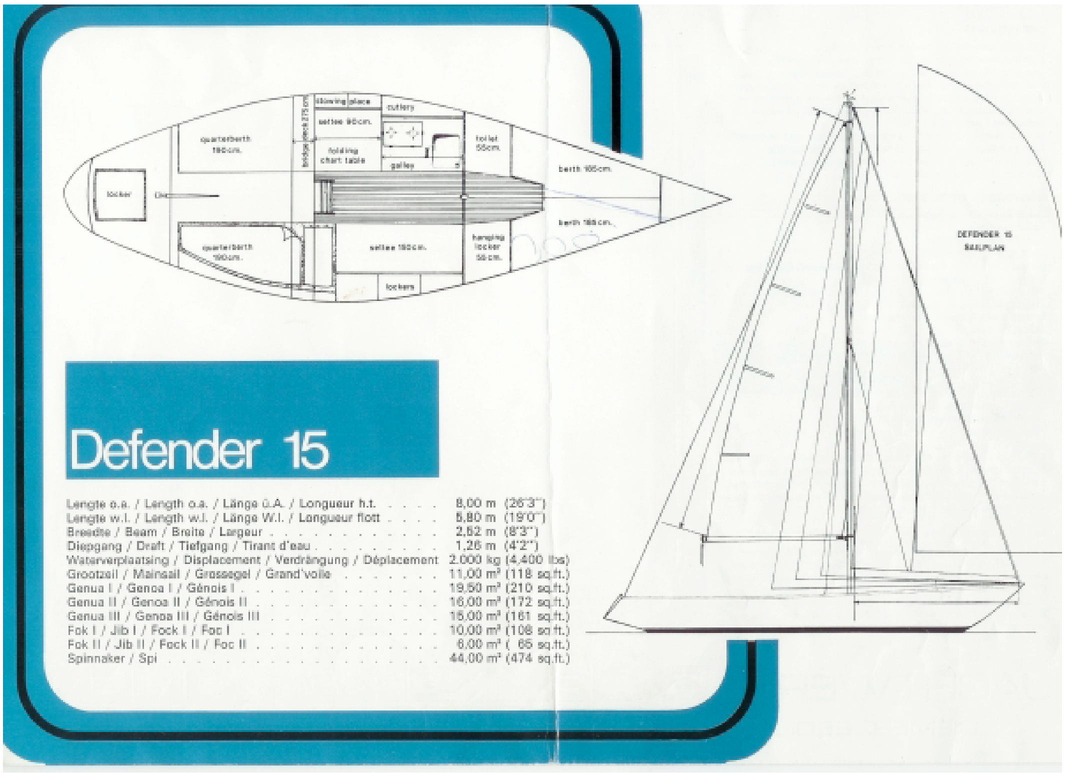
Other options for extracting the shapes out of the material can be abrasive cutting, laser cutting or milling. A whole range of settings and characteristics for both the cutting method and the material should be taken into consideration before picking a cutting method.

4.3 Maintenance

Most of the glass fibre-reinforced polymers are covered in a gel coating. This is a barrier that protects the material from degrading under influence of solar radiation. This coating typically degrades first. This process can take up to 20-30 years in nautical use, depending on a large set of variables. When this material is applied in architectural use, starting after the 10th year of application, a yearly monitoring of the GFRP's coating is advised in order to prolong the usage of the material. If done properly it could extend the lifetime of the material by significant amounts.

4.4 Boat types

The test that will be done in this thesis take only one boat type as input. This originates from a previous course, Bucky Lab, a part of the master courses of the track Building Technology at the Delft University of Technology. In that course a practical approach has been employed where J. Sturkenboom, R. Wisse & author visited a boat recycling yard that cut out the material that can be seen in the previous figures. The material for that mock-up and the service of cutting it was provided by Marine Recycling Goossens (http://www.marinerecycling.nl). The material originates from the 'Defender 15' boat type. This was one of the boats that was a part of the environmental problem that is



Defender 15

Lengte o.a. / Length o.a. / Länge ü.A. / Longueur h.t.	8,00 m (26'3")
Lengte w.l. / Length w.l. / Länge W.l. / Longueur flott	5,80 m (19'0")
Breedte / Beam / Breite / Largeur	2,52 m (8'3")
Diepgang / Draft / Tiefgang / Tirant d'eau	1,26 m (4'2")
Waterverplaatsing / Displacement / Verdrängung / Déplacement	2.000 kg (4,400 lbs)
Grootzeil / Mainsail / Grossegel / Grand'voile	11,00 m² (118 sq.ft.)
Genua I / Genoa I / Génois I	19,50 m² (210 sq.ft.)
Genua II / Genoa II / Génois II	16,00 m² (172 sq.ft.)
Genua III / Genoa III / Génois III	15,00 m² (161 sq.ft.)
Fok I / Jib I / Fock I / Foc I	10,00 m² (108 sq.ft.)
Fok II / Jib II / Fock II / Foc II	6,00 m² (65 sq.ft.)
Spinnaker / Spi	44,00 m² (474 sq.ft.)

Figure 12 The Defender, image from Jachtwerf F. Dekker en Zonen

caused by glass fibre-reinforced plastic. In the figures below the type can be seen on an old poster and the accompanying 3D model that is generated by the author, based upon the poster. The 3D model of the Defender 15 boat type is later on used as material environment which the agents search through for the best matching part of the surface compared to the design input.

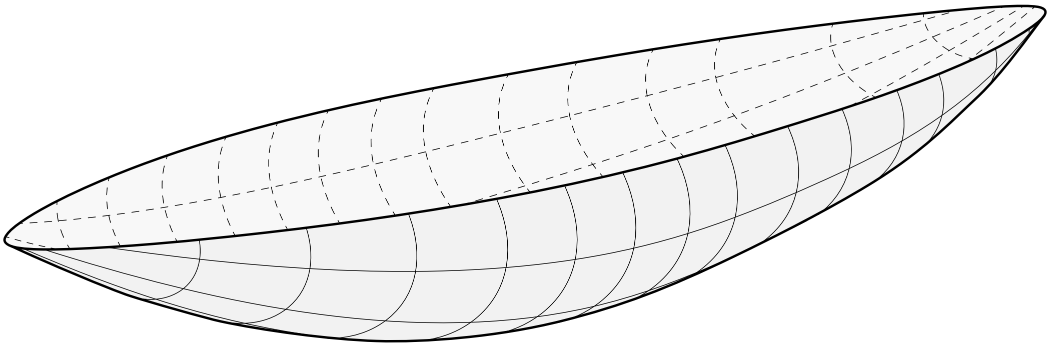


FIGURE 13 THE 3D MODEL OF THE DEFENDER 15, BY AUTHOR.

4.5 Conclusion

We know there is an increasing issue with the recycling of GFRP materials within the Netherlands. This issue is caused by the difficulties that the material composition poses for a recycling proses, such as including chemicals and the high material strength(Knippers et al., 2011). Furthermore the recycling is, in many cases, breaking the material down. This process destroys properties found in the materials current forms, such as the mechanical strength and the geometric complex shapes.

This research uses this material in order to offer an option for reusing the material without completely breaking it down, but rather cutting it into panels so shape and strength properties can still be used, in this case in architectural application.

The solution proposed later on within this research is not to be considered a final solution for the environmental impact this material is causing, but rather it gives another possible direction which could be explored further in order to tackle such a complex problem.

5. Case study design

In order to test the gained knowledge from the literature review, a case study design is picked along with the material mentioned in chapter 4. The design is a pavilion that has been created during the course Bucky Lab that is a part of the master curriculum for the track Building Technology.

This pavilion serves as the basis for a set of test that are run to see if the application of agents in this case holds any value in the future.

This design itself comes with a panelling distribution in place. This distribution of panelling and also the size of the panels are changed compared to the initial design so that the panel sizes have more realistic values (Figure 14). The pavilion design created during the Bucky Lab course in February 2015, by Jelle Sturkenboom, Rolien Wisse and Lieuwe Thys Meekma.

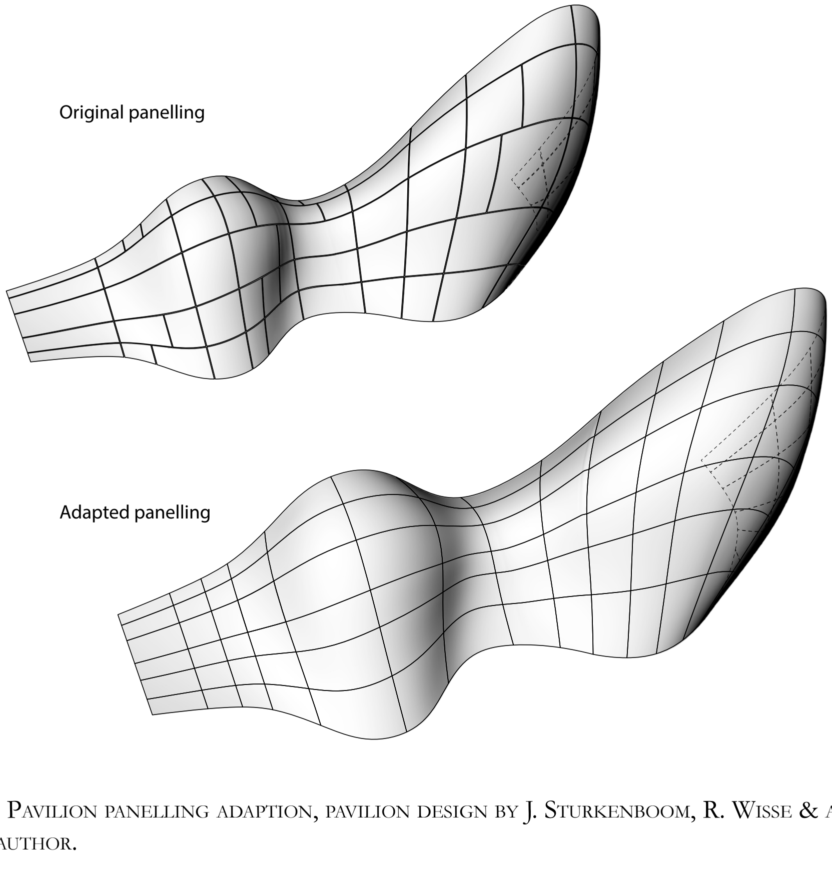


FIGURE 14 PAVILION PANELLING ADAPTION, PAVILION DESIGN BY J. STURKENBOOM, R. WISSE & AUTHOR. EDITED BY AUTHOR.

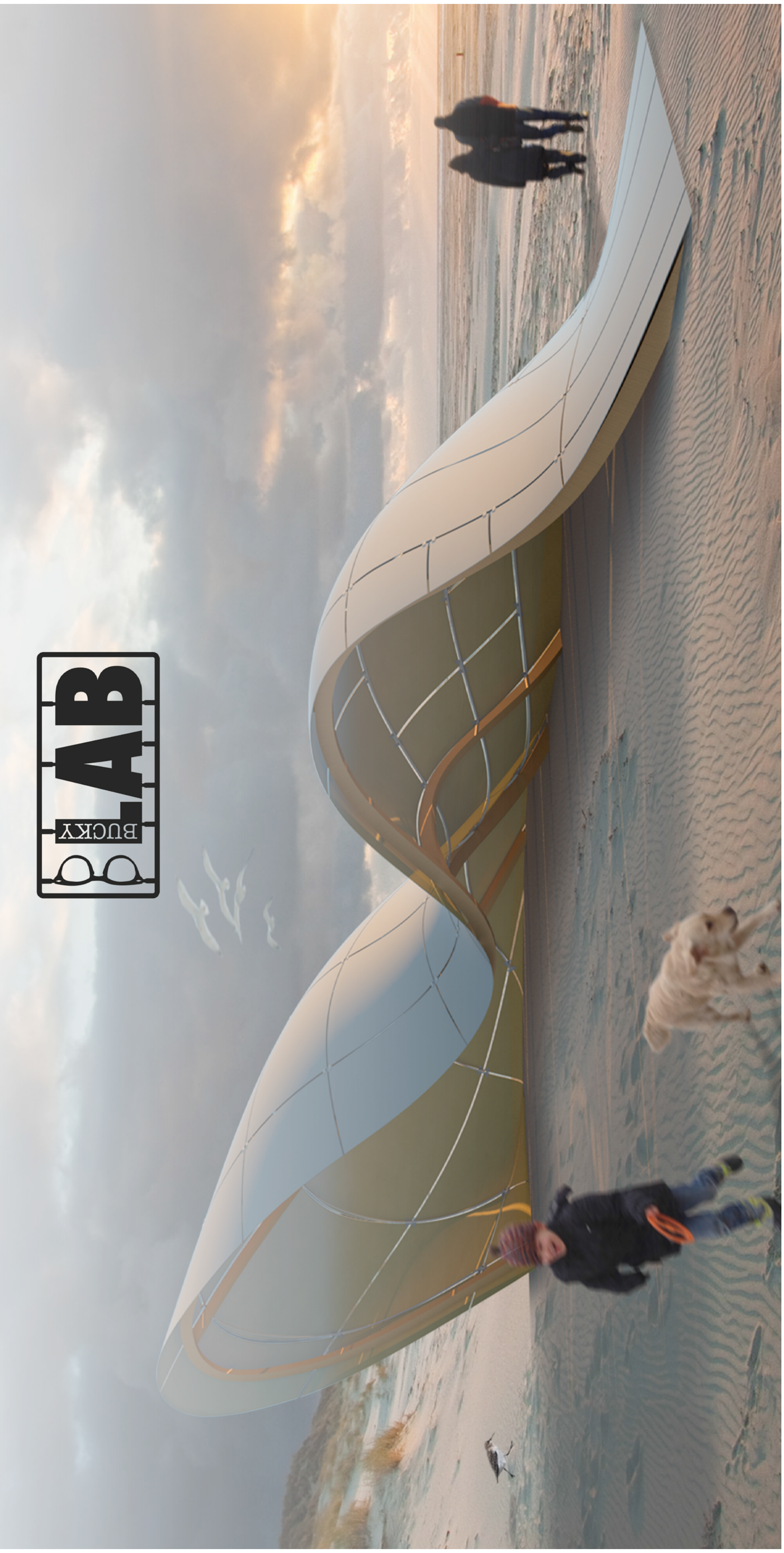
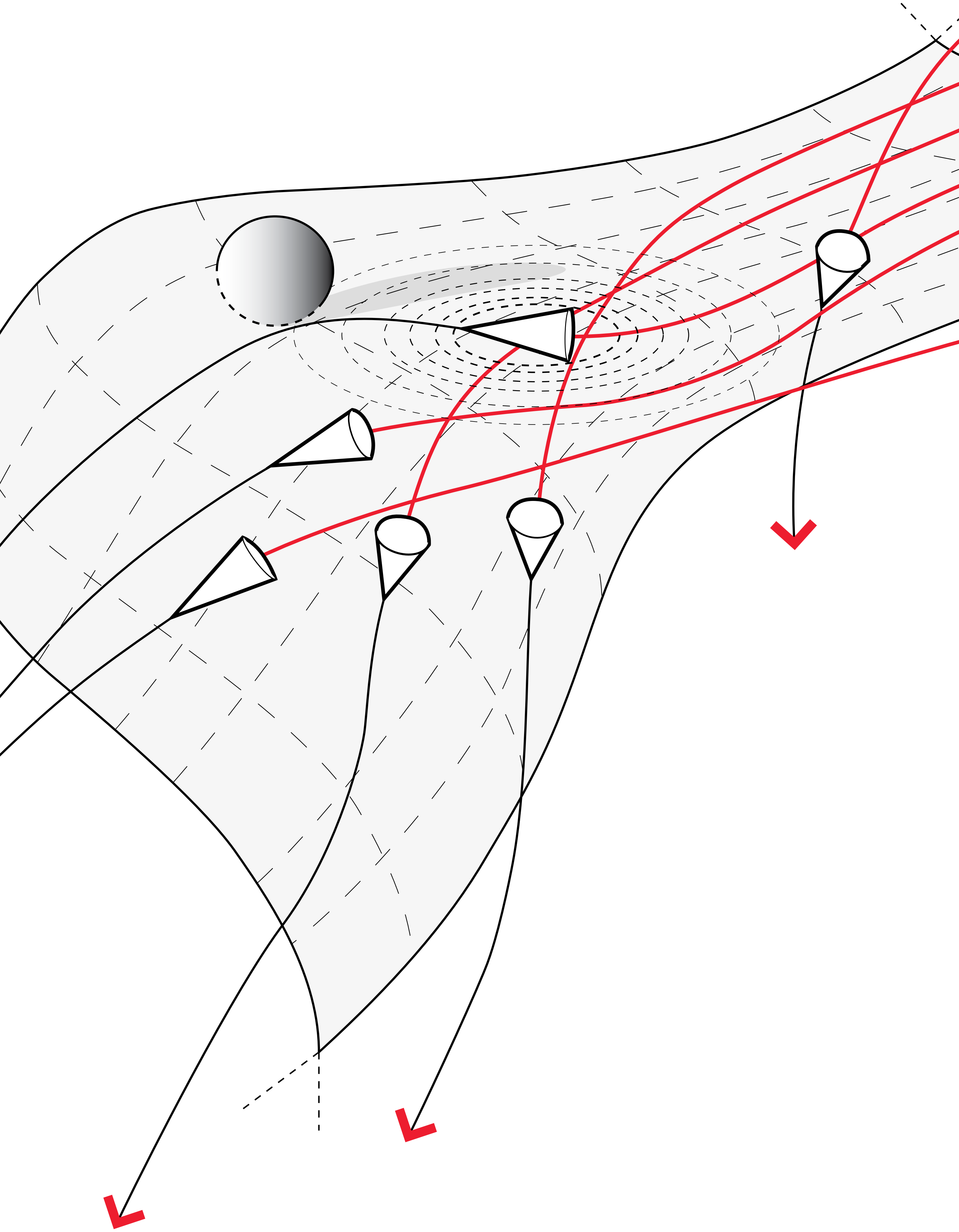


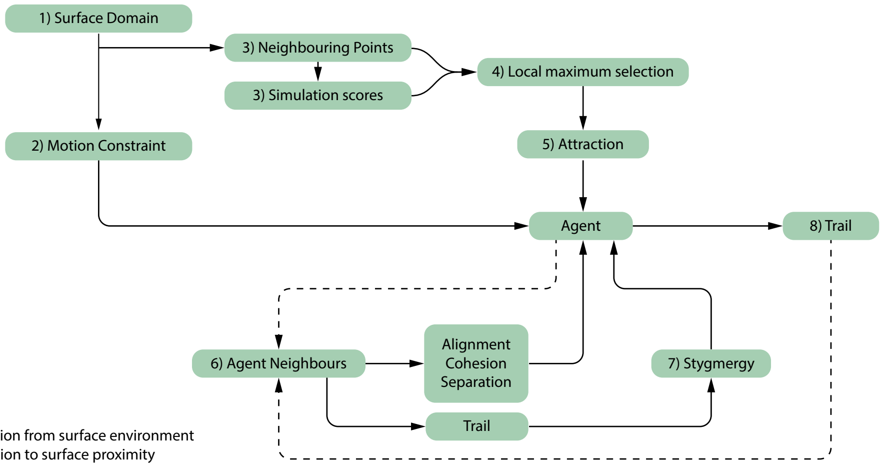
FIGURE 15 PAVILION DESIGN, BY JELLE STURKENBOOM, ROLIEN WISSE & AUTHOR

Application



6. Algorithm basic setup

This chapter will present the basics of the behavioural rule set that is applied within this research. The weight that each rule carries in comparison to the others influences the overall behaviour of the agents in the system. This weight ranges from zero (0) to one (1). If all the rules in system carry a weighting factor of one, all rules will equally influence the behaviour of the agents. If one rule is given a weighting factor of 0.5, this rule will have half the influence on the behaviour compared to the other rules. The weighting factors are picked in such a way that appropriate emergent behaviour is occurring. Firstly an explanation is given about the overall behaviour that is occurring in all of the data presented. The second part of this chapter showcases an overview of the rule set used for the optimisation. Further details can be seen in Appendix A –



1) Read information from surface environment
2) Constrain motion to surface proximity
3) Evaluate neighbouring surface points,
   Access their simulation data
4) Select local maximum, set as target.
5) Move towards target.
6) Evaluate neighbouring agents.
   Follow alignment, cohesion and separation rules.
7) Evaluate neighbouring trails. Follow trails. (stygmergy)
8) Generate trail
9) Repeat

Figure 16 Scheme of agent logic, by D. Gerber, edited by author.

Agent behavioural rule set. Finally a short summary and conclusion about agent rule set is given.

6.1 Introduction

In order for the agents to properly respond to the curvature values on the surfaces the agents need to read and react to those values. Moreover, they will interact with each other and generate and motivate each other's behaviour. The tool used in this research is a plug-in called Quelea (Fischer, 2015) that generates the agent behaviour. The results shown are a result of the usage of that plug-in within the programming environment grasshopper which is a part of the 3D modelling software Rhinoceros, also known as Rhino3D (Robert McNeel & Associates, 2018).

The way in which the Gaussian curvature data is presented to the agents can be seen in Figure 18, part 1. The material surface is sub divided into smaller parts, the amount of which can be determined before running the algorithm. The values shown are the values measured at the centre point of each subdivision and subsequently applied to the area of that subdivision. Since Gaussian curvature measurement carries information about a specific point, to gain greater feedback and more precision, the subdivision count of the material surface can be increased.

Furthermore the agents have a viewing radius, as can be seen in part 2 and 3 of Figure 18. Hereby it is important to consider that an agent always has a minimum of two data points that represent a Gaussian curvature value in it's viewing radius. Those are the centre points of a subdivision. If a smaller viewing radius is picked, the agent can be stuck in a local point of the material surface, as will be later explained in 6.2.3, *Mapping Geometry Data*. Based on the neighbouring agents that each agent can see within its viewing radius, it follows rules for alignment, cohesion and separation, as can be seen in Figure 18, parts 4,5 and 6 respectively.

Along with these rules the previously mentioned, Gaussian curvature values are used as

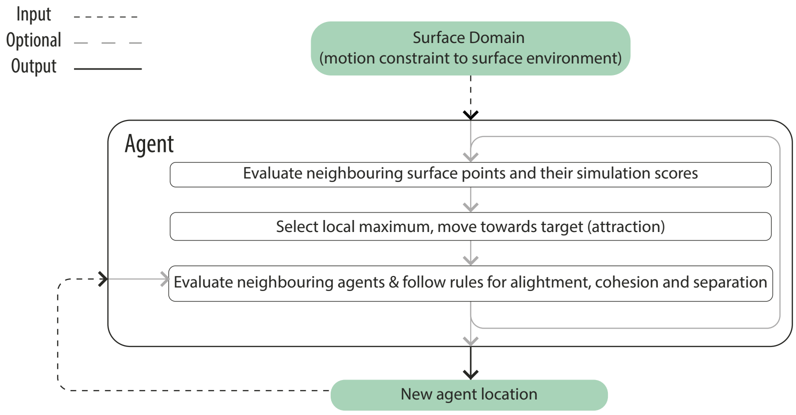
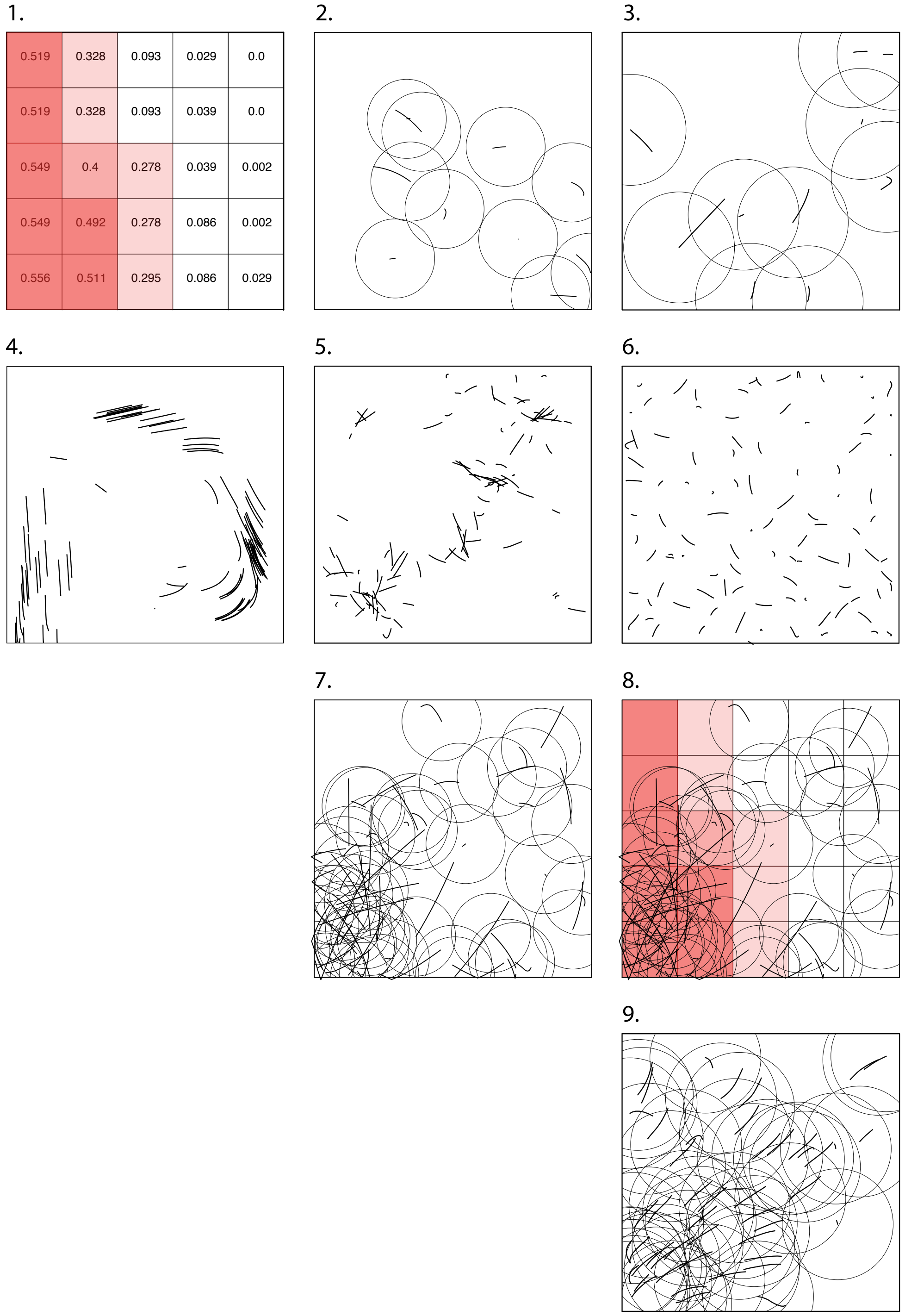

Input
Optional
Output
Surface Domain
(motion constraint to surface environment)
Agent
Evaluate neighbouring surface points and their simulation scores
Select local maximum, move towards target (attraction)
Evaluate neighbouring agents & follow rules for alightment, cohesion and separation
New agent location

Figure 17 Applied agent logic, by author.

values to which the agents are attracted, shown in part 7 and in part 8 with the colour overlay of those values. Finally we see the agents in part 9, where they follow the rules for alignment, separation, cohesion and attraction.



0.519	0.328	0.093	0.029	0.0
0.519	0.328	0.093	0.039	0.0
0.549	0.4	0.278	0.039	0.002
0.549	0.492	0.278	0.086	0.002
0.556	0.511	0.295	0.086	0.029

Figure 18 Agent rule set examples, by author.

6.2 Rule set overview

On the following pages an overview is given of the rules that the agent follows. All of them are separately displayed, in Figure 19 and Figure 20.
Each rule set shows four images, each of them representing an amount of time steps passed and thus iterations done. Those are 10, 50, 100 and 150.
Further details can be seen in Appendix A – Agent behavioural rule set.

	Rule	Radius:	Edge constraints:
1	Alignment	5	No
2	Alignment	5	Bounce Contain
3	Alignment	5	Contain
4	Alignment	5	Bounce Contain & Contain
5	Cohesion	5	Bounce Contain & Contain
6	Separation	5	Bounce Contain & Contain
7	Attraction	5	Bounce Contain & Contain
8	Attraction	7	Bounce Contain & Contain
9	Alignment, separation & cohesion	5	Bounce Contain & Contain
10	Alignment, separation, cohesion & attraction	5	Bounce Contain & Contain
11	Alignment, separation, cohesion & attraction	7	Bounce Contain & Contain

TABLE 1 OVERVIEW AGENT BEHAVIOUR TESTS, BY AUTHOR.

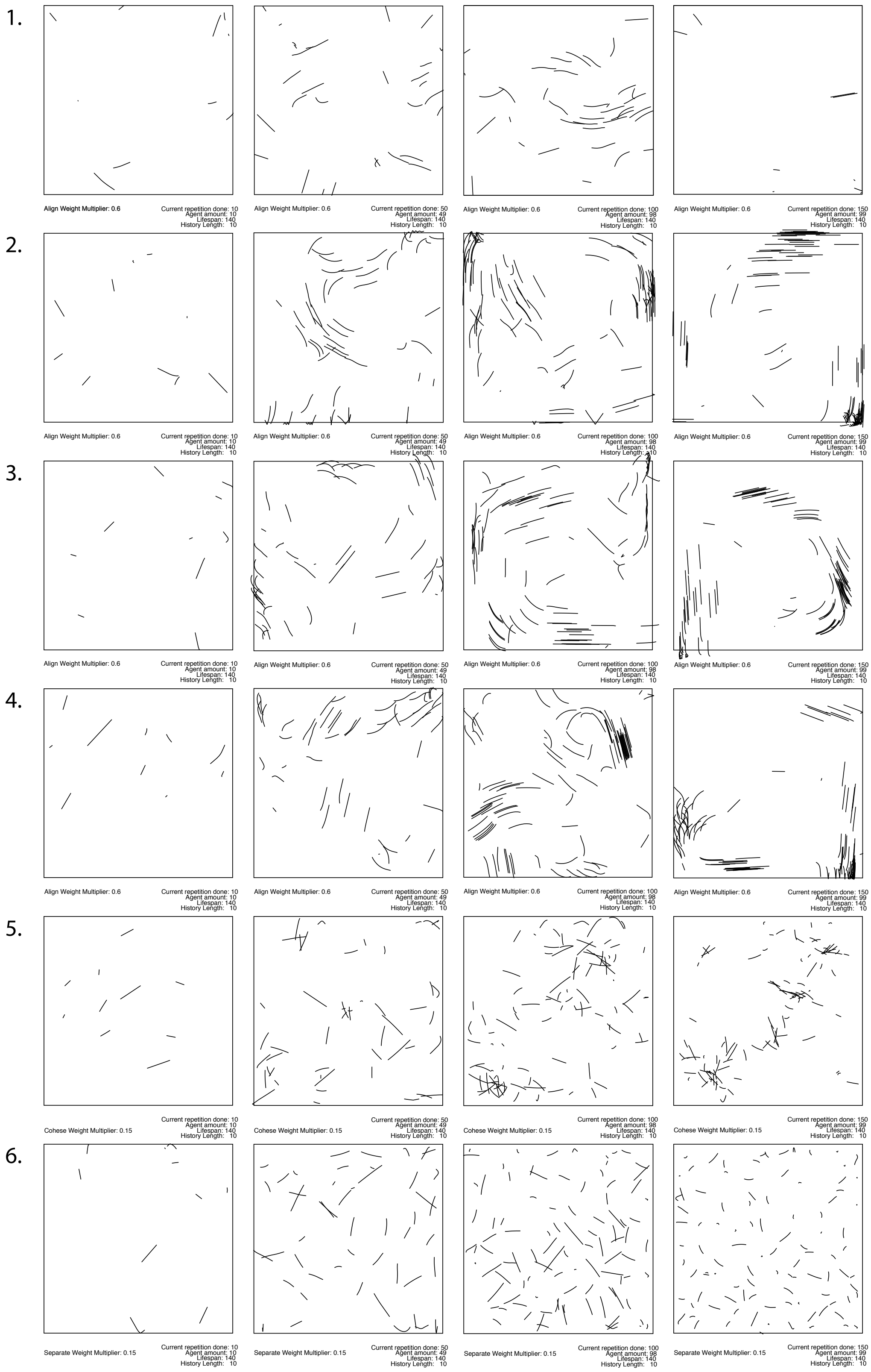

1.
Align Weight Multiplier: 0.6
Current repetition done: 10
Agent amount: 10
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Current repetition done: 50
Agent amount: 49
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Current repetition done: 100
Agent amount: 98
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Current repetition done: 150
Agent amount: 99
Lifespan: 140
History Length: 10
2.
Align Weight Multiplier: 0.6
Current repetition done: 10
Agent amount: 10
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Current repetition done: 50
Agent amount: 49
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Current repetition done: 100
Agent amount: 98
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Current repetition done: 150
Agent amount: 99
Lifespan: 140
History Length: 10
3.
Align Weight Multiplier: 0.6
Current repetition done: 10
Agent amount: 10
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Current repetition done: 50
Agent amount: 49
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Current repetition done: 100
Agent amount: 98
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Current repetition done: 150
Agent amount: 99
Lifespan: 140
History Length: 10
4.
Align Weight Multiplier: 0.6
Current repetition done: 10
Agent amount: 10
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Current repetition done: 50
Agent amount: 49
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Current repetition done: 100
Agent amount: 98
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Current repetition done: 150
Agent amount: 99
Lifespan: 140
History Length: 10
5.
Current repetition done: 10
Agent amount: 10
Lifespan: 140
History Length: 10
Cohese Weight Multiplier: 0.15
Current repetition done: 50
Agent amount: 49
Lifespan: 140
History Length: 10
Cohese Weight Multiplier: 0.15
Current repetition done: 100
Agent amount: 98
Lifespan: 140
History Length: 10
Cohese Weight Multiplier: 0.15
Current repetition done: 150
Agent amount: 99
Lifespan: 140
History Length: 10
Cohese Weight Multiplier: 0.15
6.
Current repetition done: 10
Agent amount: 10
Lifespan: 140
History Length: 10
Separate Weight Multiplier: 0.15
Current repetition done: 50
Agent amount: 49
Lifespan: 140
History Length: 10
Separate Weight Multiplier: 0.15
Current repetition done: 100
Agent amount: 98
Lifespan: 140
History Length: 10
Separate Weight Multiplier: 0.15
Current repetition done: 150
Agent amount: 99
Lifespan: 140
History Length: 10
Separate Weight Multiplier: 0.15

Figure 19 Agent rule set, part 1, by author.

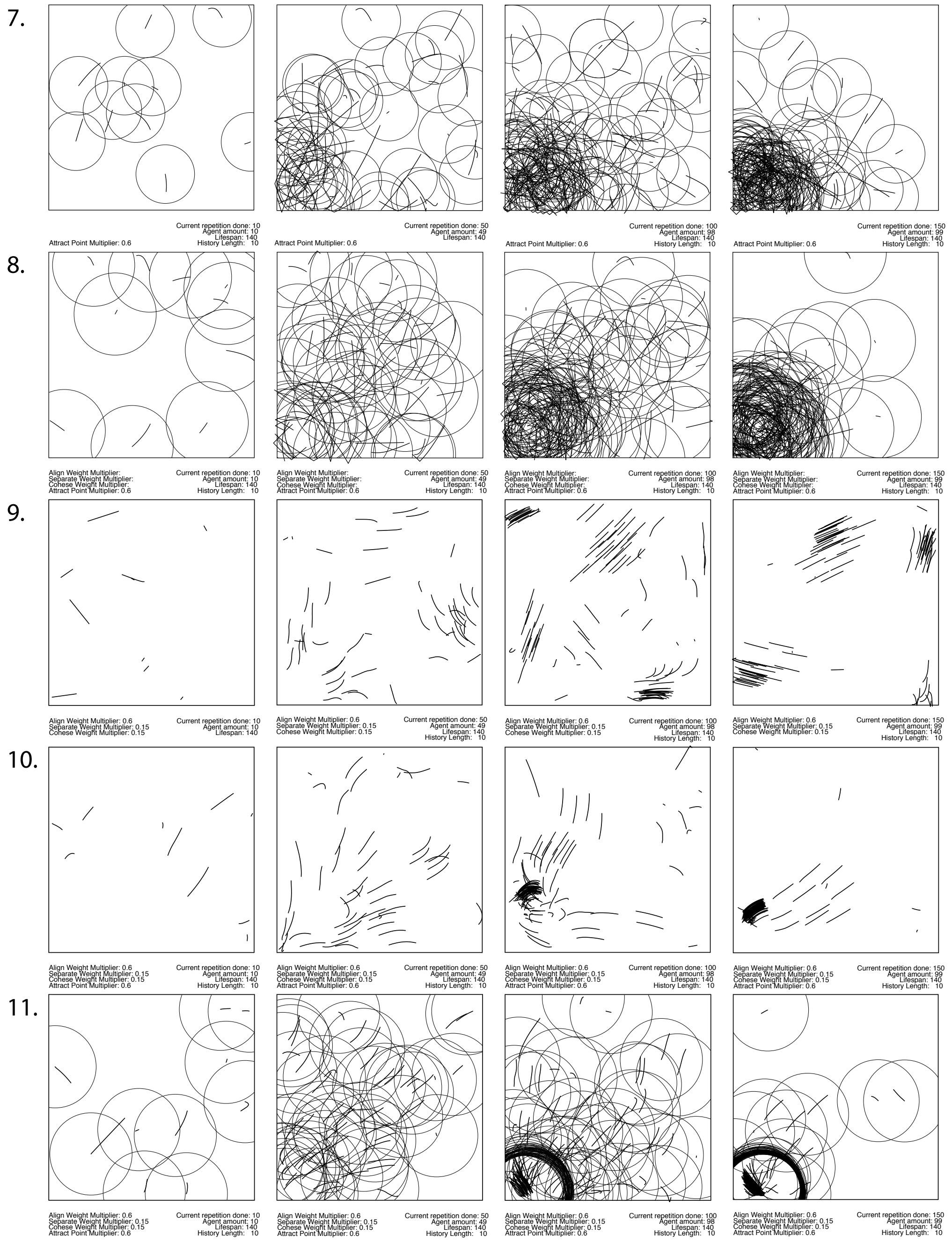

7.
Current repetition done: 10
Agent amount: 10
Lifespan: 140
Attract Point Multiplier: 0.6
History Length: 10
Current repetition done: 50
Agent amount: 49
Lifespan: 140
Attract Point Multiplier: 0.6
Current repetition done: 100
Agent amount: 98
Lifespan: 140
Attract Point Multiplier: 0.6
History Length: 10
Current repetition done: 150
Agent amount: 99
Lifespan: 140
Attract Point Multiplier: 0.6
History Length: 10
8.
Align Weight Multiplier:
Separate Weight Multiplier:
Cohese Weight Multiplier:
Attract Point Multiplier: 0.6
Current repetition done: 10
Agent amount: 10
Lifespan: 140
History Length: 10
Align Weight Multiplier:
Separate Weight Multiplier:
Cohese Weight Multiplier:
Attract Point Multiplier: 0.6
Current repetition done: 50
Agent amount: 49
Lifespan: 140
History Length: 10
Align Weight Multiplier:
Separate Weight Multiplier:
Cohese Weight Multiplier:
Attract Point Multiplier: 0.6
Current repetition done: 100
Agent amount: 98
Lifespan: 140
History Length: 10
Align Weight Multiplier:
Separate Weight Multiplier:
Cohese Weight Multiplier:
Attract Point Multiplier: 0.6
Current repetition done: 150
Agent amount: 99
Lifespan: 140
History Length: 10
9.
Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Current repetition done: 10
Agent amount: 10
Lifespan: 140
Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Current repetition done: 50
Agent amount: 49
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Current repetition done: 100
Agent amount: 98
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Current repetition done: 150
Agent amount: 99
Lifespan: 140
History Length: 10
10.
Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Attract Point Multiplier: 0.6
Current repetition done: 10
Agent amount: 10
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Attract Point Multiplier: 0.6
Current repetition done: 50
Agent amount: 49
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Attract Point Multiplier: 0.6
Current repetition done: 100
Agent amount: 98
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Attract Point Multiplier: 0.6
Current repetition done: 150
Agent amount: 99
Lifespan: 140
History Length: 10
11.
Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Attract Point Multiplier: 0.6
Current repetition done: 10
Agent amount: 10
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Attract Point Multiplier: 0.6
Current repetition done: 50
Agent amount: 49
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Attract Point Multiplier: 0.6
Current repetition done: 100
Agent amount: 98
Lifespan: 140
History Length: 10
Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Attract Point Multiplier: 0.6
Current repetition done: 150
Agent amount: 99
Lifespan: 140
History Length: 10

Figure 20 Agent rule set, part 2, by author.

6.3 Conclusion

The rules applied for each and every agent carries a weighting factor. This is an important feature that needs to be considered carefully whenever adapting the values given to it. In the examples and the application of the research, the following values are given to each of the four rules, alignment 0.6, cohesion 0.15, separation 0.15 and attraction 0.6 (to point with Gaussian curvature values).
The most important is the relative difference between these values, which should be kept the same if we would like to recreate similar behaviour. Yet, as agents are based on behavioural rules it is important to keep in mind that time we run the algorithm for a certain amount of iterations, the outcomes will be different. We consider this to be a positive effect that the agents have. Instead of generating an optimal solution, we are able to generate several outcomes that aren't necessarily the 'most' optimal, yet we know they are performing according to the rules that we have set.

Furthermore, we have to consider the way in which the agent's read and react to the Gaussian curvature values, which in this case are applied to points on the surface. This is caused by the choice of software, based upon familiarity with the software and the geometric feedback that is available with this software.
Another option within the software would be to apply an image that with a gradient. In that case the gradient would represent the transition from less-desirable to desirable curvature. If enough data points are queried upon which the gradient can be based it is another good solution. However, this option is not used within this research due to time constraints.

There are other, possibly more elegant solutions available. Within this research a specific plug-in (Quelea) within the Grasshopper programming environment is used.
More specific software packages are available to simulate agent behaviour, which might be more suitable and have to possibility to be efficiently and more 'tailor made' towards to problem, yet still includes geometric functionalities. One of these is the software program, called 'Processing' (https://processing.org/). Further software packages that are often used for multi-agent systems, are packages such as *Swarm, Repast, Mason, StarLogo and NetLogo* (Chen, 2012).

Algorithm basic setup

7. Algorithm Design and work-flow

As previously explained we are now aware of the impact that different settings of the agent behaviour have on the locations the agents travel to in their environment, which in our case, is the material surface. Within this chapter the logic behind the algorithm is explained through a step-by-step approach.

7.1 Parameters

The work-flow is created with these main parameters in mind:

1. **Panel size**
2. **Panel shape**
3. **Panel curvature deviation (between source and goal)**

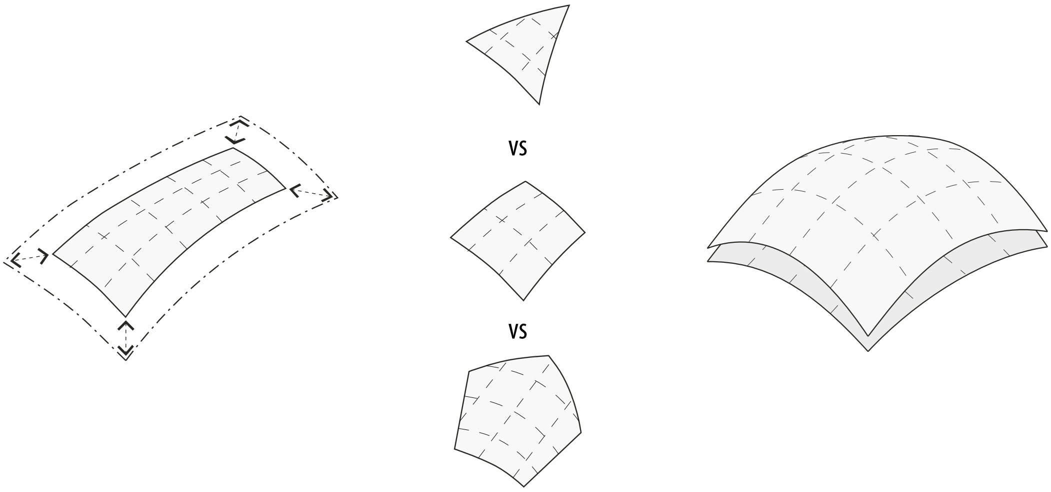


FIGURE 21 WORK-FLOW PARAMETERS, BY AUTHOR.

With this set of parameters a wide range of panelling systems can be accommodated. The *panel size* of the architectural design is determined by the material's mechanical properties, an example of which is the maximum allowed deflection, limitations from handling the material either done manually or by machine and the maximum allowable size for transportation to the job site.

The *shape of the panel* is mostly determined by the aesthetic preferences of the designer, in our case the architect of the pavilion.

Panel curvature deviation is determined by both technical and aesthetic limitations. The technical limitation based on two aspects. Firstly by the forces the panel material is allowed to have, secondly by nature of the connection detail between the panels and the structure below. In a scenario where the panels are clamped to the load-bearing structure, the deviation between the actual panel and the designed panel can not be more then what movement the connection detail allows for, or what extra movement the material allows. This could be a force that bends certain parts of the panel in place for it to be clamped down to the load-bearing structure.

7.2 Functionality

In order to determine which panel goes where, we need to have data that can guide the computational process to a good solution, an optimum. The given design in this research is a pavilion from which the data is generated. It carries info about what the preferred size of the panels is, the shape of the panels and the type of curvature each panel contains. Thus having both the panel size and shape predetermined by the design.

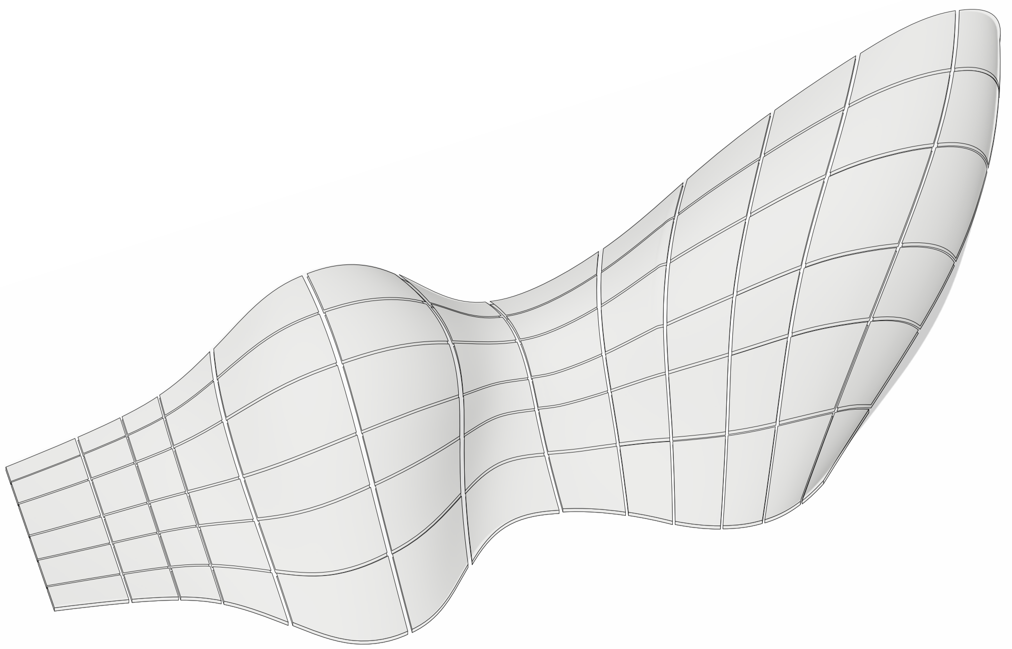


Figure 22 Case study pavilion design. By J. Sturkenboom, R. Wisse & author, 2015.

7.2.1 *Introduction*

Within this introduction the general functionality of the algorithm is shortly explained before we move on to the individual parts of the algorithm.

The first step is analyzing the design panel that we want to find a material match for. First the panel surface is divided for an amount of data points. For each point we determine the Gaussian curvature value, which relates to the surface geometry characteristics of that point. We also do this for the material surface, generating another grid of Gaussian curvature values. The following step is mapping the values of the panel onto the material surface, thus creating new values on the material that correspond to the panel values. This is done so that the agents converge to locations on that material where possible matches with the design panel are to be found.

Then the algorithm executes a set amount of times. Data is recorded about the panel geometry which each of the agents carry for each of the iterations that the algorithm is set to execute. Furthermore, the agents adjust their direction after each iteration, or time step, this is displayed in Figure 23. This is based on rules between the agents. The rules are alignment, cohesion, separation and attraction to points with corresponding curvature values, as explained in chapter 5.

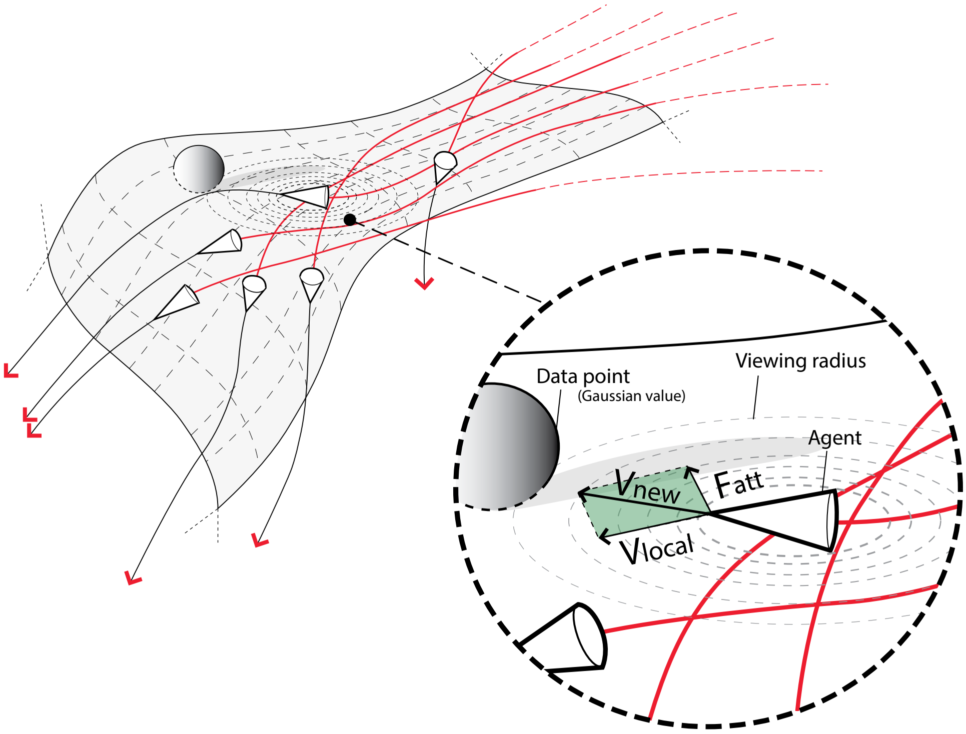


Figure 23 Agent viewing radius and point attraction. Based on image by M. Tsiliakos, 2012. Edited by author.

Also after each time step the faulty data entries are culled from the generated dataset and a visual update is given that shows the cut-out of the panel that most closely matches the design panel. After a set amount of time steps the algorithm terminates.

7.2.2 Determining and measuring geometry for Gaussian Curvature

Before the process described above, the first step in the process is taking a panel out of the design geometry and analyse this to determine Gaussian curvature, providing a certain range of Gaussian curvature values. The amount of queried surface points is of importance for the accuracy of the eventual matches of the material and design, and thus the optimisation as a whole. As explained earlier, Gaussian curvature is a value that contains information concerning a specific point of surface geometry. Having only a single data points on the surface, for instance on the middle, only concerns that specific point without having any relation to other information about the surface. By increasing the amount of points on a surface for which the Gaussian curvature is evaluated, the more specific the range of the panel will be.

Within this research this is done manually. Both the checking of the value range and making sure there is a match before running the algorithm. In the future this can be included into the algorithm, this is partly done because it is outside of the main focus

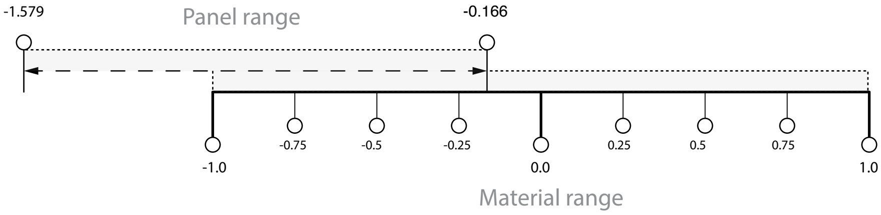


FIGURE 24 PANEL AND MATERIAL GAUSSIAN CURVATURE RANGE

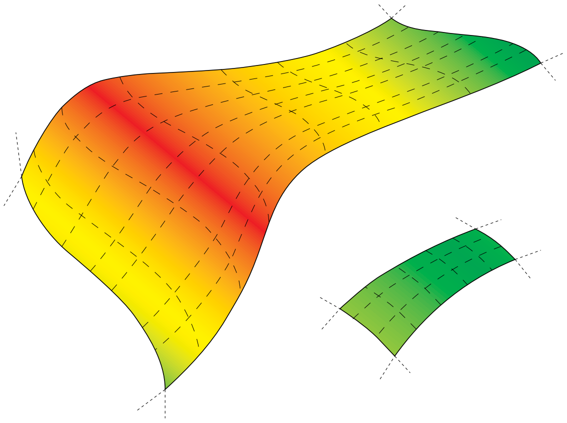


FIGURE 25 GRAPHICAL REPRESENTATION OF GAUSSIAN CURVATURE VALUE MEASUREMENT.

of the research and partly because of time constraints. The reasoning and steps to be taken in that case are explained below. Once the Gaussian curvature range for a specific panel is determined, the same process is done for the material out of which the panel needs to be made. By doing so we also generate a range of values for the material, as can be seen in. This gives us a first, clear indication if there is a chance for a match between both surfaces. For this process there are four scenarios to be considered as can be seen in Figure 26.

Firstly we have the option of inclusion within the range, where the data of the panel falls within the range of the material. This triggers the algorithm to look for a potential match. *Secondly* there is an option where the range of the panel only partly matches the Gaussian curvature range of the material surface. In that case it might still be useful to run the algorithm, since the tolerance of panel deviation might be large enough to provide matches still. *Thirdly* we have the option where the Gaussian curvature range of the panel has no match with that of the intended material. This should be a reason for the algorithm to pick the next material shape or type, in our case boat type, to continue looking for matching values. The last and *fourth* option occurs when the algorithm has to deal with flat source material. This results in a Gaussian curvature value of 0, thus making a range that exceeds the panel design if it has non-planar geometry.

The assumption made here is that we will be able to find matches for scenario 1 and 2. Scenarios 3 and 4 might prove more difficult to generate a proper match. Tests should be done in order to verify this.

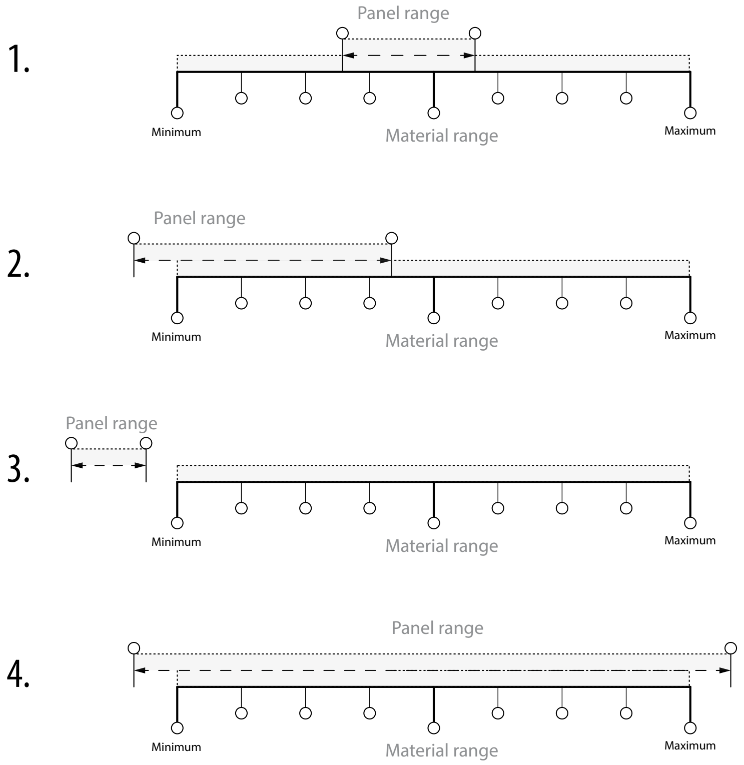


FIGURE 26 GAUSSIAN CURVATURE RANGE OPTIONS, BY AUTHOR.

7.2.3 Mapping geometry data

In order to start looking for potential matches we first need to properly convert the Gaussian curvature data from the panel onto the material surface in a way that the agents within the algorithm properly make use of it.

This can be done in two main approaches. One approach is giving the material surface a colour gradient representing the Gaussian curvature values. The other is a subdivision of the surface with a centre point in each subdivision along with the corresponding Gaussian curvature value of that point.

However, for both options the Gaussian curvature values that are found on the panel are considered the relative highest values and mapped as such on the material surface. The values that have been found on the material surface which do not exist on the panel will be adjusted to a lower value, making it less desirable. Resulting in agent behaviour that in theory quicker converges to the right values.

Within the first option, the colour gradient matches the Gaussian curvature of the previously analysed panel. Thus the part of the colour gradient to which the agent is attracted to the most represent the values found on the panel.

For the second option of subdividing the surface, the value given to a subdivision is the value of the centre point of that subdivision which is applied to the rest of the subdivision. Essentially giving that subdivision a single value for the agent to react to.

Increasing the amount of subdivisions on the material surface results in a more accurate representation of the subdivisions curvature values.

When applying a value to a surface subdivision for agents to react to, the viewing radius of the agent should always exceed the distance that exists between the centre points of

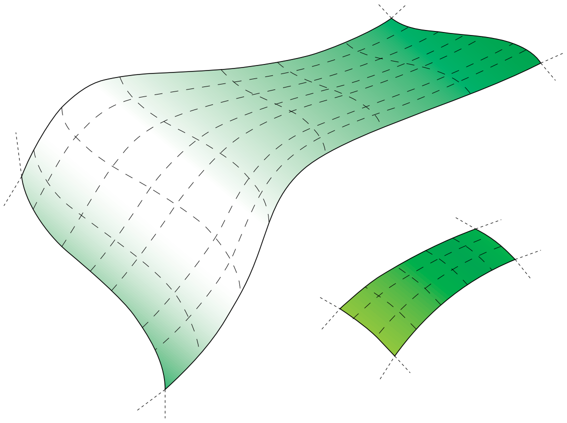


Figure 27 Gaussian curvature values graphically remapped

two or more subdivisions. The agents will only find the value within the subdivision and disregard other values if the viewing radius does not exceed the previously mentioned distance. By doing so we provide information in a manner that downplays the strength of agents to find local optima. Within this research we will use the second option of subdividing the surface and using the Gaussian curvature values of the centre point of each subdivision. This choice is made due to the limitations posed by the plug-in used for agent behaviour, Quelea(Fischer, 2015).

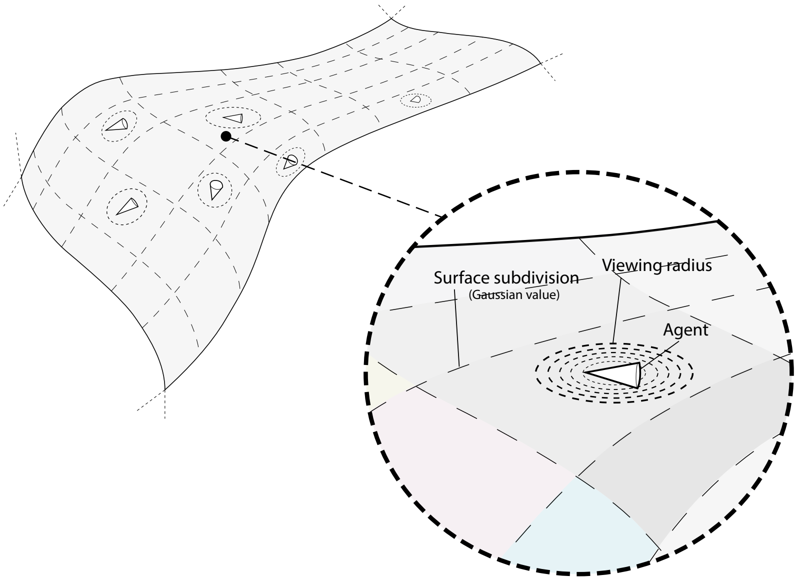


Figure 28 Subdivided surface with agent radius below size of subdivisions, by author.

7.2.4 *Information carried by agents*

After correct values are applied onto the material surface, agents are placed on the material. Each group of agents placed onto the material represent the same panel that we are looking to match with the material. This means that within this agent group each agent carries the same data for one particular panel. In order to let the agent carry the data about the panel, it needs proper orientation onto the source material.
For each panel the middle point on the surface geometry is determined by considering the surface of the panel in UV coordinates. The centre point is found at both U and V values of 0.5.

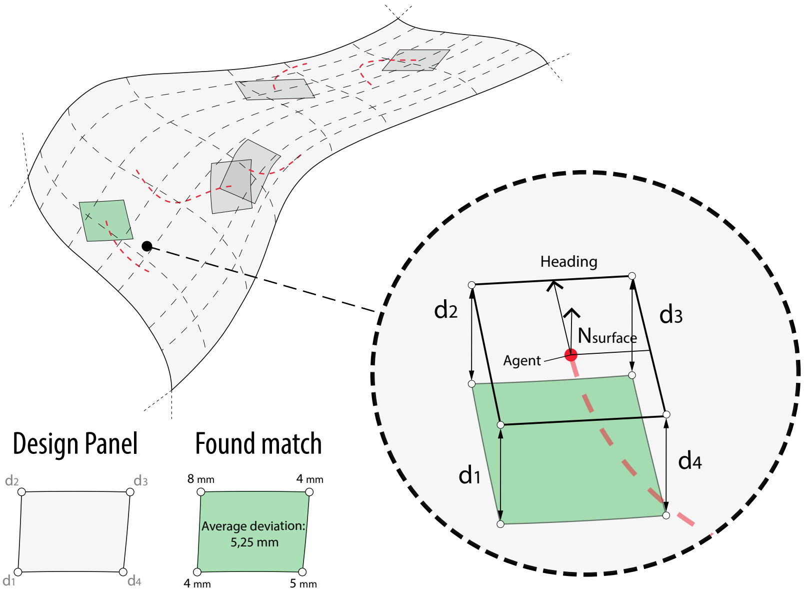


Figure 29 UV space of panel geometry and measuring distance between plane at agent location and material surface, by author.

The panel geometry is projected on a plane that is located on the agent's position in the environment. The normal direction of the plane always matches the normal direction of the point on the material surface that is the agent location. This gives us the tangent plane on the material surface, at the location of the agent.

Furthermore the panels that are analysed by the *evaluate surface* component found in grasshopper. This component analyses a surface at UV-coordinates. Each panel that is searched for on the material carries the previously mentioned tangent plane. Since we now know how the panel geometry relates to a predetermined plane and the location of the plane on which it will be projected, we still lack a definition that determines the orientation of the plane. This is defined by matching the y-axis of the design panel with the direction of the agent. The direction of the agent is a result of rules the agent follows. The rules are a combination of alignment, separation, cohesion and attraction (to Gaussian curvature) forces. Each of these rules is given a certain weight value in order to control the significance of force. This is relevant in case we would like to double the significance of cohesion compared to the other forces acting on the agents. Changing the main direction in which the panel is oriented on the plane carried by each agent will influence the results. We know that the principal curvatures of a point determine the direction of the minimum and maximum of the normal curvature at that point. These are also options to be taken as direction of the panel. It could be interesting to see how different orientations of the panel on the agent plane influence the matches that are found by the algorithm.

However, the Gaussian curvature value of a point on a surface is by definition not related to the rest of that surface. Furthermore principal curvature has a direction that can change if the location of the analysed point on a surface is changed by a small margin, effectively flipping the plane (and thus the design panel) upside down. This will influence the data generated by each agent and gives opportunity to unpredictable and harder to control behaviour. For that reason matching the panel orientation with a principal curvature direction is not considered.

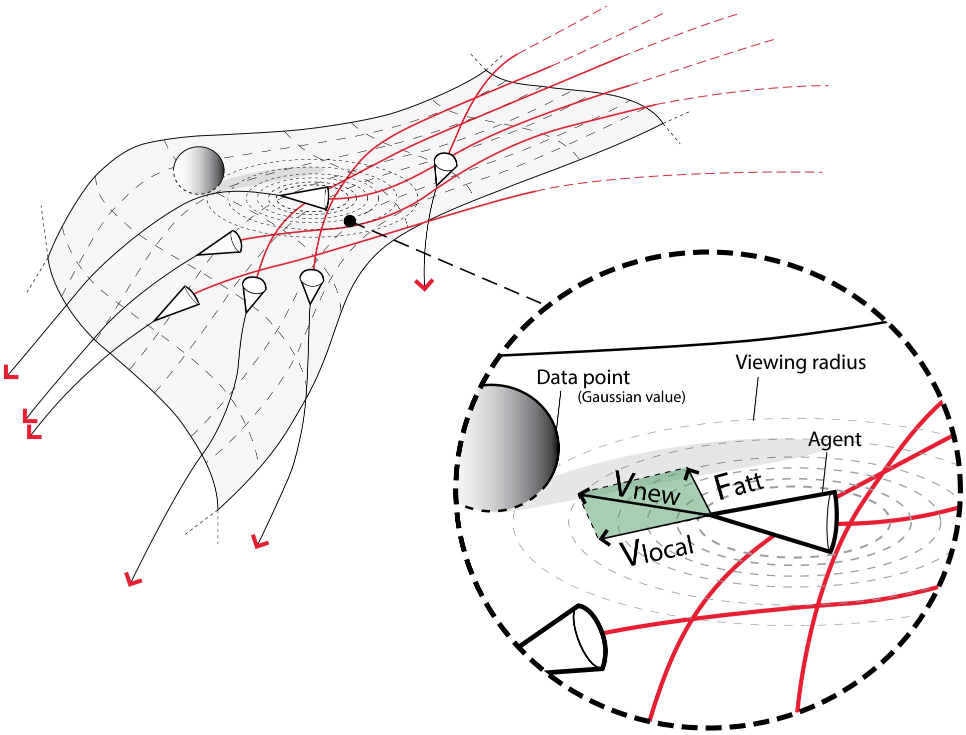


Figure 30 How an agent’s direction is determined. By author.

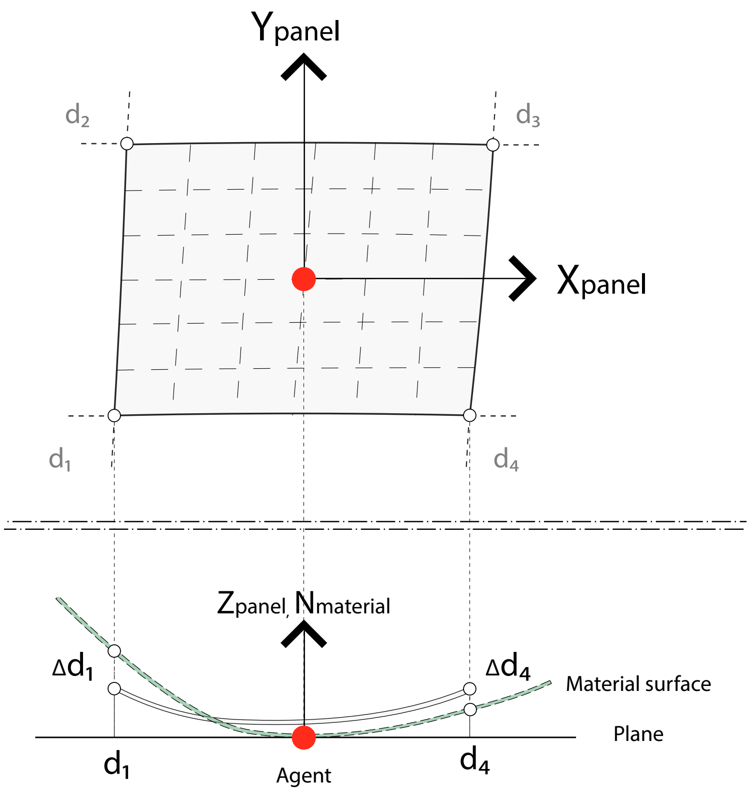


Figure 31 Minimizing the distance between the design panel and the match found on the material surface, by author.

7.2.5 *Data generation*

When all the data is correctly oriented the algorithm can be executed. The algorithm is controlled by a timer, which determines the amount of time between each calculation and the total amount of times the algorithm carries out the calculation. Thus the timer limits the algorithm to a predetermined amount of time steps so the computation time and results can be controlled. Moreover, this gives us the ability to see what the effects are of changes made in the settings of the algorithm if we keep the amount of iterations of the algorithm the same.

As previously mentioned, the panel we analyse is projected on a flat plane. We measure both distances, from the flat plane to the design panel and from the flat plane to the material surface. The algorithm's goal is to minimise the distance between the two.

When running the data is recorded for each data point that is projected on the plane. If a low amount of data points is analysed, deviation between the design panel and the match found on the material can be significant. By increasing the amount of data points on the panel this deviation can become less significant.

Furthermore faulty entries in the data set are removed. A panel and its data generated are considered to be faulty if a part of the panel surface is not on the material surface.

More specifically meaning if one or more data points analysed are not projected on the material surface and thus do not record any measured distance, the panels will be considered bad and removed from the data set.

This becomes relevant for determining the solution that best matches the design panel. The process of determining the best match is done by taking all the distances measured between the data points added together and dividing them by the amount of data points that are analysed for the panel. This solution can lead to significant deviations between the design panel and the best match, especially when the amount of data points taken into consideration is low.

If only 4 data points for a panel are considered, a most fitting solution could still have 3 data points matching and one deviating by a significant margin, resulting in the lowest average deviation per data point, yet resulting in an undesirable aesthetic result. Yet, we still use this solution as it is a relatively simple approach and increasing the data points can result in a reduction of the deviation between the panels and also minimize the possible negative effect on the aesthetic result.

To further increase the accuracy of the found matches a more delicate approach could be taken in the future, where such effects as explained above can eliminated.

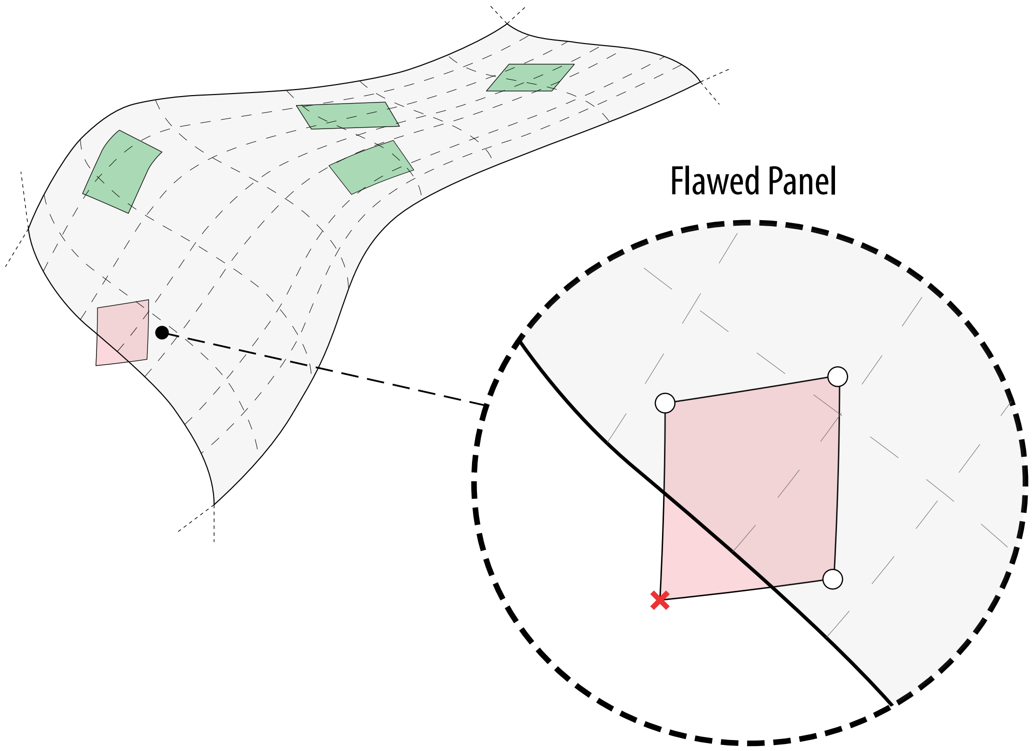


Figure 32 A data entry where one or more data points of the searched panel exceed the environment boundary, resulting in a faulty panel, by author.

7.3 Conclusion

We consider one panel for one agent group in this algorithm. This generates a iterative process where a panel is cut out of the material, and the leftovers are then again used as a new environment for the agent to look through. The advantage of this is there will be no conflicts between cutting patterns on the material. However, this consequently terminates the ability to consider sub-optimal solutions generated by the agents for panel choice, which might have favourable placement on the material compared to other panels that need to be found. Thereby also eliminating the chance of being more efficient with the material that is used.

This specific problem is significant enough to take into consideration when continuing the research. A solution could possibly be found when generating multiple groups being active at the same time in one environment. In such a way each agent group carries information about a panel. The agent groups can communicate between each other and generate solutions that do not interfere with one another. A similar setup of the agent groups would tackle the previously mentioned problem of having to iteratively readjust the agent's environment with the material surface that the last panel has been cut out of.

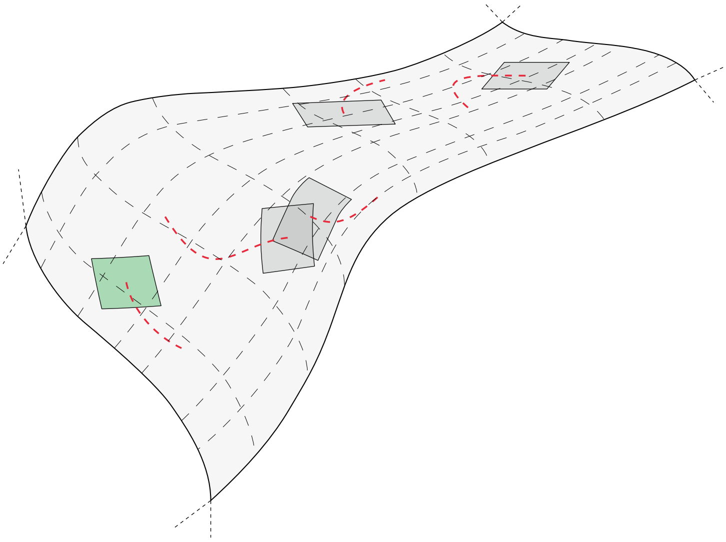


Figure 33 The algorithm has found the most optimal solution on the given material surface, by author.

8. Algorithm tests

The logic behind the algorithm is explained in chapter 6 along with the previously mentioned constraints for the algorithm within this research in terms on material and case study design.
To gain further understanding about the functioning of the algorithm and the capabilities it has within this applications several tests have been carried out on the specified case. A set of trial runs of the algorithm have been carried out to generate a basis which we can use to conclude if the outcome of the algorithm is as expected and useful for further application.

8.1 Input

For these tess the input is changed to those specified earlier. The input is the panel with which a match has to be found and the material surface we search through for the match. By running the algorithm we converge towards examples that more accurately showcase realistic shapes that we expect to find on a designed surface and on the material surfaces used.

In the tests ran we have found matches for a total of 30 panels instead of all the panels that the pavilion has in the design. This is done due to time constraints that the research was bound to.

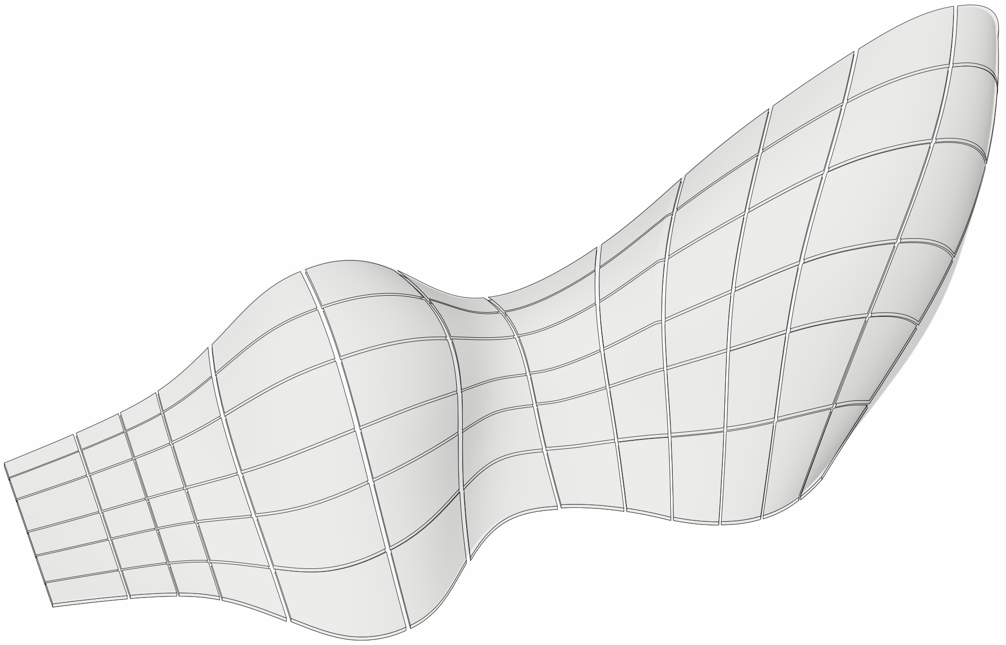


FIGURE 34 THE CASE STUDY PAVILION DESIGN, BY AUTHOR.

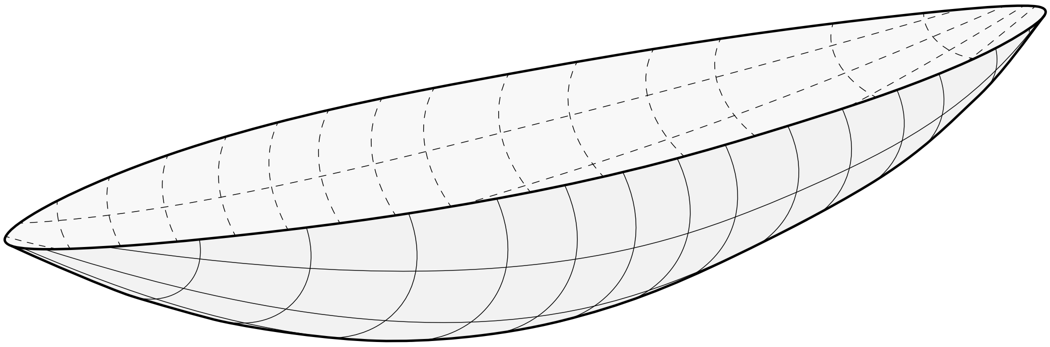


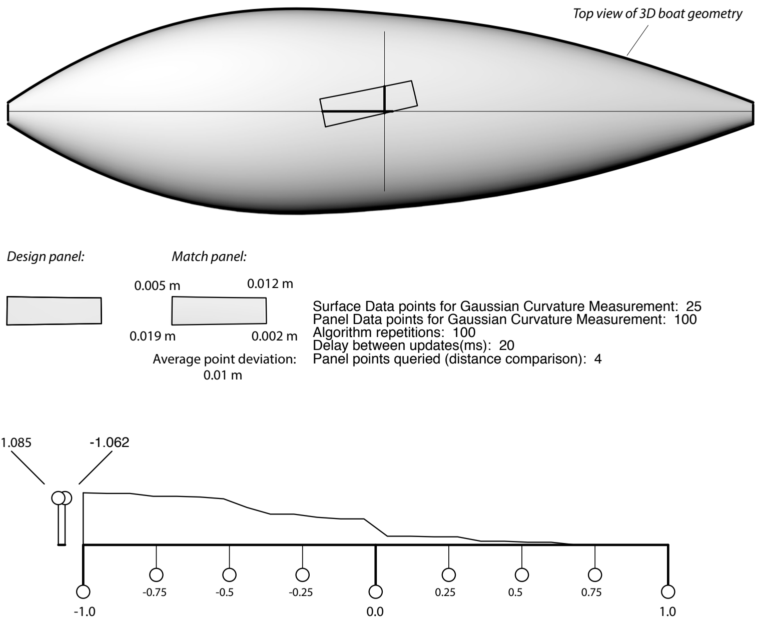
Figure 35 The 3D geometry of the boat, type 'Defender 15', by author.

The panels originate from the design specified in chapter 5 and can be seen in Figure 34. The other main component needed is the material out of which the design will be made in theory. This serves as the environment in which the agents are placed and which the agents search for the optimal place on the boat hull to cut the panels. The boat type used for this process is the Defender 15.

8.2 Output

The output of the algorithm consists of several data points and some visual feedback. Figure 36 shows an example of the output the algorithm generates.
The feedback consists of the amount of data points used to generate the range of Gaussian curvature found, both on the material and the panel. This refers to the data points which determine the Gaussian curvature value range, where the range from -1.0 to 1.0 are generated by the material surface (environment) and the values on top, are the values found on the panel. Thus the material Gaussian curvature determines the range created. The actual Gaussian values have been remapped (from -1.0 to 1.0) in order to have the proper weighting for the agents to react to.
Furthermore the amount of data points is provided that the agent carries during the search for matches, in this case 4, one for each corner point. These are the data points that are compared between the design panel (left) and the found match (right). Furthermore the amount of algorithm iterations that has been ran is provided, which is 100 for all the found panels. Also the delay between iterations is given, which was at 20 milliseconds for each of the algorithm runs.
The amount of agents used in the algorithm for each panel is set at 30.

This process has been repeated for 30 panels, on the material. The data below shows what the deviation is for the best match found by the algorithm. The panels are selected by average deviation, but since there might be discrepancy between the data points, also the values for each data point are presented.



Surface Data points for Gaussian Curvature Measurement: 25
Panel Data points for Gaussian Curvature Measurement: 100
Algorithm repetitions: 100
Delay between updates(ms): 20
Panel points queried (distance comparison): 4

Figure 36 Algorithm output, by author.

8.2.1 Panel location

When the algorithm is terminated and the best option for one panel is found, the new panel and environment for the agents to search through (the material) have to be provided manually. In Figure 37 a top view is shown of the case study pavilion that provides the locations on the panels that have been searched for by the algorithm.

In Figure 38 a representation is given of the locations that the agents have found on the boat hull for each panel, present per row. All the data found per panel is displayed in Table 2 and in Figure 39 we can see what the range of Gaussian curvature values are of the panels compared to the range from -1.0 to 1.0 that is determined by the material surface.

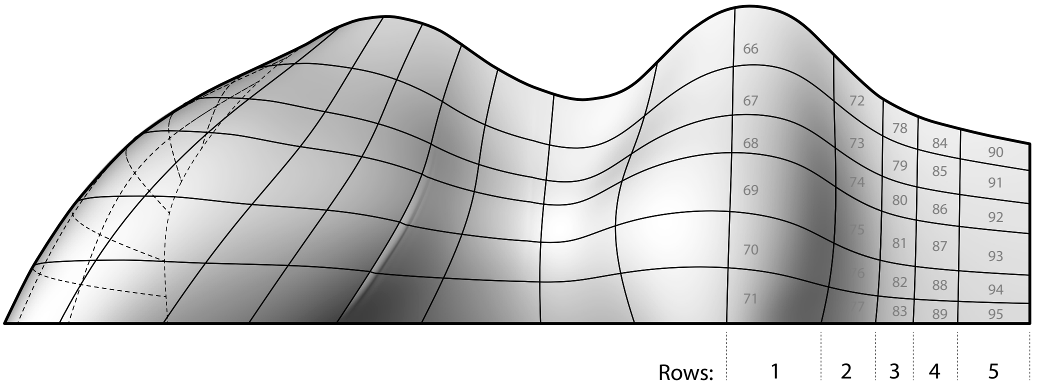


Figure 37 Top view of case study pavilion, by author.

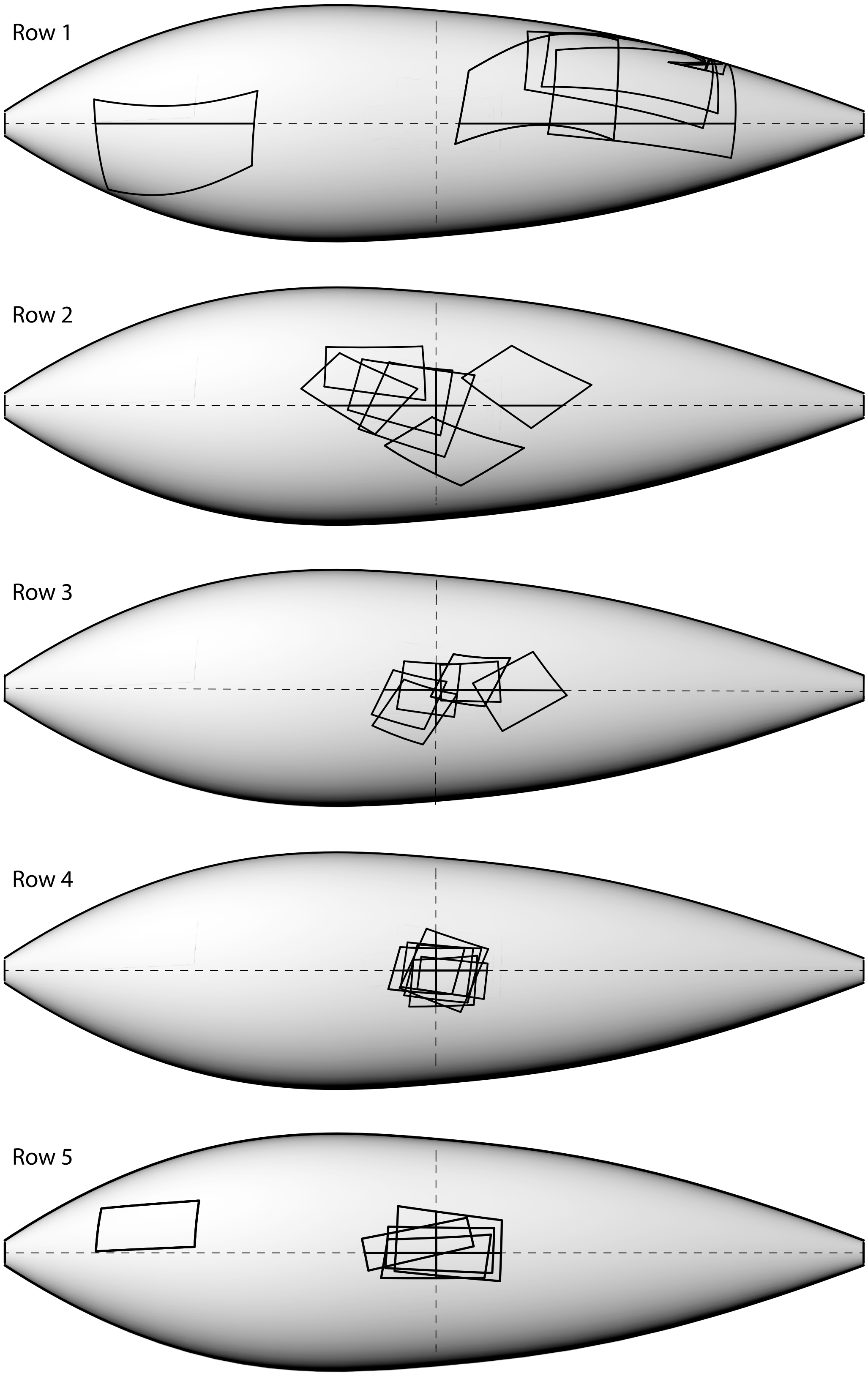


Figure 38 Cutting location of the found matches, by author.

Row	Panel	Average deviation (m):	Data points (m):			
			1	2	3	4
1	66	0.015	0.011	-0.002	-0.018	0.028
	67	0.017	-0.016	-0.015	0.02	0.018
	68	0.033	-0.088	0.022	0.002	-0.018
	69	0.034	-0.074	-0.005	0.026	-0.033
	70	0.032	-0.104	-0.004	-0.005	-0.015
	71	0.022	-0.023	-0.039	-0.004	0.022
2	72	0.006	0.008	-0.004	0.001	0.010
	73	0.004	0.002	0.004	0.004	0.008
	74	0.007	0.001	0.013	-0.006	0.006
	75	0.017	0.035	0.024	0.001	0.007
	76	0.014	0.030	0.010	0.000	-0.015
	77	0.012	0.024	0.004	-0.002	-0.020
3	78	0.007	0.000	0.014	0.006	0.007
	79	0.009	0.009	0.007	0.011	0.007
	80	0.007	0.003	0.011	0.006	0.010
	81	0.017	0.026	0.007	0.028	0.007
	82	0.009	0.011	0.009	0.009	0.007
	83	0.002	0.004	0.001	0.002	-0.001
4	84	0.006	0.004	0.008	0.006	0.007
	85	0.010	0.009	0.011	0.010	0.009
	86	0.008	0.002	0.015	0.002	0.014
	87	0.020	0.034	0.009	0.031	0.007
	88	0.011	0.014	0.010	0.011	0.008
	89	0.005	0.012	0.000	0.008	0.000
5	90	0.006	0.001	0.009	0.005	0.010
	91	0.010	0.008	0.012	0.009	0.010
	92	0.033	0.022	0.036	0.036	0.037
	93	0.020	0.001	0.038	0.004	0.036
	94	0.009	0.006	0.013	0.003	0.013
	95	0.010	0.019	0.005	0.012	0.002
Average:		0.014				

Table 2 Algorithm outcome for panel matches

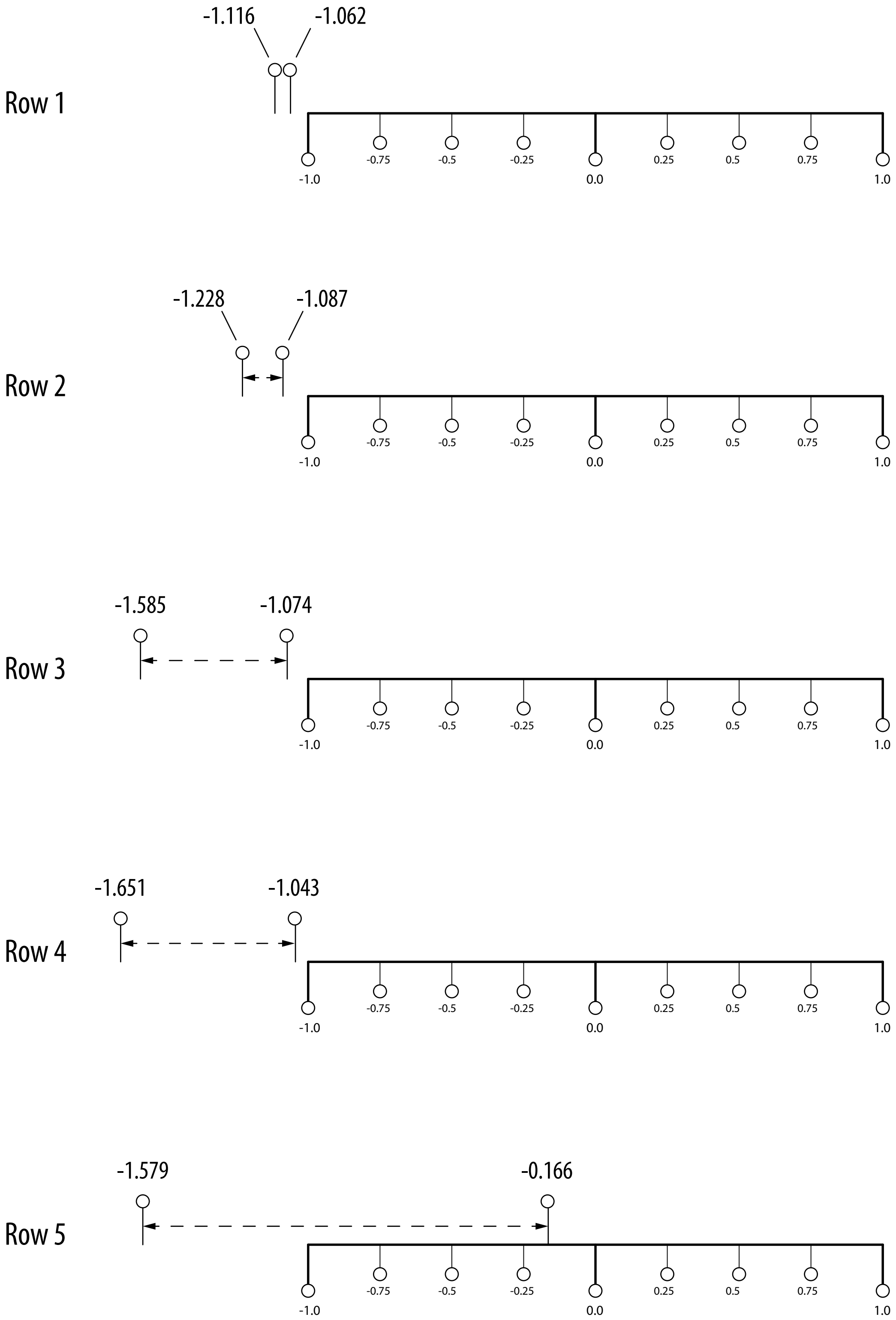


Figure 39 Gaussian curvature values of the panels of each row, mapped on a Gaussian curvature range determined by the material.

8.3 Visual comparison

Here a set of views is displayed in which the matched panels are displayed alongside the original design of the pavilion. The panels displayed as 'found' are all coming from the same boat type, the defender (Figure 35), as is the case through this chapter.

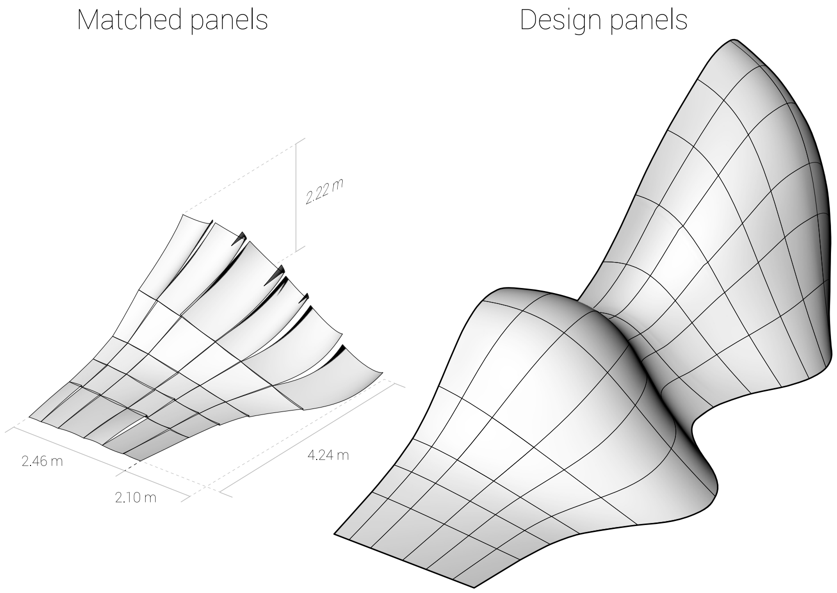


FIGURE 40 PERSPECTIVE VIEW OF THE MATCHED PANELS WITH THE DESIGN PANELS, BY AUTHOR.

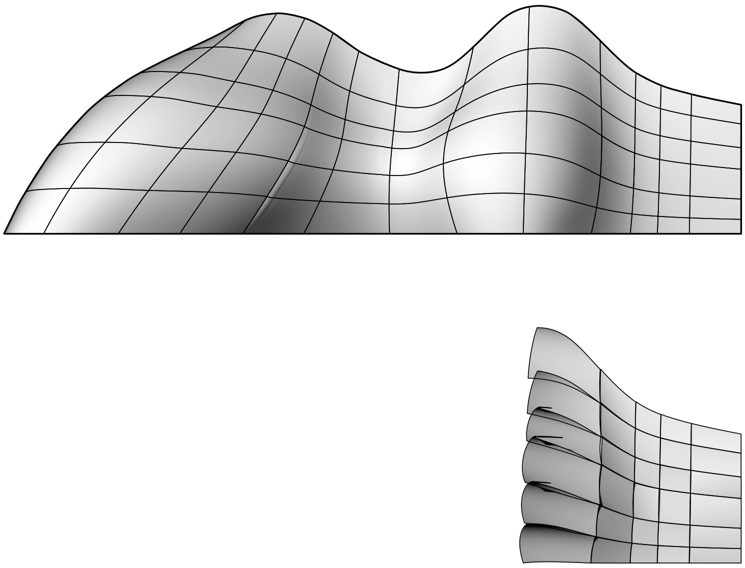


FIGURE 41 TOP VIEW OF THE MATCHED PANELS AND DESIGN PANELS, BY AUTHOR.

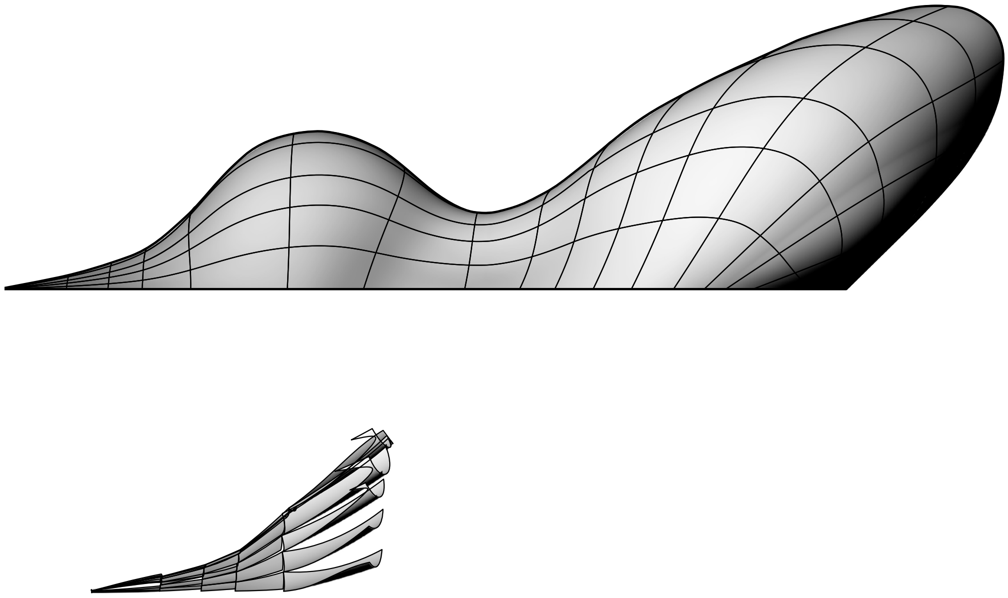


Figure 42 Back view of matched panels and design panels, by author.

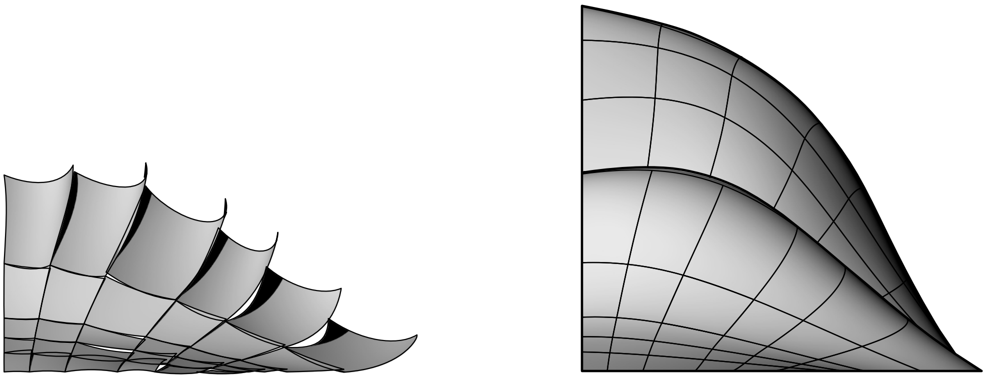


Figure 43 Side view of matched panels and design panels, by author.

9. Building method

This chapter provides an overview of the building method and where this research fits into that process. The research itself is an algorithm and only a small part of the overall implementation of any building realisation work-flow in reality. In order to generate a better understanding of the overall work-flow, this chapter will further explain how such a work-flow is envisioned in practice.

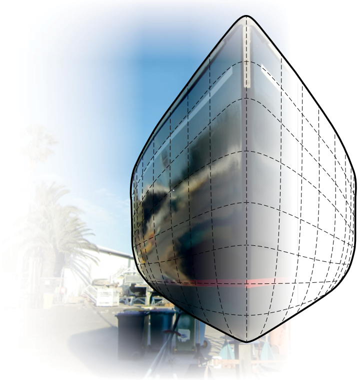
Firstly the creation of a digital library will be explained. This is a digital warehouse that holds all the shapes and values necessary to find matches between material and design. The next step in this method is the design analysis where we measure and determine which values that need to be searched for in the material.
Once these two initial steps are taken, the algorithm has the required input in order to find matches between the design and the material.
The output the algorithm generates can later be used for material ordering lists, cutting locations on the material and the accompanying code that the computer numerical controlled machines require. The output can then be used in a processing factory where cutting, pre-assembly of the structure, disassembly and packaging is done. The created material is transported to the construction-site and assembled. The final step is the creation of a maintenance plan in order to extend the usage (life-time) of the material to the maximum possible length.

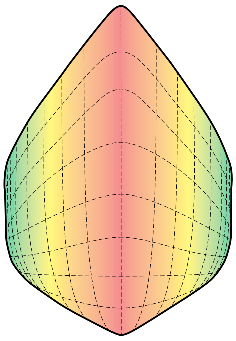
9.1 3D Library

Research the **total amount** of material available (mass and quantity)

Determine most common **types** out of the total amount

3D-scan the most common types, effectively creating a large part of the actual waste material as a digital 3D library/warehouse, to be queried by the algorithm/agents. Most likely there will be no 3D models available of the boats that are the material resource, considering that the majority of the boats previously mentioned in this research date from around the 1970's to the 1980's. As the building industry moves towards more awareness about material use (a sustainable building industry), the trend seems to be moving towards generating material passports of buildings. Similar principle is applied on this material, where we want a full passport (identification) of the material, the amounts, which types represent the biggest mass.





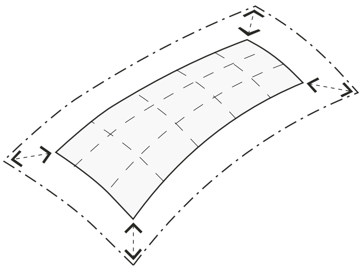
Supplement the material **library** with relevant **information** in general (amounts per type) and for the optimisation goal(Gaussian curvature). In our case this means analysing all the types of material in the 3D library for Gaussian curvature.

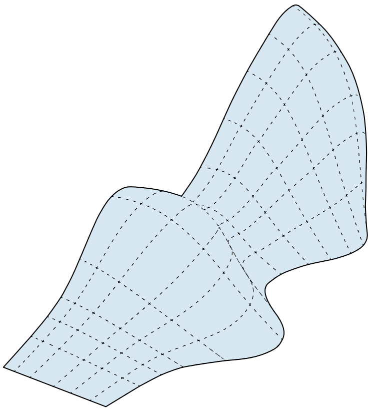
Input for the algorithm: Variety of 3D shapes for the algorithm to query.

9.2 Design analysis

The input for the design could either be a surface with panelling idea already in place (predetermined by the designer, architect), or with future development a full, undivided surface instead of a subdivided version of it. In the case of providing a full surface without panelling, the panelling subdivision of the surface could be given as an outcome of the algorithm, with certain materials and panel shapes resulting in different aesthetic outcomes. This would however require significant further development.

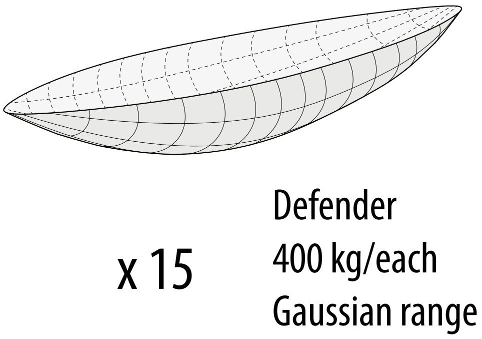
Determining the **tolerances** for the algorithm. Maximum deviation, maximum gap between panel edges, size & curvature. Each design comes with certain criteria. These criteria influence the aesthetics of the design and the end results. Criteria can be the maximum width gaps between adjacent panels, the surface continuity level and more.

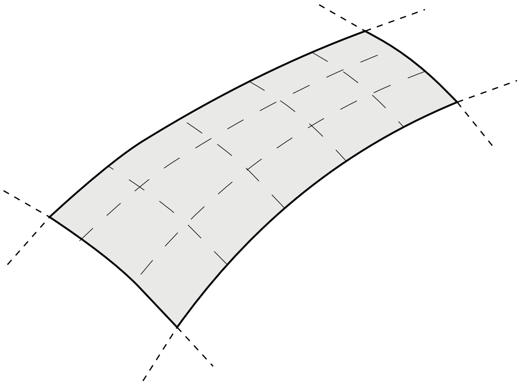




Analysis of the design **full surface.** Match curvature data range with material. This information can be used to make a pre-selection of the material suitable for the design.

Pre-selection of **material.** For example, a set of 20 boats could be (almost) fitting the curvature range of the design, thus possibly housing solutions.





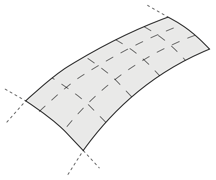
Analysis of a **panel** (from the design). Pick a panel from the design, analyse the specific panel for its curvature values, find the best matching boat out of the pre-selected material set. Iterative process until design surface is fully covered.

Input for the algorithm: Panels (3D surface shapes) for the algorithm to find in the material.

9.3 Algorithm

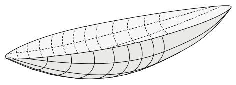
The input for the algorithm is established, a digital warehouse/library with 3D shapes of the material and the design surfaces (panels) that need to be realized.

Panel



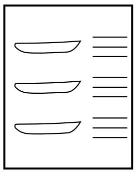
Running the **algorithm**
Matching geometry of the source material and design. After the materials are digitized and the correct settings are determined, the algorithm can find the proper match for the design panelling.

Material

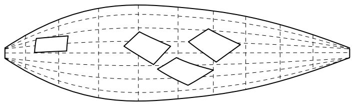


Algorithm output
The algorithm generates the location of the panels on the material surface. This subsequently results in a set of boats that are needed, the cutting locations of the panels on those boats and the necessary code to run the CNC machines.

Material list



Cutting locations

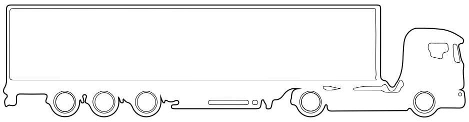


Code for machinery

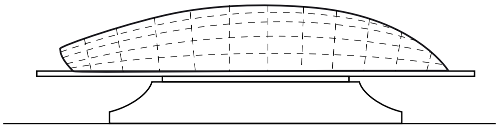
```
0100111011001110111101011010101001110110011101111010110101010011101100111011110101101010100111
0110011101111010110101010011101100111011110101101010100111011001110111101011010101001110110011
1011110101101010100111011001110111101011010101001110110011101111010110101010011101100111011110
1011010101001110110011101111010110101010011101100111011110101101010100111011001110111101011010
1010011101100111011110101101010100111011001110111101011010101001110110011101111010110101010011
1011001110111101011010101001110110011101111010110101010011101100111011110101101010100111011001
1101111010110101010011101100111011110101101010100111011001110111101011010101001110110011101111
0101101010100111011001110111101011010101001110110011101111010110101010011101100111011110101101
0101001110110011101111010110101010011101100111011110101101010100111011001110111101011010101001
1101100111011110101101010100111011001110111101011010101001110110011101111010110101010011101100
1110111101011010101001110110011101111010110101010011101100111011110101101010100111011001110111
101011010101001110110011101111010110101010011101100111011110101101010100111011001110111101011
```

9.4 Processing

Transporting the selected boats to the factory. The selected material is on a list, including information about the types and amount per type. These are transported (physically), from their current location to the processing factory. Transportation most likely will happen by truck.

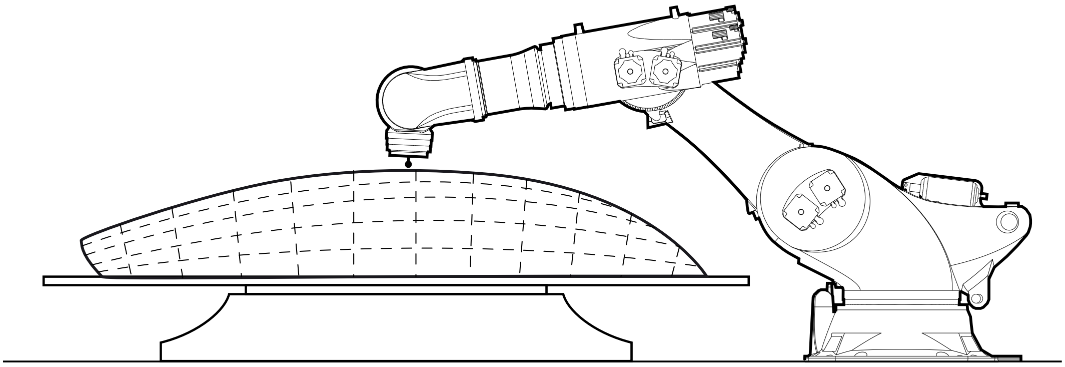


Placement and fixation of the material for processing. The material is fixating into place and the exact location for cutting is determined. This fixating can be done with one or more robotic arms, or some kind of clamping system on a machine bed. The advantage of holding the material with a robotic arm is a greater freedom of movement. This allows for more cutting actions once the robotic arms hold the material properly. However, the carrying capacity of the robotic arms should allow for such an approach and possible vibration of the material and the arms during cutting should be considered, as it might reduce the accuracy of the final outcome.Clamping the material is place allows only for rotational movement of the material if the machine bed allows for it. Otherwise repositioning of the material between cutting actions might be necessary.

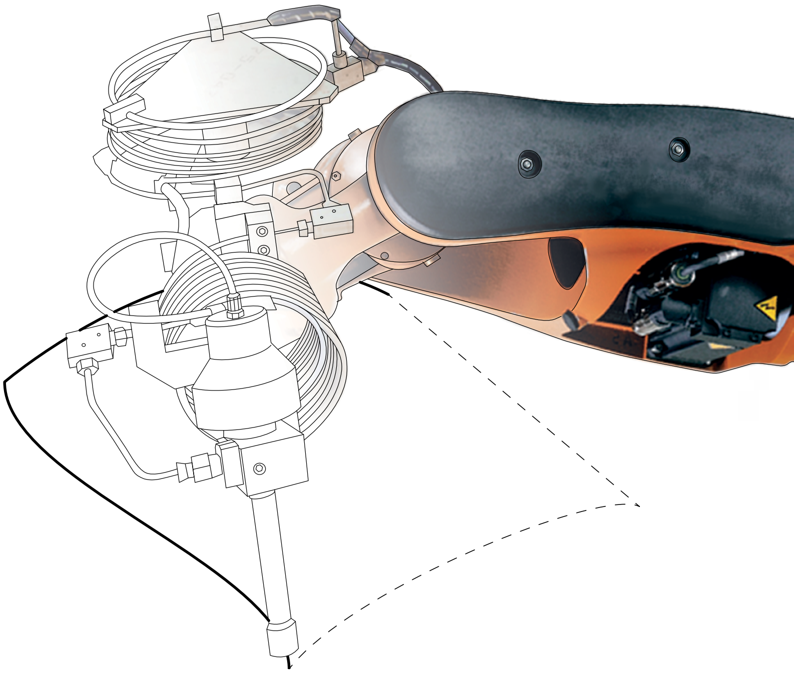


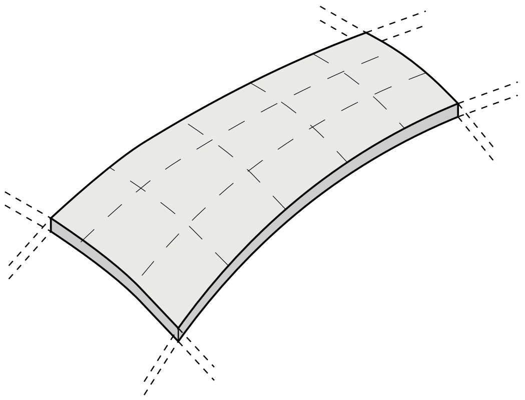
Precise location of the material

A robotic arm can determine the precise location of the material on the machine bed with a probing tool.



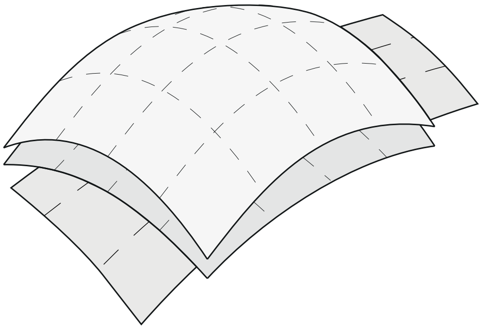
Cutting the material A robotic arm, equipped with the proper cutting tool, cuts the panels out of the material and applies an identification tag on each of the panels before it is placed in temporary storage. Another robotic arm could hold the cut out panel in place. This prevents the panel from falling and allows proper stacking of the panels and possibly reduces the vibration during the cutting process, thereby increasing the accuracy of the final result.

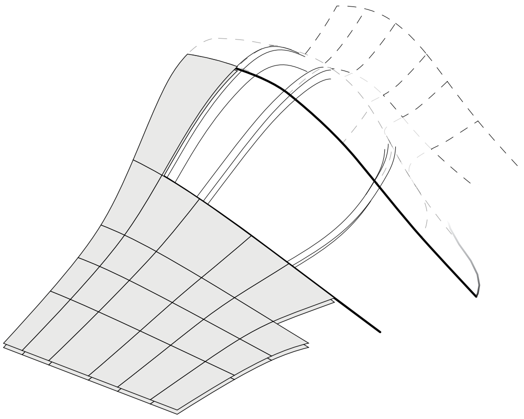




Post-processing. After cutting the panels could still be altered in their appearance if needed. This could, for example, be the sanding of the panel edges or applying a new coating on the panel.

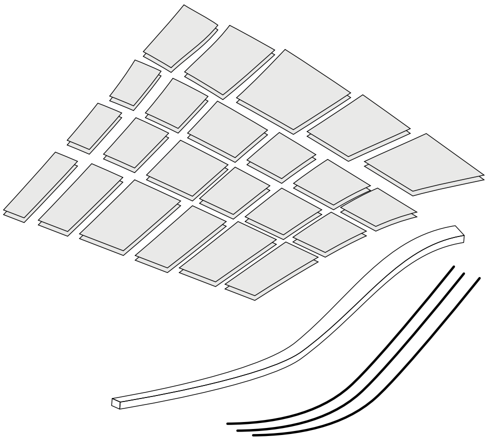
Temporary storage. The material is placed temporarily in the factory hall until all the panels are produced. This is a useful addition to the work-flow as it eliminates the need to take assembly sequence into account when cutting the material. The simple reasoning is that panel x and panel x+143 might be found on boat y. The most important determining factor for this is time. Time increases rapidly if the process requires many changes of material in the cutting process, so we want to keep the amount of material changes for the cutting process as low as possible.

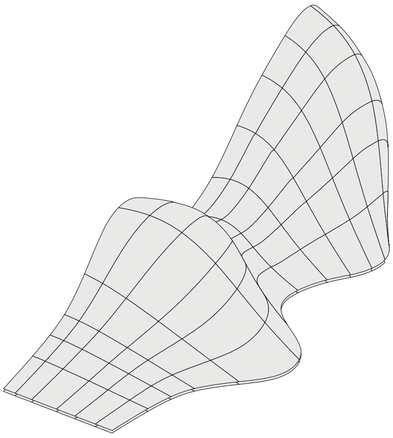




Pre-assembly. Pre-assembly should be done at the factory to prevent time-loss at the construction site and in order to find possible deficiencies in the panel material, the detailing or the load-bearing structure. After full assembly, the material is disassembled and packed in order of disassembly.

Transport and assembly. The panels are transported to the construction site and assembled.





Maintenance. A maintenance plan is need in order to extend the usage (life-time) of the material to the maximum possible length.

Evaluation

10. Conclusions & Recommendations

The conclusions and recommendations can read in two separate parts. The first parts of each paragraph concerns the algorithm process and how this determines what part of the material input is used in order to create a part of the case study design. The second part concerns how the algorithm fits into the overall design process and what implications this has on realisation.

10.1 Conclusions

This research concerns agents, an optimisation technique that is used on a specific case study design where the facade panelling of the design have been made out of discarded glass fibre reinforced boat hulls. Thereby creating the theoretical starting point of the work-flow that will use discarded materials and re-use them in architectural application. The implications of this research can be diverse. It goes into a specific optimisation technique that is used for a complex geometry problem, yet it also proposes a work-flow and the vision for realising the optimisation logic proposed in this research. Realisation of such a logic has the potential to generate a positive impact on the environmental problem some materials cause, which currently might be considered waste material.
The conclusions can read in two separate parts. The first part concerns the algorithm process and how this determines what part of the material input is used in order to create a part of the case study design. The second part concludes how the algorithm fits into the overall design process and what implications this has on realisation.

Agents can be considered a computer system that follow a set of rules determining their behaviour. As the agents function, a certain behaviour emerges based upon the set of rules they follow. This emergent behaviour of agents is used to find a match between curved surfaces by matching the Gaussian curvature values. This is done by analysis the separate surfaces and using their Gaussian curvature values as a metric within the behavioural system of the agent software. Furthermore, each rule has a certain amount of importance for the agent's behaviour, a weighting factor. These rules are forces that influence the redirection of each agent's heading. The agent re-computes its heading after each computation of the algorithm. The previously stated rules are alignment, cohesion, separation and attraction towards one or more points. The first three rules respectively determine how the agents flock together, much like birds do in nature, whereas the point attraction force represents the amount of surface curvature. By giving the right amount of importance to the behavioural rules, the agents generate desired results for the amount of iterations and data point it is based upon.
Only rectangular designed panels are considered in this research, where each of the

four corner points are measured against a flat surface. As the agents move across a curved surface, they also carry these points with them on a plane perpendicular to the surface. After each time step they measure and record the distance between that perpendicular plane and the material surface. When comparing the design panel with the panel that is found on the material surface, the panel that has the least amount of deviation across the 4 data points is considered to be the best match after 100 iterations, or time-steps, of 30 agents. This process is then repeated for each panel that needs to be materialized. The logic used within this research it has a downside towards the objective of efficiently using materials, since the repetitive nature of the algorithm does not account for comparison between locations of separate panels on the material surface, which might lead inefficiency of material use.

We can see that the results of the 30 panels for which a material match is found in this research, the average deviation of all 30 panels is 0.014 meter. The panel with the highest average deviation of its four points has a value of 0.034 meter, which is panel 69 in row 1 of the dataset. The panel with the lowest average deviation has a value of 0.002 meters, which is panel 83 in row 3 of the dataset. These are also the panels with the highest surface area and the lowest surface area respectively. Panel 69 with 1.62 m^2 and panel 83 with 0.19 m^2. As this concerns 4 data points per panel, a number that can be increased if computing power allows it, deviations can be reduced and the accuracy of the matched panels increased. The amount of 4 data points measured per panel was chosen in order to keep computational time low during the development of the algorithm.

As the range of Gaussian curvature of each of the panels, compared to the material, was mostly outside of the range, we can conclude that we are still able to find matches of the design panels on the material. Thus a range of Gaussian curvature values of panels that lies outside of the range that is found on the material surface does not necessarily mean that there is no match to be found. Yet the aesthetics that the matches produce might argue otherwise, that is an argument for the designer and outside of the scope of this research.

Furthermore the outcome of the algorithm provides us with the knowledge that the logic created in a parametric environment works for the objective of this research. We are able to place parts of the material source in such a way that it closely resembles the design intent that is also the input of the algorithm. This can be further increased by tweaking the settings of the algorithm. But more importantly it is a good starting point. It does however need future comparison with other computation techniques in order to validate which one is the most efficient in terms of computational time and power and which technique produces desirable results.

The algorithmic process described by this research produces some outcome, based upon settings and input of the algorithm. These are the input material, design and the settings with which the optimisation is being done. The outcome of the algorithm then provides us with some visual feedback and a set of numbers that relate to the visuals. If the settings or input are changed, so will the output. Without proper context in which the algorithm is used, it loses value. Therefore an overall context is provided in this research for the architectural application in which it is used.

This starts with the generation of a digital library that holds 3D geometric models of all the material with the accompanying values, in our case Gaussian curvature values. In order to tackle the environmental problem that feeds the input of the algorithm and in order to generate outcomes to closely resemble the design intent, it is important to grow this digital library. Once this digital library is established, the design input can be analysed for curvature values found. The algorithm is then able to do the hard work of finding a proper match between the two before all the required material is transported to the processing facility. There the cutting of the proper panels out of the material takes place. Once this process is completed and the material is properly labelled, it can be transported to the building's construction site. In the multitude of steps in such a process, one of the steps in such a process can not be considered inferior to another, as this might negatively impact the final outcome of the work-flow. If all the steps proposed are considered with equal importance and care, there is a chance to apply the theory in such a way that it provides a new direction that is able to help combat an emerging environmental problem of glass fibre-reinforced plastic boat hulls in the Netherlands while also increasing awareness as the materials can be seen on the façades of the built environment.

10.2 Recommendations

For future development of the research there are several actions to be taken to improve upon the work-flow as it is explained. Firstly a comparison should be made between the logic as it is now which uses agents and other optimisation techniques. This subject is left untouched due to time constraints mainly, yet it is interesting to see if other optimisation techniques offer better functionality in terms of design output or if other techniques are more efficient and able to better use available computational power.

Also the functionality of the algorithm can be improved. An interesting step could be the coding of the process in a better dedicated software package, as the plug-in used in this research, Quelea(Fischer, 2015) and Grasshopper(Robert McNeel & Associates, 2018) do pose limitations, where other software might not, or be able to more efficiently compute such a process.

The logic as it is within this research is based upon iterations that need to be started and fed with the right information manually, meaning that one panel is sought for by 1 group of agents at a time. A potential lies in the characteristics of agents, where multiple agents, or agent groups, each could represent a panel each. Thereby looking for multiple panels at the same time whilst also providing the opportunity to improve the placement on the material and thereby having a possibility to more efficiently use the material within the algorithmic process.

Agents can also have a certain level of intelligence, one such part of intelligence as described in agent theory is learning. Future research should be done into the addition of an algorithm that applies learning into the process. An imagined example of learning agents in relation to this research; once the algorithm has ran an amount of iterations, it could link parameters such as a design with certain curvature and pattern to specific boat types. Once the relation is formed between hexagonal patterns with curvature κ and boat type X, the algorithm could directly jump to the proper boat(material) type when facing a similar problem. Preventing the algorithm to check a vastly bigger database and thereby wasting energy and (computational) time.

Finally an improvement can be made in how the Gaussian curvature values of the panel are displayed on the material surface for the agents to use. Right now the weighted values given to a point is a functional and robust solution, yet it is rudimentary and has limitations.

The main body of the report concerns the explanation of the logic behind the algorithm that finds matches between curved surfaces. The context of the algorithm also holds large significance towards the algorithm. In order to develop this, certain steps should be considered to be taken. Eventually the overall work-flow would greatly benefit from a phase where the overall process is tested in a realistic setting. Before applying this work-flow in a realistic case, other steps need to be taken.

The first of which is the digital library that feeds the algorithm with source material . It should be expanded upon and be specified more precisely as the library has a large influence on the design output and might face problems such as 3D scanning the proper material types and how these scans can be used by the algorithm.

Once a proper library is established, intelligence could be added to the agents that provide learning and a possibility to improve functionality of the algorithm. Also the direction into a design tool or software plug-in for architects is an important future development. If the content of this research can be used to create a plug-in that would provide architects and designer with a direct link to the digital library, they could be included early in the design process and have control over what they are able to design with the available materials. This can be seen as making a loop out of the linear logic applied in the algorithm as it is described in this research.

The work-flow has focussed on glass fibre-reinforced polymers as double curved façade panels. Because of the parametric nature of this work-flow, adapting it towards another source material or another design intent can be done with ease while also increasing the utility of the logic. Therefore it is also of interest to further explore what the potentials of this logic are for other materials that could be fed into the algorithm and for what other design goals it could be used.

In a later stage for the particular material chosen in this research, additional research is required into the processing of the glass fibre-reinforced plastic material. One part of processing is the cutting of the material. This creates edges where fibre layers are exposed and will most likely lead to increased rate of deterioration of the panels. This process can create structural failures if layers of the material de-laminate. Moreover, material treatment options should be researched, such as the impact of sanding or re-coating of the material, as such a treatment might be able to extend the usage and lifetime of the material and could create a positive impact on the aesthetic qualities of the material. This could increase the options for application of the material in different designs, where the architect strives for a more uniform appearance of the façade.

11. Bibliography

Aschwanden, G., Halatsch, J., & Schmitt, G. (2008). Crowd Simulation for Urban Planning. *Architecture in Computro - 26th ECAADe Conference Proceedings*, 493–500.

Baharlou, E., & Menges, A. (2000). Generative Agent-Based Design Computation, *2*, 165–174.

Brooks, R. A. (1986). A Robust Layered Control System For A Mobile Robot. *IEEE Journal on Robotics and Automation*, *2*(1), 14–23. https://doi.org/10.1109/JRA.1986.1087032

Chen, L. (2012). Agent-based modeling in urban and architectural research: A brief literature review. *Frontiers of Architectural Research*, *1*(2), 166–177. https://doi.org/10.1016/j.foar.2012.03.003

Evans, R. (1995). *The Projective Cast: Architecture and Its Three Geometries*. The MIT Press.

Fischer, A. (2015). Quelea. Retrieved from http://quelea.alexjfischer.com/

Francez, N. (1986). *Fairness*. Springer-Verlag.

Gangolells, M., Casals, M., Gassó, S., Forcada, N., Roca, X., & Fuertes, A. (2009). A methodology for predicting the severity of environmental impacts related to the construction process of residential buildings. *Building and Environment*, *44*(3), 558–571. https://doi.org/10.1016/j.buildenv.2008.05.001

Gerber, D. J., & Pantazis, E. (2016). Design Exploring Complexity in Architectural Shells. *Complexity & Simplicity - Proceedings of the 34th ECAADe Conference*, *1*, 455–464.

Gerber, D. J., Pantazis, E., Marcolino, L., & Heydarian, A. (2015). A Multi Agent Systems for Design Simulation Framework: Experiments with Virtual Physical Social Feedback for Architecture. *Proceedings of the Symposium on Simulation for Architecture & Urban Design*, (April), 205–212. Retrieved from http://dl.acm.org/citation.cfm?id=2873021.2873049

Gerber, D. J., Pantazis, E., & Wang, A. (2017). A multi-agent approach for performance based architecture: Design exploring geometry, user, and environmental agencies in façades. *Automation in Construction*, *76*(January), 45–58. Retrieved from http://dx.doi.org/10.1016/j.autcon.2017.01.001

Gerber, D. J., Shiordia, R., Veetil, S., & Mahesh, A. (2014). Design Agency: Prototyping Multi-agent System Simulation for Design Search and Exploration. *Proceedings of the 2014 Symposium on Simulation for Architecture & Urban Design*. Retrieved from http://dl.acm.org/citation.cfm?id=2664323.2664347

Gilbert, N. (2008). *Agent-based models*. SAGE Publications.

Gruber, P., & Imhof, B. (2017). Patterns of Growth—Biomimetics and Architectural Design. *Buildings*, *7*(2), 32. https://doi.org/10.3390/buildings7020032

Hewitt, C. (1986). Offices are open systems. *ACM Transactions on Information Systems*, *4*(3), 271–287. https://doi.org/10.1145/214427.214432

Jennings, N. R., Sycara, K., & Wooldridge, M. (1998). A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems*, *38*, 7–38. https://doi.org/10.1023/A:1010090405266

Knippers, J., Cremers, J., Gabler, M., & Julian Lienhard. (2011). Construction Manual for Polymers + Membranes, 100–104.

Kolarevic, B. (2003a). *Architecture in the digital age. Design and manufacturing*. Spon Press.

Kolarevic, B. (2003b). Information master builders. In *Architecture in the Digital Age: Design and Manufacturing* (pp. 88–96). Taylor & Francis. https://doi.org/10.1007/s00004-004-0025-4

Krieg, O. D., Schwinn, T., Menges, A., Li, J.-M., Knippers, J., Schmitt, A., & Schwieger, V. (2015). Biomimetic Lightweight Timber Plate Shells: Computational Integration of Robotic Fabrication, Architectural Geometry and Structural Design. In P. Block, J. Knippers, N. J. Mitra, & W. Wang (Eds.), *Advances in Architectural Geometry 2014* (Vol. 1, p. 379). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-11418-7

Macal, C. M., & North, M. J. (2009). Agent-based modeling and simulation. In *Proceedings of the 2009 Winter Simulation Conference (WSC)* (Vol. 53, pp. 86–98). IEEE. https://doi.org/10.1109/WSC.2009.5429318

Maes, P. (1993). Behavior-based Artifical Intelligence. In *From Animals to Animats 2. Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society* (pp. 74–83).

Marcolino, L. S., Xu, H., Gerber, D., Kolev, B., Price, S., Pantazis, E., & Tambe, M. (2015). Multi-agent team formation for design problems. https://doi.org/10.1007/978-3-319-42691-4_20

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, *21*(4), 25–34. https://doi.org/10.1145/37402.37406

Robert McNeel & Associates. (2018). Rhinoceros. Robert McNeel & Associates. Retrieved from https://www.rhino3d.com/

Russell, S., & Norvig, P. (1995). *Artificial Intelligence: a Modern Approach.*

Schumacher, P. (2009). Parametricism: A new global style for architecture and urban design. *Architectural Design*, *79*(4), 14–23. https://doi.org/10.1002/ad.912

Ten Busschen, A., Bouwmeester, J., & Schreuder, P. (2016). *Hergebruik van thermoharde composieten - Onderzoek van afvalproductsoorten*. Zwolle.

Turner, J. S. (2012). Evolutionary Architecture? Some perspectives from biological design. *Architectural Design*, *82*(2), 28–33.

WA Yachting Consultants. (2015). *Number of End of Life Boats (ELB) and waste material flows in the Netherlands.*

Weinand, Y., & Hudert, M. (2010). Timberfabric: Applying Textile Principles on a

Building Scale. *Architectural Design*, *80*(4), 102–107. https://doi.org/10.1002/ad.1113

Wooldridge, M. (2002). An introduction to MultiAgent Sysemts. John Wiley & Sons, Ltd.

Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: theory and practice. *The Knowledge Engineering Review*, *10*(02), 115. https://doi.org/10.1017/S0269888900008122

www.rhino3d.com. (2018). Retrieved from https://docs.mcneel.com/rhino/5

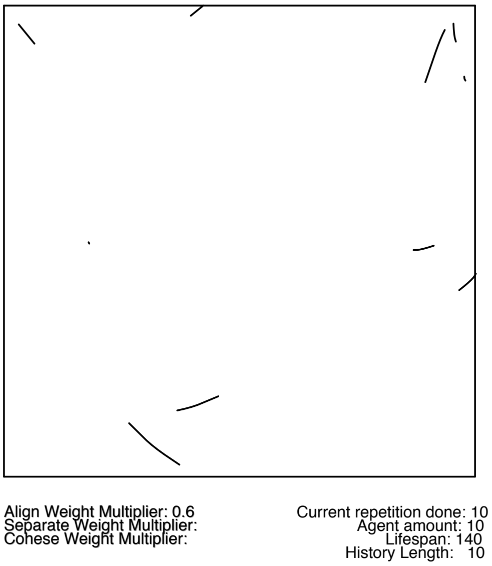
12. List of figures

13. List of tables

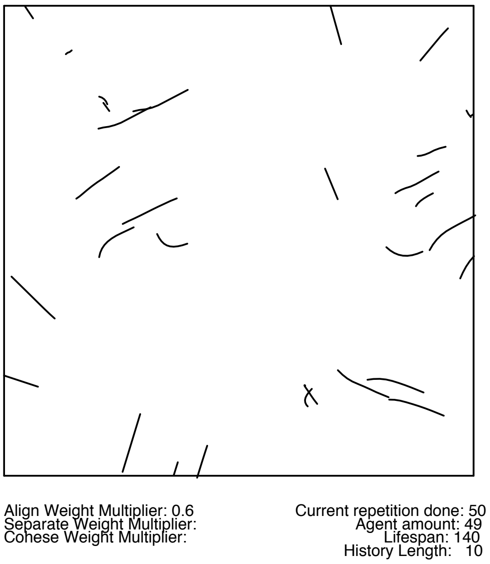
List of tables

14. Appendices

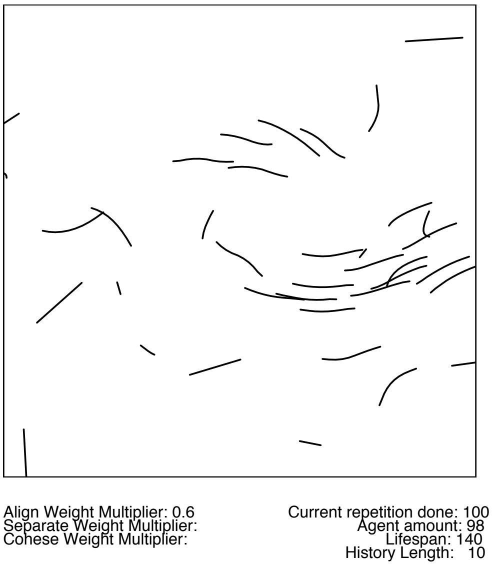
14.1 Appendix A - Agent Behavioural rule set



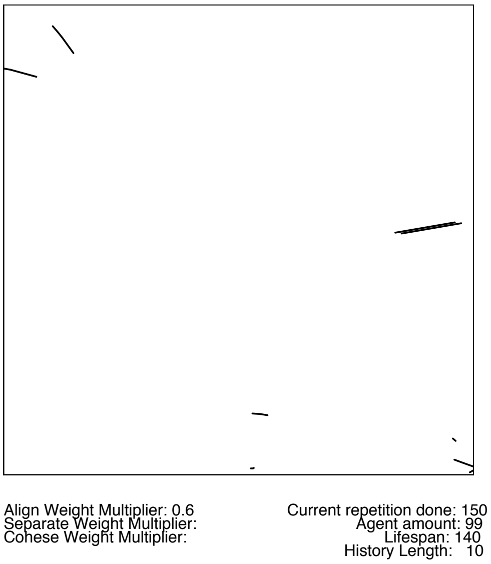
```
Align Weight Multiplier: 0.6          Current repetition done: 10
Separate Weight Multiplier:                     Agent amount: 10
Cohese Weight Multiplier:                        Lifespan: 140
                                            History Length:   10
```



```
Align Weight Multiplier: 0.6          Current repetition done: 50
Separate Weight Multiplier:                     Agent amount: 49
Cohese Weight Multiplier:                        Lifespan: 140
                                            History Length:   10
```

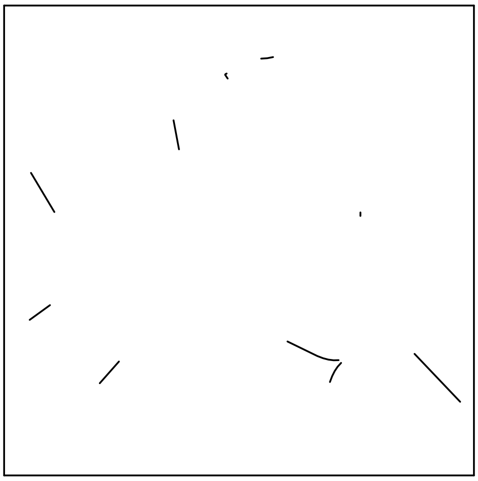


```
Align Weight Multiplier: 0.6         Current repetition done: 100
Separate Weight Multiplier:                     Agent amount: 98
Cohese Weight Multiplier:                        Lifespan: 140
                                            History Length:   10
```



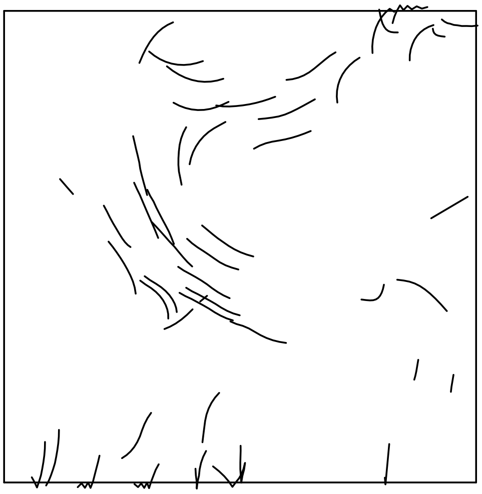
```
Align Weight Multiplier: 0.6         Current repetition done: 150
Separate Weight Multiplier:                     Agent amount: 99
Cohese Weight Multiplier:                        Lifespan: 140
                                            History Length:   10
```

Bounce Contain:	No
Contain:	No
Radius:	5



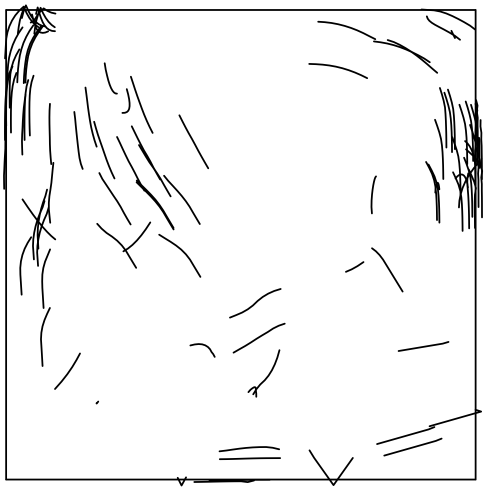
Align Weight Multiplier: 0.6
Separate Weight Multiplier:
Cohese Weight Multiplier:

Current repetition done: 10
Agent amount: 10
Lifespan: 140
History Length: 10



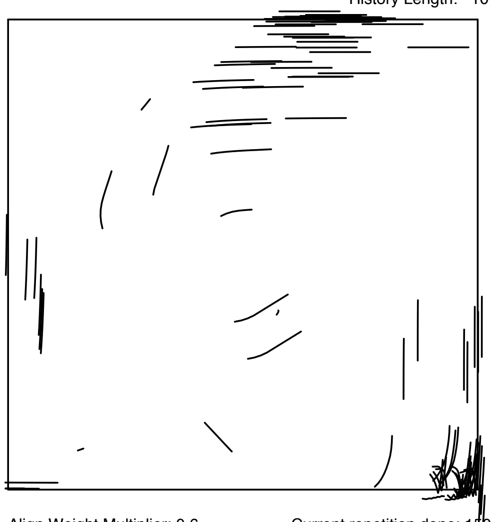
Align Weight Multiplier: 0.6
Separate Weight Multiplier:
Cohese Weight Multiplier:

Current repetition done: 50
Agent amount: 49
Lifespan: 140
History Length: 10



Align Weight Multiplier: 0.6
Separate Weight Multiplier:
Cohese Weight Multiplier:

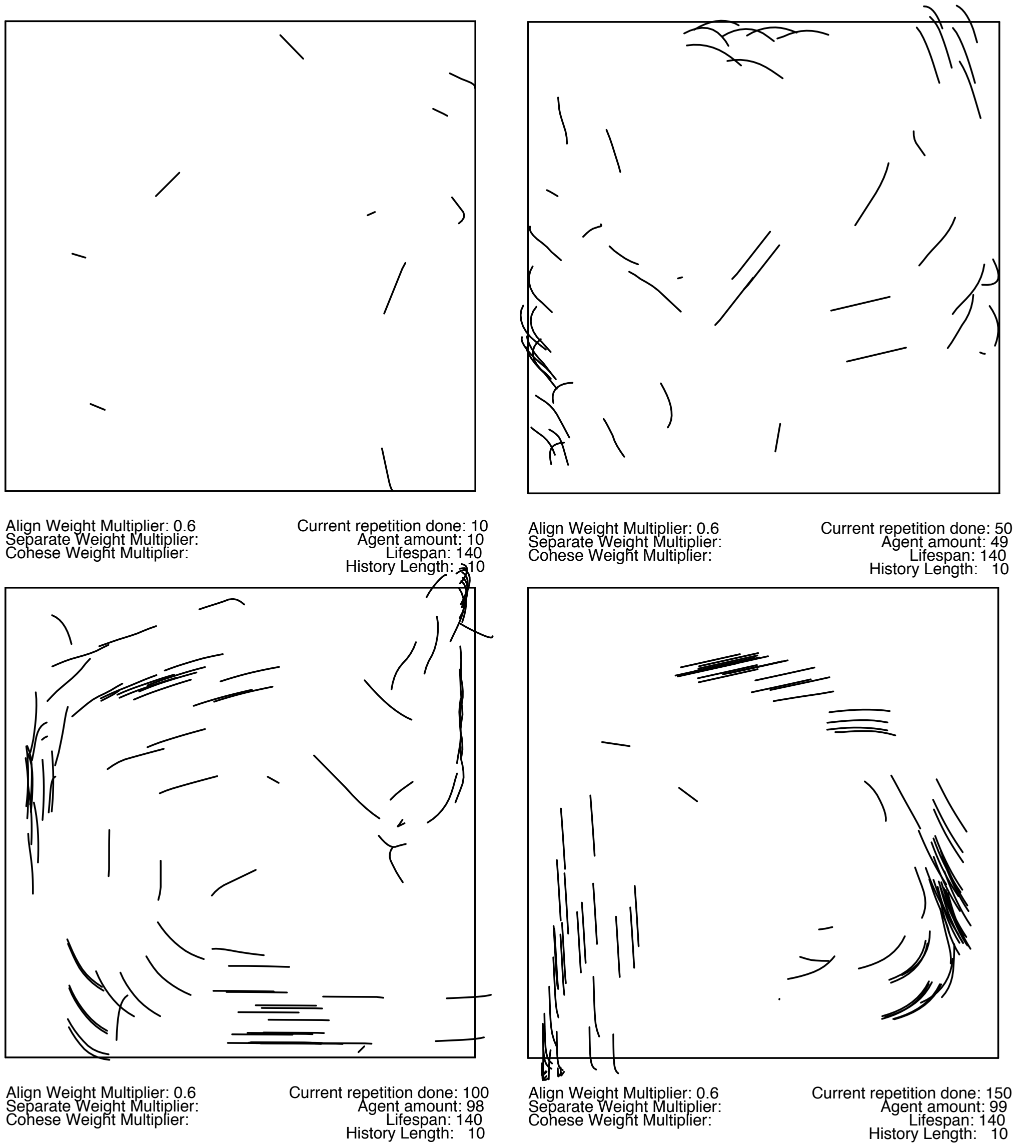
Current repetition done: 100
Agent amount: 98
Lifespan: 140
History Length: 10



Align Weight Multiplier: 0.6
Separate Weight Multiplier:
Cohese Weight Multiplier:

Current repetition done: 150
Agent amount: 99
Lifespan: 140
History Length: 10

Bounce Contain:	Yes
Contain:	No
Radius:	5



Align Weight Multiplier: 0.6
Separate Weight Multiplier:
Cohese Weight Multiplier:

Current repetition done: 10
Agent amount: 10
Lifespan: 140
History Length: 10

Align Weight Multiplier: 0.6
Separate Weight Multiplier:
Cohese Weight Multiplier:

Current repetition done: 50
Agent amount: 49
Lifespan: 140
History Length: 10

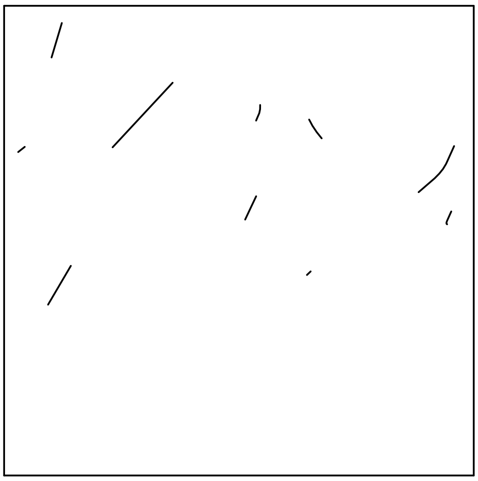
Align Weight Multiplier: 0.6
Separate Weight Multiplier:
Cohese Weight Multiplier:

Current repetition done: 100
Agent amount: 98
Lifespan: 140
History Length: 10

Align Weight Multiplier: 0.6
Separate Weight Multiplier:
Cohese Weight Multiplier:

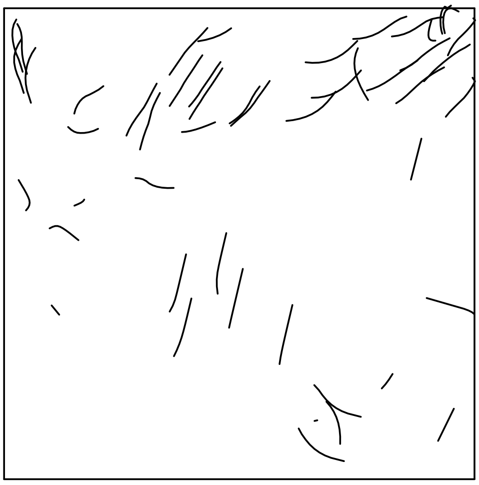
Current repetition done: 150
Agent amount: 99
Lifespan: 140
History Length: 10

Bounce Contain:	No
Contain:	Yes
Radius:	5



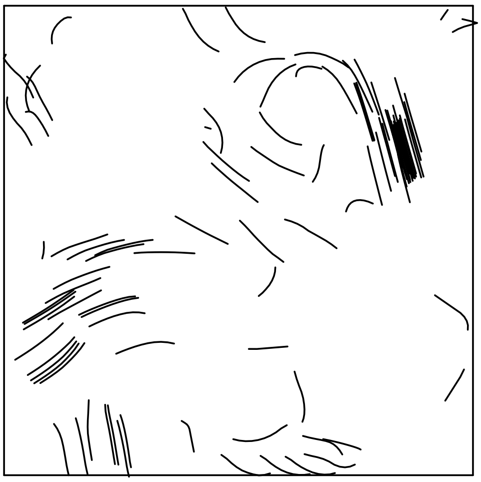
Align Weight Multiplier: 0.6
Separate Weight Multiplier:
Cohese Weight Multiplier:

Current repetition done: 10
Agent amount: 10
Lifespan: 140
History Length: 10



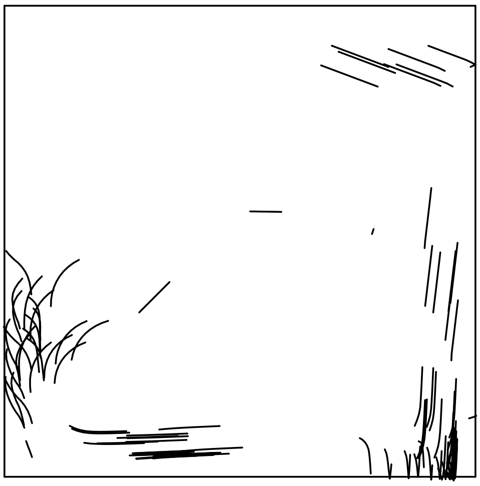
Align Weight Multiplier: 0.6
Separate Weight Multiplier:
Cohese Weight Multiplier:

Current repetition done: 50
Agent amount: 49
Lifespan: 140
History Length: 10



Align Weight Multiplier: 0.6
Separate Weight Multiplier:
Cohese Weight Multiplier:

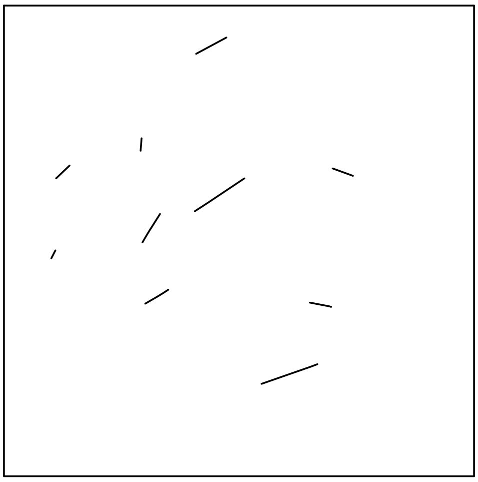
Current repetition done: 100
Agent amount: 98
Lifespan: 140
History Length: 10



Align Weight Multiplier: 0.6
Separate Weight Multiplier:
Cohese Weight Multiplier:

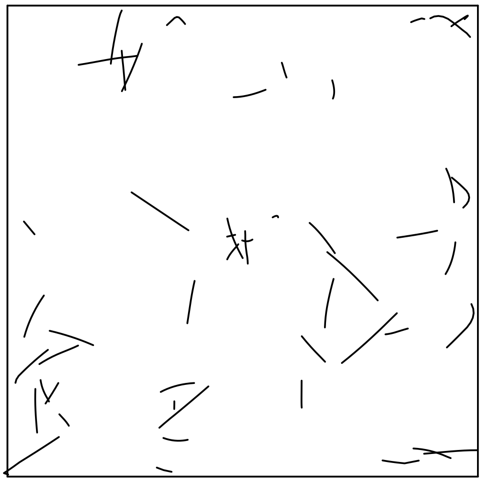
Current repetition done: 150
Agent amount: 99
Lifespan: 140
History Length: 10

Bounce Contain: Yes
Contain: Yes
Radius: 5



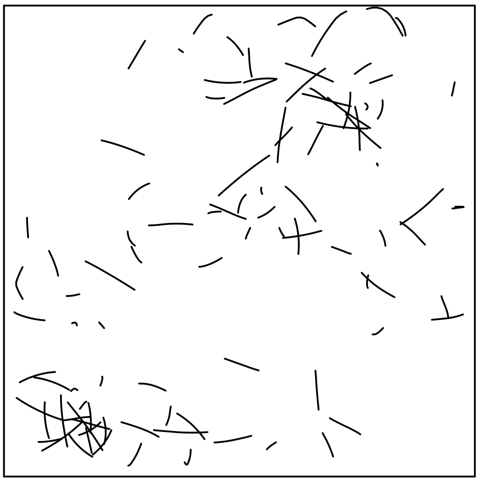
Align Weight Multiplier:
Separate Weight Multiplier:
Cohese Weight Multiplier: 0.15

Current repetition done: 10
Agent amount: 10
Lifespan: 140
History Length: 10



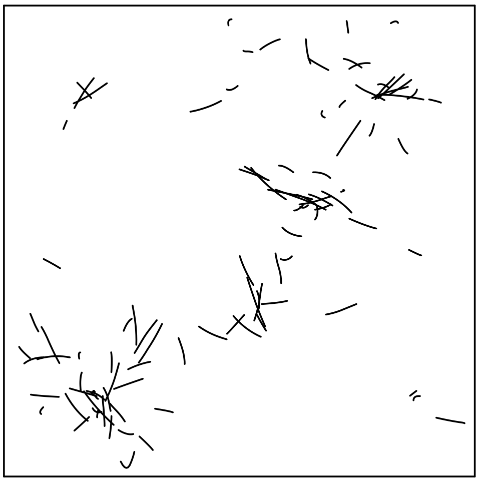
Align Weight Multiplier:
Separate Weight Multiplier:
Cohese Weight Multiplier: 0.15

Current repetition done: 50
Agent amount: 49
Lifespan: 140
History Length: 10



Align Weight Multiplier:
Separate Weight Multiplier:
Cohese Weight Multiplier: 0.15

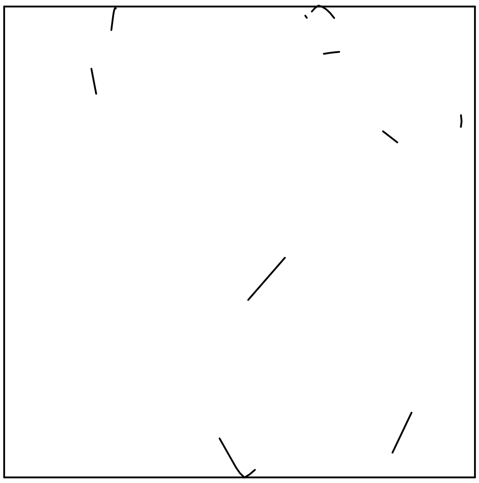
Current repetition done: 100
Agent amount: 98
Lifespan: 140
History Length: 10



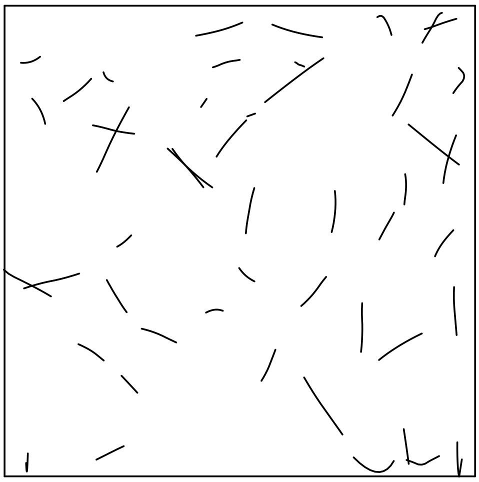
Align Weight Multiplier:
Separate Weight Multiplier:
Cohese Weight Multiplier: 0.15

Current repetition done: 150
Agent amount: 99
Lifespan: 140
History Length: 10

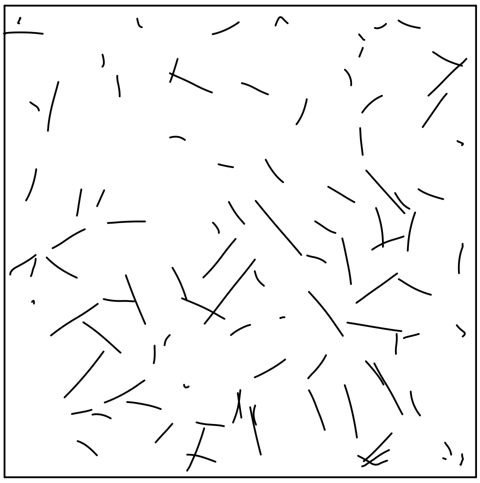
Bounce Contain:	Yes
Contain:	Yes
Radius:	5



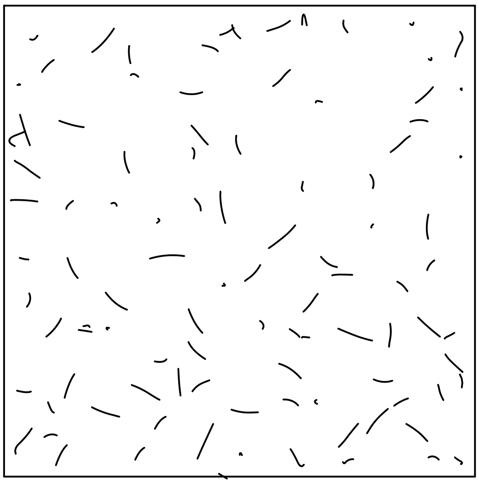
Align Weight Multiplier: Current repetition done: 10
Separate Weight Multiplier: 0.15 Agent amount: 10
Cohese Weight Multiplier: Lifespan: 140
History Length: 10



Align Weight Multiplier: Current repetition done: 50
Separate Weight Multiplier: 0.15 Agent amount: 49
Cohese Weight Multiplier: Lifespan: 140
History Length: 10

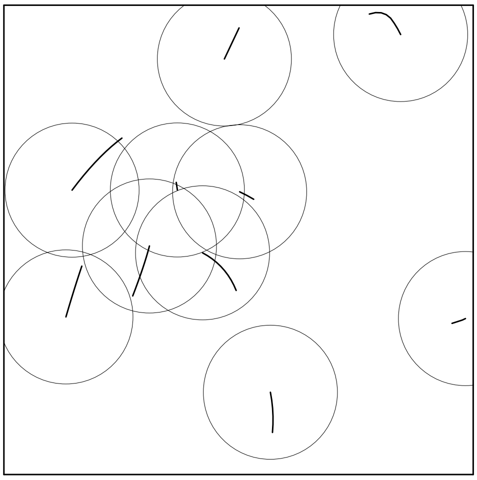


Align Weight Multiplier: Current repetition done: 100
Separate Weight Multiplier: 0.15 Agent amount: 98
Cohese Weight Multiplier: Lifespan: 140
History Length: 10

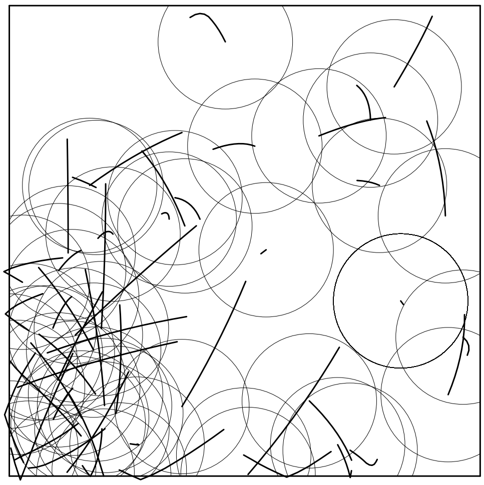


Align Weight Multiplier: Current repetition done: 150
Separate Weight Multiplier: 0.15 Agent amount: 99
Cohese Weight Multiplier: Lifespan: 140
History Length: 10

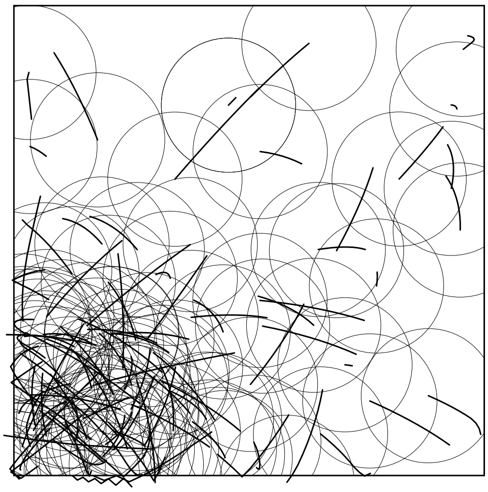
Bounce Contain:	Yes
Contain:	Yes
Radius:	5



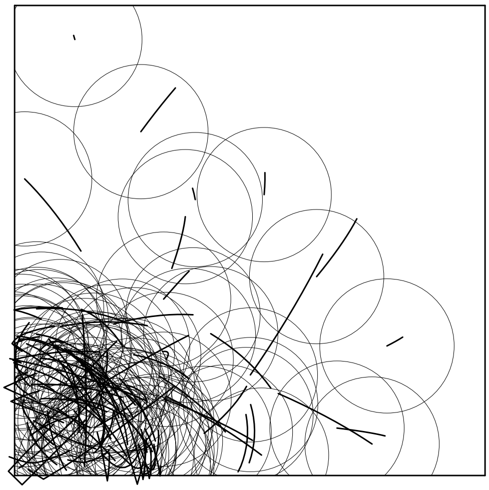
```
Align Weight Multiplier:                Current repetition done: 10
Separate Weight Multiplier:                        Agent amount: 10
Cohese Weight Multiplier:                          Lifespan: 140
Attract Point Multiplier: 0.6                      History Length:   10
```



```
Align Weight Multiplier:                Current repetition done: 50
Separate Weight Multiplier:                        Agent amount: 49
Cohese Weight Multiplier:                          Lifespan: 140
Attract Point Multiplier: 0.6                      History Length:   10
```

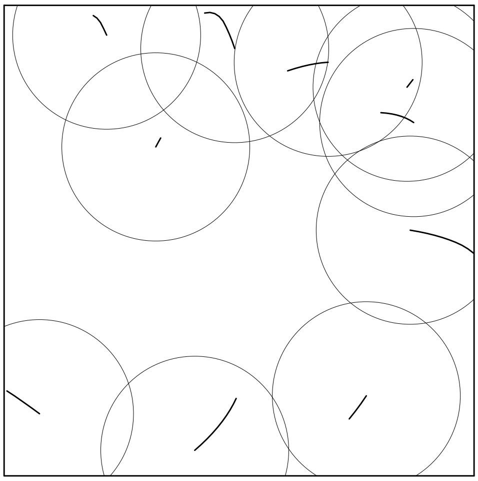


```
Align Weight Multiplier:                Current repetition done: 100
Separate Weight Multiplier:                        Agent amount: 98
Cohese Weight Multiplier:                          Lifespan: 140
Attract Point Multiplier: 0.6                      History Length:   10
```

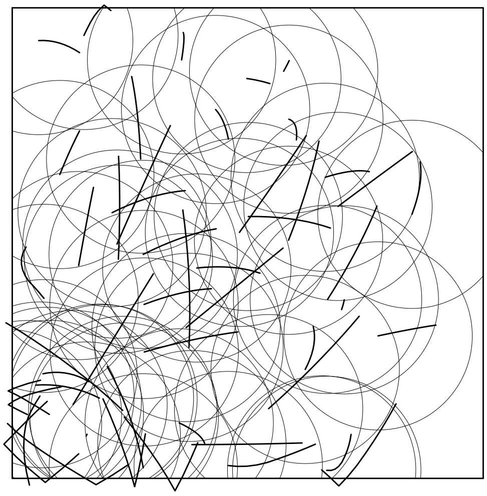


```
Align Weight Multiplier:                Current repetition done: 150
Separate Weight Multiplier:                        Agent amount: 99
Cohese Weight Multiplier:                          Lifespan: 140
Attract Point Multiplier: 0.6                      History Length:   10
```

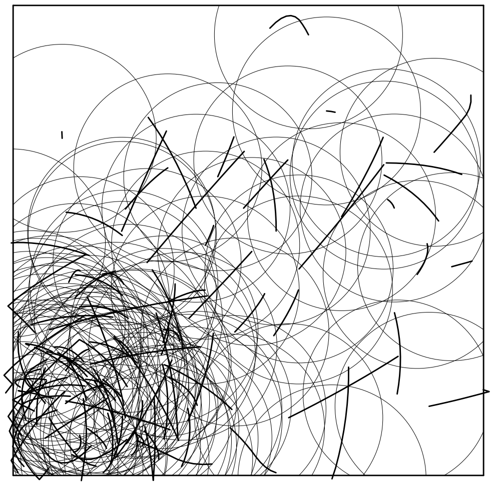
Bounce Contain:	Yes
Contain:	Yes
Radius:	5



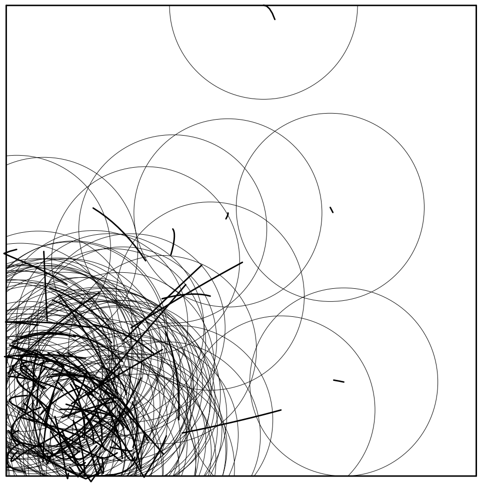
```
Align Weight Multiplier:              Current repetition done: 10
Separate Weight Multiplier:                      Agent amount: 10
Cohese Weight Multiplier:                           Lifespan: 140
Attract Point Multiplier: 0.6                 History Length:   10
```



```
Align Weight Multiplier:              Current repetition done: 50
Separate Weight Multiplier:                      Agent amount: 49
Cohese Weight Multiplier:                           Lifespan: 140
Attract Point Multiplier: 0.6                 History Length:   10
```

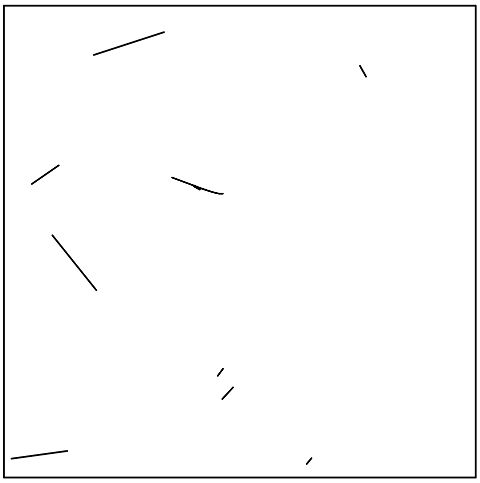


```
Align Weight Multiplier:             Current repetition done: 100
Separate Weight Multiplier:                      Agent amount: 98
Cohese Weight Multiplier:                           Lifespan: 140
Attract Point Multiplier: 0.6                 History Length:   10
```



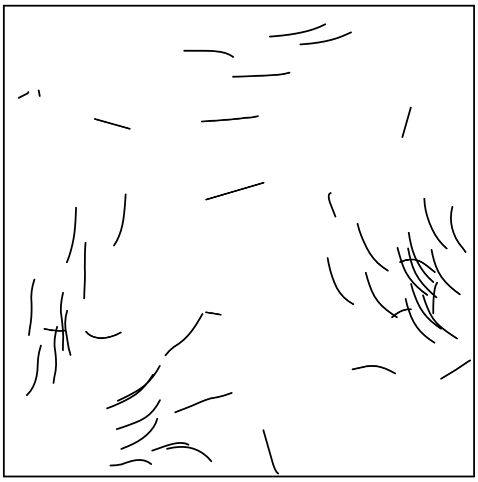
```
Align Weight Multiplier:             Current repetition done: 150
Separate Weight Multiplier:                      Agent amount: 99
Cohese Weight Multiplier:                           Lifespan: 140
Attract Point Multiplier: 0.6                 History Length:   10
```

Bounce Contain:	Yes
Contain:	Yes
Radius:	7



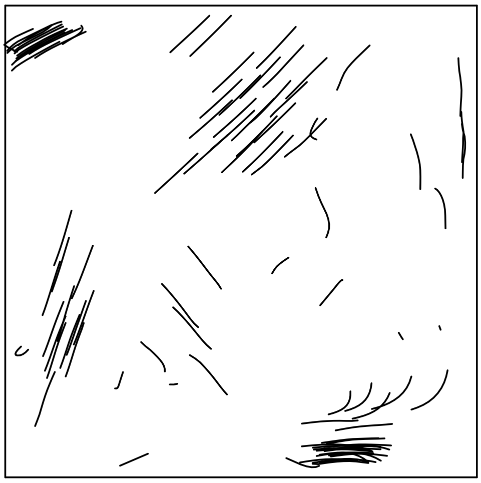
Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15

Current repetition done: 10
Agent amount: 10
Lifespan: 140
History Length: 10



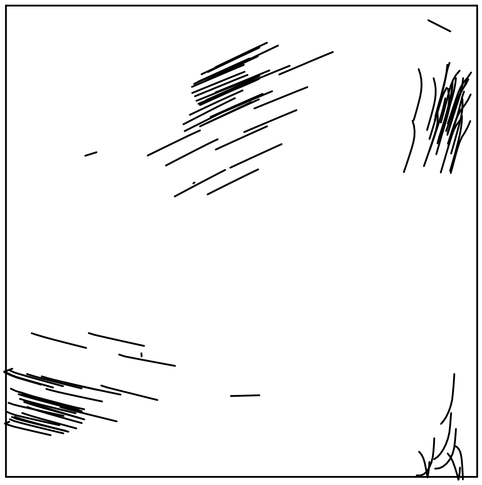
Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15

Current repetition done: 50
Agent amount: 49
Lifespan: 140
History Length: 10



Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15

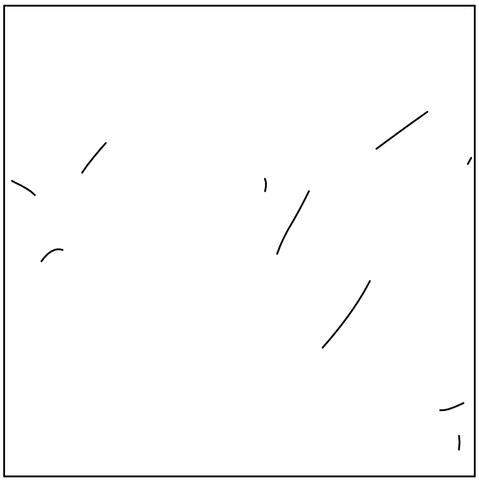
Current repetition done: 100
Agent amount: 98
Lifespan: 140
History Length: 10



Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15

Current repetition done: 150
Agent amount: 99
Lifespan: 140
History Length: 10

Bounce Contain:	Yes
Contain:	Yes
Radius:	5



Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Attract Point Multiplier: 0.6

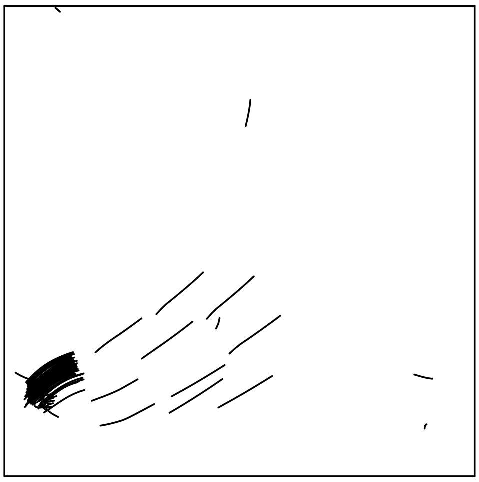
Current repetition done: 10
Agent amount: 10
Lifespan: 140
History Length: 10

Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Attract Point Multiplier: 0.6

Current repetition done: 50
Agent amount: 49
Lifespan: 140
History Length: 10

Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Attract Point Multiplier: 0.6

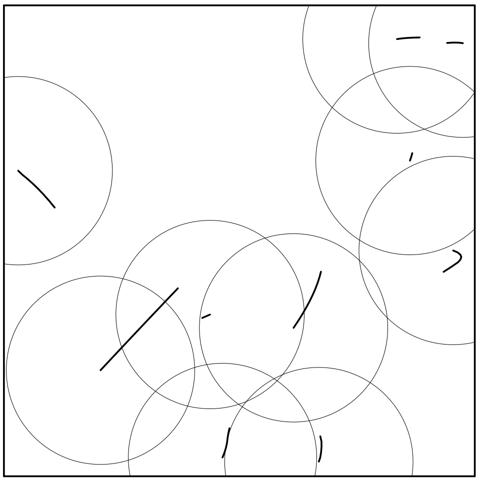
Current repetition done: 100
Agent amount: 98
Lifespan: 140
History Length: 10



Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Attract Point Multiplier: 0.6

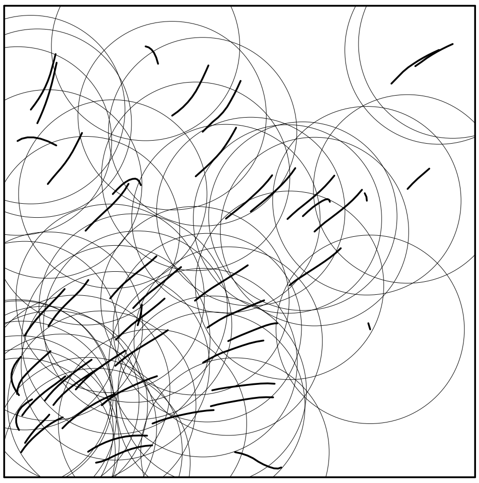
Current repetition done: 150
Agent amount: 99
Lifespan: 140
History Length: 10

Bounce Contain:	Yes
Contain:	Yes
Radius:	5



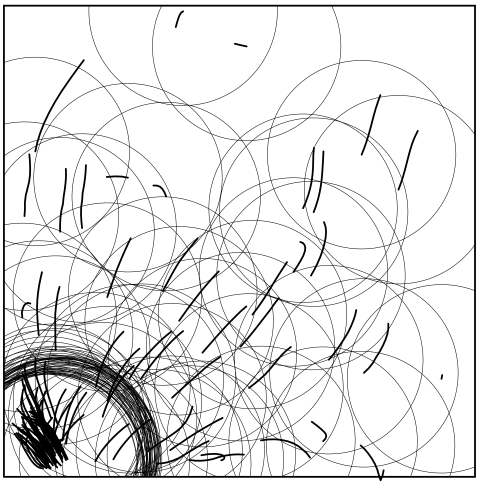
Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Attract Point Multiplier: 0.6

Current repetition done: 10
Agent amount: 10
Lifespan: 140
History Length: 10



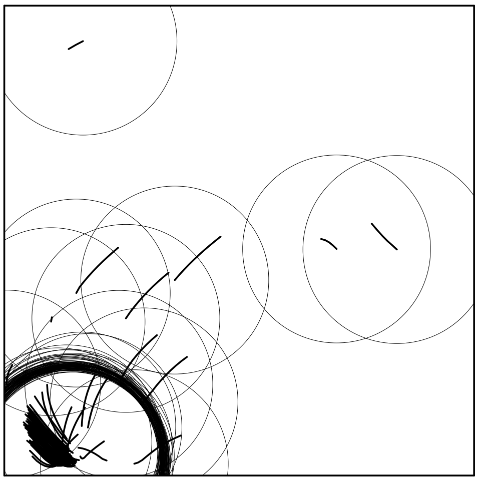
Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Attract Point Multiplier: 0.6

Current repetition done: 50
Agent amount: 49
Lifespan: 140
History Length: 10



Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Attract Point Multiplier: 0.6

Current repetition done: 100
Agent amount: 98
Lifespan: 140
History Length: 10

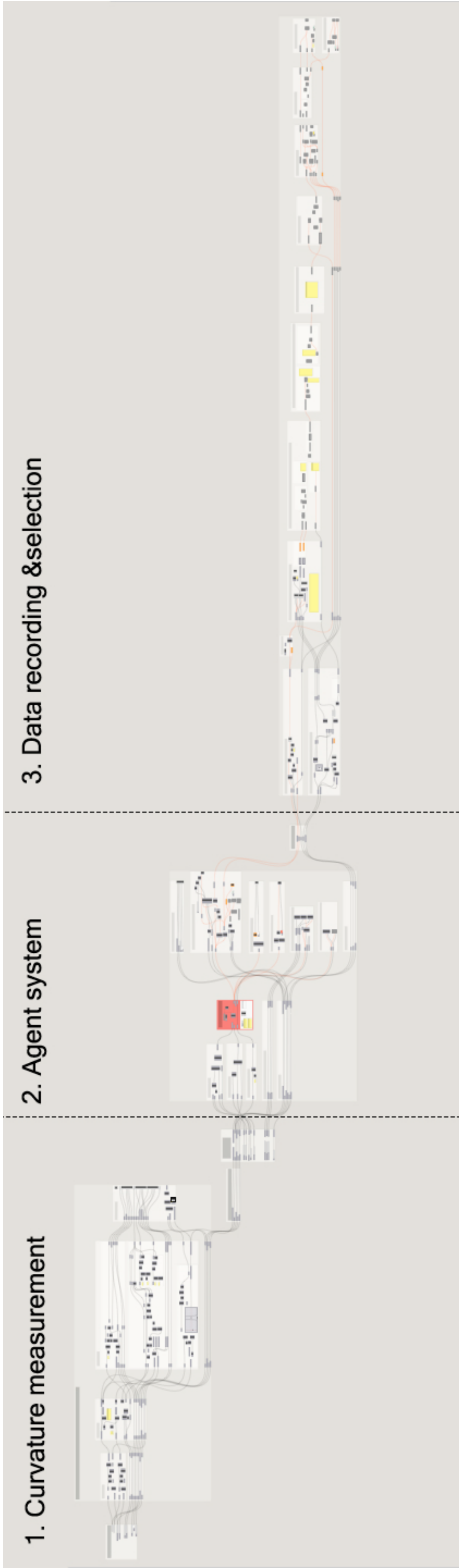


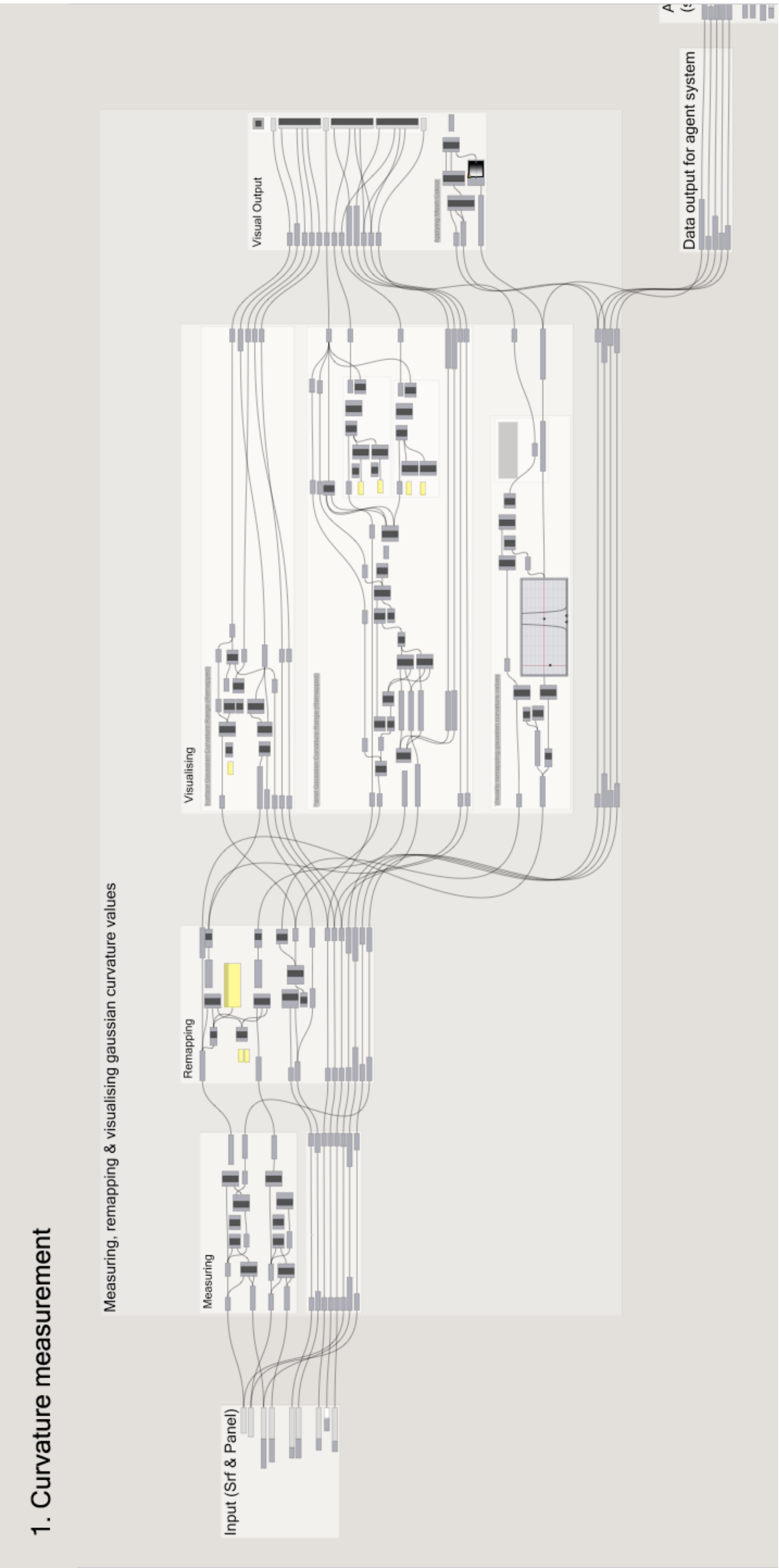
Align Weight Multiplier: 0.6
Separate Weight Multiplier: 0.15
Cohese Weight Multiplier: 0.15
Attract Point Multiplier: 0.6

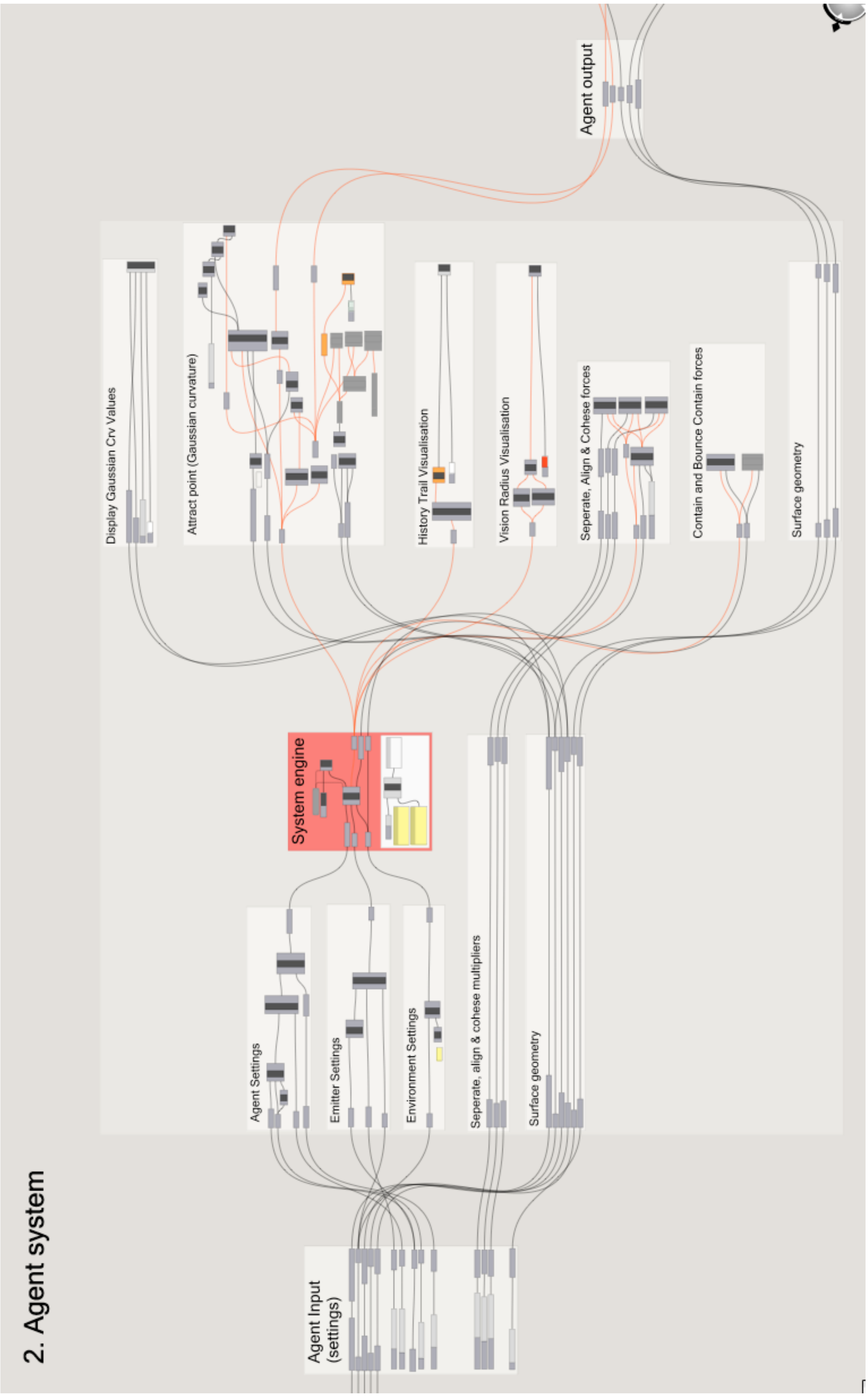
Current repetition done: 150
Agent amount: 99
Lifespan: 140
History Length: 10

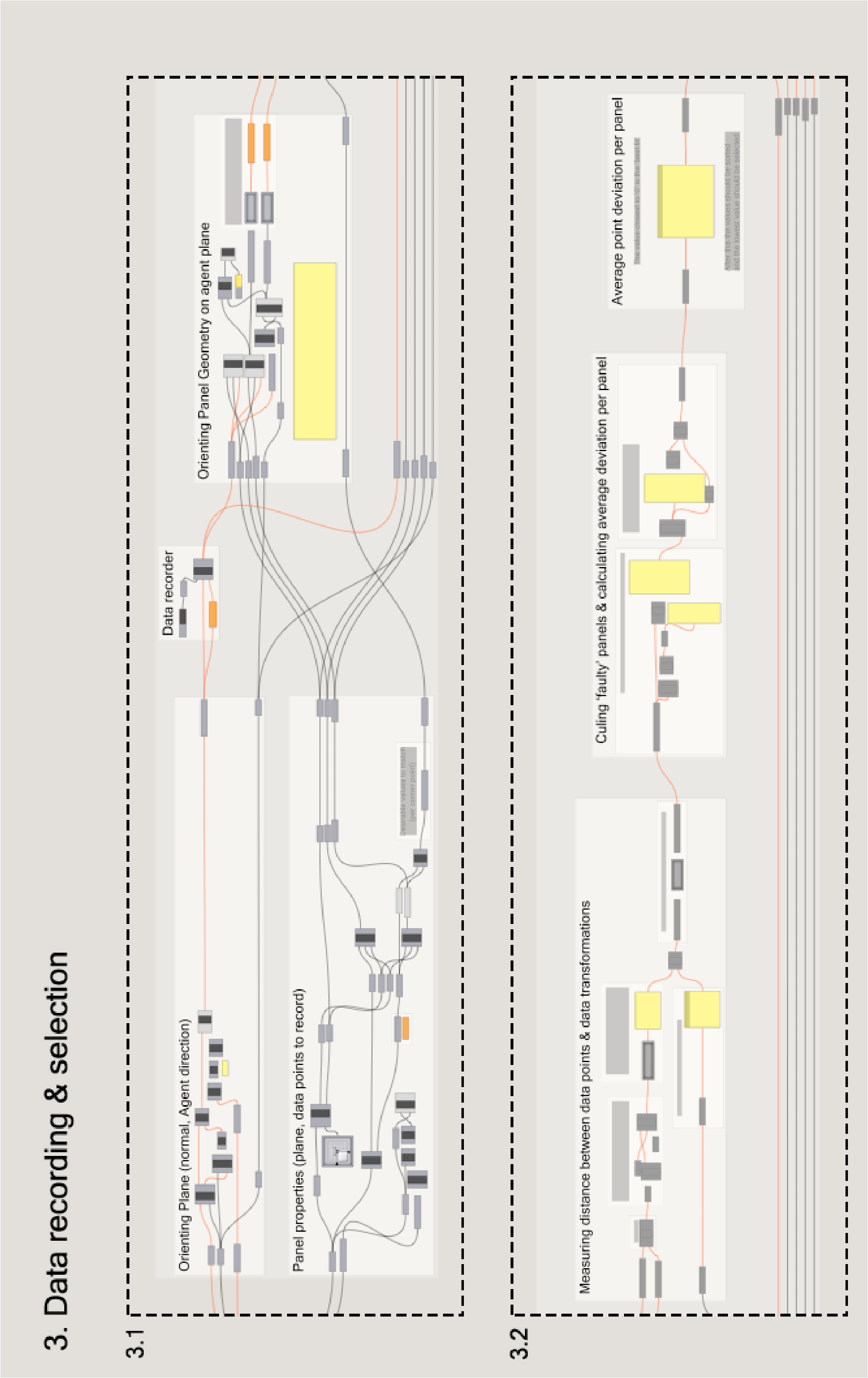
Bounce Contain:	Yes
Contain:	Yes
Radius:	7

14.2 Appendix B - Grasshopper Script



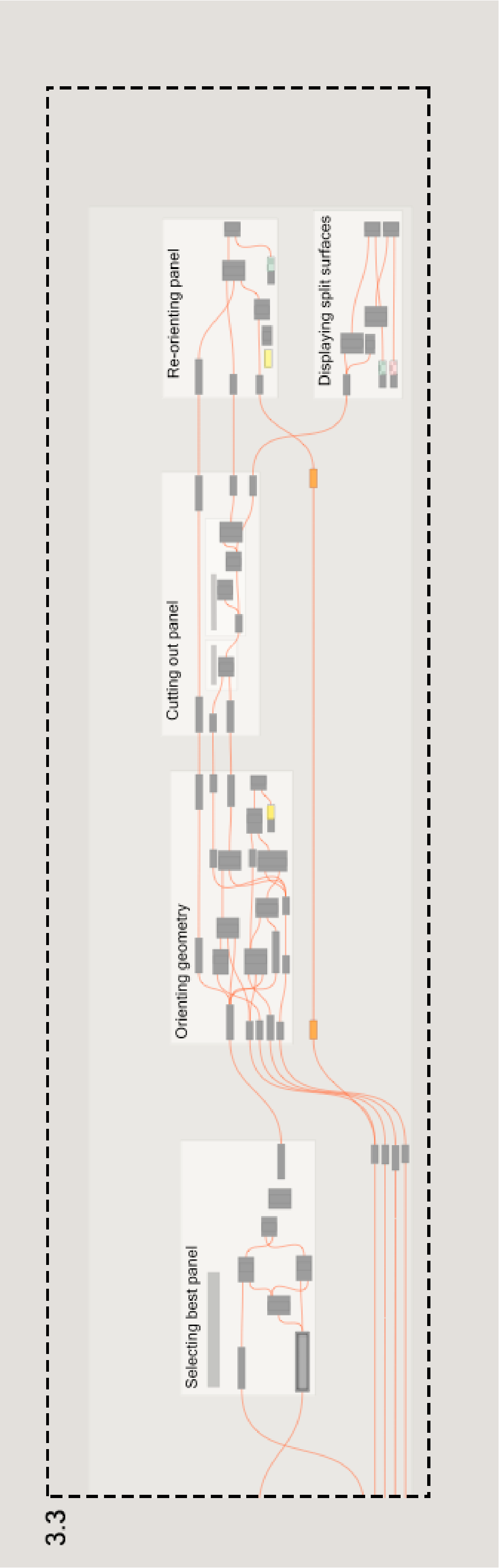

1. Curvature measurement
Measuring, remapping & visualising gaussian curvature values
Input (Srf & Panel)
Measuring
Remapping
Visualising
Visual Output
Data output for agent system


2. Agent system
Agent Input (settings)
Agent Settings
Emitter Settings
Environment Settings
Seperate, align & cohese multipliers
Surface geometry
System engine
Display Gaussian Crv Values
Attract point (Gaussian curvature)
History Trail Visualisation
Vision Radius Visualisation
Seperate, Align & Cohese forces
Contain and Bounce Contain forces
Surface geometry
Agent output



3. Data recording & selection

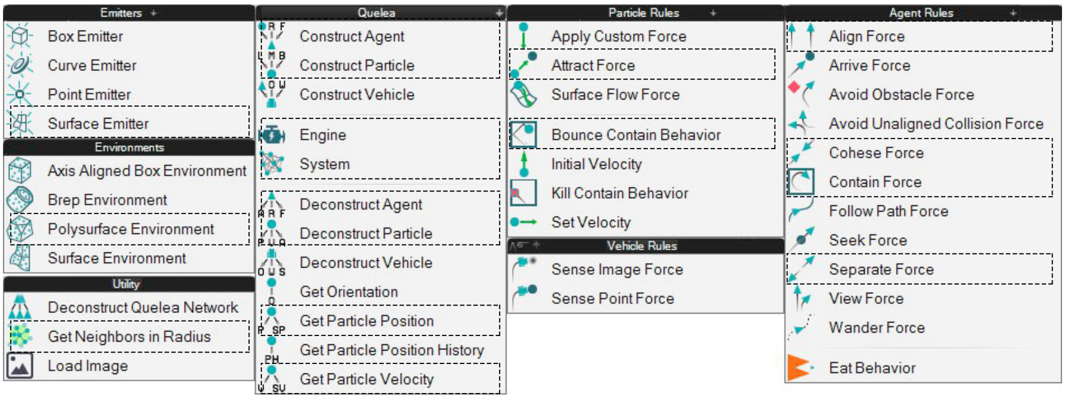
3.1
Orienting Plane (normal, Agent direction)
Panel properties (plane, data points to record)
Data recorder
Orienting Panel Geometry on agent plane
3.2
Measuring distance between data points & data transformations
Culing 'faulty' panels & calculating average deviation per panel
Average point deviation per panel


3.3
Re-orienting panel
Displaying split surfaces
Cutting out panel
Orienting geometry
Selecting best panel

14.3 Appendix C - Grasshopper plug-in: 'Quelea'

The plug-in used in this research for creating agent behaviour is called Quelea.
It is a plug-in for Grasshopper and is developed by Alex Fischer.
The components used in this research are highlighted and further explained within this appendix.

14.3.1 *Component overview*



14.3.2 *Forces*

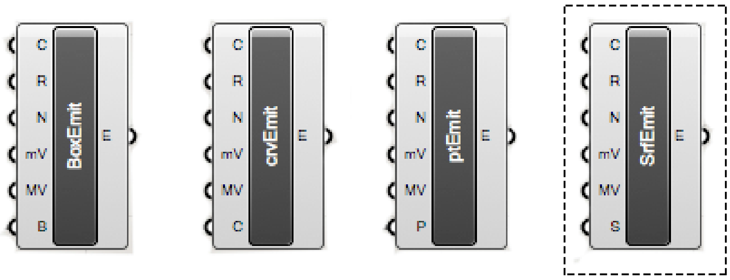
Forces calculate a vector to apply to the quelea's acceleration causing their position to be changed over time.
Particle Forces are divided by the particle's mass.
Agent Forces are converted to steering forces that cause the agent to turn to make forward movement.
Vehicle Forces change the angular velocity of the wheels that cause the vehicle to turn when the velocities differ.

14.3.3 *Behaviours*

Behaviours change the state of the quelea. It has quelea.
For instance, the Bounce Contain Behaviour sets the quelea's velocity to be the reflection of its current velocity about the normal of the border of its environment. This differs from the Contain Force because the force calculate a vector and give it to the quelea to choose how to interpret it, ultimately adding the result to the quelea's future acceleration. Behaviour directly sets a field in the quelea's state whereas a force indirectly updates the quelea's position.

14.3.4 *Emitters*

Emitters create agents for every time step that the algorithm runs. When continuously emitting agents they can be given a set lifespan. The agent's lifespan is defined by an amount of time steps. The lifespan setting can also be negative, indicating an infinite lifespan. A limit can be set to the total amount of agents that are allowed to live in the system at any time when using the infinite lifespan setting. This might negate effects such as an overload of the system in case of lacking computation power.



- **Box Emitter:** Randomly emits one or more agents in the inside of a box shape
- **Curve Emitter:** Randomly emits one or more agents from a curve
- **Point Emitter:** Emits one or more agents from a given point
- **Surface Emitter:** A surface from which quelea can be emitted.

  Input parameters:

  - Continuous Flow **(C)** (Boolean):

    Boolean toggle. If true, particles will be emitted every Rth time step. If false, N particles will be emitted once.

  - Creation Rate **(R)** (integer)**:**

    Rate at which new Quelea are created. Every **R**th time step.

  - Number of Quelea (**N)** (integer):

    The number of Quelea that are allowed to be alive in the system at once.

  - Minimum Initial Velocity **(mV)** (vector):

    The minimum initial velocity from which a random value will be taken.

  - Maximum Initial Velocity **(MV)** (vector):

    The maximum initial velocity from which a random value will be taken.

  - Surface **(S)** (surface):

    Surface for emitter.

Output parameters:

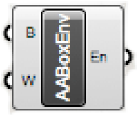
- Emitter **(E)** (generic data):

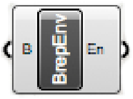
  The geometry from which quelea will be spawned.

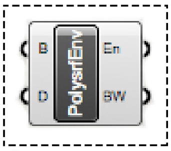
14.3.5 Environments

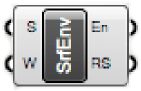
Environments within grasshopper can be referenced surfaces, boxes, boundary representations or polysurfaces. Apart from this a custom direction can be given to the surface that is created at the edge of environments with an edge.

Axis Aligned Box Environment
Brep Environment
Polysurface Environment: A 3D polysurface environment.
Input Parameters:
Brep(B) (brep):
A closed Brep with normal facing out.
Border Extrusion Direction(D) (vector):
A vector indicating which direction to extrude the borders of the polysurface to create border walls for containment.If the zero vector is supplied, the default is to extrude each border point normal to the surface.
Output Parameters:
Environment (En) (generic data):
A geometrical object representing the environment the quelea will interact with.
Boundary Walls (BW) (generic data):
The walls that represent the boundary of the polysurface.
Surface Environment



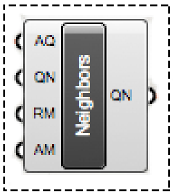


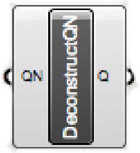


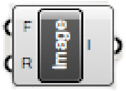


14.3.6 *Utility*

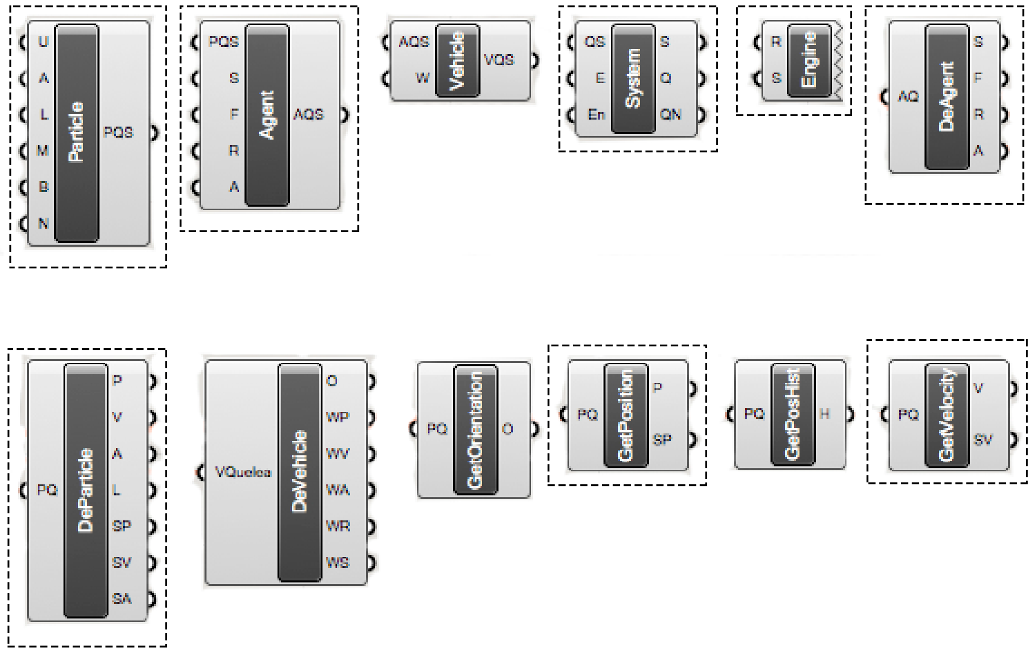
Get Neighbours In Radius: Gets the neighbours of an agent within a specified radius.
Input parameters:
Agent Quelea (AQ) (generic data):
The agent to get neighbours for.
Quelea Network (QN) (generic data):
The quelea network to search through.
Vision Radius Multiplier (RM) (generic data):
The factor by which the agent's vision radius will be multiplied. The result will determine how far out the agent will see other neighbours.
Vision Angle Multiplier (AM) (generic data):
The factory by which the agent's vision angle will be multiplied. The result will be used to determine the angle from the velocity that the agent will be able to see neighbours.
Output parameters:
Quelea Network (QN) (generic data):
The neighbours of the agent.
Deconstruct Quelea Network
Load Image







14.3.7 *Quelea*



- Construct **Particle**

  Input parameters:

  - Up Direction **(U)** (vector):

    The up direction for the calculation of the initial orientation.

  - Acceleration **(A)** (vector)**:**

    The vector of the quelea's acceleration.

  - Lifespan **(L)** (integer)**:**

    Number of time steps that the quelea will be alive for. If negative, life span will be infinite.

  - Mass **(M)** (number):

    Affects how strongly the quelea reacts to forces. Larger masses will lead to more cumbersome movement.

  - Body Size **(B)** (number)**:**

    The diameter of the extent of the quelea's bounds. This is used for collision detection among other things.

- History Length **(H)** (integer)**:**

  The number of past positions to remember for each quelea.

Output parameters:

- Particle Quelea Settings **(PQS)** (generic data):

  A particle is a point that can be affected by forces that do not require a perception of its surroundings or used to supply settings for an agent.

- Construct **Agent**

  Input parameters:

  - Particle Quelea Settings **(PQS)** (generic data):

    A particle is a point that can be affected by forces that do not require a perception of its surroundings or used to supply settings for an agent.

  - Max Speed **(S)** (number):

    Rather than teleporting, the quelea will move incrementally by this speed towards targets that it seeks.

  - Max Force **(F)** (number):

    Steering ability can be controlled by limiting the magnitude of the steering force.

  - Vision Radius **(R)** (number):

    The maximum radius around the agent that it can see.

  - Vision Angle **(A)** (number):

    The maximum angle, taken form the velocity vector, that the agent can see around it.

  Output parameters:

  - Agent Quelea Settings **(AQS)** (generic data):

    An autonomous agent that is an extension of a particle and can perceive its surroundings and make decisions on how to act.

- Construct **Vehicle**

- **System:** Represents a self-contained system of quelea and emitters.

  Input parameters:

  - Quelea Settings **(QS)** (generic data):

    The settings for your particle, agent, or vehicle; collectively referred to as 'quelea'

- Emitters **(E)** (generic data):

  Emitters from which quelea will be spawned.

- Environment **(En)** (generic data):

  Restricts an agent's position to be contained within the environment. This is most useful for surface and polysurface environments. For brep environments, consider using the contain rules as they will run much faster.

Output parameters:

- System **(S)** (generic data):

  Represents a self-contained system of quelea and emitters.

- Quelea **(Q)** (generic data):

  A quelea which can be a particle, agent, or vehicle.

- Quelea Network **(QN)** (generic data):

  A container object of quelea to provide for fast lookup of neighbours.

- **Engine:** Engine that runs the simulation.

  Input parameters:

  - Reset **(R)** (Boolean):

    Reset the simulation.

  - System **(S)** (generic data):

    Represents a self-contained system of quelea and emitters.

- **Deconstruct Agent:** Deconstructs an agent to expose its fields such as max speed, max force, and vision radius. Use Deconstruct Particle to expose particle fields such as position.

  Input parameters:

  - Agent Quelea **(AQ)** (generic data)

    An autonomous agent that is an extension of a particle and can perceive its surroundings and make decisions on how to act.

  Output parameters:

  - Max speed **(S)** (integer):

    Rather than teleporting, the quelea will move incrementally by this speed towards targets that it seeks.

  - Max force **(F)** (integer):

Steering ability can be controlled by limiting the magnitude of the steering force.

- Vision radius **(R)** (integer):

  The maximum radius around the agent that it can see.

- Vision angle **(A)** (integer):

  The maximum angle, taken from the velocity vector, that the agent can see around it.

- **Deconstruct Particle:** Deconstructs a particle to expose its fields such as position, velocity, and acceleration.

  Input parameters:

  - Particle Quelea **(PQ)** (generic data):

    A particle is a point that can be affected by forces that do not require a perception of its surroundings or used to supply settings for an agent.

  Output parameters:

  - Position **(P)** (point):

    The point 3d position of the agent.

  - Velocity **(V)** (vector):

    The direction and magnitude of the quelea's movement.

  - Acceleration **(A)** (vector):

    The vector of the quelea's acceleration.

  - Lifespan **(L)** (integer):

    Number of time steps that the quelea will be alive for. If negative, lifespan will be infinite.

  - Surface position **(SP)** (point):

    For particles bound to surface and polysurface environments, the position of the agent mapped to a 2D plane representing the bounds of the surface.

  - Surface velocity **(SV)** (vector):

    For particles bound to surface and polysurface environments, the velocity of the agent mapped to a 2D plane representing the bounds of the surface.

  - Surface acceleration **(SA)** (vector):

    For particles bound to surface and polysurface environments, the acceleration of the agent mapped to a 2D plane representing the bounds of the surface.

- **Deconstruct Vehicle**
- **Get Orientation**
- **Get** Particle **Position**

  Input parameters:

  - Particle Quelea **(PQ)** (generic data):

    A particle is a point that can be affected by forces that do not require a perception of its surroundings or used to supply settings for an agent.

  Output parameters:

  - Position **(P)** (point):

    The point 3d position of the agent.

  - Surface position **(SP)** (point):

    For particles bound to surface and polysurface environments, the position of the agent mapped to a 2D plane representing the bounds of the surface.

- **Get** Particle **Position History**
- **Get** Particle **Velocity**

  Input parameters:

  - Particle Quelea **(PQ)** (generic data):

    A particle is a point that can be affected by forces that do not require a perception of its surroundings or used to supply settings for an agent.

  Output parameters:

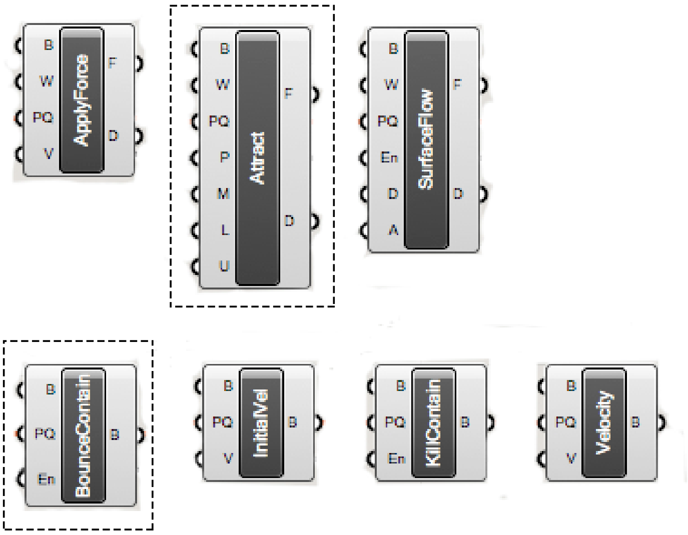
  - Velocity **(V)** (vector):

    The direction and magnitude of the quelea's movement.

  - Surface velocity **(SV)** (vector):

    For particles bound to surface and polysurface environments, the velocity of the agent mapped to a 2D plane representing the bounds of the surface.

14.3.8 *Particle rules*



- **Apply** Custom **Force**
- **Attract** Force: Attracts quelea within the radius of the point.

  Input parameters:

  - Apply? **(B)** (Boolean):

    If false, the force will not be applied to the quelea. This is useful for having behaviours override forces. Can also be used for only applying the force if the quelea is within a certain area.

  - Weight multiplier **(W)** (number):

    The factor by which the force will be multiplied.

  - Particle Quelea **(PQ)** (generic data):

    A particle is a point that can be affected by forces that do not require a perception of its surroundings or used to supply settings for an agent.

  - Target Point **(P)** (generic data):

    Point to be attracted to.

  - Mass **(M)** (number):

    More massive attractors will exert a stronger attraction force than smaller ones.

  - Distance Lower Limit **(L)** (number):

    The lower limit of distance by which the strength is divided.

  - Distance Upper Limit **(U)** (number):

    The upper limit of distance by which the strength is divided.

  Output parameters:

  - Force **(F)** (generic data):

    The resulting force vector for debugging purposes.

  - Desired Velocity **(D)** (generic data):

    The calculated desired velocity of this rule before it is applied to the quelea. Supplied for debugging and visualization purposes.

- **Surface Flow** Force
- **Bounce Contain** Behaviour: Causes particles to bounce off environment boundaries.

  Input parameters:

  - Apply? **(B)** (Boolean):

    If false, the force will not be applied to the quelea. This is useful for having behaviours override forces. Can also be used for only applying the force if the quelea is within a certain area.

  - Particle Quelea **(B)** (generic data):

    A particle is a point that can be affected by forces that do not require a perception of its surroundings or used to supply settings for an agent.

  - Environment **(En)** (generic data):

    The environment which to bounce off of.

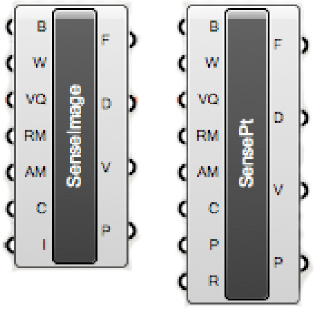
  Output parameters:

  - Behaviour applied **(B)** (Boolean):

    True if the behaviour was applied to the quelea.

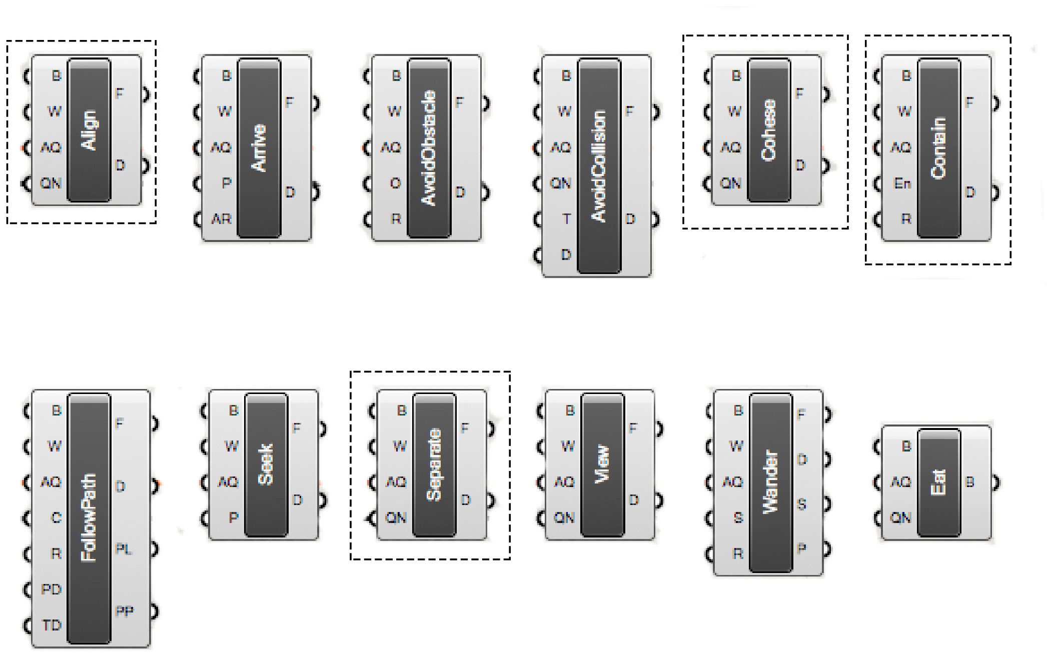
- **Initial Velocity**
- **Kill Contain** Behaviour
- Set **Velocity**

14.3.9 *Vehicle rules*



Sense Image Force
Sense Point Force

14.3.10 Agent rules



- **Align** Force: Steer towards average heading of neighbours

  Input parameters:

  - Apply? **(B)** (Boolean):

    If false, the force will not be applied to the quelea. This is useful for having behaviours override forces. Can also be used for only applying the force if the quelea is within a certain area.

  - Weight multiplier **(W)** (number):

    The factor by which the force will be multiplied.

  - Agent quelea **(AQ)** (generic data):

    An autonomous agent that is an extension of a particle and can perceive its surroundings and make decisions on how to act.

  - Quelea network **(QN)** (generic data):

    The neighbours to react to. Use the 'Get Neighbours in Radius' component.

  Output parameters:

- Force **(F)** (generic data):

  The resulting force vector for debugging purposes

- Desired velocity **(D)** (generic data):

  The calculated desired velocity of this rule before it is applied to the quelea. Supplied for debugging and visualization purposes.

- **Arrive** Force: Steer towards target point, slow down and stop as it approaches the target
- **Avoid Obstacle** Force: Steer away if about to intersect (hit) an object
- **Avoid** Unaligned **Collision** Force: Steer away from predicted potential collision
- **Cohese** Force: Steer to move toward the average position of neighbours.

  Input parameters:

  - Apply? **(B)** (Boolean):

    If false, the force will not be applied to the quelea. This is useful for having behaviours override forces. Can also be used for only applying the force if the quelea is within a certain area.

  - Weight multiplier **(W)** (number):

    The factor by which the force will be multiplied.

  - Agent quelea **(AQ)** (generic data):

    An autonomous agent that is an extension of a particle and can perceive its surroundings and make decisions on how to act.

  - Quelea network **(QN)** (generic data):

    The neighbours to react to. Use the 'Get Neighbours in Radius' component.

  Output parameters:

  - Force **(F)** (generic data):

    The resulting force vector for debugging purposes

  - Desired velocity **(D)** (generic data):

    The calculated desired velocity of this rule before it is applied to the quelea. Supplied for debugging and visualization purposes.

- **Contain** Force: Applies a force to keep agents away from environment boundaries.

  Input parameters:

  - Apply? **(B)** (Boolean):

    If false, the force will not be applied to the quelea. This is useful for having behaviours override forces. Can also be used for only applying the force if the quelea is within a certain area.

  - Weight multiplier **(W)** (number):

    The factor by which the force will be multiplied.

  - Agent quelea **(AQ)** (generic data):

    An autonomous agent that is an extension of a particle and can perceive its surroundings and make decisions on how to act.

  - Quelea network **(QN)** (generic data):

    The neighbours to react to. Use the 'Get Neighbours in Radius' component.

  - Vision Radius Multiplier **(R)** (number):

    The factor by which the agent's vision radius will be multiplied. The result will determine how far out the agent will see other neighbours.

  Output parameters:

  - Force **(F)** (generic data):

    The resulting force vector for debugging purposes

  - Desired velocity **(D)** (generic data):

    The calculated desired velocity of this rule before it is applied to the quelea. Supplied for debugging and visualization purposes.

- **Follow Path** Force: Steer towards path (curve) within radius
- **Seek** Force:
- **Separate** Force: Applies a force to steer to avoid neighbours.

  Input parameters:

  - Apply? **(B)** (Boolean):

    If false, the force will not be applied to the quelea. This is useful for having behaviours override forces. Can also be used for only applying the force if the quelea is within a certain area.

  - Weight multiplier **(W)** (number):

    The factor by which the force will be multiplied.

  - Agent quelea **(AQ)** (generic data):

An autonomous agent that is an extension of a particle and can perceive its surroundings and make decisions on how to act.

- Quelea network **(QN)** (generic data):

  The neighbours to react to. Use the ‘Get Neighbours in Radius’ component.

Output parameters:

- Force **(F)** (generic data):

  The resulting force vector for debugging purposes

- Desired velocity **(D)** (generic data):

  The calculated desired velocity of this rule before it is applied to the quelea. Supplied for debugging and visualization purposes.

- **View** Force: Steer away laterally if any agent is in it’s view (radius)
- **Wander** Force: Steer towards new direction, based on previous direction
- **Eat** Behaviour: Eliminates agents within radius.