Synthesis of 1-DOF hinging rigid panel structure kinematic transmissions using evolutionary algorithms.

Christian Bijvoets

Master of Science Thesis

TU Delft Delft University of Technology

Synthesis of 1-DOF hinging rigid panel structure kinematic transmissions using evolutionary algorithms.

Christian Bijvoets

Precision and Microsystems Engineering
Delft University of Technology
Leeghwaterstraat, 2628 CN Delft
email: c.j.p.bijvoets@student.tudelft.nl

For the degree of Master of Science at Delft University of Technology

Faculty of Mechanical, maritime and Materials Engineering (3mE) – Delft University of Technology

Table of Contents

Abstract

Origami structures are found in mechanical technology more and more often. A less strict derivative of origami are Hinging Rigid Panel Structures, which are not bound to be developable from a single flat sheet of planar material. These Hinging Rigid Panel Structures can be applied to many fields of mechanical design, and moreover are interesting for their kinematic behaviour. The benefits of using a structure that originates from pieces of flat material include but are not limited to being able to surface treat uninterrupted sections of material that can subsequently be folded into part of the structure's configuration. In order to describe and programmatically manipulate Hinging Rigid Panel Structures and their kinematics adequately and systematically, this research takes a new and specific approach on modelling them. The Hinging Rigid Panel Structures are modelled as 1-DOF mechanisms, consisting of a base pyramid, to which 2 arms of a variable amount of pairs of triangles are attached. To sufficiently test the versatility of this method of describing Hinging Rigid Panel Structures, a classification has been set up to cover a range of kinematic input-output relations, which will be used to test algorithms against. This classification differentiates by dimension and displacement distribution over both the input and output curves, and captures all kinematic transmissions in ***1545*** cases. With a sample size of $n_p = 10$ randomly generated versions of each of these input-output relations and a motion path resolution of $r_c = 25$, a data set was generated using this classification method. An evolutionary algorithm was created to navigate the solution space more efficiently than using conventional algorithms like (gradient based) hill climber algorithms. The main variables of the evolutionary algorithm like generation size ($\mathbf{s_g} = 60$), amount of generations ($\mathbf{n_g} = 75$), mutation rate amount ($\mathbf{n_m} = 180$) and parent splicing methods have empirically been determined. Running the optimisation for all the categories of the previously mentioned classification was done by adapting the code to be able to be run in crowd computational capacity across 17 contributors' computers. The results show that Hinging Rigid Panel Structure mechanisms are to a certain extend able to be synthesised to generate requested transmissions, but some cases are harder to reach than others. Complex compound 3D paths are harder to optimise for than other categories, such as planar circular motions. This research lays out a valuable basis that contains all aspects to create a reasonable performing optimiser for 3-D Hinging Rigid Panel Structures that follow requested input-output relations. This research could for example be used as a starting point for developing a software package that allows designers to implement Hinging Rigid Panel Structures in their CAD designs for mechanical transmissions.

Literature Review

Introduction

The ancient art form of origami recently has been applied to scientific research more and more often. Origami offers more than the joy of creation out of a single flat sheet of paper. A novel topic of research concerns the ability of origami structures to serve mechanical purposes and possibly replace conventional mechanisms.

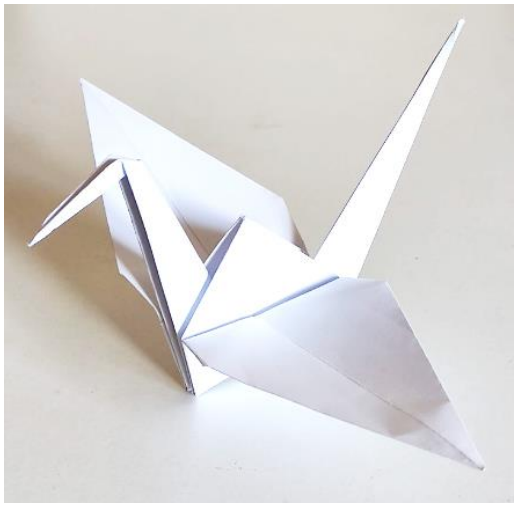


Figure 1: Origami flapping bird. In essence a simple origami kinematic transmission to transform a linear motion (pulling the tail) to a circular motion (rotating the wings).

An underexplored area of origami structures are kinematic transmissions. Kinematic transmissions find applications in a wide range of fields, including but not limited to space technology (deploying and reconfiguring structures from small stowage envelopes [1]) and medical technology [2], but also the packaging industry [3] and architecture design [4] can benefit from moving origami structures. To enable designers and engineers in the future to synthesise origami structures that couple certain user-specified motions, this literature review focuses on finding the research gaps that still exist in this area. Ultimately, the review leads into the proposal of researching the development of a computer program that allows users to generate origami kinematic transmissions using computational heuristic algorithms as a dissertation topic. Kinematic transmissions that can be created from a planar base material can be beneficial over conventional kinematic transmission mechanisms for different reasons, amongst which the fact that planar methods of surface treatment are more widely (financially) available than comparable surface treatments for 3d structures with hard-to-reach areas like cavities. This benefit can be seen in for example medical applications where bone-growth is enhanced by certain microscopic surface treatments [5].

Figure 2: The origami space array as designed by Miura and Natori [8].

Another argument for using origami or hinging rigid panel structures is the robustness. When designed correctly, origami models can handle loads far larger than their base material without creases can handle, for example the Miura-Ori tessellations in the form of zippered tubes [6]. The power of origami lies in the seemingly simple act of folding (or in more technical terms: hinging), a strategy that many structures found in nature have adopted. From plant leaves, insect wings, flower opening, proteins and therefore even the DNA, lots of natural mechanisms revolve around folding or hinging to create substructures. With its versatility, origami bridges gaps between many different research fields, since it is so widely applicable.

Overview

In this literature review the state of the art on 3 subtopics will be investigated. First, origami and its interesting features and existing applications will be explored (I). Then, available origami design software will be visited (II). Next, mechanism design using artificial intelligence strategies will be reviewed (II). Finally, an overview of the existing origami subcategories and possible research gaps will be identified.

I – Origami And Its Properties

Introduction into Origami

Origami is an originally ancient Japanese art form that studies the folding of paper into 3d shapes. One of the most common origami artefacts is the Japanese crane, see figure 1. The word origami finds its roots in Japanese; “ori” means folding, and “kami” is Japanese for paper. Origami traditionally focusses on creating complex three dimensional shapes out of one uncut piece of paper, for aesthetic reasons. Tearing, cutting and gluing are not allowed in this purest form of the art. The crux to origami is that every folded configuration has been folded from, and therefore can be folded back into, a planar surface which makes it mathematically interesting to investigate what shapes can be accomplished. Erik D. Demaine delivered the proof that any shape can be flat folded by approaching the origami structure to be developed out of a very long, narrow strip [9].

The mechanical points of interest of origami main revolve around the fact that paper, which is the basis for the origami, has an incredibly small ratio of thickness versus size of the paper. Moreover, it has the property to bend along fold lines, but it has no stretch whatsoever in planar directions. If this principle is projected out into a theoretical description, origami can be considered to have no bending stiffness in the fold lines, and a very large (theoretically infinite) stretching stiffness. This is the reason that to describe origami mathematically, infinitely thin, rigid plates are being modelled that are linked along their edges using 1 degree of freedom rotational hinges. This analogy is called “rigid origami”.

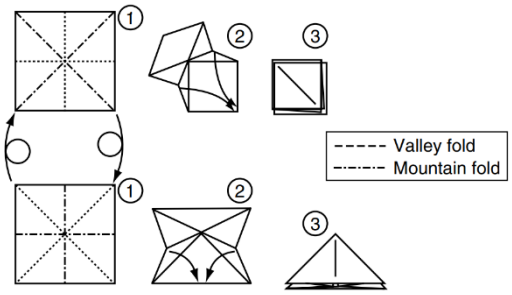


Figure 3: Waterbomb and preliminary base with their crease pattern and folding steps.

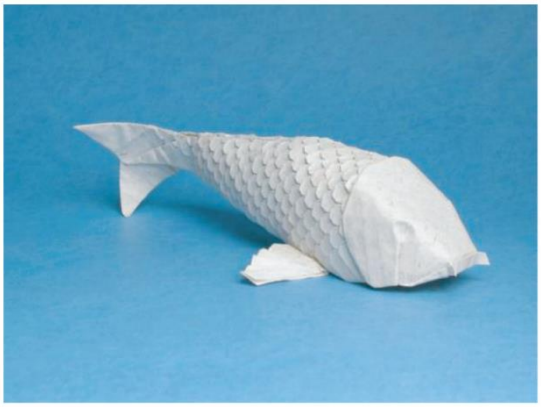


Figure 4: Covering compound curved surfaces with tessellations in the scales of the fish (art by R. J. Lang).

Describing Origami

The folds in any origami can be divided into two obvious categories: mountain folds and valley folds (respectively locally responsible for convex and concave shapes). According to D. Dureisseix [10], there are 2 bases or starting points often used for conventional origami figures, which use these two type of folds to develop the origami into the desired shape. These bases are the so-called “waterbomb” and “preliminary” base, as can be seen in figure 3, where their so-called crease patterns are depicted. Relatively simple origami structures can also be stacked. This method of origami repetitive patterning is often referred to as origami tessellations [11]. Origami (periodic) tessellations are connected pieces of origami, representable as crease patterns that can be expanded infinitely in horizontal and vertical direction by translating unit cells. Note that the ratio between the paper size and the pattern cell size is of course the limiting factor to this expansion. When every unit cell is exactly the same, a regular pattern arises which may or may not possess curvature properties in specific states of developability, but when every unit cell is altered slightly in aspect ratio or size (cell distortion) to a certain extent, these tessellations can possess properties enabling it to follow certain reasonably large radius compound curves (a 3d curve in more than 1 direction). This property is often being used in origami art, for example the scales of an origami fish, see figure 4, but one can imagine this property might also be of interest in creating curved surfaces in mechanical applications.

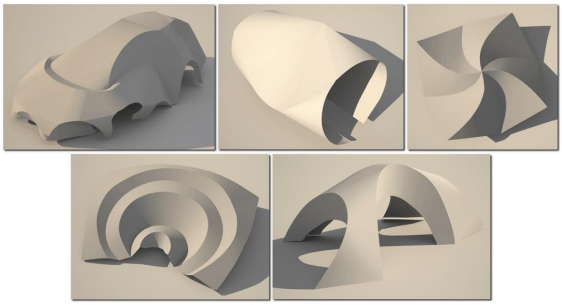


Figure 8: Curved origami shapes by Stanford University [13].

Related to tessellations is modular origami, in which multiple separate origami-folded pieces interconnect without being folded from the same piece of material, and form a new, possibly kinematic structure, for example the ring in figure 5, which can move by sliding the pieces along each other.

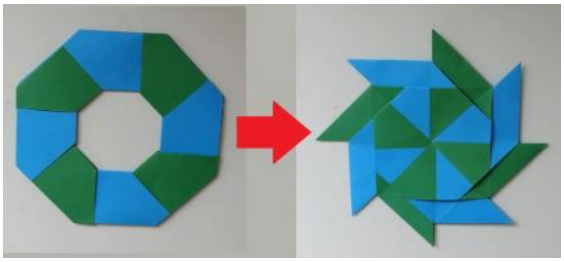


Figure 5: modular origami ring/star. Pieces slide together to create a kinematic art piece [12].

An extensively researched topic in origami is flat-foldability and developability. Developability is a property that describes whether or not a certain folded configuration originates from a flat piece of unfolded paper. In order to check if a structure originates from a flat piece of material, for each angle around a vertex, this simple condition must be true:

$$\sum_{i=0}^{n} \theta_i = 2\pi$$

Here, theta (θ) represent all the angles between sequential creases through this vertex point, see figure 6.

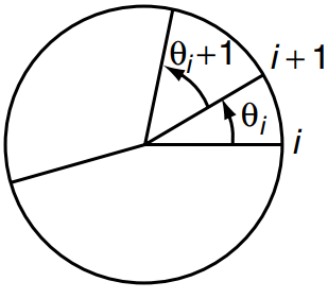


Figure 6: Angles between creases intersecting in one vertex.

The (local) flat-foldability however is described by Kawasaki's theorem. Flat-foldability is a term used to describe a state in which the paper areas around an intersection of crease lines (a vertex) can be folded on top of each other, in such a way that they all end up parallel to each other, theoretically in one planar surface. Kawasaki's theorem states that the sum of the alternating angles between the creases around the vertex should always equal to 180 degrees, or pi radians, so (see figure 7):

$$\alpha_1 + \alpha_3 + \cdots = \alpha_2 + \alpha_4 + \cdots = \pi$$

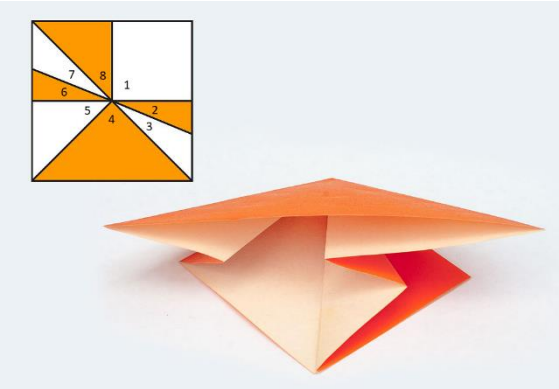


Figure 7: Alternating angles add up to pi radians, according to Kawasaki's theorem.

Note however that Maekawa's theorem requires that the difference between the number of mountain and valley folds that intersect at a vertex should always be equal to two, in formula form:

$$|M - V| = 2$$

Here, M denotes the amount of mountain creases, and V denotes the amount of valley creases meeting in the vertex. If this condition is not met, the local flat-foldability cannot be guaranteed.

Origami does not only focus on sharp folds; the subcategory of origami that studies the art of curved surface origami is another topic of active research.
However, for the scope of this research, this type of origami will not be further explored. An image can be seen in figure 8. The subcategory of origami that this paper will revolve around largely is known as action origami. In this category belong the origami creations that can be animated. The most well-known traditional action origami object is the Japanese flapping bird, the crane, see figure 1. This model moves its wings when the tail is being pulled. A linear motion is converted to a rotational movement. This property of creating a kinetic transmission using nothing more than folds is a simple form of an origami transmission.

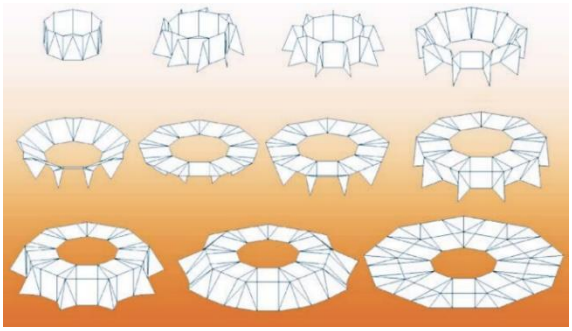


Figure 10: Unfolding a ring into a large surface area, by Robert Lang and LLNL.

In contrast to the traditional origami, another artform exists, the so-called "kirigami", where one is allowed to cut the paper in order to create (moving pop up) shapes.

As stated priorly, kinematic origami mechanisms have been mainly introduced in structural engineering in solutions regarding deployable structures [22]. Of course, theoretically, one could model origami structures as zero-energy storage mechanisms. However, the elasticity fold-lines in paper actually offers, can be used in certain cases to the advantage of a kinematic transmission design, for example to aid certain motions. To narrow down the scope of this paper however, the elasticity in hinges will not be included in the simulations.

To assess motion, an often used analysis is modelling origami as conventional 3d linkage mechanisms. One of the most famous folds is the so-called Miura-Ori fold, which can be tessellated as cells, as described earlier. In figure 9, as seen and described in [10] we can better understand how this famous fold works in its proposed mechanism form. It is important to note that some hinges have been replaced by multiple degree of freedom hinges, to prevent an overconstrained system. Figure 11 shows the origami version of this Miura-Ori tessellation. Note that modelling the origami structure like in figure 9, using mechanical hinges with appropriate amounts of degrees of freedom, does not change any of the kinematics. It is just a way to convert origami into some mechanism that has been studied for decades, and therefore is easier to comprehend.

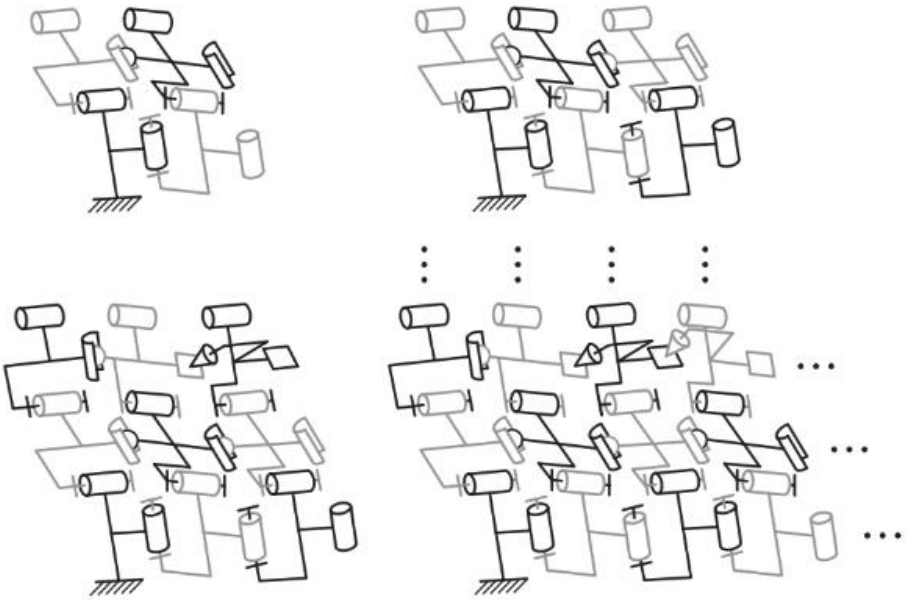


Figure 9: Miura-Ori fold in mechanism representation. In the top left, one unit cell of this fold. The others are tessellations of this cell.

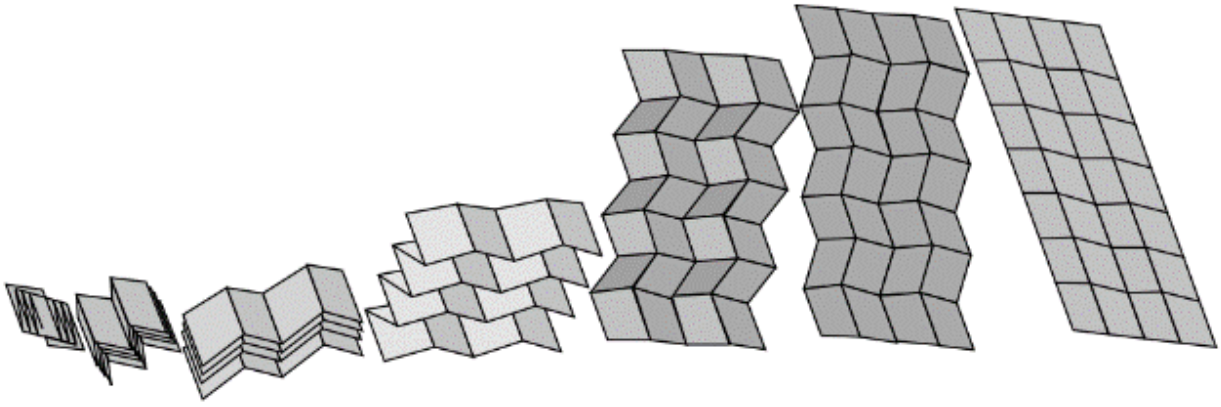


Figure 11: Miura-Ori folding state. Being used for stowage of membranes and the simplest, most well-known application is folding of maps.

Applications of Origami

Origami has increasingly been applied in novel mechanical technology recently. In his paper [14], Y. Nishiyama explains how Miura folds [15] are constructed and how they are used in deployable solar structures in space as shown in figure 2. For the space industry, minimal stowage envelopes are of great importance and foldable solar panels therefore have an advantage over conventional solar panels since they tend to take up way less space in stowed configuration. However, this often does add complexity to the system, and with complexity often comes weight and/or loss of robustness. Therefore, exploration of effectiveness and determining disadvantages of deploying origami-inspired panels has been studied more frequent the past decades, to be able to identify in what exact conditions origami related benefits outweigh the disadvantages it inevitably brings. Another space-related example is the in 2021 to be launched James Webb Space Telescope [16], which will have a foldable main mirror construction, which unfolds once the satellite is positioned in its orbit. It has simple folds to again minimise stowage space.

One of the most active origami researchers is Robert J. Lang [17], an American physicist but also well-known origami artist and theorist. In 2002, he and the Lawrence Livermore National Laboratory (LLNL) of California designed a new way to stow a 100 m diameter polygonal surface into a ring, as can be seen in figure 10, as part of a concept for a new space telescope known as the Eyeglass. The main challenge was to prevent permanent marks or creases when unfolded. Another area where moving origami is adopted is healthcare engineering. Studying buckling behaviour of cylinders, a novel type of tubular heart stent was created by Zhong You from Oxford University [18]. This new type of stent was able to be shrunken, so it would fit through veins easier, and was able to be inserted in the body in a minimal invasive manner, see figure 12.

Figure 12: Zhong You's origami heart stent design, in deployed and stowed state.

Another interesting example is the quick-folding protection shield for law enforcement, by from Brigham Young University [21], which unfolds swiftly under a certain input motion.

A final example of an application of action origami, or perhaps a better way to call it: kinematic origami, is in car air bags. The way airbags are folded largely determine the swiftness with which they are able to deploy [19]. For this airbag research, Robert J. Lang was again closely involved, to help in the development and assessment of this project, also specifically with the finite element modelling used [20].

II – Computer Aided Origami Design

Existing Origami Design Software

A few computer applications exist to aid origami design using software. There is a very well-known design tool by Robert J. Lang, called *TreeMaker* [23], that enables the user to create tree-like structures, which is helpful for creating layouts for developable origami pieces, see figure 13. More programs exist, including *ReferenceFinder Online* by Robert J. Lang [24], *OrigamiDraw* by Akira Terao [25] and *Origamizer* and *FreeForm Origami* by Tomohiro Tachi [26]. These programs all focus on finding crease patterns for developing flat sheets into specified objects, but none of them have options to explore kinematics of the model once they are folded or while being developed into a folded state.

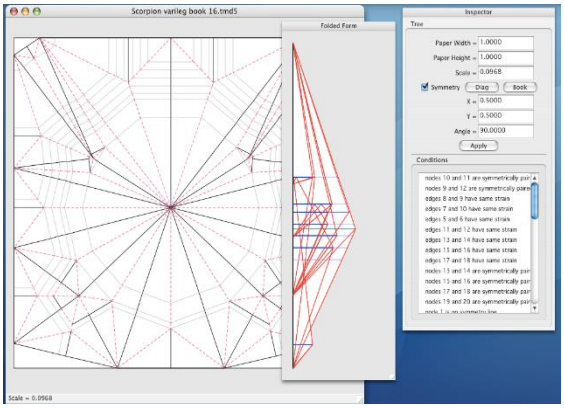


Figure 13: Robert Lang's TreeMaker 5, displaying a full crease pattern, the folded form and an inspector for editing the design.

There are however a few less known programs that do explore kinematics to a certain extend. Extending work from T. Tachi [27] and M. Schenk and S. D. Guest [28], A. Ghassaei created an online webpage [29] that interactively tries to fold origami crease patterns. The unique feature

about this simulation is that rather than trying to fold the origami in sequence of steps, it tries to fold all folds at the same time [30]. It does this by iteratively solving a small displacements problem, using the forces exerted by creases as input source. This simulator uses the FOLD-file extension format [31], created by E. D. Demaine to standardise origami crease patterns and enabling programs and functions in programs to exchange and quantify information easily. The principles of this simulator program are being used by universities around the world to aid them in their research, for example by Florida International University, where M. Khan et al. used it to model reconfigurable antenna applications [32].

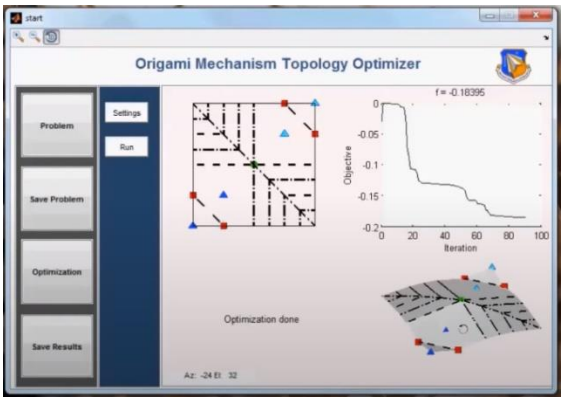


Figure 14: Screenshot of OPTO, showing the GUI that allows users for a more intuitive workflow.

Next to these few standalone programs, a few plugins exist for large software packages that aid origami design and touch on kinematics. An obvious drawback from using these plugins might be (apart from it not doing exactly what this research focusses on) the fact that they require external (expensive) digital products. For example, another project by Robert J. Lang, called *Tessellatica* [33] is capable of analysing many origami routines, however it is a plugin for *Mathematica* by *Wolfram* [34]. Another powerful plugin tool is created by K. Fuchi at the United States Air Force Research Laboratory. It is called *OMTO* [35], and offers a graphical user interface (GUI) for ease of use. It solves displacement objective problems where the user can specify force input nodes, displacement output nodes and boundary conditions on a grid. Then, it determines the fold lines that produce the largest displacement in the directions and locations that were specified. This plugin is made for *MATLAB* plus its optimisation toolbox. A screen capture can be seen in figure 14. Although this program has potential, it still does not allow for more complicated kinematics in origami to be synthesised.

Software selection for programming Kinematic Origami Transmissions

To explore the possibilities of the graphics programming and the basics of displaying origami while still being able to perform significant amounts of iterative calculations in the background, a quick test program was written on how this origami Miura-Ori zippered tubes configuration [6] would behave when increasing the volume (see figure 15). The aim of this program was to identify obstacles in the (less important) graphics part of the project. With the excellent libraries included in *Processing* [36], written in *Java*, 3 dimensional shapes could be displayed, altered by using custom knobs, graphs could be displayed, 3d geometry could be exported as .obj files, and the entire application could be exported as an .exe file, cross-platform executable, which might come in handy at a later stage when a large test would have to be run by computers on various locations simultaneously, a process called crowd computing. The *Processing* programming environment therefore seems very suitable for the early-stage development of the program.

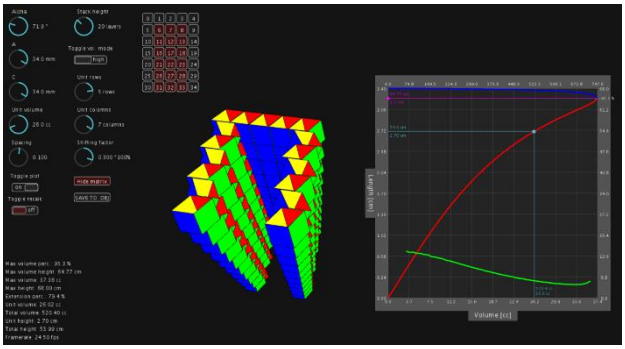


Figure 15: The Miura-Ori zippered-tubes tessellation analyser test program. Merely a check to test the feasibility of Processing as a development environment.

III – Mechanism Design Using AI

Optimising mechanisms

Designing mechanisms that are capable of following user-defined trajectories can quickly become a complex task. If done without any computer aid, it involves quite some experience from the designer to prevent a time consuming iterative design process. Recently however, reinforcement learning has been applied to different engineering fields, and proven quite effective to synthesise two-dimensional kinematic mechanisms [37]. To be able to implement these kind of optimisation strategies, a new framework had to be developed, which

defined the mechanisms in a game-like manner, in order to be able to apply reinforcement learning. This approach needed a reward system, which has been done before in the gaming sector more often [38]. This type of learning has been incorporated in researches recently more and more, since it yields promising results with tasks of which the optimum solution is not known, or problems with a very large solution space. This same analogy applies to origami mechanisms; to optimise properties of an origami structure to make it behave as wanted in a certain developability range, one needs to have a computer generate patterns, rank them and learn from their behaviour. This means scoring a solution, and altering it in a certain way. This can be done as previously mentioned, using reinforcement learning of some sort (think of neural networks for example), or using a heuristic approach, for example gradient based optimisation or an evolutionary algorithm to traverse the solution space fast and efficient.

A closely related research topic concerns mechanical bar linkage problems, these too have recently been approached using AI methods. Again, synthesising these mechanical linkages to generate specified motion paths proved to be a difficult problem, since even the simplest curves are of high polynomial degree, and relationship between these curves and the link properties is very nonlinear [39]. In their paper, J. C. Hoskins and G. A. Kramer examined the problem of selecting the parameters describing a four-bar, planar linkage, such that the linkage generates a coupler curve which optimally approximates the user-specified curve. This problem is strategically very close to the one as described to be the topic of this research. As they describe it, there is no exact inversion possible from the curve back to a 4 bar linkage, so the best fit is sought. In their paper, they use neural networks in combination with post-refinement using gradient based numerical optimisation techniques, to show that ANN's (artificial neural networks) in conjunction with optimisation are capable of inverse modelling of multi-dimensional, highly nonlinear systems. Of course, their basis is of greater simplicity than a full-on origami crease pattern, since they use the four-bar mechanism, but the paper distinguishes three different main purposes that are applicable to the origami problem as well: path, motion and function generation. For creating a path, the orientation of the object does not matter, while for motion, the orientation has a specified prescription along its movement. For the function generation, the approach is different; a desired functional relationship between input and output cranks of the four-bar linkage is being approximated.

The neural network they proposed to use is a so-called Radial Basis Function (RBF) ANN. RBF ANN's are known to be able to generate approximations to unknown (non-linear) functions by using a series of examples of input-output mappings. They are usually single hidden-layer, feedforward networks. These terms are explained in more detail in Appendix A1, where a short research on neural networks is presented, performed to learn if a neural network approach would be the route to take for this research. It however is too lengthy to include in this literature review.

In order to enable a computer to work with origami to synthesize them, score them and ultimately optimise them, a numeric representation is necessary. For mechanisms, this has been done before using various methods, using truss models or building blocks, but the crease patterns that are the basis for origami work a little different. In 2016, Eric Demaine et all. introduced the FOLD (Flexible Origami List Datastructure) format [31]. This format is different from available vector formats, since it supports coplanar facet layer ordering. It is based on the JSON (JavaScript Object Notation) format, a format of which parsers exist in all major programming languages. The FOLD format is designed in such a way that the object tree is of a very flat nature, which makes it very easily adjustable. The FOLD format could be a good starting point to use for a kinematic mechanism synthesis program, since apart from enabling the computer to manipulate and evaluate crease patterns in a numeric way, it is of use as well to convey information between projects or programs, for example to share projects or setups between users.

AI Strategies

An important topic in this research focusses on the different ways to tackle optimisation problems in general, and specifies which strategy to use for this exploration of origami kinematics synthesis. The term Artificial Intelligence (AI) is the overarching term for the science of creating (computer) artefacts that possess some form of intelligence. An exact definition is hard to give: the border between a (heuristic) algorithm and AI is often very vague. Some people claim that any optimisation program can be labelled as artificially intelligent, others think a system should make decisions that are not predictable by human beings anymore. Artificial intelligence is often thought of being very closely related to neural networks. Neural networks try to solve problems using a brain-like analogy, with neurons, synapses and connections. Since the brain is being associated with intelligence, some think neural networks are a requirement to label a system as artificially intelligent, but this is not necessarily the case. Again, a strict border of artificial intelligence is very hard to draw.

Optimisation problems are often solved using heuristics and algorithms. A very well-known example is the hillclimber algorithm. This algorithm starts with an initial guess, scores this solution, and then alters the solution slightly. It is then tested again, and the new solution is kept if the score result is better than the score of the old solution. If it is worse, the old solution will be kept. Then again, the solution is randomly altered and tested against its unaltered version. The altering happens with random mutations to the solution. This way, a solution gets better and better over time. It however is very well possible that the solution space has local maxima where this hillclimber algorithm will get stuck. To prevent this, many different tricks are being applied.

One of the most well-known is simulated annealing. In simulated annealing, the chance of the solution performing a random mutation changes over time. First, it mutates a lot, but in the end, it does not mutate at all anymore. This is a strategy to prevent being stuck in a local maximum. Without going too far in-depth on this topic, there exist a lot more similar but clever algorithms like the two mentioned to search through a solution space and prevent getting stuck with local maxima.

Another promising heuristic approach is the use of an evolutionary algorithm. Nature has been using this approach in evolution of the world and all species that live on it. When the solution space is very large, an evolutionary algorithm might be of interest, since it explores multiple routes through the solution space at once. In an evolutionary algorithm, a set of solutions is bundled in a population. All individual solutions in such a generation are scored, and based on these scores they get a chance to reproduce to the next generation. This reproduction is done by combining multiple solutions from a generation (parents) to result in a new unit (child). This solution then can be mutated slightly as is done in nature, and a new generation gets filled with these new units. The beauty of this system is that inferior solutions still get the chance to propagate, and therefore could influence the final solution still or turn out to become superior. The drawback to an evolutionary algorithm is that the problem needs to be setup in such a way that solutions can be combined into offspring.

There is another way to tackle optimisation problems as mentioned before: neural networks. Neural networks are so-called universal approximators [40]. This means they work very well for problems where associations need to be captured, regularities need to be discovered, or for problems where the relationships between variables are hard to understand or describe with conventional approaches. Also, neural networks are great in handling a large amount of data, or if the volume, number of variables or diversity of the data is large. In general, one can say that for dynamic or non-linear relationships, neural networks are strong when it comes to capturing relationships between model phenomena that are difficult or impossible to explain otherwise. In comparison to conventional computing techniques, neural networks are not sequential nor necessarily deterministic [40]. The tasks that the computer has to do is very simple, simply taking some weighted sums and putting them through a simple formula. Neural networks do not execute instructions in the conventional way, they work in parallel to the pattern of inputs which it is exposed to.

There however is a major drawback to using neural networks, which is that the problem to optimise needs to be made suitable for a neural network to interface with it. Some optimisation problems do not have an easy way to do this. In this paper, this problem will be explored.

Concluding, we can say that neural networks are ideally suited to solve complex, hard to describe problems, they can learn to model relationships between inputs and outputs that are nonlinear and complex. Neural networks can recognise patterns, make predictions and generalisations. All of these come in handy when trying to design a mechanism. The features the mechanism possess might not be identified yet, but the neural network possibly finds groupings of creases that can be responsible for certain motions and stack these features to create complex motions. However, neural networks do have a complexity in interfacing with the problem, which is not as hard when using more conventional optimisation techniques like evolutionary algorithms.

Discussion

When reviewing the state of the art of the three topics, origami and its features, existing design software and artificial intelligence strategies, it becomes clear that there is a research gap when it comes to origami transmission synthesis. In table 1, an overview is given of all relevant subcategories of origami, plotted against the different types of mechanisms that are possible to quantify and calculate the motion of mechanisms. We can see that cell C3 has the focus of this thesis, a combination of modular origami (not bound to developability from a single sheet) and heuristics to search through the solution space.

	Analytical	Linearised	Heuristic
Regular (developable)	A1	A2	A3
Tessellations	B1	B2	B3
Modular	C1	C2	C3
Curved	D1	D2	D3

Table 1: Research gap matrix

Quite a bit of work on kinematic behaviour of folding has been done, where mostly perfectly rigid panels and perfectly strain-free frictionless hinges are assumed. In this thesis, this will be assumed as well, to narrow down the scope. However, in almost all real-world folding mechanisms, the behaviour is strongly affected by the issues of panel bending, stiffness of panels and hinges and friction. All of these factors will be complicating accurate modelling a lot. Moreover, in direct correspondence with Robert J. Lang via email, it became clear that the use of neural networks, or AI in general can introduce quite some problems. The feasible subspaces of parameter space tends to be quite sparse, typically of much lower dimensionality than the parameter space of the problem. This might result in the algorithms returning merely infeasible solutions. Therefore, a few clear pre-determined compromises for the program need to be set, to make sure the parameter space that is being explored by the program is not so large that finding feasible solutions becomes virtually impossible.

Considering all of this, the optimising mechanism proposed is using an evolutionary algorithm. This decision was made with regards to the authors knowledge of object oriented programming, but moreover the interfacing problems between neural networks and origami structures and the concerns raised by Robert J. Lang.

Thesis proposal

The thesis proposal that has been hinted on heavily in this literature review can be formally written down as being:

"The aim of this thesis is to set up the fair principles of a computer aided design tool that allows users to synthesise hinging rigid panel kinematic transmissions coupling user-specified input and output motions, using an evolutionary algorithm to optimise these."

The thesis has some aspects that are to the authors knowledge not combined before, especially the use of evolutionary algorithms to approach origami kinematics analysis. To develop this into a tool that people could use in their designing progress would be something that feels like could help many people in many fields of research and work. Moreover, I hope it will inspire people to rethink existing solutions for mechanisms, and actually bring new ideas to live.

Introduction

Research Overview

Following the recommendations that resulted from the literature review, this research will focus on synthesising hinging rigid panel structures using evolutionary algorithms to connect defined input and output paths. It is important to note that rather than sticking to classical origami, which can always be developed from a single sheet of planar material, this research deliberately lets go of that property. This restriction could prevent good kinematic transmissions to be created, as it limits the solution space significantly. If good solutions can be found that do require some assembly of different sub-structures if they would be built physically, or if more than 2 planes need to hinge together in 1 edge, this is still of value for reaching the goal this research focusses on. Therefore, another, possibly better covering description of the structures this research revolves around is: Hinging Rigid Panel Structures, from now onwards referred to as HRPS. By Hinging Rigid Panel Structures, structures are meant that are created out of triangular planes that hinge together over the full length of their sides. In this study, the thickness of the panels are considered zero. In contrast to classical origami structures, Hinging Rigid Panel Structures can have edges where more than two planes hinge together, introducing a large area of the solution space of kinematic transmissions.

First, the methodology will be presented, to give the reader an overview of how the problem was tackled. The methodology comprises the largest novel work in this research, and should be seen as the main work done in this study. The methodology is relatively concise, but many more detailed problems are further elaborated upon in the appendices, so for further reading please follow the references in the text. Also, a brief overview will be given of the crowd computation strategy that was followed to generate the data set during this research. After that, a section with results will be presented. Next, the discussion section will further interpret the results, and a few interesting cases will be highlighted that are of special interest for mechanical applications. Thereafter, a section will present an overview of steps that might need further work or revisiting. Finally, a conclusion will be given to summarise the work.

Note that the results are interesting and teach us about how the methodology worked out, and the results should be seen as a validation of the method section, where the novelty lies, the approach to how the origami structures can be described and altered.

During this thesis process, a computer aiding tool to enable (product) designers and researchers to create kinematic mechanisms using flat sheets as starting points has been developed, but it is in a researching state, not finalised for active use in designing processes. I want to emphasise that the authors intention with this research is for it to serve as a starting point for further work in this direction, to try to kickstart development and use of origami structures in motion. A possible follow-up project on this study would consist of implementing all principles and methods from this study, and transform it into an actual software package that can be rolled out to use in mechanical design, or other fields. Kinematic transmissions created out of origami can find their application in many fields where origami is slowly being introduced, from packaging and stowage problems (aerospace) [1] to medical purposes [2], material sciences, forming steel, chemical engineering for 3D structures [5], architecture [4] and even art. Moreover, applications can be found in fields not considered here since it is not conventional (yet) to apply origami mechanisms in many fields of engineering or research.

Methodology

Introduction

To easily handle Hinging Rigid Panel Structures and display information in a concise way, a graphical interface has been created using *Processing* [36]. With its JAVA backend, it is a relatively fast, efficient and easily accessible programming environment to develop graphical projects in for engineering students. It also allows for easy distribution of the program in *.exe* format, which proved handy when crowd computing large data sets, as has been done during this research. Using *Processing* the algorithm has be created that drives the synthesis of the Hinging Rigid Panel Structures, but it needed a thorough method to model the Hinging Rigid Panel Structures, which will be described in this section. Once this method was created, a first approach of an evolutionary algorithm to evolve and optimise solutions for input-output relations has been applied.

In order to adequately analyse the system, as well as test and compare the performance of various algorithms attempting to optimise the shape of hinging rigid-panel structures (HRPS), a thorough and accessible coding approach has been adopted, to accommodate tests and iterations easily and modularly. Various methods and structures have been explored, and this section will summarise the main strategies and approaches used within the code, which comprises the basis for this research. It will allow the reader to reproduce the structure of the program, although it will not go in detail on code level problems.

Firstly, the HRPS model itself and its accompanying functionalities will be discussed and elaborated upon. Then, an overview and explanation about the motion paths and their classification will be presented. Finally, the working principles of the algorithm used will be elucidated.

Describing Hinging Rigid Panel Structures

Strategy overview of describing an HRPS

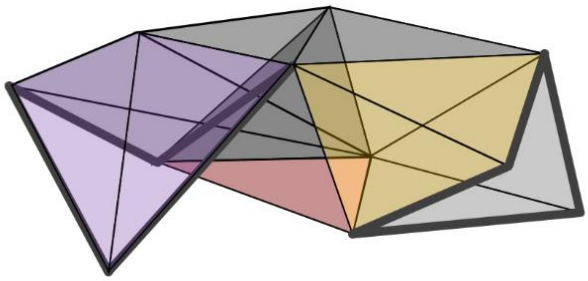


Figure 16: One possible HRPS, with input branch and output branch both 3 tristructs long. The emphasized lines represent the current input and output loop. The purple and yellow planes are the ends of the branches, The red plane is the fixed plane.

It is important to understand what strategy this research follows to build up HRPS's. Although there is a very strict and simple set of rules an HRPS has to follow in order for it to be physically constructed (not intersect with itself or needs to be stretched to construct), the HRPS is a structure that cannot necessarily be prescribed by any fixed plan for its construction. This however is necessary to enable an algorithm to easily modify the HRPS properties later on, even if the complete structure is already generated. In creating such a building method however, it is of great importance that the system still includes as much of the parameter space of HRPS as possible, to prevent disregarding possible solutions in an early stage of the algorithmic process. The building method that was used in this research will be described below.

All information to define the geometry of one enclosed HRPS and the interconnections within this set is contained in one programmatic class. The chosen construction method always uses a base pyramid, to which two branches of additional planes can be added, one for the input side of the kinematic transmission, and one for the output side of the kinematic transmission. These branches consist out of pairs of triangular planes, and as many pairs can be added to either branch, which will be elaborated upon shortly. This specific method ensures the entire system always maintains a single degree of freedom. These pairs are called tristructs throughout this paper. For a visual representation, see figure 16.

In essence, the structure transforms a multi-variable input path through the input branch into a single rotation in the base structure, definable by angle theta as depicted in figure 17. The other branch is connected to this same base structure and transforms it back into the desired output motion. By doing this, instinctively it seems like we are disregarding certain very low-end, robust solutions for certain input-output relations. But be

reminded that it is not always necessary to add very long input and output branches, so simple (possibly more robust) models are still within the parameter space.

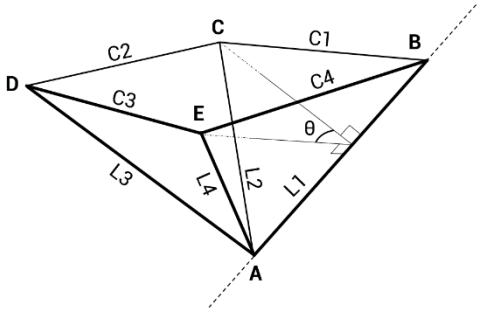


Figure 17: the system of 4 triangles that forms the basis of a hinged rigid panel system. It can be considered a pyramid with a non-planar base (i.e. points B, C, D and E are not restricted to be contained in one single plane.)

The HRPS model

Any HRPS in this research is constructed with at its base a set of 4 triangles that are connected into a pyramid with a non-planar base. One of these triangular sides of the pyramid is the plane that is fixed to the world frame, to the ground. The four triangles come together in 1 vertex, therefore forming a four-vertex. The relative configuration of this pyramid shape is fully definable with 8 lengths, as depicted in figure 17. The reason this shape was chosen as a basis is its limited degrees of freedom. It is a structure that possesses a single degree of freedom only when 1 plane is locked to the world ground. This degree of freedom can be set by angle *theta* in figure 17. *Theta* can vary from 0 degrees to 180 degrees, and this value *theta* will be from now on referenced to as the "*development angle*" of the HRPS. The development angle of an HRPS will always be set between 0 and 180 degrees.

After a base has been set up, planes will be added to the structures earlier mentioned branches. To maintain the single degree of freedom, this has to be done in a specific manner. Firstly, 2 loops are defined that will form the base of these input and output branches. They consist of 4 adjacent edges of the structure. Since the base pyramid only possesses a single degree of freedom, these loops will always be fully defined by setting the development angle. These two loops are in further reading referenced to as the input loop and the output loop. They will change when a set of triangles is added, as will be described below. One can imagine that there are multiple possibilities to choose from when assigning these loops. As depicted in figure 18, the most obvious loop is the loop around the non-planar quadrilateral bottom, simply framing the open side of the pyramid (closed loop BCDE in figure 17 or figure 18a). The second loop can be chosen to be any of the other possibilities in figure 18. To these loops, sets of triangles can be added.

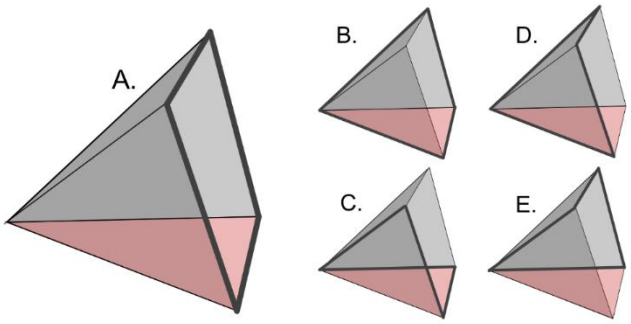


Figure 18: A: Open loop that frames the non-planar bottom of the pyramid. B-E: Open loops that do not frame any opening in the pyramid.

By adding these triangle sets (the tristructs), the input or output loop will change, and the 2 edges that were chosen to build the new tristruct on will be replaced by the 2 newly created open edges. For a visual clarification, please see figure 19, in which sub image A-D demonstrates a possible branch synthesis with at its basis loop A from figure 18, and sub image E-H demonstrates a possible branch generation with at its basis loop E from figure 18. It is clear that displaying states of the model quickly gets very unclear, as can be seen in figure 16, where figures 19d and 19h are combined to create one full HRPS.

Throughout the rest of the construction of various hinging panel structures, this one degree of freedom condition is always maintained, since the scope of this research focusses on single-input/single-output relations. It does not matter how many tristructs (*n*) are added to the system (see appendix B1 for further explanation):

$$DOF = 6 \bullet (5 + 3 \bullet n) - 3 \bullet (8 + 5 \bullet n) \ - (5 + 3 \bullet n) = 1$$

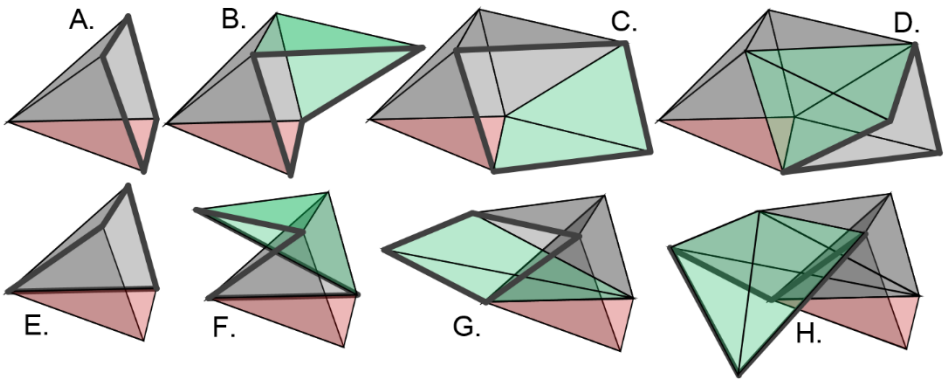


Figure 19: 2 possible branch generation patterns. A-D demonstrates generating a branch with 3 additional tristructs on figure 18 loop A. E-H demonstrates generating a branch with 3 additional tristructs on figure 18 loop E. Combining figure D and H will result in a full HRPS as depicted in figure 16. The green planes represent the added tristructs in each step.

All geometry described in the system is build up using trilateration. Trilateration is very frequently used in the code, since it describes the location of a point when 3 other reference points and 3 corresponding distances to the unknown point are given. By keeping track of which tristruct originates from what edges, the full structure is easily defined. In order to do this, lists with connection information are stored within the programmatical HRPS class. There are separate lists for the input branch and for the output branch, in which the information of the tristructs is stored as well. Important to note is that the tristruct (which is contained in a class too) also stores a number related to which side of the previous loop it is connected to.

For more information about this tristruct class and how the trilateration problem was solved, see appendix B2. To read more about how solvability was checked and how collision detection was used to prevent impossible solutions, see appendices B3 and B4.

Classification of Hinging Rigid Panel Structures

Describing motion paths

A classification of the infinitely large solution space of kinematic transmissions of path generation mechanisms has been set up in such a way that the computer program can systematically cycle through it. Moreover, we can clearly demarcate which part of the solution space has been covered and how, and where possible future research could continue.

All transmissions are in essence nothing more than an input motion path, and an output motion path, both discretised as a set of points, the amount of which is called the path resolution (r_c). In If a set of n paths exist that are separately classifiable, a set of n^2 transmissions exist, since every input path can be combined with every path as an output path. Therefore, the classification focuses on single paths, and with some exceptions to narrow down the duplicate options that arise (which will be elaborated upon), these single paths can be combined with each other to create a set of *1545* different transmission cases. An exact explanation of this number is given in appendix B8. For a graphical overview of this classification (that will be discussed below), please see figure 20.

Distinction is made between 1D, 2D and 3D paths with various subclasses, and in addition to these, every path can have a different motion distribution over it. A short explanation of this will be provided in the next section.

1D Motion Paths

In this research, a 1D path is not confined to a singular axle in the global coordinate system, but it rather constitutes all the rectilinear motions, no matter their 3D orientation in space. Within this 1D subclass however, a subdivision is made for a few important cases, which are the three principal origin directions, so a rectilinear motion parallel to the x, y and z axes, and a fourth case which is a randomly orientated rectilinear motion.

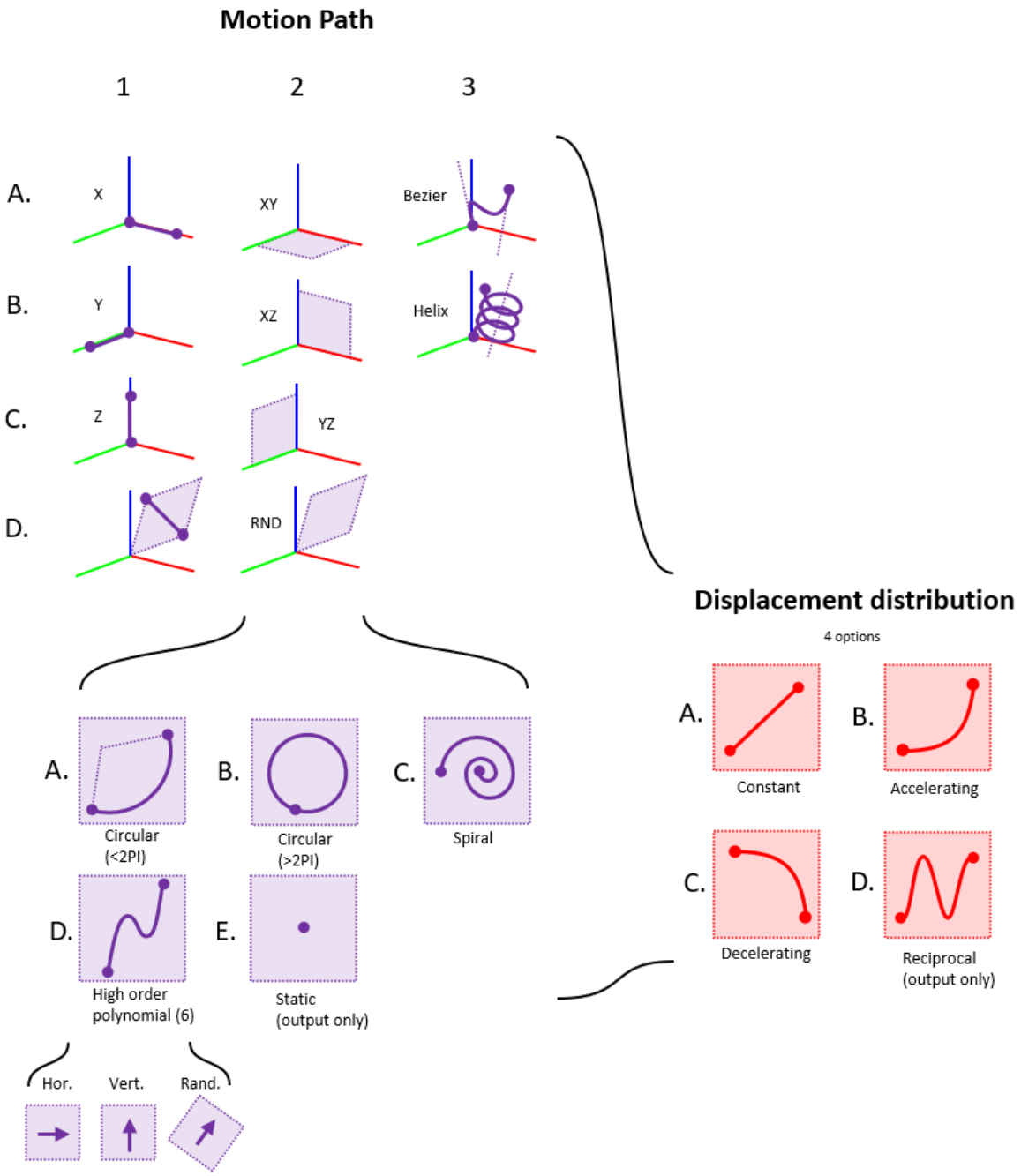


Figure 20: Graphical overview of the classification used to describe transmissions.

2D Motion Paths

The 2D subdivision is more comprehensive. It comprises a 2-step subclassification, where the first step is setting a plane in space (XY, YZ, ZX or randomly oriented), and the second step is creating a curve on this plane. Five distinct function groups are chosen to be the most common paths and therefore have their own subclass, which are:

- A circular path with an angle less than 2π radians, from which the diameter and angle can be set (randomly if required).
- A circular path with an angle larger than 2π radians, from which the diameter and angle can be set (randomly if required).
- A spiral from which the amount of circulations and the radial spacing can be set (randomly if required).

- A static single point (which can only be used in an output path).
- A polynomial with an order of 6 from which each orders magnitude can be set (randomly if required). Within this polynomial group, a subdivision is made for polynomials that are oriented with a horizontal x axes, a vertical x axes or a randomly oriented axes.

3D Motion Paths

The 3D subdivision consists out of just 2 subclasses:

- A 3D Bezier curve (see appendix B6 on more information about the exact shape of Bezier curves), from which all 4 anchor and control points can be set in 3d space.
- A helix that uses a 2d plane as its base plane, from which its diameter and height and amount of rotations can be set (randomly if required).

Describing path distribution

Not only does this research focus on kinematic transmissions that link two motion paths together using a HRPS structure, also different path distribution spacings are taken into consideration on top of the previously described shape classification of the paths. This heavily increases the amount of possible transmissions. Four displacement distributions are distinguished (see figure 21), these are:

- Constant distribution (all points over the curve are equal distance spaced along the curve)
- Accelerating distribution (the spacing between each set of points increases over distance along the curve)
- Decelerating distribution (the spacing between each set of points decreases over distance along the curve).
- Reciprocal distribution (the path is not unidirectionally used, but the motion may go back over itself multiple times in a sinusoidal manner). This last distribution will only be applied to output curves, which limits the amount of options significantly again.

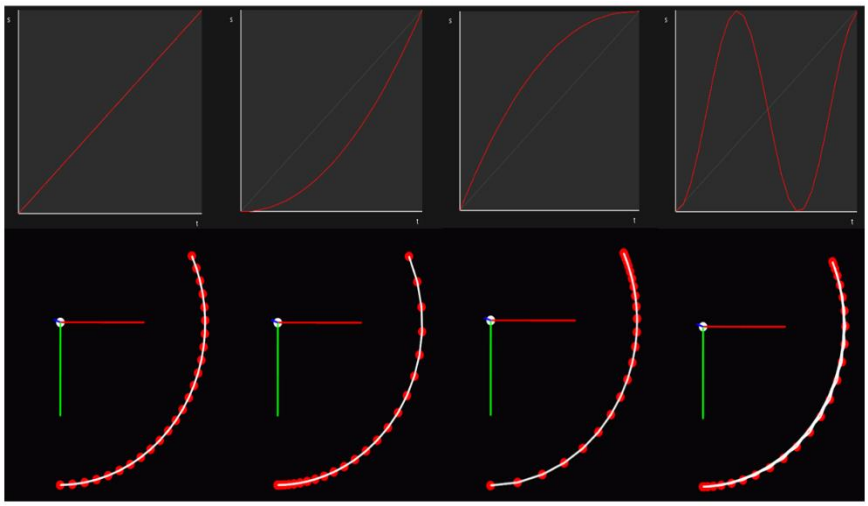


Figure 21: The four different path distribution shown as graphs and as distributions over a 2d open circle. With a low path resolution and a

Of course, more standardised distribution curves can be used, but in order to limit the amount of categories in the general classification and keep the scope concise, these four were chosen. Spacing points with a prescribed distribution pattern over certain curves is not always a trivial problem, please see appendix B5 on how this problem was approached. To get an insight over how to intuitively manipulate all these factors in the program using a GUI, please see appendix B7.

Each of the motion paths is combined with each of the path distributions and each of the in the above section described motion and distribution parameters can be randomised in the program, in order to generate a set of $n_p = 10$ different versions of each of the transmissions. Using a set of samples rather than using a single

sample for each transmission would eliminate outliers, taking the averages of scores of the final solutions over these 10 samples. This however results in a data set of 15450 input-output relations, which all had to be optimised using the evolutionary algorithm.

Naming conventions

To analyse how cases are performing and how their overarching categories from the classification as shown in figure 20 are performing against each other, a solid naming convention was needed. The naming convention used exists out of 2 parts, linked with a hyphen. The two parts represent the input and output path of the transmission. An example could be *2AD3B-3BC*, which happens to be case number *1230* of *1545*. It means *2AD3B* is the input path, 3BC is the output path. As described in figure 20, *2AD3B* represents a 2d motion on a plane parallel to the XY plane. The letter *D* makes it a polynomial, oriented with a random orientation (*3*). The last *B* states it is using an accelerating motion along this path. The output side, *3BC*, describes a helix with a decelerating motion path.

Evolutionary algorithm

Tuning the algorithms parameters

In order to run the entire simulation of 15450 optimisations, the evolutionary algorithm needed to be coded efficiently, to keep the run time down. Of course, runtime could heavily be reduced by reducing the motion path resolution to $r_c = 25$ instead of a larger number which would return more precise results.

The main variables of the evolutionary algorithm are generation size ($s_g = 60$), amount of generations ($n_g = 75$) and mutation rate amount ($n_m = 180$). These numbers will be explained in the section below. Also, parent splicing have empirically been determined, which will be elaborated upon. Running the optimisation for all the categories swiftly was done by setting up the code to be able to be run in crowd computational capacity across 17 contributors' computers, which will be explained below as well.

.HRPS files

In order to save, mutate, recombine and copy data regarding the properties of an origami structure, a solid file format was needed. Although the literature review suggested using Eric Demaines FOLD (Flexible Origami List Datastructure) format [31], this research moved away from solely optimising fully developable origami structures from crease patterns. The underlying JSON format however was used, and a new format was created, see figure 22. A brief note on sections 5 and 6 of the file format (figure 22) is needed. These sections describe the input and output arm of the system, using tristructs. Each tristruct contains one integer, and three floats. The three floats describe the three lengths of the tristructs (trilateration is used to calculate where they form a new vertex), and the integer describes how it is connected to the previous input- or output loop.

The .HRPS file is easily exchangeable with the program, as JSON arrays are supported by JAVA.

Recombination

Recombining 2 or more structures into 1 new structure is essential for an evolutionary algorithm. Various options have been explored, where 2 HRPS information arrays were merged with different methods. Ultimately, a method has been adopted where the base pyramid information will always entirely come from 1 parent, and only the input and output arm data will be recombined.

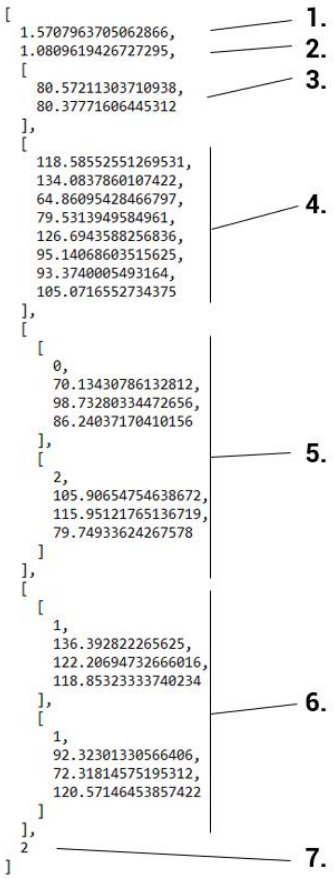


Figure 22: The .HRPS file format. 1 is the development state of the origami and is not a number that is subject of optimization. 2 is the angle of the grounded triangle, 3 are the coordinates of the grounded triangle on the ground plane. 4 contains the 8 base lengths to define the base pyramid. 5 contains a list with (in this case 2) tristructs that represent the input arm. 6 contains a list with (in this case 2) tristructs that represent the output arm. 7 contains a number that determines how the output arm is connected to the base pyramid, in other words which of cases B-E is chosen from figure 18.

This was done by choosing a random amount of tristructs from one parents input arm from the base outwards, and a random amount of tristructs from the other parent from the end of the input arm inwards. The same was done for the output arm. This way, there is the possibility of just copying the already existing arm entirely, but also of adding features from the other parent, or even entirely adopting the arm from the other parent.

Mutation

After recombination, the entire generation was submitted to mutation. For the entire generation of HRPS systems, a fixed amount of total mutations was empirically determined (n_m = *180*). For each of the mutations, first one of the HRPS systems was chosen randomly (therefore mutations could be happening to a single HRPS multiple times), and then a random value in the HRPS array was randomly slightly altered with a given range of change. Moreover, the chance of the mutations actually happening depends on which variables are randomly selected for change, and how many generations already have been generated. Namely, the dimensions of the main pyramid and its position have a larger chance of not being mutated if selected for mutation as more generations have passed. The underlying idea is that changes to the base coordinates and the base pyramid dimensions are of such great influence that ending up in a totally different area of the solution space is easily achieved. While this is great in the beginning of the algorithm, when the solution spaced needs to be explored quickly, it can cause overmutation that leads away from good solutions near the end of the optimising process. Note that the chance of the mutations being performed to the base coordinates and base pyramid lengths decrease linearly over time, from 1 to 0, if chosen to be mutated. So the chance will not be 100% in the beginning, but if the base coordinates or base pyramid lengths are chosen to be mutated, this is not blocked in the beginning of the algorithm, and always blocked in the very end.

Score function

The score function is of great importance; it determines how good a certain solution is. It is a very straightforward process. It follows the output and input point on the HRPS, and for each point compares it with the desired output and input path points set to be optimised for. It calculates the distance between each corresponding point, and sums these distances. This means that a path that is far away from the desired path gets a high value, which is a worse score than a path with a low value. Note that this distance is used directly for the score, so it uses a linear distance function. Please see the Discussion section of this paper to learn how this could be changed or how non-linear score functions might be beneficial to use. Due to the already quite large scope of this research, the linear function was kept for sake of progress.

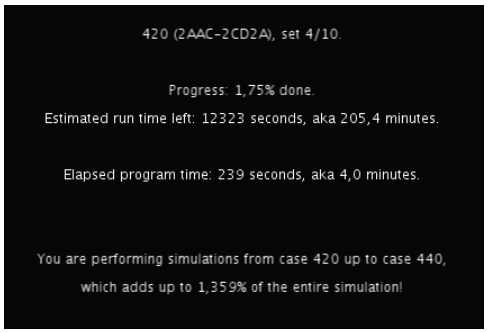


Figure 23: The contributors were notified about the process on their computer by this information screen.

Crowd computing

The algorithm was estimated to take around 40 seconds per generation on a standard high-end laptop. Optimising all 15450 cases would therefore take just over 7 times 24 hours. In order to bring this down significantly, the program was made suitable for crowd computation, splitting tasks up in chunks of 20 cases (see figure 23). 17 volunteers were found to perform this task, and in only 14 hours, the entire simulation was completed. The results of all the optimisations were stored in 2 files. One file containing the information about the optimisation (figure 24) process and one containing the HRPS that was the final solution of the optimisation. Not only the result of each generation was stored, but also information about the runtime (depending on the computer it was run on), the uniqueness of solutions in the generation (duplicates can happen when some

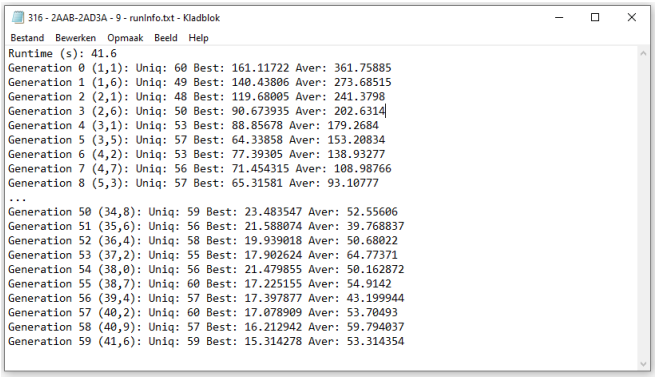


Figure 24: The information file about one optimization.

HRPS'es are much better than the rest in the generation and therefore will be chosen very often for reproduction to the next generation), and the average score of a generation. This information is used in the results section of this research to compare sections of the classification.

Analysing the results

In order to adequately analyse all results, a program was created that cycles through all the result files and stores all information in a instance of a programmatical class. Another class was created to store 10 results at once, in order to analyse average results per case number. On some results, some low-pass filtering has been done to reduce noise, using a moving average filter with an empirically determined filter extent of *18* in both directions.

Results

Classification simulation results

Iterating through all the result files, point cloud graphs have been produced to give a quick overview of the results. Figure 25 presents all the 15450 cases best scores against their average scores in the last generation. A centroid can be seen towards the left bottom of the graph, where both scores and average scores are relatively

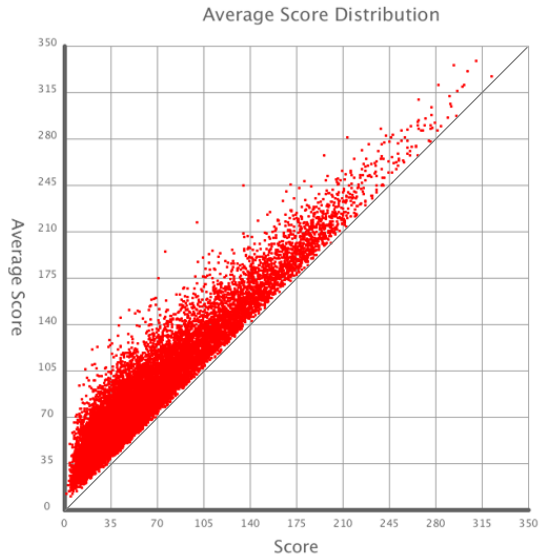


Figure 25: Average score versus best score. The clear diagonal line split can be explained by the fact that the average score can never be better (lower) than the best score from a generation. For visual reference, the diagonal is depicted as well.

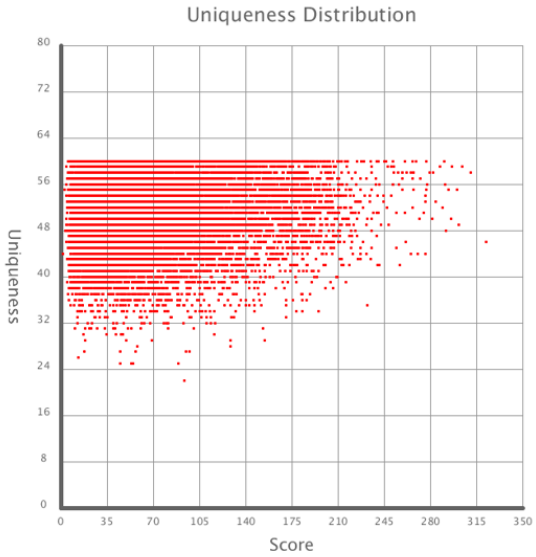


Figure 26: Uniqueness of the generation versus best score.

good. Also, a clear diagonal split can be seen, beneath which no points are plotted. This is a result of the average score of a generation never being able to be lower than the best score of a generation. Figure 26 gives an overview of the spread of the uniqueness of solutions in the last generation. Again, a centroid can be observed, this time towars the left top of the cloud.

Figure 27 presents the run time for each case versus the (best) score. Important to note for this graph is that the results are influenced by which computer ran the cases, as some computer systems proved to be faster in finalising a set than others, this graph might be skewed. However, a centroid towards the left bottom can be observed, and an average absence of solutions that both have a large score and a large run time.

Figure 28 shows all the score of all 15450 cases against their case number index, 0 to 1544 (1545 total). Groups with similar scores or similar spread can be observed.

Figure 29 shows the same set, but with a black line following the average of the 10 subcases per category.

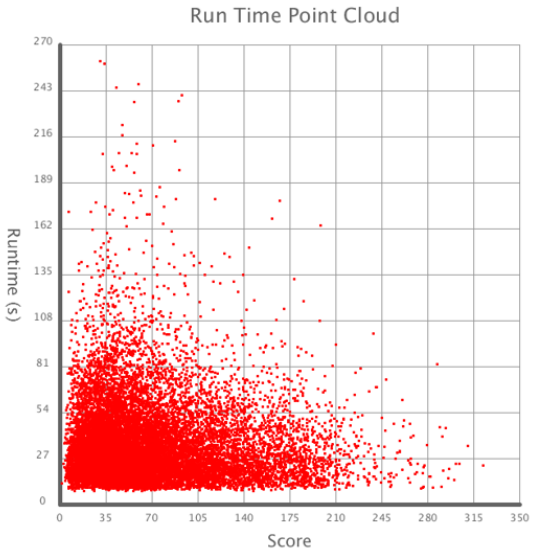


Figure 27: Runtime of the versus best score.

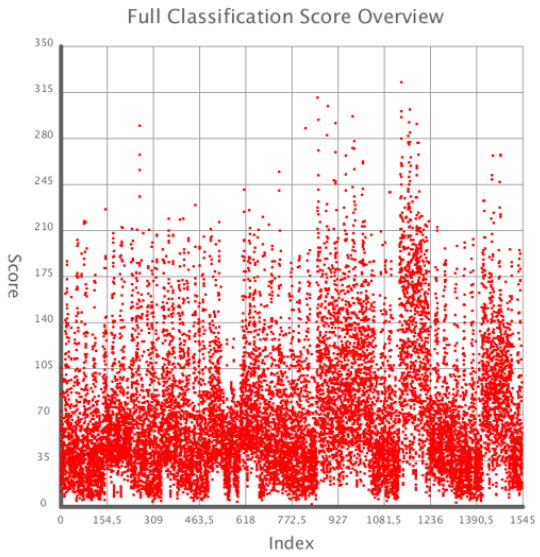


Figure 28: All cases best score against their case number. Groups can clearly be seen.

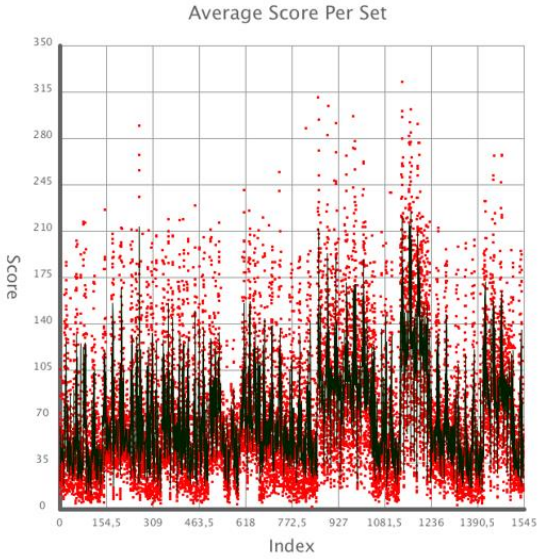


Figure 29: Average of al 10 subcases per case against their case number.

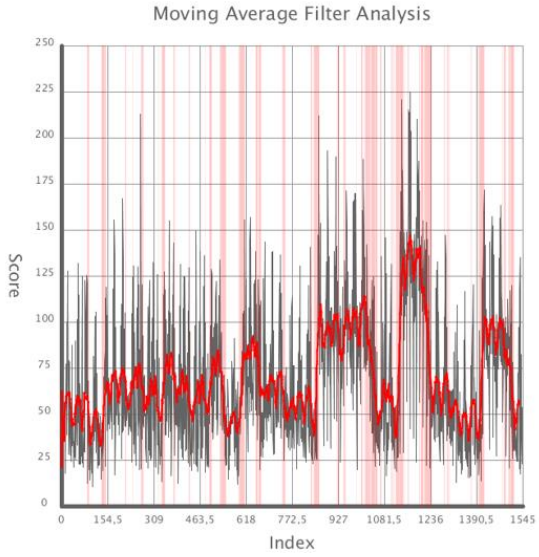


Figure 30: The average of all 10 subcases versus the case number, but with an applied moving average filter width of 18 (to both sides). Areas of heavy increase or decrease are demarcated with red vertical bars. These are border batches of cases with similar results.

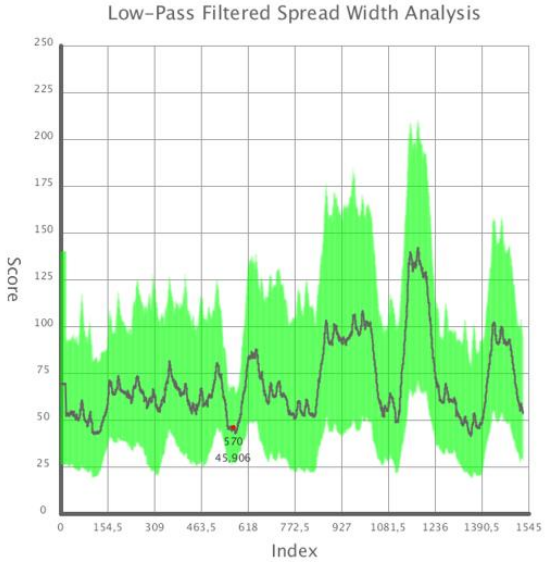


Figure 31: The average of all 10 subcases versus the case number, but with an applied moving average filter width of 18 (to both sides). The green area shows the height of the score range in which scores were found in the last generation.

Figure 30 displays information regarding the filtered average of the scores, and the red line through it is a low-pass filtered version, to emphasise trends. The red vertical bars represent areas of heavy change of score, therefore demarcating batches of equally scoring cases.

Figure 31 depics the same filtered average line, but also shows the spread range of the scores in the last generation per case. One point of interested is marked, where the spread is relatively narrow around case number 570. Also, a relatively bad scoring area can be seen around case 1180. Moreover, the overal trend seems to be that the low-indexed cases seem to perform better or at least less inconsistent than the higher cases (remember that low values indicate better scores).

For this following section, please refer back to figure 20, as the same symbols and terms have been used to indicate cases and subsets. Figure 32 to figure 38 show comparisons between different subsets of the full dataset of 15450 cases.

The orange bars represent the input paths, purple represents the output paths. The numbers under the zero-line represent the amount of cases processed with the condition stated. The height of the bars represent the average score of all the particular subsets. In figure 32, all cases are summarised by their dimension. The difference in amount of subcases in the 2-D category in comparison to the 1-D or 3-D cases can clearly be distinguished.

Figure 33 displays how the subcategories of the different path spacings compare to each other. A relatively good score average has been found for

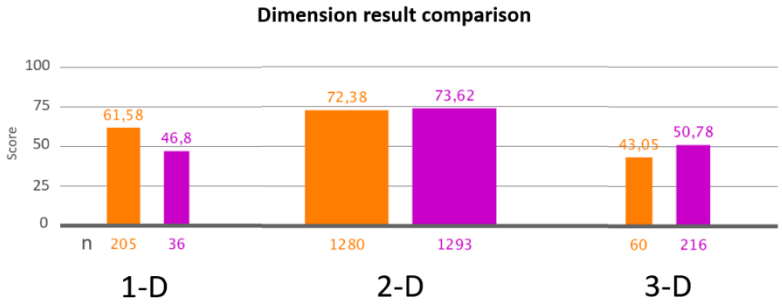


Figure 32: Comparison graph to analyse the differences between the sets of the dimensionally different paths. Orange for input, purple for output. The number under the line represents amount of cases (linked to bar width). The bar height represents average score of this case.

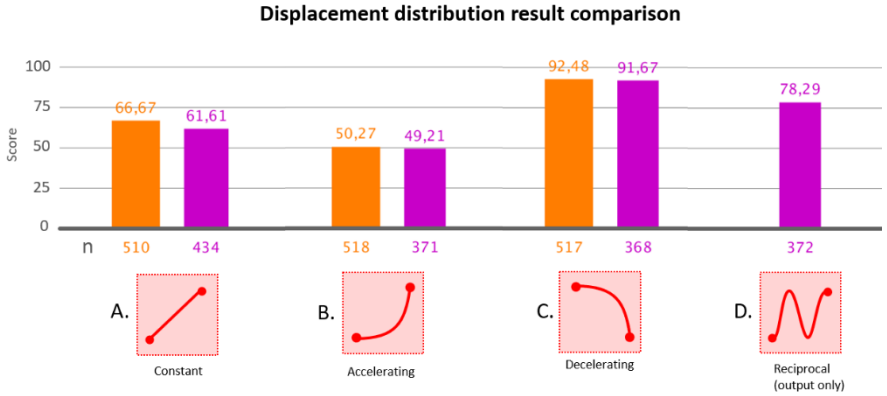


Figure 33: Comparison graph to analyse the differences between the sets of the distributional-wise different paths. Orange for input, purple for output. The number under the line represents amount of cases (linked to bar width). The bar height represents average score of this case.

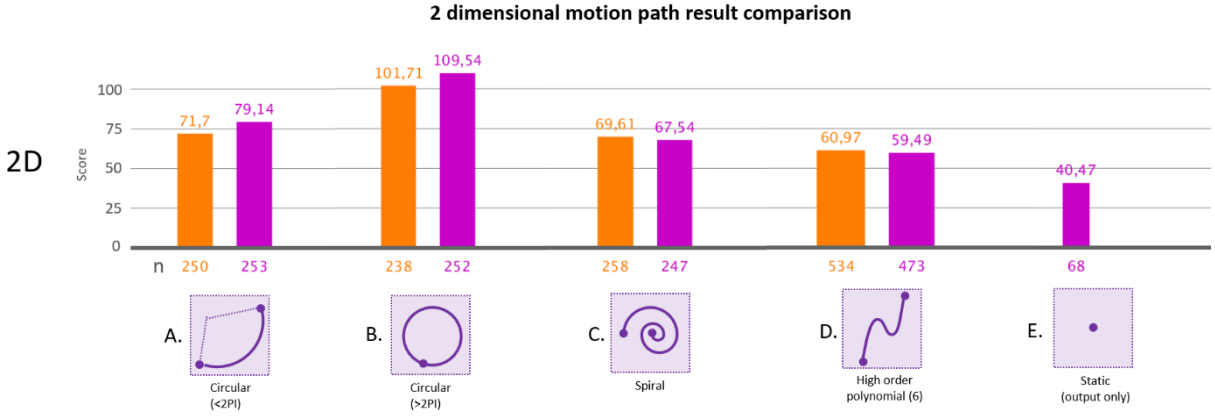


Figure 34: Comparison graph to analyse the differences between the sets of the different 2 dimensional paths. Orange for input, purple for output. The number under the line represents amount of cases (linked to bar width). The bar height represents average score of this case.

accelerating curves, while decelerating curves on average have a worse score. The accelerating curves have a better score than the constant spaced curves. Note that for reciprocal spacings, only an output bar is given, as in this research, input paths with a reciprocal nature have not been considered.

The 5 subcases for the 2-D domain are shown in figure 34. The subcategory of the circular paths (B) with a circle angle of more than 2π has a significantly worse score, both for the input and the output situations. Also, the subcategory of the static points

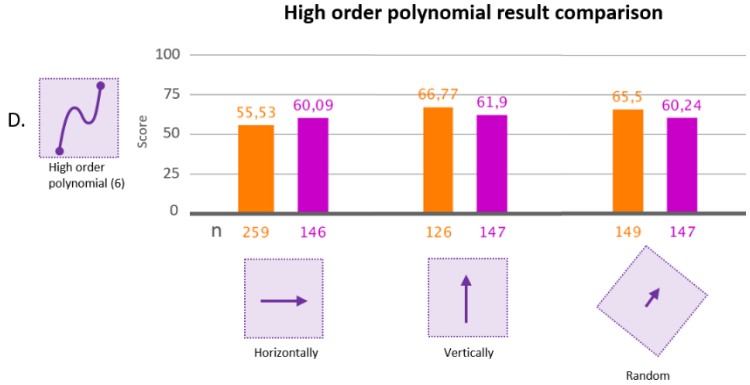


Figure 35: Comparison graph to analyse the differences between the sets of the different oriented high-order polynomial paths. Orange for input, purple for output. The number under the line represents amount of cases (linked to bar width). The bar height represents average score of this case.

only has an output bar, since there are no cases with a static point as an input point for this research. In figure 35, a deeper subcategory is explored, that of the differently oriented polynomials within the 2-D category. They do seem to perform relatively equal, although a slight advantage for horizontally developing polynomials seems to be present.

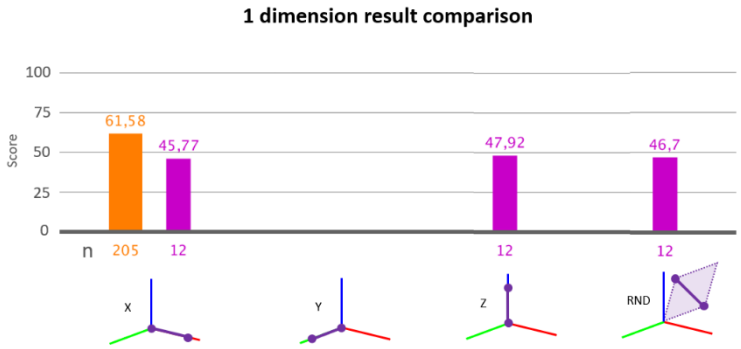


Figure 36: Comparison graph to analyse the differences between the 1 dimensional paths. Orange for input, purple for output. The number under the line represents amount of cases (linked to bar width). The bar height represents average score of this case.

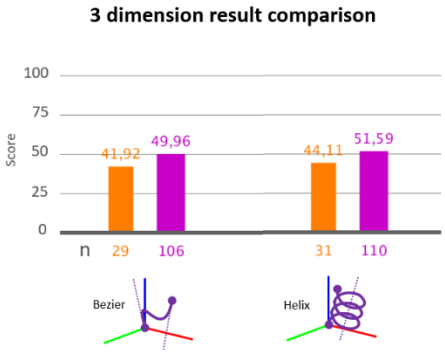


Figure 37: Comparison graph to analyse the differences between the 3 dimensional paths. Orange for input, purple for output. The number under the line represents amount of cases (linked to bar width). The bar height represents average score of this case.

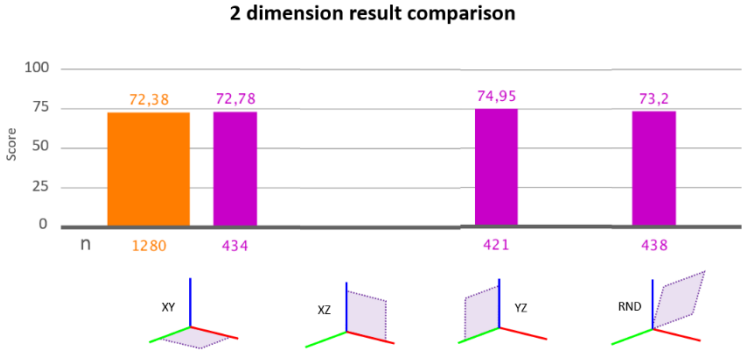


Figure 38: Comparison graph to analyse the differences between the 2 dimensionally oriented paths. Orange for input, purple for output. The number under the line represents amount of cases (linked to bar width). The bar height represents average score of this case.

Figure 36 summarises how the different 1 dimensional cases compare. Note that for the y-axis, there are neither input cases nor output cases. This is the result of the selective classification, where duplicates were removed as much as possible. Please see appendix B8 to read more about this. Figure 37 shows the two 3 dimensional subsets, where the output curves have a slight improvement over the input curves.

Finally, figure 38 displays how the 2 dimensional cases compare. Again, no data was generated for the XZ plane, as a result of eliminating duplicate cases.

Discussion

In this section, the results will be discussed and interpreted. First, the graphs as shown in the results section will be reviewed. Then, some more detailed information about some interesting cases will be highlighted. Finally, recommendations and suggestions for follow-up work will be given.

Discussion about the results

In figure 25, the centroid clearly lies within the bottom left side of the point cloud. This indicates that relatively more good to average solutions were found than average to bad solutions. At the far left bottom, a larger space between the diagonal and the cloud can be seen. This means that for really good scores, the average stayed a bit higher and didn't come down as well. Figure 27 shows similar results, but one interesting fact here is that the best scores were on average found with lower run times. This could be explained by the fact that with a relatively well-scoring random starting generation, the optimiser can iterate through the generations more easily, which is substantiated by the fact that relatively more low uniqueness was found near better scoring functions (figure 26). On the other hand, there are more low scoring solutions than there are high scoring ones, so naturally more density is expected towards the low end of the scores in the uniqueness graph.

In figure 28, some clearly different bands can be observed. Upon inspection of the indices of these bands, the well scoring cases seem be grouped around 3 cases, polynomials, semicircular shapes and spirals. In figure 34, this is also seen from the bars. Only the circular shapes with an angle larger than 2π seemed to be scoring significantly worse. From figure 30 and 31 however, the worse scoring cases seemed to be also grouped around 3d shapes.

The fact that circular shapes or round shapes in general seem to perform good, makes sense. The nature of the origami structures as approached in this paper all contain a rotating planes in the base pyramid. One can imagine that transforming this motion with multiple additions into another rotating motion could be easier than making straight line mechanisms out of them. In figure 39, we can see a solution for a linear input motion with a linear path distribution and a semicircular output motion with a linear path distribution, with its plane perpendicular to the input paths direction. With a result of 72.44, the optimiser approached the curved output relatively well, but the input motion still is a circular shape, although located in the middle of the input line, as that is where this path would get the highest score. We do see though that the algorithm tried to make the circular shape a compound shape, by using 2 tristructs for the input arm, and a single one for the output arm. This is quite interesting, as we know that with more tristructs, more compound and complex shapes can be achieved.

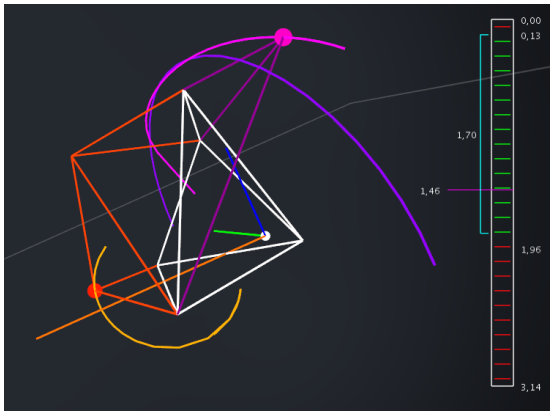


Figure 39: Linear input to a semi-circular output, both with a linear path distribution. Score of 72.44.

Moreover, figures 32 to 38 show us that there is quite some distinction between cases, but that mostly it doesn't matter too much if a shape is an input or an output path, as most of the bars have a similar height within one case compared to other cases. This can be seen in figure 33 for example, where it becomes clear that accelerating curves were significantly easier to solve for than for decelerating curves. Accelerating curves performed even better than constant curves.

Figure 32 shows us that the bigger the data set, the closer the input and output scores for one subcase seem to become. The large amount of cases in the 2-D domain originates from the fact that there are 5 subcases for each of the plane orientations (see figure 20), and for the high order polynomial subcase, there are 3 more subcategories. In other words, the 2-D domain, and specifically polynomials might have been a bit overpopulated in the dataset. This might explain why the bars for 2-D situations always seem to be more equal than in the other cases, where more difference between input and output variants of the same curve can be seen. The 3 subcategories for polynomials however seem to perform quite equal, which means that no matter the orientation of the polynomial, the optimiser handles it equally well.

Highlighted cases

An interesting case that has been quite well optimised is the one where 2 half-circles were placed on perpendicular intersecting planes, see figure 40 and figure 41. With a score of 43.94, this solution is quite good, as can be seen from the fact that both the input line and output line are followed quite closely. Also, note that in figure 40, a highly elegant solution is found using only a singular tristruct for both the input and the output side, which makes this solution quite robust.

The same problem has been solved differently in figure 41, where bye eye the curve seems to be followed more closely, but the score is almost 20% worse. Note that in this solution, the algorithm used 2 tristructs for the output curve, and still a single tristruct for the input curve. Also note that for both solutions, the range of the development angle as indicated by the bar on the right, covers quite a large portion of the total movement range. Some solutions only use a very small portion of this range, to almost generate a singular point that is quite close to the average shape of the input and/or output path. The examples from figure 40 and 41 therefore are quite elegant and physically well-constructable.

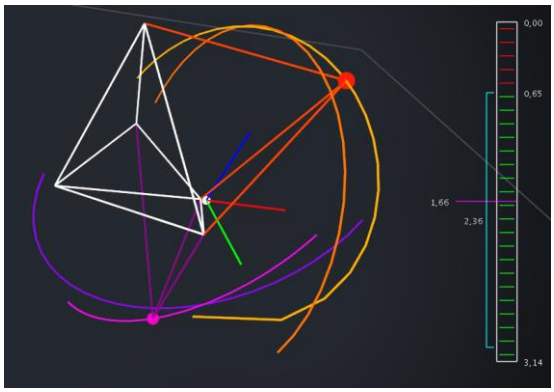


Figure 40: Score of 43.94.

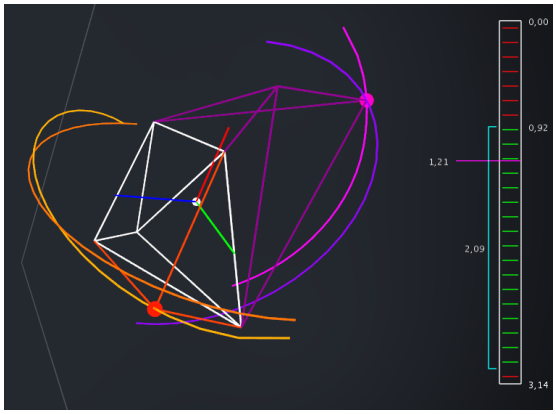


Figure 41: Score of 52.7.

In figure 42, a different transmission was optimised for using, and in figure 43, the algorithm process is visualised. Interesting to note is that the final generation apparently did not contain the overall best solution, which can be seen in figure 43C to be found around 3/4th of the process. But due to the recombination and mutation in every step, sometimes worse scores are the result. In general, the shapes seem to follow a asymptotical shape.

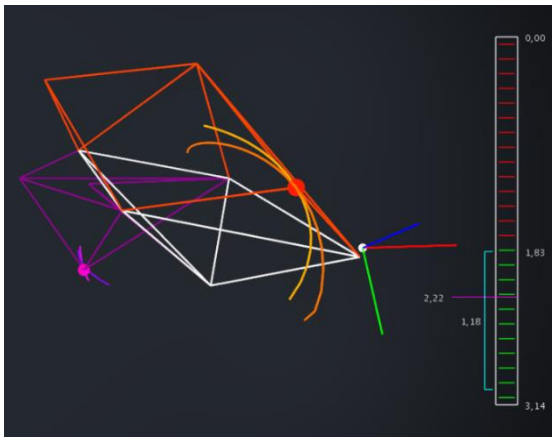


Figure 42: Score of 36.4.

Recommendations and future work

There are some factors and strategies used in the optimiser that could be revisited for further improvement. For example, the linear decrease of the mutation rate for the base pyramid and base coordinates could be changed to a different shape decrease, or even increase after long periods of minimal change. This simulated annealing type of strategy could further improve the evolutionary algorithm.

Another large influence is the score function. As described in the method section, the score function is based on the absolute distance of each point in the path with respect of each point in the path that is optimised towards. The distance however could be used in a non-linear way, for example with an exponential relation. This way, further away solutions will be punished harder than closeby solutions. Although this might improve optimisation speed, it might reduce precision solutions. Also, a hybrid score function could be adapted, where this relation within the score function changes over time or depends on the score that was previously found.

Also, various methods could be further explored to refine the way the solution are recombined and mutated. The principles shown in this research could be seen as a feasibility study that could lead into creating an actual software tool that creates kinematic transmissions using hinging rigid panel structures.

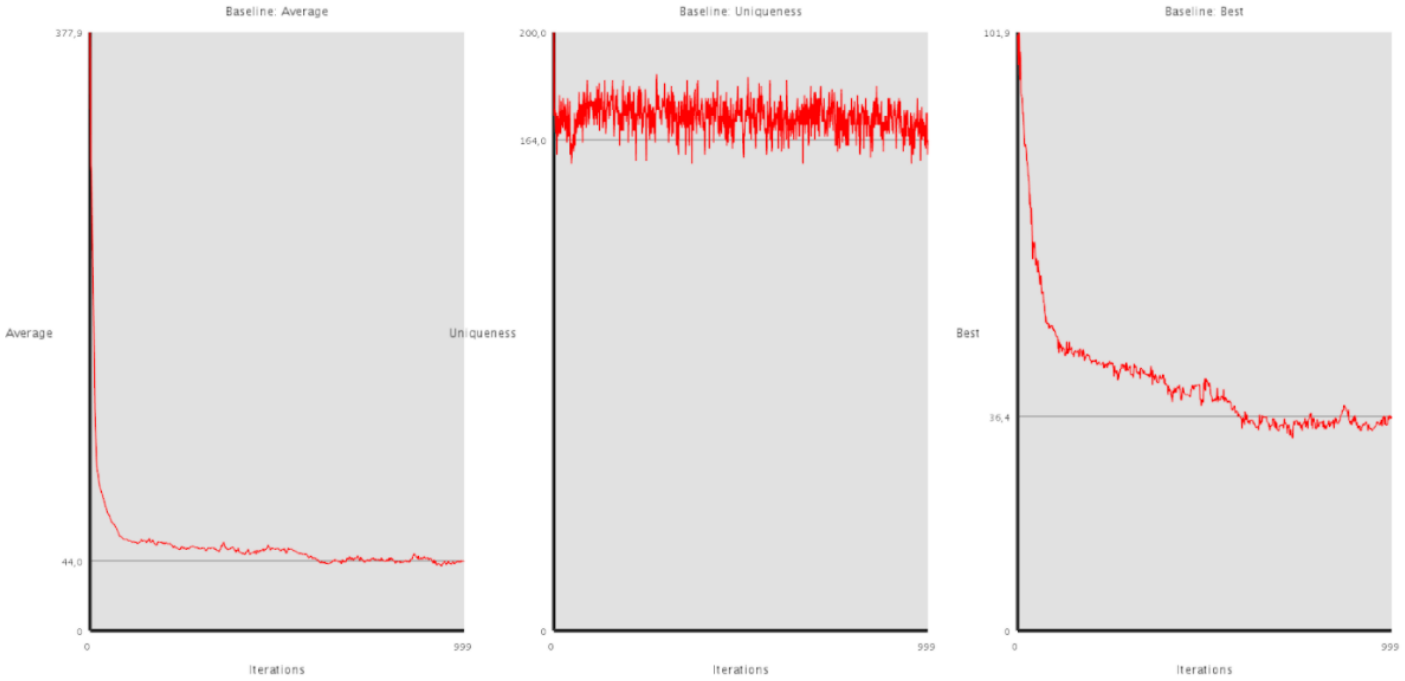


Figure 43: Optimisation process for a 1000 generations, population size of 200, 200 mutations per generation evolutionary algorithm. Graph A represents the average score versus the amount of iterations. Graph B shows the uniqueness of the population, and graph C displays the final score.

Conclusion

This research focuses on optimizing Hinging Rigid Panel Structures (HRPS) to be used as kinematic transmissions between various input and output relations, using an evolutionary algorithm. Hinging Rigid Panel Structures were chosen over conventional origami, because the latter is restricted to be always developable from a single flat sheet of planar material. A method was created using units which properties could later easily be programmatically altered by an algorithm. The HRPS consisted out of a base with a pyramid that has one side fixed to the world reference frame, and to specific places on this pyramid, sets of triangles called tristructs were placed. Using this method, a HRPS was modularly built up, and could be easily altered. A file format was created that allowed the software to easily work with these HRPS's.

To adequately analyse the effectiveness of the evolutionary algorithm, a classification was created (figure 20) that enabled the results to be split out in their different subclasses and to analyse which subclasses performed better. The evolutionary algorithm made use of generations with a set size, that were filled with HRPS's that attempted to solve the problem at hand, and were given a score each. From this generation, new HRPS's were created by selecting parents, recombining their properties and mutating them with a random chance and a random amount, also depending on how many generations already had passed. After a set amount of generations the best result was selected to be the final solution of the optimisation. The large data set that resulted from the classification earlier mentioned, was computed using crowd computation, where 17 contributors worked together to run the entire dataset in under 14 hours.

Found was that certain input-output relations were approached better than others, and interpretations were given on why these specific cases were easier to optimise for. Curved paths seemed to be easier approached by HRPS's than straight lines, which is a conclusion backed by literature and previous research. Circular motions or curved motions within a plane were better approached.

This research could certainly be used as a basis for further investigation into the field of kinematic origami. The results are promising, and with more research into optimising strategies and further work into subproblems like alternative parent recombination, the exact results can be improved even further. Hopefully, this research will lower the threshold for future researchers to start investigating the field of origami in motion more and more. It is not just the factual results that look promising and make this topic one that needs more active research. The beauty and simplicity of origami in motion is something that should have already been widely adopted in mechanical engineering.

Bibliography

[1] Oxford Space Systems. (2018, March 5). OSS OrigamiSat Concept [Video]. Vimeo. https://vimeo.com/258677350.

[2] Hollingshead, T. (2016, March 2). Tiny origami-inspired devices opening up new possibilities for minimally-invasive surgery. BYU. Retrieved from https://news.byu.edu/news/tiny-origami-inspired-devices-opening-new-possibilities-minimally-invasive-surgery.

[3] Tangible Media Group. (2016, December 6). aeroMorph [Video]. Vimeo. https://vimeo.com/194560714.

[4] F. Osório, A. Paio, S. Oliveira, "Kinetic Origami Surfaces", Conference on Computer-aided architectural design research in Asia, 2014.

[5] Kendall, C. (2018, November 15). Dental implant surface treatments: What you need to know. Nobel Biocare Blog. Retrieved from https://www.nobelbiocare.com/blog/news/dental-implant-surface-treatments-what-you-need-to-know/.

[6] E. T. Filipov, T. Tachi, G. H. Paulino, "Origami tubes assembled into stiff, yet reconfigurable structures and metamaterials", PNAS Early Edition 112 (40) 12321-12326, 2015, September 8.

[7] Jana, R. (2012, May 7). Why origami is a leading inspiration in design and science. ZDNet. Retrieved from https://www.zdnet.com/article/why-origami-is-a-leading-inspiration-in-design-and-science/.

[8] K. Miura, M. Natori, "2-D array experiment on board a space flyer unit". Space Solar Power Review, vol. 5, no. 4, p. 345–356, 1985.

[9] E. D. Demaine, M. L. Demaine, J. S. B. Mitchell, "Folding Flat Silhouettes and Wrapping Polyhedral Packages: New Results in Computational Origami", Proceedings of the 15th Annual ACM Symposium on Computational Geometry (SoCG'99), Miami Beach, Florida, June 13–16, 1999, p. 105–114.

[10] D. Dureisseix, "An overview of mechanisms and patterns with origami", International Journal of Space Structures, Multi-Science Publishing, 27 (1), 2012, p. 1-14.

[11] T. Tarnai. "Folding of uniform plane tessellations". In K. Miura, editor, Origami Science and Art. Proceedings of the Second International Meeting of Origami Science and Scientific Origami, pages 83–91, Otsu, Japan, 1994. Seian University of Art and Design.

[12] TUK Crafts. (2014, August 27). How to make a transforming 8-pointed ninja star (origami) [Video]. Youtube. https://www.youtube.com/watch?v=q1DE-2tkVj0.

[13] M. Kilian et al., "Curved folding", ACM Transactions on Graphics, 2008.

[14] Y. Nishiyama, "Miura folding: applying origami to space exploration", International Journal of Pure and Applied Mathymatics, vol. 79, no. 2, p. 269–279, 2012.

[15] K. Miura, "Utyuni hiraku Mahono Origami [The Magic "Origami" that Opens in Space] ", Kagaku Asahi, 1988.

[16] J. P. Gardner et all., "The James Webb space telescope". Space Science Reviews, vol. 123, p. 485-606, 2009.

[17] R. J. Lang, "Eyeglass Telescope", langorigami.com, https://langorigami.com/article/eyeglass-telescope/ (retrieved 27-04-2020)

[18] Z. You, K. Kuribayashi, "A novel origami stent". Summer Bioengineering Conference, 2003.

[19] R. J. Lang, "Airbag folding", langorigami.com, https://langorigami.com/article/airbag-folding/ (retrieved 28-04-2020)

[20] R. Hoffmann, et al. "A Finite Element Approach to Occupant Simulation: The PAM-CRASH Airbag Model", SAE-Paper 890754, International Congress and Exposition, Detroit, Michigan, 1989.

[21] A. Avila, S. P. Magleby, R. J. Lang, "Origami fold states: concept and design tool", Mechanical Sciences, vol. 10, iss. 1, (2019), p. 91-105

[22] F. Escrig. General survey of deployability in architecture. In F. Escrig and C. A. Brebbia, editors, Second International Conference on Mobile and Rapidly Assembled Structures II (MARAS 96), Seville, 1996. WIT Press

[23] R. J. Lang, "TreeMaker", langorigami.com, https://langorigami.com/article/treemaker/ (retrieved 28-04-2020)

[24] R. J. Lang, "ReferenceFinder Online", langoriagmi.com, https://finder.origami.tools/ (retrieved 28-04-2020)

[25] A. Terao, "OrigamiDraw", https://origamidraw.wordpress.com/ (retrieved 29-04-2020)

[26] T. Tachi, "Software", http://origami.c.u-tokyo.ac.jp/~tachi/software/ (retrieved 29-04-2020)

[27] T. Tachi, "Freeform Variations of Origami", Journal for Geometry and Graphics, vol. 14, no. 2, 2010, p. 203-215.

[28] M. Schenk, S. D. Guest, "Origami Folding: A Structural Engineering Approach", 5OSME, 2010.

[29] A. Ghassaei, "Origami Simulator", http://apps.amandaghassaei.com/OrigamiSimulator/ (retrieved 29-04-2020)

[30] A. Ghassaei, E. D. Demaine, N. Gershenfeld, "Fast, Interactive Origami Simulation using GPU Computation", 7th International Meeting on Origami in Science, Mathematics and Education, 2018.

[31] E. D. Demaine, J. S. Ku, R. J. Lang, "A New File Standard to Represent Folded Structures", Abstracts from the 26th Fall Workshop on Computational Geometry, 2016.

[32] M. R. Khan, C. L. Zekios, S. V. Georgakopoulos and S. Bhardwaj, "Automated CAD and Modeling of Origami Structures for Reconfigurable Antenna Applications", 2019 International Applied Computational Electromagnetics Society Symposium (ACES), Miami, FL, USA, 2019, pp. 1-2.

[33] R. J. Lang, "Tessellatica", langorigami.com, https://langorigami.com/article/tessellatica/ (retrieved 28-04-2020)

[34] Wolfram Research, Inc., Mathematica, Version 12.1, Champaign, IL (2020). https://www.wolfram.com/mathematica/.
[35] K. Fuchi, “Origami Mechanism Topology Optimizer (OMTO)”, MATLAB Central File Exchange, https://www.mathworks.com/matlabcentral/fileexchange/51811-origami-mechanism-topology-optimizer-omto/ (retrieved 03-05-2020)
[36] Processing 3, C. E. B. Reas, B. Fry, https://processing.org/.
[37] K. M. Vermeer, P. R. Kuppens, J. L. Herder, “Kinematic synthesis using reinforcement learning”, IDETC/CIE, 2017.
[38] I. Szita, “Reinforcement Learning in Games”, Reinforcement learning: State-Of-The-Art, p 539-577, 2012.
[39] J. C. Hoskins and G. A. Kramer, "Synthesis of mechanical linkages using artificial neural networks and optimization," IEEE International Conference on Neural Networks, San Francisco, CA, USA, 1993, pp. 822-J.
[40] J. T. Burger, “A basic introduction to neural networks”, University of Wisconsin-Madison, http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html (retrieved 06-05-2020)
[41] J. Heaton, “The number of hidden layers”, heatonresearch.com, https://www.heatonresearch.com/2017/06/01/hidden-layers.html (retrieved 20-04-2020)
[42] G. E. Hinton, S. Osindero, Y. The, “A fast learning algorithm for deep belief nets”, Neural Computation, vol. 18, no. 7, 2006, p. 1527-1554.
[43] M. Senthilkumar, “Use of artificial neural networks (ANNs) in colour measurement”, Colour Measurement, Principles, Advanced and Insdustrial Applications, Woodhead Publishing Series in Textiles, ch. 5, p. 125-146, 2010.
[44] B.T. Fang: "Trilateration and extension to global positioning system navigation", Journal of Guidance, Control, and Dynamics, vol. 9 (1986), pp 715–717.

Appendix A: Exploring Neural Networks

1. Neural Networks – a brief introduction

To gain understanding on how to implement neural network reinforcement learning to this study, research was performed into how neural networks function in general, with an evolutionary algorithm to evolve them as focal point.

Neural networks are build up out of unit elements, called neurons. Each neuron is part of a (often vertically depicted) layer. The first layer is called the input layer, the last one the output layer. All layers in between are called hidden layers. Each layer can have a different amount of neurons, and the choice of the amount of neurons in each layer is critical. According to J. Heaton [41], from the Sever Institute at Washington University, there are two rules of thumb about how many hidden layers one should use to prevent underfitting (using too little neurons and not being able to detect signals in complicated data sets) and overfitting (having so much information capacity that the amount of training data is not sufficient to train all neurons adequately). These two rules state:

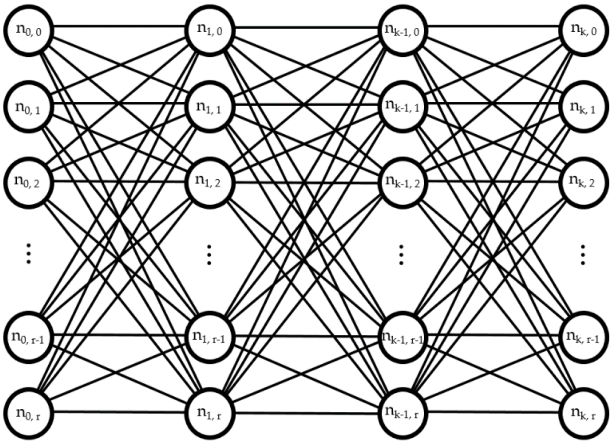


Figure 44: Graphical representation of a neural network with *r* rows and *k* columns.

The size of every hidden layer should:

- be in between the size of the input layer and the size of the output layer.
- be less than twice the size of the input layer.

The amount of hidden layers is another interesting property. According to J. Heaton, single hidden layer neural networks are capable of universal approximations, being able of approximating continuous functions. Since there was no good way to train multi-hidden-layer neural networks at that time, this was the standard up till 2006, when Hinton, Osindero and Teh proposed that using multiple hidden layers could be done, creating so-called deep neural networks, with their new "fast, greedy algorithm". They showed examples of being able to read handwritten digits using a three hidden layered neural network [42].

In short, one can conclude that using zero hidden layers, a neural network can only be able to represent linear separable functions or decisions. With one hidden layer, it will be able to approximate any function that contains continuous mapping in between two finite states. With 2 hidden layers, it will be able to represent any smooth mapping to any accuracy, and with more than two hidden layers, it is able to learn even complex representations, by J. Heaton described as automatic feature engineering. This is something that is attractive to use when dealing with the complexity of three-dimensional origami kinematics.

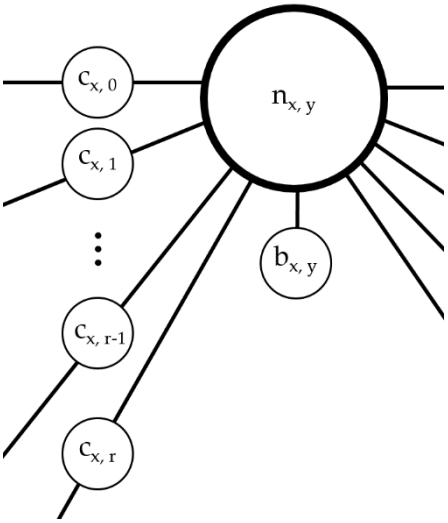
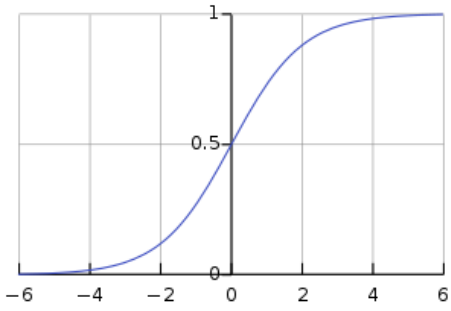


Figure 45: Graphical representation of a single neuron in a neural network. Bias value *b* and input connection values *c* are depicted.

Now, a brief explanation will follow on how neural networks work in detail. Please use figure 44 and 45 as a visual reference. In a neural network, the neurons in column 0 are the so-called input neurons. They are assigned a quantified value

from the physical problem. Every neuron stores one value, for the non-input neurons that is calculated as the sum of all input connections multiplied by their respective source neurons values. Every neuron has *r* inputs which are to be multiplied with their corresponding connection values *c*, but also, every neuron has a bias value *b* that is added to this sum before its value is calculated by putting this result in a so-called activation function. Often, a Sigmoid function (figure 46) is used, a curve to map all values back into a reach from 0 to 1. The neuron value, which will be the output for the next layer, can therefore be described by:

$$n_{x,y} = \frac{1}{1 + e^{\sum_{i=0}^{r}(n_{x-1,i} * c_{x,i}) + b_{x,y}}}$$



$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^{x}}{e^{x} + 1}$$

Figure 46: Sigmoid function, squeezes values back into a range of [0, 1].

The entire network is calculated in a forward manner like this, starting at the input layers and calculating all layers consequently until everything is updated. Once this is done, the output layer neurons will have values between 0 and 1. The highest valued neuron in the output layer will represent the decision that the neural network makes.

Now the basics of a neural network are understood better, it is important to know how to potentially deploy these neural networks to reach our goal if this strategy is choosen: enabling the program to optimise origami structures to reach certain input-output motion relations, in other words, function generation. It is important to note that all learning of a neural network boils down to nothing more than deciding what the exact numbers in all the thousands of connections should be. Neural networks are designed to be able to learn how to solve problems by finding the best possible values for all these thousands of properties. There are generally three different ways to approach neural network learning [43]: Supervised learning, Unsupervised learning and Reinforcement learning.

In supervised learning, one needs to feed a neural network with examples and solutions that correspond to these examples. This enables the computer to compare the result from its network to the actual solution. An often used strategy to adjust values in a neural network using these example solutions is called back propagation. This set of known solutions with problems, the so-called training data, is not available in the case of the origami transmission generation, since the best solutions are not known analytically.

For unsupervised learning, there simply is no solution and no feedback [43]. The neural network must organise its connections and outputs itself, based on random decisions.

Reinforcement learning is a special case of supervised learning. Where the supervised learning normally uses training data, the reinforcement learning uses a score function, which in fact is some sort of numerical feedback to evaluate how good the networks output is with a certain input.

One of the ways to evolve the neural network and train it, is using an evolutionary. A generation should be created with a set amount of neural networks. They all should be used to generate an origami structure to attempt solving the problem at hand. Next, they all should be given a numerical score using a to be determined metric that represents how good a solution is (some numerical comparison of the output motion of an origami generated by the neural network and the asked output motion). Based on the output score, every neural network in the generation gets a chance to propagate, a chance to live on in the next generation. The better the output score, the larger the chance it gets to be selected to propagate. Next, a to be determined number of neural networks will be selected (the chance of being selected being the chance every neural network got from the scoring) to be the parent of a new neural network. Conventionally, two parents are selected for each offspring neural network, but this can also be a different number. From the connection values and bias values these neural networks store, a recombination will be generated and a new offspring neural network is created. It is important that there must be a chance for the new neural networks to have random mutations as well, just like in nature.

This process of creating a new neural networks is repeated until the generation size is met. Once the entire new generation is created, the entire process will repeat itself. The amount of generations would have to be determined empirically, and investigation will have to be performed in order to choose adequate generation sizes, mutation rates, amount of parents and maybe most importantly: an algorithm that determines how multiple neural networks are recombined into one new neural network. An advantage of using an evolutionary algorithm to develop neural networks that are good in creating origamis to couple motions is that there can be

solutions that cannot be easily transformed into each other. In other words, their solution space might not be connected through conventional heuristic approaches. With an evolutionary algorithm, solution approaches that seem inferior in early generations are still able to propagate through their lower chance of being selected, and in later generations might be of critical influence into a final neural network that could solve the problem. The downside of this strategy is the computational complexity it requires and the relatively architecture.

2. Neural Networks – programmatical setup

To explore the complexity of programming neural networks in Processing, a test program was written to setup the infrastructure behind neural networks and discover possible problems in an early stage. In this section a brief explanation of the classes and methods to do this will be presented.

A class called *Neuron* was created to store the value of the neuron, its bias value, a list of input connection values and information about the amount of input connections and the layer which this neuron is part of. Instances of this neuron-class were used in a matrix within another class called *NeuralNetwork*. This class also stores information about the amount of input neurons it has, the amount of output neurons it has, the amount of neurons per hidden layer and the amount of hidden layers. While setting up this matrix, which is a direct representation of the neural network as depicted in figure 47, the neural network is being calculated through, using the activation function (squeeze function) as described earlier. This calculation is done in a loop, and done "chronologically", top-down and from left to right, in order to guarantee correct use of all the previous layers. The entire class structure as described above, including graphical methods to display the network as depicted, takes up a total of 160 lines of code, which seems quite compact.

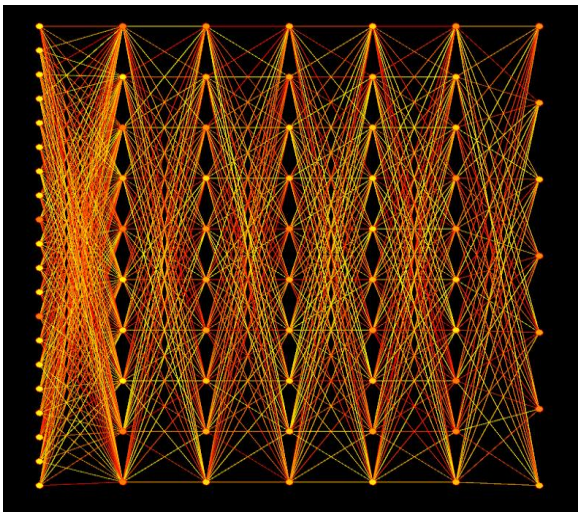


Figure 47: Graphical representation of a neural network with 20 input neurons, 7 output neurons, 5 hidden layers and 10 neurons per hidden layer. The yellow-red gradient represents the value of the connections between -1 and 1.

Concluding, it is quite simple to get the neural network to be characterised programmatically, which means it could be used in the research.

Appendix B: Details about HRPS

1. Degrees of freedom of an HRPS

An important property of the chosen structure addition method is that the single degree of freedom is always preserved, no matter how many TriStructs are added. In this appendix, the reasoning behind the formula that was given in the method section will be elaborated upon in more detail:

$$DOF = 6 \bullet (5 + 3 \bullet n) - 3 \bullet (8 + 5 \bullet n) \ - (5 + 3 \bullet n) = 1$$

This formula can easily be understood when an explanatory image is used, see figure 48. The model is built-up out of rods that are connected together via ball-joints, which all have 3 degrees of freedom (all rotations but no translations), and therefore also 3 constraints. When only the base pyramid (gray) is considered, the structure has 1 base plate (GND) and 5 bodies. Each of these 5 bodies has 6 degrees of freedom when unconstrained, so a total of 30 degrees of freedom are counted. In the system, there are 8 ball joint connections visible (some ball joints connect to multiple rods, each of these connections counts as one joint), and therefore degrees of freedom are taken away. Subtracting these from the amount of degrees of freedom results in a system with 6 degrees of freedom. Using this method however, all the rods can still rotate freely around their own longitudinal axes, which in the real system is not the case. The last term in the formula accounts for these free motions that have to be ignored, ending up with a single degree of freedom. When adding arms with several tristructs to the system, the mechanism still holds the single degree of freedom. Each tristructs brings 3 bodies, but also brings 5 joints to the equation, as is represented with the *n*-terms in the formula. For a total of 2 additions as depicted in figure 48, the formula equates back to 6*11 – 3*18 – 11 = 1. Also, if you simply work out the equation, n disappears altogether and the equation always equals 1:

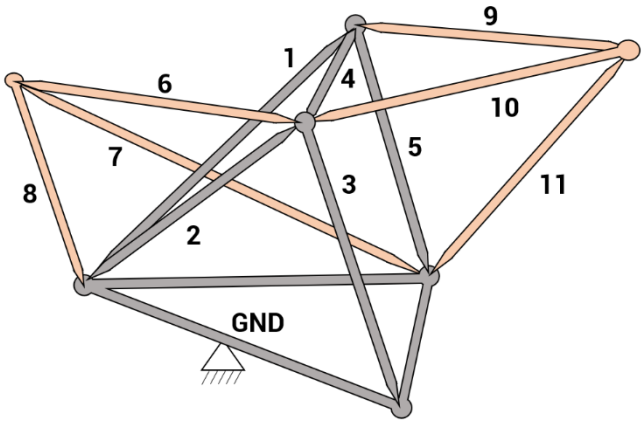


Figure 48: A possible HRPS with a base pyramid (gray) and two arms, each 1 tristruct long. Filling in the formula presented, proves the structure will always be restricted to 1 degree of freedom.

$$\begin{aligned} DOF &= 6 \bullet (5 + 3 \bullet n) - 3 \bullet (8 + 5 \bullet n) \ - (5 + 3 \bullet n) \\ &= 30 + 18 \bullet n - 24 - 15 \bullet n - 5 - 3 \bullet n \\ &= 30 - 24 - 5 + 0 \bullet n \\ &= 1 \end{aligned}$$

2. Multilateration in code

Performing multilateration in code is done using a vectoral procedure, following a well-known method by Fang [44], which boils down to finding the intersections of three spheres, which always results in 2 or less points. The fact that 2 points will often be found means that the code needed a way to decide which point to return. This is programmatically done using an extra statement when the trilateration is requested. A simple boolean value switches the method to return the first or the second solution. In case of the base pyramid, which is built up using 1 base triangle (and after finding the second triangle with the developability angle, the third and fourth triangles can be found using a single trilateration), the return point is always the same one, i.e. the boolean is always in the same state. For the input arm, after observation, it was found that this boolean should always be set to *true*, the output arm always needs the opposite state for it to work. This results out of the way the coordinate systems were set up and the order of points that are being fed in the trilateration method. Figure 49 provides a graphical overview of what a trilateration using 3 spheres looks like.

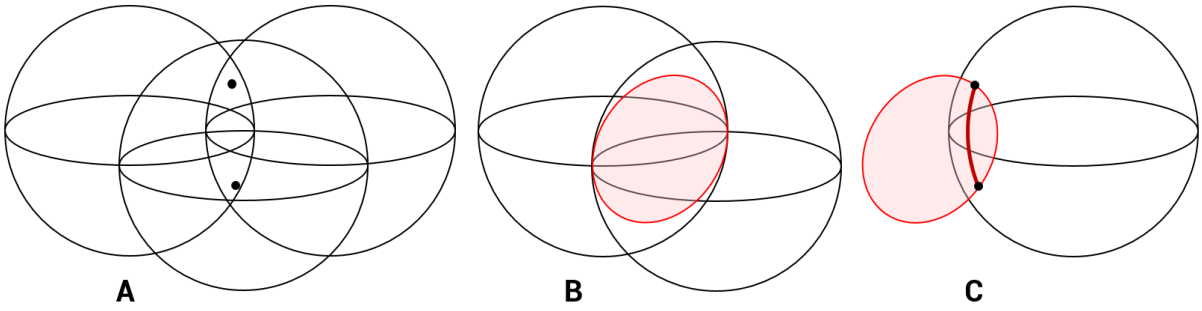


Figure 49: It can be quite hard to see where the intersection points of 3 spheres lie (A). By first finding the intersection circle between two of them (B), it becomes more obvious what is happening, and why there are almost always 2 intersection points (C).

3. Checking solvability of the HRPS

The programmatical HRPS class also contains functionality to remove the latest added tristructs, to deep copy the system, to recalculate the structure in case any length value in the system has been changed, and a set of methods to change to different view options. One of the most important functions however is the option to generate the paths all planes endpoints travel along when iterating through the development of the HRPS. By setting the developability to 0 and increasing it step-wise with a set amount of steps (the movement resolution), every step generates a set of points that are the midpoints of all the planes and the endpoint of the 2 arms. These points can be stored in lists.

It is important to note that some developabilities for certain HRPSs result in unsolvable geometries. More specifically, the trilateration of certain tristructs, which happens through finding the common points of 3 spheres (see figure 49) will be unsolvable. If this trilateration fails while recalculating one of the developability steps, a programmatic flag is raised and a list is created that indicates for which developability steps the structure is physically possible. Physically, this would mean the system can be created without stretching any of the planes out of its exact shape as defined. This (un)creatability does not take into account collisions of different planes however. Collision detection is built in the program as well, see appendix B4.

In order to display and calculate the motion, the program needs to find where the longest continuous list of solvable development steps is. In order to do this, first, the entire development range is tested. Then, the longest range of solvable steps is taken, and again run in a higher resolution. Note that this operation will be performed thousands of times while optimising the structure using algorithms, therefore it is of utmost importance that the algorithm filtering out the longest continuous section of developable steps is coded very efficiently. See figure 50 for more information about how this was done.

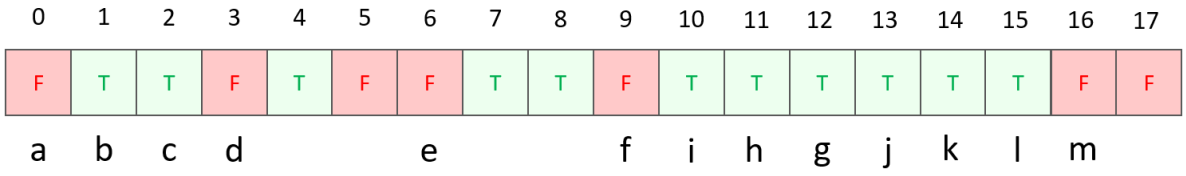


Figure 50: In order to minimize time it takes to find the longest continuous section of possible developability steps, a special method is applied. In this figure, the letters represent the search order the code applies to the array with booleans that represent the various developability step solvabilities. It starts by checking the first element, and once it has found a section with continuous positive values, it will step through the array by that amount. If it finds a negative value at the landing position, it will just skip one, but if it finds a positive value, the section will be explored, as depicted above. This way, large sections can be skipped, and only a fraction of the array needs to be analysed. Obviously, in the image above the sections are very short and so the efficiency benefits are small, but if the array contains large sections that are still too small, they will be skipped entirely.

4. Collision detection triangles and line sections in 3D space

To represent an actual physically possible structure, collision detection was implemented, to prevent triangles from glitching through each other and creating infeasible solutions. This was done by checking each side of every triangle against each triangle in the system, excluding cases where a line is part of the triangle or a line directly connects to a corner of the triangle.

It was done by using a series of vector calculations to first find the intersection point of the extended line and the extended plane, then check if the point on the plane that has been found lies within the triangle that is being checked. See figure 51 for a graphical explanation of this method. To simply do this, vectors are being created between the found point and the corners of the triangle, called point *k*. The 3d line equation for line AB is used to check if this point lies between A and B, and if so, the check is performed if the summation of the three angles between the three vectors from each point of the triangle to the newly created point *k* is less than 2π radians.

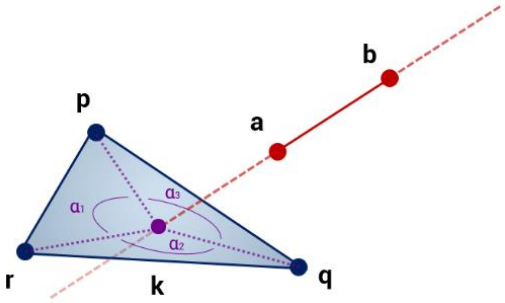


Figure 51: Line AB and triangle PQR in 3d space intersect if and only if 2 conditions are met. First, the variable that represent position along line AB needs to be within the range of 0 to 1, i.e. the plane intersects the line between point A and B, and secondly, the sum of α_1, α_2 and α_3 has to be smaller than 360 degrees or 2π. In this case, the first condition is not met, but the second is. Therefore the line section does not intersect with the plane.

5. Prescribed Displacement Patterns Over Prescribed Curves

As described in the *Method* section of this paper, the displacement pattern over the to be followed paths is not always equal to the one that would naturally follow from the formula or method that underlies the various motion curves. While most shapes (line, circle, spiral, helix) have an analytical way to find equally spaced points, Bezier curves and high order polynomials are more difficult.

Using Casteljaus method, one can determine the length of a Bezier curve using infinitesimal small sections. If this full length of a path is known, it is not difficult to find a distribution over this curve by first calculating how large each piece of the distribution should be (depending on the distribution and the just found path length), and then another iteration can be performed to find a piece of the path with requested length. Doing this for each of the points of the distribution, the location of all points over this curve can be found. For polynomials the same strategy can be applied, only using an integral to determine path length.

6. Bezier Curves

One of the subclasses of the 3d category of the classification comprises the widely used quadratic Bezier curves were used. These type of curves are defined by two anchor points and two control points. To get the exact shape of the spline that is defined by them, one can use Casteljaus method. This method describes a way to get an exact point on the spline, and is even applicable for 3D situations, as explained in figure 53. However, for defining the entire spline as a set of a set amount of points, an iteration needs to be performed, to get all the points through Casteljau constructions, as depicted in figure 52.

One might want to add more construction points, to create more complex paths. To allow this, multiple quadratic Bezier segments can be connected, and their order and connectivity is stored in a list. To guarantee continuity, the two control points on either side of the connecting point, need to be mirrored through the point that connects the two segments. More precisely, the direction needs to be mirrored but the distance to the control point doesn't influence continuity. As the spline always is tangent to the line between the control point and the end point, it is always continuous as long as the connection point and the two control points on either side lie in one line.

In order to compare a spline with a set of points that the midpoint of a plane of the HRPS has moved through, this compound Bezier curve set needs to be split into an equal amount of points as the planes movement point list contains. This is done with Casteljau over the various segments a compound Bezier curve exists out of. Note that the distance between the consecutive points found using Casteljaus method in figure 53 are not spaced equal-distance along the spline. Since X in figure 52b is increased linearly from 0 to 1 and there is no linear coupling between this variable X and the position on the Bezier curve, the distance between the points will never be equally spaced over the curve. This is an issue since the points will be used to compare the HRPSs movement paths to. The distance between them corresponds to the velocity the HRPS planes midpoint moves with. To solve this problem, the spline can be heavily oversampled, then this samples can be selectively erased again, to end up with samples that are all equal distance away, or at least substantially more equally spaced. Of course resolution trade-offs have to be made here to keep the efficiency of the code high.

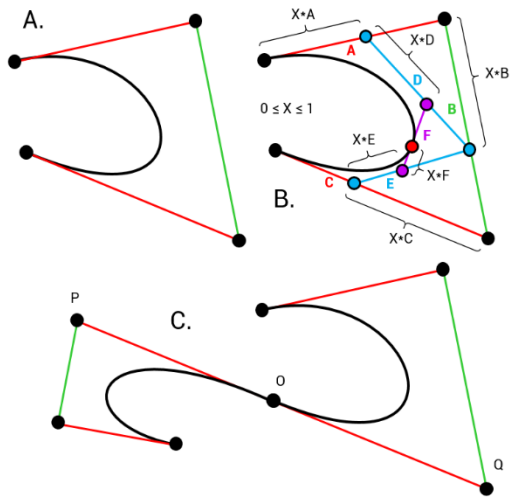


Figure 52: Splitting lines A, B and C in equal ratios gives 3 new control points. Connect these, and split them in the same ratio again, and again connecting these 2 points up gives a single line, that, if being split again in the same ratio returns a point exactly on the spline. In more exact terms, choosing value X in image B to be anywhere between 0 and 1 returns a point on the spline. In subfigure C, point P is a mirror of point Q, through point O. The direction is important, but the length of OP can be any length.

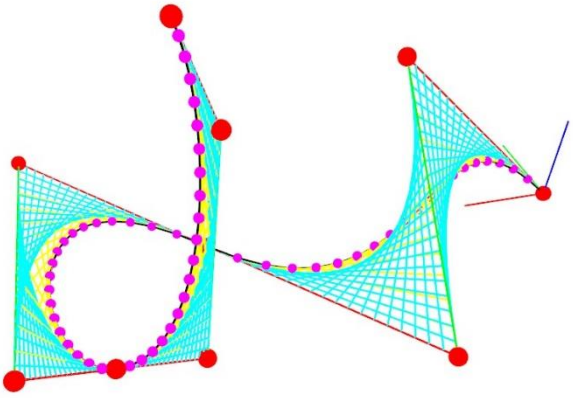


Figure 53: A spline consisting of three quadratic Bezier curve splines in 3D, divided in 60 pieces. In this case, 57 Casteljau constructions are drawn, since the 4 start and end points of the 3 curves are known. Continuity is guaranteed since the connecting point and the surrounding control points lie on a straight line.

7. Editing Curves With A GUI

To accommodate all options and sliders for adjusting the curves, a collapsible menu structure was used. First, one chooses the *Shape* or the V*elocity* (displacement distribution) button, depending on what needs adjustment. If one chooses the *Velocity* option, a second selection menu appears (Figure 54a), in which the type

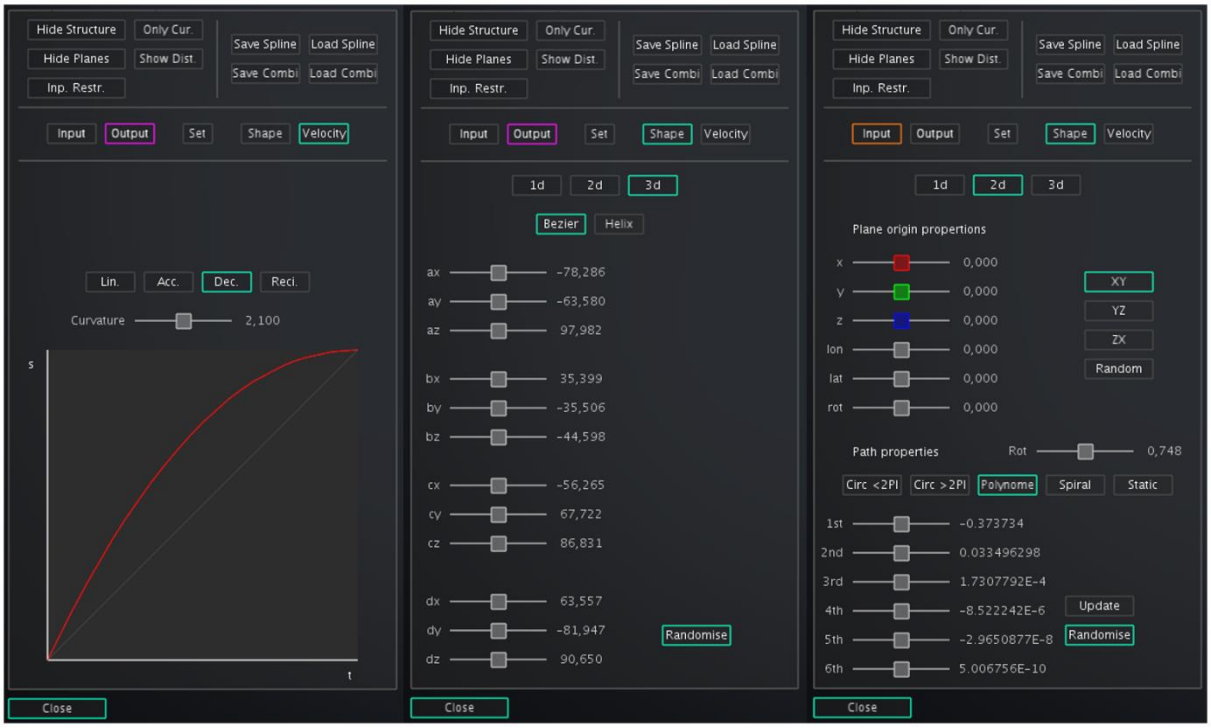


Figure 54: A: Manipulating the curves displacement distribution to be of decelerating nature. B: A randomised parameter adjustment slider deck to set a Bezier curve by its 4 anchor and control points in 3D space (see appendix B6). C: A randomised parameter adjustment slider deck for polynomials.

of distribution curve can be chosen, with an adequate slider to provide adjustability. If however the *Shape* button is selected, one gets to choose from a dimension menu, and under the different menus are different options and sliders to set and adjust all necessary parameters. Near the top of this pop up menu, there is the option to *Set* the currently created motion path and distribution to either the input or output curve, again depending on which of the *Input* and *Output* buttons is currently selected. Also, there are some visual options and saving/loading options.

8. Breakdown of the 1545 cases

The classification as shown in figure 20 has *136* subcategories. If this number would be squared, you end up with over *18* thousand cases. This would result in a very big data set, and a lot of duplicates being introduced. For example, the kinematic transmission of a straight line in x direction to a straight line in x direction would be the same as a straight line in y direction to a straight line in y direction, if the paths were followed in relatively the same direction with respect to each other. This minefield of duplicates and unnecessary combinations has carefully been navigated, and a reduction was applied, leaving less than 10% of the original amount of cases necessary, while still fully exploring all mentioned transmissions. The full dataset was approached as being 11 separate cases. These are explained in the section below:

- 1d to 1d transmissions (36 cases)
  - For the input path, only a line along the x axis is considered. This has 3 possible path distributions, as a reciprocal distribution is used for outputs only.
  - For the output path, three options are possible. A line parallel to the input, perpendicular to the input (so either along the y axis or the z axis), and a line on a random plane. Each of these three options has 4 possible displacement distributions, so 12 possible output lines are considered.
  - Multiplying the 3 input options with the 12 output options results in 36 options.

- 1d to 2d transmissions (153 cases)
  - For the input path, only a line along the x axis is considered. This has 3 possible path distributions, as a reciprocal distribution is used for outputs only.
  - For the output path, 3 planes are considered, each plane being able to carry 4 output motions (semi-circular, rotational, spiral, polynomial) with 4 distributions, and the static option has only 1 motion, as different displacement options do not change that single point. This results in 3 times 16 plus 3 times 1 equals 51 options.
  - Multiplying the 3 input options with the 51 output options results in 153 options.

- 1d to 3d transmissions (24 cases)
  - For the input path, only a line along the x axis is considered. This has 3 possible path distributions, as a reciprocal distribution is used for outputs only.
  - For the output path, 2 shapes are considered, each having 4 possible distribution functions. This results in 8 possibilities.
  - Multiplying the 3 input options with the 8 output options results in 24 options.

- 2d non-polynomial to 2d non-polynomial transmissions (351 cases)
  - For the input path, only a single plane parallel to the XY plane is considered. On this plane there are the 3 options that are non-polynomial and not static as it is an input. Each of this 3 options has 3 possible displacement distributions, so a total of 9 input paths is possible.
  - For the output path, 3 planes are considered, each plane being able to carry 3 output motions (semi-circular, rotational, spiral) with 4 distributions, and the static option has only 1 motion, as different displacement options do not change that single point. This results in 3 times 12 plus 3 times 1 equals 39 options.
  - Multiplying the 9 input options with the 39 output options results in 351 options.

- 2d non-polynomial to 2d polynomial transmissions (324 cases)
  - For the input path, only a single plane parallel to the XY plane is considered. On this plane there are the 3 options that are non-polynomial and not static as it is an input. Each of this 3 options has 3 possible displacement distributions, so a total of 9 input paths is possible.
  - For the output path, 3 planes are considered, each plane being able to carry 3 output motions (the three different orientations for the polynomials). Each has the option of the 4 displacement distributions. This results in 3 times 12 equals 36 options.
  - Multiplying the 9 input options with the 36 output options results in 324 options.

- 2d polynomial to 2d non-polynomial transmissions (351 cases)
  - For the input path, only a single plane parallel to the XY plane is considered. On this plane there are the 3 options that are the three different orientations for the polynomials. Each of this 3 options has 3 possible displacement distributions, so a total of 9 input paths is possible.
  - For the output path, 3 planes are considered, each plane being able to carry 3 output motions (semi-circular, rotational, spiral) with 4 distributions, and the static option has only 1 motion, as different displacement options do not change that single point. This results in 3 times 12 plus 3 times 1 equals 39 options.
  - Multiplying the 9 input options with the 39 output options results in 351 options.

- 2d polynomial to 2d polynomial transmissions (108)
  - For the input path, only a single plane parallel to the XY plane is considered. On this plane, there is only 1 necessary polynomial to consider, because otherwise duplicates would be introduces. It has 3 possible displacement distributions, so a total of 3 input paths is possible.
  - For the output path, 3 planes are considered, each plane being able to carry 3 output motions (the three different orientations for the polynomials). Each has the option of the 4 displacement distributions. This results in 3 times 12 equals 36 options.
  - Multiplying the 3 input options with the 36 output options results in 108 options.

- 2d non-polynomial to 3d transmissions (72 cases)

- For the input path, only a single plane parallel to the XY plane is considered. On this plane there are the 3 options that are non-polynomial and not static as it is an input. Each of this 3 options has 3 possible displacement distributions, so a total of 9 input paths is possible.
- For the output path, 2 shapes are considered, each having 4 possible distribution functions. This results in 8 possibilities.
- Multiplying the 9 input options with the 8 output options results in 72 options.

- 2d polynomial to 3d transmissions (72 cases)
  - For the input path, only a single plane parallel to the XY plane is considered. On this plane there are the 3 options that are the three different orientations for the polynomials. Each of this 3 options has 3 possible displacement distributions, so a total of 9 input paths is possible.
  - For the output path, 2 shapes are considered, each having 4 possible distribution functions. This results in 8 possibilities.
  - Multiplying the 9 input options with the 8 output options results in 72 options.

- 3d to 3d transmissions (48 cases)
  - For the input path, 2 shapes are considered, each having 3 possible distribution functions, as a static point is not considered for inputs. This results in 6 possibilities.
  - For the output path, 2 shapes are considered, each having 4 possible distribution functions. This results in 8 possibilities.
  - Multiplying the 6 input options with the 8 output options results in 48 options.

- 3d to 2d static transmissions (6) cases
  - For the input path, 2 shapes are considered, each having 3 possible distribution functions, as a static point is not considered for inputs. This results in 6 possibilities.
  - For the output points, only a single option is possible, a static point.
  - Multiplying the 6 input options with the single output option results in 6 options.

This brings the total amount of kinematic transmissions to 1545 cases.