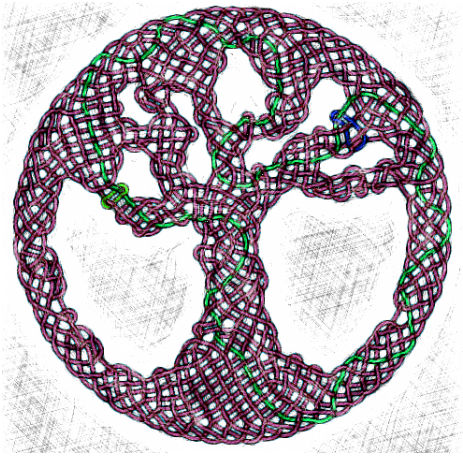
Bachelor Thesis

Transforming Objects on Images into Celtic Knotwork

Samuel Kopp

Supervisor: Dr. Cameron Browne



Thesis submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science in Data Science and Knowledge Engineering

at

Maastricht University
Faculty of Science and Engineering
Department of Data Science & Knowledge Engineering
Maastricht, The Netherlands

This chapter began with the quest of learning and understanding how consciousness emerges.
It continued with finding joy in solving problems not just alone but together.
And it ends with the discovery that fundamental processes found
in nature can lead to the emergence of beauty.

Acknowledgements

I wish to show my gratitude to my thesis supervisor, Dr. Cameron Browne who suggested this thesis topic and provided great guidance throughout this work. I also want to thank my parents who enabled these studies for me and Samiksha who was always with me in the last weeks. Finally I want to thank Dr. Steven Kelk who I give credit of formulating the mesh conversion as a graph matching problem and who reminded me of and taught me Integer Linear Programming.

Abstract

This thesis presents a method for transforming shapes within natural images into a visual art style based on Celtic knotwork. The method segments the image into regions which are filled with meshes consisting of triangular and quadrilateral elements. Centroidal Voronoi tessellation, a wavefront and a force-based smoothing procedure is applied to generate a point cloud with the right geometric properties. These points are then converted to a triangular and quadrilateral mesh structure. Analysis reveals which variation of the mesh generation leads to the most regular and symmetric knotworks. The results demonstrate that we can successfully create knotwork designs that resemble the look of objects extracted from images.

Contents

Chapter 1

Introduction

1.1 What is Celtic knotwork?

Celtic knotwork is a form of decorative interlace pattern that was developed and primarily used by the British and Irish Celts in early medieval times.[Bain, 1973] It was a key component of what was termed "Celtic art" and was taken up by artists in metalwork, monuments, manuscripts and books like the Book of Kells and the Book of Lindisfarne. The interlace as a decorative element can be found throughout many eras in a variety of cultures, such as the symbolic endless knot in Buddhism or as part of mosaics and wall paintings in arabesque Islamic art. The interesting part for this work is that Celtic knotwork is made of a simple, repetitive pattern: One or more closed threads are alternatingly woven over and under other parts of themselves.

(a) (b) (c) (d)

Figure 1.1: a) Celtic cross on a grave in Brompton Cemetery, London b) Decorated letter in the book of kells c) Close-up of a triquetra on one of the Funbo Runestones d) Geometrical ornament with interlaced knotwork border in a Quran, c. 1180

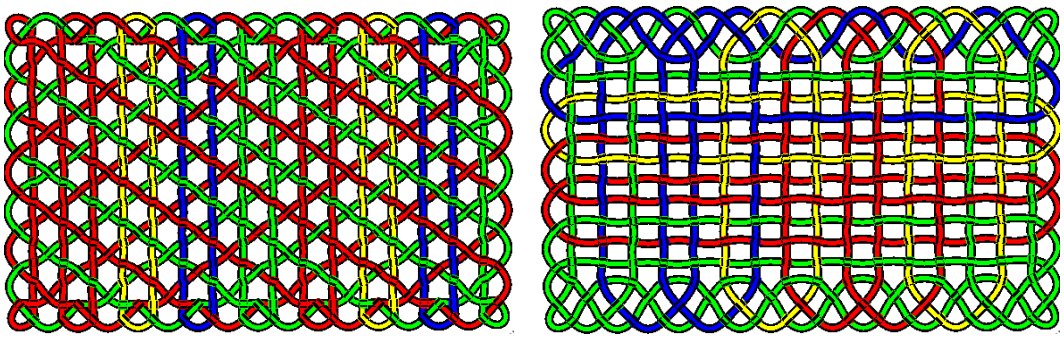
1.2 Motivation

Once these knotworks become more intricate and the number of crossings gets larger, most people will find it difficult to design one without the help of some sort of geometric procedure (The reader could at this point try to sketch one for themselves). Already in pre-digital times, methods like the grid-procedure have been developed to aid the artist in the design.[Bain, 1973] Today having a computer generating knotworks becomes useful as a tool for digital artists, game developers or even fashion and tattoo designers.[Whitehead, 2010]

1.3 Objective

A common method to construct Celtic knots is called the Mercat algorithm and it determines the shape of the knotwork based on a graph of nodes and edges.[Mercat, 1997] The objective of this work

was to develop a method to transform arbitrary objects on images into Celtic knotwork by generating such a graph. A type of graph that subdivides a geometric space into nodes, vertices and cells is called a mesh and the practice of generating a mesh is called mesh generation (or sometimes tesselation). To generate a knotwork with artistic quality, the mesh should incorporate a high degree of symmetry, repetition and self-similarity. This makes a mesh with cells of regular connectivity, similar cell sizes and angles desirable. The most common type of Celtic knotwork results from weaving the thread around quadrilateral cells, but also other types of polygons can lead to aesthetic and interesting interlace patterns.



(a) (b)

Figure 1.2: a) Knotwork woven around triangular cells b) Knotwork woven around quadrilateral cells

Ideally we would also like the mesh to be curved or directed similar to the shape it fills, such that our knotwork looks more dynamic and less like it was generated as a static grid.
Therefore the research question this work addresses are:
1) *Can we generate a Celtic knotwork that resembles the exterior shape and the internal features of a given object using image segmentation and mesh generation?*
2) *How can we optimize a mesh structure to incorporate aesthetic qualities such as regularity and symmetry?*
3) *Can we adapt the mesh structure to be aligned with the geometry of its enclosing region?*

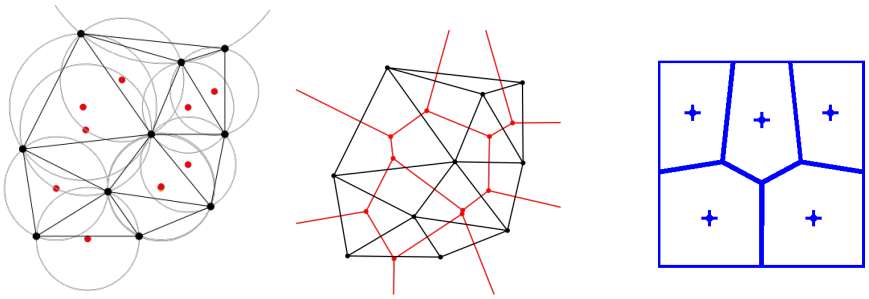
Chapter 2

Background

2.1 Mesh Generation

Mesh generation techniques often make use of a concept called **Delaunay triangulation** (after the Russian mathematician Boris Delaunay). A Delaunay triangulation connects a set of points to form triangles such that the circumcircle of each triangle does not contain any points which do not belong to its respective triangle. There exist many fast algorithms to compute the Delaunay triangulation, some of which are O($n$log$n$). [Aurenhammer, 1991] The dual of a Delaunay triangulation is the Voronoi diagram. A Voronoi diagram (or Voronoi tesselation) is a partition of geometric space into regions called Voronoi regions, where each region is generated from a seed point. A region consists of all points closer to its seed point than to the seed points of other regions. The process of moving a set of points such they become they centroid of its corresponding Voronoi regions is called **centroidal Voronoi tesselation(CVT)**. We can compute the CVT of a given set of points by approximating its Hessian matrix and using a Quasi-Newton method such as L-BFGS to minimize its energy function. [Liu et al., 2009] The energy function which is associated with the set of seed points $\mathbf{z}$ and their Voronoi tesselation V is formulated as:

$$\epsilon(\mathbf{z}, V(\mathbf{z})) = \sum_{i=1}^{N} \epsilon_i(\mathbf{z}, V(\mathbf{z})) = \sum_{i=1}^{N} \int_{V_i} \rho(x)\|x - z_i\|^2 dx \tag{2.1}$$



(a) (b) (c)

Figure 2.1: a) Delaunay triangulation with center of circumcircles marked in red. b) Voronoi Diagram with seed points marked in red. c) A Voronoi diagram where each seed point is in the center of its region (CVT).

In certain situations CVT based mesh smoothing has been shown to be superior to the classical *Laplacian smoothing*.[Wang and Du, 2005] Another common mesh smoothing scheme is to move mesh vertices based on physical forces. Dave Hale showed a method to mesh seismic images (for the purpose of flow simulation, oil reservoir characterization, etc), which models the mesh vertices as atoms that attract and repel its neighboring vertices. Minimizing the overall potential field of these atoms leads to a regular, crystallized lattice of points.[Hale and Emanuel, 2002] Similar approaches model the triangulated lattice as an elastic grid or a truss structure and try to find an equilibrium of its spring forces.[Persson and Strang, 2004][Soleng and Holden, 1998]
Advancing front methods start placing mesh elements from its boundary and advance incrementally towards the interior of a shape.[Zienkiewicz et al., 2005] They have the advantage that the generated mesh is often more aligned with its shape geometry. Itoh and Shimada presented a heuristic for the conversion of triangular to quadrilateral mesh elements that scores the edge alignment of the cells with a pre-defined vector field. [Itoh and Shimada, 2002]

2.2 Computer Generated Celtic Knotworks

We can find many examples of Celtic knot generators on the internet that let the user create a knotwork by placing knot elements on a grid-like surface.[Mueller, 2013] Detailed instructions for programming a simple knot generator were given among others by Andrew Glassner.[Glassner, 1999] Kaplan and Cohen showed how one can display Celtic knotwork using cubic Bezier curves. They also showed how to smooth the threads and attach images to the knotwork.[Kaplan and Cohen, 2003] In another paper, Kaplan and Cohen gave a method to decorate the surface of 3D objects with Celtic knotwork based on user defined 2D tilings of polygonal patterns.[Kaplan et al., 2004]
As part of the development of Canons font type "Glorious Fonts", Cameron Browne demonstrated how character sets can be filled and decorated with Celtic knotwork with a high degree of regularity. He applies a simpler version of the advancing front method to obtain a close to regular lattice that fits an arbitrary character shape. This lattice is triangulated and converted to a set of triangular and quadrilateral mesh cells which is then mapped to knotwork tiles.[Browne, 2003]
With a similar objective as this thesis, Denman et al. first asked the research question if a Celtic knotwork can be generated that captures the shape and internal features of an image. Their attempt iteratively moves a randomly initialized lattice of points into the direction of the centroid of the respective points Voronoi region weighted by the underlying pixel intensity to obtain a mesh that results in a higher point density in darker regions of an image. [Denman et al., 2019]

Chapter 3

Methodology

3.1 Proposed Method

The proposed method for transforming objects on images into Celtic knotwork consists of three main steps. The first step is to segment the object into distinct regions in order to capture the shape of its exterior and internal features. In the second step we generate for each region a lattice of points in regular distances to each other and form a triangular or triangular-quadrilateral mesh fitting into this region. To generate an initial regular lattice we either use a standard force based approach or Browne's wavefront method.[Browne, 2003] Then we triangulate this lattice and smooth the resulting mesh further using CVT. The triangular mesh elements are then converted into quadrilaterals based on geometric regularity and alignment with its enclosing region. We search for the optimal solution of this conversion using Integer Linear Programming(ILP). Finally, we apply the Mercat algorithm and form Bezier curves to display the resulting knotwork.

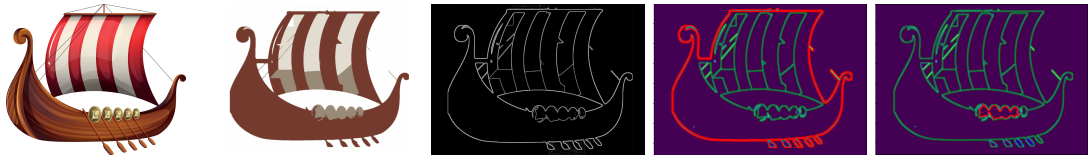
3.2 Object Segmentation

3.2.1 Contour Finding

To identify segments in an object, we first want to retrieve its external and internal outlines (contours). In image processing we call this procedure "contour finding". Initially the imagine is converted to a binary image using a threshold function.

$$binary(x,y) = \begin{cases} maxval & color(x,y) > thresh \\ 0 & otherwise \end{cases} \tag{3.1}$$

This allows a contour finding algorithm to find the borders between connected components of 1-pixels and 0-pixels (background) to extract the contours of the given binary image.[Suzuki et al., 1985] Due to noise and missing pixels, the actual contour lines in an image may be bumpy or have gaps. Applying a blur to the image can close these gaps and smooth the contour lines. Another optional step that helps to make the contour finding process more robust to noise is to filter the image with an edge detection algorithm such as CannyEdge detection. This leaves the image with just its edges (points at which image brightness changes sharply). A closed curve of continuous edge points directly corresponds to an image contour. Sometimes an image has too many fine details (hairs, rough texture, shading) and it becomes difficult to find the contour lines we are looking for. Then it helps to apply a clustering algorithm such as k-means to merge segments of similar color.[Ilea and Whelan, 2006]

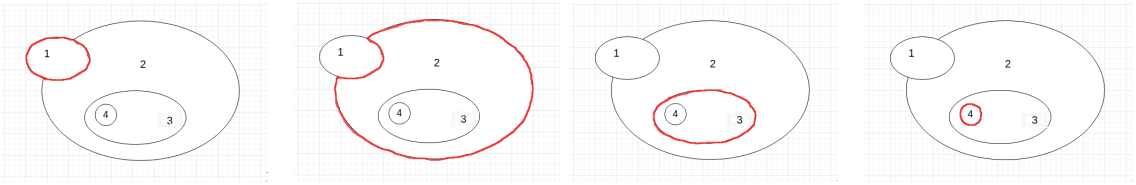


(a) (b) (c) (d) (e)

Figure 3.1: a) Input image b) Linear k-means clustering c) CannyEdge detection d) Outer contour highlighted e) One of the inner contours highlighted

3.2.2 Identifying Segments

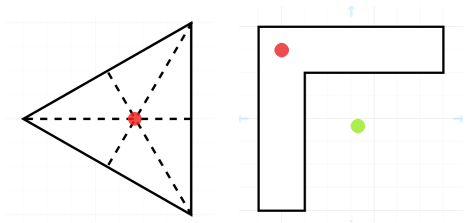
We label an image contour directly as a segment if it contains no other contour lines in its interior. If an image contour contains another contour line, we label the difference of its outer contour line and its inner contour (hole or interior boundary) as a segment and the inner contour as another segment.



(a) (b) (c) (d)

Figure 3.2: a) b) Segment 1 and segment 2 are not contained in any other segment. c) Segment 3 is contained in segment 2. d) Segment 4 is contained in segment 2 and segment 3.

It is useful to know the diameter or "thickness" of a segment to decide on the desired edge length for its corresponding mesh and whether we want to discard this segment as too small to be relevant. We take the radius of the largest inscribed circle as a measurement for the diameter of an arbitrary shape. For convex shapes we can simply take its centroid to locate it, but for non-convex shape the centroid may lay outside its boundary. In this case we want to take the visual center of the shape instead. Finding the visual center of a shape is a problem that was first solved for the purpose of finding the remotest places on earth (pole of inaccessibility).[Garcia-Castellanos and Lombardo, 2007] We use the implementation of V. Agafonkin's polylabel which uses an iterative grid algorithm to find the center.[Agafonkin and other contributors, 2016]



(a) (b)

Figure 3.3: a) Triangle with centroid in red. b) Non-convex shape with centroid in green and visual center in red.

To make the internal features of an object visible we can display the knotwork for each segment in a different color, width or decoration. We can also mask segments based on their topological hierarchy to visualize features more clearly. (see section 3.5)

3.3 Mesh Generation

3.3.1 Spring Force Meshing

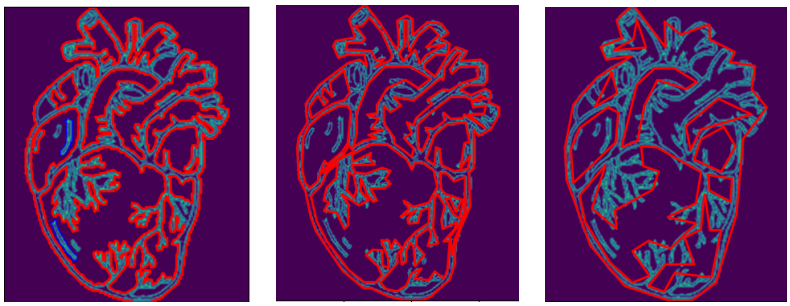
A classical approach of smoothing meshes for regularity is to apply a force on its nodes based on the distances to its adjacent nodes. When we compute this force based on its edge lengths and a desired length l_0, it is similar to finding an equilibrium of a set of springs which contract and extend.
In our implementation we first create a lattice of staggered points over the bounding box of a segment and then remove the points that fall outside the boundary of the segment to obtain our initial lattice. Then we iteratively compute the Delaunay triangulation of these points and move them according to a force that is based on the current edge lengths l_0 of the triangulation. We compute only repulsive forces like shown in [Persson and Strang, 2004]:

$$f((l, l_0) = \begin{cases} l_0 - l & l \leq l_0 \\ 0 & l \geq l_0 \end{cases} \tag{3.2}$$

Should a point fall outside the boundary of the segment, we move it to the closest point inside the segment. The algorithm stops iterating when the difference of the current to the last position of the points is less than a specified tolerance ϵ.

3.3.2 Wavefront Meshing

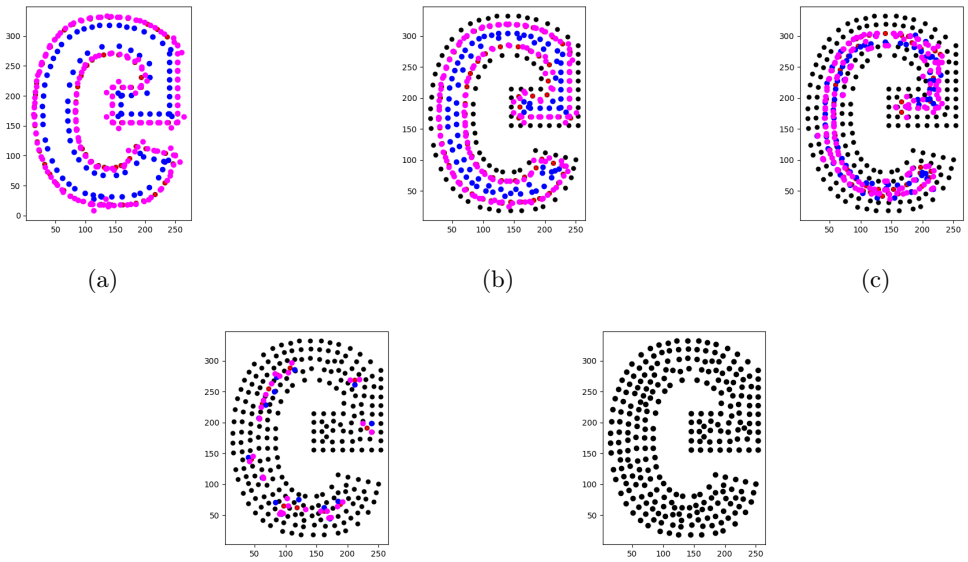
A method of creating a lattice of points which is setup to form quadrilaterals in regular distances in alignment to the shape geometry is to move in steps from the boundary of a segment towards its interior.[Browne, 2003] First we initialize a set of points in regular distances on the exterior and interior boundaries of the segment. These distances should align with our desired edge length l. We found that simply interpolating points on the contour line to obtain this set of points can sometimes erase details or change the topology of the shape. To avoid this we apply a topology preserving variation of the Visvalingam & Whyatt line simplification algorithm.[de Magalhaes et al., 2014]



(a) (b) (c)

Figure 3.4: Line simplification: a) $\epsilon = 16$ b) $\epsilon = 32$ c) $\epsilon = 320$

After the initialization we iteratively create three new points for each point that was added in the last iteration: one in the perpendicular and two in the tangent left-right direction to the initial boundary points. At each iteration we remove points which are located too close or outside the boundary shape and merge points in proximity less than $\sqrt{l}$ of each other by taking their averaged coordinates. This process converges when no new points can be added without merging. In the end we compute the Delaunay triangulation of the resulting lattice to obtain a triangle mesh.(See [Browne, 2003] for further details on the wavefront method)



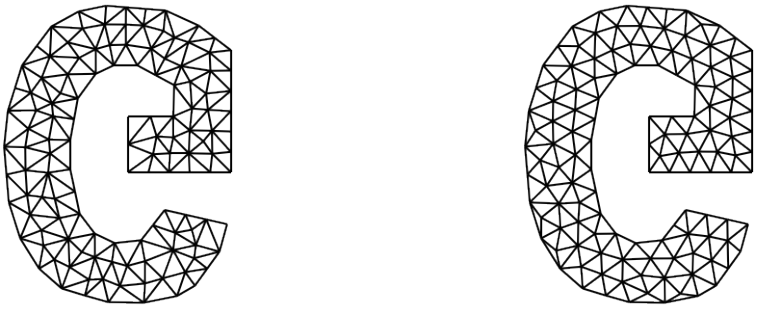
(a) (b) (c)

(d) (e)

Figure 3.5: a)-d) show 4 wavefront iterations. The pink dots are the left and right steps, the blue dots are the forward steps and the black dots the the points from the previous iteration. e) shows the finished lattice after merging. We notice that we might improve our implementation if "we would modify the regular spacing by local outline so that convex intervals are more densely packed and concave intervals more sparsly spaced".[Browne, 2003]

3.3.3 Centroidal Voronoi Tesselation

We find that the lattices generated by the force based method or the wavefront can be improved further by moving the points in the lattices into the maximum possible distance to each other. We can do this using CVT smoothing and even avoid altering the shape by fixing the nodes at the boundary. [Schloemer, 2018b]



(a) (b)

Figure 3.6: a) triangulation after wavefront b) triangulation after wavefront and CVT smoothing

3.3.4 Conversion of Triangles into Quadrilaterals

A mesh consisting of triangular elements can be converted into a quadrilateral mesh or mixed mesh of triangles and quadrilaterals (3,4 mesh). This can simply be achieved by joining pairs of neighboring triangles. If a triangle has multiple adjacent triangles then we have multiple choices of forming a quadrilateral with this triangle. If there are no adjacent triangles then the triangle can't be converted and it is left over. In our case we want to obtain a quadrilateral mesh with not only the minimum amount of left over triangles, but also the one which maximizes the degree of geometric regularity and alignment with its boundary. When we see each triangle in a mesh as a vertex in a graph with neighbours as adjacent triangles and if each edge has a score based on the resulting quadrilateral of the two triangles, then we have a maximum weight matching problem. The only difference is that in our case, vertices which are not matched have a score as well. This is because we do not want two triangles to join when the resulting quadrilateral is non-convex. We could anyways solve for the maximum matching without scoring triangles and then convert some quadrilaterals back into triangles. Edmond's algorithm solves maximum weighted matching problems in $O(V^2E)$ and there are also algorithms which run in approximated linear time.[Edmonds, 1965] [Duan and Pettie, 2014]

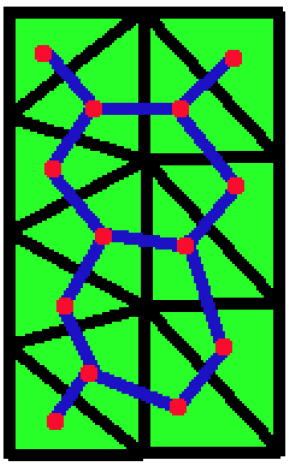


Figure 3.7: A triangle mesh shown as a graph. A red dot is a vertex representing a triangle. A blue line is an edge representing a quadrilateral made by joining two triangles.

We avoid having to reconvert the quadrilaterals into triangles and solve the problem directly using Integer Linear Programming:

Quadrilaterals:

$$x_{ij} \in \{0,1\} \text{ for each triangle } i \text{ and adjacent triangle } j \in Adj_i \tag{3.3}$$

Triangles:

$$x_i \in \{0,1\} \text{ for each triangle } i \tag{3.4}$$

Weights:

$$q_i \in [0,1] \text{ for each } x_i \text{ and } q_{ij} \in [0,1] \text{ for each } x_{ij} \tag{3.5}$$

Objective:

$$maximize \sum_{i=0}^{N} x_i q_i + \sum_{i=0}^{N} \sum_{j \in Adj_i}^{M} x_i q_i \tag{3.6}$$

s.t

$$x_i + \sum_{j \in Adj_i}^{M} x_{ij} = 1 \text{ for each triangle } i \tag{3.7}$$

Each polygon is weighted with the following heuristic:

$$q = \alpha \epsilon_{reg} + (1-\alpha)\epsilon_{dir} \text{ [Itoh and Shimada, 2002]} \tag{3.8}$$

$$\epsilon_{dir} = \frac{2\mu_\theta}{\pi} \tag{3.9}$$

$$\epsilon_{reg} = 1 - \frac{\sigma_\angle}{\pi} \tag{3.10}$$

ϵ_{dir} scores the directional alignment with the boundary of a polygon by taking the average of the angle $\theta_k \mod \frac{\pi}{2}$ for each edge and the vector of the two closest boundary points for each vertex belonging to that edge. We take $\mod \frac{\pi}{2}$ of θ_k so that the angle a tanget and a perpendicular angle is equally "well aligned" with the boundary.

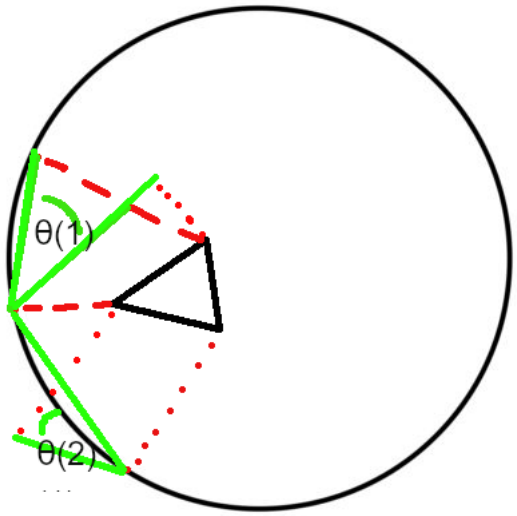


Figure 3.8: Shows two of three angles for calculating ϵ_{dir} of this triangle given the displayed shape.

ϵ_{reg} scores the geometric regularity of a polygon by taking the standard deviation of its internal angles. α is a value between 0 and 1 that serves as a trade-off between these two heuristics.[Itoh and Shimada, 2002] We always prefer convex quadrilaterals over triangles. Because of this we multiply triangle scores by $\frac{4}{10}$ so that one parallelogram would be preferred over two perfectly regular triangles. To avoid non-convex quadrilaterals we score them with 0.

3.4 Weaving and Display of the Knotwork

Based on the generated mesh we compute the interlacement using the Mercat algorithm. [Mercat, 1997] We initialize four nodes at the midpoint of each edge with each having a direction (either 45°, 135°,225°or 315°). Starting at a random node the algorithm creates curves from the current node to the next node until it ends up back at the beginning. The next node is selected by finding the closest adjacent edge based on the junction and angle the current node is facing. If it ends up back at the beginning, we close this thread and start a new one at one of the nodes that was left over. When we pass an edge the second time, we alternatingly store if our thread is laced over or under the edge to visualize the interlacement.

Each time we jump from the current node a to the next node b we store a cubic Bezier curve, for which we compute the control points in the follow manner[Kaplan and Cohen, 2003]:

$$cntrl1 \leftarrow \begin{bmatrix} x_a \\ y_a \end{bmatrix} + \begin{bmatrix} cos(angle(a)) \\ sin(angle(a)) \end{bmatrix} l\tau;\ cntrl2 \leftarrow \begin{bmatrix} x_b \\ y_b \end{bmatrix} + \begin{bmatrix} cos(angle(b)) \\ sin(angle(b)) \end{bmatrix} l\tau \tag{3.11}$$

$$\tau = \begin{matrix} angle\frac{\beta}{pi} & \text{angles between edges} \leq 90° \\ \frac{\beta}{pi} & \text{otherwise} \end{matrix} \tag{3.12}$$

β is a parameter that controls how "tight" the thread should be weaved.(See Fig. X in Appendix)

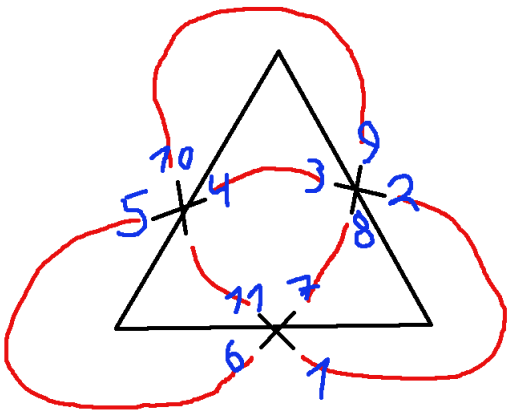


Figure 3.9: This sketch depicts the order the Mercat algorithm constructs curves.

3.5 Procedure

input : RGB *image*, EdgeLength l_0,
output : Bezier *curves*

segments ← segmentImage(image);
skip ← ∅;
curves ← ∅;
for $i \leftarrow 0$ **to** $length(segments)$ **do**
  $l \leftarrow min(l, segment(i).diameter)$;
  $enclosingSegment \leftarrow \emptyset$;
  for $j \leftarrow 0$ **to** i **do**
    if $segments(j).contains(segment(i)$ **then**
      $enclosingSegment \leftarrow segments(j)$;
    end
  end
  if *(enclosingSegment* ***not*** *in skip)* ***and*** *(enclosingSegment* $\emptyset$*)* **then**
    skip ← skip ∪$segment(i)$;
  end
  nodes, cells ← $generateTriangleMesh(segment(i), l_0)$;
  nodes, cells ← $cvtSmoothing(nodes, cells)$;
  nodes, cells ← $convertTriangles2Quads(nodes, cells)$;
  curves ← $mercat(nodes, cells)$;
end

Algorithm 1: Image to Knotwork (masking segments to visualize internal features and adapting edge length l_0 based on segment diameter)

Chapter 4

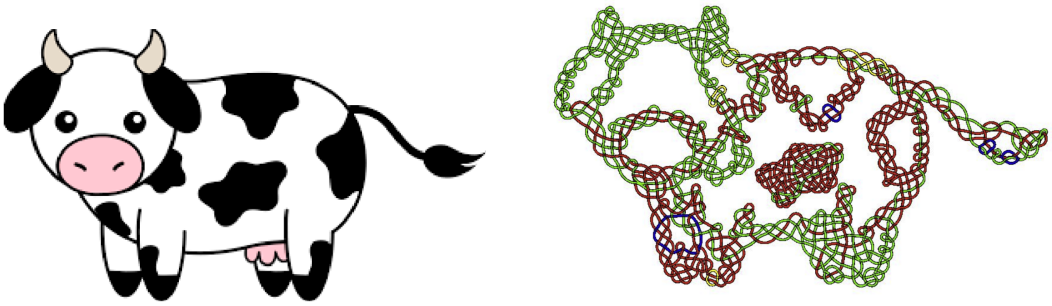
Results

4.1 Setup

We implement the method in Python 3.6 using Numpy, Scipy, OpenCV and *Shapely* for fast geometric computing and image processing. [Gillies et al., 07] [Itseez, 2015] Our implementation of the spring force meshing is built on top of the work of *dmsh* and *meshplex*.[Schloemer, 2018a] For the CVT smoothing we use *optimesh* and for the ILP we use CBC (**C**oin-or **b**ranch and **c**ut).[Schloemer, 2018b] [Forrest et al., 2018] The visualization is done using Qt5. Each thread in a knotwork is displayed with a distinct random color.

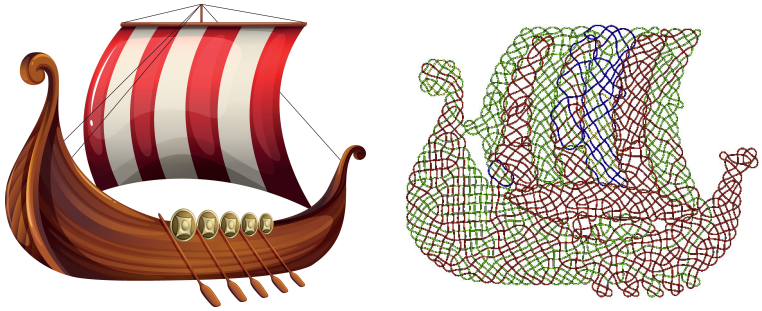
4.2 Knotworks from Images

We present outputs of the method based on three "difficulty levels" of images. The first image is a cartoon without shading or details. The second image is a cartoon with more details, shading and subtle color transitions. The last image is a photograph of a human face.



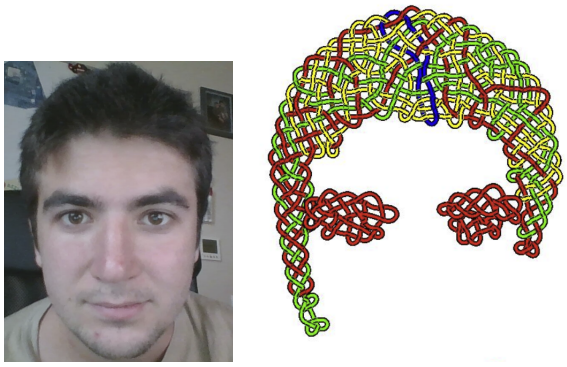
(a) (b)

Figure 4.1: a) Input image b) Generated knotwork using wavefront meshing and CVT with α=0.7



(a) (b)

Figure 4.2: a) Input Image b) Generated knotwork using adaptive edge size, force based meshing and CVT with $\alpha = 0.9$. It takes around 30 seconds to generate this knotwork.

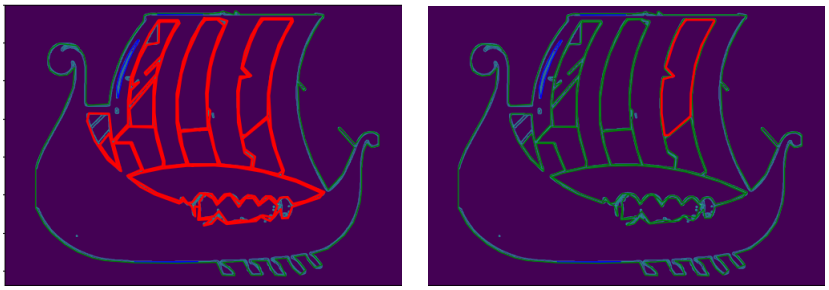


(a) (b)

Figure 4.3: a) Input Image b) Generated knotwork using wavefront meshing, CVT and greedy triangle conversion.

4.2.1 Discussion

The first knotwork captures the shape and internal features of the input image. The only detail that is missing are the eyes. This is due to their segments having a too small diameter. The tolerance can simply be lowered to incorporate them. The second image does not mask the stripes in the sail as expected. This happens because there is an enclosing segment surrounding the stripes due to a thick contour line. (see Fig. 4.4) For the last image we fail to segment the eyes, nose and mouth. K means clustering either makes too many unnecessary color segments or erases important features.(see Fig 4.5) Without clustering we fail to avoid detected contours of regions that depict regions of similar shading instead of internal feature. Another issue that can be seen especially on the second and third image is that sometimes knotworks created of one segment overlap the knotwork of another segment (see eyes of figure 4.3). This could be checked and avoided by either shrinking this segment or by altering β for the overlapping parts of the thread. More generated knotworks can be found in the Appendix.



(a) (b)

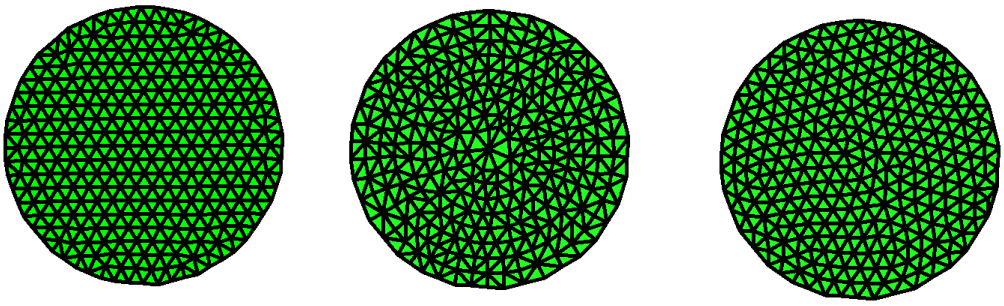
Figure 4.4: a) Failed segmentation due to too thick edges. b) Region falsely enclosed by a)

(a) (b)

Figure 4.5: a) Color segmentation with too many unwanted regions due to shading. b) Larger region size erases internal features.

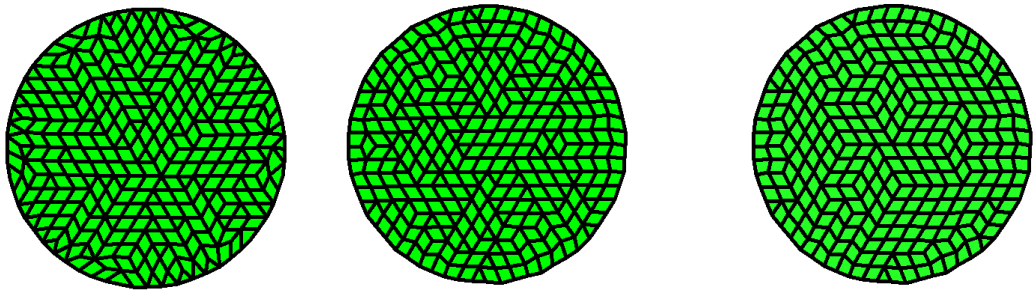
4.3 Regularity and Symmetry

We are mainly interested in the regularity and symmetry of the 3,4 mesh after conversion of the triangles into quadrilaterals. We compare the optimal solution found with ILP to a greedy solution and a solution found by the method of C. Browne(note that its uses a different regularity heuristic).[Browne, 2003] We choose a simple shape (which is not a perfect circle due to small irregularities in the contour line) for the following experiments to ease a visual judgement. For around 150 nodes the ILP finds the optimal solution in 0.01 seconds (CPU time).



(a) (b) (c)

Figure 4.6: Triangulation after: a) force based meshing b) wavefront c) wavefront + CVT smoothing



(a) (b) (c)

Figure 4.7: Conversion after force based meshing using a) greedy search $\alpha = 1$ b) Brown's method th=0.7 c) optimal solution $\alpha = 1$

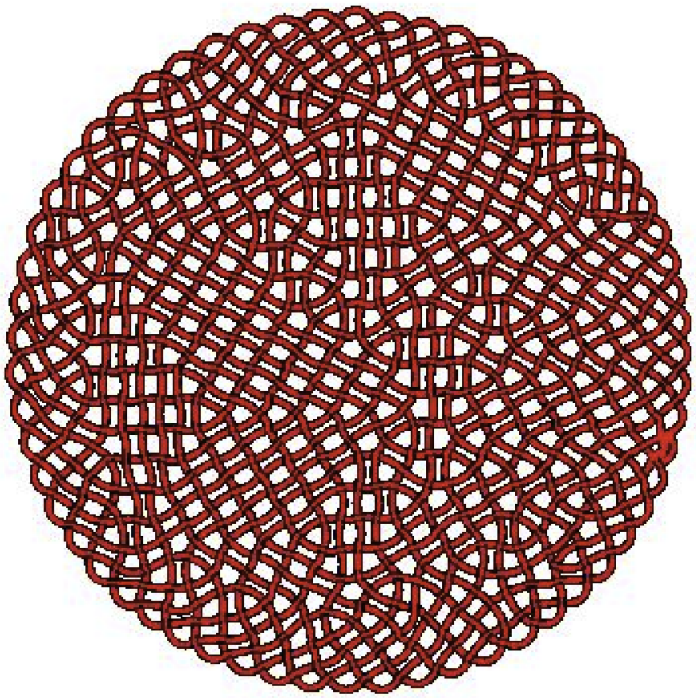


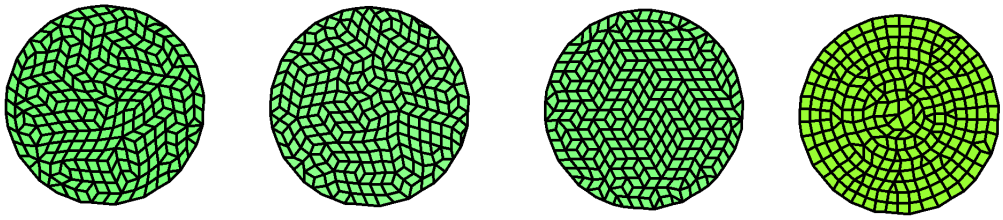
Figure 4.8: Knotwork of optimal solution $\alpha = 1$ after force based meshing

4.3.1 Discussion

We can see that force based meshing generates the most regular and symmetric meshes. It is also clearly visible that the optimal solution found with ILP results in a mesh with the least amount of left over triangles. The mesh elements are on average the most similar to other elements in their local neighbourhood. This lets the mesh and resulting knotwork appear the most regular and symmetric compared to the other solutions. It is to mention that in our experiments we rarely generated global symmetry (based on the axis of symmetry for the given shape).

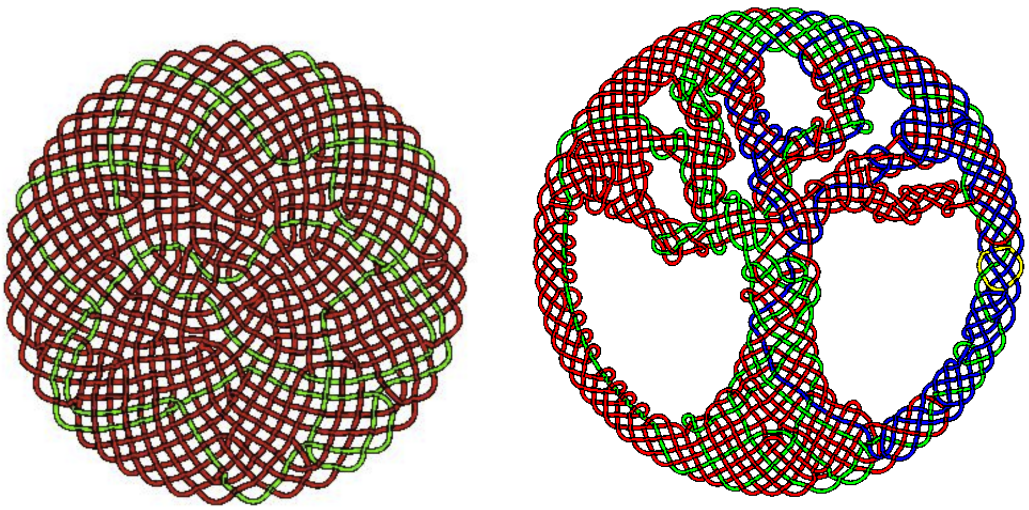
4.4 Shape Aligment

We saw already in Fig. 4.6 that the wavefront method clearly produces cells which are the most aligned to the boundary of its shape. We also want to see whether ϵ_{dir} in our heuristic (Eq 3.9) leads to quadrilateral conversions which are more aligned to the boundary as well.



(a) (b) (c) (d)

Figure 4.9: optimal solution using: a) wavefront and cvt smoothing with $\alpha = 0$ b) wavefront and cvt smoothing with $\alpha = 1$ c) force based meshing with $\alpha = 0$ d) wavefront with $\alpha = 1$



(a) (b)

Figure 4.10: Knotwork of optimal solution using wavefront with $alpha = 1$ on a) circle b) tree of life.

4.4.1 Discussion

We find in our results that that ϵ_{dir} introduces some degree of curvyness, which is not always aligned with the shape (see Fig 4.9 a) b)). On Fig 4.9 c) it does not even seem to have this effect. On further experiments it becomes clear that the initial lattice has a bigger effect on shape alignment than the conversion. Solving the mesh generated by the wavefront method for regularity generates knotworks that are aligned and dynamically incorporate the shape geometry (see Fig 4.10).

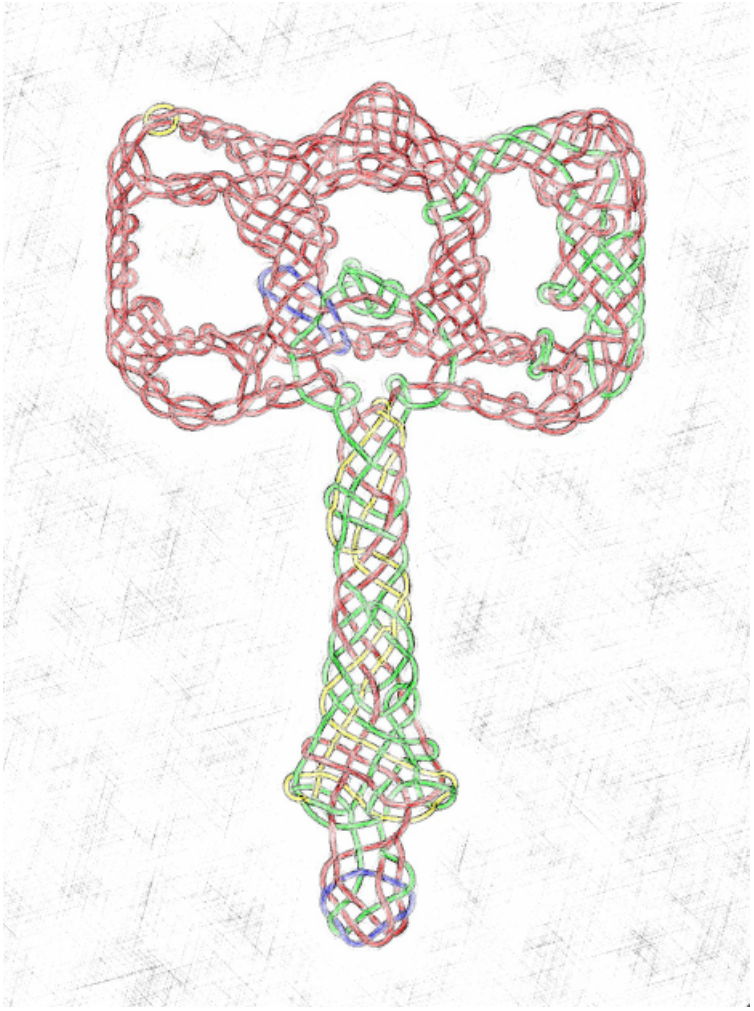
Chapter 5

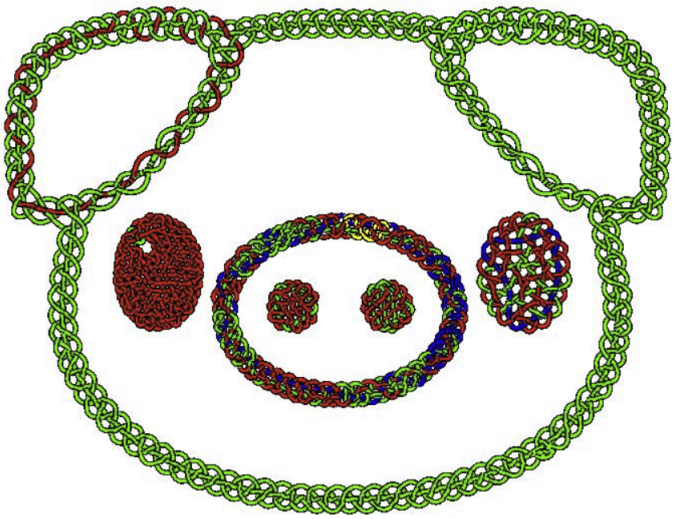
Conclusion and Further Research

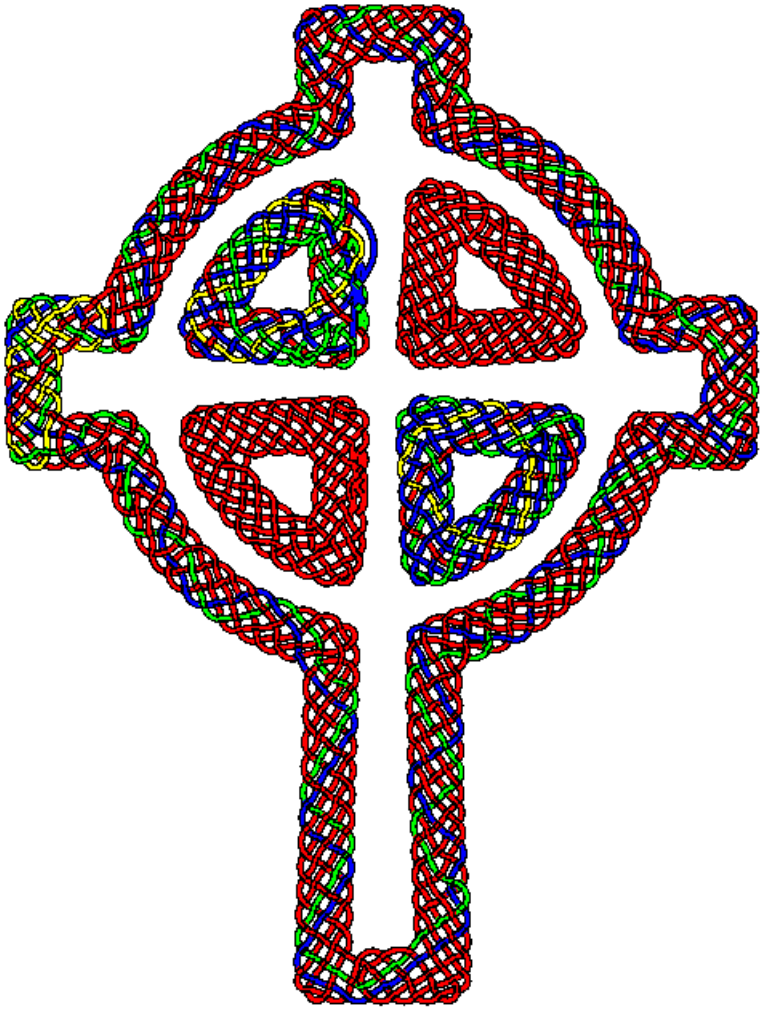
We saw that an optimal mesh conversion is crucial for a regular and symmetric 3,4 mesh, but the method to generate the initial lattice has a bigger effect on the alignment of the mesh with its enclosing region. It is advisable to apply the wavefront method when this outcome is preferred. We find that smoothing the lattice generated by the wavefront method leads to more regularity, but the resulting lattice is slightly less aligned with the shape boundary and is less setup to form quadrilaterals. It would be beneficial to have a smoothing method without these losses. We could imagine that applying force based meshing on the initial wavefront lattice could lead to this, if instead of triangulating the lattice at each iteration, one would form a 3,4 mesh each time. This way one would directly optimize a 3,4 mesh that was initialized in alignment with the shape boundary. Overall we find that the presented method provides a good basis for generating Celtic knotworks automatically from images. Combining it with the decoration techniques [Kaplan and Cohen, 2003] and with coloring schemes [Larboulette, 2007], we think that it would result in stunning artworks. The author plans to develop this further as a tool for digital creations. But first, the object segmentation has to become more robust to shading, fine details and subtle color transitions in the image. It would also be desirable to adapt the shape and thickness of the knotwork to its local shape geometry. Furthermore we would like to incorporate the axis of symmetry for each segments in the mesh creation for a more artistic design.

Chapter 6

Appendix







List of Figures

Bibliography

[Agafonkin and other contributors, 2016] Agafonkin, V. and other contributors (2016). polylabel.

[Aurenhammer, 1991] Aurenhammer, F. (1991). Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405.

[Bain, 1973] Bain, G. (1973). *Celtic art: the methods of construction*. Courier Corporation.

[Browne, 2003] Browne, C. B. (2003). Font decoration by automatic mesh fitting. US Patent 6,542,157.

[de Magalhaes et al., 2014] de Magalhaes, S. V., Andrade, M. V., Franklin, W. R., and Li, W. (2014). An efficient map generalization heuristic based on the visvalingam-whyatt algorithm.

[Denman et al., 2019] Denman, Q., Hanrieder, M., Higler, S., Hufkens, L., and Schneider, S. (2019). Context free celtic knotwork.

[Duan and Pettie, 2014] Duan, R. and Pettie, S. (2014). Linear-time approximation for maximum weight matching. *Journal of the ACM (JACM)*, 61(1):1–23.

[Edmonds, 1965] Edmonds, J. (1965). Paths, trees and flowers. *CANADIAN JOURNAL OF MATHEMATICS*, pages 449–467.

[Forrest et al., 2018] Forrest, J., Ralphs, T., Vigerske, S., LouHafer, Kristjansson, B., jpfasano, EdwinStraver, Lubin, M., Santos, H. G., rlougee, and Saltzman, M. (2018). coin-or/cbc: Version 2.9.9.

[Garcia-Castellanos and Lombardo, 2007] Garcia-Castellanos, D. and Lombardo, U. (2007). Poles of inaccessibility: A calculation algorithm for the remotest places on earth. *Scottish Geographical Journal*, 123(3):227–233.

[Gillies et al., 07] Gillies, S. et al. (2007–). Shapely: manipulation and analysis of geometric objects.

[Glassner, 1999] Glassner, A. (1999). Celtic knotwork, part i. *IEEE Computer Graphics and Applications*, (5):78–84.

[Hale and Emanuel, 2002] Hale, D. and Emanuel, J. (2002). Atomic meshing of seismic images. In *SEG Technical Program Expanded Abstracts 2002*, pages 2126–2129. Society of Exploration Geophysicists.

[Ilea and Whelan, 2006] Ilea, D. E. and Whelan, P. F. (2006). Color image segmentation using a spatial k-means clustering algorithm.

[Itoh and Shimada, 2002] Itoh, T. and Shimada, K. (2002). Automatic conversion of triangular meshes into quadrilateral meshes with directionality. *Int. J. CAD/CAM*, 1(1):11–21.

[Itseez, 2015] Itseez (2015). Open source computer vision library. `https://github.com/itseez/opencv`.

[Kaplan and Cohen, 2003] Kaplan, M. and Cohen, E. (2003). Computer generated celtic design. *Rendering Techniques*, 3:9–19.

[Kaplan et al., 2004] Kaplan, M., Praun, E., and Cohen, E. (2004). Pattern oriented remeshing for celtic decoration. In *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.*, pages 199–206. IEEE.

[Larboulette, 2007] Larboulette, C. (2007). Celtic knots colorization based on color harmony principles. In *Proceedings of the Third Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging*, pages 65–72.

[Liu et al., 2009] Liu, Y., Wang, W., Lévy, B., Sun, F., Yan, D.-M., Lu, L., and Yang, C. (2009). On centroidal voronoi tessellation—energy smoothness and fast computation. *ACM Transactions on Graphics (ToG)*, 28(4):1–17.

[Mercat, 1997] Mercat, C. (1997). Les entrelacs des enluminures celtes.

[Mueller, 2013] Mueller, J. (2013). Celtic knot drawing tool.

[Persson and Strang, 2004] Persson, P.-O. and Strang, G. (2004). A simple mesh generator in matlab. *SIAM review*, 46(2):329–345.

[Schloemer, 2018a] Schloemer, N. (2018a). dmsh.

[Schloemer, 2018b] Schloemer, N. (2018b). optimesh.

[Soleng and Holden, 1998] Soleng, H. H. and Holden, L. (1998). Gridding for petroleum reservoir simulation. *Numerical Grid Generation in Computational Field Simulations, edited by Cross, M., Soni, BK, Thompson, JF, Hauser, J. and Eiseman, PR, Missisipi State University.*

[Suzuki et al., 1985] Suzuki, S. et al. (1985). Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46.

[Wang and Du, 2005] Wang, D. and Du, Q. (2005). Mesh optimization based on the centroidal voronoi tessellation. *International Journal of Numerical Analysis and Modeling*, 2:100–113.

[Whitehead, 2010] Whitehead, J. (2010). Toward proccedural decorative ornamentation in games. In *Proceedings of the 2010 workshop on procedural content generation in games*, pages 1–4.

[Zienkiewicz et al., 2005] Zienkiewicz, O. C., Taylor, R. L., and Zhu, J. Z. (2005). *The finite element method: its basis and fundamentals.* Elsevier.