Covering Polyhedra by Motifs with Triangular Fundamental Regions

A Thesis

submitted to the faculty of the Graduate School

of University of Minnesota

by

Lakshmi Ramya Pathi

In partial fulfillment of the requirements

for the degree of

Master of Science

Dr. Douglas Dunham

September 2011

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my heartfelt gratitude to all the people who have motivated me towards the completion of the thesis.

Firstly, I sincerely thank Dr. Douglas Dunham, my faculty advisor, for his guidance throughout the course of my thesis. I enjoyed working with him and I am really grateful for his valuable feedback on my work.

I would also like to thank Dr. Pete Willemsen and Dr. Steven Trogdon for being a part of my thesis committee and taking time to read my thesis in detail.

I would like to thank all the professors of the CS department at UMD who have helped me learn various concepts during my Masters. I would also thank Lori Lucia and Clare Ford for their timely help regarding the office formalities. I am thankful to my fellow graduate students for their help during my stay in Duluth.

I am thankful to my parents Bhaskara Rao and Uma and my sister Haritha for their endless patience, support and encouragement when it was most needed.

Abstract

In geometry, a “net” of a polyhedron is a two-dimensional figure where all the polygons are joined by edges, which when folded becomes a three-dimensional polyhedron. A “subnet” is a subset of a net which is formed by the faces of the polyhedron. Technically, multiple nets can exist for a polyhedron and different polyhedrons can be obtained from a single net.

The algorithm designed takes any arbitrary subnet of a polyhedron as an input and maps a triangular motif onto each of the polygon faces of the subnet. Each polygon face is assumed to be convex and will be triangulated from its centroid. The triangles of that triangulation will then be filled in with transformed versions of the motif. Currently, Dr Dunham's work creates a pattern on a specific polyhedron while my research aims at mapping a single pattern onto each of the possibly different polygons of a net that can be used to construct any patterned polyhedron.

Table of Contents

	Page
List of Figures	v
1. Introduction	1
2. Motivation	3
3. Basic Terminology	5
3.1 Polygon	5
3.2 Polyhedron	6
3.3 Polyhedron Nets	7
3.4 Polyhedron Subnets	10
3.5 Triangular fundamental regions	11
4. Overview	17
4.1 Subnet Generation	17
4.2 Centroid Generation	18
4.3 Applying Transformations	19
4.3.1 Scaling	20
4.3.2 Rotation	20
4.3.3 Translation	21
4.4 Clipping	22
4.4.1 Circles	22
4.4.2 Polylines & Polygons	22

5. Algorithms 26

5.1 Centroid Generation Algorithm 26

5.2 Algorithm for applying transformations 27

5.3 Clipping Input pattern data 28

6. Results 30

7. Conclusion and Future work 33

7.1 Future work 33

8. References 34

9. Appendix 35

9.1 Output data for Subnet Generator 35

9.2 Output data for Centroid Generator 35

9.3 Sample data file used for clipping 36

List of Figures

Figure	Page
1. Polyhedra in crystal forms	3
2. Net of a cube	7
3. Nets of pyramid and tetrahedron	7
4. Possible nets of a cube	8
5. Possible nets of an icosahedron	8
6. Not a net of a cube	8
7. Labeling edges to avoid ambiguity	9
8. No net polyhedra	9
9. Subnets of a pyramid	10
10. Subnets of a cube	11
11. Unclipped Pattern (Stage1)	12
12. Stage 2 in Clipping	13
13. Stage 3 in Clipping	14
14. Stage 4 in Clipping	15
15. Stage 5 in Clipping (Final Pattern)	16
16. Triangle formed by the centroid	19
17. Transformed Figure	19
18. Resultant scaling matrix	20
19. Resultant rotation matrix	21
20. Resultant translation matrix	21

21. Polygon Clipping 22

22. Four cases of polygon clipping 23

23. Similar triangles 24

24. Patterned subnet of a icosahedron …..... 30

25. Patterned subnet of a cube …..... 31

26. Patterned subnet of a tetrahedron …..... 32

1. INTRODUCTION

The 'net' of a polyhedron consists of a single connected polygon, formed from the faces of polyhedron, arranged in a plane without any overlaps. Some of the polygons of the net share edges. In theory, the net of the polyhedron can be cut out and folded along the shared edges to obtain a polyhedron. This implies that the union of the faces of a net of a convex polyhedron is a simple polyhedron in a plane, subdivided into convex polygons.

In fact, it is difficult to find a net for a particular polyhedron. This is because in geometry, some polyhedra do not have nets. Various examples of no-net polyhedra are presented in [5]. Some of the polygons like the cube and tetrahedron have multiple nets. However, not every figure that looks like a net of a polyhedron is a net for that particular polyhedron.

A subnet is a subset of a net of a polyhedron. The combination of several subnets of a polyhedron can form a three-dimensional polyhedron. Kailash Aurangabadkar has designed an algorithm which can find the subnets of any arbitrary convex or non-convex polyhedron[4].

My research work is focused on covering polyhedra by motifs with triangular fundamental regions. The motifs are applied to any arbitrary polyhedron, irrespective of its shape and its associated polygons. There is a limitation in Sameer Atar's algorithm; it generates patterns only on the triangular faces of polygons [9]. Ankur Nepalia's algorithm has overcome this limitation [3].

Ankur's algorithm maps patterns onto polygons, but each polygon pattern must be constructed “by hand”. The proposed research will automate the process and map a single pattern onto each face of the polygon. My algorithm assumes each polygon face is convex. Each face will be triangulated from a central point by finding the centroid of each face of polyhedron. The triangles of that triangulation will then be filled in with transformed versions of the motif. The present algorithm maps triangulated patterns onto any subnets of a polyhedron, irrespective of the shape of its constituent polygons.

The remainder of this thesis is organized into eight sections. Section 2 explains the motivation behind the research work. Section 3 reviews the basic terminology associated with this work. Section 4 describes the overview of the algorithms and procedures used for accomplishing this task. Section 5 discusses the algorithm to cover the faces of the polyhedron with triangulated motifs. Section 6 reviews the results. Section 7 presents the conclusion and future work of this research work. The list of references are provided in Section 8. Finally, Section 9 presents the appendix which includes the data formats used as inputs for the programs.

2. MOTIVATION

Polyhedra have been closely related to the world of art since the Pre-Renaissance period. According to Cromwell [1], polyhedra have been used in various areas such as architecture, art, ornamental designs, etc. Basic examples of ancient polyhedra include the Archimedean polyhedra and the Egyptian pyramids. The regular polyhedra, the Platonic solids, include the cube, tetrahedron, octahedron, dodecahedron, and icosahedron. Crystallography has very close relation to such geometries. Figure 1 shows several polyhedra which can occur in crystal forms [6].

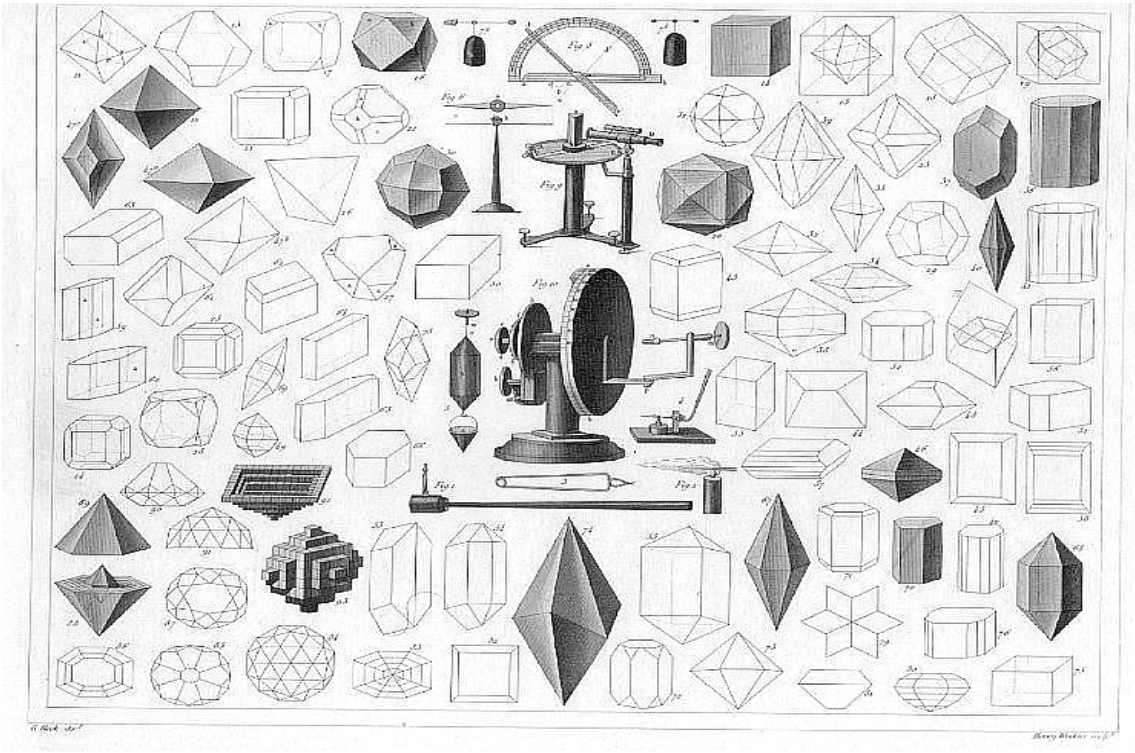

Figure 1

Also, people have created nets of polyhedron out of paper. Such paper models have been created for Platonic solids and many other polyhedra which include the hexagonal prism and the cuboid. Nets are very helpful to

study the properties of a polyhedron. Albrecht Durer introduced the notion of creating “paper” nets of polyhedra. Dr. Dunham has been working on visualizing the polyhedra and generating repeating patterns on nets of polyhedra . However, finding a connected net for a polyhedron is a difficult task. Kailash Aurangabadkar’s research work was focused towards generating a subnet for an arbitrary polyhedron[4]. His algorithm takes an arbitrary polyhedron as input and generates subnets for this polyhedron. Upon folding up and gluing these subnets, a complete polyhedron can be formed. Ankur Nepalia's research work concentrated in mapping a single pattern onto each of the possibly different polygons of a net that can be used to construct a patterned polyhedron.

The current program can map a motif within a triangular fundamental region onto each of the polygon faces of a polyhedron. This program improves the program developed by Kailash that draws a single motif within triangular fundamental region of the subnet. The current program can generate a pattern on any arbitrary convex polygon. The pattern generation algorithm developed by Sameer Atar[9] works only for triangular faces, and non-triangular faces had to be treated differently. To overcome such limitations, the present program generates a centroid for any convex polygon in a given subnet. Then based upon the centroid as well as the subnet, a postscript file is generated for the subnet decorated with the pattern.

3. Basic Terminology

Now we will take a look at the basic terminology used in this thesis. Some of these terms are used in our everyday life.

3.1 Polygon

The word 'polygon' is derived from Greek words- 'poly' meaning 'many' and 'gon' meaning 'angle'. A polygon is a closed plane figure for which all the sides of the polygon are line segments. All the sides of a polygon are connected end to end to form a closed figure. The number of sides of a polygon signifies the name of a polygon. If a polygon has more than 10 sides, the people generally name them as 'n-gon'. In general, there must be at least three sides to form a closed figure.

There are several types of polygons: regular, irregular, convex, concave and crossed. A simple polygon is one whose edges do not cross. Such a polygon can either be convex or concave. In a convex polygon, all the interior angles of the polygon are less than 180°. Also, all the vertices of the polygon point outwards from the interior. In case of concave polygons, one or more interior angles are more than 180°. Some of the vertices point towards the interior of the polygon. All regular polygons are convex. Some of the regular polygons includes equilateral triangles, squares, regular pentagons etc. Non-convex polygons include the chevron and the pentagram (which is also not simple).

3.2 Polyhedron

The word 'polyhedron' is derived from Greek words- 'poly' meaning 'many' and 'edron' meaning face. The plural of polyhedron is 'polyhedrons' or 'polyhedra'. In Geometry, a polyhedron is a three-dimensional closed shape which is comprised of polygons for each of its faces. Hence a polyhedron has no curved surfaces. In a polyhedron, a vertex is a point at which three or more edges of a polyhedron intersect. And an edge is a line along which two faces of the polyhedron intersect.

Similar to polygons, polyhedrons are of two types: Convex and Concave. A polyhedron is said to be convex if a line segment joining two vertices of a polyhedron lies completely inside the polyhedron. A concave polyhedron is a non-convex polyhedron. In general, for a regular polyhedron, all the faces are comprised of regular polygons. The regular polyhedrons include the cube, tetrahedron, icosahedron, octahedron and dodecahedron.

There is an interesting fact about polyhedra which is called 'Euler-Poincare formula'. This formula states that 'the number of faces plus number of vertices minus number of edges is always equal to 2-2g [11].

That is,

For an oriented surface,

$F+V-E = 2-2g$ ----------(1)

where g is the genus,

F is the number of faces,

V is the number of vertices,

E is the number of edges.

Here, the genus g is the number of holes in a polyhedron.

3.3 Polyhedron Nets

A net of a polyhedron is a two-dimensional figure which is comprised of a group of connected polygons. These polygons can be cut out and folded to form the original three-dimensional polyhedron. Figures 2 and 3 show examples of nets of a cube, pyramid and tetrahedron respectively.

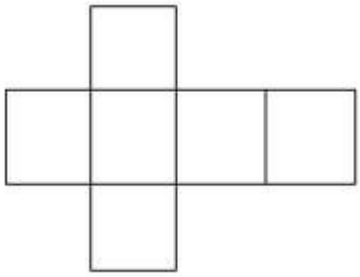

Figure 2: Net of a cube

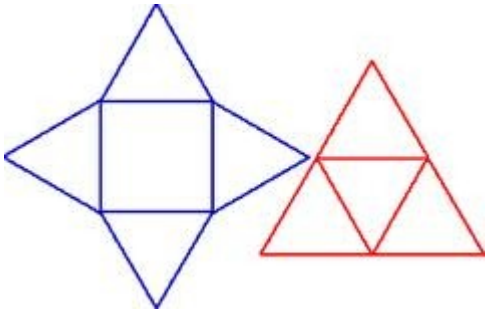

Figure 3: Nets of pyramid and tetrahedron.

Each regular polyhedron has a net. But each polyhedron may also have more than one net. For example, a cube has 11 distinct nets, an octahedron has 11 nets, while a dodecahedron and an icosahedron have 43380 possible nets and a tetrahedron has only 2 possible nets[10]. Figures 4 and 5 shows the 11 possible nets of a cube and octahedron respectively.

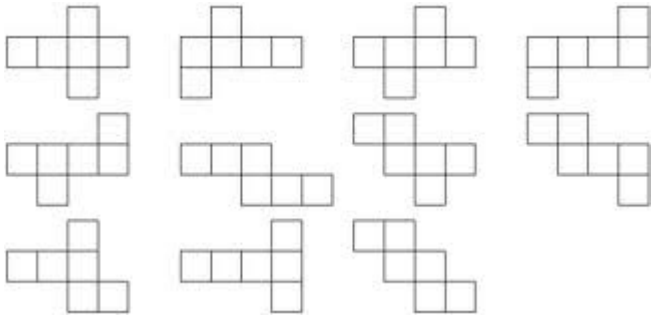

Figure 4: Possible nets of a cube.

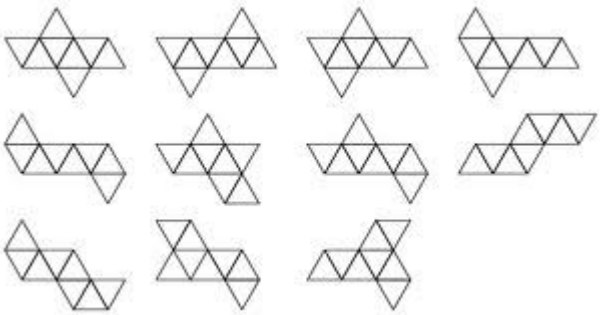

Figure 5: Possible nets of an icosahedron.

Also, one has to remember that all the two-dimensional planar figures composed of polygons do not form a net of a polyhedron. For example, in Figure 6, the figure is similar to a net of a cube. But it does not form the net of a cube.

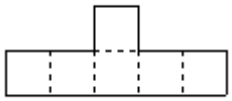

Figure 6: Not a net of a cube

It is also possible to obtain different shapes of polyhedrons from a single net. Some mathematicians label which edges are to be folded (Figure 7) to avoid ambiguous results in the shapes of polyhedra. In some situations, the folding of the net is done in a single way to avoid labeling of edges.

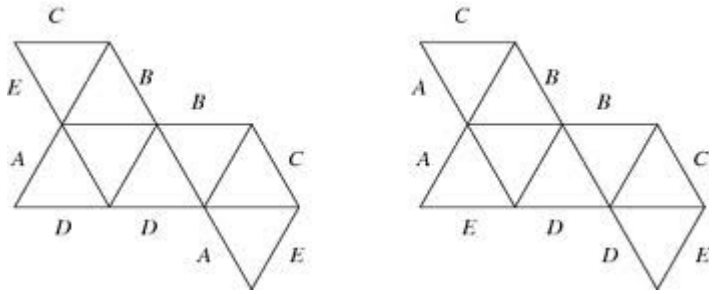

Figure 7: Labeling edges to avoid ambiguity

More interestingly, there exists some polyhedrons which do not have nets. One reason for such no net polyhedra is that these polyhedra have vertices whose sum of angles is greater than 360°. Figure 8 shows an example of a no net polyhedron[7].

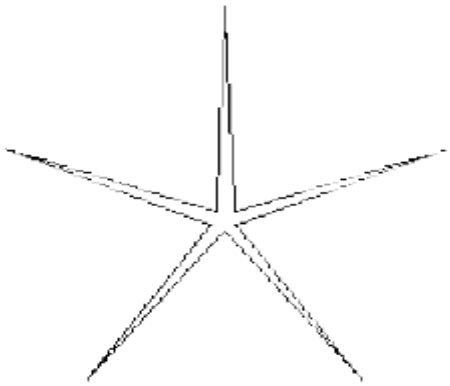

Figure 8: The prism over this star has no net.

The problem of finding a net for a polyhedron is much more complicated in case of non-convex polyhedrons when compared to convex polyhedrons. An algorithm was presented to overcome such problems[4].

3.4 Polyhedron Subnets

A subnet of a polyhedron is a connected subset of the polygons of a net. In other words, a polyhedron cannot be formed by just folding the faces of the polygons present in a proper subnet. A net is a combination of subnets of a polyhedron which in turn, can be joined together to form a complete polyhedron.

A complete set of subnets is a group of subnets that can be folded up and joined together to form a complete polyhedron. A complete set of subnets does not contain overlapping polygons.

Figures 9 and 10 shows an example of a subnets of a pyramid and a cube respectively.

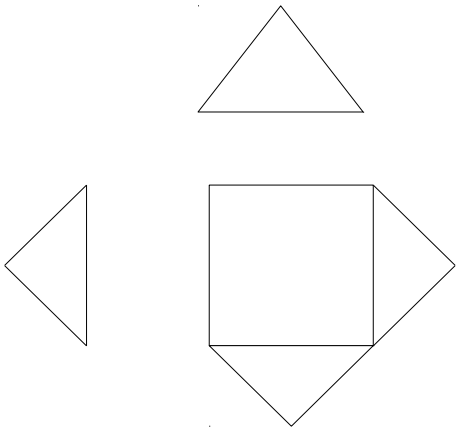

Fig 9: Subnets of a pyramid

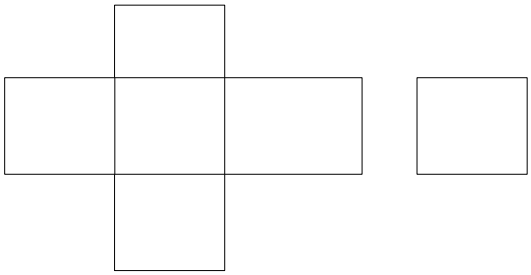

Fig 10: Subnets of a cube

3.5 Triangular Fundamental Region

A fundamental region is a region containing the motif that will cover the polyhedron. In our case, this region may consist of an entire butterfly pattern or a combination of the broken pieces of different butterflies which make up the motif. In case of broken pieces of the pattern, if we combine all those pieces and map them onto the triangular region, we obtain what we call the triangular fundamental region containing the motif.

The list of figures (Figures 11-15) gives a better idea regarding the triangular fundamental region.

Figure 11 represents the unclipped pattern. This pattern needs to be clipped to obtain the final pattern.

In Figure 12, one of the side for the pattern is clipped.

In Figure 13, the second side of polygon is also clipped. In this figure, if observed, one of the green dots on the yellow butterfly is present while the

other dot has been clipped off. It is because the center of the circle lies outside the clipped edge of the pattern.

The pattern in Figure 14 is the result of further clipping on Figure 13. The pattern hence obtained, is scaled so that it can be fit to the triangular region.

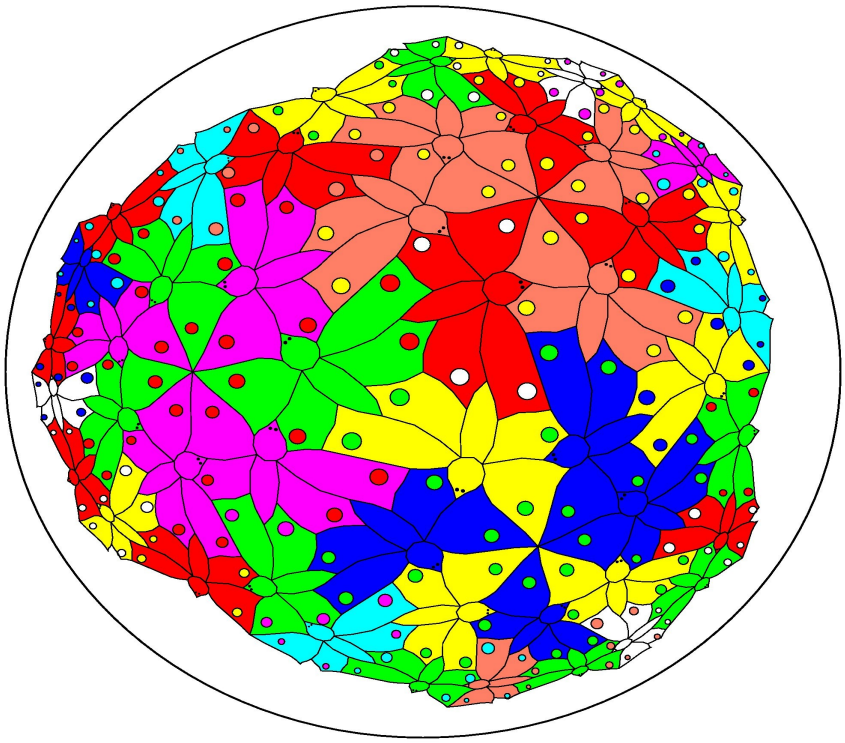

Figure 11: Unclipped Pattern (Stage 1)

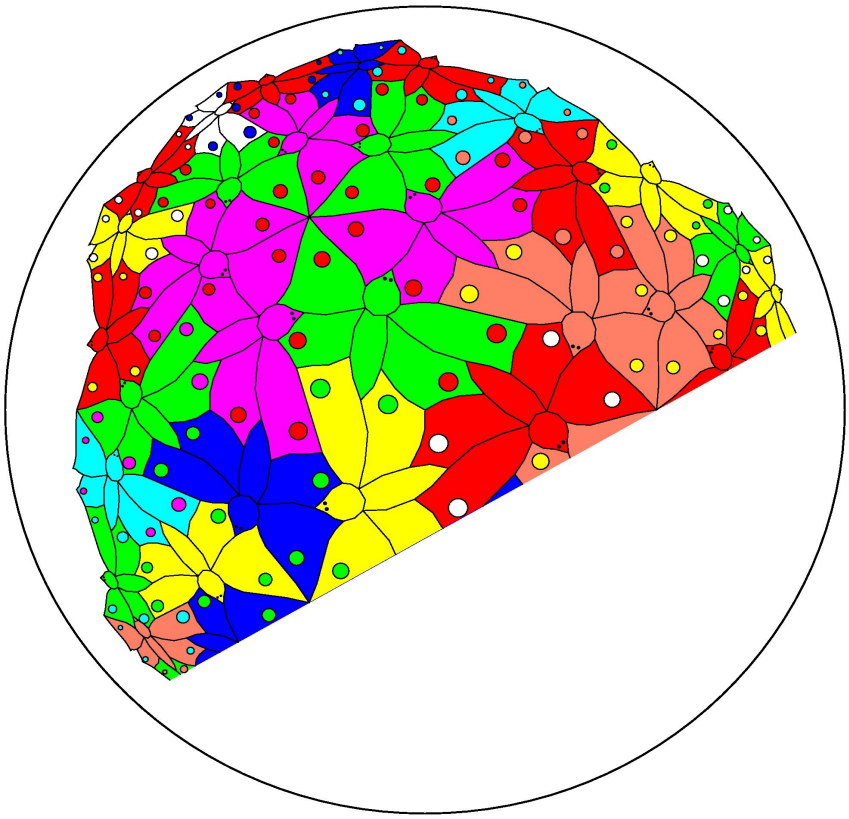

Figure 12: Stage 2

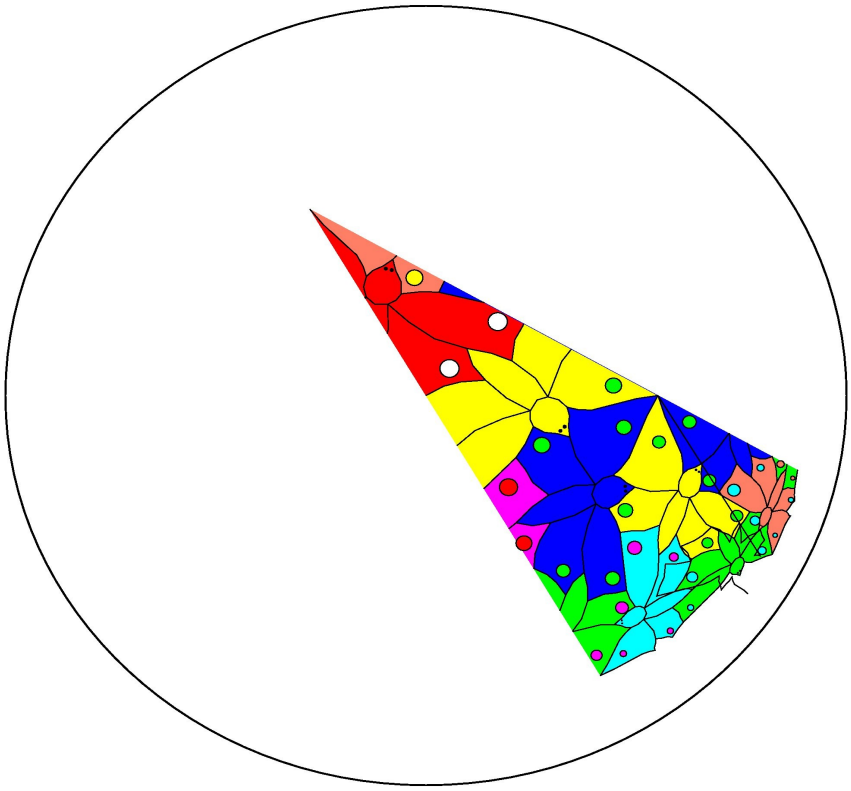

Figure 13: Stage 3

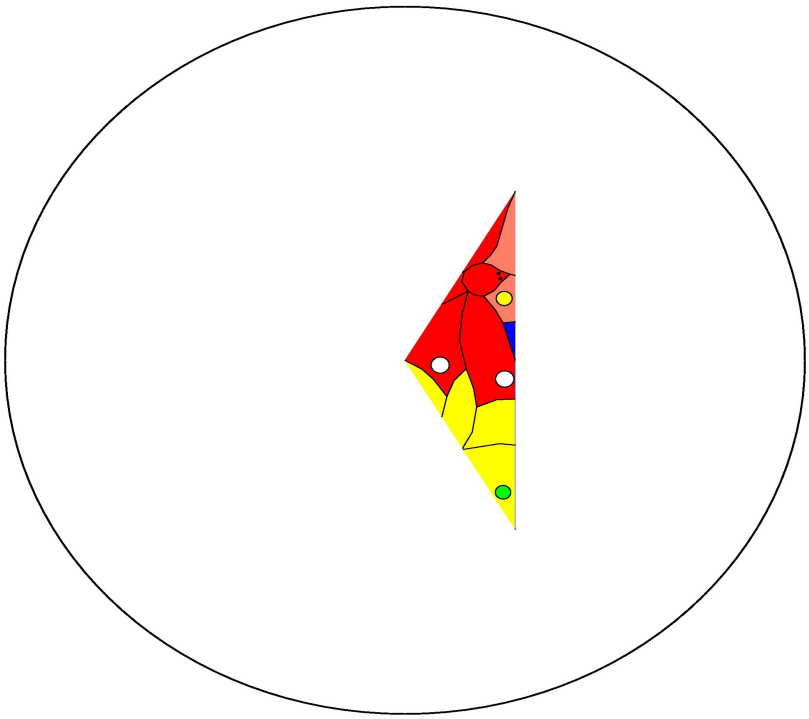

Figure 14: Stage 4

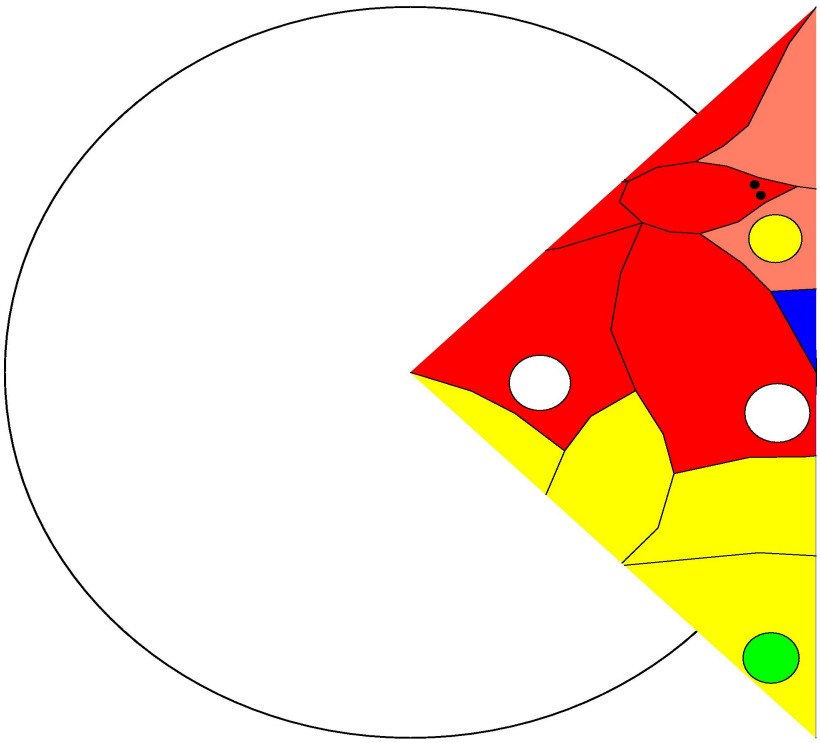

Figure 15: Stage 5 (Final Pattern)

4. Overview

The job of covering the polyhedra by the triangular fundamental regions is divided into five tasks. In this regard, five programs are used in succession to accomplish this job. The five tasks include (1) **subnet generation**, (2) **centroid generation**, (3) **applying transformations**, (4) **clipping,** (5) **Mapping final triangular patterns**. The sections below give a detailed overview of the steps involved in applying patterns onto the polyhedron.

4.1 Subnet Generation

The subnet generator accepts any three-dimensional polyhedron (convex or non-convex) as input. This program generates output to a file containing the information regarding the subnets of polyhedron [Section 9.1] as well as a postscript file containing the subnets of that particular polyhedron. The details of this algorithm are present in Kailash's thesis [4].

Instead of trying to find a net for the polyhedron, this algorithm fits as many faces of the polyhedron into the subnet as possible in such a way that there are no overlaps. If there are more faces, a seed face is chosen and then adjacent faces of the polyhedron are added to that seed face of the subnet of the seed polygon if these faces were not added to a previous seed. This process is repeated until all the faces of the polyhedron have been added to some seed. The details about the output file format of the algorithm are described in Section 9.1. This program is implemented in C.

4.2 Centroid Generation

The centroid generator algorithm finds the centroid of each face of the polyhedron. The program accepts the output file from the subnet generator algorithm as input to compute the centroid of each face. The input file of the algorithm contains the face count (number of faces of the polyhedron) of the polyhedron in the first line. The remaining lines contain the coordinates of the faces of the polyhedron of the subnet. The algorithm, upon accepting the input file, finds the centroid of each face and connects each vertex of the polygon to the centroid, dividing each polygon face into triangular regions.

Therefore, the output generated divides each face of a polyhedron into triangular regions. The output file which is generated looks similar to the input file except that the centroid of each face (Xc,Yc) is concatenated to the beginning of each of the lines containing the coordinates of the faces of polygon. The output file format of the centroid generator algorithm is presented in Section 9.2. This algorithm is implemented in C++.

4.3 Applying Transformations

Once we obtain the centroid for each face of polyhedra and divide each face into triangular regions, the next task is to apply transformations to each polygon. The transformed co-ordinates can in turn be used to map the motifs onto each face. The transformations used for generating patterns include rotation, scaling and translation.

We can assume that the fundamental equilateral triangle containing the motif is as shown in Figure 16 with (0,0),(1,1),(1,-1) as vertices. Now, this triangle has to be transformed to the figure shown in Figure 17 so that the clipped pattern can be mapped to the resultant transformed figure.

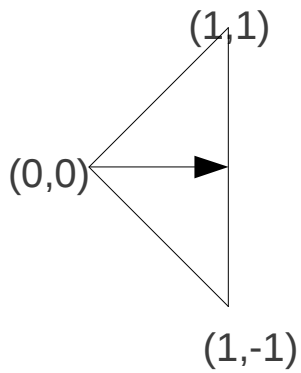

Figure 16

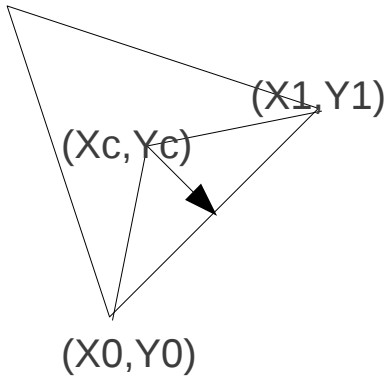

Figure 17

The following transformations are applied to Figure 16:

4.3.1. Scaling

Scale the triangle in Fig 16 so that the dimensions of the resultant triangle match the figure in Figure 17.

X-Scaling factor, Xscale = sqrt(sqr(Xc – (X_0+X_1)/2) + sqr(Yc – (Y_0+Y_1)/2))

--- (2)

Y-Scaling factor, Yscale = sqrt(sqr(X_1-X_0) + sqr(Y_1-Y_0))/2 --- (3)

The resultant scaling matrix is represented in Figure 18.

$$\begin{bmatrix} Xscale & 0 & 0 \\ 0 & Yscale & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure. 18

4.3.2. Rotation: Rotate the scaled fundamental triangle so that it is aligned with the triangle of the polygon in the subnet.

For Rotation,

cosΘ = ((X_0+X_1)/2 – Xc) / Xscale (Xscale is the X-scaling factor) ---(4)

sinΘ = ((Y_0+Y_1)/2 – Yc) / Xscale (Xscale is the X-scaling factor) ---(5)

The resultant rotation matrix is represented in Figure 19.

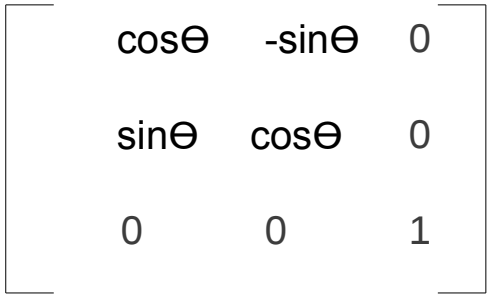

Figure 19

4.3.3. Translation

Translate the triangle so that the origin goes to the centroid.

X-Translation factor, Xt = Xc(Centroid) --- (6)

Y-Translation factor, Yt = Yc(Centroid) --- (7)

The resultant translation matrix is represented in Figure 20.

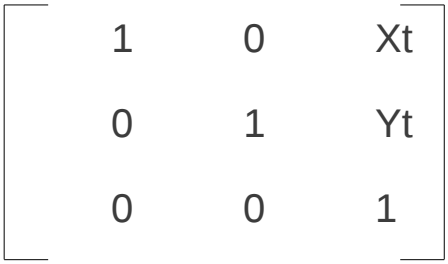

Figure 20

Once the transformed co-ordinates are computed, the clipped motif is now mapped to each of the faces which are divided into triangular regions.

4.4 Clipping

For clipping, we consider enough copies of the motif and clip the unwanted portions of the motifs so that the triangular fundamental regions is exactly filled with pieces of the motifs. The clipping process is basically done for circles, polylines and polygons.

4.4.1. Circles: This is one of the simplest cases of clipping. In case of circles, if the center of the circle lies inside the fundamental triangle, then the circle is retained. Otherwise, the whole circle is clipped off. We are basically trying to fit the entire circle into the pattern. This is not correct if a circle overlaps the fundamental triangle.

4.4.2. Polylines and polygons: The clipping of polygons is done by modifying the Sutherland-Hodgeman polygon clipping algorithm. An example of polygon clipping is shown in Figure 21.

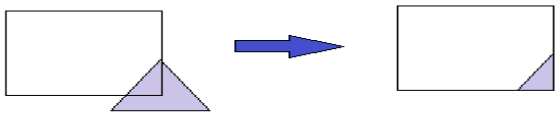

Figure 21

In the above figure, the edges of the triangle are clipped successively against the boundaries of the rectangle.

In case of polygons, there are four cases of clipping against one edge- the clip boundary. These four cases are analyzed in Figure 22. For analyzing various cases, consider the edge from vertex 'a' to vertex 'b' of the polygon to be clipped.

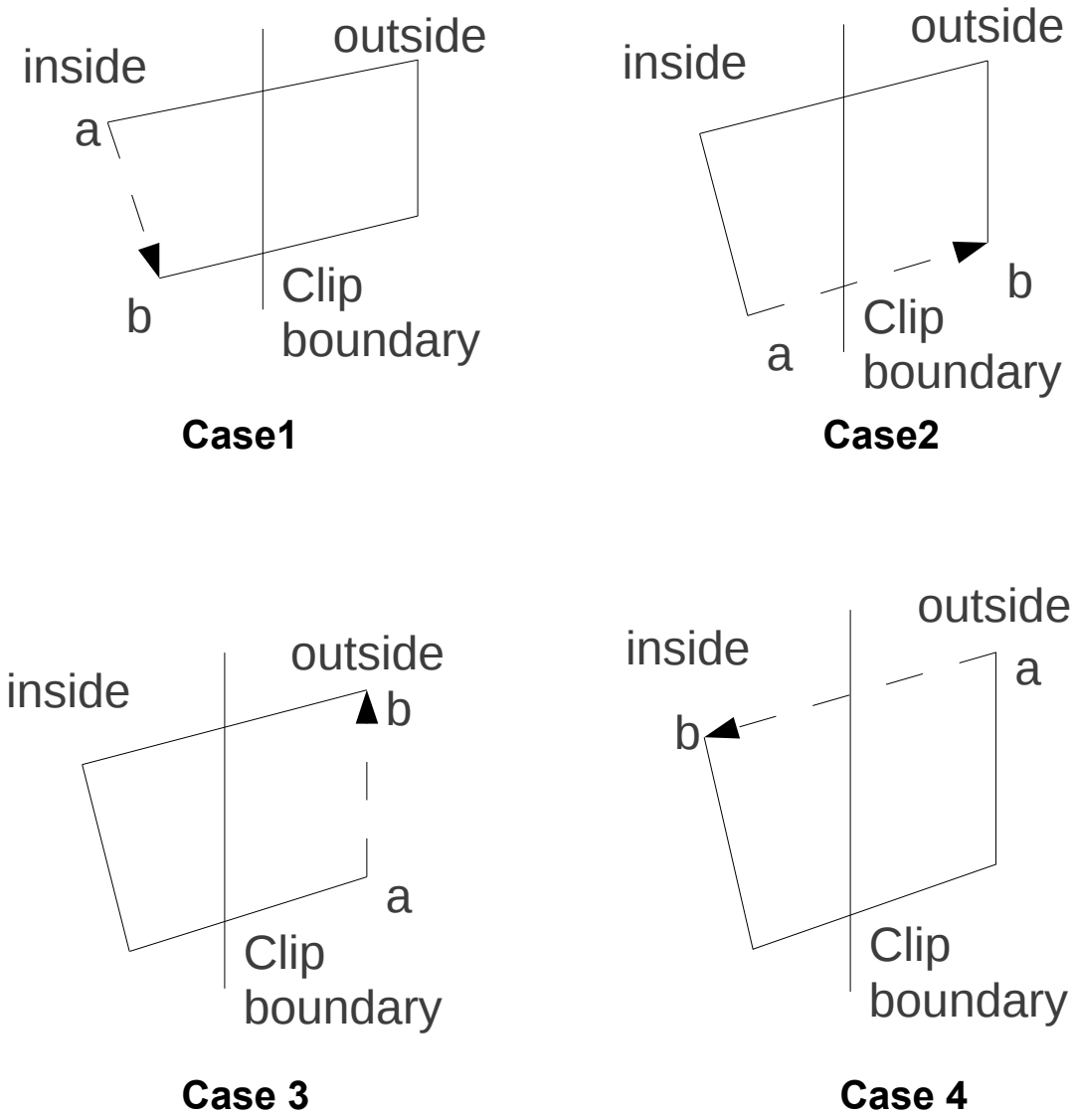

Figure 22

Case 1: Here the polygon edge is completely inside the boundary. In this case, the output vertex b is added to the output.

Case 2: In this case, one of the vertex lies outside the boundary. Hence, the intersection point of the boundary and edge is computed. The vertex 'b' is clipped off and the intersection point is added to the output.

Case 3: In this case, both the vertices 'a' and 'b' lie outside the boundary. So, no output is generated.

Case 4: In this case, one of the vertex lies outside the boundary. Hence, the intersection point of the boundary and edge is computed. Here, the vertex 'a' is clipped off and both the intersection point and vertex 'b' are added to the output.

In order to find the intersection point of a vertical boundary and the edge of the polygon, the concept of similar triangles is used.

Consider the Figure 23.

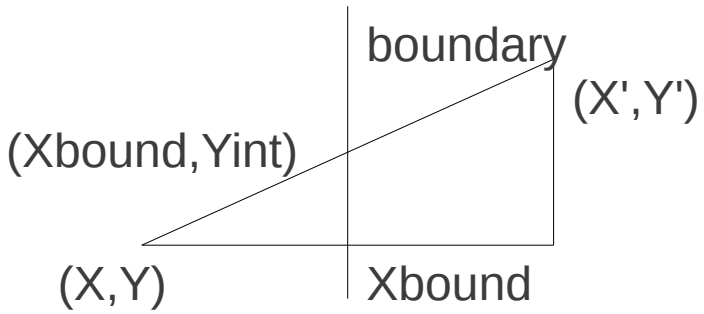

Figure 23

In Figure 23, to clip out the portion of line outside the boundary, we have to find Yint.

We find Yint by similar triangles,

$$(Yint - Y)/(Xbound - X) = (Y' - Y)/(X' - X) \text{ ---------(9)}$$

From (9), $Yint = Y + (Xbound - X) * ((Y' - Y)/(X' - X))$ -----(10)

Hence, the co-ordinates of the clipped line would be (X,Y) and (Xbound,Yint).

The clipping procedure can be analyzed in detail with the help of the figures. As an initial step, the unclipped pattern is considered. And then clipping is done in succession against the three sides/boundaries of the fundamental triangle.

5. Algorithms

5.1. Centroid Generator Algorithm

1. Open the output file of subnet generator.[4][section 9.1]
2. Store the face count of the input polyhedron. (This value has to be included in the output file).
3. For each of the remaining lines of input

    3.1. Read the co-ordinates from each line of input. Store x and y co-ordinates of each face of the polyhedron.

    3.2. Compute the area of polygon (each face of polyhedron) using the formula:

    A = $1/2(i=0\text{to}(N-1))\sum((Xi*Yi+1)-(Xi+1*Yi))$

    Here, A = Area,

    N = Number of vertices of polygon

    Xi,Yi = X and Y co-ordinates of each vertex of polygon respectively.

    3.3. Compute the centroid for each face of polygon for each polygon making use of the area of polygon which is computed in 3.2.

    Cx = $1/6\text{A}(i=0\text{to}(N-1))\sum((Xi+Xi+1)((Xi*Yi+1)-(Xi+1*Yi)))$

    Cy = $1/6\text{A}(i=0\text{to}(N-1))\sum((Yi+Yi+1)((Xi*Yi+1)-(Xi+1*Yi)))$

    Here, Cx, Cy are the co-ordinates of the centroid for each face of polyhedron.

Once we obtain the centroid for each face of polygon, each edge of the polygon will form a triangle with the centroid. These triangles along with the motifs form the fundamental regions of the polyhedra.

This algorithm was implemented in C++. The output of the centroid generator algorithm will look similar to the output of the subnet generator algorithm. The Cx, Cy co-ordinates are concatenated to the start of the x and y co-ordinates of each face of the polyhedron. The output file format of the centroid generator algorithm is presented in Section 9.2.

5.2. Algorithm for applying transformations

1. Read the input co-ordinates (which includes centroid) for each face from the input file.
2. Find the length of each side and compute Yscale [From (3)].
3. Calculate the mid-point of each side and hence calculate Xscale [From (2)].
4. Compute the resultant scaling matrix [Figure 18].
5. Calculate the sinϴ and cosϴ values for the rotation matrix [From (4) and (5)].
6. Compute the resultant rotation matrix [Figure 20].
7. Calculate the Xt and Yt values using centroid values from the input. This translates the origin to the centroid
8. Compute the resultant translation matrix [Figure 19].
9. Once all the three matrices are computed, Compute the result of

translate * rotate * scale matrices.

The co-ordinates, hence obtained, are the transformed co-ordinates of each of the triangular regions formed by the centroid. This algorithm is implemented in C++. The output co-ordinates of each triangular region is given as input to the clipping program.

The resultant matrix, hence obtained, is applied to each of the points of the motif. The x- and y- co-ordinates of each point are obtained from the data file presented in Section 9.3. Based upon the resultant co-ordinates, the x- and y-co-ordinates present in Section 9.3 are replaced by the newly obtained co-ordinates. These newly obtained co-ordinates are used to map the motifs into each face of the polyhedron.

5.3. Clipping Input pattern data

Dr. Dunham has many data files describing motifs for repeating hyperbolic patterns. Most of these motifs are based on one of the p isosceles triangles (with angles 2 pi / p, pi/q, and pi/q) that make up a regular p-sided polygon of the regular tessellation {p,q} composed of regular p-gons meeting q at each vertex ((p-2)(q-2) must be greater than 4 for the tessellation to be hyperbolic). Usually the motif is contained in one of the p isosceles triangles with angles 2 pi/p, pi/q, pi/q that make up the p-gon. Such isosceles triangles are the fundamental triangular regions mentioned in Section 3.5. In such cases, motifs can transformed to Euclidean motifs that are suitable for filling Euclidean polygons of a subnet for a polyhedron. One problem is that most of

the existing motifs overlap the equilateral triangles and also have unfilled gaps corresponding to the overlaps. The solution is to generate a larger pattern that includes all the motifs in adjacent equilateral triangles and then clip the unwanted parts of the motifs outside the fundamental equilateral triangle of interest. The motifs are defined in the Poincare circle model of hyperbolic geometry, so that a central fundamental triangle would have two radii and one edge of the polygon as its sides. By transforming the motif from the Poincare model to the Klein circle model, the sides of the fundamental triangle become Euclidean line segments, which is simpler for clipping. Currently, we can clip polylines and filled polygons that make up the motif, using the Sutherland-Hodgman clipping algorithm for filled polygons and a modification of it for polylines. In the future, it would be useful to extend clipping to circles and filled circles, since some of the motifs contain those elements. Once the motif is clipped to an isosceles triangle in the Klein model, it is scaled in the x- and y-directions so that it fits in the Euclidean 45-45-90 isosceles triangle with vertices (0,0), (1,1), and (1,-1), which can then be further scaled, rotated, and translated to fit in any of the polygons of the subnet.

6. Results

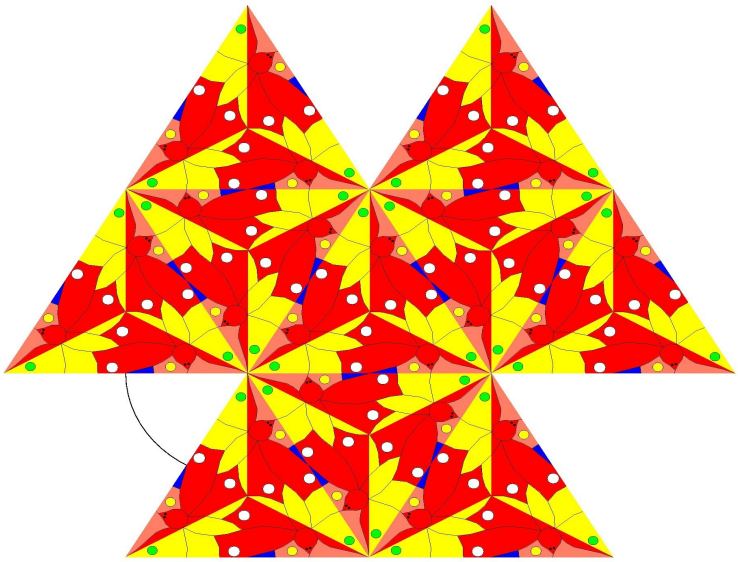

Fig 24: Patterned subnet of icosahedron

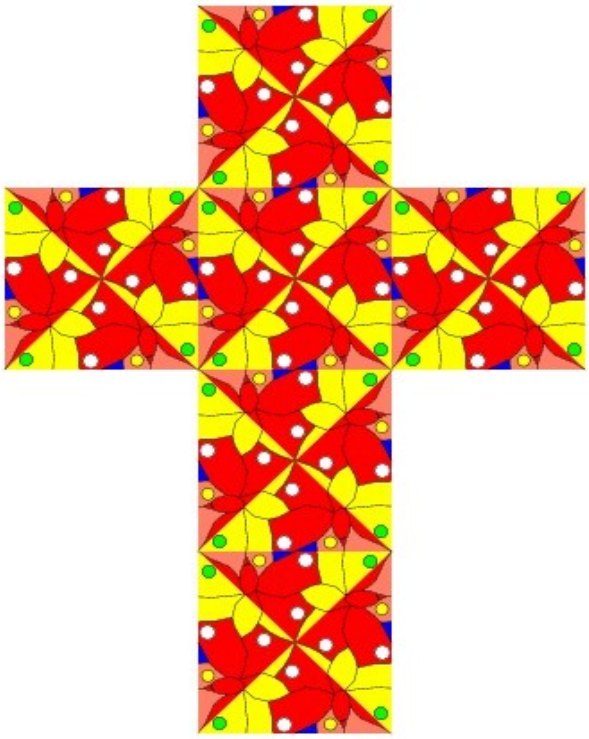

Fig 25: Patterned subnet of cube

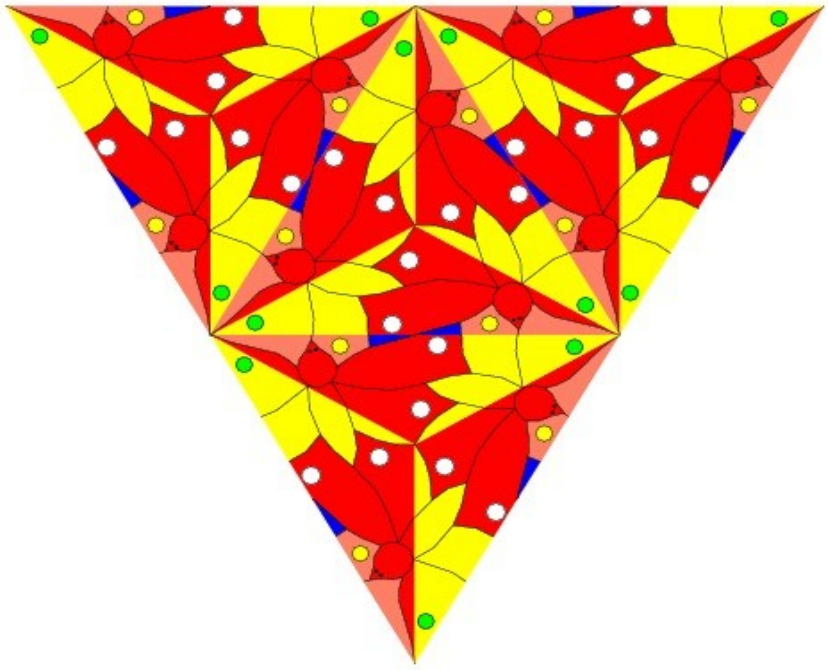

Fig 26: Patterned subnet of tetrahedron

7. Conclusion and Future work

We have mapped a motif within the triangular fundamental region to each of the polygon faces of a polyhedron. Hence, a single pattern is mapped onto each of the polygon of a net in order to construct a patterned polyhedron. We have successfully automated the process of generating the patterns in a regular way based upon the triangulations of the polygons. This research has helped Dr. Douglas Dunham in his research on polyhedra and geometric patterns.

7.1 Future Work

Currently the clipping algorithm clips polylines and filled polygons. In the case of circles, if the center of the circle lies inside the boundary, then the circle is retained; otherwise, the whole circle is clipped off. This is not correct if a circle overlaps the fundamental triangle since we are trying to fit the whole circle into the pattern. Hence, further work should be done to clip the circles appropriately.

The current algorithm has successfully generated patterns onto the triangulations of each polygon in a polyhedron. But the patterns generated onto each subnet of the polyhedron lacks color coherence. Hence, the algorithm must be improved to allow for color symmetry so that motifs are colored consistently.

8. References

[1] ”Polyhedra”, Cromwell. Peter Cambridge University Press,1999

[2] Douglas Dunham, “168 Butterflies on a Polyhedron of Genus 3”, Bridges 2002 Conference Proceedings, July 2002, Baltimore, Maryland, pp. 197-204. ISBN 0-9665201-4-9

[3] Ankur Nepalia, ”Creating repeated patterns on polyhedra” Computer Science Graduate Thesis, University of Minnesota, Duluth 2008

[4] Aurangabadkar, K., “Generating subnets for polyhedra,” Computer Science Graduate Thesis, University of Minnesota, Duluth 2004

[5] Bern, M., Demaine, E.D., Epstein, D., Kuo, E., Mantler, A. and Snoeyink, J. “Ununfoldable polyhedra with convex faces.” Computational Geometry: Theory and Applications Vol. 24, Issue. 2, 51 – 62, 2003

[6] http://www.georgehart.com/virtual-polyhedra/crystals.html

[7] Branko Grünbaum , “A starshaped polyhedron with no net ”. Geombinatorics 11(2001), 43 – 48.

[8] Computer Graphics: Principles and Practice in C, 2/E. ISBN-10: 0201848406 Publisher: Addison-Wesley Professional.

[9] Sameer Atar, “Generating repeating patterns on subnets of a polyhedron” Computer Science Thesis, University of Minnesota, Duluth 2006

[10] Buekenhout, F. and Parker, M. "The Number of Nets of the Regular Convex Polytopes in Dimension <=4." Disc. Math. 186, 69 – 94, 1998.

[11] http://mathworld.wolfram.com/PoincareFormula.html

9. Appendix

9.1. Output data for Subnet Generator:

fc

X_{10} Y_{10} X_{11} Y_{11} X_{12} Y_{12} X_{13} Y_{13}

X_{20} Y_{20} X_{21} Y_{21} X_{22} Y_{22} X_{23} Y_{23}

X_{30} Y_{30} X_{31} Y_{31} X_{32} Y_{32} X_{33} Y_{33}

..... X_{ni} Y_{ni}

Here, fc is the face count

X_i Y_i are the co-ordinates of each of the face.

X_{ni} Y_{ni} correspond to the number of lines of input co-ordinates.

Here n = face count.

9.2. Output data for Centroid Generator:

fc

X_c Y_c X_{10} Y_{10} X_{11} Y_{11} X_{12} Y_{12} X_{13} Y_{13}

X_c Y_c X_{20} Y_{20} X_{21} Y_{21} X_{22} Y_{22} X_{23} Y_{23}

X_c Y_c X_{30} Y_{30} X_{31} Y_{31} X_{32} Y_{32} X_{33} Y_{33}

..... X_{ni} Y_{ni}

Here, fc is the face count

X_i Y_i are the co-ordinates of each of the face.

X_{ni} Y_{ni} correspond to the number of lines of input co-ordinates.

Here,n = face count.

X_c Y_c is the centroid of each face of polyhedron.

9.3. Sample data file used for clipping

3 7 1 0 9 0

1 3 4 2 9 6 5 8 7

1 2 3 4 5 6 7 8 9

1 1 4 5 2 3 7 6 9 8

1 1 2 9 3 4 5 6 7 8

1 1 3 7 4 2 9 6 5 8

155

0.000000000000000e+00 0.000000000000000e+00 3 4 3

1.510406132892085e-01 -5.130905907913125e-02 3 5 3

6.067808169527305e-01 5.607401468248555e-01 1 10 3

7.021587100891782e-01 5.765674029716744e-01 1 11 3

3.342113596522163e-01 3.342113596522163e-01 1 9 3

8.543475579385948e-01 4.918018260696986e-01 1 8 3

3.189879277470568e-01 -2.857503154365017e-02 2 8 3

1.000000000000000e+00 0.000000000000000e+00 1 11 3

1.000000000000000e+00 1.000000000000000e+00 9 4 3

….....

Here, the first six lines are related to the hyperbolic geometry. Since, we are dealing with the Euclidean geometry, these lines are ignored.

The next line consists of a single number. This number indicates the total number of remaining lines in the file. In this case, the input file consists of 155 lines. Each line specifies a point and has the following format:

x-coordinate y-coordinate color type-of-point number-of-layers

where X-coordinate and y-coordinate are related to the central pgon.

Color ranges from numbers 1 to 10.

The type of point ranges from :

1 -- a "Move To"

2 -- a "Draw To"

3 -- a "Circle" (there should be two of these in succession)

4 -- start a (Euclidean) "Filled Poly"

5 -- continue a (Euclidean) "Filled Poly"

6 -- end a (Euclidean) "Filled Poly"

7 -- a "Hyperline" (there should be two of these in succession)

8 -- a "Filled Circle" (there should be two of these in succession)

9 -- start a (Euclidean) "Polyline"

10 -- continue a (Euclidean) "Polyline"

11 -- end a (Euclidean) "Polyline"

12 -- start a (hyperbolic) "Filled pgon"

13 -- continue a (hyperbolic) "Filled pgon"

14 -- end a (hyperbolic) "Filled pgon"

The number-of-layers is not used.