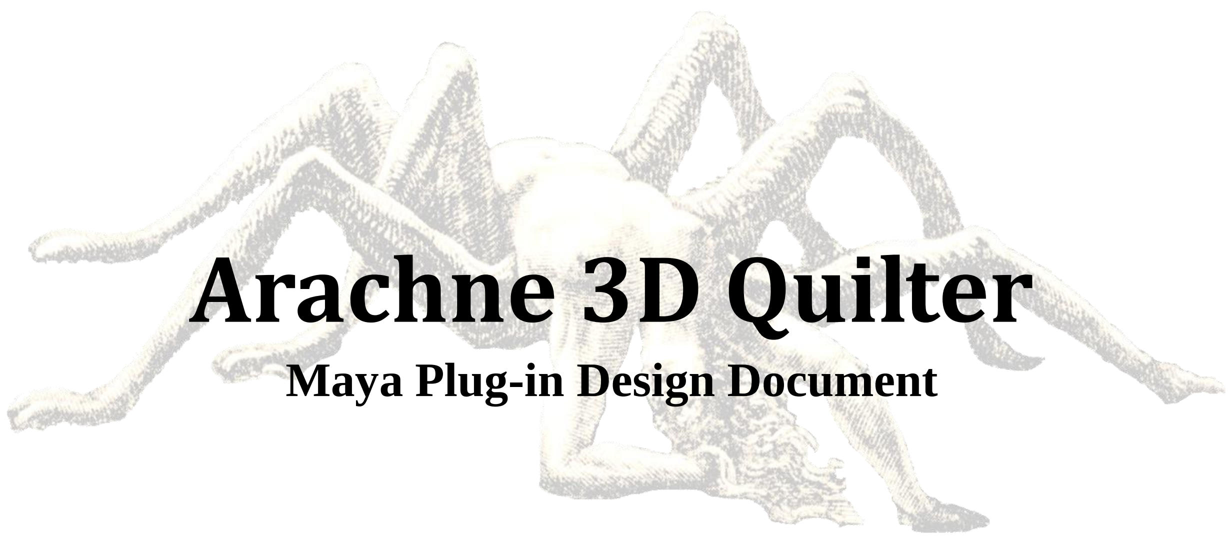


Arachne 3D Quilter

Maya Plug-in Design Document

Daniel Garcia

Stephen Lovejoy

Based on:

Mesh quilting for geometric texture synthesis.

Zhou, K., Huang, X., Wang, X., Tong, Y., Desbrun, M., Guo, B., and Shum, H. 2006. ACM Transactions on Graphics 25, 3, 690-697.

PROJECT SUMMARY

Our project goal is to build a Maya plug-in that is capable of producing a 3D quilted mesh around an arbitrary polygon surface given an arbitrary mesh texture. It will integrate with Maya's modeling workflow, which currently has no similar functionality for this type of task.

Replicating the results of our tool would be difficult and time consuming if done by hand. Normally, if a similar effect is desired, artists will use a variety of texture based methods to simulate the quilted effect. However there are cases where simple texture maps will not provide the required level of detail. This is where our tool will be most useful.

Our tool, which will be designed for 3D artists, will be simple to use and produce aesthetically pleasing results. The interface for using this tool consists of one window with clear indications for what the user is to do. In additional to a minimal interface, no technical knowledge of the underlying algorithms needs to be known by the user. Once they get the quilt as output, they can modify it like any other polygonal mesh.

The algorithms as described in the SIGGRAPH paper for this tool build upon the traditional texture synthesis methods by extending it to provide a mesh-based texture synthesis. They also have a method that stitches and deforms the mesh-based texture with minimal distortion around the features of the base mesh. Minimizing the distortion is the key to the resulting quilt being aesthetically pleasing. Throughout the algorithm, several methods from other research papers are utilized. We aim to achieve similar performance in stitching speed as compared to the moderate times shown in the paper.

The planned schedule calls for two alpha versions, a beta version, and a final version. The order of tasks for the following weeks closely follows the multistep layout of the algorithm presented in the paper. By handling each step of the algorithm on its own in order, the plug-in's development will be streamlined where one step is not waiting for another step to be finished to continue.

1. AUTHORING TOOL DESIGN

1.1. Significance of Problem and Production/Development Need

Traditionally, if a 3D artist wants to model something with a detailed repeating surface covering an object, they would have done so using texture mapping. However, as the complexity of 3D scenes and geometry have greatly increased over time, texture mapping is unable to provide the needed increase as well. Bump mapping and volumetric textures were introduced to compensate, but still have the artifacts inherent to texture mapping methods.

By creating a 3D quilted mesh around the surface of another object, we get the maximum visual fidelity of the desired effect, as well as an easy way to manipulate or simulate the resulting quilt. It would be possible to achieve this effect by hand, however the time and precision required for such a task would be unfeasible, especially when dealing with complex textures. This tool will streamline the process, automatically creating the quilt which will leave the artist to focus on the model as a whole.

1.2. Technology

This tool is based on the 2006 SIGGRAPH paper "Mesh Quilting For Geometric Texture Synthesis" by Kun Zhou, Xin Huang, Xi Wang, Yiying Tong, Mathieu Desbrun, Baining Guo, and Heung-Yeung Shum. It builds upon the traditional texture synthesis methods by extending it to provide a mesh-based texture synthesis around an arbitrary given mesh. In addition, they have a method that stitches and deforms the mesh-based texture with minimal distortion around the features of the base mesh.

Throughout the quilting algorithm and process, the paper cites the use of several techniques presented in other papers. These include:

- The sub-patch matching technique in Kwatra et al. 2003
- Laplacian-based mesh editing and mesh merging techniques in Yu et al. 2004
- The graph cut algorithm in Boykov et al. 2001
- Discrete conformal mapping from Desbrun et al. 2002
- Controlling the direction of synthesis using a vector field from Praun et al. 2000
- Low distortion parameterization of triangle meshes from Sander et al. 2001

We chose this paper as the basis for a Maya plug-in since, not only does Maya not have this type of functionality built-in, but also that it would fit in nicely with Maya's modeling workflow. Also, using various mesh textures on a base mesh can give intricate and unique effects that are not typically found in 3D modeling.

1.3. Design Goals

Our tool will address the stated problem by providing a fast and simple interface for creating 3D quilted meshes using the techniques in the above paper. It presents a method to wrap a 3D texture sample, given as a triangle mesh, seamlessly around the shell of an arbitrary object. The wrapping is done through local stitching and deformation of the sample mesh around the object. This is typically needed when a stitched texture on the object would not give the desired effect or level of detail required of an object being wrapped up by another object. Also, objects that are made entirely out of the repeating mesh can quickly be made using this tool with a separate shell mesh to shape the quilt, such as wicker furniture.

1.3.1 Target Audience

This will be an artist's tool, mainly for modelers. Since the tool will require the base shell and mesh texture to be already constructed, the artist that made those meshes will most likely be the person to use our tool to combine the two. Also, the quality of the result of the tool is determined primarily by its aesthetics. So an artist will be the most qualified to change or modify the output as needed.

1.3.2 User goals and objectives

With this tool, the user will be able to quickly and easily create a repeating mesh-based texture that surrounds a given polygon surface. For their given mesh inputs, they should be able to have a reasonable sense of how the quilt will synthesize on the shell by their appearance and structure. The user should spend a minimal amount of time with this tool, less the computation time. Also, the various algorithms operating behind the scene should be invisible during its use.

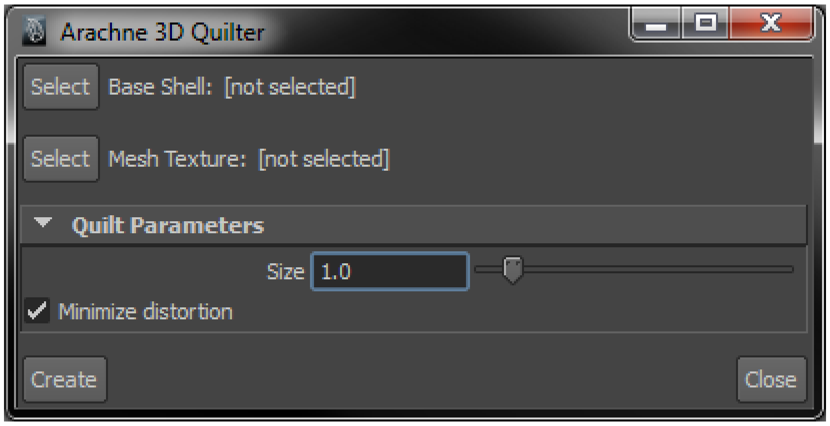
1.3.3 Tool features and functionality

In addition to the tool creating a 3D mesh quilt, it will have the ability to change the parameters that go into forming the quilt. Those parameters include the overall scale of the mesh-based texture, and whether or not the low distortion parameterization of the shell space is used for the output. The tool should also indicate an estimated time remaining during its synthesis process

1.3.4 Tool input and output

The input is a polygonal mesh to serve as the shell, along with a triangle mesh to serve as the 3D texture. The output is a contiguous mesh that not only wraps around the given 3D object, but that also appears undistorted and smooth in its construction.

1.4. User Interface

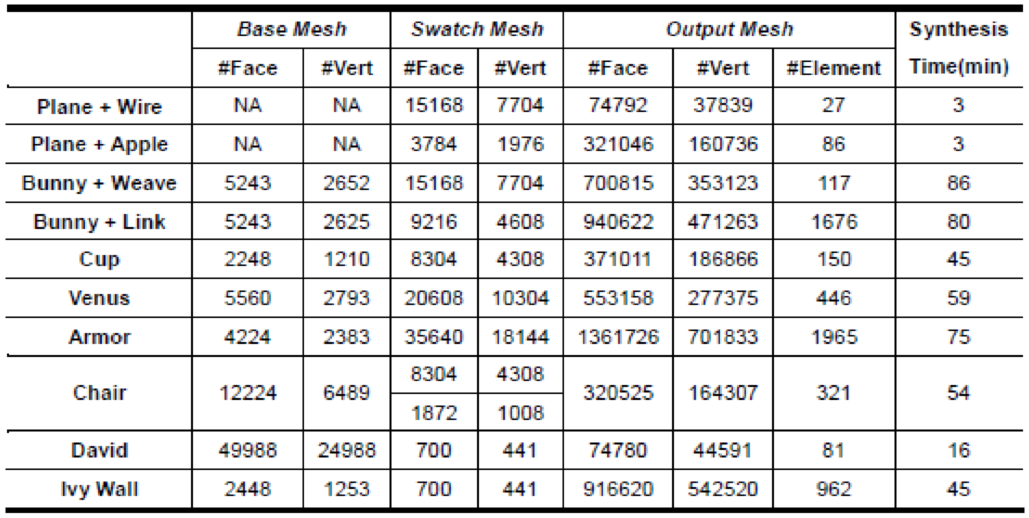


1.4.1 GUI Components and Layout

The GUI for this tool is relatively minimal, consisting of only one window containing the controls for the input to our tool. Much of what determines the final output from our tool comes from the base shell and mesh texture created within Maya.

Once our interfaced is opened, all that is required is for the user to select the base shell and mesh texture by selecting the corresponding object in Maya and then pressing the corresponding select button in our GUI. If the user elects to, they can easily modify some of the parameters that influence the quilt generation before creating it.

Also, due to the long synthesizing times associated with this tool, as shown below, a progress indicator will be used after the create button is pressed to let the user know how long the computation should take.



	Base Mesh		*Swatch Mesh*		*Output Mesh*			Synthesis
	#Face	#Vert	#Face	#Vert	#Face	#Vert	#Element	Time(min)
Plane + Wire	NA	NA	15168	7704	74792	37839	27	3
Plane + Apple	NA	NA	3784	1976	321046	160736	86	3
Bunny + Weave	5243	2652	15168	7704	700815	353123	117	86
Bunny + Link	5243	2625	9216	4608	940622	471263	1676	80
Cup	2248	1210	8304	4308	371011	186866	150	45
Venus	5560	2793	20608	10304	553158	277375	446	59
Armor	4224	2383	35640	18144	1361726	701833	1965	75
Chair	12224	6489	8304	4308	320525	164307	321	54
			1872	1008				
David	49988	24988	700	441	74780	44591	81	16
Ivy Wall	2448	1253	700	441	916620	542520	962	45

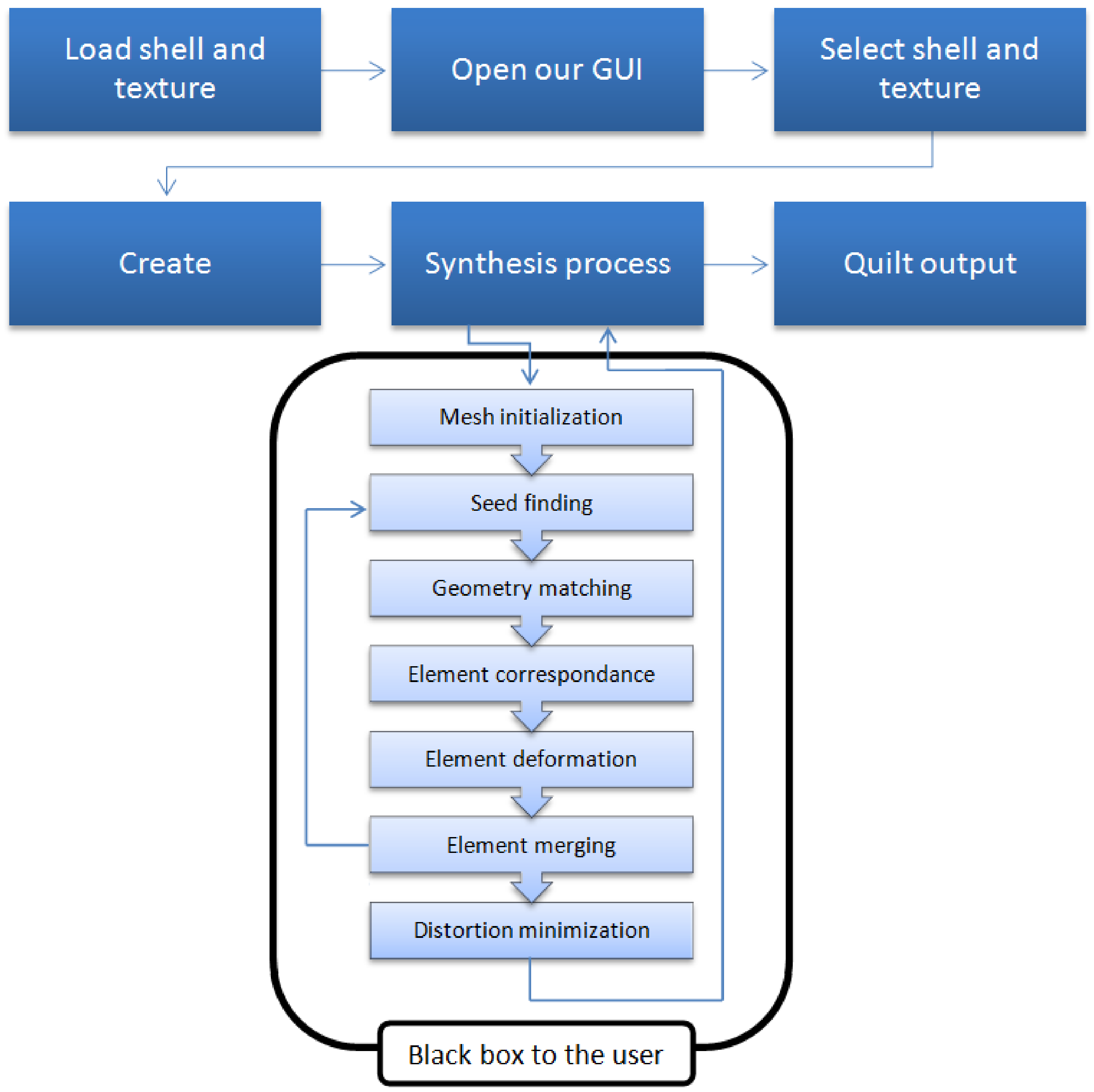
Table 1: *Statistics on all base meshes, texture samples and synthesis results presented in this paper.*

1.4.2 User Tasks

There are parameters that are used in the quilt generating algorithm available for the user to modify before it is created. The user can modify the size parameter for the mesh texture, as well as if the quilt will be generated with minimal distortion. By default the quilt will have a size modifier of one and will minimize the distortion of the shell mapping using the described algorithms in the SIGGRAPH paper.

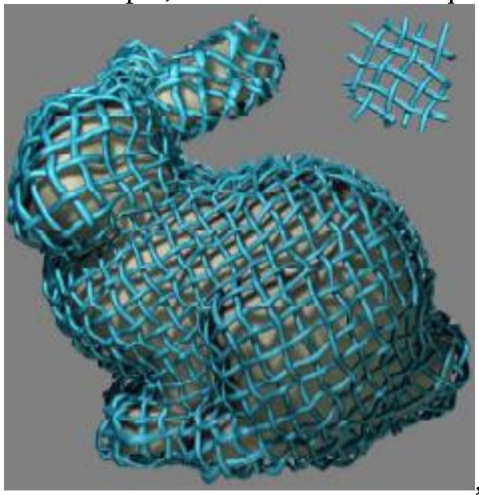
The user does not need to know the technical background behind the tool. However to produce aesthetically pleasing results, they will need to know what type of output they should expect from it. Normally the user will want to leave the minimize distortion option checked. If they want to intentionally have a quilt with distortion, they will need to know what types of base meshes are susceptible to distortion. Since every artist will be looking for something different, they will either need to experiment with some of the sample scenes provided, or go through trial and error to understand how different mesh textures react to being quilted on different surfaces.

1.4.3 Work Flow



Before using our tool, the user will need to have a mesh texture patch and a base shell to wrap it around ready. Once they are both in the same scene in Maya, the user will open our GUI window, select the base shell and mesh texture, then click create. After the synthesis computations by the tool, the user will have a polygonal mesh quilt around their designated base shell. They can then edit it as they would any other mesh for their purposes.

For example, to create the woven quilt around the bunny as shown here,



, first the underlying bunny and the mesh square in the upper right of the image would need to be placed in the same Maya scene. Then, once our GUI is opened, select the bunny as the base shell and the mesh square as the mesh texture. In this case the default quilt parameters will be used. Then, after clicking create, the bunny will have the woven quilt around it as show in the picture.

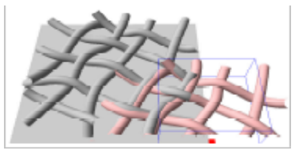
In that simple work flow, compared to the original where the user would have to model the quilt by hand, our tool provides a large shortcut, greatly reducing the time invested as well as producing a much more consistent stitched result compared to what would result from doing it by hand.

2. AUTHORING TOOL DEVELOPMENT

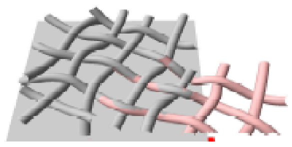
2.1. Technical Approach

2.1.1 Algorithm Details

The algorithm for creating a quilt on a 2D plane is as follows:
The 2D plane is split into numerous cells, and at the start of each iteration of the algorithm, the seed cell with the most already-processed cells is chosen.



Next we pick a small output-sub-patch of cells around the seed cell. We now look for translations of the input mesh texture such that the portion of the input overlapping the output-sub-patch matches it best. Of course, only translations that allow complete overlap of the input with the output-sub-patch are considered. The translation is chosen to minimize a specific energy function that will assure the optimal overlap between the input mesh square and the already established mesh.



Once the translation is found, we build correspondences between the input mesh and the established mesh. This is done by looking in the overlapping region and finding the closest input face to each vertex in the established mesh. This is pruned for false positives and then we tally the corresponding input elements for each element in the established mesh.

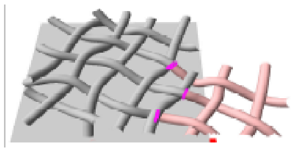
Then, for each output (established) element corresponding to an input element, both elements are deformed to better align them. The deformation is based on Laplacian-based mesh editing techniques, where the Laplacian coordinates of each vertex is:

$$\mathscr{L}(\mathbf{v}^i) = \mathbf{v}^i - \frac{1}{\#\mathscr{N}(\mathbf{v}^i)} \sum_{\mathbf{v}^j \in \mathscr{N}(\mathbf{v}^i)} \mathbf{v}^j,$$

Where $N(v^i)$ is the 1-ring vertex neighbors of v^i. The follow energy equations are minimized for the input and output elements, respectively:

$$E_{in}(\{\mathbf{w}^i\}) = \sum_{i=1}^{N_{in}} \|\mathscr{L}(\mathbf{w}^i) - \mathscr{L}(\mathbf{v}^i_{in})\|^2 + \mu \sum_{i=1}^{m} \|\alpha^i \mathbf{w}^{i,1} + \beta^i \mathbf{w}^{i,2} + \gamma^i \mathbf{w}^{i,3} - \mathbf{c}^i\|^2.$$

$$E_{out}(\{\mathbf{w}^i\}) = \sum_{i=1}^{N_{out}} \|\mathscr{L}(\mathbf{w}^i) - \mathscr{L}(\mathbf{v}^i_{out})\|^2 + \mu \sum_{i=1}^{m} \|\mathbf{w}^i - \mathbf{c}^i\|^2,$$



Finally, the elements are merged together. Every element without correspondence is added to the complete mesh. For each corresponding element pair, if one is entirely within the overlapping region, it is ignored. If not, the proper stitching location needs to be found. This is done by creating a network flow graph for the output elements representing dual graph adjacency between triangles. For each pair of triangles sharing an edge (v^i,v^j), the edge weight is:

$$(1 + \|\mathbf{v}^i_{out} - \mathbf{v}^j_{out}\|)(1 + Dist(\mathbf{v}^i_{out}, C_{in}) + Dist(\mathbf{v}^j_{out}, C_{in})),$$

Where the Dist function computes the shortest distance from the triangle to the input element. A source and sink node are added. Triangles outside of the overlapping region are linked to the source node with an infinite-weight edge, and Suppose that a vertex v^i_{out} in C_{out} has a closest face f^i_{in} in C_{in}. If f^i_{in} lies outside of the overlapping region or there exists a face which is adjacent to f^i_{in} and does not have any corresponding vertices in C_{out} , then all triangles sharing vertex v^i_{out} are linked to the sink node with infinite weight to assure that they will be deleted. A min-cut algorithm is then run to find the proper cut location. The same is done with the input elements.

Cut elements are stitched together by using mesh merging. We simply set the average boundary points as position constraints and deform the two cut elements using the deformation energy defined in the earlier equation.

In order to convert the above algorithm to a 3d mesh, the following changes are made:

First, a scale parameter is added to specify the size of the swatch in comparison to the 3d surface.

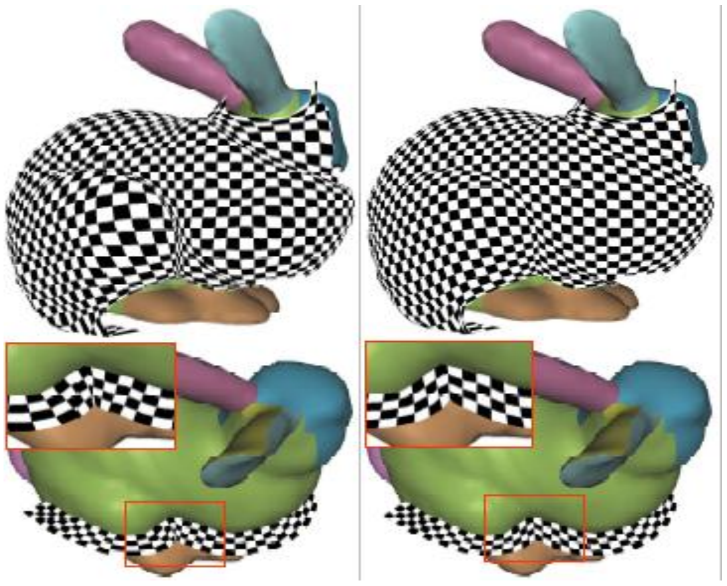
A surface patch is defined by starting from the chosen triangle and growing the region using breadth-first traversal until we reach a certain depth or when the total area of the patch exceeds a user-defined threshold.

The base mesh replaces the 2d grid, with triangles of the mesh replacing the individual cells of the grid.

The surface patch is flattened to a 2d square using discrete conformal mapping. The local operations for mesh quilting can be done on this parameterized plane, then the positions of the new vertices can be reprojected onto local mesh-based coordinates.

Guided vector fields are also used as an input in the case that it is necessary to control the orientation of the texture over the surface.

The vertex positions of the new mesh, once computed in the local coordinates relative to the 3D surface, must be converted to global coordinates. In order to do so, we create a shell space around the 3D surface and map the vertices to the shell space.



In order to map the vertices properly without distortion, we map the vertices through optimizing a stretch metric on the tetrahedral mesh. We define g as the mapping between a point in shell space to a point in texture space with a Jacobian $J = [\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}, \frac{\partial g}{\partial z}]$
And π_1, π_2, π_3 as the eigenvalues of the Cauchy defomation tensor J^TJ. The root-mean-square stretch is calculated as:

$$L^2(g,T_s) = \sqrt{(\pi_1+\pi_2+\pi_3)/3} = \sqrt{(a_g+b_g+c_g)/3},$$

with $a_g = \frac{\partial g}{\partial x}\cdot\frac{\partial g}{\partial x}$, $b_g = \frac{\partial g}{\partial y}\cdot\frac{\partial g}{\partial y}$ and $c_g = \frac{\partial g}{\partial z}\cdot\frac{\partial g}{\partial z}$.

And the total stretch as:

$$L^2(g,M) = \sqrt{\sum_i \left(L^2(g,T_s^i)\right)^2 |T_s^i| / \sum_j |T_s^j|}$$

We optimize the mapping to minimize this measure to find a more even mapping over the 3D surface.

Describe the main features and details of the algorithms you plan to implement. List any assumptions or simplifications you will be making.

2.1.2 Maya Interface and Integration

The MEL code will largely be for an interface and input mechanisms. The user will give a 3D surface for the quilt to wrap around, a mesh as a texture sample, a scaling factor, and an optional vector field.

The actual running of the algorithm and creation of the quilting mesh will be done within the C++ API. It will receive the mesh data and other inputs from the MEL script and perform the necessary iterations of the algorithm, then properly place the mesh around the 3D surface. The API will then return the mesh into a new Maya object.

We will make use of the MFnMesh objects for Maya for much of the calculations, as it provides the functionality that we require.

Describe how you plan to implement the algorithms in the Maya runtime environment. What features will be implemented in MEL? What features will be implemented in the C++ plug-in. Provide descriptions of Maya objects, data structures and class hierarchies you plan to use, as appropriate.

2.1.3 Software Design and Development

Cell

A basic cell data structure will be used, corresponding to triangles in the 3D case and a basic cell in the 2D case. It will contain information on surrounding cells, so we are able to determine which cells are surrounded by the most previously computed cells. A patch will simply be a set of cells.

Graph

A basic graph structure will be created to allow for running of the network flow algorithm to determine the proper cutting points for mesh fusing.

Vector Fields
As vector fields may be employed in the 3D case, a vector field object defining the field over the surface mesh will be created.

Local Mesh Coordinates
In the 3D case, it is necessary to store the coordinates of the new mesh with respect to the surface mesh instead of in world space. In order to factor for this, we will create a new coordinate object to contain the relative coordinate information for a given vertex. There will be a mapping between this new coordinate data and the vertices being created for the new mesh.

2.2. Target Platforms

2.2.1 Hardware

The hardware requirements for running our tool are equivalent with the hardware requirements for running Maya. More memory and faster processing speeds will decrease computation time and speed up rendering, but are not necessary to make use of this tool. The requirements for running Maya as are the following:

- Windows: Intel® Pentium® III or higher, or AMD Athlon™ processor
- 512 MB RAM
- CD-ROM Drive
- Hardware-Accelerated OpenGL® graphics card
- 3-button mouse with mouse driver software
- 450 MB of hard disk space

We recommend more than 512 MB of RAM for the detailed quilts and an Intel® Core 2 processor or better for faster performance.

2.2.2 Software

The tool is intended to run on a Windows platform. Since this tool will be developed as a Maya plug-in, it will require the user to own a copy of Maya 7 or greater.

2.3. Software Versions

2.3.1 Alpha Version Features (first prototype)

Alpha 1
The alpha 1 prototype will allow for an input of a quilt sample, which will be extended over a 2D plane according to the quilting algorithm, although the element deformation will not be fully implemented. A mesh will be returned, though there will not be complete connectivity due to the lack of deformation.

Alpha 2
Similar to the above, but with element deformation completed, so a fully functional mesh creator for the 2D plane.

2.3.2 Beta Version Features

The 3D version will be implemented, allowing for the creation of a mesh over a 3D surface. However, distortion minimization will not be implemented, instead using a simple conversion to global coordinates. There will be some stretching of the mesh as a result.

2.3.3 Description of any demos or tutorials

Inputting a 3D surface and quilt sample should be simple to perform with basic instructions. There may be an added tutorial for creating a vector field if Maya does not have one already implemented. Also, sample Maya scenes containing mesh textures and polygon shells ready to use will be provided for experimentation

3. WORK PLAN

3.1. Tasks

The tasks planed over the following weeks closely follow the step-by-step breakdown of the 3D quilting process as detailed in the SIGGRAPH paper. By dividing our milestones this way, we keep different tasks from overlapping on each other so that we can focus on the task at hand and handle its construction in a logical manner. Finally, we will both be handling each task to keep the interconnectedness of the underlying algorithm as smooth as possible instead of separating it.

3.1.1 Alpha Version

Design Document: 6 days
This document will detail the various aspects that are associated with the tool we plan to build. In it contains the background information about the underlying technology, our design goals for the tool, how the user will interact with out tool, the technical details about its implementation, and a schedule of tasks to be completed. Having this document will give us a guide for the order of tasks to complete, and how we should allocate our time efficiently.

Setup/Seed Finding: 10 days
Since this will be the starting point for our development, there is an initial setup of our development environment and associated tools to prepare for the construction of our plug-in. While doing that, we will be developing the seed finding portion of the algorithm. This step will provide an appropriate cell of the 3D texture for the following steps to use.

Geometry Matching: 10 days
This task implements the corresponding next step in the main algorithm shown in the paper. It will take as an input the selected region from the seed finding step, compute the optimal patch placement to expand the quilt, and send that information as its output to the next step. Research into the sub-patch matching technique in Kwatra et al. 2003 will be needed.

Element Correspondences: 5 days
This task implements the corresponding next step in the main algorithm shown in the paper. Taking the output from the previous step, it calculates the similarities in geometry between the quilt and the new patch, and outputs the overlapping elements of the geometry that correspond to each other.

Element Deformation: 12 days
This task implements the corresponding next step in the main algorithm shown in the paper. Taking the corresponding elements computed in the previous step, it moves and deforms them to minimize their dissimilarities. It is here that we must ensure that our tool is producing visually smooth matching of geometry. Research into the transformed Laplacian coordinates from Yu et al. 2004 will be needed.

Element Merging: 5 days
This task implements the last step in the main algorithm shown in the paper. It is where the actual stitching takes place. Research into the graph cut algorithm of Boykov et al. 2001 will be needed. The quality of the stitch being made in this task will directly depend on the quality of computations from the previous tasks.

Testing/Debugging for Alpha: 6 days (non-consecutive)
Time has been designated specifically for testing and debugging. This is not only to ensure that our code performs as expected for each milestone, but also have stable code for the next tasks to be developed on as well as serve as a buffer if something unexpected happens.

3.1.2 Beta Version

Shell Mapping: 14 days
Now that our tool is able to quilt a given 3D texture, this task involves mapping that quilt to the surrounding shell of the object. To do that will require the algorithm built for the alpha version be modified to accommodate 3D objects. Research into the discrete conformal mapping in Desbrun et al. 2002 and vector field direction synthesis of Praun et al. 2000 will be needed.

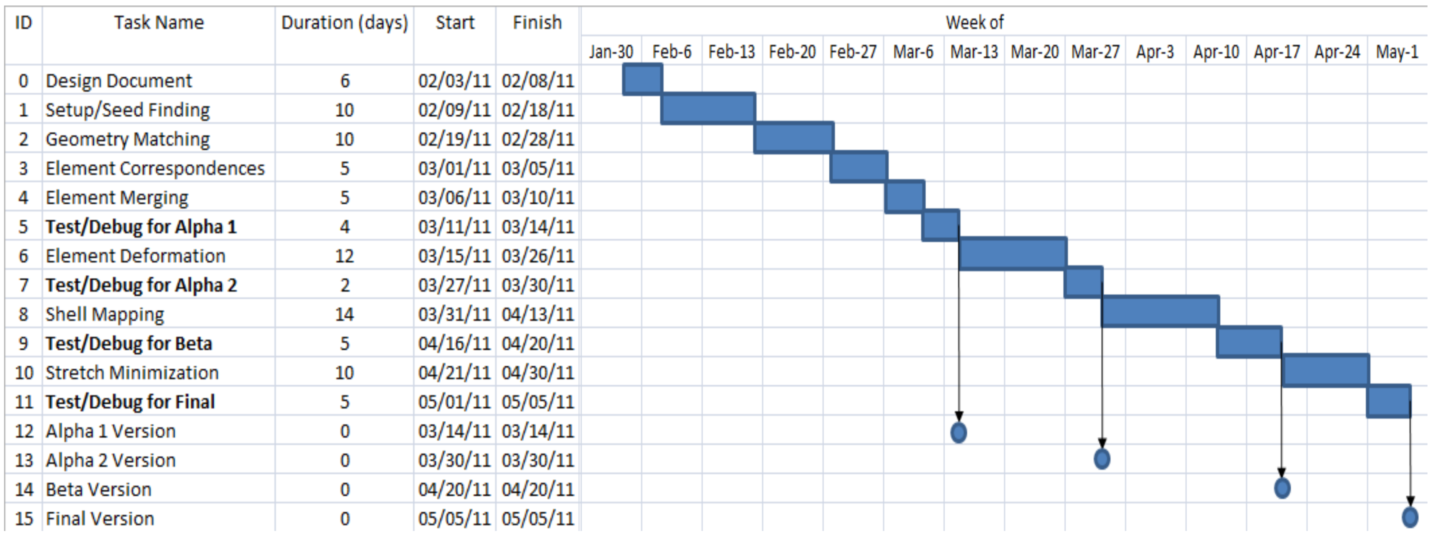
Testing/Debugging for Beta: 5 days
Time has been designated specifically for testing and debugging. This is not only to ensure that our code performs as expected for each milestone, but also have stable code for the next tasks to be developed on as well as serve as a buffer if something unexpected happens.

3.1.3 Final Version

Stretch Minimization: 10 days
This task includes the several sub-steps involved with the stretch minimization section of the 3D quilting process. It is the final phase before a low distortion mesh is output to the scene. Research into the low-distortion parameterization of triangle meshes of Sander et al. 2001 will be needed.

Testing/Debugging for Final: 5 days
Time has been designated specifically for testing and debugging. This is not only to ensure that our code performs as expected for each milestone, but also have stable code for the next tasks to be developed on as well as serve as a buffer if something unexpected happens.

3.2. Schedule



ID	Task Name	Duration (days)	Start	Finish
0	Design Document	6	02/03/11	02/08/11
1	Setup/Seed Finding	10	02/09/11	02/18/11
2	Geometry Matching	10	02/19/11	02/28/11
3	Element Correspondences	5	03/01/11	03/05/11
4	Element Merging	5	03/06/11	03/10/11
5	**Test/Debug for Alpha 1**	4	03/11/11	03/14/11
6	Element Deformation	12	03/15/11	03/26/11
7	**Test/Debug for Alpha 2**	2	03/27/11	03/30/11
8	Shell Mapping	14	03/31/11	04/13/11
9	**Test/Debug for Beta**	5	04/16/11	04/20/11
10	Stretch Minimization	10	04/21/11	04/30/11
11	**Test/Debug for Final**	5	05/01/11	05/05/11
12	Alpha 1 Version	0	03/14/11	03/14/11
13	Alpha 2 Version	0	03/30/11	03/30/11
14	Beta Version	0	04/20/11	04/20/11
15	Final Version	0	05/05/11	05/05/11

4. WORKS CITED

BOYKOV, Y., VEKSLER, O., AND ZABIH, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Trans on Pattern Analysis and Machine Intelligence* 23, 11, 1–18.

DESBRUN, M., MEYER, M., AND ALLIEZ, P. 2002. Intrinsic parameterizations of surface meshes. In *Eurographics*, 209–218.

KWATRA, V., SCHODL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics* 22, 3, 277–286.

PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2000. Lapped textures. In *Proceedings of SIGGRAPH 2000*, 465–470.

SANDER, P., SNYDER, J., GORTLER, S., AND HOPPE, H. 2001. Texture mapping progressive meshes. In *Proceedings of SIGGRAPH 2001*, 409–416.

YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2004. Mesh Editing With Poisson-Based Gradient Field Manipulation. *ACM Transactions on Graphics* 23, 3, 644–651.

ZHOU, K., HUANG, X., WANG, X., TONG, Y., DESBRUN, M., GUO, B., AND SHUM, H. 2006. Mesh quilting for geometric texture synthesis. *ACM Transactions on Graphics* 25, 3, 690-697.