Harmonic Coordinates

INF 585 - Irada Buyatova and Gaël Van der Lee

1. Abstract

In computer animation, mesh deformation using a grid allows for the manipulation of the space containing the mesh, which renders certain operations much easier. To deform the space within the grid when grid points are moved, many different functions have been proposed. Here, we show an implementation of a method called Harmonic coordinates, which allows for smoother and more intuitive mesh point manipulation using a cage.

2. Introduction

We are working with the problem of cage based mesh deformation, a method by which a user can deform many points of a mesh by moving a few points of a cage around it. Looking at methods for mesh deformation, harmonic deformation seemed like a great solution to problems that are usually brought up in this field.

The standard way to do deformation is using barycentric coordinates. Barycentric coordinates have been studied for a long time and were first introduced in 1927 by Mobius [1], they are a way to put points in a coordinate system where a point's coordinates is defined in relation to a simplex. It can be seen as the center of mass of that simplex where every vertex of the reference simplex is weighted individually. This approach has been generalized to polygons by Hormann and Floater [2] and refined by many and was found to be a good way to implement cage based deformations since it allows for a recalculation of the center of mass as the points of the simplex are moved around. It was shown that it would work with any 2D polygon, that the deformations would remain smooth and that they reproduced linear functions. However, this approach still had some flaws that make its use less intuitive for cage based deformations. For example, it is possible for a point to move in the opposite direction from the cage point under certain circumstances. This prompted the search for a better system, namely one which would be a generalization of barycentric coordinates wolving 2 main problems:

1. Non negativity: ensuring that points of the polygon move in the same direction as the cage point
2. Interior locality: ensuring that the coordinates fall off as a function of the distance between cage points and mesh points as measured within the cage.

The solution that we decided to implement was found by Tony DeRose and Mark Meyer [3]. The authors show that such a generalization can

be found as solutions to Laplace's equation with chosen boundary conditions. And since Laplace's equations are often referred to as harmonic functions, they chose to name this coordinate system harmonic coordinates.

3. Theory

As we have mentioned before, harmonic coordinates is a cage based deformation method. As stated by the authors of the paper, for every vertex of the cage introduced by the user we intend to determine a function that would satisfy 7 conditions. Those are: interpolation, smoothness, linear reproduction, affine invariance, generalization of barycentric coordinates, non-negativity and interior locality. 5 of those conditions are satisfied with mean value coordinates [2] , except the last two conditions that are not satisfied.

In order to determine the function that would satisfy all of those conditions, Laplace's equation is introduced:

$$\nabla^2 h_i \ (p) = 0 \quad (1)$$

The Laplace's equation is used for the cells that are determined to be interior cells, assuming that the boundary cells are already computed.

For all the vertices of the cage the following conditions are applied:

$$h_i\left(C_j\right) = \phi_i\left(C_j\right) = \delta_{i,j} \quad (2)$$

Following this equation, the harmonic coordinate is equal to 1 if i = j and 0 in all other cases. After calculation of harmonic coordinates of cage vertices, for all the points that are located on the boundary lines, the interpolation is applied and harmonic coordinates on all the bounds are found.

By following these rules, all the values of harmonic coordinates are calculated. For calculating the harmonic coordinates it is impossible to get it using closed form solution, thus they are obtained using numerical solver. It was then proved in the Section 2 of the paper [4] that the solution to Laplace's equation (harmonic functions) with boundary conditions would satisfy the 7 conditions that were mentioned before. All the equations provided are applicable for the case of 2D and could be potentially extended to 3D.

4. Implementation

4.1 User interface

To stay in the continuation of our class work and use libraries that we are familiar with, we chose to code the project in C++ using the libigl library. We decided to create a 2D implementation because it would be faster and simpler than solving the equations in the 3D case. Our first task was to implement a simple interface for the user to interact with the 2D board. We wrote code so that the user could change the mouse click behavior using the number row from 1 to 6. Here is what we implemented and what number it is mapped to:

1. Place the mesh vertices, edges are created between subsequent vertices, and between the first and last one
2. Place the cage vertices, edges are created between subsequent vertices, and between the first and last one
3. Allows you to drag cage vertices. If one of the cage vertices is selected by mouse click, then the harmonic coordinates related to that vertex are displayed. The

inside and boundary cells are displayed by coloring them in blue and green respectively. The intensity of the color is dependent on the value of harmonic coordinate.

4. Toggles the visibility of mesh and cage edges

5. Runs all the grid calculations required to compute harmonic coordinates.

6. Shows the grid points, with different colors for grid points inside, outside and at the boundary of the cage.

Here you can notice that we mapped a key to run the grid computation. Since this computation needs to run only once before moving the cage coordinates. We preferred to have the user press it once he is done drawing the mesh and cage to make the program run faster. Results of drawing vertices can be seen in Figure 1.

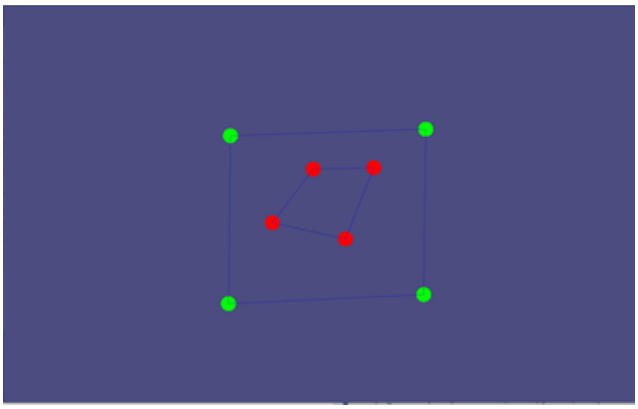

Figure 1: User defined mesh points in red and cage points in green.

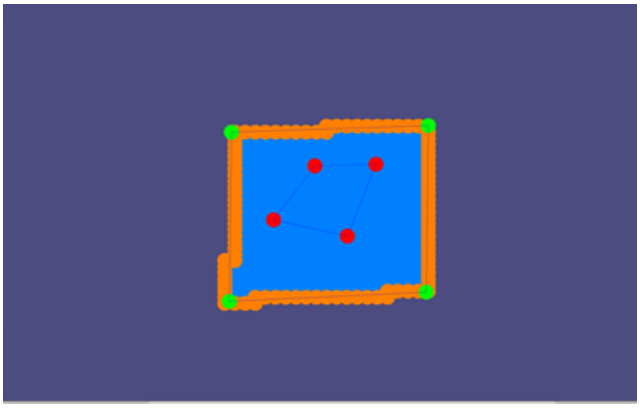

Figure 2: Visualization of grid points, with boundary grid points in orange and inside grid points in blue.

4.2 Grid

Once the user interface was implemented, we had to create a grid as described in the paper [3], which is used for the computation of harmonic coordinates. We chose to create a grid object, which would store all the properties and contain all the functions associated with it. This grid has fixed resolution with each side having 2^S cells. We chose to use the paper's recommended size with $S = 6$. Once the grid is created, we need to compute for each grid cell two properties.

First, we need to categorize each grid cell and determine if it is inside, outside or on the boundary of the cage. For this purpose, we created a member function which takes a point and a mesh and returns a boolean stating whether the point is in the mesh. We used a simple ray casting method, checking if a ray going in the positive y direction crosses edges of the mesh an even or odd number of times. Odd means that the points are inside, even means outside. Then it's just a matter of iterating over all the grid cells, and checking all 4 corner points of each grid cell. If they are all inside so is the grid cell, same with outside, and if there is a mix then the grid cell is on the boundary. You can see the results in Figure 2.

Secondly, we store an associated vector with each grid cell. This vector contains the harmonic coordinates of each of the cells. Thus, the size of this vector is equal to the number of cage vertices that were introduced by the user.

4.3 Computation of harmonic coordinates

After creating the Grid with cells inside of it, we have defined the way to calculate harmonic coordinates for inside and boundary cells.
Firstly, the harmonic coordinates for boundary cells are computed.
We have described in the previous section, that each cell of a grid contains the N harmonic coordinates (where N is the number of cage vertices). As stated in equation 2, the harmonic coordinate of the vertex of the cage should be equal to 1, according to the piecewise linear function.
As such, for each vertex of the cage its position is calculated. After determining the cells which correspond to the position of the i-th vertex, the i-th element of the vector of harmonic coordinates of that cell is equal to 1. The rest N-1 coordinates of that cell is equal to 0.
For finding the harmonic coordinates of the cells that are located on the boundary we first find the number of cells located between 2 following vertices of the cage. In order to do this we have used the information that is stored in the grid cells, and traversed all the cells that are located between the minimum x,y and maximum x,y positions. After this step we end up with the number of boundary cells between two vertices.
Then we started the calculation of harmonic coordinates with the same step of determining the positions of the cells that we found by previous step. Then using the two vertices of the cage and position of the cell, we find the harmonic coordinates of that cell by interpolating the values of the cage vertex coordinates.
After performing the calculations for the boundary cells, we can finally use Laplace's equation to find the values for interior cells. Since this equation can't be solved by using a closed form solution, a numerical solver was used. In order to find the values we have calculated the average of the 4 neighbour cells as this implementation is for 2D. This value for cell is repeatedly calculated and

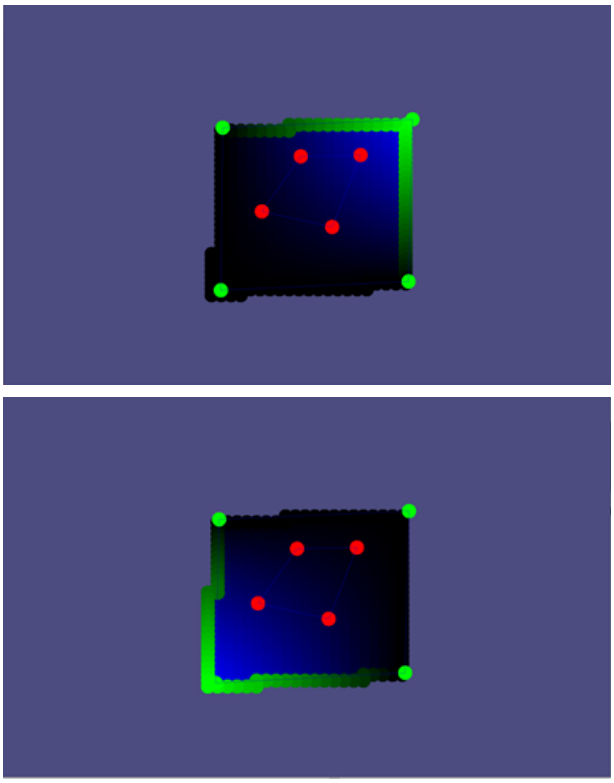

Figure 3: Harmonic values when selecting cage vertices. Green is harmonic values for boundary cells and blue is for inside cells, intensity is proportional to the value

replaces the current value, until the average change to the cell is not less than 10**(-5) - the same value as indicated in the paper. By performing all the steps mentioned above, we have successfully approximated the values of harmonic coordinates (Figure 3). These values are calculated in the beginning, after the user indicates that he has finished picking the cage and mesh vertices. The harmonic coordinates are recalculated again each time the user changes the positions of cage vertices. After calculating the values, now the new positions of the mesh vertices were calculated as following:

$$p' = \sum_i h_i(p) C'_i$$

In order to check the correctness of the harmonic coordinates that we have calculated, at each iteration we also print the sum of all harmonic coordinates (which is approx. 1) for each vertex.

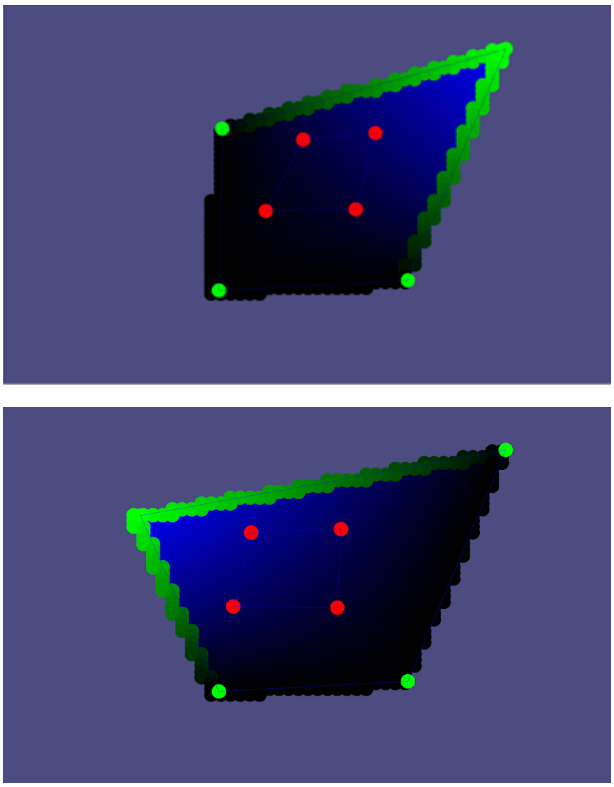

Figure 4: Mesh deformation when moving the cage vertices around.

5. Results

We managed to get satisfactory results for our implementation of harmonic coordinates. The user interface works as expected and we can drag cage points in order to deform the mesh.
The mesh points are moved according to the movement of the cage points proportionally to the harmonic values in an expected manner. You can see the results of dragging the cage in Figure 4. For a grid of size 2^6 with 4 points for the mesh and 4 points for the cage, the program first takes about 1.8 seconds to generate the values of the grid for that specific mesh and cage, then takes around 0.6 seconds to perform the harmonic calculations.

6. Possible improvements

There are many possible improvements and additions we could make to this project. We will try to give them here in order of importance:

- Optimization. The computation and point dragging introduce a small delay before the program becomes responsive again, which should be possible to avoid given modern computational power.
    - A possible avenue for optimization would be parallelization of the computation steps. This would also allow for higher grid resolution.
    - We can also greatly optimize the grid computations since our current approach checks if a certain point is inside the mesh multiple times. Implementing a smarter check could reduce the times for grid computation.
- Allow for different zoom levels. Currently, there is an issue where point coordinates are not calculated properly if the window zoom level is changed. We force the zoom level to a certain value in the code but this solution isn't ideal
- Generalize our program to also work in 3D
- Allow for external mesh import (such as .obj files)

7. Conclusion

The Harmonic Coordinates system is an efficient and robust method for cage based mesh deformation. We managed to implement a simple proof of concept of this method in this project, allowing the user to draw and mesh as well as a grid, and deform the mesh in an intuitive manner by dragging a cage point.

References

[1] MOBIUS ¨ , A. F. 1827. Der barycentrische Calcul: ein neues Hulfsmittel zur analytischen Behandlung der Geometrie ¨ . Barth.

[2]HORMANN, K., AND FLOATER, M. S. 2006. Mean value coordinates for arbitrary planar polygons. ACM Trans. Graph. 25, 4, 1424–1441.

[3] TONY DEROSE, AND MARK MEYER, Harmonic Coordinates, Pixar Technical Memo #06-02

[4] PUSHKAR JOSHI et al., Harmonic Coordinates for Character Articulation