Material based Mesh Deformation

by

DARSHIL S. PATEL
202011034

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
in
INFORMATION AND COMMUNICATION TECHNOLOGY
to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

July, 2022

Declaration

I hereby declare that

i) the thesis comprises of my original work towards the degree of Master of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,

ii) due acknowledgment has been made in the text to all the reference material used.

[illegible]

Darshil Patel

Certificate

This is to certify that the thesis work entitled Material based Mesh Deformation has been carried out by Darshil Sureshbhai Patel for the degree of Master of Technology in Information and Communication Technology at *Dhirubhai Ambani Institute of Information and Communication Technology* under my/our supervision.

[illegible]

Prof. Aditya tatu
Thesis Supervisor

Acknowledgments

A successful accomplishment of any work cannot be done by one single person. Similarly, my M.Tech Thesis work has reached the finish line due to multiple contributions.

I am privileged to express my sincere gratitude to Prof. Aditya Tatu for supervising me in this entire journey by showing accurate directions in solving every problem. Without the brainstorming sessions and fruitful discussions, the process of devising methods would not have been smoother. Additionally, I heartily thank Dr. Sumukh Bansal for providing technical resources and supplementary guidance.

It was a great experience sharing the ideas with my dear friends Dhyanil Mehta, Shivangi Gajjar, Kishan Vaishnani, and Tarang Ranpara. I am grateful to them for all the absorbing discussions and help they provided in the process of thesis work as a whole. Technical guidance is necessary for work but is not possible without a calm mind. I extend my gratitude to my parents and family for constantly believing in me and pushing me in the right direction when needed.

Lastly, I thank the support staff of DA-IICT for providing the laboratory space and required facilities.

Contents

Abstract

Shape deformation is one of the fundamental techniques in geometry processing. Shape deformation algorithms aim to mimic 3D object deformations on digital representations of objects efficiently. Deformation refers to the change in the shape and size of the object. Rigidity performs a significant role in the deformation. The actual 3D object could have parts with different amounts of rigidity. The shape deformation algorithm should be able to model the rigidity of the entire object and spatially varying rigidity. We propose a rigidity controllable mesh deformation method where the user can specify the rigidity of each part of the object. There are several popular deformation models and algorithms to deform objects. However, none of them explicitly capture the rigidity of the object. Additionally, we try to estimate the rigidity of the object given various deformations of the object.

List of Tables

List of Figures

CHAPTER1

Introduction

The thesis topic falls under the area of geometry processing. The scope of geometry processing is vast; it ranges from digitally representing the object and matching the shape to changing shapes or applying deformation. Deformation refers to modifications of the shape or size of an object. Deformation is a fundamental research area in geometry processing, and it is used in animation, mechanical modeling, design, augmented reality, and simulation. Due to time and resource constraints, it is not feasible to apply deformation manually as it involves changing the positions of every point of the object. Applying transformation only to the handle points overcomes the limitation of manual deformation. Every point of the object would be linked with some handle points so one can find the transformation or position of the deformed position of the vertex. In this way, the movement of those handle points would define the movement for the respective linked points.

Some well-known algorithms for applying various deformations using handles on a given mesh are skinning[7, 9], Deformation using bone and cage[18], Laplacian mesh edition[16], As-Rigid-As-Possible[15], Rigidity controllable As-Rigid-As-Possible[3] etc. Some algorithms learn deformation space from the example meshes and find the appropriate deformation from the learned space, for example, [5, 17]. The mentioned algorithms take only shape information, not material information. Therefore, resultant deformations would not be as similar as they should be if they were provided with the material information. Every material responds differently to deformations like twisting, bending, compression, and expansion. For example, bending deformation applied on a rod made up of *flexible* material like rubber and *rigid* material like iron would be significantly different.

Additionally, deformation applied on objects with spatially varying rigidity would be different. Building on this fact, deformation applied to an object made up of various materials with different rigidity would be a combination of the

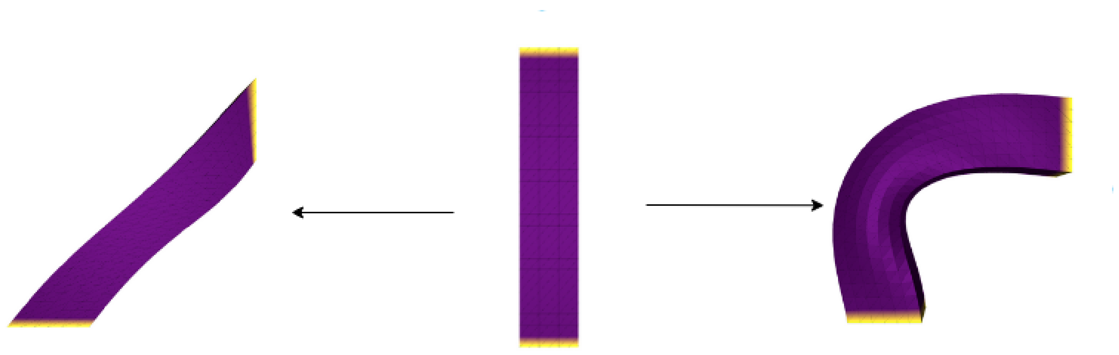


Figure 1.1: Same shape and same movement of handles but different rigidity leads to different deformed mesh

neighborhood-level and object-level properties. Because of the reasons stated above, it makes the study of material-aware deformations relevant.

1.1 Problem statement

The deformation should depend on the rigidity of the object. Objects with the same shape but different materials are likely to undergo different deformation even if they are subject to the same force. For example, in Figure 1.1, The bar in the center is the reference bar and handles and free vertices are represented in yellow and purple color respectively. Applying the transformation on The handles (kipping the bottom portion as it is and 90° rotation with respect to the center of the bar on the top portion), how other vertices should move is dependent on the material. If the material is more rigid, we will use the deformed mesh that is on the right, and if the material is less rigid, we will use the deformed mesh that is on the left in Figure 1.1. The digital representation of the object as a mesh does not contain information about the material it is made up of. This work tries to capture the material's rigidity via a rigidity controlling cost function. As shown in Figure 1.1, our deformation algorithm gives the user control over the rigidity of the object. Additionally, we try to estimate the rigidity of the object given various deformations of the object, as shown in Figure 1.2.

1.2 Thesis Organization

Background topics like object representation, transformations, neighborhood, and cotangent weights in Chapter 2. We provide a survey of the most related work in

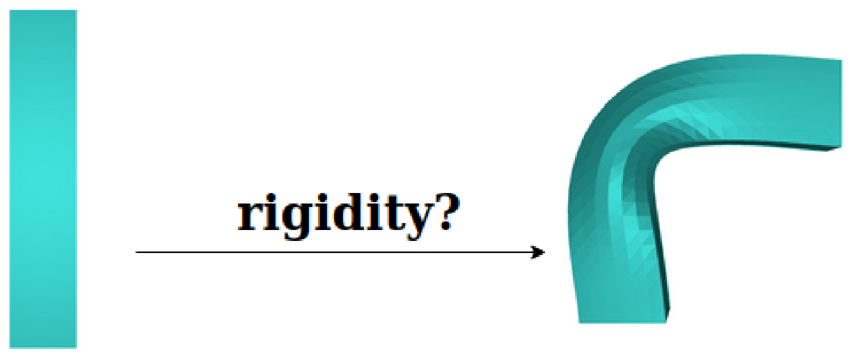


Figure 1.2: How to estimate the rigidity from the reference mesh and its deformed mesh or two deformed mesh of the same object?

Chapter 3. Our proposed approach rigidity controllable deformation and learning material from reference meshes are described in Chapter 4. Results and discussions are presented in Chapter 5. Finally, the conclusion and future work is given in Chapter 6.

1.3 Contribution

The contributions of this work are summarized as follows:

- We propose an algorithm that takes material information along with shape information and performs the deformation.
- The proposed algorithm also works where different parts of the object are made of different materials. The proposed algorithm gives functionality to change the neighborhood size as mentioned in RC-ARAP[3] and the material of that neighborhood to get more realistic deformation.
- We also propose an algorithm that takes in multiple meshes of the same object and estimates the material the object is made of.

CHAPTER2
Background

Deformation refers to the change in the size or shape of an object. Since our problem domain is mesh deformation, some concepts and terminologies are necessary to understand various deformation algorithms and proposed work.

2.1 Object Representation

There are many ways to represent the 3D object digitally, including Mesh representation, Implicit representation, point cloud, and skeletons. The most prevalent representation is mesh. Mesh is a graph-like data structure that contains information about the position of the vertices and connectivity between them, where vertices represent points on the object. A mesh represents a larger geometric domain by small discrete cells. Examples of some discrete cells are shown in Figure 2.1. The object is continuous, so to represent the object in digital form, one needs to convert it into discrete form, taking the samples or points from the object. Mesh is a data structure to store these sampled points and their connectivity. Mesh can be classified into two types based on the discrete cell or connectivity.

1. Surface mesh: Discrete cells are 2D shapes like triangles or quadrilateral in the surface mesh. The surface mesh contains only the samples from the surface of the object.

2. volumetric mesh: Discrete cells are 3D shapes like tetrahedrons, pyramids, or hexahedrons in the volumetric mesh. The volumetric mesh contains the samples from the surface and interior part of the object.

Among all the meshes, triangular meshes have become the ubiquitous surface representation in recent years in computer graphics, so a lot of research has been put into efficiently manipulating them. Triangular mesh has two components:

- Set of Vertices, which can be represented as a $n \times 3$ matrix V, each row of the Vertices matrix V contains 3D coordinates of the vertex.

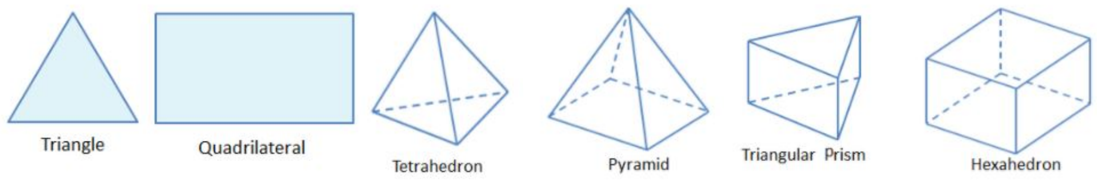


Figure 2.1: Discrete cells[1]

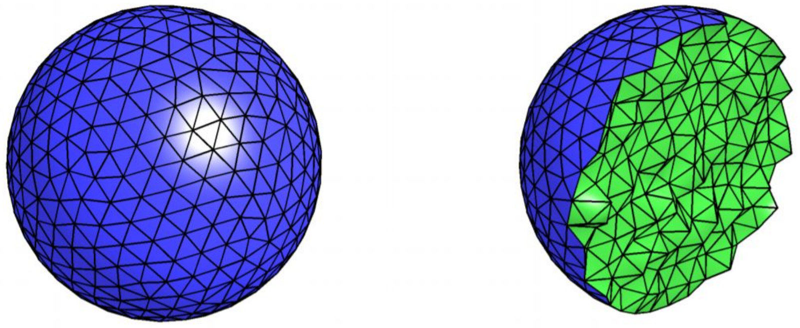


Figure 2.2: surface mesh and volumetric mesh[11]

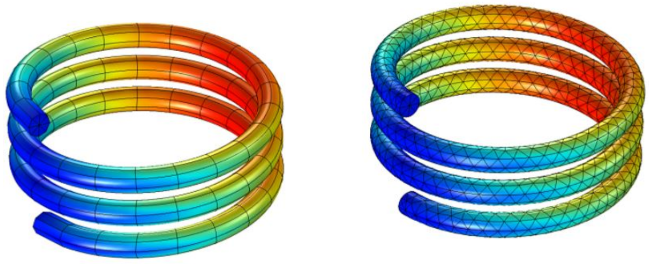


Figure 2.3: surface meshes[4]

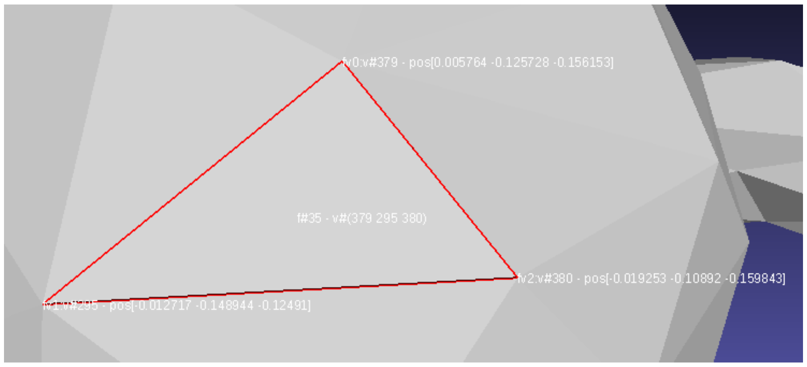

fv0:v#379 - pos[0.005764 -0.125728 -0.156153]
f#35 - v#(379 295 380)
fv2:v#380 - pos[-0.019253 -0.10892 -0.159843]
fv1:v#295 - pos[-0.012717 -0.148944 -0.12491]

Figure 2.4: Vertices and Faces

- Set of faces that provides connectivity/topological information, which can be represented as a $m \times 3$ matrix F, Each row of F contains indices of 3 vertices from V that form a triangular face.

.

This thesis will assume that all objects are represented as triangular meshes.

2.2 Transformations

Mesh has some properties like the position of vertices, distance between vertices, the angle between edges, and the orientation of faces. Transformation changes these properties, and, based on these changes, transformations can be classified into two types.

1. Rigid transformations
2. Non-rigid transformations

2.2.1 Rigid Transformations

Transformations that do not change the distance between two points are called rigid transformations. Rigid transformations do not change the size and shape of the object. Translation, Rotation, and their combinations are rigid transformations.

In $\mathbb{R}^3$ translation by a vector $r \in \mathbb{R}^3$ can be defined as,

$$f : \mathbb{R}^3 \to \mathbb{R}^3, f(x) = x + r, \forall x \in \mathbb{R}^3$$

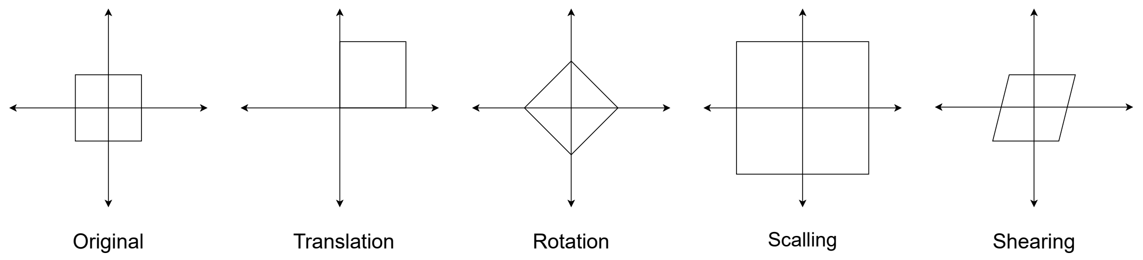


Figure 2.5: Transformations

and a 3D rotation can be represented as,

$$f : \mathbb{R}^3 \to \mathbb{R}^3, f(x) = R(n, \theta)x, \forall x \in \mathbb{R}^3$$

where the 3×3 matrix $R(n, \theta)$ is an orthogonal matrix that represents rotation by an angle θ with unit vector n as axis of rotation.

The translation is a non-linear transformation, so it can not be represented using a 3×3 matrix, but using a homogeneous coordinate system, translation can be represented using a 4×4 matrix. Reflection may also be considered a rigid transformation with determinant -1. It changes the orientation of the axes.

2.2.2 Non-Rigid Transformations

The transformations that are not rigid are called non-rigid transformations. Scaling and shearing are examples of non-rigid transformations. A non-rigid transformation can change the object's size, shape, or both.

In $\mathbb{R}^3$, scaling can be defined as,

$$f : \mathbb{R}^3 \to \mathbb{R}^3, f(x) = Sx, \forall x \in \mathbb{R}^3$$

where S is a 3 x 3 diagonal matrix, whose diagonal elements represents the scaling factor along the three axis.

In $\mathbb{R}^3$, shearing can be define as,

$$f : \mathbb{R}^3 \to \mathbb{R}^3, f(x) = Tx, \forall x \in \mathbb{R}^3$$

where 3×3 matrix T has diagonal elements as 1 and off-diagonal elements represent the amount of shear and direction along with different axis.

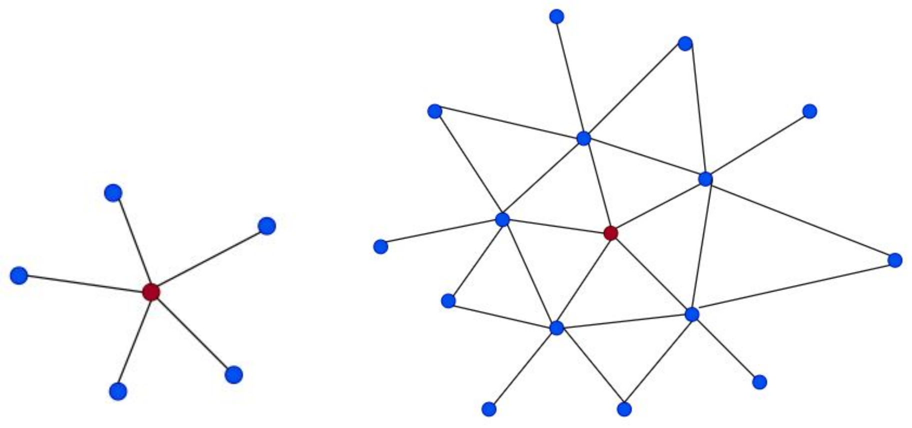

(a) 1-ring neighbour (b) 2-ring neighbour

Figure 2.6: the r-ring neighborhood of the vertex

2.3 Neighborhood

As shown in Figure 2.6, the set of 1-ring neighbors of a vertex contains other vertices that are directly connected to the vertex with an edge. In comparison, the set of 2-ring neighbors contains vertices that can be reached using two steps from the vertex. The *r*-ring neighborhood can be represented using two sets.

1. Vertex set $\mathcal{N}(k, r)$ is the set of vertices within the *r*-ring neighborhood of vertex k, where a vertex $j \in \mathcal{N}(k, r)$ if and only if there exists a path connecting vertex k and j with the number of edges no more than r.

2. Edge set $\mathcal{E}(k, r)$ is the set of edges within the *r*-ring neighbourhood of vertex k, where edge $(i, j) \in \mathcal{E}(k, r)$ if $i \in \mathcal{N}(k, r)$ and $j \in \mathcal{N}(k, r)$.

The vertex and edge set can be obtained efficiently using a breadth-first search.

2.4 Cotangent weight

The Laplace-Beltrami operator (LBO) is the swiss-knife of Geometry Processing. The discrete Laplace-Beltrami operator[2] works for the discrete surface, for example, triangular mesh. One should arrive at a definition of a discrete LBO by requiring it to satisfy all the properties satisfied by the continuous LBO. Cotan laplacian is one of the doscrete laplace beltrami operators. Cotan weights are part of the Cotan laplacian. Cotan weights reduce the effect of discretization of the triangular mesh.

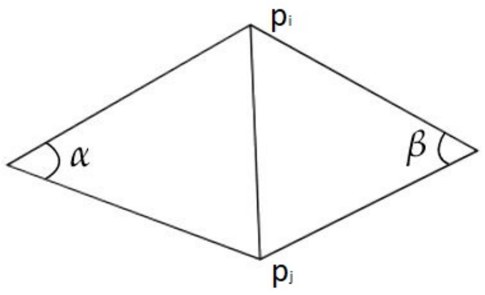


Figure 2.7: Cotan Weight

Consider $\mathbf{p}_i$ and $\mathbf{p}_j$ are the vertices of a mesh, and there is an edge that connects these two vertices, then the cotan weight of this edge is,

$$w_{ij} = \frac{1}{2}\,(cot\ \alpha + cot\ \beta)$$

where α and β are two angles opposite to the edge (i, j) as shown in the Figure 2.7.

This background knowledge will help to understand the popular deformation algorithms in Chapter 3 and the proposed method.

CHAPTER3

Related work

Shape deformation has received much attention over the last three decades. This chapter reviews important milestones in this area and discusses algorithms or approaches close to ours.

Skinning

Skinning [7] is the earliest algorithm proposed for mesh deformation. Skinning has three components.

- Control Structure
- Weights
- Transformations

A low dimensional structure, typically known as the control structure, is used to obtain user define deformation, which is then used to obtain the deformation of the entire mesh. Some examples of the control structures are cage, skeleton, and set of vertices as shown in figure 3.1. Points or parts of the control structure where the user provides deformation via transformation are called handles. From these transformations, transformations for each vertex of mesh is obtained by a weighted combination of transformation defined on handles.

Suppose a mesh contains n vertices. Among n vertices, m vertices are the handles. Linear Blend Skinning(LBS) uses a linear combination of handle transformations to deform mesh vertices:

$$\mathbf{p}'_i = \sum_{j=1}^{m} w_{ij}\mathbf{T}_j\mathbf{p}_i$$

where $\mathbf{p}_i$ and $\mathbf{p}'_i$ is i^{th} vertex of reference mesh and deformed mesh respectively. $\mathbf{T}_1$ to $\mathbf{T}_m$ are the transformations applied on the handle vertices. w_{ij} is weight assigned to pair of i^{th} vertex and j^{th} handle.

Weights are critical in skinning because it determines how much a vertex is affected by the transformation on handles. There are some methods to define weights, for example, rigid weights, inverse distance weights, Bounded Biharmonic Weights [6]. Different weigh assign methods result in different deformations. Some methods are described below.

In rigid weights, for the closest handle, the weight will be 1, and for others handles, it will be 0. Thus any vertex will be affected by only its closest handle. i.e.,

$$w_{ij} = \begin{cases} 1 & \text{if } d(\mathbf{p}_i, H_j) \leq d(\mathbf{p}_i, H_k) \forall j \neq k \\ 0 & \text{otherwise} \end{cases}$$

where, $d(\mathbf{p}_i, H_j)$ is the euclidean distance or geodesic distance between vertex $\mathbf{p}_i$ and handle H_j. Another method is inverse distance weights, in which weight can be defined as,

$$w_{ij} = \frac{1}{d\left(\mathbf{p}_i, H_j\right)^p}$$

where $p > 0$ and it is hyper-parameter.

Figure 3.2 shows the different types of weights. Figure 3.2 (a) is a bar mesh with two handle points that are represented in red color. Figure 3.2 (b) represents the weight assigned(yellow color for 1 and purple color for 0) to all vertices for respective handles points(left figure for left handle point and right figure for right handle point). Similarly, inverse distance is shown in Figure 3.2 (c), where the yellow color represents a higher value and the purple color represents a lower value.

While assigning weights, we need to consider its properties like,

- weights should be continuous and non-negative
- weights should be local and sparse
- weight should be shaped aware
- sum of all weights at a vertex must be equal to 1. This property is known as the partition of unity.

Some updated skinning algorithm are Dual Quaternion Blending technique [9], Spherical blend skinning [10], Example-based dynamic skinning [13].

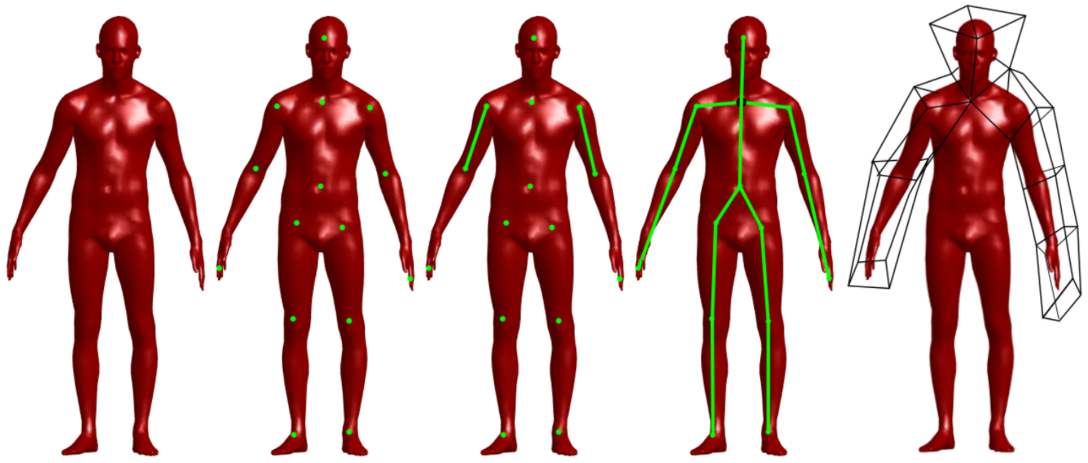
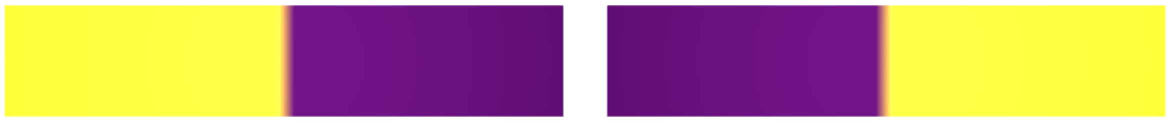
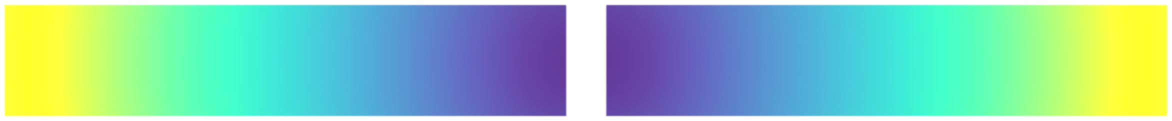


Figure 3.1: Examples of control structures: point handles, bones, skeleton [18], and cage

(a) bar example with handle



(b) Rigid weights (yellow for 1 and purple for 0)



(c) Inverse distance weights

Figure 3.2: Vertices are colored with the respective weights where yellow color represents high value and purple color represents low value

Laplacian Mesh Editing

Sometimes it is challenging to operate on a mesh that uses absolute coordinates(3D coordinate system) to store the geometric properties. Because while applying the transformation to each point, it is challenging to preserve topology and overall shape.

Laplacian mesh representation maintains track of differential vertex information rather than absolute information while storing the geometry of triangle meshes. When certain modifications (particularly deformations) are performed on the mesh, this proves to be a far better approach for maintaining the relationship between vertices. It also enables smooth function interpolation across the surface using a natural technique. And the quick approximation of several differential geometry metrics like means curvature.

A Laplacian coordinate for vertex i is defined as,

$$\delta_i = \mathbf{p}_i - \frac{\sum_{j \in \mathcal{N}(i)} w_{ij} \mathbf{p}_j}{\sum_{j \in \mathcal{N}(i)} w_{ij}} \tag{3.1}$$

where w_{ij} represents the weight related to edge between i^{th} and j^{th} vertex. Weights can be cotan weights. $\mathcal{N}(i)$ is the set of all neighbor vertices of the i^{th} vertex.

A Laplacian coordinate is the linear combination of the vertices, so we can use matrix multiplication to find the Laplacian coordinate.

$$\delta = \mathbf{L}V$$

where, V and δ are $n \times 3$ matrices contains absolute coordinates and Laplacian coordinate respectively. $\mathbf{L}$ is $n \times n$ Laplacian matrix whose entries are followed:

$$\mathbf{L}_{ij} = \begin{cases} 1 & \text{if } i = j \\ -\frac{w_{ij}}{\sum_{k \in \mathcal{N}(i)} w_{ik}} & \text{if } j \in \mathcal{N}(i) \\ 0 & \text{otherwise} \end{cases}$$

The energy function for the laplacian mesh editing is,

$$E\left(V', T_i\right) = \sum_{i=1}^{n} \left\| T_i \delta_i - \left(LV'\right)_i \right\|^2 + \sum_{i=m}^{n} \left\| \mathbf{p}_i' - \mathbf{u}_i \right\|^2 \tag{3.2}$$

where V' is the matrix containing all vertex coordinates of the deformed mesh, with $\mathbf{p}_i'$ denoting the new coordinates of the i^{th} vertex, L is the district Laplace

Beltrami operator (laplacian matrix) [2], δ_i is laplacian coordinate corresponding to the i^{th} vertex, $\mathbf{u}_i$ is the position of the handles and T_i is transformation that applied on the 1-ring neighbourhood of i^{th} vertex.

As-Rigid-As-Possible

The paper[15] describes a mesh deformation technique called ARAP(As-Rigid-As-Possible) that allows you to change the shape of a mesh while still preserving its detail. ARAP fits a rotation transformation between the 1-ring neighborhood of reference and the same neighborhood of deformed mesh. In this way, ARAP tries to apply global non-rigid transformation using local rigid transformation. In ARAP, the authors assume that the 1-ring neighborhood of mesh is rigid, and during any non-rigid deformation also, it should behave as rigidly as possible. For each vertex, the neighborhood overlaps each other, so multiple transformations are applied on the particular edge. The energy function is defined as,

$$E\left(\mathcal{S}, \mathcal{S}'\right) = \sum_{i=1}^{n} \left\{ \sum_{j \in \mathcal{N}(i)} w_{ij} \left\| \left(\mathbf{p}'_i - \mathbf{p}'_j\right) - \mathbf{R}_i \left(\mathbf{p}_i - \mathbf{p}_j\right) \right\|_2^2 \right\} \tag{3.3}$$

where $\mathcal{S}$ is the reference mesh, $\mathcal{S}'$ is the deformed mesh, n is the number of vertices, $\mathbf{p}_i$ is i^{th} vertex of the reference mesh, $\mathbf{p}'_i$ is i^{th} vertex of the deformed mesh, $\mathbf{R}_i$ is rotation defined on i^{th} vertex, w_{ij} is cotan weight of edge between $\mathbf{p}_i$ and $\mathbf{p}_j$ and $\mathcal{N}(i)$ is set of 1-ring neighbours of i^{th} vertex.

The Equation 3.3 is popularly known as ARAP energy. For this energy function, $\mathbf{R}$ and $\mathbf{p}'$ are unknown. The author proposes using a simple alternating minimization strategy to find the next local minimum energy state (starting from a given initial vector of positions and rotations). i.e., find positions $\mathbf{p}'$ that minimize energy function 3.3 for a fixed set of rigid transformations $\mathbf{R}$. Then, for the given set of positions $\mathbf{p}'$, find the rigid transformations $\mathbf{R}$ that minimize energy function 3.3. Repeat these interleaved iterations until the local energy minimum is reached. This kind of strategy is called Alternating Minimization, and for more details, refer Section 12.3 of [14]. The important features of this approach are robustness, simplicity, and efficiency.

Rigidity Controllable As Rigid As Possible

The author extends the ARAP energy function 3.3 in Rigidity controllable ARAP [3] energy. The goal is to control the rigidity by varying the size of the local neighborhood. The larger the size, the more local geometric details will be preserved,

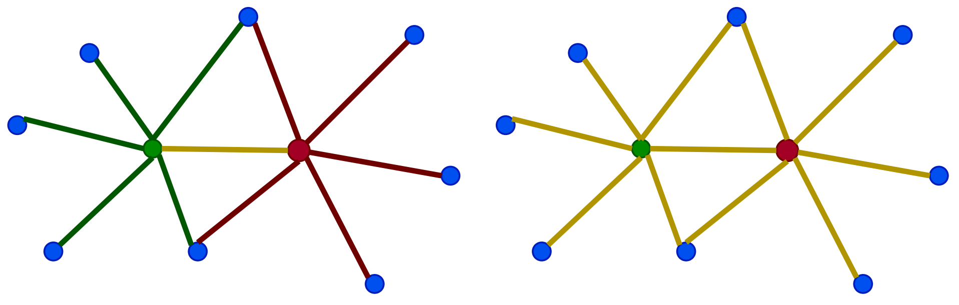


Figure 3.3: overlapping area represented by yellow color in 1-ring(a) and in 2-ring (b) neighborhood

or the material will appear more rigid. This is because expanding the local neighborhoods increases the size of overlapping areas between adjacent vertices, improving the coherence of the rigid local transformation. For example, as shown in the figure 3.3 green and red colors represent the 1-ring neighborhood of respective vertices. The yellow color represents an overlapping region. We can conclude from figure 3.3 that as r increases, the overlapping region will increase. Another benefit of improved rigid transformation consistency is that it reduces the variability of local rigid transformations, allowing the optimization to converge faster.

$$E\left(\mathcal{S},\mathcal{S}'\right)=\sum_{k=1}^{n}\left\{\sum_{(i,j)\in\mathcal{E}(k,r)} w_{ij}\left\|\left(\mathbf{p}'_i-\mathbf{p}'_j\right)-\mathbf{R}_k\left(\mathbf{p}_i-\mathbf{p}_j\right)\right\|_2^2\right\} \tag{3.4}$$

This energy function is similar to the ARAP energy function. Here, the only difference is that the r-ring neighborhood is considered instead of the 1-ring neighborhood. Rigidity Controllable ARAP allows such objects to be simulated within a unified framework using the r-ring ARAP formulation with different r values specified for different regions.

The limitation of the ARAP and RC-ARAP is that they try to fit rotation transformation between the local neighborhood of reference mesh and deformed mesh because they assume that the local neighborhood is rigid.

CHAPTER4
Proposed Approach

4.1 Deformation

We assume that an input model has a set of vertices as handles, similar to traditional ARAP[15] deformation. Given a mesh $\mathcal{S}$, and user defined deformation on handles (subset of $\mathcal{S}$), the algorithm generates a deformed mesh $\mathcal{S}'$ that satisfies the handle requirements while maintaining geometric details. In this Chapter $\mathcal{C}_k$ and $\mathcal{C}_k{}'$ denotes the neighbourhood of k^{th} vertex of the $\mathcal{S}$ and $\mathcal{S}'$ respectively. $\mathcal{S}$ and $\mathcal{S}'$ contains same number of vertices n and same connectivity that is represented using m triangular faces. $\mathcal{N}(k,r)$ is the set of vertices within the r-ring neighbourhood of vertex k, $\mathcal{E}(k,r)$ is the set of edges within the r-ring neighbourhood of vertex k, $\mathbf{p}_i$ and $\mathbf{p}_i'$ denotes the position of i^{th} vertex in $\mathcal{S}$ and $\mathcal{S}'$ respectively. w_{ij} is cotan weight of edge (i,j). $\mathbf{T_k}$ is the transformation applied on the r-ring neighbourhood of k^{th} vertex of $\mathcal{S}$. The following energy function should be minimized for each vertex k to maintain the geometric details.

$$E\left(\mathcal{C}_k, \mathcal{C}_k{}'\right) = \sum_{(i,j)\in\mathcal{E}(k,r)} w_{ij} \left\| \left(\mathbf{p}_i' - \mathbf{p}_j'\right) - \mathbf{T_k}\left(\mathbf{p}_i - \mathbf{p}_j\right) \right\|_2^2 + \lambda \|\mathbf{T_k}^{\mathbf{T}}\mathbf{T_k} - \mathbf{I}\|_F^2 \quad (4.1)$$

The above equation is similar to the RC-ARAP[3] energy, but here $\mathbf{T_k}$ can be a non-rigid transformation. The rigidity controlling term in the energy function 4.1 is defined as $\|\mathbf{T_k}^{\mathbf{T}}\mathbf{T_k} - \mathbf{I}\|_F^2$,where $\mathbf{I}$ is 3×3 identity matrix. For a high value of λ, Equation 4.1 tries to minimize this term. Hence, $\mathbf{T_k}$ will try to converge to a nearly orthogonal matrix containing the maximum effect of rotation. Similarly, a low value of λ tries to minimize the first term in the Equation 4.1. Therefore, $\mathbf{T_k}$ may behave as a shear transformation.

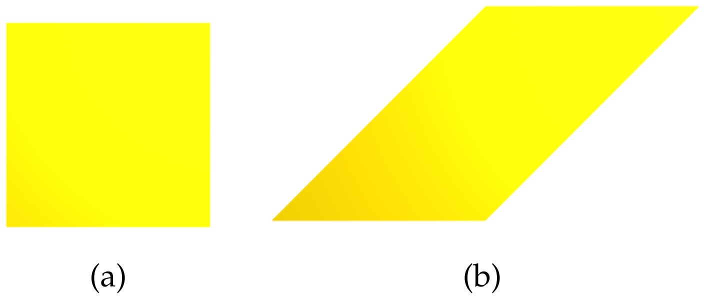


Figure 4.1: (a) Reference mesh, (b) Deformed mesh

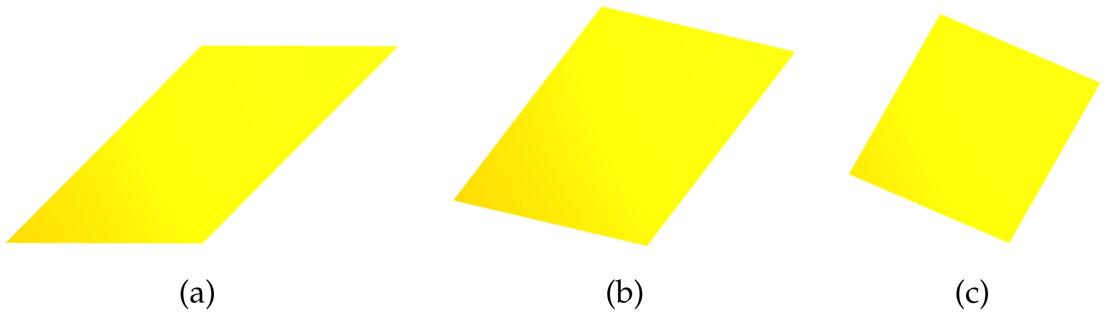


Figure 4.2: (a) $\lambda = 0.01$, (b) $\lambda = 1.5$, (c) $\lambda = 10$

For example, one square mesh Figure 4.1 (a) containing nine vertices. The center of the square is at the origin and it is fixed. Another mesh (b) in Figure 4.1 is a Transformed version of (a). For the deformation, shear transformation is applied on the center vertex. The idea is to get the closest deformed mesh to the Figure 4.1 (b) using the deformation algorithm. The proposed algorithm finds the best transformations for each vertex to minimize the energy function 4.1. Figure 4.2 shows the deformed meshes obtained after applying the transformations, which are generated using the proposed approach of substituting different values of λ on the given mesh. In Figure 4.2 we can see that if λ is small, mesh behaves as less rigid and gives the very close result to the Figure 4.1 (b), but as λ increases, the mesh behaves as more rigid. Therefore we can see some rotation effects to find the closest deformation to the Figure 4.1 (b).

4.1.1 Optimization

We iteratively optimize the local transformation matrix $\mathbf{T_k}$ and the deformed position $\mathbf{p}$ for each vertex given the input model and the handle positions after deformation. Two alternating steps are applied in each iteration to make this tractable. We optimize the transformation in the first step with the vertex positions fixed, and then we optimize the vertex positions in the second step given the transformation. In experiments, we study that considering relative energy as stoping criteria will converge after 5-6 iterations. Because after some iterations, the change in the energy becomes significantly less, but the function is non-convex, so there may be more convincing deformation for the same energy. So we keep the number of iterations as a stopping condition for this iterative algorithm.

Local Step:

Given the optimized vertex positions $\mathbf{p}'$ the transformation $\mathbf{T_k}$ for each vertex k is computed as follows. Let us denote the edge $\mathbf{p}_i - \mathbf{p}_j$ as $\mathbf{e}_{ij}$ and corresponding edge in the deformed mesh by $\mathbf{e}'_{ij}$. Then for single vertex we can rewrite Equation 4.1 as,

$$E(\mathbf{T_k}) = \sum_{(i,j)\in\mathcal{E}(k,r)} w_{ij} \left(\mathbf{e}'_{ij} - \mathbf{T_k}\mathbf{e}_{ij}\right)^T \left(\mathbf{e}'_{ij} - \mathbf{T_k}\mathbf{e}_{ij}\right) + \lambda\, tr\left(\left(\mathbf{T_k}^{\mathbf{T}}\mathbf{T_k} - \mathbf{I}\right)^T \left(\mathbf{T_k}^{\mathbf{T}}\mathbf{T_k} - \mathbf{I}\right)\right)$$

where $tr(X)$ denotes trace of matrix X. To get the optimal transformation we minimize $E(\mathbf{T_k})$. After simplification the optimization function can be rewritten as,

$$\underset{\mathbf{T_k}}{\text{argmin}} \quad tr\left(\lambda\mathbf{T_k}^{\mathbf{T}}\mathbf{T_k}\mathbf{T_k}^{\mathbf{T}}\mathbf{T_k} - 2\lambda\mathbf{T_k}^{\mathbf{T}}\mathbf{T_k} + \mathbf{T_k}^{\mathbf{T}}\mathbf{T_k}\mathbf{B_k} - 2\mathbf{T_k}\mathbf{C_k}\right) \tag{4.2}$$

In the optimization problem 4.2, $\mathbf{B_k}$ and $\mathbf{C_k}$ are 3×3 matrices defined as,

$$\mathbf{B_k} = \sum_{(i,j)\in\mathcal{E}(k,r)} w_{ij}e_{ij}{e_{ij}}^T$$

$$\mathbf{C_k} = \sum_{(i,j)\in\mathcal{E}(k,r)} w_{ij}e_{ij}{e'_{ij}}^T$$

Using the gradient descent method, the optimization problem 4.2 can be solved, and we will get local minima $\mathbf{T_k}$ that minimizes the energy function 4.1. For the optimization and simplification, we refer to [12].

for the optimization problem 4.2 gradient can be calculated as,

$$\frac{E(\mathbf{T_k})}{d\mathbf{T}_k} = 4\lambda\mathbf{T_k}\mathbf{T_k}^{\mathbf{T}}\mathbf{T_k} - 4\lambda\mathbf{T_k} + 2\mathbf{T_k}\mathbf{B_k} - 2\mathbf{C_k^T} \tag{4.3}$$

Global step:

Simple approach for measuring the energy of a deformation of the mesh is to sum of the the energy of all vertex, as expressed by Equation 4.1. Thus, we obtain the following energy function:

$$E\left(\mathcal{S},\mathcal{S}'\right) = \sum_{k=1}^{n} \left\{ \sum_{(i,j)\in\mathcal{E}(k,r)} w_{ij} \left\| \left(\mathbf{p}_i' - \mathbf{p}_j'\right) - \mathbf{T_k}\left(\mathbf{p}_i - \mathbf{p}_j\right) \right\|_2^2 + \lambda \|\mathbf{T_k}^{\mathbf{T}}\mathbf{T_k} - \mathbf{I}\|_F^2 \right\} \tag{4.4}$$

Given the optimized transformation $\mathbf{T_k}$, the proposed energy becomes a quadratic function w.r.t. the deformed positions $\mathbf{p}_i'$. In Equation 4.4 the second term is not dependent on $\mathbf{p}_i'$. So, the optimal position for $\mathbf{p}_i'$ can thus be obtained by solving the linear system $\frac{\partial E(\mathcal{S},\mathcal{S}')}{\partial \mathbf{p}_i'} = 0$ as shown in the RC-ARAP [3], and reproduced bellow for the sake of completeness.

$$\begin{aligned}\frac{\partial E\left(\mathcal{S},\mathcal{S}'\right)}{\partial \mathbf{p}_i'} = \sum_{j\in\mathcal{N}(i,1)} \Bigg(& \sum_{k:(i,j)\in\mathcal{E}(k,r)} 2w_{ij}\left(\left(\mathbf{p}_i' - \mathbf{p}_j'\right) - \mathbf{T_k}\left(\mathbf{p}_i - \mathbf{p}_j\right)\right) \\ & + \sum_{s:(j,i)\in\mathcal{E}(s,r)} -2w_{ji}\left(\left(\mathbf{p}_j' - \mathbf{p}_i'\right) - \mathbf{T_s}\left(\mathbf{p}_j - \mathbf{p}_i\right)\right)\Bigg)\end{aligned}$$

where $\{k|(i,j)\in\mathcal{E}(k,r)\}$ is the vertex set containing all the vertices whose r-ring neighborhood covers the edge (i,j). Since $w_{ij} = w_{ji}$, $\frac{\partial E(\mathcal{S},\mathcal{S}')}{\partial \mathbf{p}_i'}$ can be rewritten as,

$$\sum_{j\in\mathcal{N}(i,1))} 2w_{ij}\left(d_{ij}\left(\mathbf{p}_i' - \mathbf{p}_j'\right) - \sum_{k:(i,j)\in\mathcal{E}(k,r)} \mathbf{T_k}\left(\mathbf{p}_i - \mathbf{p}_j\right)\right)$$

where d_{ij} is the number of elements in $\{k|(i,j)\in\mathcal{E}(k,r)\}$. The linear system $\frac{\partial E(\mathcal{S},\mathcal{S}')}{\partial \mathbf{p}_i'} = 0$ gives,

$$\sum_{j\in\mathcal{N}(i,1)} d_{ij}w_{ij}\left(\mathbf{p}_i' - \mathbf{p}_j'\right) = \sum_{j\in\mathcal{N}(i,1)} w_{ij} \sum_{k:(i,j)\in\mathcal{E}(k,r)} \mathbf{T_k}\left(\mathbf{p}_i - \mathbf{p}_j\right) \tag{4.5}$$

Equation 4.5 can be represented as linear system $\mathbf{A}\mathbf{p}' = \mathbf{b}$. Assume that H is the set of handle vertices with user specified positional constraints. For vertex $i \in H$, the specified handle position is h_i that can be put as a hard constraint $\mathbf{p}' = h_i$. To use this hard constraint we divide $\mathbf{A}$ into two matrices: **A1** and **A2**.

A1 contains all rows whose indices are handles, and **A2** is used for the remaining vertices. we also remove all columns whose indices are handles from **A2**.

4.1.2 Initialization

The starting guess for vertices $\mathbf{p}'$ is obtained using laplacian editing [16] by simple linear minimization of $\|\mathbf{L}\mathbf{p}' - \delta\|$. under the positional modeling constraints. where, $\delta = \mathbf{L}\mathbf{p}$ are the Laplacian coordinates, and $\mathbf{p}$ and $\mathbf{p}'$ is the matrix contains absolute coordinates of the vertices of the reference mesh and deformed mesh respectively. Although this method produces distorted results for large deformations, the subsequent iterations manage to recover. However, the convergence may be slow for a significantly distorted initial guess.

Transformations are computed using the gradient descent method. It is also required to use good initialization for transformation because gradient descent gives local minima. We minimize the given energy function to initialize transformations.

$$E\left(\mathcal{C}_i, \mathcal{C}_i'\right) = \sum_{j \in \mathcal{N}(i)} w_{ij} \left\| \left(\mathbf{p}_i' - \mathbf{p}_j'\right) - \mathbf{T}_i \left(\mathbf{p}_i - \mathbf{p}_j\right) \right\|_2^2 \tag{4.6}$$

Minimization of the Equation 4.6 with respect to $\mathbf{T}_i$ gives a closed-form solution.

The energy function in Equation 4.1 is not a convex function with respect to the transformations $\mathbf{T}$ and is dependent on λ, r and initialization of vertices. Different initialization leads to different deformed vertices positions, and gradient descent gives the local minima.

4.1.3 Effects of normal

Consider a neighborhood of a vertex that contains all the co-planar edges in the reference mesh and the deformed mesh. In this case, transformations are not full rank. So multiple transformations are possible that transform the neighborhood

The solution is to consider the unit normal on the vertex and apply the transformation on that unit normal. The normal can be obtained using cross-product of any two non-parallel edges. After including normal, the optimization problem can be re-written as,

$$E\left(\mathcal{C}_k,\mathcal{C}_k'\right) = \sum_{(i,j)\in\mathcal{E}(k,r)} w_{ij}\left\|\left(\mathbf{p}_i'-\mathbf{p}_j'\right)-\mathbf{T_k}\left(\mathbf{p}_i-\mathbf{p}_j\right)\right\|_2^2 + \lambda\|\mathbf{T_k}^{\mathbf{T}}\mathbf{T_k}-\mathbf{I}\|_F^2+\|\hat{\mathbf{n}}_k'-\mathbf{T_k}\hat{\mathbf{n}}_k\|_2^2 \tag{4.7}$$

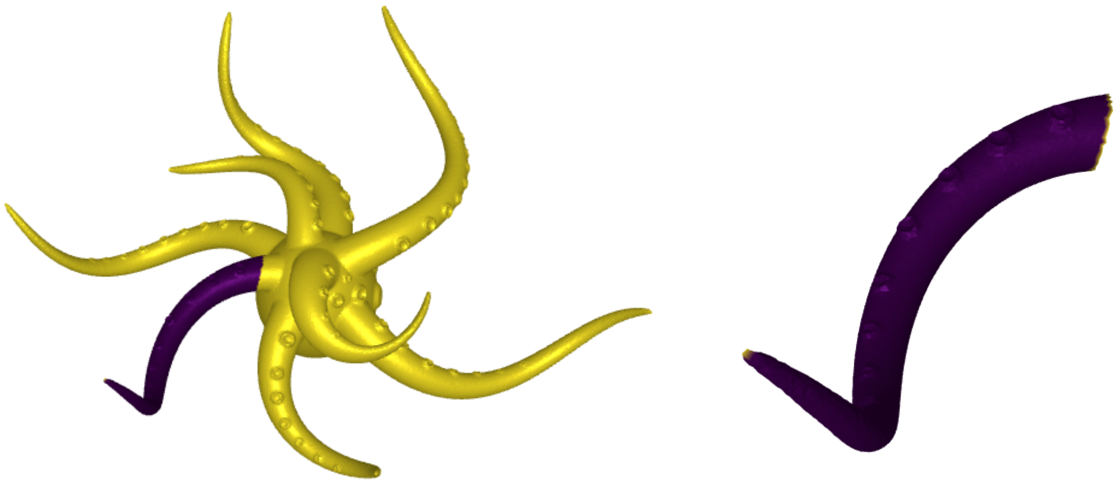
where $\hat{\mathbf{n}}_k$ and $\hat{\mathbf{n}}_k'$ are the unit normal of the k^{th} vertex in the reference mesh and deformed mesh respectively.

In this way, the rank deficiency problem is resolved, And the optimal transformation will be unique and invertible.

4.1.4 Deformation for submesh

In the proposed approach, if there are n vertices, there will be $9n$ unknowns for the transformations and $3n$ unknowns for the deformed vertices' position. When we change the position of the handles, then the handle's position will be fixed. If a large portion of the mesh is used as a handle the transformations on these vertices are also fixed. In this case, we can neglect some handles whose neighborhood also contains only handles in the optimization process.

In other words, we are considering those vertices in a sub mesh with at least one free vertex in their neighborhood. This is because the transformation applied on that vertex will not be fixed. This will reduce the number of unknowns in the optimization problem and takes less computation time.



(a) octopus mesh (b) sub mesh

Figure 4.3: Octopus mesh and its sub mesh used for the deformation

For example, If only one tentacle of the octopus is to be deformed, we keep the rest of the mesh fixed. The purple color represents the free vertices, and the yellow color represents the handles in Figure 4.3. Octopus mesh contains 50001 vertices. Among them, 45355 vertices are fixed, and only 4646 vertices are free (vertices on tentacle). We can observe that number of free vertices is significantly less than the number of total vertices. While optimizing, it takes unnecessary time to compute the transformations for the handles. Instead of finding the transformations for the whole mesh, we can compute the sub mesh of the one tentacle as shown in Figure 4.3 (b) and deform only the sub mesh. It is visible that in the sub mesh, free vertices are more and handles are less. For a vertex, the transformation is fixed if all the positions of the vertices in its neighborhood are fixed. Because of that, the sub mesh also contains some fixed vertices that have at least one free vertex.

Now, the whole algorithm will be applied to the sub mesh, and after performing the deformation, we can compute the whole mesh.

4.2 Estimating λ

Using the proposed energy function given in Equation 4.1, we can also estimate the rigidity of the object, i.e., λ.

Algorithm 1 Learn Material

```
procedure LEARN_MATERIAL($V_1$, $V_2$, $step\_size$, $threshold$)
    Initialize $i \leftarrow 0$, $step \leftarrow 0$
    $\lambda_0 \leftarrow 10^{-6}$
    FIND_HANDLES($V_1, V_2$)
    $\lambda_{final} \leftarrow 0$
    while True do
        $deformed\_V \leftarrow$ DEFORM($V_1$,$\lambda_i$)
        $step \leftarrow step\_size \times \|deformed\_V - V_2\|_F$
        $\lambda_{i+1} \leftarrow \lambda_i + step$
        if $\frac{\lambda_{i+1} - \lambda_i}{\lambda_{i+1}} < threshold$ then
            $\lambda_{final} \leftarrow \lambda_{i+1}$
            break
        end if
        $i \leftarrow i + 1$
    end while
    return $\lambda_{final}$
end procedure
```

The algorithm 1 takes two deformed meshes of the same object as input and finds appropriate λ for that object. We designate one mesh as a reference mesh and the other as the final mesh. In the algorithm 1 $v1$ is the vertices matrix of the reference mesh, and $v2$ is the vertices matrix of the final mesh. The assumption is that the final mesh ($v2$) is the deformed version of the reference mesh ($v1$). The idea is to deform the reference mesh to the final mesh using the proposed approach for an estimated λ. If the estimated λ is incorrect, the obtained deformed mesh will be different from the final mesh. Based on the difference between the deformed mesh and the final mesh (i.e., error in vertex position), change the value of λ and repeat the process until the difference between the vertices of the deformed mesh and the resulting mesh becomes very small.

In the proposed deformation approach user needs to provide the handle positions, but while learning λ handles are unknown. We find the transformations that minimize the energy function 4.6. We will get the identity transformation on those vertices whose 1 ring neighborhood is not deforming, and we can consider those vertices as handles.

If the relative change of λ is less than a fixed threshold, the algorithm will stop.

An algorithm 1 can also be used if more than two deformed meshes of the same object are given. The approach is simple, estimate the value of λ for each pair, and consider the maximum value among all the estimated λ.

CHAPTER5
Experiments and Results

The experiments have been performed on different geometrical meshes: bar, cylindrical, and cactus mesh, to support the mathematical arguments discussed in the previous chapter. Experiments are implemented on a Python backend using NumPy and libigl[8] with the processing power of an Intel core i5 processor with 8 GB RAM.

Meshes	Bar	Cylinder	Cactus
Vertices	722	4802	5261
Faces	1440	9600	10518
pre-computation	1.2 sec	7.5 sec	13.2 sec
Fastest time	1.7 sec	30.6 sec	132 sec
slowest time	11.4 sec	170.5 sec	261 sec
mean time	4.5 sec	100 sec	170 sec

Table 5.1: Details of meshes and corresponding computation time for deformation

Table 5.1 shows the information on the meshes and the corresponding computation time for computing the deformed meshes. In Table 5.1 pre-computation includes computing edges, normals and weights, finding r-ring neighbours and r-ring edges, computing matrix **A** mention in global step Chapter 4 and initialization of the vertices position using Laplacian mesh editing [16].

5.1 Effect of different λ for same deformation

Figure 5.1 shows the results on bar mesh for different values of λ. The vertices represented by yellow color are handles for deformation. r is set to 1, and the number of iterations is 100. Initially, the value of λ is very small, i.e., $\lambda = 0.01$ shows non-rigid deformation. It can be observed from the Figure that, as the value

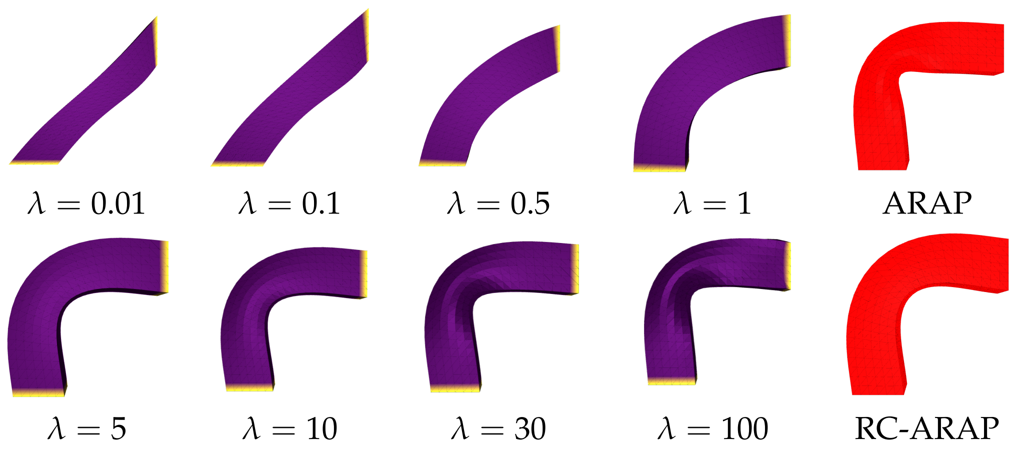


Figure 5.1: Effect of λ for same deformation for r = 1 and comparision with ARAP and RC-ARAP(r = 2)

of λ increases, the deformation becomes more rigid. Ultimately, for a higher value of λ, i.e., $\lambda = 100$, the deformation is similar to the ARAP result.

5.2 Effect of spatially varying λ

Figure 5.2 shows the results of different values of λ on different portions of the bar mesh. The values of λ are depicted using different colors, as shown in the first row of Figure 5.2. Higher the value of λ, darker the color. The largest value of the λ is 30, and the smallest value is 0.01. Second row in Figure 5.2 shows the bending effect on the mesh for corresponding values of λ in the first row. Similarly, the third row shows the twisting effect on the mesh. It can be observed that there is more stretching on the darker side that has a lower value of λ and the part of the mesh with higher value of λ shows a more rigid behavior.

5.3 Deformation using spatially varying r-ring and λ

Along with rigidity, experiments are also performed on controlling the rigidity of the neighborhood. To fulfill this purpose, we change the size of neighbourhood. Figure 5.3 shows the results on cylindrical mesh incorporating different r-ring neighborhood. The first row of the image grid represents the λ values on different mesh regions. Here, purple refers to $\lambda = 0.001$ and yellow refers to $\lambda = 1$. The first column of the image grid represents the value of r. Here, purple refers to $r = 1$ and yellow refers to $r = 2$. The entire image grid is constructed as a combination of different values of λ for the selected neighborhood of rigidity.

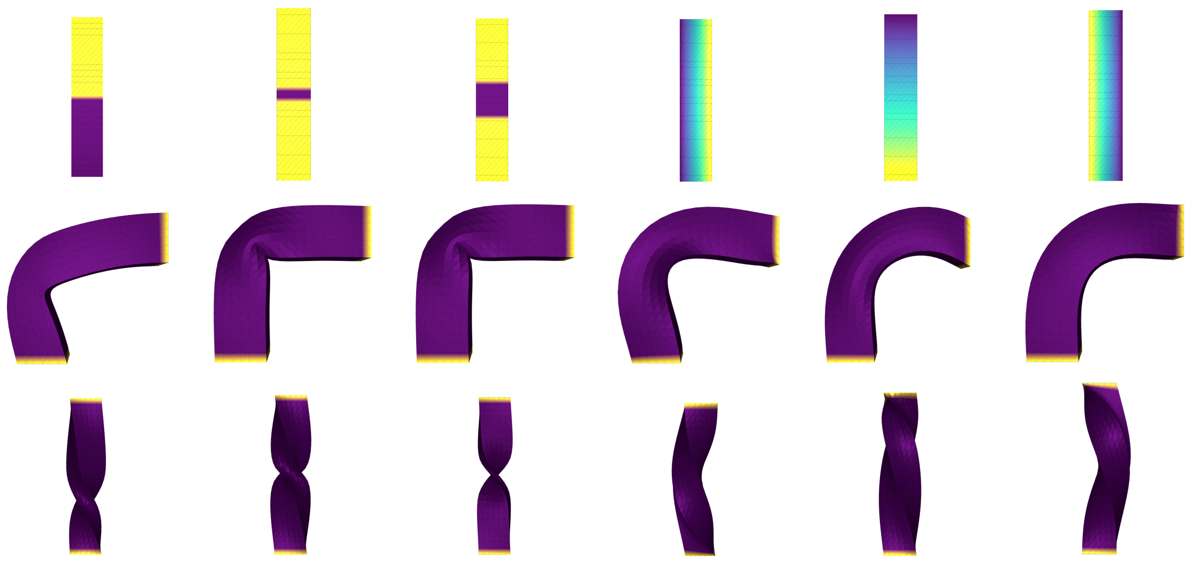


Figure 5.2: Bar contains different λ values for the different regions

.

For example, sub-figure 2A shows less rigidity, and the neighborhood chosen for rigidity is 1-ring. The last column (column-F) shows the results of the Rigidity control ARAP[3] method considering corresponding neighborhoods.

The deformation shown in Figure 5.3 had a bending effect. Identical experiments are done on the cylindrical mesh for the twisting deformation. These results are shown in Figure 5.4.

5.4 Comparison with Laplacian mesh deformation

In Figure 5.5 (a), handles are represented in yellow, and purple shows free vertices. The 2-ring neighborhood is represented in pink, and the 1-ring neighborhood is represented by blue in (b). Green represents $\lambda = 50$ and orange represents $\lambda = 0.01$ in (c). (d) and (e) is the deformation of the laplacian mesh edition(initialization) and proposed method. The green portion in (c) and pink portion in (b) show high rigidity. In the Laplacian mesh deformation, the middle portion of the cactus is squeezed, but the proposed approach is able to preserve thickness and appears more natural.

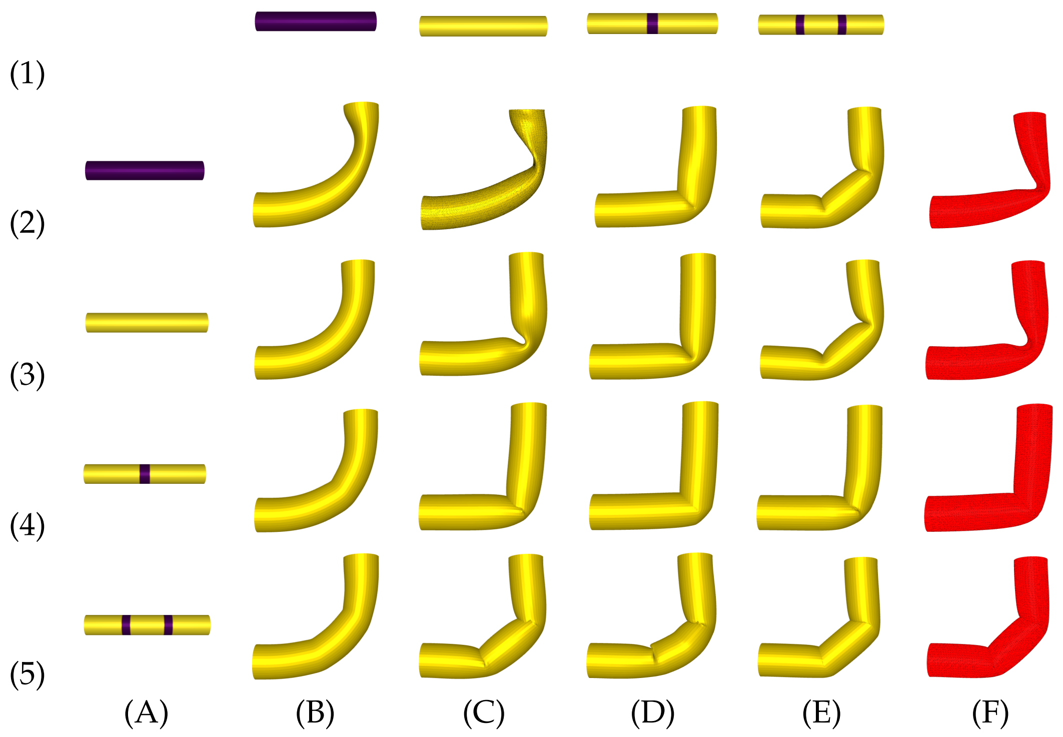


Figure 5.3: Bending deformation on Cylinder using different r-ring(column A) and λ(row 1)

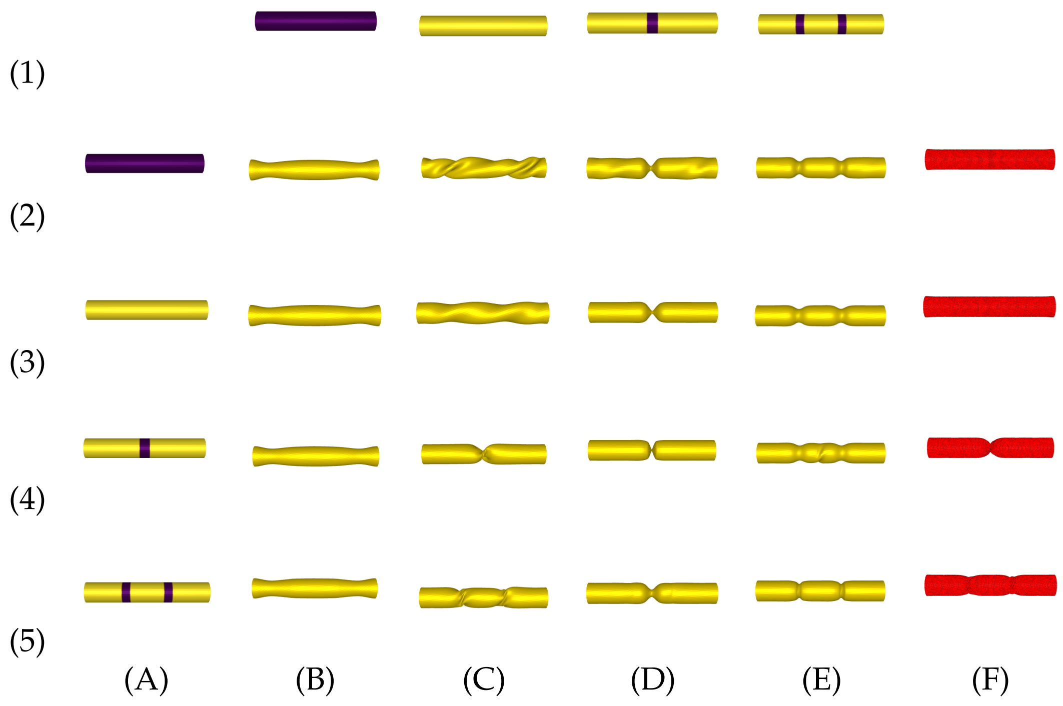


Figure 5.4: Twisting deformation on Cylinder using different r-ring(column A) and λ(row 1)

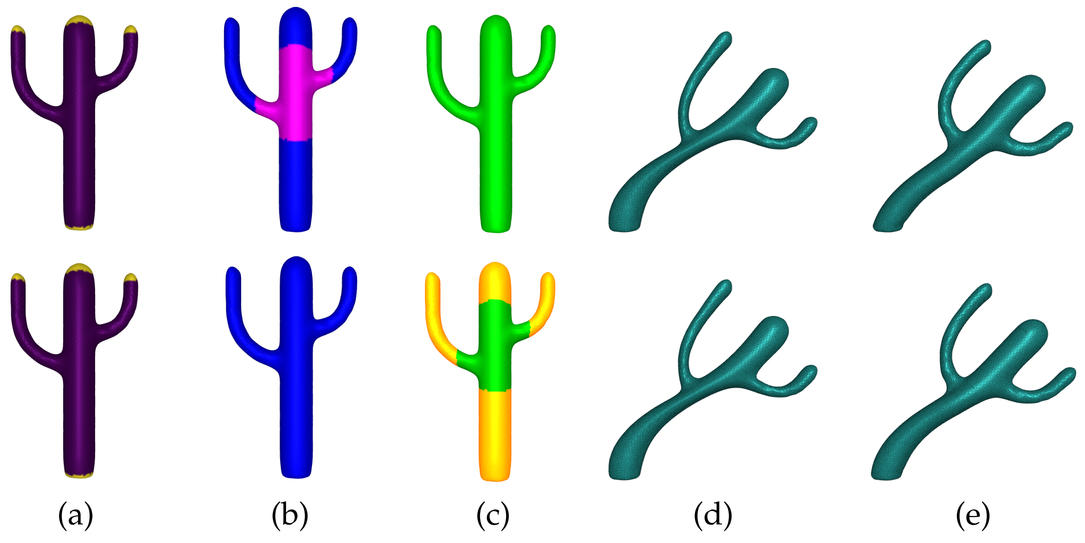


Figure 5.5: Deformation of cactus with different r-ring and λ, (a) Handles (b) r-ring (c) λ (d) Laplacian mesh editing deformation (e) deformation using Proposed method

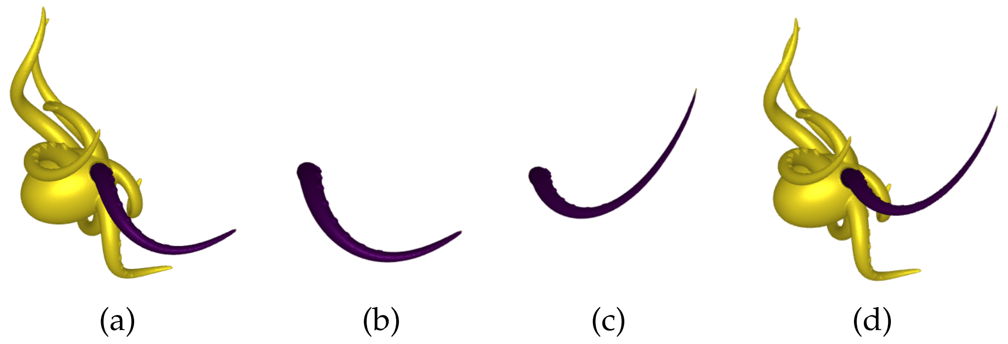


Figure 5.6: deformation of the the octopus mesh (a) to (d) by deforming only one the tentacle (b) to (c)

process	computational time
Computation of sub mesh (a) to (b)	5.6 sec
Deformation of sub mesh (b) to (c)	155.6 sec
Restore the whole mesh (c) to (d)	2.2 sec
deform the whole mesh without computing sub-mesh (a) to (d)	700 sec

Table 5.2: computational time distribution for Figure 5.6

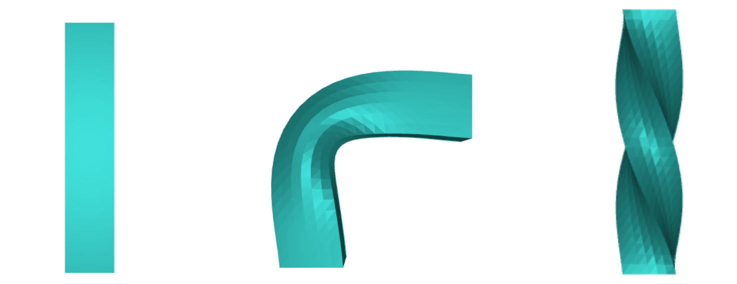

(a) Mesh 1 (b) Mesh 2 (c) Mesh 3

Figure 5.7: Meshes used to estimate rigidity(λ)

5.5 Deformation using sub mesh

As shown in Figure 5.6, we want to deform one octopus tentacle represented by the purple in (a). Time distribution is shown in the Table 5.2. Generating a deformed mesh (d) takes nearly 700 seconds if we deformation the whole octopus mesh. However, if we extract one tentacle as sub mesh and deform it only, that tentacle takes 163.4 seconds to deform during mesh.

5.6 Experiment for Estimating λ

To show the experiment of the estimating λ, we use three different deformed meshes, as shown in the Figure 5.7 of the bar mesh. Mesh 2 and Mesh 3 are generated from Mesh 1 using $\lambda = 30$. The result of the estimation of λ is given in Table 5.3.

The Figure 5.8 algorithm minimizes the difference between deformed and result mesh vertices. As λ increases, we can see that the difference between vertices' location decreases. Because the rate of change in λ is dependent on the difference between all vertices of the deformed mesh and the resulting mesh after some value of λ deformation remains the same. The algorithm converges at $\lambda = 8.052$. We can consider the largest value of λ among all predicted values of λ. In this experiment, we are not getting the exact λ. one can conclude from this is, using the estimated λ, we can get similar deformation that shows that to perform this particular deformation, we do not need too much rigidity.

Figure 5.9 shows the output of the deformations using estimated λ where column number represents the final mesh, and row number represents the reference mesh.

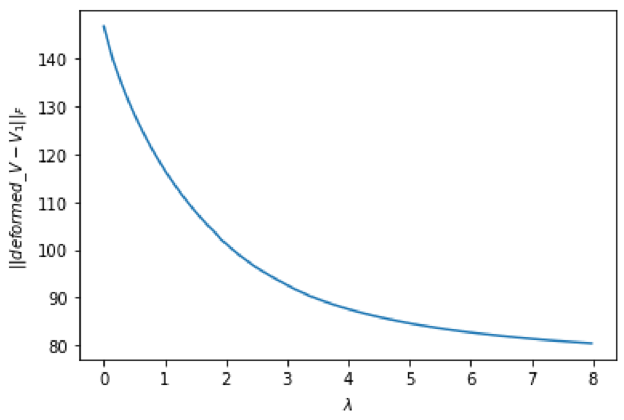


Figure 5.8: λ vs $\|deformed\_V - V_1\|_F$ for Mesh 3 as reference mesh and Mesh 1 as final mesh

Reference mesh	**Final mesh**	**Estimated λ**
Mesh 1	Mesh 2	5.557
Mesh 1	Mesh 3	1.920
Mesh 2	Mesh 3	20.471
Mesh 2	Mesh 1	23.123
Mesh 3	Mesh 1	8.052
Mesh 3	Mesh 2	5.557

Table 5.3: Learned λ from reference mesh and final mesh

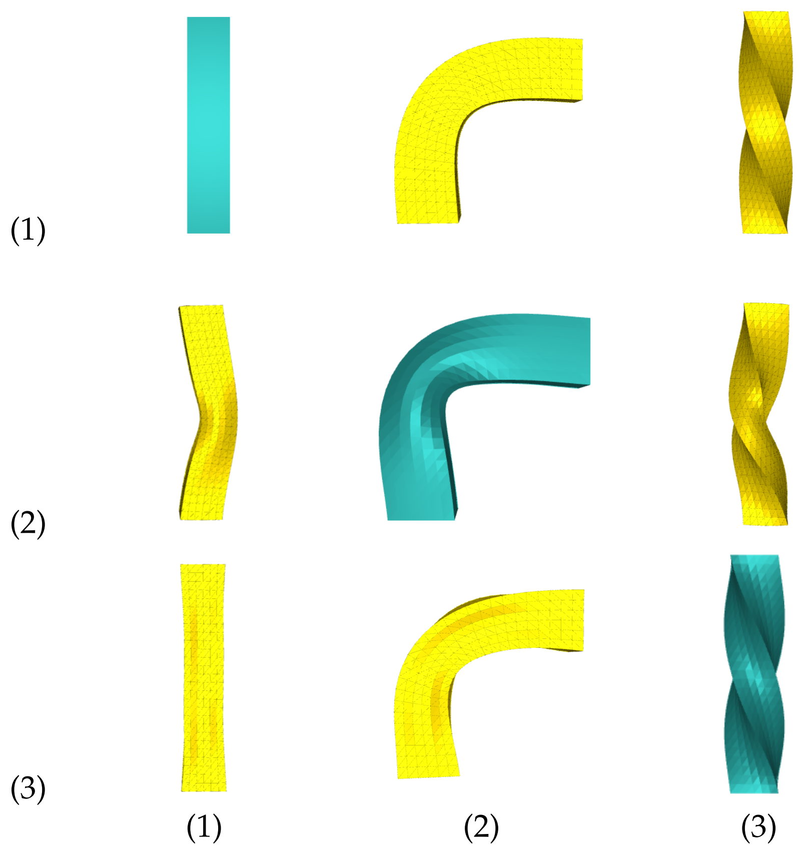


Figure 5.9: Deformed meshes using Learnt λ where row number represents reference mesh and column number represents final mesh

CHAPTER6

Conclusion and Future Work

The proposed approach gives all functionality of ARAP and RC-ARAP. It also removes the limitations of both methods providing the choice to control the r ring neighborhood's local rigidity, as well as allows spatially varying rigidity control. We have also proposed a method to estimate the rigidity of a mesh from examples. Our algorithm, by design, provides the least rigidity required to produce the examples.

As of now, the deformation algorithm runs much slower than interactive speeds. The bottleneck is gradient descent based optimization. We want to explore faster optimization methods and parallel implementation to reach interactive speeds. Estimating the rigidity of the mesh is currently a preliminary study. Faster execution time working with a larger set of meshes and estimating spatially varying rigidity are some future directions we would like to explore.

References

[1] "types of mesh - wikipedia". https://en.wikipedia.org/wiki/Types_of_mesh. (Accessed on 05/17/2022).

[2] A. I. Bobenko and B. A. Springborn. A discrete laplace–beltrami operator for simplicial surfaces. *Discrete & Computational Geometry*, 38(4):740–756, 2007.

[3] S.-Y. Chen, L. Gao, Y.-K. Lai, and S. Xia. Rigidity controllable as-rigid-as-possible shape deformation. *Graphical Models*, 91:13–21, 2017.

[4] W. Frei. "meshing your geometry: When to use the various element types | comsol blog". https://www.comsol.com/blogs/meshing-your-geometry-various-element-types/. (Accessed on 04/29/2022).

[5] L. Gao, Y.-K. Lai, J. Yang, L.-X. Zhang, S. Xia, and L. Kobbelt. Sparse data driven mesh deformation. *IEEE transactions on visualization and computer graphics*, 27(3):2085–2100, 2019.

[6] A. Jacobson, I. Baran, J. Popovic, and O. Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, 30(4):78, 2011.

[7] A. Jacobson, Z. Deng, L. Kavan, and J. P. Lewis. Skinning: Real-time shape deformation (full text not available). In *ACM SIGGRAPH 2014 Courses*, pages 1–1. 2014.

[8] A. Jacobson, D. Panozzo, C. Schüller, O. Diamanti, Q. Zhou, N. Pietroni, et al. libigl: A simple c++ geometry processing library, 2018.

[9] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics (TOG)*, 27(4):1–23, 2008.

[10] L. Kavan and J. Žára. Spherical blend skinning: a real-time deformation of articulated models. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 9–16, 2005.

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

[12] K. B. Petersen and M. S. Pedersen. The matrix cookbook, Oct. 2008. Version 20081110.

[13] X. Shi, K. Zhou, Y. Tong, M. Desbrun, H. Bao, and B. Guo. Example-based dynamic skinning in real time. *ACM Transactions on Graphics (TOG)*, 27(3):1–8, 2008.

[14] J. Solomon. *Numerical algorithms: methods for computer vision, machine learning, and graphics*. CRC press, 2015.

[15] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007.

[16] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, 2004.

[17] R. W. Sumner, M. Zwicker, C. Gotsman, and J. Popović. Mesh-based inverse kinematics. *ACM transactions on graphics (TOG)*, 24(3):488–495, 2005.

[18] A. Tagliasacchi, T. Delame, M. Spagnuolo, N. Amenta, and A. Telea. 3d skeletons: A state-of-the-art report. In *Computer Graphics Forum*, volume 35, pages 573–597. Wiley Online Library, 2016.