CS 399: Automating Anamorphic Art using Signed Distance Functions

Patel Vrajesh (20110134)
Indian Institute of Technology, Gandhinagar, India
patel.vrajesh@iitgn.ac.in

R Yeeshu Dhurandhar (20110152)
Indian Institute of Technology, Gandhinagar, India
r.yeeshu@iitgn.ac.in

Supervisor: Prof. Shanmuganathan Raman

Guide: Mr. Ashish Tiwari

Github: https://github.com/RYeeshuDhurandhar/automating-anamorphic-art

***Abstract*—In the field of anamorphic art, the representation of 3D objects is a crucial aspect that determines the visual quality of the final artwork. Two commonly used approaches for representing 3D objects are mesh-based and voxel-based techniques. However, these techniques are often memory-intensive, which can lead to performance issues when dealing with complex objects. We did an exhaustive study of accuracy of mesh-based, voxel-based, and Signed Distance Function (SDF) approaches for representing 3D objects in anamorphic art. We understood that while mesh-based and voxel-based approaches can be memory-intensive, SDFs provide a more accurate representation of 3D objects with significantly lower memory requirements. Furthermore, SDFs can be used to generate high-quality anamorphic art with minimal distortion, making them a suitable choice for applications that require high visual fidelity. We conclude that SDFs are a more efficient and practical solution for representing 3D objects in anamorphic art, and suggest that they should be considered as a preferred alternative to traditional mesh and voxel-based techniques.**

I. Introduction

Anamorphic art refers to the use of digital techniques to create 3D representations that appear differently flattened images when rendered from different angles and/or positions.

Fig. 1. Anamorphic Art by Michael Murphy: Left view followed by 3D arrangement followed by right view

The shapes shown in Fig.1. are made up of tiny spheres hanging in such a way that if it is seen from a central location, it seems like any random shape. When seen from the left, it looks like a power button, and when seen from the right, it seems like a symbol of women's rights movements. The current approach for such kind of art is to use some software for representing 3D shapes and render the scene to find the projection on a different plane, then manually rotate and translate to get the optimal rotation and translation of objects. Complex arts may take days and even weeks. This is our attempt to reduce the time significantly by automating the process of rendering and getting the optimal potion and translation of different shapes.

Signed Distance Functions (SDFs) are a powerful tool for creating 3D representations due to their ability to represent more efficiently and accurately. Given a spatial point [x,y,z], SDFs will output the distance from that point to the nearest surface of the represented object. The sign of the SDF indicates whether the queried point is inside, outside, or on the surface of the underlined object.

$$SDF(x) = s : x \in \mathbb{R}^3, s \in \mathbb{R}^1 \tag{1}$$

The concept of the signed distance function can be understood by a simple example in 2D space using the circle. The equation of circle can be represented as:

$$SDF(x, y) = (x - a)^2 + (y - b)^2 - r^2 = 0 \tag{2}$$

Here, a and b are the x and y coordinates of the center of the circle, and r is the radius of the circle. For any (x, y) in 2D space and a particular value of r, there are three possible scenarios: i) $SDF(x, y) < 0$, for points inside the circle, and its magnitude represents the distance of the queried point (x, y) from the nearest surface. ii) SDF(x, y) = 0, for points on the circle. iii) $SDF(x, y) > 0$, for points outside the circle, and its magnitude represents the distance of the queried point (x, y) from the nearest surface. Similarly, for any shape in any dimension (as we can generalize it for any dimension), if the function returns the values that satisfy the above three conditions, we call that particular function the signed distance function of the given shape.

A complex shape can be represented using simpler objects with the help of three set operations: union, intersection, and difference. Fig.2. shows an example of constructive solid geometry obtained using set operations of simple shapes whose signed distance functions are known.

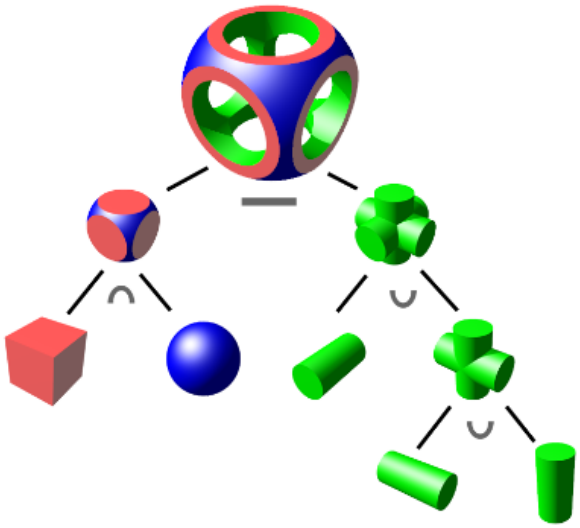

Fig. 2. An example of Constructive Solid Geometry

II. Problem Statement

Given some rigid objects, we have to arrange them in such a way that when we see them from different directions, we perceive different meaningful images. This problem boils down to the following:

1. Finding a way to represent 3D objects that can be rotated and translated computationally efficiently and find intersection volume and projection is possible.
2. Finding the optimal translation (coordinates of the centroid of shape) and rotation (angles of rotation in x, y, and z direction) parameters of each object inside a bounding box.
3. Minimize the intersection volume between every pair of objects, as they can not intersect in real life. We will refer to this as intersection loss.
4. Minimize the difference between true and projected images; we will refer to this difference as projection loss.

This can also be seen as a packing problem. Among the different 3D shape representation techniques, namely Mesh-based, Point-based, Voxel-based, and Signed Distance Function (SDF) based representation, SDFs are most suitable for the abovementioned criterion. There were several attempts to solve the packing problem but not using signed distance functions. A previous attempt to solve the same problem statement was made by a research group under Professor Shanmuga at IIT Gandhinagar. In work, only projection loss was considered, which was able to produce an object whose projection is similar to the true image, but it suffered from the problem of collisions, as shown in Fig.3. Therefore, it motivated us to use the signed distance function to get the intersection loss easily, which we can reduce using machine learning algorithms. To the best of our knowledge, this is the first attempt to solve it using signed distance functions.

The intersection loss is not considered in Fig.3 since the set operations are not trivial for the objects represented using meshes. However, when objects are represented using SDFs, we can represent union, intersection, and other set operations just by taking min./max./other simple mathematical operations as shown in Fig.4 and equations (3),(4), and (5). Moving forward, we have made two assumptions. The objects are rigid, and orthographic projection is used to get the projected image of the scene.

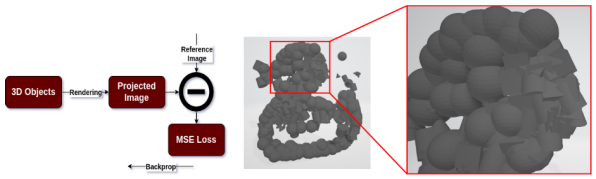

Fig. 3. Intersection loss not considered: Mesh representation of Bunny has problem of object collision

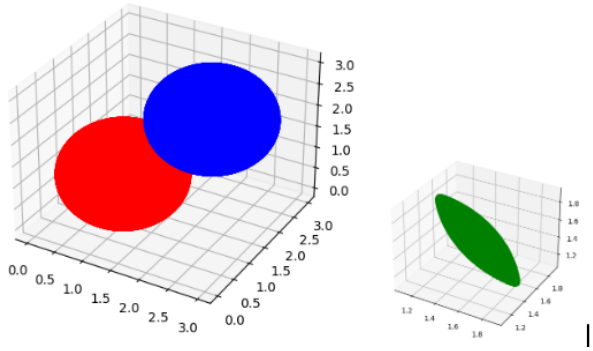

Fig. 4. Union and Intersection of a cube and a sphere represented using SDFs

$$\text{union}(\text{obj1}(p), \text{obj2}(p)) = \min(\text{obj1}(p), \text{obj2}(p)) \quad (3)$$

$$\text{intersection}(\text{obj1}(p), \text{obj2}(p)) = \max(\text{obj1}(p), \text{obj2}(p)) \quad (4)$$

$$\text{difference}(\text{obj1}(p), \text{obj2}(p)) = \max(\text{obj1}(p), -\text{obj2}(p)) \quad (5)$$

III. Methodology

We approached the problem through two paths; the first uses the functional form of signed distance functions (represented in blue), and the second uses DeepSDFs, which uses a .obj file and passes it through the trained model to get sampled SDF (represented in brown). In the first approach, the 3D shapes are represented using signed distance functions. The goal is to learn each shape's optimal x, y, and z coordinates of the centroid and rotation angles along the x, y, and z axes. Therefore, k number of shapes results in kx6 number of learnable parameters. All these boxes are placed inside a bounding box, and the points are taken from the bounding box. Therefore, we get n points from a bounding box and pass these points and the learnable parameters through signed distance functions to get the rotated and translated representation of shapes. These functions take n points and return n signed distances. After getting the rotated representation of shapes, we need to find two losses, projection loss, and intersection loss. To get the projection loss, we took the union of all shapes

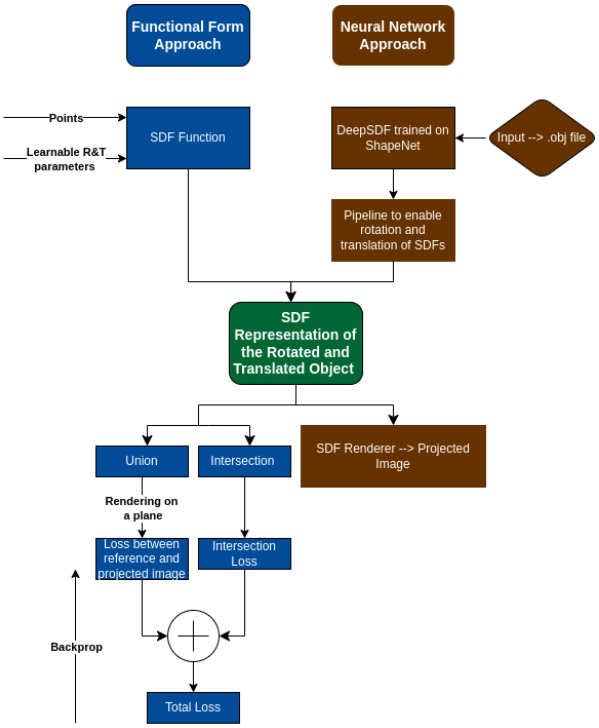

Fig. 5. Pictorial Representation of our workflow

and projected them in the x-y plane. To do this, suppose we have a bounding box, and at each discrete point inside the box, we have signed distances of all the shapes considered. To get the union, take the minimum distances at each end. Then to project the scene, for each x-y coordinate, traverse through all the possible z values. If we find any negative or zero values (showing the point is inside or on the surface), we store one corresponding to that x-y coordinate. If only positive values are found, then we store 0. This way, we get a 2D tensor having 1's and 0's. We then take the mean square loss of the predicted image with the true image, which is also represented using 1's and 0's (Fig. 7) To get the intersection loss, we take

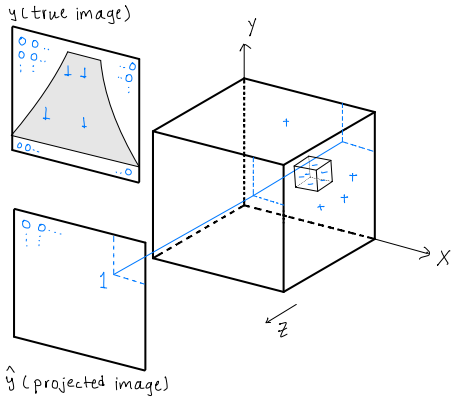

Fig. 6. Projection of union of shapes represented using signed distance functions

each pair of objects and the max of signed distances at each point, giving the intersected points. For all pairs of points, we sum the number of intersecting points which constitutes the intersection loss. The final loss is the sum of projection loss and intersection loss. Then we backpropagate using Adam optimizer and update the value of learnable parameters, i.e., rotation and translation parameters.

In the second approach, we take a .obj file of the object and pass it through the pre-trained DeepSDF model, which samples points near the boundary of the surface to get better results. Then the found sampled SDF is passed through a pipeline which rotates and translates it. Then we render it using an SDF renderer to get a projected image. This can be used as projection loss.

In the first approach, the whole pipeline is created by us, including the rendering of the scene. In the second approach, the rotation and translation pipeline is built by us.

IV. Results

One of the essential parts of the project was to find the signed distance functions of 3D shapes, which can be rotated and translated, and set operations that are easy to perform. Therefore, we write functions that take points in 3 dimensions, shape parameters, translation, and rotation parameters, and return the signed distance of each point from its nearest surface. We have written functions of different shapes in Python. To speed up the process, we implemented these functions in vector representation which takes n points and returns n signed distances. The result of these functions is shown in Fig.7.

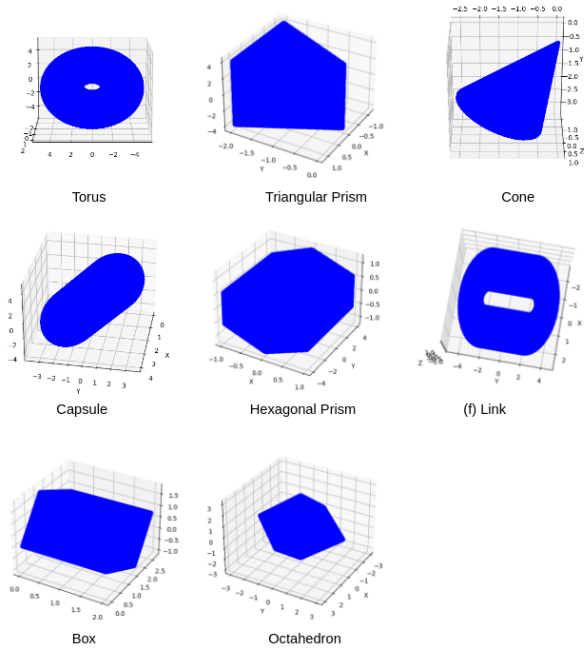

Fig. 7. Dataset prepared using the functional definition of SDFs

Considering intersection loss needed to be included in the previous approach using meshes; therefore, we first focused on

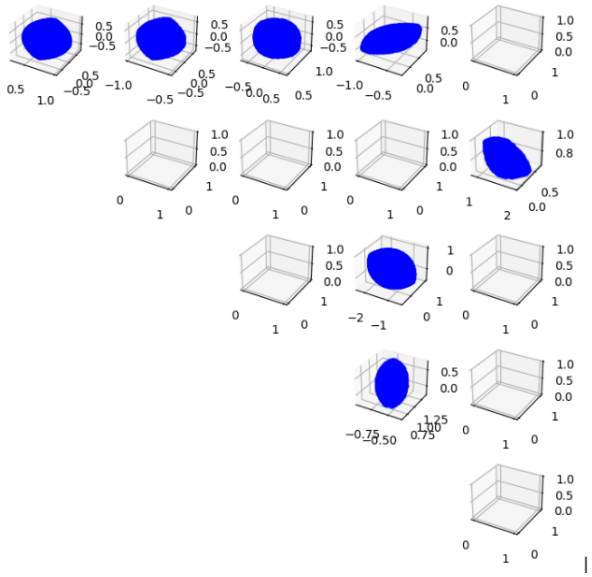

Fig. 8. Intersection loss of each pair of shapes before backpropagation

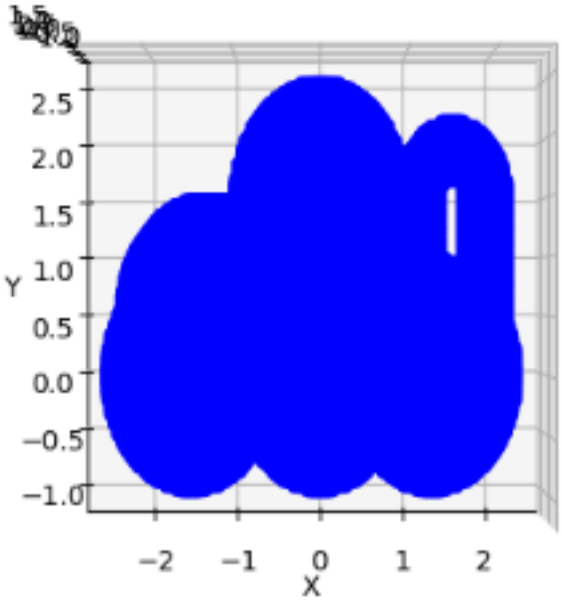

Fig. 9. Projection of all shapes in the x-y plane before backpropagation

reducing the intersection loss. After getting the signed distance function of individual shapes, we find intersection loss and backpropagate it to learn translation and rotation parameters, reducing the loss. For this purpose, we take five spheres and one link and find the intersection between each pair of shapes (Fig.8). In Fig.8, the first image shows the intersection between sphere 1 and sphere 2, the second image shows intersection between the first sphere and the third sphere, and so on. Fig.9 shows the projection of all objects in the scene in the x-y plane. After back-propagating using the Adam optimizer, we observed that the rotation and translation parameters had been changed so that the object moved far apart, and intersection

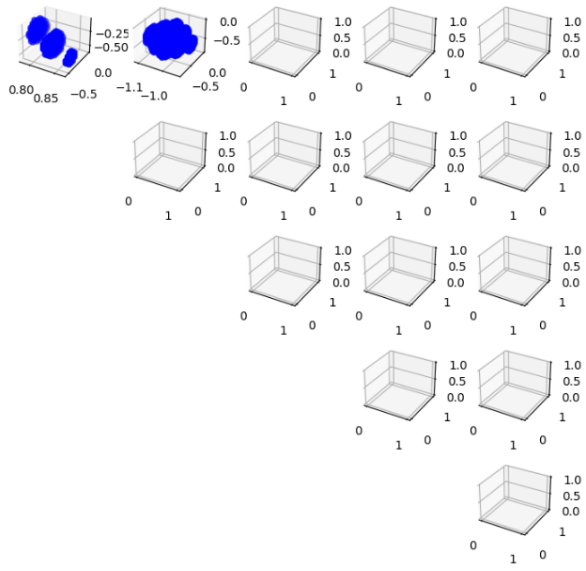

Fig. 10. Intersection loss of each pair of shapes after backpropagation

loss was reduced. Fig.10 shows the intersection losses between each pair of shapes after backpropagation. It can be seen that the intersection volume has been significantly reduced. If the number of iterations increases, we eliminate all the intersections. Fig.11 shows the projection of all shapes in the x-y plane. It can be observed that there are some gaps between objects as compared to Fig9, which represents the translation of objects in the x and y directions. The object's translation in the z direction is not visible in the projection.

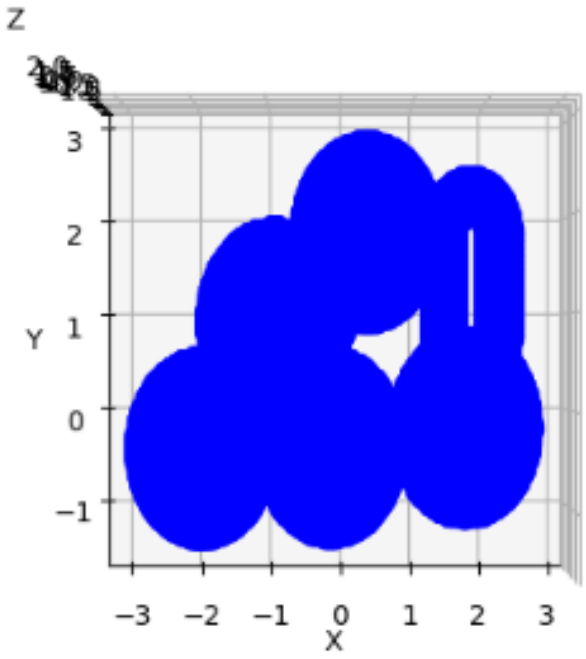

Fig. 11. Intersection and Projection before and after training

Apart from this. we have incorporated the rotation and transla-

tion pipeline in the DeepSDF neural network implementation. Fig.12 shows the results of the same. We have also rendered the image of a sphere (Fig.13) using the SDFRenderer of the facebook's Implicitron project, which is also pipelined in our model. It can also be noted that meshes did not capture the fine details of the object.

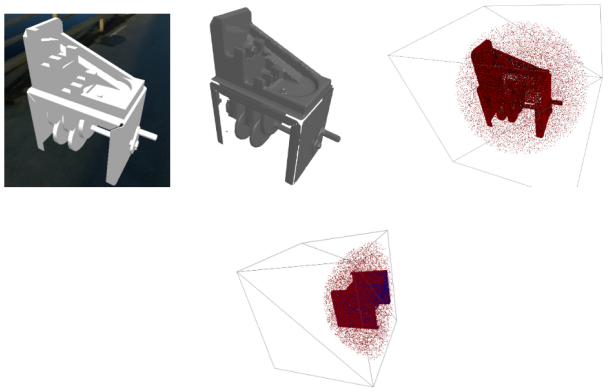

Fig. 12. Input Object : Mesh Based Representation : SDF Representation : R and T of SDF Representation

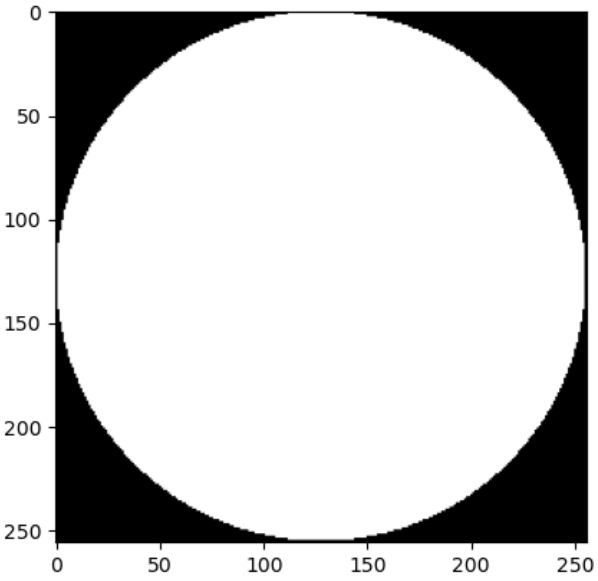

Fig. 13. SDFRenderer to render a SDF scene (here SDF representation of sphere)

V. Conclusion

We have developed a pipeline using signed distance functions (SDFs) to represent shapes and perform set operations for anamorphic art accurately. The pipeline involves determining the optimal translation and rotation parameters of different objects using the SDF to calculate intersection and projection loss. Backpropagation is then used to obtain the optimal parameters for each shape. We have also made our custom renderer to render the scene represented using SDFs. We observed that the intersection loss has higher gradient flow while the projection loss has very low/no gradient flow. Hence, we plan to improve the same in future.

VI. Future Work

Till now, our work reduced intersection loss. It also finds the projection of the scene and projection loss. We will improve to incorporate parameter learning by back-propagating projection loss. Also, we approached the problem with two paths, as shown in Fig. 6, and they are independent. We plan to merge them to get the best out of both approaches. Further, we plan to extend the model to incorporate anamorphic art seen from different angles by adding projection losses from different projections. We plan to incorporate color and texture into the images to achieve real-world anamorphic art.

VII. Acknowledgement

We are immensely grateful to Prof. Shanmuganathan Raman for giving us the invaluable opportunity to work under his guidance and providing us with the necessary references to smoothen our journey. Additionally, we would like to express our sincere appreciation to Mr. Ashish Tiwari, our Ph.D. mentor, for presenting us with the problem statement and for providing unwavering support throughout the project. His constant guidance, insightful suggestions, and prompt resolution of our queries were instrumental in helping us progress, especially when we encountered obstacles along the way.

VIII. References

1. Park, Jeong Joon, et al. "Deepsdf: Learning continuous signed distance functions for shape representation." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019.
2. Sadekar, Kaustubh, Ashish Tiwari, and Shanmuganathan Raman. "Shadow art revisited: a differentiable rendering based approach." Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2022
3. https://jamie-wong.com/2016/07/15/ray-marching-signed-distance-functions/
4. https://michaelwalczyk.com/blog-ray-marching.html
5. https://www.ronja-tutorials.com/post/034-2d-sdf-basics/
6. https://iquilezles.org/articles/distfunctions/
7. https://fpcv.cs.columbia.edu/
8. https://pytorch3d.org/docs/dataset
9. https://opensource.fb.com
10. https://en.wikipedia.org/wiki/Constructivesolidgeometry
11. https://www.perceptualart.com