CS 499: Automating Anamorphic Art using Signed Distance Functions

R Yeeshu Dhurandhar (20110152)
r.yeeshu@iitgn.ac.in
IIT Gandhinagar

Ashish Tiwari
ashish.tiwari@iitgn.ac.in
IIT Gandhinagar

Shanmuganathan Raman
shanmuga@iitgn.ac.in
IIT Gandhinagar

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.

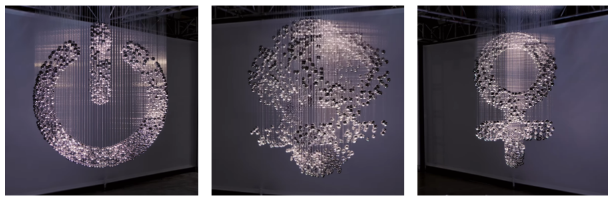


Fig. 1: Anamorphic Art by Michael Murphy: Left view followed by 3D arrangement followed by right view

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.

II. Importance of the problem

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

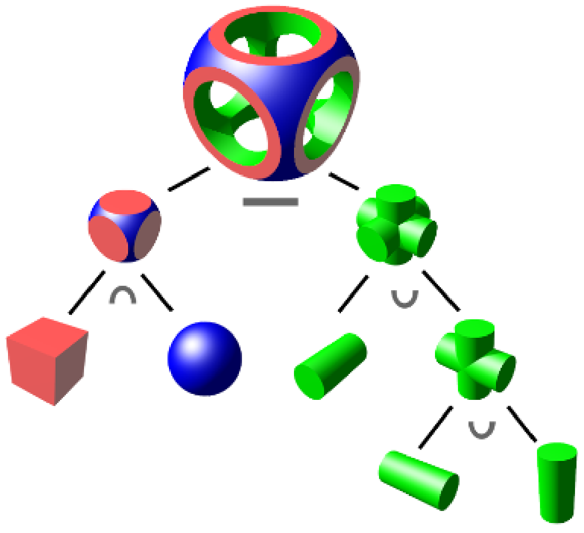


Fig. 2: An example of Constructive Solid Geometry

the process of rendering and getting the optimal potion and translation of different shapes.

III. Problem Statement

To tackle the challenge of arranging rigid objects in a way that yields different meaningful images when viewed from different angles, we can break down the problem into several key components. The goal is to optimize the arrangement of these objects in a computational framework. This multifaceted problem can be structured as follows:

1. Efficient Representation of 3D Objects: One of the foundational aspects is to devise a computationally efficient representation for 3D objects that allows for both rotation and translation. This representation should also facilitate the calculation of intersection volumes and projections. Achieving this efficiency is crucial for real-time applications.

2. Determining Optimal Object Placement: To create the desired effect, it's necessary to pinpoint the optimal translation parameters, specifically the coordinates of the centroid of each object, and the rotation parameters, which involve angles of rotation about the x, y, and z axes. These parameters dictate how each object fits within a bounding box to achieve the desired visual outcomes.

3. Minimizing Intersection Volume: Real-world objects cannot intersect with one another. Hence, it's imperative to minimize the intersection volume between every pair of objects. This minimization of intersecting regions is referred to as the "intersection loss." Solving this problem ensures that the arrangement adheres to physical constraints.

4. Aligning True and Projected Images: To create a convincing visual experience, it's essential to minimize the difference between the true images (the intended visual output) and the images projected from the current object arrangement. This disparity is quantified as the "projection loss." Achieving a minimal projection loss ensures that the rendered images closely match the intended ones.

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 above mentioned 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.

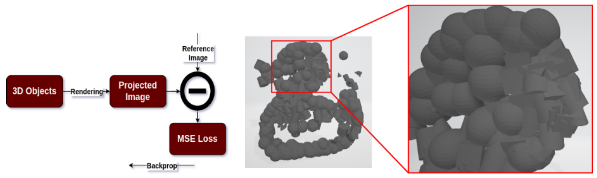


Fig. 3: Intersection loss not considered: Mesh representation of Bunny has problem of object collision

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.

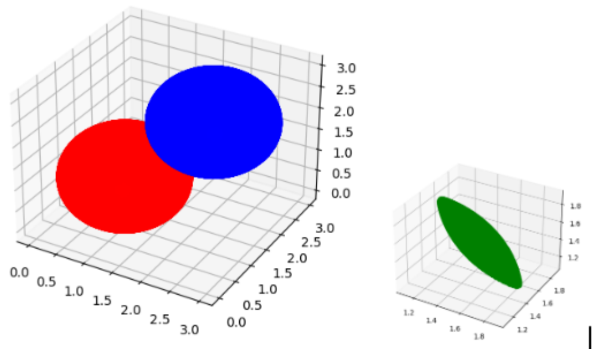


Fig. 4: Union and Intersection of a cube and a sphere represented using SDFs

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

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

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

IV. PROPOSED STRATEGY

The strategy to address this complex challenge involves two fundamental approaches:

Approach 1: Using Signed Distance Functions (SDF): The first approach employs Signed Distance Functions as a representation for objects. SDF provides a convenient means of defining the shape and position of objects, enabling efficient computational manipulation. It also facilitates the computation of intersection volumes and projections. SDF-based representation offers flexibility in achieving the desired visual effects.

Approach 2: Using Mesh Representation: The second approach involves representing objects using mesh structures. Meshes allow for the effective computation of rotations and translations, as well as the determination of intersection volumes and projections. Mesh-based representation is versatile and can be utilized to solve the optimization problem.

In practical terms, the process involves starting with binary true images and initializing the translation and rotation parameters for each object. The objective is to minimize both the intersection loss, ensuring that objects do not overlap, and the projection loss, aligning the projected images with the true images. This optimization process iteratively refines the position and rotation parameters of the objects to achieve the desired visual outcome.

In summary, the problem of arranging rigid objects for varied visual perceptions is a multifaceted challenge that involves efficient object representation, optimization of placement parameters, and minimizing intersection and projection discrepancies. The chosen strategy combines the use of Signed Distance Functions and mesh representations to enable the computation of these parameters, ultimately leading to the creation of compelling and dynamic visual experiences.

V. METHODOLOGY

A. *Signed Distance Functions (SDF) based Approach*

We used the signed distance function approach in two ways; the first uses the functional form of signed distance functions and second uses conversion of mesh to SDF.

*1) **Functional Form of SDF:*** In this case, 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 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 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 previous work during last semester [Report in the GitHub Repo], we implemented the intersection loss and projection loss. The model was able to backpropagate through intersection loss and update the rotation and translation parameters to reduce the intersection among objects. The projection algorithm finds the projection of the scene and is compared with the true image to calculate projection loss. The problem that we encountered was during the backpropagation of the projection loss. The algorithm was implemented in such a way that for a 3D scene containing multiple objects inside a bounding box, for each (x,y), we traversed through all z and if we encountered any negative value (representing the point inside the object), we take it as 1 in the projection, otherwise, 0. The operation of taking 1 if we encounter a negative value is not differentiable causing the backpropagation to not work. To solve those problems, we followed the following steps:

- Solved the vanishing gradient problem: We solved the problem of vanishing gradient while backpropagating the projection loss. The main problem was assigning 1 directly if we encountered any object for any particular (x,y) and all z's corresponding to it. Therefore, the solution is to make it mathematically rather than directly assigning it 1. For this, we used the steps showed in Fig. 8.

  In the Fig.8, (1) and (2) make the processes of calculating the projection differentiable and make backpropagation possible.
- Made attempts by taking different true images of elementary shapes (such as triangles, V-shape, etc.) and 3-D objects (such as spheres, etc.), and checked if the algorithm was working fine in the toy examples.
- Through experiments, we observed that the initial objects should be placed in such a way that their projection should have a few common points with the true image. If the projection of any object does not have any intersection with the true image, then if we move the object in any direction, the loss would remain the same which results

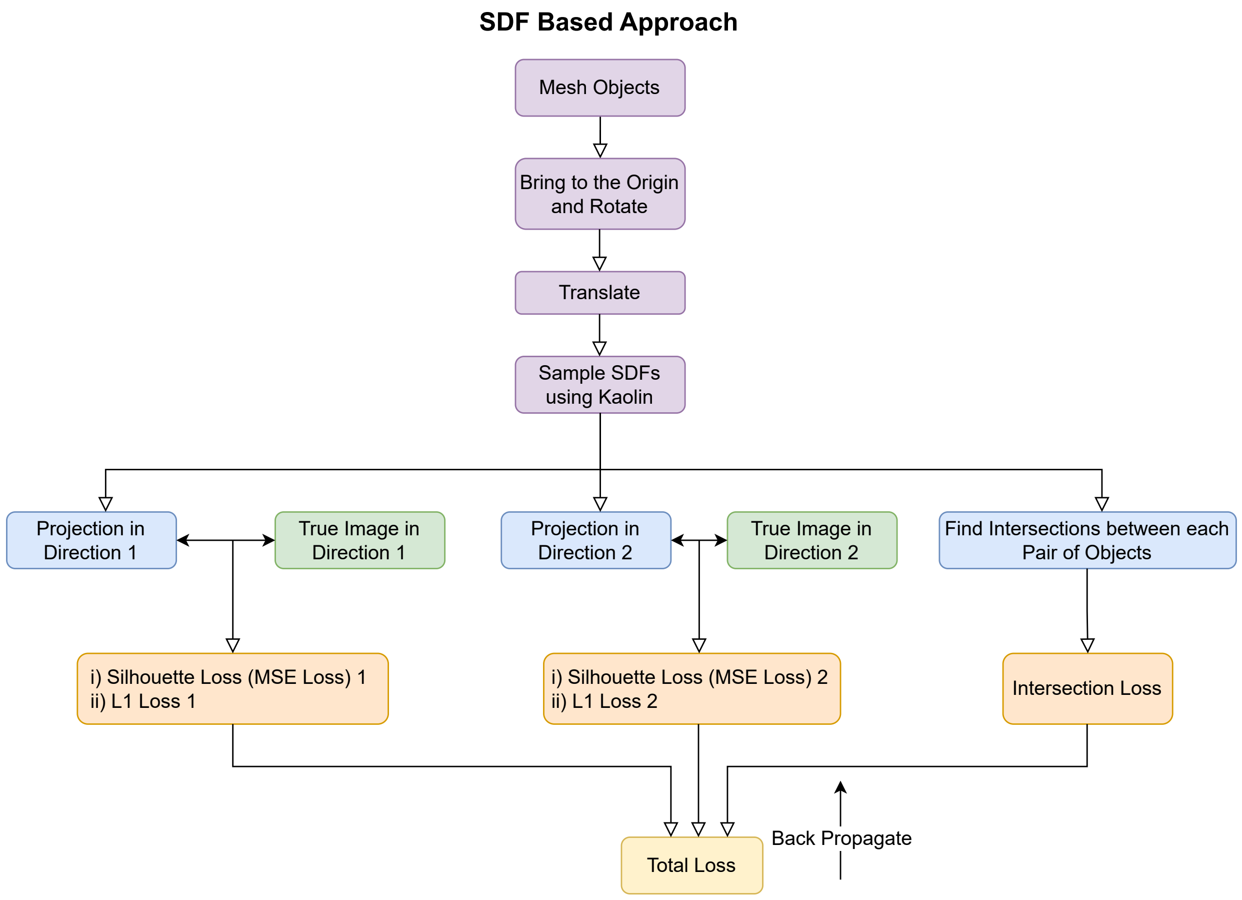

SDF Based Approach
Mesh Objects
Bring to the Origin and Rotate
Translate
Sample SDFs using Kaolin
Projection in Direction 1
True Image in Direction 1
Projection in Direction 2
True Image in Direction 2
Find Intersections between each Pair of Objects
i) Silhouette Loss (MSE Loss) 1
ii) L1 Loss 1
i) Silhouette Loss (MSE Loss) 2
ii) L1 Loss 2
Intersection Loss
Total Loss
Back Propagate

Fig. 5: Modified SDF based approach for Anamorphic Art.

in the zero gradient. Due to this, the object would not be able to move and it will remain at its original position.

*2) **Mesh to SDF conversion**:* The previous approach utilizes mathematical functions to represent objects, with input points yielding signed distances as outputs. However, the constraints of this approach became apparent due to the following reasons:

i) **Complex Shape Representation**: Representing real-life objects with intricate and complex shapes using functional forms proved to be a challenging task. The complexity of their geometry made it difficult to find an accurate functional representation. Although techniques like Constructive Solid Geometry can be employed, they are time-consuming and not always feasible.
ii) **Limitation in Object Selection**: This approach inherently restricts us from utilizing objects with available mesh representations. In scenarios where mesh data is readily accessible, the limitations of a purely SDF-based approach become evident.

In response to these limitations, our quest for an improved approach led us to Kaolin, a powerful tool that allows us to derive the signed distance function from mesh representations. This advancement revolutionizes our approach, as outlined below:

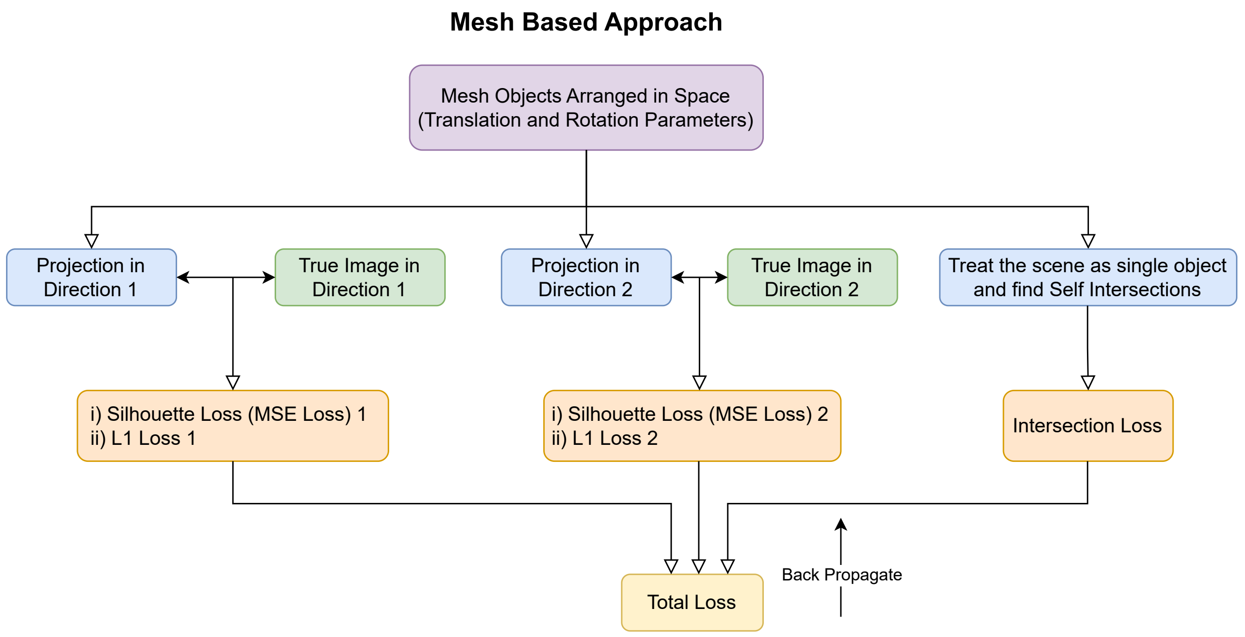
In the revised approach, we adopt a mesh-based representation, which provides remarkable advantages. We follow these steps to compute the signed distances for each object at its current location:

i) **Mesh Transformation**: Initially, we start with a mesh object, which is brought to the origin. This allows us to efficiently rotate the object about the x, y, and z axes.
ii) **Translation to Current Position**: Following rotation, we translate the object to its specific position within the scene, allowing us to explore various arrangements effectively.
iii) **Bounding Box Sampling**: We then proceed to select all the points residing within the object's bounding box. The signing process is achieved using Kaolin's capabilities, which efficiently sample the mesh, providing us with the signed distances for each object.

Subsequently, we project the scene in two distinct directions. The comparison with the respective ground truth image in each direction yields the following key metrics:

For Direction 1:

i) Silhouette Loss (MSE Loss) 1: A metric assessing the match between the projected silhouette and the ground truth in Direction 1.



Mesh Based Approach
Mesh Objects Arranged in Space (Translation and Rotation Parameters)
Projection in Direction 1
True Image in Direction 1
Projection in Direction 2
True Image in Direction 2
Treat the scene as single object and find Self Intersections
i) Silhouette Loss (MSE Loss) 1 ii) L1 Loss 1
i) Silhouette Loss (MSE Loss) 2 ii) L1 Loss 2
Intersection Loss
Total Loss
Back Propagate

Fig. 6: Mesh based approach for Anamorphic Art.

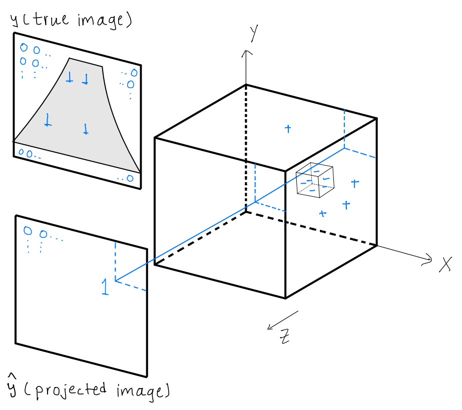


Fig. 7: Projection of union of shapes represented using signed distance functions

ii) L1 Loss 1: Evaluates the L1 discrepancy between the projection and the ground truth image.

For Direction 2:

i) Silhouette Loss (MSE Loss) 2: A measure of the silhouette consistency between the projected image and the ground truth in Direction 2.

ii) L1 Loss 2: Examines the L1 divergence between the projection and the corresponding ground truth image in Direction 2.

Moreover, we explore the intersection between pairs of objects, as previously detailed in the initial report. This intersection analysis leads to the derivation of the intersection loss.

To assess the overall quality and adherence to constraints, we aggregate all the individual losses to compute the total loss. This comprehensive loss function encompasses the following components:

$$\begin{aligned} TotalLoss = (SilhouetteLoss1) + (L1Loss1) \\ + (SilhouetteLoss2) + (L1Loss2) \\ + (IntersectionLoss) \end{aligned} \tag{6}$$

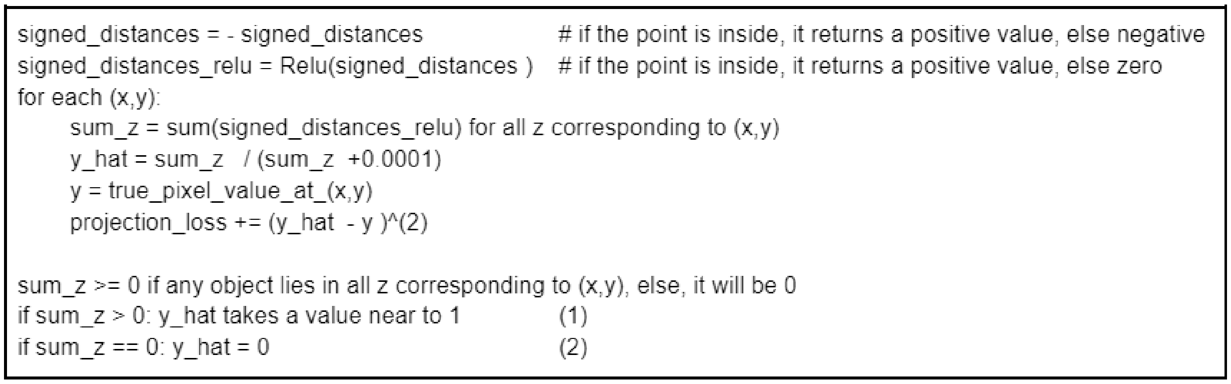
These calculated losses play a critical role in the optimization process, guiding us in the pursuit of the most effective and visually compelling object arrangements. The total loss serves as a key metric to evaluate the success of our computational framework and the alignment of projected images with our desired visual outcomes.

This refined approach, harnessing both SDF and mesh representations, empowers us to overcome the previous constraints, explore diverse object arrangements, and fine-tune our results for remarkable visual impact. It represents a significant step forward in our ongoing quest to create dynamic and engaging visual experiences.

In this approach, the whole pipeline is created by us, including the rendering of the scene.

B. Mesh-Based Approach

In the initial approach, our focus primarily revolved around sampling meshes to extract signed distances, which were subsequently used to compute losses. This strategy was employed due to the inherent challenges in accurately identifying intersections between objects when dealing with mesh representations. In such cases, the results of intersection analysis were not optimal. Hence, we relied on Signed Distance Functions (SDFs) to calculate the intersection loss.



```
signed_distances = - signed_distances             # if the point is inside, it returns a positive value, else negative
signed_distances_relu = Relu(signed_distances )   # if the point is inside, it returns a positive value, else zero
for each (x,y):
    sum_z = sum(signed_distances_relu) for all z corresponding to (x,y)
    y_hat = sum_z  / (sum_z +0.0001)
    y = true_pixel_value_at_(x,y)
    projection_loss += (y_hat - y )^(2)

sum_z >= 0 if any object lies in all z corresponding to (x,y), else, it will be 0
if sum_z > 0: y_hat takes a value near to 1       (1)
if sum_z == 0: y_hat = 0                          (2)
```

Fig. 8: Algorithm that solved the problem of zero gradients for projection loss

However, our journey toward optimization led us to a remarkable repository known as ”torch-mesh-isect.” This repository offers a groundbreaking solution for identifying self-intersections within an object, providing a valuable addition to our computational toolkit. Our strategy now entails utilizing ”torch-mesh-isect” to determine the intersection loss among mesh-based objects.

The innovative approach involves treating the entire scene as a unified object. By utilizing ”torch-mesh-isect” to assess self-intersections within the composite scene, we can efficiently derive the intersection loss. This shift in perspective, treating the entire scene as a single entity, introduces a fresh dimension to our computational framework.

The subsequent steps of our approach remain consistent with those described in the previous section. This includes the critical processes of:

1. **Projection Along Two Directions**: We continue to project the scene along two distinct directions.
2. **Comparison with Ground Truth Images**: The resulting projections are meticulously compared with the ground truth images corresponding to each direction.
3. **Loss Calculation**: The outcomes of these comparisons are utilized to calculate various losses associated with the projections.
4. **Total Loss**: These individual losses are thoughtfully combined to derive the total loss, a comprehensive metric guiding the optimization process.

The introduction of the mesh-based approach, especially with the incorporation of ”torch-mesh-isect,” enhances our ability to accurately assess and address intersection concerns among mesh-based objects. By treating the entire scene as a holistic entity, we are better equipped to derive the intersection loss and advance toward our primary goal of achieving compelling visual outcomes. The innovation represents a significant stride in our ongoing mission to create visually engaging and impactful content.

VI. Results

A. *Signed Distance Functions (SDF) based Approach*

*1) **Functional Form of SDF:*** 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.9.

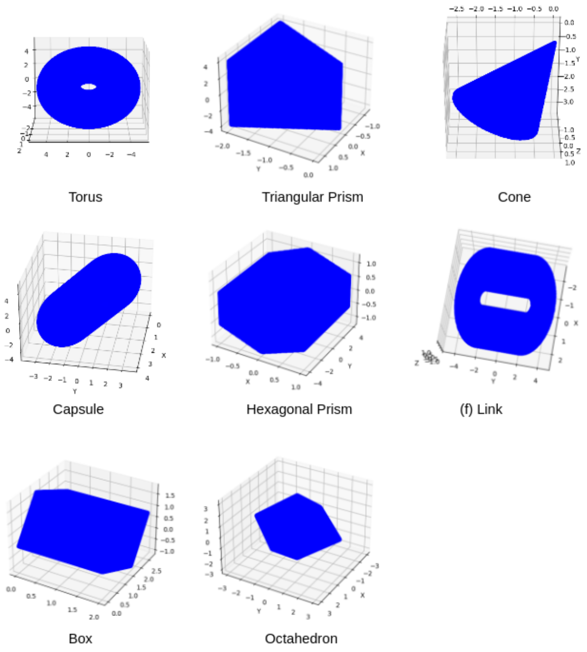


Fig. 9: Dataset prepared using the functional definition of SDFs

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

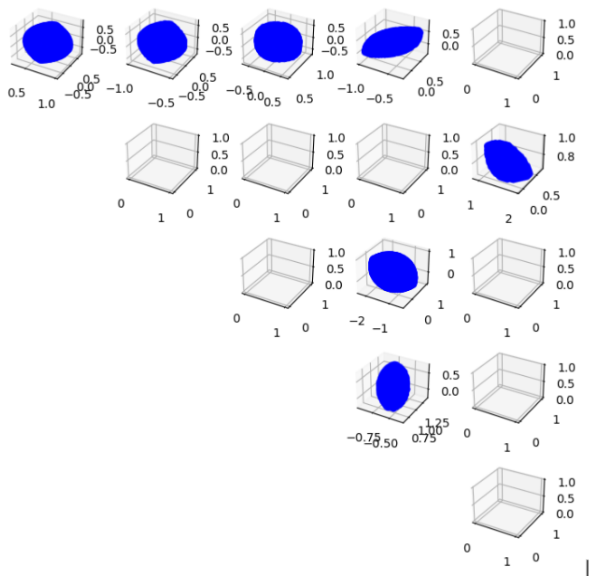


Fig. 10: Intersection loss of each pair of shapes before back-propagation

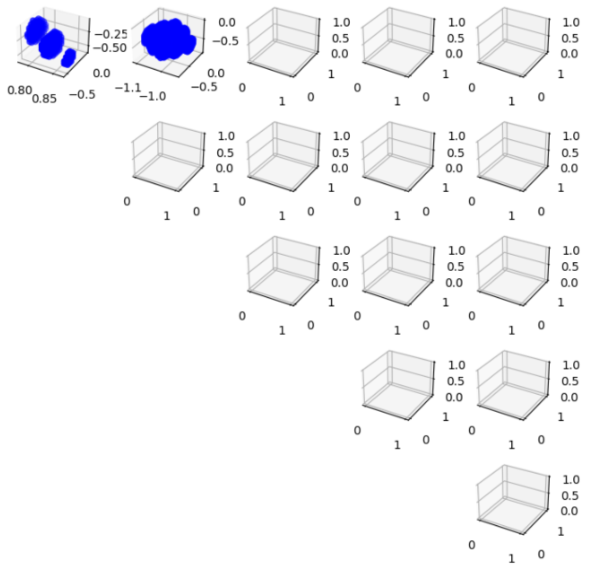


Fig. 11: Intersection loss of each pair of shapes after back-propagation

pair of shapes (Fig.10). In Fig.10, 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.12 (a) shows true image and fig.12 (b) 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 loss was reduced. Fig.11 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.12 (c) 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 Fig.12 (a), 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.

2) ***Mesh to SDF conversion:*** In this case, we have incorporated the rotation and translation pipeline in the DeepSDF neural network implementation. Fig.13 shows the results of the same.

B. *Mesh-Based Approach*

To build upon previous research efforts, we revisit an earlier attempt that involved arranging objects in 3D space to achieve a desired shape. In the illustrative example depicted in Figure 14, the objective was to recreate the images of ducks captured from two distinct viewpoints, as showcased in (a) and (b). These images serve as our ground truth references.

In the initial approach, rigid objects were randomly distributed within a bounding box. However, due to the stochastic nature of the initialization process, the resulting scene did not resemble any coherent or meaningful shapes, as depicted in image (c) of Figure 14. This randomness introduced a significant challenge in generating an arrangement that closely resembled the desired duck shape.

To address this issue, a mesh-based approach was employed to transform the chaotic scene into a structured shape resembling a duck, as illustrated in (d) of Figure 14. While this approach led to a more organized arrangement, it introduced a significant drawback: objects within the scene began to collide with each other, resulting in non-negligible intersection volumes. Such behavior was contrary to our initial assumption that the objects were rigid and, in reality, rigid objects do not intersect.

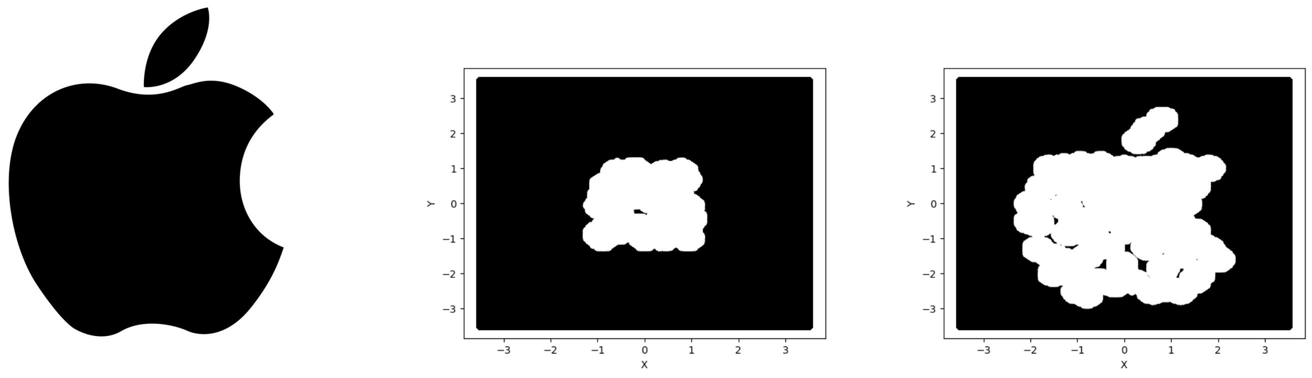
To rectify this limitation, we have embarked on a journey of refinement and optimization. We are now enhancing the approach by introducing an intersection loss component to the optimization process. As previously discussed, the intersection loss is evaluated using a specialized codebase provided by the "torch-mesh-isect" repository.

This significant modification addresses the concern of object intersections, aligning our computational framework more closely with real-world expectations. Our aspiration is to ensure that rigid objects do not intersect within the constructed scenes, thereby achieving a higher level of fidelity and accuracy in our visual representations. By enhancing the approach in this manner, we aim to make a valuable contribution to the field of computer graphics and computational geometry.

VII. Conclusion

In summary, our research involved the development of a comprehensive pipeline that leverages two distinct approaches to address the challenge of arranging rigid objects within a 3D space to create meaningful scenes.

Approach 1: Sampling Mesh Objects Using Kaolin



(a) True image (b) Projection of objects at initial positions (c) Projection of objects at final positions

Fig. 12: Results generated using SDF based approach with functional form.

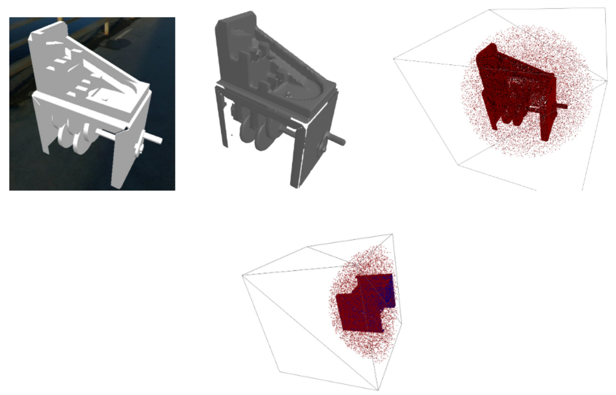


Fig. 13: Input Object : Mesh Based Representation : SDF Representation : R and T of SDF Representation

In the first approach, we used mesh objects and the Kaolin toolkit to efficiently sample and represent 3D objects. We started by obtaining mesh objects and ensuring they were properly positioned for rotations and translations. The main goal was to find the optimal translation and rotation parameters for each object to minimize overlap and achieve the desired scene.

We calculated intersection and projection losses to measure the dissimilarity between our rendered scenes and ground truth images. Backpropagation techniques were then applied to update the translation and rotation parameters, optimizing the arrangement of objects.

Approach 2: Using Mesh-Based Representation

The second approach revolved around mesh-based representations of objects. We introduced the "torch-mesh-isect" repository to detect and mitigate self-intersections among the meshes. This was crucial to ensure that the objects in our 3D space did not intersect.

We continued to calculate projection losses and intersection loss, which were combined to form a total loss metric. Backpropagation techniques were once again applied to refine the translation and rotation parameters for each object.

In both approaches, our primary objective was to create 3D scenes that faithfully represented the desired scenes while adhering to the principles of non-intersecting rigid bodies.

These two approaches together formed the core of our research, offering a multifaceted strategy to address the challenges of generating coherent 3D scenes from rigid objects.

VIII. Future Work

As we move forward, we have solidified our approach and are prepared to proceed with implementation to obtain results. Our initial testing of Kaolin for sampling signed distances from meshes has been promising. We intend to integrate this method with our existing pipeline, which originally relied on mathematical functions to derive signed distances. This integration will further enhance our pipeline's capability.

In addition to this, we plan to introduce the concept of intersection loss to our mesh-based approach. This will be a crucial step in ensuring that the final pipeline aligns with our goals and yields the desired results.

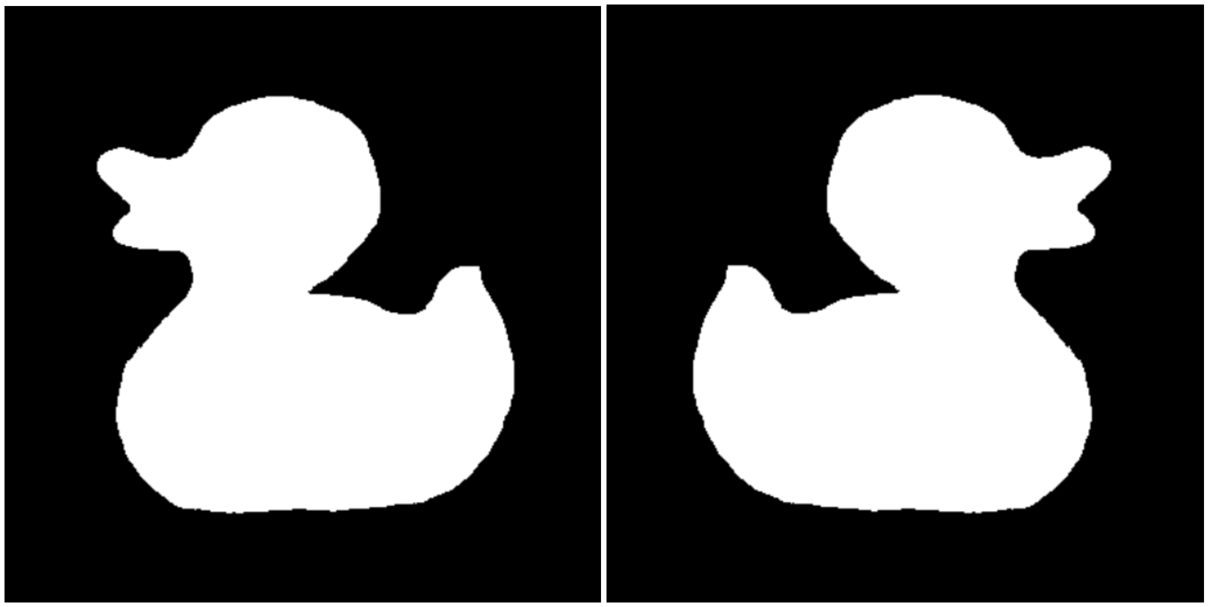
In summary, our future work primarily involves the practical application of our refined approaches, including the incorporation of Kaolin's mesh sampling and the introduction of intersection loss into our pipeline.

IX. 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.

X. References

1. Park, Jeong Joon, et al. "Deepsdf: Learning continuous signed distance functions for shape representation."


(a) True image seen from direction 1 (b) True image seen from direction 2

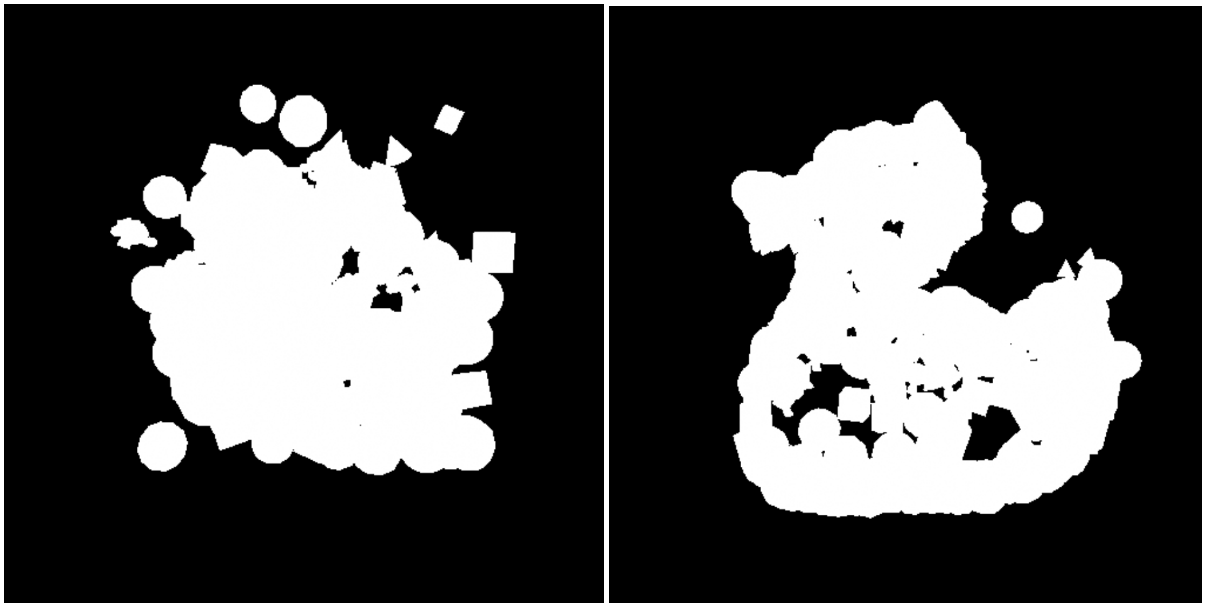

(c) Projection of objects at initial position (d) Projection of objects at final position

Fig. 14: The figure shows arranging the 3D objects in space to achieve the shape of duck. (a) and (b) shows the images of the duck from two different directions. (c) shows the initial position of the objects in 3D space. These objects are then translated and rotated using the mesh based approach without incorporating intersection loss to get the final image shown in (d). It can be seen that the objects are intersecting with each other. This is not desirable because the rigid objects can not intersect each other.

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