Anamorphic Art

Dhruv Darda
IIT Gandhinagar, India
dhruv.d@iitgn.ac.in

Prateek Jha
IIT Gandhinagar, India
prateek.kj@iitgn.ac.in

Abstract

This project, inspired by packing optimization methods, dives into art and aesthetics to make anamorphic art from 3D objects. We extend earlier packing models by constructing the structure without any ground truth. The goal of this project is to reduce the number of intersections between the arranged objects to zero. We assessed our model's performance on image-based losses as well as the number of object crossings. Experimenting with different parameters results in fewer intersections while keeping optimal visual structure. Further work to eradicate these intersections is in progress.

2012 ACM Subject Classification Anamorphic Art → Intersection reduction

Keywords and phrases SDF, Differentiable Rendering, 3D Intersections, SSIM

1 Introduction

Arts and aesthetics have been an ambitious playground for computer vision models. This project aims to produce anamorphic art from a given set of randomly selected 3D objects. Anamorphic art is arranging random objects in 3D space to create the desired image when being looked at from a certain angle. This project tries to achieve the same in 2D projections or shadows. We take 3D objects from our store randomly and arrange them to form a 2D projection of our desired object.

The initial model was able to form good projections; however, there were several intersections between the objects. In real-world situations, we cannot have any two objects fusing. Hence, we started working on removing these intersections while preserving the image structure simultaneously. Our algorithm does not have any ground truth for the final 3D arrangement; instead, it uses the target 2D projection to achieve the arrangement of 3D objects.

We took an experimental approach and played with multiple factors to figure out the optimal combination for a desirable image with none of its constituent objects fusing. However, achieving zero intersections is much more challenging than previously thought. Achieving a balance between preserving the structure of the image and having minimum intersections has not been easy. The current results, while being satisfactory, require further experimentation and study to fully understand the mechanisms of creating anamorphic art.

2 Previous Work

We built on a model that could generate shadow art from a random set of 3D objects, but there were no restrictions on the intersections between the items. The model could create projection images that matched the target image, but many intersections were missed in the process. The image created by the model for the target image is displayed below.

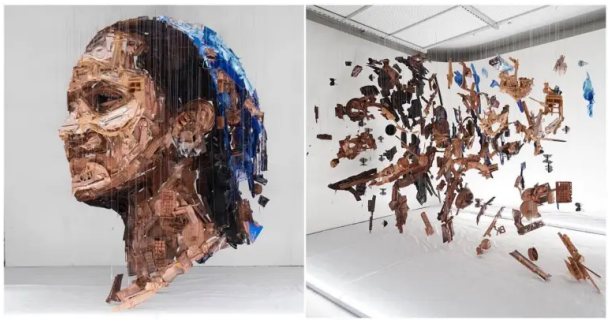


Figure 1 Example use of Anamorphic Art

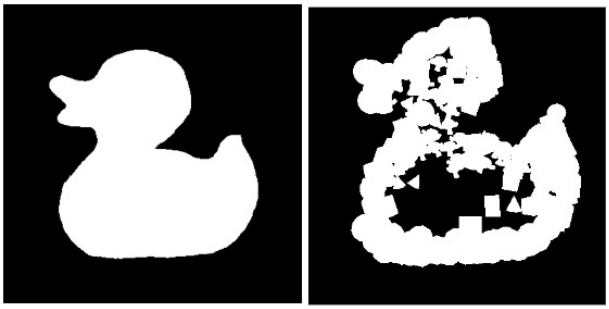


Figure 2 Model rendered image vs Target Image

3 Methodology

To reduce the intersections in the arrangement, we added an intersection loss that penalizes the model for arrangements that cause intersections between objects. We count the number of intersection pairs, and the objective is to reduce these intersections. To the existing cost function that was only based on image pixel loss, we add this intersection loss. So, while the intersection loss would try to arrange the objects in such a way that there are minimum intersections, the image loss will try to bring them together in an arrangement that renders the desired target image on rendering.

Since the objects can be of any shape, for ease of operation, we enclosed each object in a sphere such that the centers of the object and the sphere are the same, and the radius of the sphere is equal to the maximum distance from the centroid to the surface of the object. Now the transformations were applied to the objects, but for checking the intersections, we considered the outline spheres.

The updation formula for the homogeneous transformation of the mesh's vertices is:

$$R(X_i - C) + T$$

where C is the centroid of the object mesh vertices. The outline sphere equation is given by:

$$(X - X_c)^2 + (Y - Y_c)^2 + (Z - Z_c)^2 - R^2 = 0$$

A pair of spheres are in the intersection if the distance between their centers is less than equal to the sum of their radii.

$$||c_i - c_j|| <= r_i + r_j$$

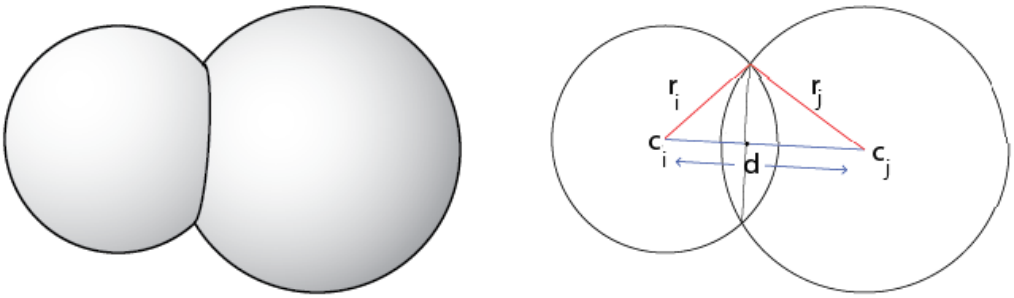


Figure 3 Intersection of two spheres

Using the above technique, we can get to know the number of pairs of intersections. We deal with pairs of intersections because we do not have the 3D arrangement ground truth available to us, so we will want to move the objects such that the structure does not get changed much while trying to reduce the intersections. To penalize the model based on the number of intersections, we added the intersection loss function to the cost function that we want to optimize. The loss function is as follows:

3.1 Intersection loss

To get the loss, we want the position of objects such that there are no intersections, and the image is also formed when rendered. Since we do not have the spatial arrangement ground truth, we sequentially first get the arrangement of the objects using image-based loss without constraining the objects on intersection, then translate the pairs of objects having intersections symmetrically in the direction of the maximum gradient of SDF to a position where there are no intersections. The positions that we get from these translations tend to disturb the structure of the image. To deal with this, we added SSIM loss while generating the first set of arrangements, which try to arrange the objects in a way that the gradient of the SDF is in the direction of the structure of the image.

We achieved the above objective as follows: For each object, $object_i$, we find all the intersection pairs:

$$object_i - object_j \ni i \neq j$$

We move each object equally in the direction of the gradient of their SDF; for spheres, this means that we move each pair of objects equally by half of the intersection magnitude in the direction of the vector joining their centers away from each other.

```
for i in all_objects:
        move_object_i_vector_xyz  = [0; 0; 0];
        for j in all_objects_except_i:
                move_object_i_vector_xyz +=
                      intersection_b/w_object_i_&_object_j / 2;
        object_i += move_object_i_vector_xyz;
```

We also add a bound loss that we integrate with the intersection loss that tries to contain the objects within the bounding box so that the objects do not go out of the bounding box to get zero intersections.

4 Experiments

The model is dependent on many parameters and hyper-parameters. We fixed some parameters after doing certain number of experiments which resulted in the best image. The following parameters were fixed for all the experiment results shown below:

1. Input image size was fixed to be 512 × 512 pixels,
2. Number of objects were fixed to be 200,
3. Box dimension was fixed to be 128 × 128 × 128,
4. Image loss weights were fixed to 10.

4.1 Intersection weight

- Firstly, we decided to estimate the effect of our intersection loss on the formed projections.
- Initial trials revealed the intersection loss to be pretty dominant as compared to previously configured image losses.
- We then lowered its weight and performed multiple trials; the results are shown below.

The table showing effect of intersection weights on number of intersections is shown in Figure 4.

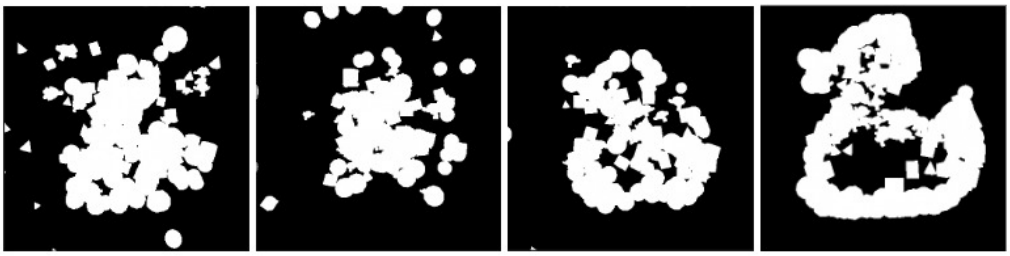


Table Fig. 1 Table Fig. 2 Table Fig. 3 Table Fig. 4

Intersections				
Intersection loss weight→ / **Iterations↓**	**1e-4** (Table Fig 1)	**1e-5** (Table Fig 2)	**1e-6** (Table Fig 3)	**Without intersection loss** (Table Fig 4)
300	400	1200	1200	1620
700	830	1100	1550	
1000	750	-	1760	

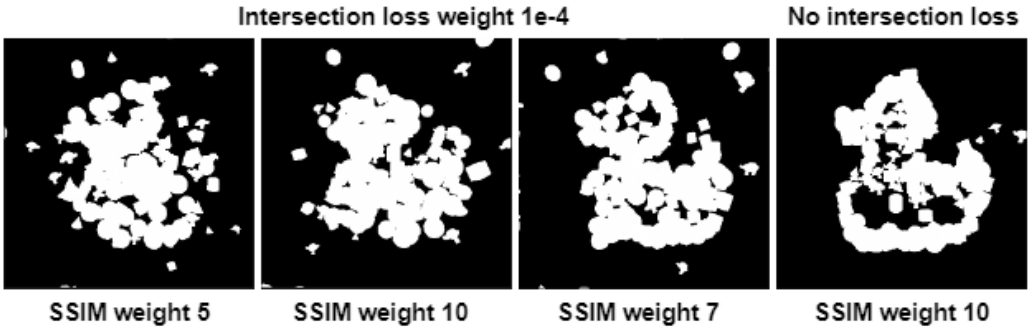
Figure 4 Effect of Intersection weights on number of Intersections

4.2 SSIM Addition

- As observed in our previous set of experiments, giving more weight to our intersection loss significantly reduced our projection's quality.
- To compensate for this loss in the projection's structure, we introduced SSIM loss.
- Initially, we fixed out intersection loss weight as 1e-4 and varied SSIM's weight.

- We then tried increasing the intersection loss weight to 1e-3, but the outputs were not desirable.

The table showing the effect of SSIM along with different intersection loss weights is shown in Figure 5.



Intersections							
Intersection loss weight→	1e-4			1e-3			Without intersection loss
ssim loss weight→	5	10	7	5	10	20	1400
Iterations↓							
400	300	400	850	300	340	-	
700	830	900	860	-	700	680	
1000	750	-	-	-	660	-	

Figure 5 Effect of ssim with intersection weights on number of Intersections

4.3 Optimizing Number of Objects

- Lastly, we believed that the objects were going out of bounds in most of the previous trials.
- To verify this theory, we reduced the number of objects to 140, and ran with the parameters that resulted in best results (without SSIM).
- The output produced confirmed our hypothesis. The intersections were reduced significantly and the projection was comaparable to the one with 200 objects.
- This adds to our future work of optimizing the number of objects required for every projection.

5 Results

We have been able to reduce the number of intersections to half of the original number, while maintaining similar desirability of produced projection. The best projection produced by our

model is shown in Figure 6. This has both intersection loss and SSIM loss along with MSE and L1 based image losses (from the original model), using 200 obejcts as specified earlier.

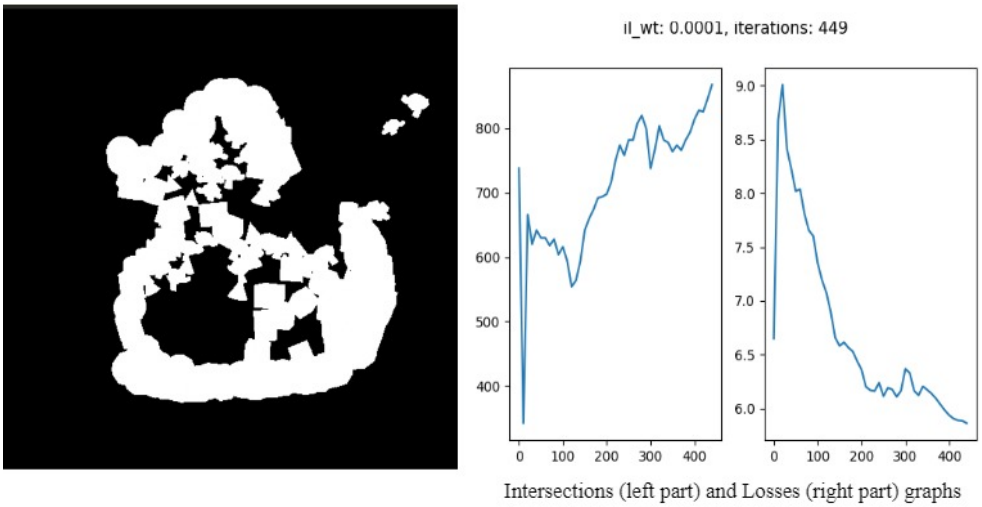


Figure 6 Best results with intersection and ssim losses

6 Limitations and Future Work

6.1 Limitations

- We considered pixel based loss and intersection loss to be independent.
- Unpredictable dominance of various loss factors.
- Objects are assumed to not to be elongated

6.2 Future Work

- Query points on the objects themselves instead of the outline spheres and use SDF on those query points to get the intersections.
- Use sampling techniques to query points for intersections of objects.
- Allow the model to determine the number of objects needed for any given target image on its own.
- Adding textures to the produced images.

References

[1] Miaojun Yao Zhili Chen Linjie Luo Rui Wang Huamin Wang. Level-Set-Based Partitioning and Packing Optimization of a Printable Model

[2] Tero Karras. Maximizing Parallelism in the Construction of BVHs, Octrees, and k-d Trees