# Determining viewing points for catoptric anamorphoses

**Casper Henkes**, **Baran Usta**, **Elmar Eisemann**
TU Delft

## Abstract

Finding good viewpoints for catoptric anamorphoses by hand is hard. However, it should be possible to find the optimal viewpoint using just the specifications of the mirror. This problem is solved by first generating a set of candidate viewpoints using the specifications of the mirror. Then all candidates are ranked based on metrics like visible surface area. The optimal is then easily found.

## 1 Introduction

Catoptric anamorphosis a form of anamorphosis that uses a reflective object and a surface with art on it. The intention is that viewing the art from a specific viewpoint shows a second image in the mirror object that is a reflection of the surface. These catoptric anamorphoses are found in multiple fields, from art to architecture.

Mirror anamorphoses have multiple aspects. They consist of a surface, an image on that surface, a reflective object, and a viewpoint. Finding the best viewpoint for a given scene can be difficult as, depending on the metric you use, any perspective can work. This paper aims to define a method that constructs the optimal viewpoint given a surface, a reflective object, and some metric to optimize for. Take any mirror shape and some surface with coloured dots on it. Now finding a viewpoint for which the coloured dots cover the maximum surface area is hard. This is unfeasible to do by hand and thus a computer-aided process is preferable.

The paper discusses topics related to image quality, optimization, and viewpoints. Several metrics for the image quality of catoptric anamorphoses and the benefits and drawbacks of each of them are discussed. These metrics then allow comparisons of viewpoints which leads to finding the most optimal one.

Given a representation of catoptric anamorphoses, how do you find an optimal viewpoint? To generate an optimal result it is required to have good metrics and optimizations. For the image quality, a method is implemented to calculate the apparent quality of a scene from a certain viewpoint which allows for direct comparison of multiple viewpoints. The metrics are also discussed. What makes a good metric is for mirror anamorphoses and can combining multiple metrics can lead to better result?

The methods proposed here makes a few assumptions about the scenes. First of all, it assumes that there is only one mirror object present. It also assumes that a surface is a plane. Furthermore, it is assumed that the wanted viewpoints have to lie outside of the reflective object.

### 1.1 Related work

Methods exist to construct catoptric anamorphoses digitally. This is preferred for more complex shapes as doing all required calculations by hand is not feasible. Francesco proposes one such method. This method requires the user to provide the surface, object, and viewpoint. Then it uses ray-tracing software to construct the image [1]. For a cylindrical mirror, interactive applications exist that implement anamorphosis. The tablet shows an image such that its reflection in the mirror displays the desired form. The mirror can then be rotated to change the image seen on the surface and the mirror. [5].

It is difficult to provide a good viewpoint for the catoptric anamorphoses in some cases. In those cases, finding such a point is required before the technique can be used successfully.

Another aspect of the method proposed in this paper is determining the quality of a given viewpoint. Prior research has studied techniques for determining image quality assessment. The book Modern Image Quality Assessment [4] gives a broad overview of these techniques. These techniques take different input compared to the method proposed in this paper and thus cannot be used directly.

The initial viewpoint generation problem is similar to a problem in visual sensor networks. The goal is to get maximum coverage using the minimal amount of camera's [3]. The problem in this requires optimally placing a single camera and in that way differs from this previous research.

The methods used here implement a small ray tracer to test the reflection and the visible surfaces. These methods could have used some of the known acceleration structures that can be used for optimizing ray tracing like bounding volumes. A lot of work has been done in this field and a small summary of a few techniques can be read in this report from 2005 [2].

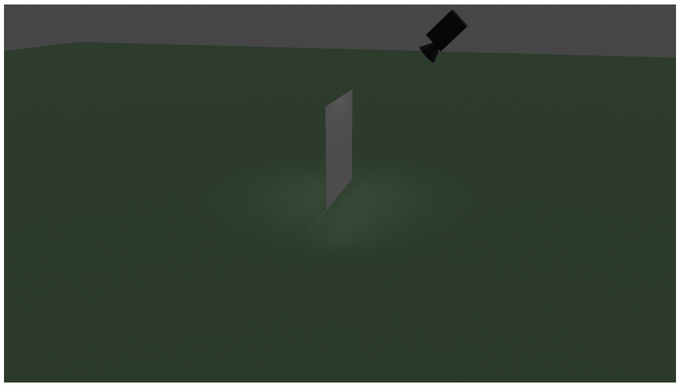
Figure 1: Side view of a scene

Figure 2: faces method used to find this viewpoint for the metric total area

There is also previous research done on candidate viewpoint selection which is very similar to one of the optimization challenges in this paper. This paper describes a new algorithm for this and compares it to other known algorithms [6].

## 2 Pipeline

The problem the paper is tackling is finding the optimal viewpoint for mirror anamorphoses. Finding these viewpoints is done using: the reflective object, the surface, and metrics. The *object*, a mesh representing the mirror in the images depicted in grey, is used to deform the image that resides on the surface, a *plane* in the images depicted as green, such that the reflected image is as desired. The surface can be anything from a plane to some complex shape. The *metrics* mentioned here are used to assign numerical values to viewpoints enabling comparison between them.

This section describes all components of the pipeline used in the method that computes the optimal viewpoint for catoptric anamorphosis. This pipeline works as follows. First of all the scene file, a side view of which can be seen in figure 1, is loaded. Then preprocessing is done on the object to find the initial set of potential viewpoints. After that, the metrics are used to determine the score of each of those viewpoints and then the best viewpoint is selected based on that score.

### 2.1 Generating initial viewpoints

This section shows the implementation of the algorithm used for generating initial viewpoints. This section also contains some analysis done on the discussed algorithms, the methods are then further compared in the results section. A good implementation would be able to find the optimal viewpoint using as few potential viewpoints as possible. To be able to make fairer comparisons all methods worked with the same *maximum distance*. This distance set to be three times the longest distance between any two vertexes of the object.

The most basic implementation of generating these viewpoints is a *brute-force* approach. This approach works by filling a sphere around the object with points. Start by creating a bounding box around the object. The center of this sphere is the *center of the object*, calculated by taking the average of all vertexes, and the radius is half the maximum distance. Then the method creates a uniform grid of points in a square around the center of the object such that an edge of the square is two times the radius of the sphere. Then add all points for which the distance from it to the center of the circle is smaller than the radius. Some optimizations can be done for this method. Note that all viewpoints close to the object can be removed. First of all the viewpoints that are within the object itself can be removed as we assumed that the optimal viewpoint would lie outside of the reflective object. And for other points close to the object another assumption made is that it is better to use more of the mirror. It is impossible to view the entire object from a close enough viewpoint and those points can thus also be removed.

The previous approach does not in any way use the object itself to determine initial viewpoints. It should be possible to reduce the number of points generated by taking the object itself into account. The first idea is to design a method, from now on called *faces* that should work well for a plane and then try to apply that for objects consisting of more *faces*. A face here means a collection of vertexes and edges forming a flat surface. For each face generate a viewpoint where the origin of the viewpoint is found by adding some linear combination of the face normal and the surface normal to the center of the face. A realization of this method for the plane can be seen in figure 2. The idea behind this method is that such a viewpoint is often used for existing artworks using a plane mirror and even for cylindrical mirrors. No single way of making this linear combination may exist such that the viewpoints for both the plane mirror and the cylindrical mirror are optimal.

Another idea for generating viewpoints is to generate viewpoints in front of the vertexes of the object, from now on called *vertexes*. to find the origin of the viewpoint shoot a ray towards each vertex and then scale that ray such that the distance is equal to the maximum distance. The assumption is that the vertexes being situated along the edges of the object could give good indications of where the object changes and might thus be interesting to include in the view. A realization of this method for the cube object can be seen in figure 3

The last method, from now on called *hsphere* generates points in front of each face that form the surface of a sphere.

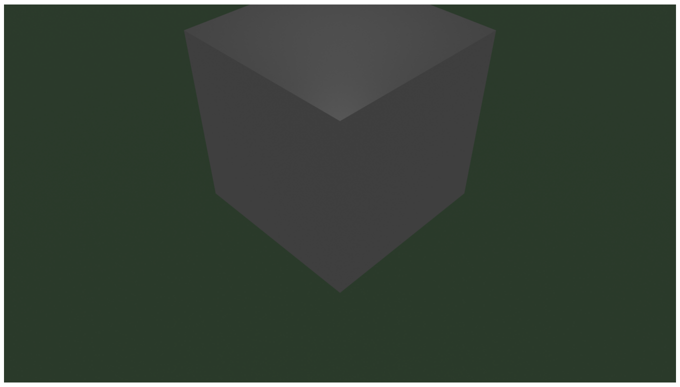
Figure 3: vertexes method used to find this viewpoint for the metric total area

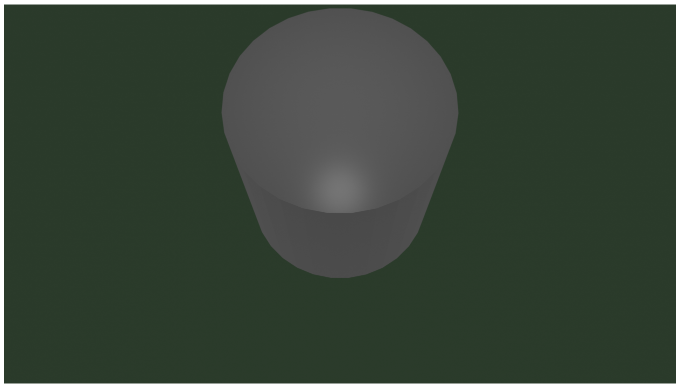
Figure 4: hspere method used to find this viewpoint for the metric total area

The assumption here is that the faces are interesting to look at. However, it is unknown from what angle it is best viewed. By generating points on the surface of a sphere gives a good spread of points for the face while still generating fewer points than the brute-force method. A realization of this method can be seen in figure 4

### 2.2 Metrics

This section describes all metrics implemented in the method and explains the reasons to use them. For all following calculations, the algorithm takes into account some camera field of view in form of a camera angle. First of all, a point in the reflective object is selected as the center of the field of view. Then the angle between the ray to check and the ray from the viewpoint to the target point is calculated. If this angle is smaller than half of the camera angle then it is counted as being within the field of view.

First of the algorithm can calculate the faces that are visible from a viewpoint. Determining the visible faces is done as follows. For each face shoot a ray at its center and calculate if that ray intersects with any other face before hitting the target face. If the ray does not hit anything else then the algorithm counts that face as visible. This does have its limitations, however. Take a plane and lay a small ball on top of

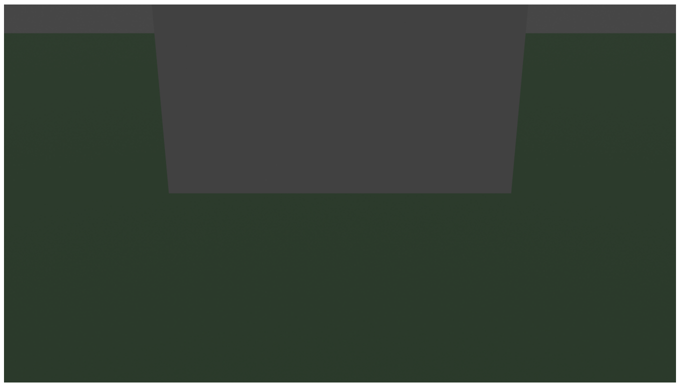
Figure 5: the plane object optimized for the visible object area

the center of the plane. Now the algorithm thinks the plane cannot be seen, however, most of it is visible. An example of this problem can be seen in figure 4 where only about half of the plane is within the field of view. However, the algorithm still counts the entire plane as visible as its center is within the field of view. This metric is mostly used in further calculations but can be used to prune viewpoints from which the object cannot be seen as those are uninteresting.

The next metric is the visible *object area*. To calculate this the area of all visible faces is added. The total area of a given face is calculated by triangulating the face and calculating the area of all triangles and adding those together. This total area is then projected based on the viewing angle. The visible surface area is used to determine how much of the mirror is visible from a given viewpoint. A viewpoint for which the object area is optimized can be seen in figure 5

After that is the visible *surface area*. Before the area is calculated, it is necessary to find where the reflection of the face on the surface is. Finding the reflected vertexes is done by shooting rays at the vertexes of the object face. Those rays are then reflected using the vertexes as reflection point and the face to get normal. Then the algorithm calculates the intersection point with the surface for that reflected ray. Using those vertexes on the surface a new face is created. After this, the area can be calculated using the method used by the visible object area. The visible surface area is used to determine how much of the mirror is visible from a given viewpoint. A viewpoint for which this is optimized can be seen in figure 6

Another metric is the *resolution*. If there is a big difference between the visible object and surface area then there are a lot of pixels on one of the areas that map to very few pixels on the other. These metrics attempt to solve that issue by comparing the two areas. it does this by dividing the smaller of the two areas by the other area. This metric can be used to limit the difference in the area.

The last metric is a combination of previous metrics, from now on called *total area*. It is calculated by adding the visible object and surface area and multiplying it by the resolution. It also prunes viewpoints for which one of the areas is zero. This metric tries to solve the problems that previous metrics

Figure 6: the plane object optimized for the visible surface area

had. The object area by itself does not say anything about the reflection. The surface area is better, but it does not say anything about the area of the object that is in use. The resolution by itself does not tell anything about the areas used possible allowing both to be very small which can be unwanted. This metric eliminates those issues as it ensures a non-zero area for both, and including the resolution ensures some balance in the size differences of the 2 areas.

## 3 Results

This section shows the results that the different generation methods create. The results are structured per scene starting with the simplest scene, the scene with the least faces, and increasing in complexity. It briefly discusses each scene giving some reasons for the performance of the different viewpoint generation methods. It is good to note that not all faces and vertexes will generate viewpoints. If a viewpoint would end up below the surface it is not added as the object would not be visible. Under here is defined as on the other side of the surface as the object.

Table 1: Plane object

| Plane | | score | amount | origin | equal |
|---|---|---|---|---|---|
| faces | object area | 3.2549338 | 1 | (0.0,-3.5,3.500001) | 1 |
| | surface area | 8.854383 | 1 | (0.0,-3.5,3.500001) | 1 |
| | total area | 4.451471 | 1 | (0.0,-3.5,3.500001) | 1 |
| hsphere | object area | 4 | 104 | (0.0,-6.0,1.000001) | 1 |
| | surface area | 7.3664336 | 104 | (0.5,-5.0,4.250001) | 2 |
| | total area | 4.858289 | 104 | (-0.5,-5.0,4.250001) | 2 |
| vertexes | object area | 0 | 2 | (4.243,-0.0,5.243) | 2 |
| | surface area | 0 | 2 | (4.243,-0.0,5.243) | 2 |
| | total area | 0 | 2 | (4.243,-0.0,5.243) | 2 |
| bruteforce | object area | 4 | 38964 | (0.0,-3.25,1.000001) | 12 |
| | surface area | 13.891902 | 38964 | (0.0,-6.0,3.500001) | 1 |
| | total area | 5.044677 | 38964 | (0.0,-5.75,4.750001) | 1 |

In the plane scene consists of one face, and four vertexes. The faces method performs relatively well which is expected as it was partly optimized for this scene. An interesting fact is that the vertexes method does not work for this scene. This is because both potential viewpoints are generated right above the plane. All rays shot from those viewpoints towards the face can only be parallel to it and thus the face is not visible from those viewpoints.

Table 2: Cube object

| Cube | | score | tested | origin | equal |
|---|---|---|---|---|---|
| faces | object area | 4.519845 | 4 | (0.0,-4.5,3.5) | 4 |
| | surface area | 20.660212 | 4 | (0.0,-4.5,3.5) | 4 |
| | total area | 5.5086536 | 4 | (0.0,-4.5,3.5) | 4 |
| hsphere | object area | 5.2061205 | 591 | (-3.5,2.75,4.25) | 24 |
| | surface area | 14.310835 | 591 | (0.0,-5.0,4.0) | 4 |
| | total area | 9.73931 | 591 | (-4.5,-2.75,3.25) | 4 |
| vertexes | object area | 5.392128 | 4 | (3.464,3.464,4.464) | 4 |
| | surface area | 9.561445 | 4 | (3.464,3.464,4.464) | 4 |
| | total area | 8.432991 | 4 | (3.4646,3.464,4.464) | 4 |
| bruteforce | object area | 5.588843 | 38964 | (-4.0,4.0,4.75) | 12 |
| | surface area | 21.263098 | 38964 | (-6.0,-0.0,3.75) | 4 |
| | total area | 10.880931 | 38964 | (-3.25,3.25,6.0) | 4 |

The cube, scene consists of six faces and eight vertexes. The faces method can find a rather good viewpoint for the surface area. Other than that both the vertexes as well as the hsphere methods get close to the brute-force solution for the visible object area.

Table 3: Cylinder object

| Cylinder | | score | tested | origin | equal |
|---|---|---|---|---|---|
| faces | object area | 3.7096868 | 32 | (0.441,-9.474,3.5) | 3 |
| | surface area | 88.67456 | 32 | (-2.852,-8.475,3.5) | 1 |
| | total area | 3.944156 | 32 | (-0.441,-9.474,3.5) | 1 |
| hsphere | object area | 3.9120922 | 2500 | (-4.702,-5.789,4.25) | 1 |
| | surface area | 915.4229 | 2500 | (-2.381,0.269,2.25) | 1 |
| | total area | 7.0510383 | 2500 | (0.750,-2.500,6.25) | 1 |
| vertexes | object area | 3.9416814 | 32 | (-4.243,-5.000,5.243) | 1 |
| | surface area | 26.070793 | 32 | (-3.528,-7.357,5.243) | 1 |
| | total area | 4.8833265 | 32 | (4.161,-5.828,5.243) | 1 |
| bruteforce | object area | 4.119752 | 38964 | (-4.5,-8.000,5.25) | 1 |
| | surface area | 1349.9253 | 38964 | (-5.25,-0.750,2.25) | 1 |
| | total area | 7.341034 | 38964 | (0,-8.000,6.75) | 1 |

The cylinder scene consists of thirty-four faces and sixty-four vertexes. the faces method is not able to perform that well on this scene, this is interesting as it is partly optimized for this scene. Here this means that the combination mentioned in the viewpoint generation is tweaked in such a way to generate the best results for both the plane and this scene as the expectation was that it could perform well on those scenes.

Table 4: Icosphere object

| Sphere | | score | tested | origin | equal |
|---|---|---|---|---|---|
| faces | object area | 2.392097 | 80 | (-1.443,-0.0,7.703) | 1 |
| | surface area | 73.30085 | 80 | (-3.958,1.373,2.019) | 1 |
| | total area | 2.6971087 | 80 | (-1.443,-0.0,7.703) | 1 |
| hsphere | object area | 2.3421383 | 8960 | (3.642,-0.0,-3.768) | 1 |
| | surface area | 2307.4412 | 8960 | (-5.017,2.470,2.176) | 1 |
| | total area | 4.6010475 | 8960 | (5.342,2.528,0.926) | 1 |
| vertexes | object area | 2.333751 | 26 | (-2.552,1.854,6.104) | 1 |
| | surface area | 67.37094 | 26 | (1.658,-5.104,3.683) | 1 |
| | total area | 3.3571286 | 26 | (3.527,4.854,1.000) | 1 |
| bruteforce | object area | 2.4158323 | 38964 | (-2.0,4.0,6.25) | 2 |
| | surface area | 1298.0822 | 38964 | (-4.0,1.5,1.750) | 1 |
| | total area | 4.6619964 | 38964 | (4.25,4.5,1.000) | 1 |

The icosphere scene consists of eighty faces and forty-two vertexes. For this scene it is interesting to note that the visible surface area found by the hsphere method is larger than the one found by the brute-force method. An explanation for this is that the brute-force method generates a grid inside of the volume and does not check the entire volume, the brute-force method likely skips over some more optimal viewpoints. The issue could be fixed by forcing all viewpoints to align to some small grid enforcing the brute-force method to iterate over all possible viewpoints. However, as can be seen here it is then possible that the actual optimal viewpoint is not contained within that set.

## 4 Discussion

This section discusses the limitations of the methods used in this paper and possible ways to improve upon them in future work. A limitation that is already discussed briefly in the result section is the precision of the methods used. It is not possible for this method to iterate over a volume and thus it is possible to miss the optimal viewpoint in the grid that is created. This might be solved by ensuring the optimal viewpoint is aligned to the grid. It could also be solved by finding a way to transform a volume into a finite set of points to be tested.Another limitation is the metrics. The total area metric seems to generate the most natural results. However, it does not check the distance to the object and it also does not test how much of the field of view is in use. Using these two factors might lead to better results in the end.Another interesting question that came up during the research is one of symmetry. It might for example be possible to find if an object has symmetry by comparing the viewpoints that generate the same scores. The cube object seems to support this possibility by generating at least 4 equal optimal viewpoints for each metric and method, one for each face not facing the surface or directly away from it.

## 5 Conclusion

The faces method performs relatively well for the plane and cube objects on the visible surface area metrics. More specifically the visible surface area it finds for the cube object is quite close to the maximum found by the brute-force solution for the cube object. The vertexes method can score better than both the faces and hsphere methods for the object area metric on the cube and cylinder objects. The hsphere method is the best alternative method for finding the optimal viewpoint for the total area as it scores second in this category for every object file. It is even able to surpass the brute-force method for finding the maximum visible surface area for the icosphere object. A possible reason for this is already explained in the results section. The brute-force method performs as expected scoring the highest on all but one of the scenes and metrics.This shows that the methods proposed here are not optimal. It does however show that some parts of the search space of the scene generally score higher than other parts. With some more work in improving the precision and refining the metrics, I believe that it should be possible to find an optimal viewpoint by checking only a fraction of the total search space. Another point that supports this claim is the optimization done on the brute-force method that all points too close to the object were removed. Since the optimal viewpoints found by this method stayed the same that is possible to prune the search space.

## References

[1] Francesco De Comité. A general procedure for the construction of mirror anamorphoses. In George W. Hart and Reza Sarhangi, editors, *Proceedings of Bridges 2010: Mathematics, Music, Art, Architecture, Culture*, pages 231–238, Phoenix, Arizona, 2010. Tessellations Publishing. Available online at http://archive.bridgesmathart.org/2010/bridges2010-231.html.

[2] Niels Thrane, Lars Ole Simonsen, and Advisor Peter Ørbæk. A comparison of acceleration structures for gpu assisted ray tracing. Technical report, 2005.

[3] Chang Wang, Fei Qi, and Guang-Ming Shi. Nodes placement for optimizing coverage of visual sensor networks. In Paisarn Muneesawang, Feng Wu, Itsuo Kumazawa, Athikom Roeksabutr, Mark Liao, and Xiaoou Tang, editors, *Advances in Multimedia Information Processing - PCM 2009*, pages 1144–1149, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[4] Zhou Wang and A.C. Bovik. *Modern Image Quality Assessment*. Morgan & Claypool, 2006. https://doi.org/10.2200/S00010ED1V01Y200508IVM003.

[5] Yuko Yanagawa, Kaori Ikematsu, Chihiro Suga, Mana Sasagawa, Yasushi Matoba, and Itiro Siio. Anamorphicons: An extended display utilizing a cylindrical mirror widget. In *Proceedings of the 29th Australian Conference on Computer-Human Interaction*, OZCHI '17, page 457–461, New York, NY, USA, 2017. Association for Computing Machinery.

[6] TianXing Yu, LiYang Xiong, Min Cao, ZhiHui Wang, YiChi Zhang, and Guo'An Tang. A new algorithm based on region partitioning for filtering candidate viewpoints of a multiple viewshed. *International Journal of Geographical Information Science*, 30(11):2171–2187, 2016.