Optimizing Art Placement using Minkowski Sums and Delaunay Triangulation

Thomas Bird

Abstract

This paper introduces a sophisticated computational technique for optimal art placement within architectural environments, leveraging the CGAL computational geometry library. The methodology involves calculating the Minkowski sum of a floor plan to understand spatial geometry, followed by Constrained Delaunay Triangulation for structuring the space into manageable units. This foundation allows for strategic placement of simulated viewer points within the environment, with a focus on maximizing visibility and enhancing spatial interaction. The algorithm's robustness in handling visibility calculation represents a novel approach in the field of architectural optimization, offering a quantitatively driven, efficient solution for art placement.

Motivation

Previous studies and projects have focused on analyzing floor visibility through heatmaps, but there has been a lack of emphasis on wall visibility. This project addresses this gap by exploring optimal art placement, targeting walls within architectural spaces such as museums and homes. The primary goal is to provide actionable advice on where to place art to maximize its visibility. By focusing on wall surfaces, the project seeks to enhance the aesthetic and cultural value of spaces, offering museums and homeowners a novel, data-driven approach to art display and spatial design.

Related Work

Introduction

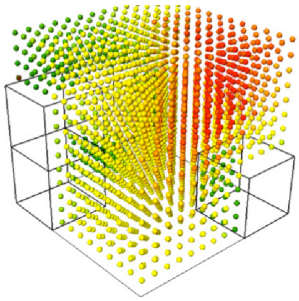
The study of visibility within architectural spaces is a nuanced intersection of computational geometry, architectural theory, and spatial analytics. Historically, this research has gravitated towards analyzing floor space visibility, employing mathematical models and simulation techniques to understand and optimize spatial dynamics. These methodologies have evolved, contributing significantly to both theoretical and practical understanding of spatial visibility and its implications in architectural design.

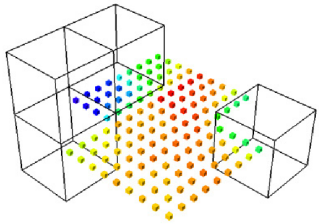
“From isovists to visibility graphs”

Turner et al.'s architectural paper on isovists represents a cornerstone in visual analysis. Focusing on floor space visibility, their methodology utilized randomized point placements within spatial confines to generate isovists—areas visible from a fixed point in space. This approach not only highlighted the potential for spatial analysis but also informed our method's aspect of observer simulation. Their work's visual results, as illustrated in the figure, showcase the depth of spatial visibility analysis possible with such methodologies, wherein each point computes an area visible to it, referred to as an “isovist polygon.”

“Beyond Two Dimensions”

In "Beyond Two Dimensions," Varoudis and Psarra expanded the scope of spatial visibility analysis into the three-dimensional realm. Their approach underscored the complexity and potential of 3D spatial analysis, particularly in multi-level architectural environments. However, acknowledging the general predominance of 2D floor plans in architectural and design work, I opted to retain a 2D approach for broader practicality and accessibility.





The intricate visibility patterns captured in their 3D models are depicted in the figures above. As you can see, spacial visibility is computed based on staggered, non-random nodes. These nodes occupy places in three dimensional space, and are colored based on their visibility – red being most visible.

Project Divergence:

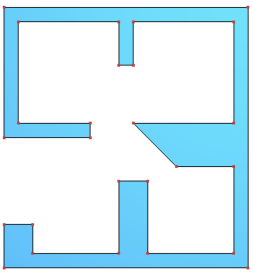
This project represents a significant divergence from these foundational studies by shifting the focus to wall visibility, an aspect often overlooked in traditional visibility analysis. Employing advanced computational geometry techniques, my method emphasizes the optimization of art placement, enriching the aesthetic and functional value of architectural spaces. This approach moves beyond conventional visibility graph techniques, offering a geometry-driven, innovative perspective on spatial visibility and its application in art placement within environments – museums and homes in particular. The program is designed to take in an input of 2-dimensional floor plan data, the output a heatmapping of viable wall space.

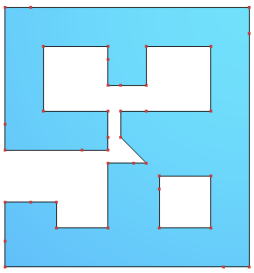
Implementation

The computational process for this project can be divided into four main steps:

1. Minkowski Sum of Input Floor Plan:

The first step involves taking the Minkowski sum of the input floor plan with a small square, adding a border around each wall. This border is crucial for displaying the heatmap in later stages.



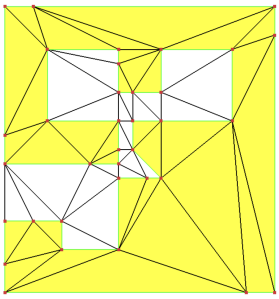


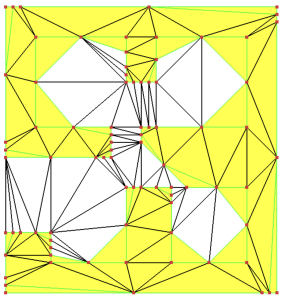
See the figures above for the before (*left*) and after (*right*) of the Minkowski sum step, as it takes the input floor plan and adds a buffer around it to create a thicker area to display the heat map upon. This sum will then be segmented and colored, and the input floor plan will be overlaid to allow for easy interpretation of the results.

1.1: Bugs and Limitations

The reasoning behind the small square is that it creates a fairly consistent and simple box around the input floor plan. I had attempted to use other shapes (a star and circle), but they created difficulties with regard to the result map and overcomplicated the shape of the final sum. More shapes could be explored, but through my trial and error, I found that the square gave the best results.

2. Constrained Delaunay Triangulation:





This Minkowski sum from step one is then triangulated via the Delaunay Triangulation algorithm. The constraints allow for marking of the appropriate faces related to the wallspace (*yellow*) in the diagram, so that excess triangles representing walkable floor space can be ignored. This can be seen in the associated images, where yellow represents heatmap area, and white is empty room.

The triangulation process is enhanced with a user-defined "precision" argument. This

parameter controls the triangulation density, with a default value of 0 leading to standard triangulation and higher values increasing triangle numbers. The improved strategy involves subdividing edges of the sum for each precision level, which effectively increases triangle counts while maintaining visibility integrity. Refer to the figure above for precision 0 (*left*) and 1 (*right*) triangulation visuals. As you can see, the triangle count is multiplied considerably.

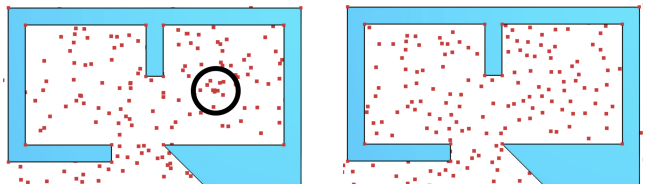
2.1: Bugs and Limitations

When considering implementation for the precision variable, I had attempted a number of other techniques. In order to create meaningful triangles, they must have at least two points on the edge of viable wall-space, as two points must be visible to a person to deem it a visible chunk of wall. My initial attempt at created more triangles involved cutting each triangle into four triangles, in a Serpinski pattern. This, however, resulted in one triangle encapsulated by three triangles – thus it was impossible for that internal triangle to be visible and resulted in useless triangles.

In order to mitigate this, the precision parameter instead designates the number of excess points to add to an edge. That is, for each increase in precision, each edge of the Minkowski sum is doubled (by adding and connecting the midpoint to the source and target vertex of each edge). This results in more triangles, which – most importantly – still border the edge of the heatmap.

3. Placement of 'People':

Random two-dimensional points, representing 'people', are placed within the open floor space. The placement adheres to either white noise or blue noise patterns, based on user input. White noise is purely randomized noise, wherein the x and y coordinates are fully random when placing the point. Blue noise is the same, except it restricts point placement by ensuring each new point is at least a set distance away from others.



Above is a comparison on white noise (*left*) and blue noise (*right*) distributions in an example floor plan. As you can see, the white noise results in clusters (*circled*), which are unrealistic for simulating a crowd of people. This overlapping, implying individuals occupying the same physical space, is impractical and not representative of real-world scenarios. Blue noise still allows for randomized placement of individuals, but within a certain distance (*epsilon*) that designates the minimum space between two people. This "socially distanced" model more accurately mimics the placement of people in real life.

3.1: Bugs and Limitations

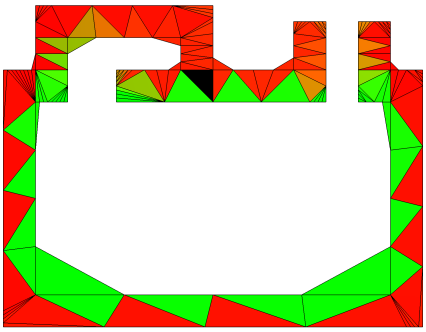
In order to accurately distance people, an arbitrary constant of 0.1 (correlating to 0.1 meters based on the scale of the floor plans) was deemed an appropriate minimum spacing between individuals – not exactly comfortable, but it can emulate a crowded and busy space quite well.

While epsilon could be an input argument, this led to infinite looping in testing. If you designate epsilon to be too high of a number, and ask it to insert a high enough number of people, then it can fully cover the useable space available and leave no room left to place the remaining points. This can be fixed by instead replacing the people and epsilon inputs with a density input, but for the sake of testing and exploring various results it was simpler to control for epsilon and allow users to specify the number of people instead.

4. Visibility Calculation and Heatmap Generation:

Each person's viewpoint is analyzed to determine visible space. For every point in the people list, it is checking for visibility against each triangle. For a single triangle, it checks if each

vertex is visible. That triangle is deemed visible if at least two of the three vertices are visible. When checking visibility of a vertex, a line segment is created between the person and the vertex. If the segment does not intersect any lines on the input floor plan, then line of sight is established.



The visibility score of each triangle is then normalized and used to generate a heatmap, coloring triangles from green (*highly visible*) to red (*low visibility*) with non-visible triangles colored in black. This heatmap (*above*) guides the optimal placement of art based on how many people can see a given wall.

During computation, these scores are stored in a map, mapping each triangle's Face Index in the surface mesh to a float (score). Score will go up a single point for each person that can see the triangle, and a maximum, minimum, and average score is computed during the process.

4.1: Troubleshooting

Some apparent "issues" can be seen with the output graph. Specifically, that triangles can cross through an entire wall segment – the concern here being the possibility for false positives. However, this is a non-issue. Since two vertices must be visible, a triangle is green only if its visible edge can be seen. Thus, when interpreting the results, one can ignore the skinny point that extends from the triangle across a wall. Understandably, this is not ideal and can be improved upon. I explored a possible solution regarding intermediate points within the wall-space, but these calculated spots weren't always accurate. Another possible avenue to remedy this could involve tossing out triangulation all together in favor of say voronoi diagrams – but the decision to use triangles simply came down to preference.

It is a concern that exterior walls are visible as well. This is due to the placement of people, where some make it outside the walls. In testing, I made some attempts at remedying this – but ultimately deemed trivial for the time being.

Technology

This program was written in C++, leveraging CGALs draw function to display the results.

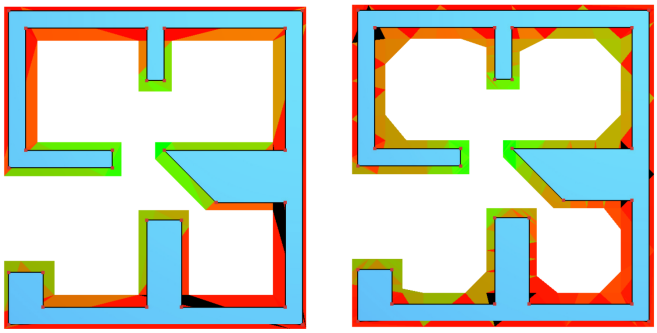
- CGAL's Polygon 2 was utilized to represent the input floor plan.
- Polygon 2 with holes was used to generate hte Minkowski sum
- A Constrained Delaunay Triangulation datastructure was then created from the sum.
- A surface mesh data structure was then used to display the triangles in color.

Results

The results yielded from this program were quite promising, and displayed exactly what I had intended it to do. There is a sizeable variance on the output given the different parameters and arguments for the program, as well as the nature of randomization. I will detail a few of the most significant, and discuss the pros and cons of each, as well as their accuracy and usefulness.

The Precision Argument

Let us discuss the results with regard to change in the precision variable:



This figure (*above*) displays the results for two runsusing 100 people placed over white noise, with two different levels of precision. The results for a precision level of 0 (*left*), or no further

triangulation, can drastically oversimplify the room and give very uninformative results.

Using a preciiosn level of 2 (*right*) gives much more specific results. The result is segmented further and thus is more informative and specific.

noise	people	precision	max vis	min vis	avg vis
white	100	0	2	77	30.24
white	100	1	2	76	32.22
white	100	2	2	78	33.50

In the table above, which shows the varying outputs among these results, it can be gleaned that the results are approximately the same. However, as we know from the ouput, higher precision results in more accurate results, and it can be assumed that the average visibility is more accurate for higher levels of precision. Note that the numbers are normalized to a visibility%, or the percentage of people simulated that can see a given triangle.

Altering the level of precision allows for more specificity in the resulting heatmap, giving more informative and fine-tuned answers. Accordingly, for the example output figures above, it would be optimal to place art in the center (green) area of this room. While they all indicate this, the higher precision output shows a more highly localized and pinpointed result. As it is clear the higher precision level is better, let us control our precision to 2 for the remaining results.

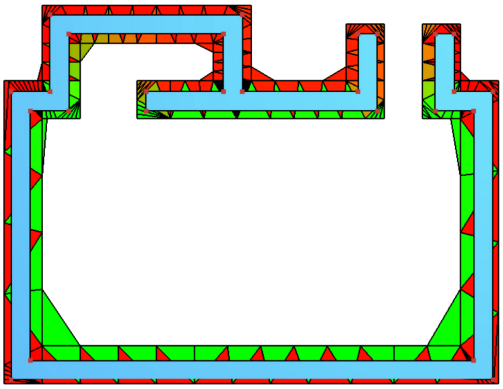
The People Argument

This figure shows the importance of the "people" variable. While we are using blue noise and precision level two, it only represents visibility from three individuals. Due to the nature of randomization, such a small sample size results in very skewed results. As such, for optimal coverage, it is desirable to use a large number of people.

Strong Results

Testing for the usefulness of blue noise over white noise can only be shown in theory, as the resultant output is nearly indiscernable, and only the numerical data is altered. However, as blue noise is more accurate to the placement of people in theory, then I've deemed it more useful in making accurate and ultimately better results.

Thus, as discussed, a more accurate and useful result can be achieved by maximizing the precision variable, maximizing the number people within a tolerable range given an epsilon of 0.1, and placing them via blue noise.



Above is a theoretically optimized result for a classroom floor plan – generated with precision level two, placing 400 people via blue noise. An interesting thing to note is that, given the simple geometry of the classroom, most walls in the larger area are approximately equally viable (76.5%). In this scenario, a possible extension to mitigate this and increase the usefulness of the program would be to overlay the normal scores of each

Conclusions

In summary, this project took about a three weeks of on and off work. The trajectory of the work changed multiple times, as I researched various possible technologies to achieve what I wanted. The results are quite promising, and offer exactly what I wanted them to, but that is not to say they are perfect by any means.

My decision to use Minkowski sums to create a buffer is not something that has been done in this specific type of visualization graphing. Furthermore, as previously mentioned, the use of triangulation of voronoi diagrams was an equally interesting decision. As such – due to my unorthodox approach – there are several avenues for improvement and future work in this field.

Future Work

Usability

A core focus of this project was usability. The simplicity of using 2D floor plans allows for quicker input and results. This decision was made based on the accessibility of modern floor plans to the public, promoting the use of this software over one that considers three-dimensional data input. This leads into the first possible extension, the interpretation of floor plan imagery for more efficient computation. That is, the ability to upload an image of your floor plan so as to avoid manual input. Incorporate image scanning would yield way more accurate input data as well, not to mention be more efficient use.

Higher Dimensions

As referenced in the architecture paper by Varoudis and Parra on higher dimensional space analysis, 3-dimensional viewpoint calculations can prove quite helpful for interpreting visibility. This extension would allow for a number of things, namely higher accuracy, precision, and more realistic viewpoint simulation. By rendering in three dimensions, certain parts of a wallspace which may be obscured at one height level, may not be at another. This cannot be accounted for in a two-dimensional model. However, as previously stated, 3D floorplan data is not readily available.

Optimization

Sticking with the current implementation, it would be interesting to see an alternative approach involving different noise types, and different segmentations for heatmap generation. It is also possible to make use of raw pixel data instead of triangles. This would lead to a higher computation cost, but ultimately more fine-tuned results. This approach is inspired from the work of Turner et. al, where pixelated data provides highly accurate and localized visibility data.

Wrapping Up

In conclusion, this project offers a new approach to art placement in architectural spaces, utilizing computational geometry. By integrating Minkowski sums, Constrained Delaunay Triangulation, and visibility calculations, this paper provides an effective method for improving art visibility. This work bridges a gap in spatial analysis, suggesting potential areas for further development, such as three-dimensional analysis and automated data input. The project melds computational techniques with architectural design, contributing useful insights and tools for enhancing aesthetic experiences in built environments.

Sources & References

Turner, Alasdair, et al. “From isovists to visibility graphs: a methodology for the analysis of architectural space.” *UCL Discovery*, https://discovery.ucl.ac.uk/id/eprint/160/1/turner-doxa-osullivan-penn-2001.pdf. Accessed 12 December 2023.

Varoudis, T., and S. Parra. “Beyond two dimensions: Architecture through three-dimensional visibility graph analysis.” *Semantic Scholar*, 27 August 2014, https://www.semanticscholar.org/paper/Beyond-two-dimensions%3A-Architecture-through-graph-Varoudis-Psarra/df2b16284e04d22a1bd90850ade192f14758cdc5. Accessed 12 December 2023.

Muratori, Casey. “The Color of Noise.” *Casey Muratori*, 23 May 2014, https://caseymuratori.com/blog_0010. Accessed 12 December 2023.

Demofox. “Generating Blue Noise Sample Points With Mitchell's Best Candidate Algorithm.” *The blog at the bottom of the sea*, 20 October 2017, https://blog.demofox.org/2017/10/20/generating-blue-noise-sample-points-with-mitchells-best-candidate-algorithm/. Accessed 12 December 2023.