SimBuild
2024

Eleventh National Conference of IBPSA-USA
Denver, Colorado
May 21 – 23, 2024

Convex Partition Zoner: A New Algorithm for Automated Thermal Zoning

Jialiang Xiang[1], Quoc Dang[2]; Carlos Cerezo Davila[2]; Holly Samuelson[1]
[1] Harvard University Graduate School of Design, Cambridge, MA
[2] Kohn Pedersen Fox Associates, New York, NY

Abstract

This paper introduces a new algorithm that utilizes an iterative process to automatically generate thermal zones of energy models from building floor geometries when the actual HVAC zones are unknown or not yet designed, adhering to the ASHRAE standard 90.1 appendix G method. Our evaluations demonstrate that it is faster and less prone to error than the state-of-the-art auto-zoning tools, while its simulation results are close to those of a manually zoned energy model. It has potential applications in early-phase architecture design explorations and urban scale energy modeling.

Introduction

As the reduction of building greenhouse gas emissions becomes an increasingly high priority in the building industry, there has been increasing interest in automating the creation of energy models, especially in the context of early design architectural explorations where interior layout and mechanical system information is limited. Automation is also essential for the generation of Urban Building Energy Models (UBEMs) of existing cities (Reinhart, Cerezo 2016) or any other large-scale energy modelling application.

Simulations are by necessity reduced approximations of reality, and simulation tools are designed with speed, memory consumption, and accuracy in mind. Since simulation tools are not supposed to generate the models they simulate, until these tools become more flexible, it is up to modelers or automated equivalents to conform with the tools' intended usage and input preferences.

Therefore, when the HVAC zones are unknown or not yet designed, a good automated thermal zoning algorithm for energy modeling needs to satisfy four requirements: First, it must be able to create perimeter and core zones for any reasonable floor geometry. In the absence of HVAC zone information, ASHRAE Standard 90.1, Appendix G requires models to have separate "thermal blocks" (thermal zones) for perimeter and core spaces, with the perimeter zones being 15 feet deep or less from an exterior wall (ASHRAE 2022). The standard also requires separate blocks for perimeter spaces of different orientations (ASHRAE 2022). Since EnergyPlus assumes air to be perfectly mixed within a single zone, strategically dividing floor space into multiple zones helps overcome the accuracy limitations of single zone models.

Second, the thermal zoning algorithm must be compatible with the geometric recommendations of most available simulation engines, such as EnergyPlus by the U.S. Department of Energy (DOE), where the presence of concave thermal zone footprints can result in modelling errors (DOE 2023, Engineering Reference).

Third, the zoning algorithm must avoid unnecessary geometric complexity and geometric aberrations that can extend simulation time and cause runtime errors within the simulation engines.

Finally, the method must produce simulation results that are accurate enough for the intended purpose.

Existing proposed algorithms available in architectural modelling environments do not fare well against these four criteria, and in practice require some degree of manual revision. For example, in the Grasshopper (GH) plug-in Honeybee, the *Split Floor 2 Thermal Zones* command cannot handle concave zones or courtyard geometries (Roudsari 2020); the AutoZoner algorithm implemented in ClimateStudio for GH (Dogan, Reinhard 2016) also cannot automate the generation of "concave-free" models and can introduce unnecessary complexity and geometric aberrations.

This paper introduces a new proposed algorithm, Convex Partition Zoner (CPZ) to address the challenges of existing methods. we will discuss the accuracy of the proposed method in the Results section.

Methodology

This new algorithm (CPZ) features an iterative optimization process that consists of four stages as

shown in Figure 1. It is implemented in C#, as a self-contained geometric analysis component compatible with any energy modelling workflow starting with a geometric zoned model within the Rhinoceros3D modelling software and its scripting environment Grasshopper (McNeel 2020). Rhino3d and GH are common software suites used for architectural modelling and environmental analysis in early design, where automated thermal zoning is most needed. It is worth noting that, if the floor geometry is already convex, the optimization process is not needed and only the WP stage is performed.

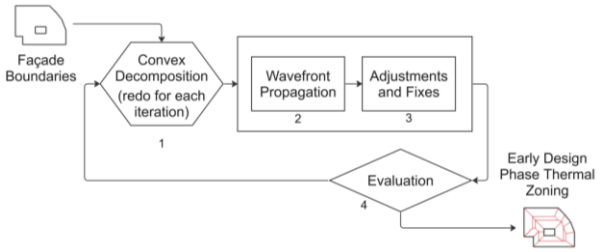

Figure 1: flowchart for the algorithm

Input and Output

The algorithm takes as input the polygon boundaries of the building floorplate for every floor modelled. If the building has courtyards, the input will include the corresponding inner boundaries. The algorithm outputs line segments forming the thermal zones and distinguishes between segments forming core zones and perimeter zones, that can be then used to generate 3d volumes for each space. The inputs previously mentioned are then processed through steps 1 to 4 in Figure 1 to produce the outputs.

Convex Decomposition (CD)

For a closed polygon, a "convex" vertex refers to a corner whose inner angle is smaller than 180º; whereas a "concave" vertex refers to a corner whose inner angle is greater than 180º (Figure 2). Convex Decomposition (CD) refers to the procedure that subdivides a polygon with straight line virtual partitions such that no concave vertices remain (Figure 3). Henceforth, the term "partition" refers to a virtual partition with no physical properties.

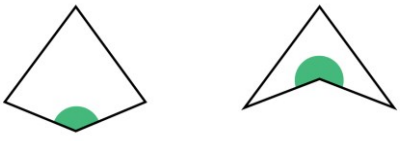

Figure 2: Examples of vertices with convex (left) and concave angles (right), marked in green

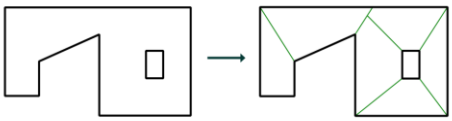

Figure 3: Example of Convex Decomposition

Extending the two incident polygon segments at the concave vertex in the opposite direction, we create the "reversed cone" of the concave vertex (Figure 4). It can be observed that any partition of the polygon that falls within this "reversed cone" divides this concave vertex into two convex ones.

Figure 4: The "reversed cone" for Convex Decomposition

For this algorithm, we adopt four potential strategies to decompose each concave vertex as illustrated in Figure 5. The rationale behind them will be explained in the Optimization section. We also disallow having a single vertex or point on boundary connected to more than one partitions, as this greatly simplifies the Wavefront Propagation (WP) algorithm.

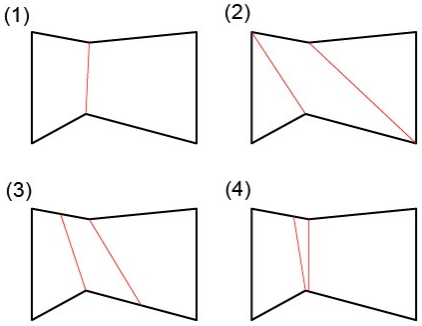

Figure 5: Partitioning options to be considered to decompose each concave vertex: (1) Connect to another concave vertex; (2) Connect to a convex vertex; (3) Connect to the midpoint of a boundary segment; (4) Bisect the concave vertex

Unlike the existing automated zoning algorithms like AutoZoner, which performs the WP to generates the straight skeleton geometry but does not perform a CD step, we choose to do the CD to partition the floor into

convex subfloors before performing the WP. The advantages are twofold: first, this greatly simplifies the WP algorithm, since implementing WP that handles concave vertices is a notorious technical challenge (Huber and Held 2011); second, since the WP now follows the partitions, this allows for a certain degree of structural control over the final form of the thermal zones, as will be explained in the Optimization section. More importantly, it eliminates the need for manual adjustment of the geometry for concave zones, that can largely extend the modelling process.

Wavefront Propagation (WP)

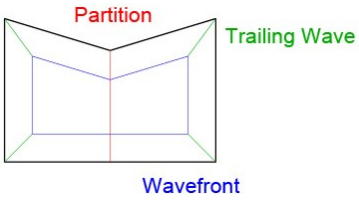

Figure 6: Terminologies for our algorithm

This is the heart of the algorithm as WP creates a rough draft of the thermal zones. The concept of WP originates from the definition of Straight Skeleton (SS) (Aichholzer et.al 1995), where all polygon edges propagate inwards at the same speeds and stay parallel to the boundaries throughout the process. For SS, the wavefronts keep propagating until further propagation becomes impossible as the wavefronts collide with each other. In existing algorithms like AutoZoner, the SS geometry is first generated before a second offset algorithm is used to cut out the core zones. As illustrated in the Results section, the imperfect compatibility between the two algorithms can be a source for modelling problems. Unlike the precedent, we redesigned and implemented the WP algorithm for thermal zone generation such that the propagation stops as soon as the desired perimeter zone depth is reached even if the wavefronts can be propagated further (Figure 7). This eliminates the need for a second offset algorithm.

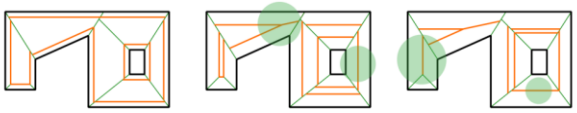

Figure 7: Our Wavefront Propagation Algorithm with events of merging wavefronts highlighted

Another major difference from the precedent is that our algorithm performs the CD before the WP. In our process, the floor polygon would be subdivided into subfloor polygons that are convex, and the WP can be performed independently for each convex subfloor in isolation. To avoid creating unnecessary zones around the partitions, only the real floor boundaries are propagated, and the partitions are not (Figure 8). When a piece of boundary is adjacent to a partition, it propagates along that partition. Therefore, unlike the SS geometry where the propagation must proceed in the directions of the bisecting lines of the vertices, here the partitions generated in the CD step can guide the propagation towards more advisable directions.

Please refer to Appendix A for a detailed description of this algorithm.

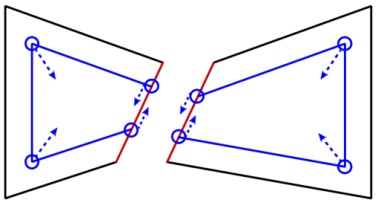

Figure 8: Data structure for our convex-only Wavefront Propagation, showing floor boundaries propagating along the partitions (red)

Adjustments and Fixes (AF)

After the CD and WP steps, we now have the rough shape of the thermal zones, but because the WP is performed independently for each subfloor, the zone boundaries need to be mended across the subfloors to eliminate unnecessary line segments or zone corners that do not perfectly align, while maintaining convexity.

Adjustments

The Adjustment 1-2 step (Figure 9, 10) is meant to mend together the fragmented wavefronts across the subfloors. This is guaranteed to generate convex geometries. Please refer to Appendix A for a detailed description of this algorithm.

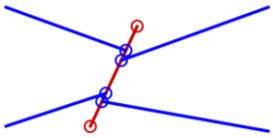

Figure 9: Data structure for Adjustment 1-2

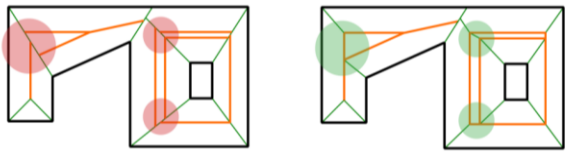

Figure 10: Adjustment 1-2 mends the wavefronts across the subfloors

In the case that one partition hits another, the Adjustment 2-2 step (Figure 11) is meant to bring the partition intersection point into a trailing wave on the other side. As long as the CD step remains valid, this generates convex geometries as every step after CD guarantees convexity.

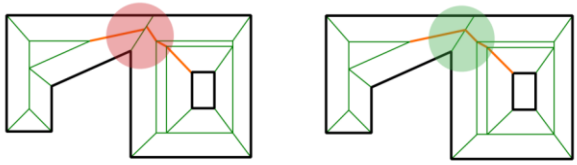

Figure 11: Adjustment 2-2

Fixes

While the Adjustments work with the wavefronts and partitions, the Fixes work with the trailing waves on the same partition that can potentially be brought together to eliminate the line segment in between. Due to their ad-hoc nature, after a Fix is performed, convexity checks are necessary for each corner involved.

Fix 1-3 intersects the trailing waves on the same partition, then bend the partition, whereas Fix 2-3 moves the trailing waves along the partition to the average point in order to eliminate the middle segment. Fix 3-3 snaps trailing waves close to a partition endpoint to that endpoint.

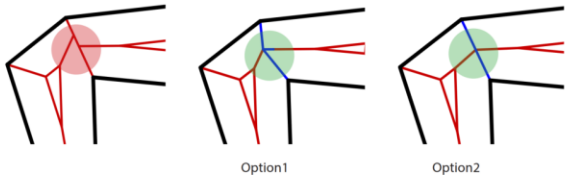

Figure 12: Fix 1-3 and 2-3

Optimization

The three stages we have introduced, the CD, WP, and AF, together constitute our workflow for generating the perimeter and core zones from a polygon floor geometry.

While it can be proven mathematically that the generated zones are free of concavity, that alone does not ensure their suitability for energy modelling: the CD stage may produce partitions leading to zones that are redundant, that are geometrically complex, that are ill-proportioned, and that are impossible to mend together at the AF stage. With this final optimization step, we aim to search for a good partitioning scheme in the CD stage efficiently and produce a set of thermal zones that are clean and well-proportioned.

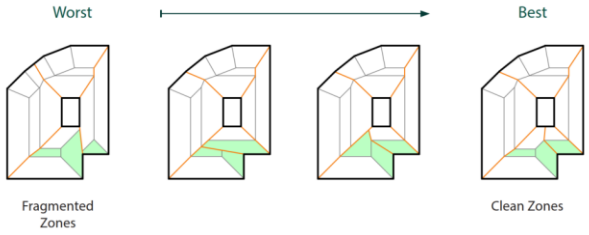

Figure 13: Some partitioning schemes are more desirable than others

Search Space

As we revisit the CD stage in the light of WP and AF stages and consider the ways to generate virtual partitions for each concave vertex, four options stand out as reasonable, in the order of advisability for any given geometry. Figure 14 illustrates these options, ordered below in terms of their desirability:

1. *Connect to another concave vertex if the same partition also decomposes that vertex.* This is usually the most advisable option, as it decomposes two concave vertices with a single partition, thereby producing fewer zones than all other options.
2. *Connect to another convex vertex.* This is also an advisable option. Since each partition doubles as two trailing waves, this produces fewer line segments than the remaining options.
3. *Connect to the middle point of another boundary segment.* Connecting to middle points is a good way to space out the partitions and avoid creating excessively small zones, considering it is illegal to have two partitions connected to the same point on the boundary, as laid out in the CD section.
4. *Bisect this concave corner and connect to another segment or partition.* This option creates as many line segments as the previous one with less control over the proportions, so it is often the least advisable among the four. However, this option has the convenient geometrical property that it does not require the AF stage and always produces clean geometry right at the WP stage, as can be proven mathematically (Figure 15). Therefore, it is a safe option and the last resort.

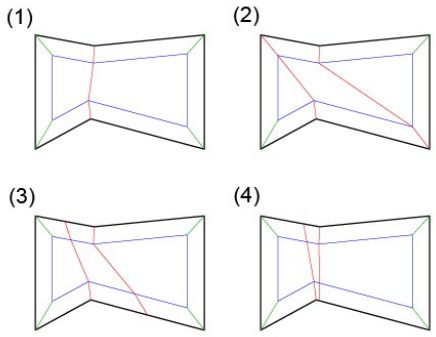

Figure 14: Partitioning options to be considered to decompose each concave vertex, after the WP and AF processes: (1) Connect to another concave vertex; (2) Connect to a convex vertex; (3) Connect to the midpoint of a boundary segment; (4) Bisect the concave vertex

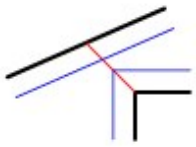

Figure 15: The AF stage is unnecessary for bisecting partitions due to their geometrical properties

First Metric: Rank

After grouping the potential partitions into the four ranked CD options above, we can observe that shorter partitions are generally more advisable than longer ones. For one, established practice often attempts to minimize the perimeter length of thermal zones. More importantly, an especially long partition often leads to overly shallow zones in the WP stage (Figure 16).

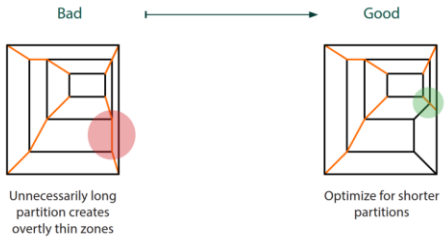

Figure 16: short partitions often create better zones

In response to these observations, we rank the potential partitions within each CD option group by length, with shorter ones given higher ranks. Combined with the ranks of the four groups, we generate a unique rank for every potential partition. Furthermore, we allow the promotion of a potential partition across groups if it is especially short. This is done by assigning the highest ranked partition within each group as its "key" and promoting a partition of lower rank to in front of a key if it is much shorter by a constant factor. In the example illustrated in Figure 16, this allows the much shorter bisecting partition (green) to have a higher rank than the long partition towards a convex corner (red). Through the metric of rank, we estimate the success of a partitioning scheme during the CD stage.

Second Metric: Cleanliness Score

After the WP and AF, and the thermal zones are produced, we measure the cleanliness of the result with a second metric, the cleanliness score. To find the cleanliness score, we add up the number of lines that remain unfixed for each partition after the AF stage, then the negative of that sum is the score for this zoning option (Figure 17).

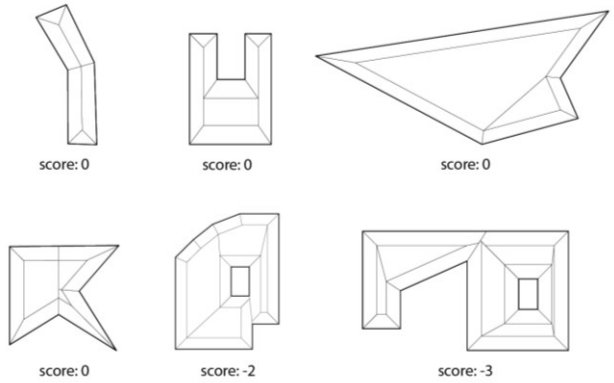

Figure 17: Examples of cleanliness scores

Optimization Objective

With the two metrics of rank and cleanliness score, the objective of the optimization is to find the zoning scheme with the highest feasible rank that produces the maximum cleanliness score in the entire solution space.

Efficient Optimization

We can quickly conclude that attempting every single possible combination of partitions in CD is too cumbersome, as the number of options explodes exponentially with the number of concave vertices in the floor geometry. For our algorithm to be useful, we need to be able to quickly find a good-enough partitioning scheme from the design space.

To our convenience, because AF is performed locally one partition at a time, the cleanliness score at a partition generally does not depend upon the configuration of the other partitions. This is also true for the rank metric, which is a unique number assigned to each partitioning option at each concave vertex. As a result, a greedy algorithm that optimizes locally one partition at a time in

fixed order tends to arrive at a globally near-optimal solution. With this method, we can reduce the number of attempts to be linear with the number of concave vertices. This can be further sped up by marking the partitions with low scores at the end of AF, so that we can focus on fixing only the ill-behaving partitions in the following iteration, knowing this is unlikely to affect the already well-behaving ones.

Please refer to Appendix A for a detailed description of this algorithm.

Robustness

As our optimization method is based on heuristics, it may face difficulties given very specific floor geometries: It may fail to find the globally optimal solution or fail to construct a design space that contains a good solution in the first place. Even then, the generated energy model is still guaranteed to be convex, and the simulation will still run, albeit at perhaps worse speed and accuracy. Furthermore, even if the optimization was not perfect, it would still improve upon the unoptimized solution and produces a result that is hopefully good enough for the desired simulation purpose.

Evaluation and Results

Our CPZ plugin for Grasshopper is being written in C#. We have implemented the CD and WP stage, along with the iterative optimization process. More work is still needed on the AF stage to cover all edge cases. We chose ClimateStudio's Grasshopper implementation of AutoZoner to represent state-of-the art in the experiments below.

To demonstrate the flexibility of CPZ within realistic scenarios, we utilized our plugin on an urban site in downtown Boston (42.3632, -71.0532). The building footprint geometries are manually cleaned up and simplified to remove complex details (such as tiny edges and minor misalignments) before feeding into the CPZ and AutoZoner components. This process should be replaced with a dedicated simplification algorithm in the future, as explained in the Future Work section. The results (Figure 18) demonstrate CPZ's capability to handle organic urban forms like the state-of-art and decompose concave building footprints into convex zones. Both CPZ and AutoZoner components had runtimes of < 500ms.

To further investigate CPZ's efficacy, we conducted a comparative study involving three thermal zoning methods: AutoZoner (representing the state-of-the-art automated algorithms), CPZ, and manual zoning following ASHRAE 90.1 Appendix G guidelines as performed by a practicing energy modeler.

We developed Grasshopper workflows for these three methods. The AutoZoner and CPZ workflows take a 2D floor plate as polylines and generate two lists of perimeter zones and core zones in 2D. For the manual workflow these are created by hand. The 2D zones are then extruded to floor height and replicated by the number of floors to create the Breps representing the volumetric zones of the model. They are then converted to ClimateStudio (CS) zones through dedicated CS components. Window surfaces are created by scaling exterior surfaces of the perimeter zones by WWR and are then converted to CS window objects. CS's built-in settings were used for the zones, including Program (office), People (0.033 people/m^3 with metabolic rate of 1.2), Equipment Power Density (7.6 W/m^2) and Lighting Power Density (7.2 W/m^2). For windows, U-Value is set to 2 W/(m^2·K) and SHGC to 0.3. Finally, these CS objects are fed into CS Model Maker and simulated with Run CS Energy Model component.

Figure 18: From 230 buildings, AutoZoner contains 64 concave zones while CPZ has none

Test runs were performed on four hypothetical yet realistic multi-level architectural massing geometry in New York City, characterized by distinct floor plans in the shape of L, V, O, and 8. Each of the massing geometries has 5 floor levels with 3.5m floor-to-floor height (Figure 19). For each model, a basic early design EnergyPlus energy model was generated and simulated, using the Climate Studio software as an interface (Solemma 2023). For the test, each model was simulated assuming ASHRAE Standard 90.1 (2022) prescriptive values for envelope performance, internal gain peak values and schedules, ventilation rates, and modeling parameters. Typical TMY3 weather data for NYC JFK airport was used in all cases.

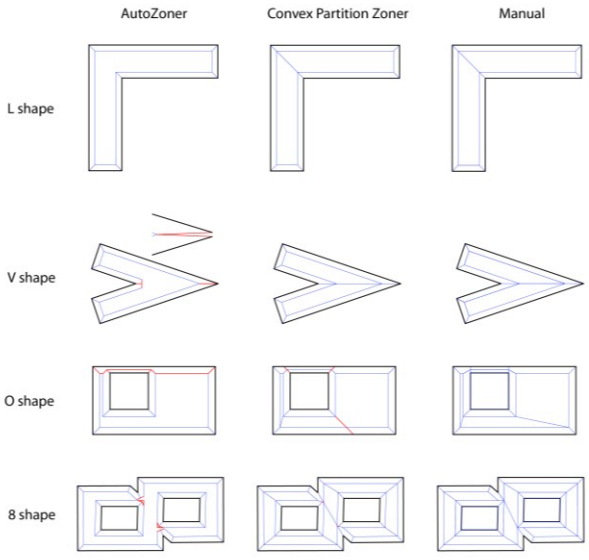

Figure 19: Early Design Phase Thermal Zoning results for different building footprints produced by Convex Partition Zoner (CPZ) and AutoZoner compared with the baseline manual zoning methods

Table 1: Comparison of EnergyPlus geometry related Error messages between AutoZoner and CPZ (Refer to Appendix B for output details)

Shading Calculation Method	V shape	AutoZoner	CPZ
Polygon Clipping	L shape	2 Warning; 1 Severe Errors	1 Warning
	V shape	Fatal Error Detected. 1 Warning; 5 Severe Errors	1 Warning
	O shape	13 Warning; 2 Severe Errors;	1 Warning
	8 shape	17 Warning; 3 Severe Errors	1 Warning
Pixel Counting	L shape	1 Warning	1 Warning
	V shape	Fatal Error Detected. 1 Warning; 5 Severe Errors	1 Warning
	O shape	11 Warning	1 Warning
	8 shape	13 Warning	1 Warning

Following industry standard, the target depth of the perimeter zones for all three methods were set to 4.5 meters (almost 15 feet). For EnergyPlus's Solar Distribution calculation method, Full Interior and Exterior was selected among the three options officially supported, for its accuracy and capacity to capture heat flows between perimeter and core (DOE 2023, Engineering Reference). Given the choice of Full Interior and Exterior, we have two options for Shading Calculation Method, Polygon Clipping and Pixel Counting, with the former depending on the Central Processing Unit (CPU) and the latter on the Graphic Processing Unit (GPU). Polygon Clipping also requires the geometry to be convex according to the documentation (DOE 2023, Engineering Reference). Test runs were performed for both methods.

The energy simulations were conducted on a standard mobile workstation, with the following specifications: Model: Dell Precision; CPU: 3.2 GHz Intel Xeon W-11855M; RAM: 32 GB; GPU: NVIDIA RTX A4000 with 24 GB of memory.

Table 2: Comparison of Best Valid Runtime (seconds) between AutoZoner and Convex Partition Zoner (CPZ)

Building Shape	AutoZoner	CPZ
L shape	52.51s (PixC)	41.34s (PolyC)
V shape	n/a	44.24s (PolyC)
O shape	68.83s (PixC)	68.29s (PolyC)
8 shape	533.68s (PixC)	145.3s (PolyC)

Table 3: Energy Model result difference compared to manual zoning (Refer to Appendix B for output details)

Building Shape	AutoZoner (no-convex) *via PixC*			CPZ (convex) *via PolyC*			Manual Zoning (convex) *via PolyC*	
	Heating & Cooling Load (kWh)	Δ (%)	Warning & Errors	Heating & Cooling Load (kWh)	Δ (%)	Warning & Errors	Average Heating & Cooling Load (kWh)	Warning & Errors
L shape	1,254,216	-14	1 Warning	1,368,577	0	1 Warning	1,368,577	1 Warning
V shape	n/a	n/a	Fatal Error Detected. 1 Warning 5 Severe Errors	833,187	0	1 Warning	833,187	1 Warning
O shape	910,557	-2	11 Warning	918,031	-1	1 Warning	918,756	1 Warning
8 shape	1,605,968	26	13 Warning	1,440,141	-2	1 Warning	1,461,982	1 Warning

Discussions

Table 1 shows Energy Plus's geometry-related log output comparing the two available Shading Calculating Methods for the two zoning algorithms. Confirming our expectations, an algorithm like AutoZoner encounters severe errors when used with *Polygon Clipping* due to its concave zones, which according to documentation means the simulation result is invalid (DOE 2023, Input Output Reference). In other words, it is limited to *Pixel Counting* whereas CPZ has the option to utilize either method thanks to its convexity. Even when the result is

valid, it still encounters many more warnings than CPZ, implying poor compatibility with EnergyPlus's underlying methods.

Table 2 demonstrates CPZ's superior speed when taking the validity of the results into consideration. An algorithm like AutoZoner runs slower in all cases despite having a smaller number of zones. This is because *Polygon Clipping*, available only to CPZ, tends to be the faster method when the number of shading surfaces is less than 200 (DesignBuilder Software Ltd 2023). The runtime for AutoZoner and CPZ themselves are insignificant for a single model but can make a difference in an urban modeling scenario where hundreds of buildings might be analyzed in multiple configurations.

In the V-shaped case AutoZoner generated a geometric aberration at the tip of the sharp corner, causing fatal errors in the simulation. This is not an isolated incident as most sharp corners at similar scale seem to experience the same issue. The 8-shaped case illustrates a more fundamental issue with Straight Skeleton (SS) based zoning algorithms like AutoZoner. Since the SS algorithm does not stop half-way during its Wavefront Propagation (WP) process, a second offset algorithm is needed to clip out the core zones. The imperfect compatibility between the two methods sometimes creates aberrations, and this is exacerbated by SS's limitation of propagating only in the bisecting direction at the concave corners, whereas the CPZ often has 3 more options. In this case two aberrations were generated near the inner corners of the courtyards, and they are caused by the compressed zones in the SS at these locations being improperly clipped (Figure 20).

Finally, Table 3 demonstrates the CPZ's simulation result is very close to the average results from human experts regarding the heating and cooling load, which is essential for the sizing of Mechanical, Electrical, and Plumbing (MEP) systems. Combined ideal heating and cooling loads as generated by the AutoZoner algorithm resulted in variations of -14 to 26 when compared with a manual ideally zoned building, while these variations were 0 to -2% CPZ.

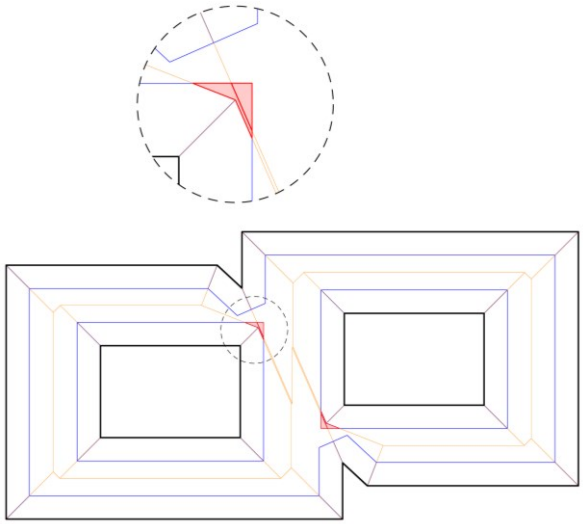

Figure 20: Artifacts (red) generated by AutoZoner when the wavefront (blue) incorrectly clipped Straight Skeleton (orange)

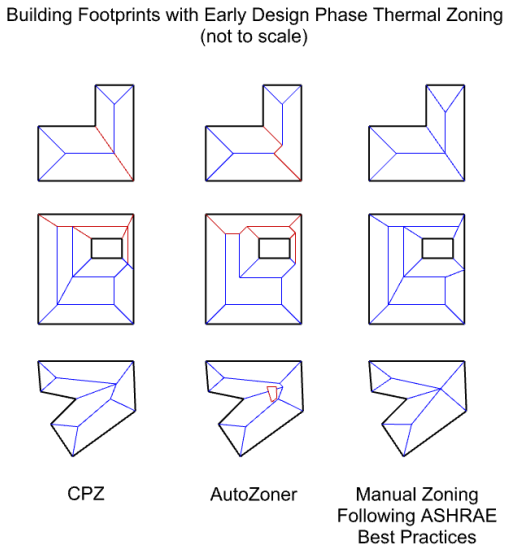

Figure 21: CPZ outputs compared with existing methods

We argue that the strength of CPZ lies not solely in its convex output. In the paper for AutoZoner (Dogan and Reinhart 2016), the authors propose to use a separate Convex Decomposition (CD) algorithm on the output of AutoZoner to create convex zones. While this may improve the simulation speed, it does not resolve the geometrical incompatibility between the SS and the offset, nor does it overcome SS's tendency to create little edges in narrow parts of the floor plates where the core zones collapse (Figure 21). These unnecessary

geometrical complexities slow down the simulation and can cause serious errors.

Future Work

This new algorithm, like the current state-of-the-art, utilizes topological properties of the floor plate to mathematically generate the zones disregarding the scale and visual shape of the generated zones. Consequentially, it faces two challenges that are difficult to overcome with the current methodology. First, when the input floor geometry has small details, especially ones that create concave corners, CPZ would still diligently divide them into tiny convex zones using all the tricks at its disposal even though they are too small to affect the simulation result. Second, since CPZ has no notion of the size of the zones generated, a hallway barely wider than twice the assigned perimeter depth would have a very skinny core zone that is unnecessary for the simulation.

Therefore, CPZ can greatly benefit from two other additions: first, a workflow performed before this algorithm that simplifies the floor geometry to remove minor edges; second, a workflow performed after CPZ that takes the generated thermal zones and merges away the zones that are overly small. Both proposed workflows are related to the actively researched field of polygonal simplification within computer graphics with applications in 3D modeling (Feciskanin, Richard, and Jozef Minár 2021; Vuillamy 2022).

An alternative is to use computer vision and neural networks (NN) to generate the thermal zones, a method that has the potential to overcome these challenges and be more robust. However, this alternative faces its own set of challenges: first, how to gather a large, high quality training set; second, since the current generation of NN works poorly with vector data (Li, Tzu-Mao et al 2020), we have to convert our vector geometries to raster, run it through the NN, then convert the raster result back to vector. The challenge is to not only generate clean vector results, but also to complete the whole process within a reasonable time.

Conclusions

This paper lays out the Convex Partition Zoner (CPZ) algorithm that utilizes an innovative iterative process to generate perimeter and core zones from floor geometries adhering to the ASHRAE Standard 90.1 appendix G method. Our evaluations demonstrate that CPZ is faster and less prone to error than the state-of-the-art, while its simulation results are close to that of human experts when instructed to create convex zones.

We plan to release our finished CPZ Grasshopper component in the future.

Appendix A: Algorithm Descriptions

Convex Wavefront Propagation Algorithm:

This is the algorithm for the WP stage. Refer to Figure 6, 7 and 8.

First, Create vertices along the boundaries (but not the partitions) for each individual subfloor. From now on consider only the convex subfloor pieces. No vertex is shared by two subfloors.

1. Move the vertices such that each wall propagates inwards at unit speed. Their propagated geometries are referred to as "wavefronts":
2. If the vertex is incident to two boundary segments, move in a direction and speed such that they both propagate at unit speeds.
3. If the vertex is incident to one segment and one partition, move along the partition at a speed such that the wall propagates at unit speed. If the wall and partition are aligned, the speed is infinite.

In other words, the speed and direction of each vertex is determined by its two incidental walls or partitions. Special case: if the angle formed by the two walls a vertex is incidental to is 180, this vertex is called a "following vertex", and its speed is determined by the vertex before and after it.

1. If a vertex is not moving along a partition, it leaves behind a "trailing wave". The trail is always straight because the floor is a polygon.
2. When a vertex moves along a partition and hits the end of it, if the other side is a partition, keep moving along that partition. Or if the other side is a wall, at this point this vertex must have been merged with another moving in the opposite direction because each subfloor is convex.
3. When two vertices hit each other, they merge into a single vertex. For two vertices to merge, they must share an incidental wave or partition. This merged vertex moves in a new direction and speed that is determined by the other two waves/partitions that are not shared. The old “trailing waves” of the two vertices stop growing and a new “trailing wave” for the merged vertex is created at the merge location.
4. When three or more vertices hit each other simultaneously, they merge into a single vertex, and it stops moving for the rest of the algorithm. This vertex is called a "stationary vertex". Because each subfloor is convex, if a “stationary vertex” exists it's always the only vertex left in the subfloor.

5. Keep moving the vertices with non-zero speed until each wall either has propagated the correct distance or has its corresponding wave merged.

Throughout the process, the waves stay parallel to their corresponding walls, because the only reason an unmerged wave cannot propagate further is that it overlaps with another wave propagating in a conflicting direction and their corresponding endpoints merge, so both endpoints would stop at the same time and the wave stays parallel with its wall the moment it stops propagating.

The zones created this way are always convex because:

- The subfloor is convex.
- All waves are parallel to the walls, so the polygon they form is convex.
- Therefore, it's impossible for a trailing wave to create a concave angle, because to create an angle > 180 the wave would be outside of the wall, which is impossible.

Adjustment 1-2 Algorithm:

This algorithm is one of a few used to mend together the fragmented wavefronts across the subfloors while maintaining convexity. Refer to Figure 9 and 10.

1. Find the intersection points between the neighbouring wave front segments propagating in the same direction along the same partitions.
2. If: both ends of the partition connect to boundaries of the floor plate, and any two points on the partition have never passed each other during the propagation: break the partition into three segments connecting its original endpoints and the intersections. Remove the immediate trailing wave of the wavefront vertex and extend or clip the remaining trailing waves to close the geometry.

This will never produce a concave zone because: For the partition segments connecting the boundary with the intersections, because the intersecting wavefronts are parallel to the boundary without overlapping, these partition segments always fall within the reversed cone. For the middle partition segment connecting between the intersections, because if the new segment lies on the reversed cone, the original opposite wave front would have merged at that moment. If a wave front on the other side as passed a wave front on this side, the second *if* condition above would have captured that.

Optimization Algorithm:

This procedure is meant to determine the optimized partition lines for the WP and AF process.

1. For each concave vertex, generate a list of valid partition candidates sorted by rank from high to low.
2. For the very first iteration, within CD, choose the highest rank candidate to decompose each concave vertex. Skip a concave vertex if it is already connected with another concave vertex.
3. Perform WP and AF, then calculate the cleanliness score. If the total score is zero (highest possible cleanliness score), return the current candidates as the solution. If the total score is below zero, find all partitions whose score is below zero and feed their indices back to the CD stage for the next iteration.
4. Starting from the second iteration, keep the choice of partition candidates from the previous iteration for all concave vertices. Then find the vertex corresponding to the first partition with sub-zero score. If the vertex is not yet marked as “resolved”, change its candidate to the one of the next highest rank. If this vertex ran out of candidates, switch to the highest ranked candidate that have achieved the highest score for this concave vertex in the optimization history and mark this vertex as “resolved”. If the vertex is already marked as “resolved”, find the vertex corresponding to the next partition with sub-zero score and perform the same operations. If all concave vertices with sub-zero scores are marked as “resolved”, return the current candidates as the solution.
5. Go back to step 3 and repeat.

Appendix B: Geometry-Related Errors and Warnings

For CPZ, all warnings are:

***CalcSurfaceCentroid: 2 Surfaces have the Z coordinate < 0*

For AutoZoner, all severe errors are:

***DetermineShadowingCombinations: There are _ surfaces which are casting surfaces and are non-convex.*

The warnings for Pixel Counting, O shape are:

***CalcSurfaceCentroid: 2 Surfaces have the Z coordinate < 0; DXFOut: **Could not triangulate surface="ZONE_0:F8", type="Roof";*

Plus 9 more “Could not triangulate” warnings.

The warnings for Pixel Counting, 8 shape are:

***GetSurfaceData: Entered Zone Floor Areas differ from calculated Zone Floor Area(s); **Entered Zone Volumes differ from calculated zone volume(s)*.

In addition, it has 11 more warnings almost identical with the ones for O shape.

For the manual case, all warnings are:

***CalcSurfaceCentroid: 2 Surfaces have the Z coordinate < 0*

References

Aichholzer, O., Alberts, D., Aurenhammer, F., & Gärtner, B. (1995, October). Straight skeletons of simple polygons. In Proc. 4th Internat. Symp. of LIESMARS (pp. 114-124).

Aichholzer, O., & Aurenhammer, F. (2005). Straight skeletons for general polygonal figures in the plane. In Computing and Combinatorics (pp. 117–126)

The American Society of Heating, Refrigerating and Air-Conditioning Engineers. (2022) ANSI/ASHRAE/IES Standard 90.1–2022. Energy Standard for Buildings Except Low-Rise Residential Buildings, Appendix G, 2022, pp 310

DesignBuilder Software Ltd. (n.d.). Simulation Calculation Options - Solar. DesignBuilder Help. Retrieved November 12, 2023, from https://designbuilder.co.uk/helpv7.0/Content/Solar_Options.htm

Dogan, T., Reinhart, C., & Michalatos, P. (2016). Autozoner: an algorithm for automatic thermal zoning of buildings with unknown interior space definitions. Journal of Building Performance Simulation, 9(2), 176–189.

Feciskanin, Richard, and Jozef Minár. "Polygonal Simplification and Its Use in DEM Generalization for Land Surface Segmentation." Transactions in GIS, vol. 25, no. 5, 2021, pp. 2361–75

Huber, & Held, M. (2011). Theoretical and practical results on straight skeletons of planar straight-line graphs. Proceedings of the Twenty-Seventh Annual Symposium on Computational Geometry, 171–178.

Li, Tzu-Mao, et al. "Differentiable Vector Graphics Rasterization for Editing and Learning." ACM Transactions on Graphics, vol. 39, no. 6, 2020, pp. 1–15

McNeel, R. (2020). Rhino3D (Version 7.n) [Computer software]. https://www.rhino3d.com/

Reinhart, C. F., & Cerezo Davila, C. (2016). Urban building energy modeling – A review of a nascent field. Building and Environment, 97, 196–202.

Roudsari, M. (2020, July 6). Honeybee for Grasshopper. GitHub. https://github.com/ladybug-tools/honeybee-legacy

Solemma LLC. (n.d.). ClimateStudio. Solemma. Retrieved November 13, 2023, from https://www.solemma.com/

U.S. Department of Energy (DOE)'s. (2023, September 29). EnergyPlus Version 23.2.0 Documentation, Engineering Reference. EnergyPlus. https://energyplus.net/documentation

U.S. Department of Energy (DOE)'s. (2023, September 29). EnergyPlus Version 23.2.0 Documentation, Input Output Reference. EnergyPlus. https://energyplus.net/documentation

Vuillamy, J., et al. "Simplification of 2D Polygonal Partitions via Point-line Projective Duality, and Application to Urban Reconstruction." Computer Graphics Forum, vol. 41, no. 6, 2022, pp. 379–93