KINETIC SURFACE RECONSTRUCTION

Jean-Philippe Bauchet, Florent Lafarge

Transactions On Graphics, Association for Computing Machinery, 2020

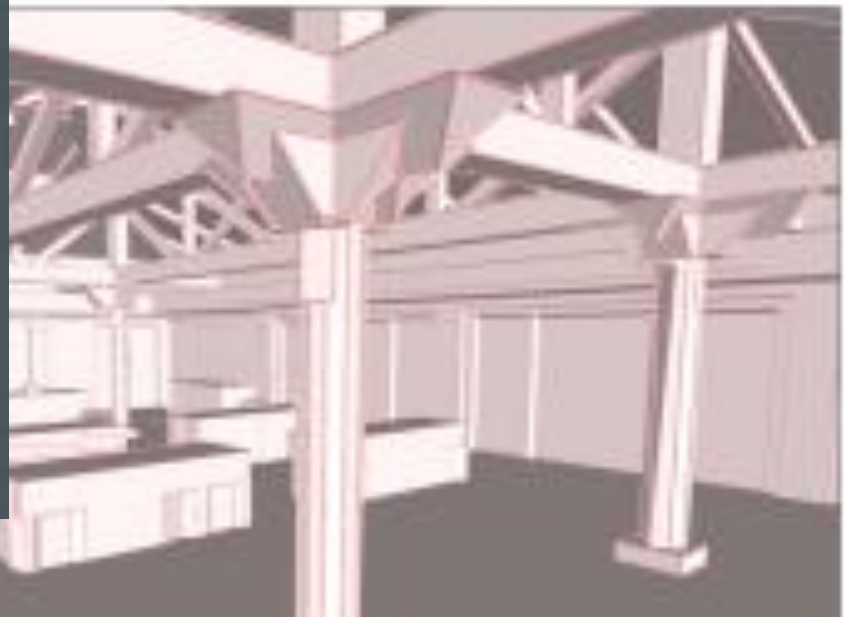
Professor:
Dr. Amal Dev Parakkat

Presented by:
Devaishi Tiwari (1701101021)

OBJECTIVE

Building a shape assembling mechanism to convert **unorganized 3D point clouds into concise surface polygonal meshes** with the following objectives:

- **Fidelity.** The output mesh must approximate well the input point cloud.
- **Watertight.** The output mesh is a watertight surface mesh free of self-intersections, bounding a 3D polyhedron.
- **Simplicity**. The output mesh is composed of a small number of facets, ideally just-enough facets given a user-defined tolerance approximation error.
- **Efficiency.** The algorithm should be fast, scalable and proceed with a low memory consumption.



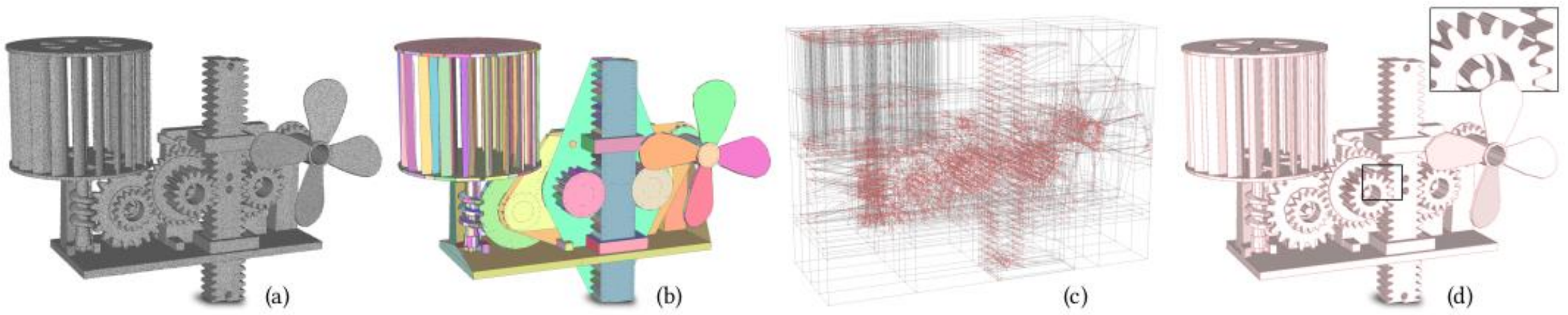
POINT CLOUD ⟹ SURFACE POLYGONAL MESH

Planar Shape Detection

Shape Assembling

Surface Approximation

Geometry Simplication



PLANAR SHAPE DETECTION

Planar shape detection consists in clusterIng the input point data into clusters of inlier points that lie near the same plane.

The common approaches operate by the following:

1. **Region Growing** propagates a plane hypothesis to neighboring points until fitting conditions are no longer valid.
2. **Ransac** proposes multiple plane hypotheses from some samples, verifies them against data, and selects the planes with the largest numbers of inliers.
3. **Learning approaches** trained from CAD model databases.

We will be using any of the mentioned methods for Shape Detection as our focus is on assembling already detected planar shapes into a watertight polygonal mesh.

SHAPE ASSEMBLING

Two main families of methods can be distinguished to assemble planar shapes into a polygon mesh,

1. Connectivity methods
   - In these methods, we **analyze an adjacency graph between planar shapes** in order to extract the vertices, edges and facets of the output mesh.
   - Although these methods are fast, they are likely to **produce incomplete models on challenging data** where adjacencygraphs often contain linkage errors.
2. Slicing methods
   - These methods operate by **slicing the input 3D space with the supporting planes of the detected shapes**. The output mesh is then extracted by selecting the polygonal facets of the cells that are part of the surface.
   - The main limitation is the **high algorithmic complexity** for constructing the 3D partition, commonly performed via a binary space partition tree.

SURFACE APPROXIMATION

Surface approximation is an alternative way to produce concise polygonal meshes by reconstructing a smooth surface from the input points and then simplifying it.

Some of the well-known methods for surface approximation are:

1. **Poisson Reconstruction method.**
2. **Edge Contraction**
   - In this method we keep contracting the edges until we reach a target number of facets.
   - To better preserve the piecewise-planar structure of objects, the edge contraction operators can account for the planar shapes detected in the dense mesh.

These surface approximation methods are in general effective, but they require as input a mesh that is both geometrically and topologically accurate to deliver faithful results.

GEOMETRY SIMPLIFICATION

Some work can further reduce the complexity of the mesh by imposing geometric assumptions for the output surface. For instance, the Manhattan-World assumption enforces the generation of polycubes by imposing output facets to follow only three orthogonal directions.

Some approaches also approximate surfaces with specific layouts of polygonal facets. Such polyhedral patterns are particularly relevant for architectural design.

Unfortunately, such assumptions are only relevant for specific application domains. Beside piecewise-planar, our algorithm does not rely upon any geometric assumptions and remains generic.

ALGORITHM: REQUIREMENTS

Input Requirements:

- Our algorithm requires as input a point cloud with oriented normals which is typically generated from MVS, laser scanning or depth camera.
- It also requires as input a configuration of planar shapes which is extracted from the point cloud by Region Growing or Ransac in our experiments.

Output:

- Our algorithm outputs a polygonal surface mesh whose facets lie on the supporting plane of the planar shapes.
- The output mesh is guaranteed to be watertight and intersection-free.

ALGORITHM: PROCESS

1. KINETIC PARTITIONING
   - The first step of our method consists in partitioning the bounding box of the object into polyhedra, each facet of a polyhedron being a part or an extension of the convex polygons.
   - This step relies on a kinetic framework in which convex polygons extend at constant speed until colliding with each other.

2. SURFACE EXTRACTION
   - The second step extracts the polygonal surface mesh from the partition by labeling polyhedra as inside or outside the reconstructed object.
   - We formulate it as a minimization problem solved by min-cut. Our objective function, called an energy, considers both the fidelity to input points and the simplicity of output models.

KINETIC DATA STRUCTURES

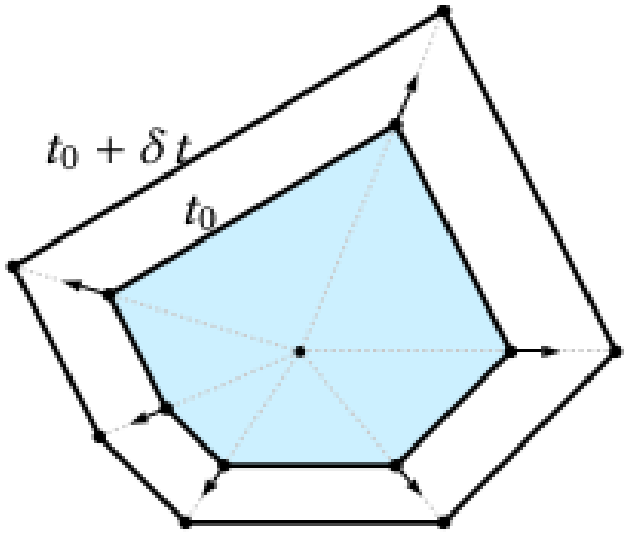
A Kinetic data structure is composed of a set of geometric primitives whose coordinates are continuous functions of time.

The purpose of kinetic frameworks is to maintain the validity of a set of geometric properties on the primitives, also called **certificates**.

When a certificate becomes invalid, geometric modifications are operated on the primitives to make the geometric properties valid again. This situation, also called an **event,** happens, for instance, when several primitives collide.

Kinetic frameworks show a strong algorithmic interest to dynamically order the times of occurrence of the events within a **priority queue**.

In our case, **the initial set of kinetic polygons is defined as the convex hulls of the planar shapes. Polygons grow by uniform scaling: each vertex moves in the opposite direction to the center of mass of the initial polygon.**

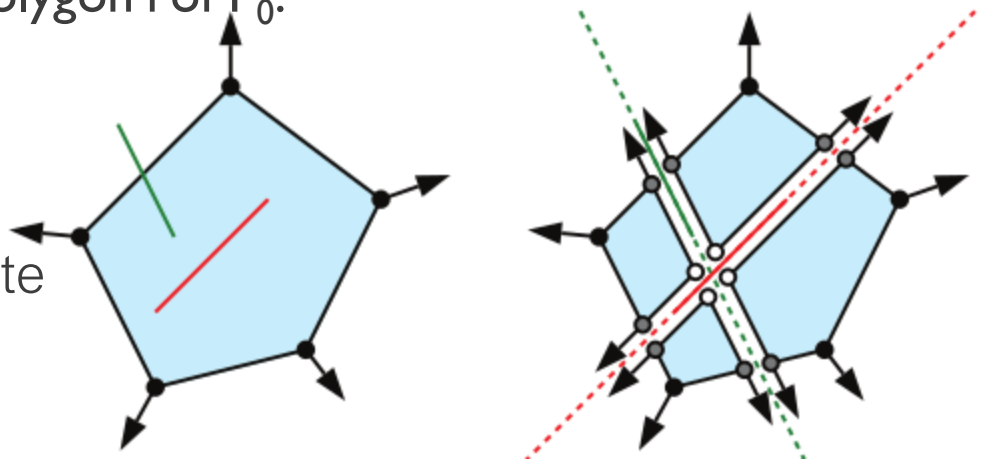


$$C_i(t) = \prod_{\substack{j=1 \\ j \neq i}}^{N} Pr_{i,j}(t)$$

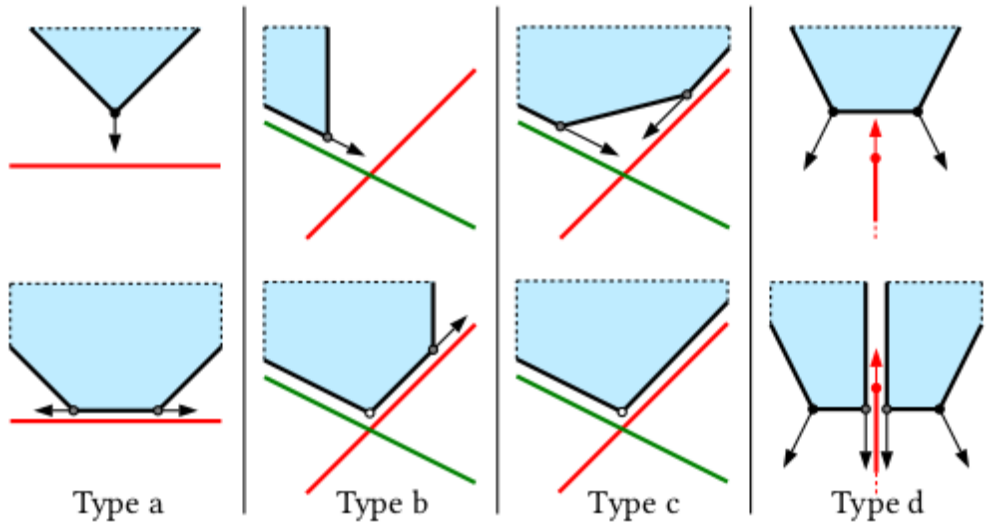
KINETIC PARTITIONING: INITIALIZATION

- Before starting growing the set of convex hulls, we first need to decompose them into intersection-free polygons. Thus, each non intersection-free polygon is cut along the intersection lines with the other polygons.
- The blue polygon intersects two other polygons, the intersections being represented by the red and green line-segments. We decompose it into intersection-free polygons with cuts along the intersection lines. Let the set of these intersection-free polygon be P_0
- **In addition, we populate the priority queue by computing and sorting in ascending order the times of collision, i.e. times for which certificates $C_i(t) = 0$ for each polygon i of P_0.**

The new vertices are either kept fixed when located at the junction of several intersection lines (white points) or are moving along the intersection lines (gray points). We denote them as **Frozen vertices** and **Sliding vertices**, respectively.



KINETIC PARTITIONING: UPDATION



a) **A vertex of the source polygon collides with the target polygon**: We replace the vertex by two sliding vertices that move along the intersection line in opposite directions.

b) **A sliding vertex of the source polygon collides with the target polygon:** We modify the direction of propagation of the sliding vertex to follow the intersection line and also create a frozen vertex where both the intersection lines meet.

c) **A sliding vertex of the source polygon collides with the target polygon:** We create a frozen vertex where the two intersection lines meet: the propagation of the source polygon is locally stopped.

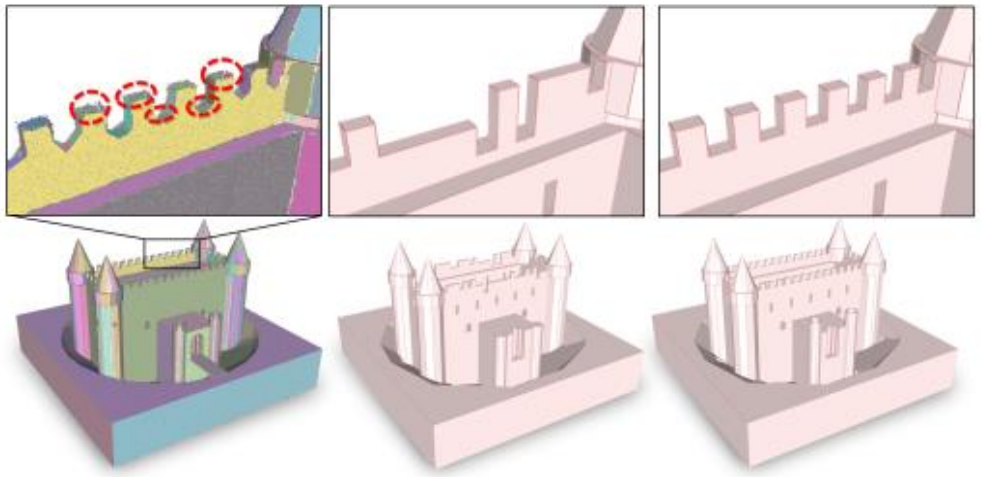
d) **An edge of the source polygon collides with an edge or vertex of the target polygon**: Source and target polygons are split along their intersection line with the creation of two sliding vertices per new polygon.

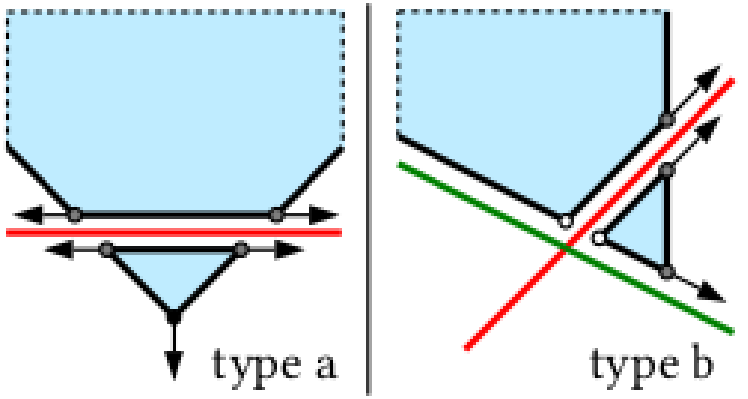
KINETIC PARTITIONING: UPDATION

- In many situations, it is interesting to extend the source polygon on the other side of the target polygon.

- We define the **crossing condition** which is valid of,
  1. the source polygon has collided several times lower than a user-specified parameter K or else if,
  2. relevant input points omitted during shape detection can be found in the other side of the target polygon.

- If the crossing condition is valid, a new polygon that extends the source primitive on the other side of the target polygon must be inserted into the kinetic data structure.





SURFACE EXTRACTION

We operate a min-cut to find an inside-outside labeling of the polyhedra, the **output surface being defined as the interface facets between inside and outside**. We propose a voting scheme that exploits the oriented normals of inlier points to more robustly assign a guess on a portion of the polyhedra only.

We denote by C, the set of polyhedra, and by $x_i = \{in, out\}$, the binary label that specifies whether polyhedron I is inside(x_i = in) or outside(x_i = out) the surface. We measure the quality of a possible output surface $x = (x_i)_{i \in C}$ with a two-term energy of the form

$$U(x) = D(x) + \lambda V(x)$$

where D(x) and V (x) are terms living in [0, 1] that measure **data fidelity** and **surface complexity**, respectively. $\lambda \in [0, 1]$ is a parameter balancing the two terms.

SURFACE EXTRACTION

DATA FIDELITY

Data fidelity D(x) measures the coherence between the inside-outside label of each polyhedron and the orientation of normals of inlier points.

$$D(\mathbf{x}) = \frac{1}{|\mathcal{I}|} \sum_{i \in C} \sum_{p \in \mathcal{I}_i} d_i(p, x_i)$$

where | I | is twice the total number of inlier points, I_i is the set of inlier points associated with all the facets of polyhedron i, and d_i (p, x_i) is a voting function that tests whether the orientation of inlier point p is coherent with the label x i of polyhedron i.

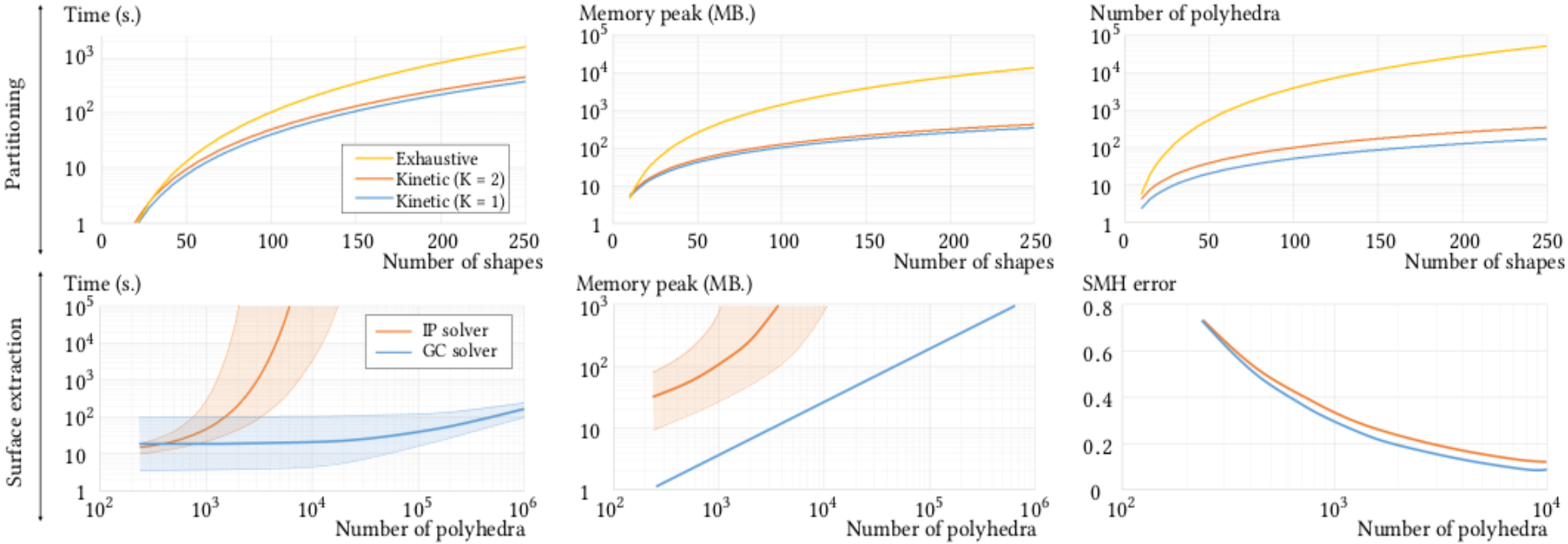
SURFACE COMPLEXITY

The second term V (x) is more conventional: it measures the complexity of the output surface by its area, where lower is simpler. It is expressed by

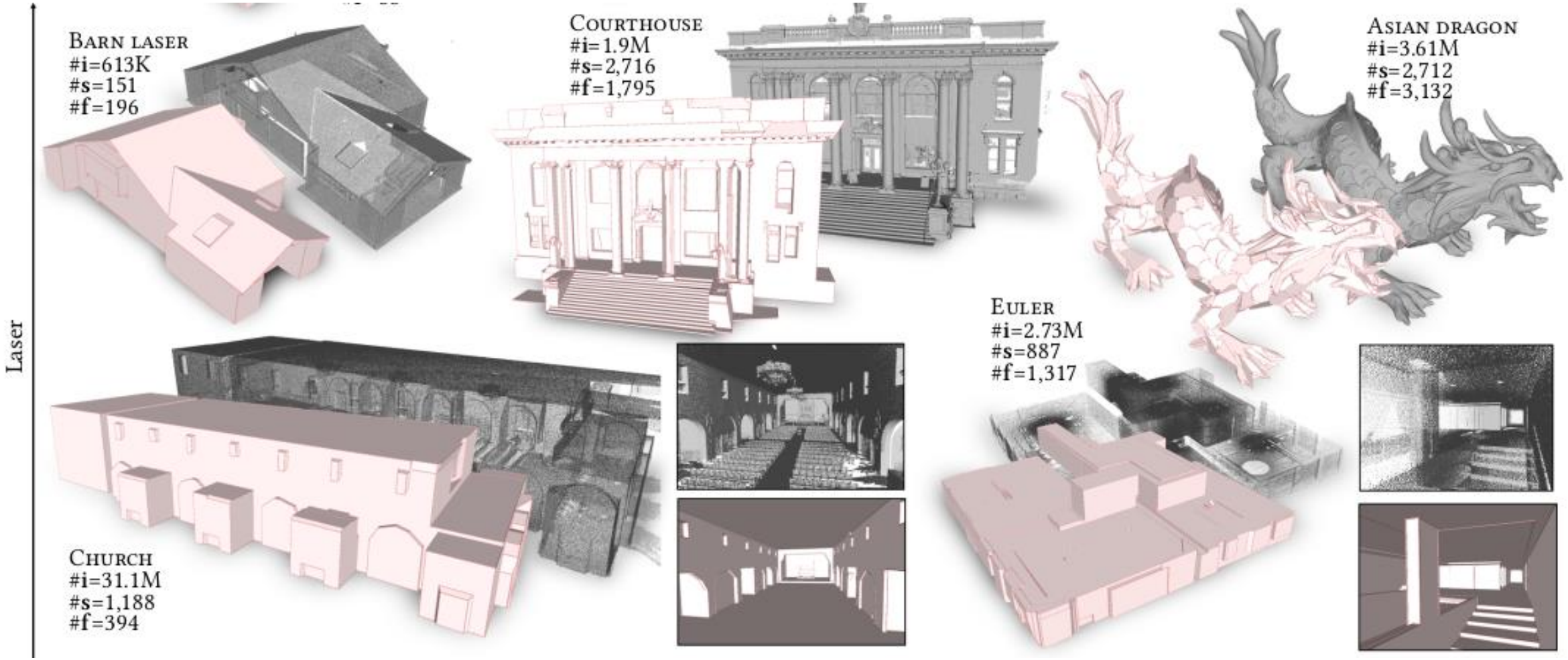
$$V(\mathbf{x}) = \frac{1}{A} \sum_{i \sim j} a_{ij} \cdot 1_{\{x_i \neq x_j\}}$$

where i ~ j denotes the pairs of adjacent polyhedra, aij represents the area of the common facet between polyhedra i and j, and A is a normalization factor defined as the sum of the areas of all facets of the partition.

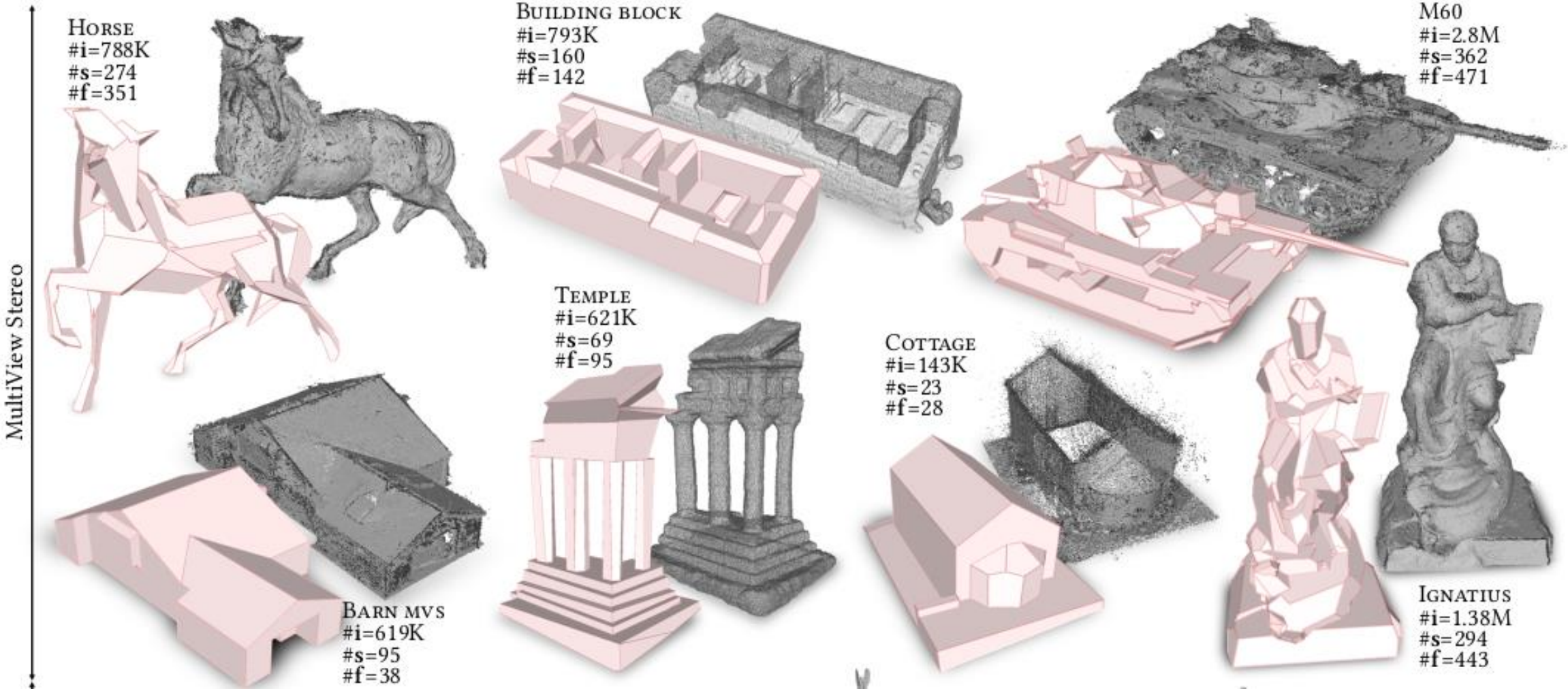
RESULTS: PERFORMANCE



RESULT : FLEXIBILITY

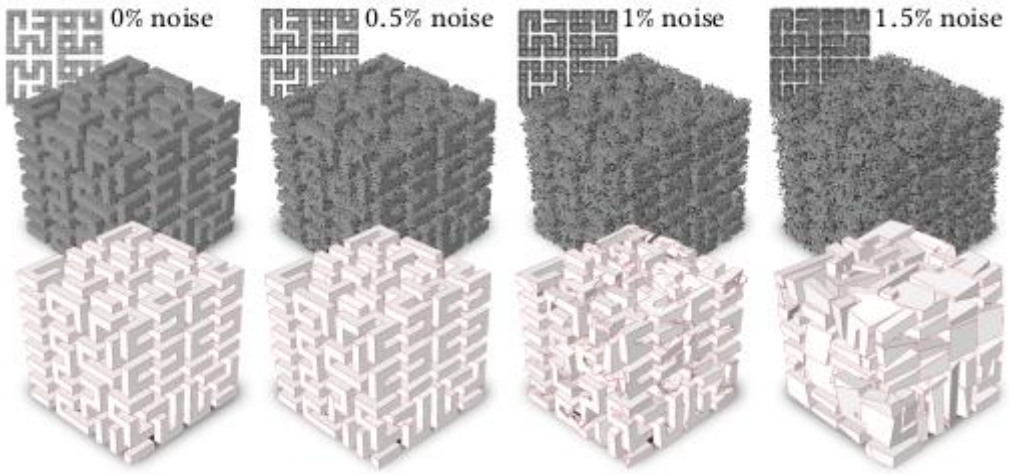


RESULT : FLEXIBILITY



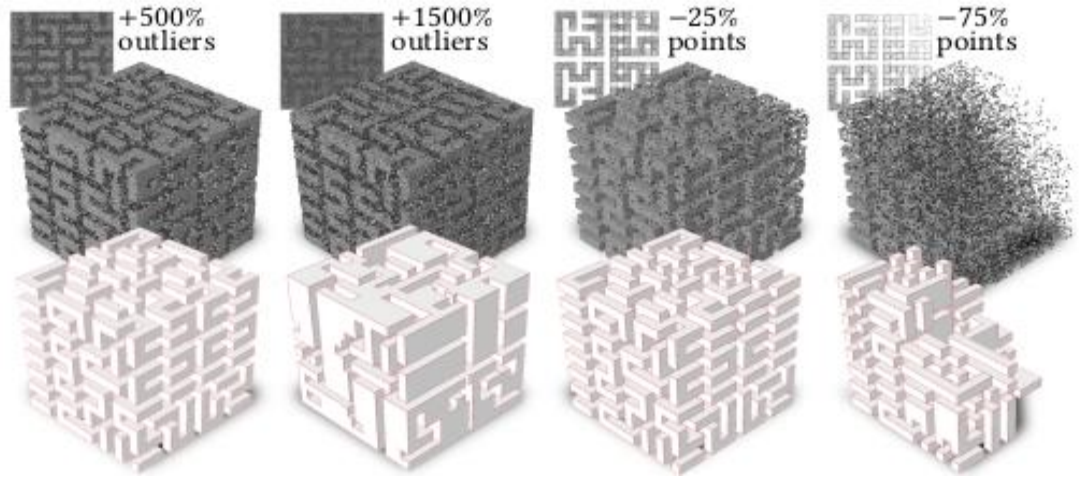
RESULT : ROBUSTNESS

ROBUSTNESS TO NOISE



Our algorithm is robust to noise if planar shapes can be decently extracted from input points. At 1% noise, some shapes become missing or inaccurately detected, leading to an overly complex output mesh. At 1.5%, the poorly extracted shapes do not allow us to capture the structure of the cube anymore.

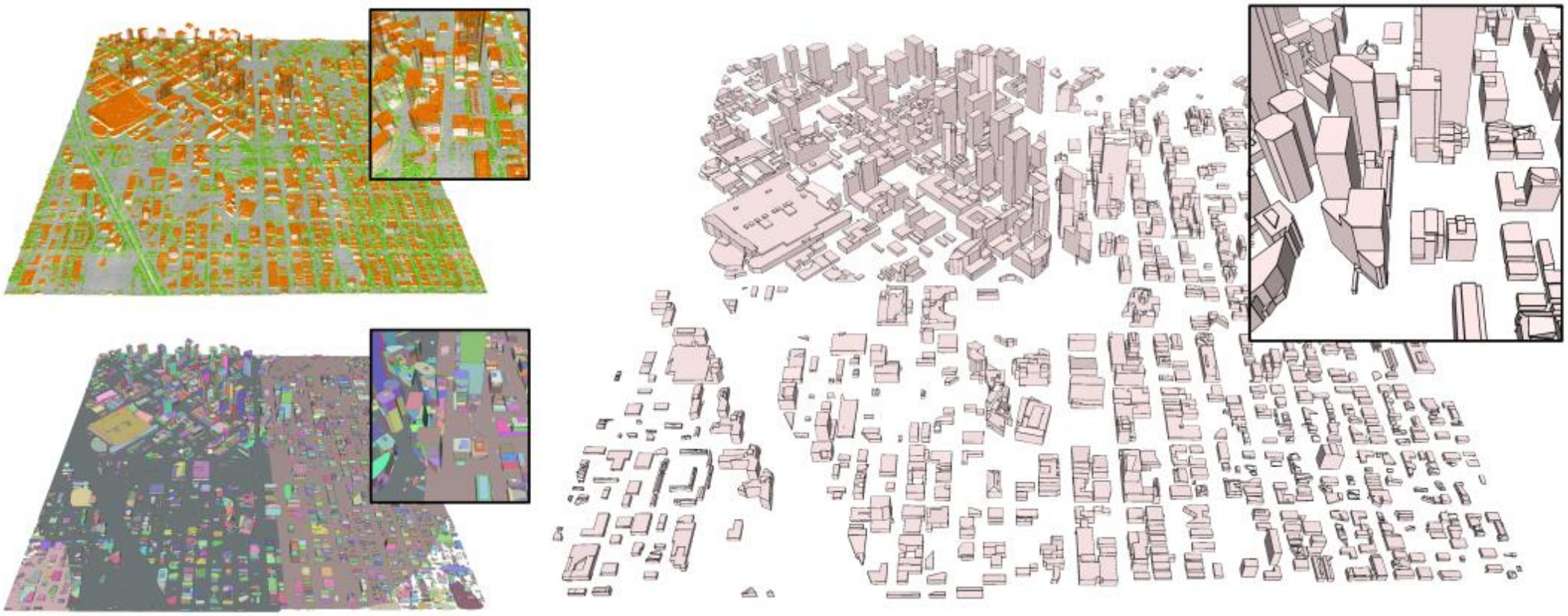
ROBUSTNESS TO OUTLIERS



Our algorithm starts compacting the structure around 1500% of added outliers. In the other two examples, the point clouds have been progressively sub-sampled. Our algorithm is resilient to such heterogeneous sampling if shapes can be retrieved in the low density of points.

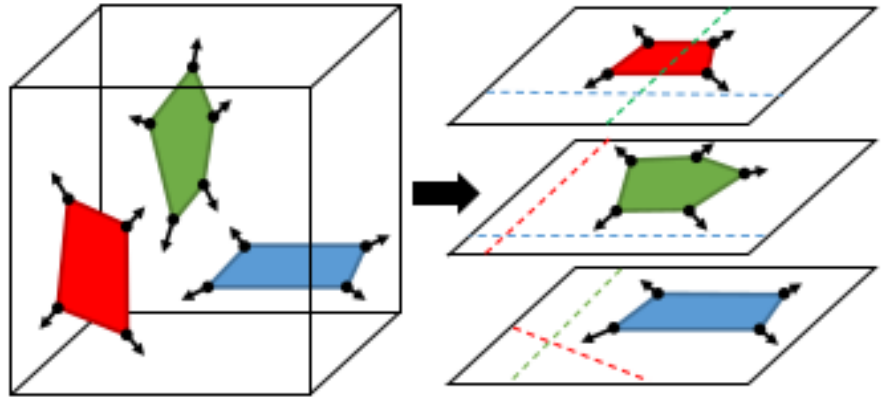
APPLICATION: CITY MODELLING

City modeling mainly consists in reconstructing buildings with a Piecewise-Planar Geometry.



IMPROVEMENTS IN IMPLEMENTATION

- **Reformulation as collaborative 2D collision problems:** Algorithmically reformulate the 3D collision problem as N collaborative propagations of polygons in 2D controlled by a global priority queue, N being the number of planar shapes.



- **Priority queue:** Restrict the number of collisions per vertex in the priority queue to three, to reduce memory consumption and processing time.

- **Spatial subdivision:** To increase scalability, subdivide the bounding box of the object into uniformly-sized 3D blocks in which independent kinetic partitionings are operated.

LIMITATIONS AND POSSIBLE SOLUTIONS

- Configurations of planar shapes returned by the algorithm can be inaccurate and incomplete, in presence of data highly corrupted by noise and severe occlusions.
  - **Shape detection methods including noise correction can be used.**
- Missing planar shapes can be accurately represented in configurations if they are a part of a repetitive structure.
- Our surface extraction solver is also not designed to reconstruct non-manifold geometry where some object parts have no thickness.
  - **Flexibility of using an integer programming solver can be provided in such cases.**
- The initialization of non-convex shapes by their convex hulls can create extra cuttings.
  - **Build even lighter partitions and thus increase the number of polygons.**

FUTURE WORKS

- In future works, we can generalize the kinetic partitioning algorithm to non-planar shapes using NURBS and other parametric functions.
- Although the collision detection for such shapes is even more challenging to compute efficiently, this perspective would allow us to reconstruct freeform objects with a better complexity-distortion tradeoff.

THANK YOU