Report, Kinetic project

Dorian Geraldes Pereira, Axel Demuth

March 2024

Contents

1 Project Context

Our project is part of a larger initiative aimed at processing IFC files representing buildings to create 3D models for energy simulations with high-performance calculations. Within this project, there is a need to complete missing data in the IFC file to enable simulations of energy consumption and losses.

2 Project Objective

Within this project scope, the primary goal is to implement an efficient and accurate conversion process for Industry Foundation Classes (IFC) files representing buildings or cities into meshes compatible with the Kinetic algorithm. Subsequently, the Kinetic algorithm will be applied to these meshes to produce watertight models, facilitating the execution of finite element calculations.

The specific steps to be undertaken are as follows:

1. **Reading Process or Mesh Conversion :** From the IFC files we will have a conversion in the stl or msh format,we will need to change the reading process to be able to read mesh from those format if possible,if we cant then we will convert the STL or MSH meshes into one of the formats accepted by the Kinetic algorithm, such as .ply, .xyz, .las, .off.

2. **Application of the Kinetic Algorithm:** Applicate the Kinetic algorithm on the the converted meshes to produce meshes optimized for finite element calculations.

3. **Recovery of Material Labels:** Ensure the preservation of information regarding materials present in the initial IFC-format mesh and correctly associate them with elements of the converted mesh.

4. **Utilization on City Modeling:** Extend the application of the Kinetic algorithm to entire city models.

3 Current Project Challenges

Currently, the project faces several technical challenges:

1. **Reading Process or Mesh Conversion:** we need to see if its possible to replace the read_point_set function by a read_STL function producing a Point range instead of Point set if we cant we will need to find a solutions to convert meshes from STL or MSH files into one of the formats accepted by the Kinetic algorithm.

2. **Parameter Optimization:** Identify and adjust appropriate parameters to avoid segmentation faults and achieve satisfactory results when applying the Kinetic algorithm.

By overcoming these challenges, the project aims to provide a comprehensive and efficient solution for analyzing urban structures using the Kinetic algorithm to facilitate finite element calculations.

4 Tools

4.1 CGAL

CGAL is a comprehensive package for geometry algorithms, providing various data structures and algorithms for working on polygons, surfaces, mesh generation, and more. It offers a wide range of functionalities for geometric processing and analysis in various fields such as computer graphics, computational geometry, and geometric modeling.

4.2 IFC project

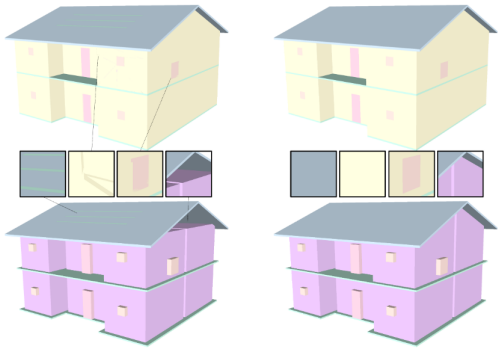
The initiative was created by buildingSMART (BIM), with the goal of supporting portability between software in the construction sector. IFC is a file format designed to replace a fragmented information system with a standardized one. This enables every actor in the sector to utilize IFC-compatible software to open files and model their contents without wasting time and computational resources on convertion between different formats.

The IFC format is based on object-oriented code, with numerous classes associated with various needs such as construction sites, tools, and materials. However, a challenge we face is the difficulty in interpreting IFC objects for use in Kinetic programs. While there are several software solutions available to convert IFC to the desired format, we often lose significant amounts of information in the process.

As a result, we need to focus on refining the objects received from the converting process to ensure that Kinetic algorithms can effectively read the files.

4.3 Kinetic

Kinetic algorithms(KSR) is a package from CGAL that allows working on meshes with some holes in them. When applied to the mesh, the Kinetic algorithms will 'extend' some surfaces to fill the mesh and make it watertight. Here's what the algorithm is capable of:



We can obtain a comprehensive understanding of how it works in general from the report by INRIA [3].

- It uses geometric primitives to create planes due to their relevance in human-made environments.
- There are multiple methods for plan fitting in a 3D model, such as Neural Network architectures or Energy-based Models. The algorithm employs the latter.
- To evaluate the quality of a primitive configuration x with an energy U of the format
  $U(x) = w_f U_f(x) + w_s U_c(x) + w_c U_c(x)$
  where all U functions pertain to fidelity, simplicity, and completeness, and all w are positive and weight.
- It then utilizes geometric operations such as merging, splitting, transfer, insertion, and exclusion on different planes and closed primitives.

Here, we present pseudocode detailing how the exploration of the set of primitives works:

Algorithm 1 Pseudo-code of the exploration mechanism

```
Initialize the primitive configuration x
repeat
    Initialize the priority queue Q
    while top operation i of Q decreases energy U do
        Update x by operation i
        Update Q
    end while
    Update x by the global transfer operator
until no update modifies x any more
```

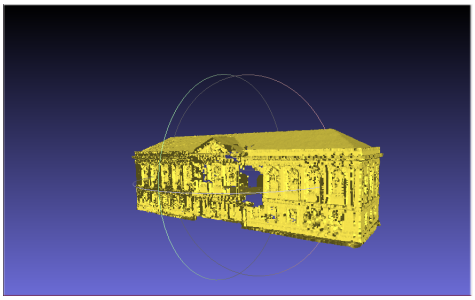
The priority queue represents the set of primitives to which we apply geometric applications.

To use the algorithm, we employ one from CGAL. Given a set of parameters and a file containing a point cloud along with the associated normals to the points, the algorithm is applied. We were fortunate to have a meeting with Florent Lafarge, one of the creators of the algorithm, who explained to us which parameters are crucial for analysis and how each one can significantly influence the results. Two parameters stand out as particularly important:

- 'dist' indicates the distance between two points required to consider them for plane construction.
- 'pmin' represents the number of points used to construct a plane.

In order to understand how these parameters affect the outcome, we will examine the same point cloud while varying the value of 'pmin' first and then varying 'dist.' The algorithm produces .off files as output, which can be visualized using software such as MeshLab.

The original point cloud represents this building:



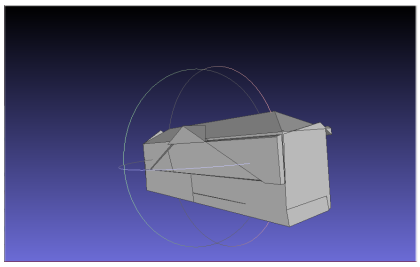


Figure 1: dist1_pmin220

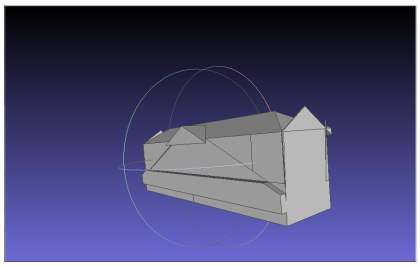


Figure 2: dist1_pmin250

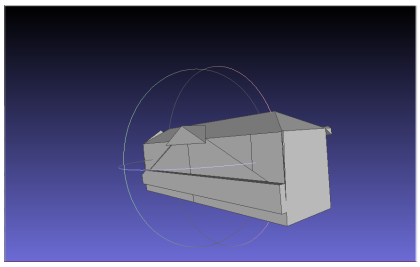


Figure 3: dist1_pmin280

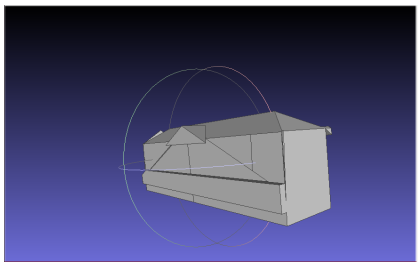


Figure 4: dist1_pmin300

First, it's important to understand that setting 'pmin' too low can lead to errors. Subsequently,if 'pmin' is too small, we may not capture enough structural detail, whereas setting 'pmin' too high may cause planes to overlap, resulting in loss of information

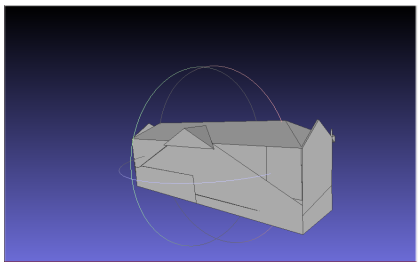


Figure 5: dist15_pmin250.png

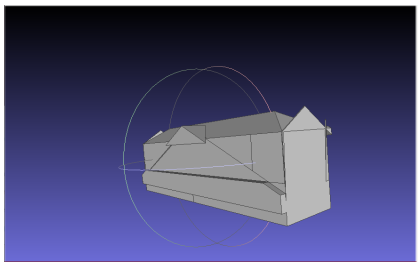


Figure 6: dist1_pmin250

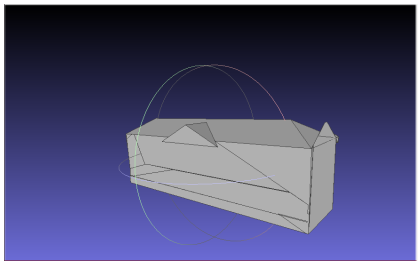


Figure 7: dist03_pmin250

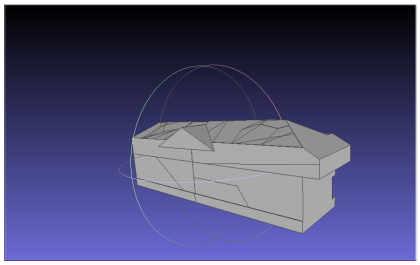


Figure 8: dist001_pmin250

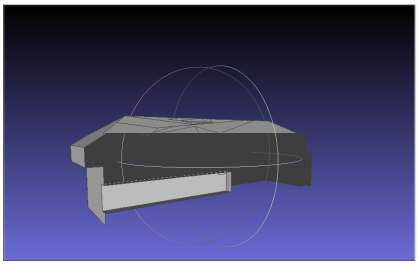


Figure 9: dist001_pmin250

When considering the 'dist' parameter, if it is set too high, we risk losing significant structural details. In the opposite, if the distance is too small, surfaces may be divided into too many planes, potentially resulting in an incomplete coverage of the entire mesh, as observed at the rear of the building with a 'dist' of 0.001.

One of the challenges in studying this algorithm is determining the appropriate 'qmin' and 'dist' to achieve the desired number of space partitions. Setting it too high may enhance precision but extend execution time, and in some cases, incorrect values can lead to program crashes.

This algorithm could be a crucial tool in our project, particularly when dealing with massive meshes for large buildings, hospitals, etc. When creating large meshes, issues can arise, and errors within the mesh can be detrimental during simulations,potentially leading to false results. This algorithm allows us to rectify such mesh problems.

One limitation of the algorithm is its input requirement, as it currently only

works with scatter plots. Consequently, we are unable to utilize the IFC format, resulting in the loss of valuable information regarding the types of structures present. To address this limitation, we plan to implement a process involving the conversion of IFC to scatter plot with associated data, followed by conversion to formats such as .plt, .ply, .xyz, which are supported by Kinetic CGAL, ultimately resulting in a mesh with associated data.

5 Implementation

We are using the most complete example of the cgal kinetic algorithm as base for our main. In this main we can change every parameters of the algorithm.
To use the algorithm we currently need to give him a file from supported format such as .ply , .off, .xyz, .las . We will try to change it to be able to read STL and MSH format. If the parameters dont respect the critera we described in the previous section the algorithm can end up in segmentation fault. The result of the algorithm will be in an .off format in "resultat" folder.
We can present an example of the command to execute the program:
" ./build/default/bin/kinetic -data data/flame.ply -dist 0.3 -minp 50 -regangle 5 "

5.1 STL Reading and Use

In the CGAL library, we have methods to read various file formats such as ply, off, xyz, and there is one for STL files. If we use the read_stl function, we get the following output:

- a list of all points used in the mesh
- a list of vertices with all points used in every polygon

So, we have access to a point cloud and a list of polygons. The KSR algorithm only needs a point cloud to work. We then proceed to calculate the normals associated with all points, write all the points and the normals into an .xyz file, then read the .xyz file with our main program to apply the KSR algorithm. However, one issue with STL files is the low number of points given by the files, so the algorithm doesn't work well with that, and an error is generated. As a result, we need to find a way to densify our point cloud.

To execute KSR on our files we tried different option:

- use a free open sourced software as CloudCompare
- use CGAL method to densified our point cloud with function as CGAL poisson surface reparation.

5.2 CloudCompare

We used CloudCompare to create a densified point cloud with an stl file as basis. This software can also calculate normals of each point, we used it sometimes to see the differences with cgal and meshlab because the normals of the point are very important to get the best result possible

5.3 CGAL method

CGAL has multiple way to generate point cloud in theory, for example:

- $CGAL :: Spatial\_sort\_traits\_adapter\_3, CGAL :: natural\_neighbor\_coordinates\_3()$, Interpolate method like moindres carre or spatiale interpolation
- $CGAL :: Delaunay\_triangulation\_3 Delaunaytriangulation$ allow to creat a point cloud on a triangle
- $CGAL :: point\_generators\_3 :: regular\_grid\_points()$ generation on a grid

If we succeed to generate a big enough point cloud with one of those function, we could try to write it in some off,xyz,ply files to read them and try to assemblate the mesh and to get the result we want.

6 Analysis of results

6.1 first results

Analysis of the meshes reveals significant differences between those produced by the CGAL algorithm and those generated by the INRIA algorithm. The mesh obtained with CGAL demonstrates satisfactory watertightness but is characterized by noticeable roughness. Conversely, the mesh generated by the INRIA algorithm is remarkably smoother, offering a more uniform surface.

For better visual understanding, three images are provided below:

- Initial point cloud
- Result of the mesh with the CGAL algorithm
- Result of the mesh with the INRIA algorithm

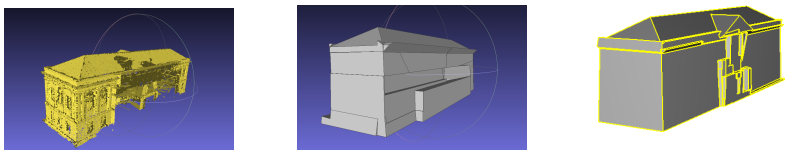


Figure 10: Visualization of results with a building

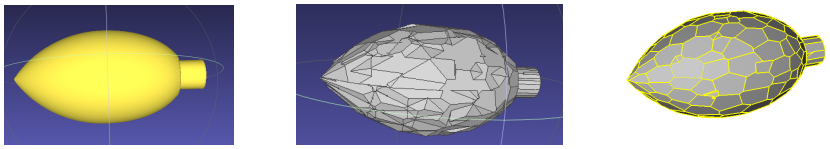
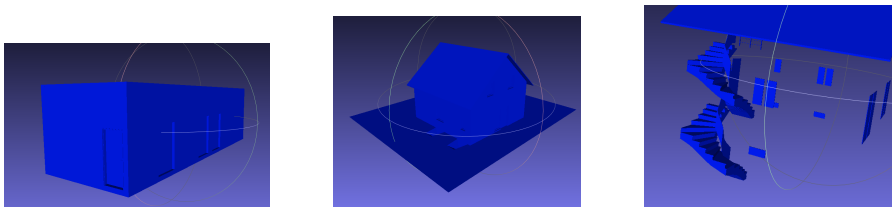


Figure 11: Visualization of results with a flame

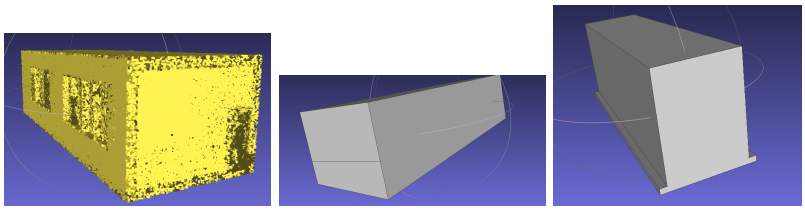
6.2 Final Result

For the final analysis we worked on 2 of the example provided by Vincent Chabannes:



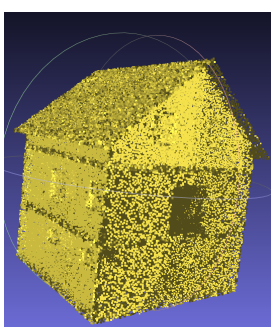
3zones ACJasmin ACJasmin_inside

For the first example, 3zones, we used CloudCompare to create the PLY point cloud file for the CGAL algorithm, and we calculated the necessary normals in Meshlab, which gave us the best results. However, the resulting watertight mesh lost all the details from the STL file, such as the windows and doors, which have disappeared.

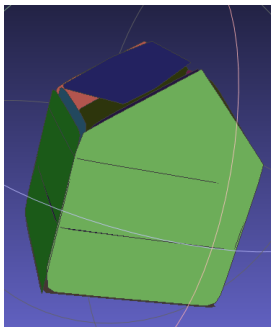


3zones point cloud 3zones result normal by MeshLab 3zones result normal by cgal

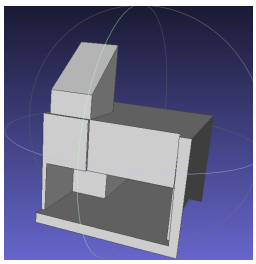
For the second example, ACJasmin we used the same protocol as the first example but for this example we decided to remove the ground around the house because it created a lots of problems, the result we got were a lot more disapointed than the first example. The mesh we got wasnt watertight and we lost nearly every detail like the first example just a bit of the stairs still remained This is the result we got from the cgal version:



ACJasmin point cloud

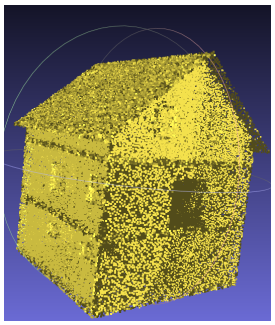


ACJasmin primitives by cgal



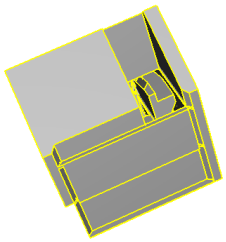
ACJasmin result by cgal

After the cgal version we tried the IRNIA version the result was still disappointing but better:



ACJasmin point cloud

ACJasmin primitives by INRIA



ACJasmin result normal by INRIA

7 Roadmap

At this point those are the main Issues that we will need to work on:

- Change the reading process to accept stl or msh format file or if its not possible convert PLT or MSH meshes into a format compatible with the Kinetic algorithm.
- If possible, be able to retrieve the labels on the mesh generated by the Kinetic algorithm.
- Quality check of the obtained mesh from the kinetic algorithm (same volume as the first mesh)
- Apply physical theories to the base mesh and the output mesh to analyze the differences between them.
- Investigate the feasibility of applying the algorithm on a larger scale, such as a city or a large building, instead of a basic one.

For the V1 we modified and completed some issues:

- The Issues Create watertight mesh with kinetic we completed what we wanted to do with this issue but now we want to go further so we modified it to create watertight mesh from files coming to IFC format for the V2
- The issue Convert IFC to PLY format was also changed to change the reading process or convert MSH and STL to supported cgal format for the V2
- The issue setup programming environment for CGAL with github action and submodul,the creation of the main program,the study of the kinetic program were completed

8 Reference

References

[1] Jean-Philippe Bauchet and Florent Lafarge. Kinetic Shape Reconstruction. *ACM Transactions on Graphics*, 2020.

[2] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 5.6.1 edition, 2024.

[3] Mulin Yu and Florent Lafarge. Finding Good Configurations of Planar Primitives in Unorganized Point Clouds. In *CVPR 2022 - IEEE Conference on Computer Vision and Pattern Recognition*, La Nouvelle-Orléans, United States, June 2022.

[4] Mulin Yu, Florent Lafarge, Sven Oesau, and Bruno Hilaire. Repairing geometric errors in 3D urban models with kinetic data structures. *ISPRS Journal of Photogrammetry and Remote Sensing*, 192, October 2022.