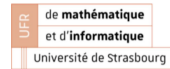


# KINETIC AND BUILDING LOD2



Intern: Demuth Axel

Supervisor: Vincent Chabannes, Pierre Alliez, Florent Lafarge

# Table of Contents

## 1 Introduction

- Context
- Issue with Kinetic Algorithm
- Objectives
- CGAL

## 2 Data

- Files Format
- Software and Data

## 3 Methodology

- Kinetic
- preprocessing

- Labelling

- Metric

## 4 Implementation

- Contribution to Ktirio library
- test

## 5 Result

- Point cloud generation Result
- Self Intersection Result
- Performance

## 6 Conclusion

## 7 References

# Introduction

- Need of more sophisticated tool for energy simulations
- As a part of Exa-MA project from Numexp : CEMOSIS want to port Ktirio application to exascale power
- My objectives was to developed a geometric reconstruction tools of building using Kinetic algorithm

# Context

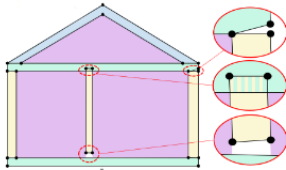


Figure: Mesh with issues

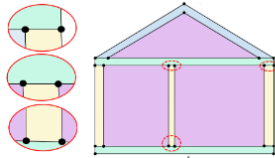


Figure: Mesh without issues

## Issue with orientation

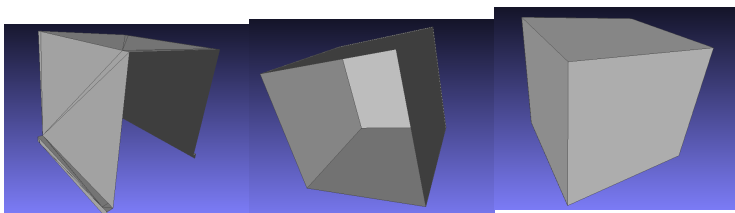


Figure: Cube not oriented

Figure: Cube badly oriented

Figure: Cube oriented by CGAL

# Self Intersection issue

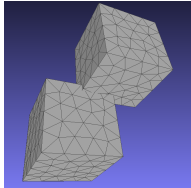


Figure: Two cube self intersecting

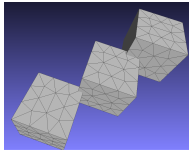


Figure: Two cubes intersecting a third one

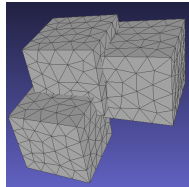


Figure: Three cubes self intersecting

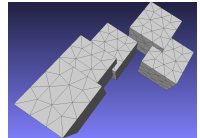


Figure: Five cubes intersecting randomly

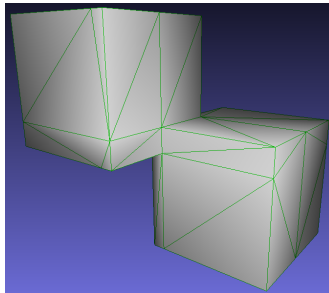


Figure: Two Cubes fixed

All other result in a execution error

# Objectives

- Check the validity of the Mesh
- Create a workflow for automatic generation using KSR Algorithm
- Keep the correspondence of surfaces between both meshes
- Run some simulations using the Feel++ library



# CGAL

- C++ library for geometric calculations, providing data structures for mesh generation and manipulation.

The main packages utilized are:

- `CGAL::Polygon_mesh_processing`
- `CGAL::Surface_mesh`
- `CGAL::Point_set_processing`
- `CGAL::IO_streams`
- `CGAL::AABB_tree`

# File Format

- IFC : Standart for buildding data modeling,similar to class oriented code
- CityGML : 3D format for city modeling with representation of geographic details
- STL : 3D Modeling format
- OBJ :A standard file format for 3D models
- OFF : A file format for 3D mesh data
- PLY : A file format for 3D mesh data,stocking the cloud point of the mesh
- MSH : A file format for mesh data use by GMSH software

# Software

- Github : Platforme for collaborating work on a project
- Visual Studio Code : Versatil tools for coding with various extensions
- Paraview : Open-source data analysis and visualisation
- Meshlab : A tool for processing,editing,visualisation of 3D mesh
- GMSH : a 3D finite element mesh generator

# Data

The following Data were given by Vincent Chabannes

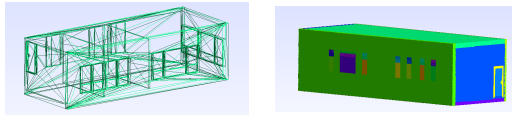


Figure: Three zones mesh

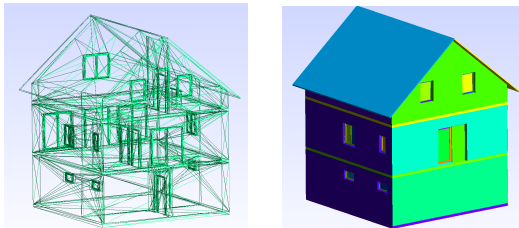


Figure: ACJasmin mesh

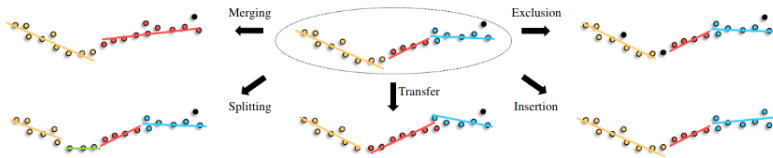
# Kinetic

We get information from a INRIA report [1] Kinetic algorithm is an geometric algorithm generate 3D mesh from a point clouds,it uses geometric primitive with an energy based model to fit the primitives to the model.

Energy formule:

$$U(x) = w_f U_f(x) + w_s U_s(x) + w_c U_c(x)$$

to calculate the best primitive  
to fit the mesh. then we have a list of geometric operation on each primitive



# preprocessing

To improve Kinetic outcome we pre-process the mesh :

- Isotropic remeshing of the mesh
- Unified and regularize the mesh with grid simplify
- Fix self Intersection
- Calcul normals

# Labelling

**Issue:** Inria developed a method to preserve the semantic information of IFC elements, but it has not yet been implemented in CGAL.

Two potential solutions:

- Modify the Kinetic Solver to recognize and utilize markers on each point used to form a shape.
- Compare the input and output meshes to apply the same markers to the closest faces.

# Labelling

Exemple of result of second solutions:

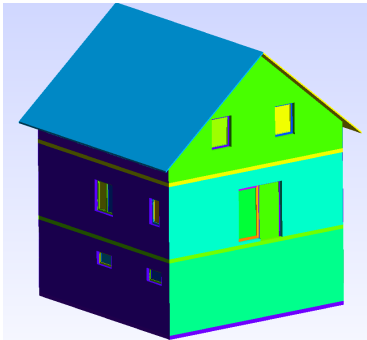


Figure: Input Mesh

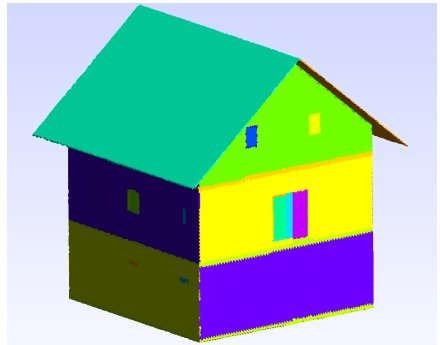


Figure: Output Mesh



# Metric

We also want to add method to check the quality off the output mesh

- Properties Check (closed,connected,triangulated...)
- Correspondance between input and output

To check the Correspondance between mesh, we can compare bounding box of each labelled elements.

Table: Bounding Box value

% of marker correct	Three Zones	ACJasmin
<5%	22/57	3/82
between 5 and 10 %	11/57	7/82
between 10 and 20 %	13/57	9/82

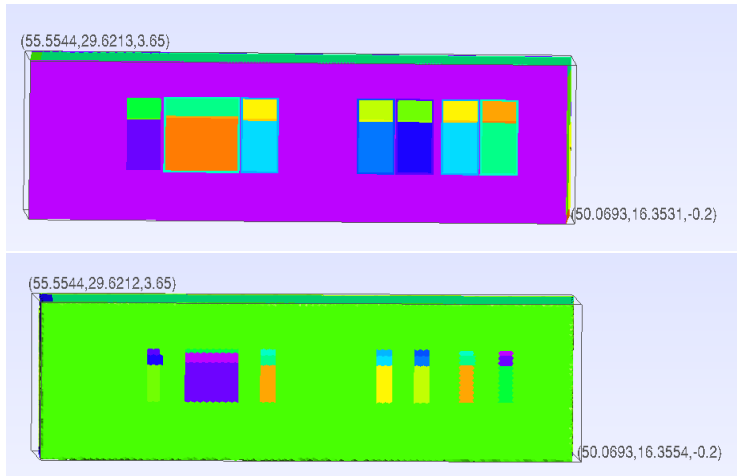


Figure: Three zones Bounding Box comparison

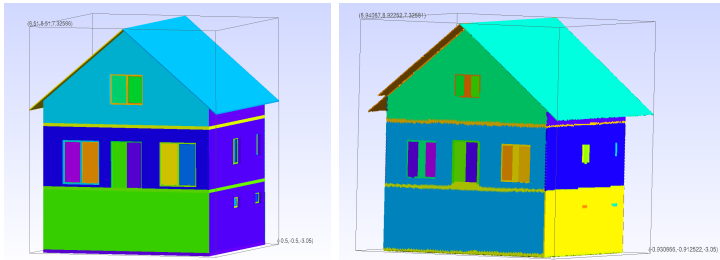


Figure: ACJasmin Bounding Box comparison

- checkProperties
- gridSimplify
- remesh
- KSR
- IO function for off,ply,obj Files
- Point set class
- etc..

# test

- test on Surface Mesh Check
- test on Kinetic algorithm
- test on Point set class and manipulation function

# Point cloud

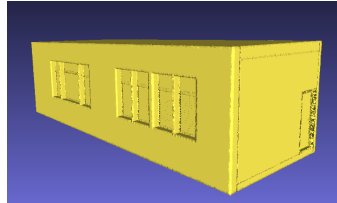
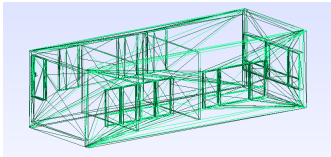


Figure: Three zones mesh point cloud

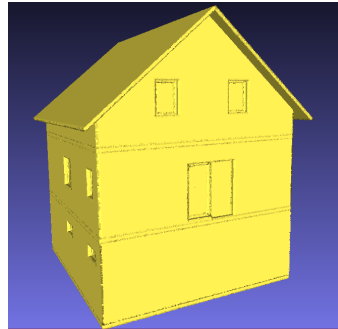
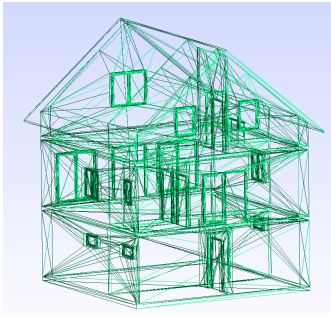


Figure: ACJasmin mesh point cloud

# Comparison of Kinetic Outcome

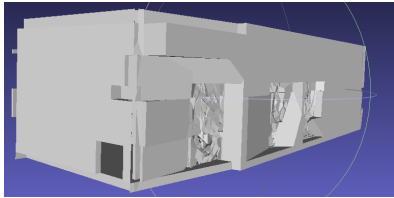


Figure: Old KSR outcome

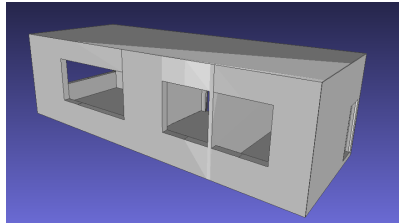
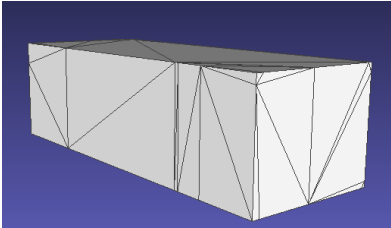


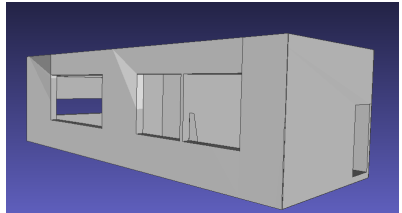
Figure: New KSR outcome



## Comparison of 3 Zone Results



**Figure:** Three Zone min region size = 500



**Figure:** Three Zones min region size = 45

## Comparison of Jasmin Results

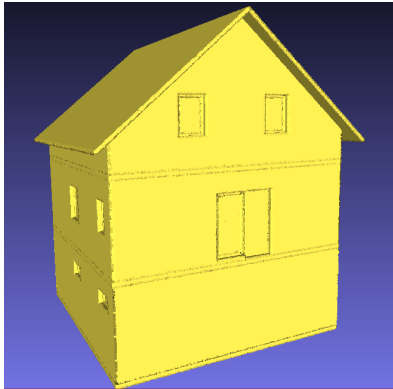


Figure: Jasmin Ply

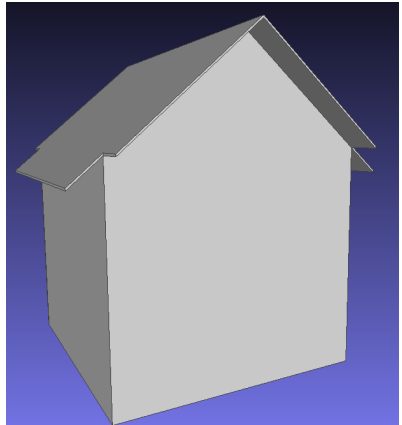


Figure: Jasmin Image

# Self Intersection fixing

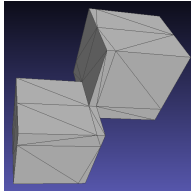


Figure: Same  
result as intro

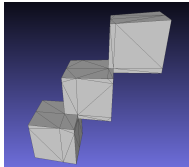


Figure: Worked

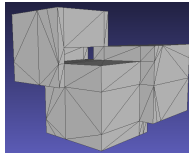


Figure: Worked

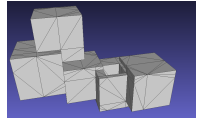


Figure: Worked

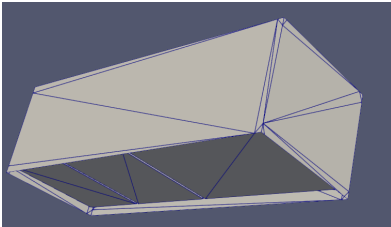


Figure: Refined Zones

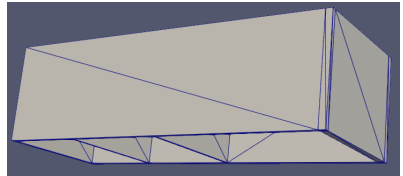


Figure: Not Refined Zones

## Comparison of Refined and Not Refined Zones

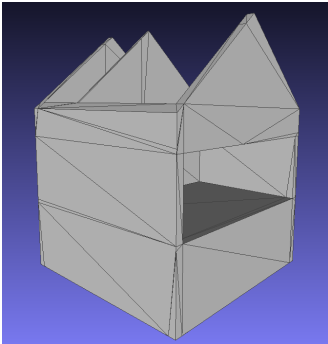


Figure: ACJAsmin Refined

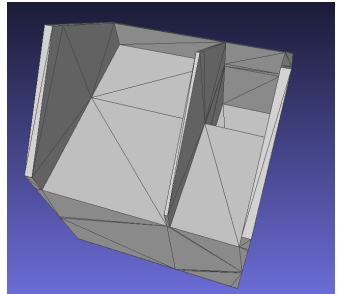


Figure: Jasmin Refined roof

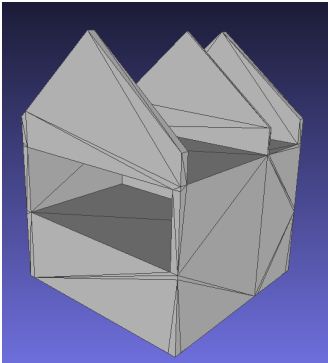


Figure: ACJAsmin Not Refined

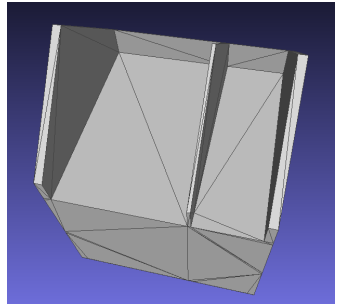


Figure: Jasmin Not Refined  
Roof

## Comparison of ACJASMIN Results

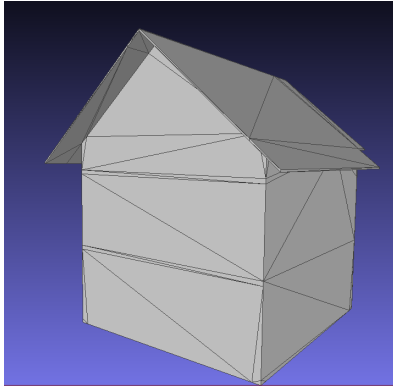


Figure: ACJASMIN Refined

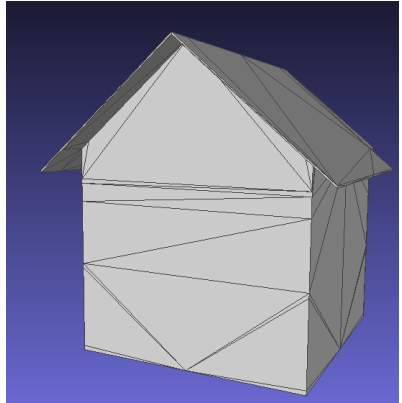


Figure: ACJASMIN Not Refined

## Comparison of performance

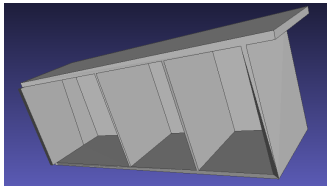


Figure: Execution times 125s

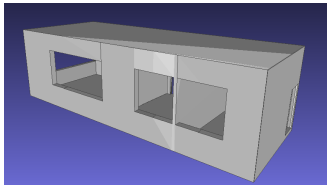


Figure: Execution times 3s



# Execution Time with Our Workflow on Three Zones

Parameters	Default	minp=2000	minp=500	minp=100
Shape detection	7.97s	8.18s	5.76s	1.62s
Kinetic space partition	0.22s	0.22s	0.36s	1.35s
Total execution	8.2s	8.41s	6.13s	2.98s

**Table:** Execution time with our workflow on Three Zones





Table: Execution time on ACJasmin






default	minp=3000,dist=0.08	minp=2000,dist=0.08	minp=1200,dist=0.
407s	58s	39.8s	31.5s
0.4s	0.8s	0.7s	0.9s
407.5s	58.8s	40s	32.5s





# Conclusion

- From result overview: Better results and performance, but could not tried to apply FEM to the mesh.
- From personnal overview: earned how to code with an unknown package and the importance of team communication.

## references

-  Yu, M., Lafarge, F., Oesau, S., & Hilaire, B. (2022). *Repairing geometric errors in 3D urban models with kinetic data structures*. *ISPRS Journal of Photogrammetry and Remote Sensing*, 192, 1-12.  
<https://inria.hal.science/hal-03767910>
-  Bauchet, J.-P., & Lafarge, F. (2020). *Kinetic Shape Reconstruction*. *ACM Transactions on Graphics*.  
<https://inria.hal.science/hal-02924409>
-  The CGAL Project. (2024). *CGAL User and Reference Manual (5.6.1)*. CGAL Editorial Board.  
<https://doc.cgal.org/5.6.1/Manual/packages.html>
-  Yu, M., & Lafarge, F. (2022). *Finding Good Configurations of Planar Primitives in Unorganized Point Clouds*. In *CVPR 2022 - IEEE Conference on Computer Vision and Pattern Recognition*.  
<https://inria.hal.science/hal-03621896>

-  Numpex. (2024). *Numpex Homepage*. <https://numpex.org/>
-  Cemosis. (2024). *Cemosis Homepage*. <https://www.cemosis.fr/>
-  Hidalgo2. (2024). *Hidalgo2 Homepage*.  
<https://www.hidalgo2.eu/>
-  Ktirio. (2024). *Ktirio Homepage*. <https://ktirio.fr/>
-  Inria Titane Team. (2024). *Inria Titane Team Home Page*.  
<https://team.inria.fr/titane/team/>

-  CGAL. (2024). *CGAL Homepage*. <https://www.cgal.org/>
-  Numpex. (2024). *Numpex Exa-MA methods and algorithms for exascale*. <https://numpex.org/exama-methods-and-algorithms-for-exascale/>
-  CGAL. (2024). *User Manual for KSR*. [https://cgal.geometryfactory.com/CGAL/doc/master/Kinetic\\_surface\\_reconstruction/index.html](https://cgal.geometryfactory.com/CGAL/doc/master/Kinetic_surface_reconstruction/index.html)
-  Feel++ Project. (2024). *User Manual Feel++ Toolboxes*. <https://docs.feelpp.org/toolboxes/latest/index.html>