



pcl

3D SCANNER

UBFC

UNIVERSITÉ
BOURGOGNE FRANCHE-COMTÉ



PROJECT SOFTWARE ENGINEERING GROUP 2



VIBOT:

WAJAHAT AKHTAR

OMAIR KHALID

THOMAS DREVET



MSCV:

MARC BLANCHON

ZAIN BASHIR

MOHIT AHUJA

UTPAL KANT

YAMID ESPINEL

SEPHIDEH HADDADI



MAIA:

LEV KOLEZHUK

MARIA DEL CARMEN

MORENO GENIS

YULIIA KAMKOVA

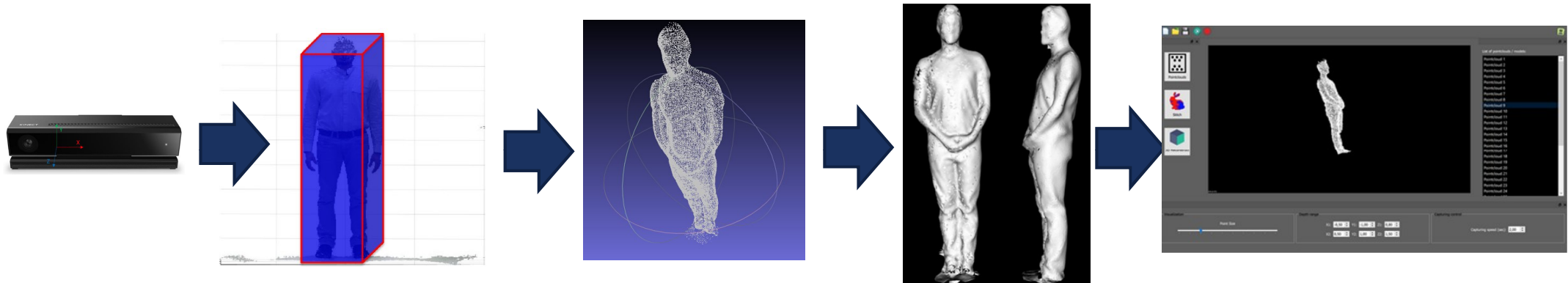
THE GOAL OF THE PROJECT

To create the software product for performing the 3D scanner in the QT environment on Windows using C++ language of the programming

Tasks:

- Choose the model of the 3D camera (sensors) for the project
- Choose the libraries and install them to obtain best results
- Interfacing sensor with the QT and obtaining points clouds
- Point registration (ICP)
- Implementation of Meshing/3D Construction
- Graphical User Interface (GUI)

STEP DIAGRAM



1. Getting data
from Kinect 2

2. Filtering by
coordinates

3. Applying ICP

4. 3D reconstruction

5. Performing in the
Graphical User
Interface (GUI)

CHOICE OF THE LIBRARIES

PRESENTED BY MARIA DEL CARMEN MORENO GENIS



RECONSTRUCT ME

- Software interface

Kinect

└─> 3D data

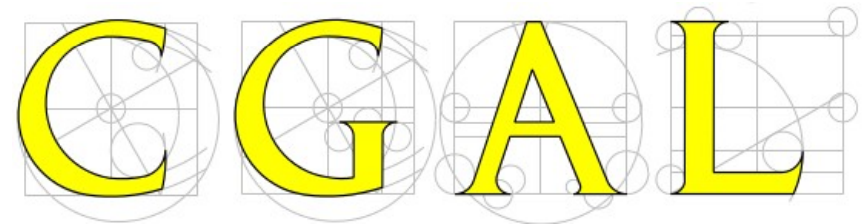
└─> 3D mesh file

- Wide applications and easy manipulation.
- Good approach for real-time 3D scanning
- Not open source



COMPUTATIONAL GEOMETRY ALGORITHMS LIBRARY (CGAL)

- Class library
- Extensive use of advanced C++ features:
 - Templates
 - Traits
- Include set of algorithms
 - Point Clouds and polygonal
 - Volumetric mesh
- Extensive documentation and example sets.
But not precisely with point cloud processing.
- Focuses more on computational geometry/computer graphics.
- Complication with interfacing with QT



POINT CLOUD LIBRARY (PCL)



- Class library
- Useful components to operate Point Clouds
 - Normal Estimation.
 - Registration.
 - Visualization.
 - Filterings
 - Spatial search
 - Surface reconstruction
- Large set of updated research-grade algorithms
- Focuses more on the 3D perception/robotics communities
- Extensive usage of non trivial C++ feature.

ACQUISITION AND FILTERING

PRESENTED BY ZAIN BASHIR



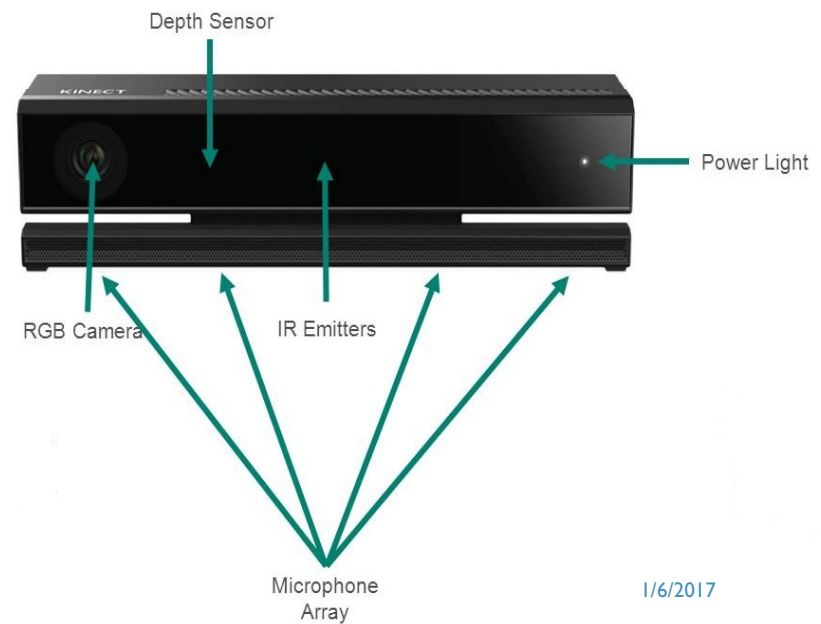
pcl



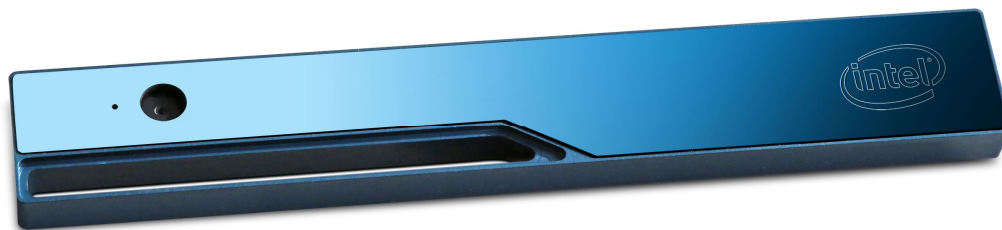
THE KINECT V2 SENSOR

- Time of Flight sensor
- RGB camera of 1280x960 pixel resolution
- Viewing angle = 43 degree (horizontal) by 57 degree (vertical)
- Frame rate = 30 FPS

What is Kinect?



THE INTEL R200 SENSOR



	color	depth / IR
Active Pixels	1920 x 1080	640 x 480
Aspect ratio	16:9	4:3
Frame rate	30 fps	30/60 fps
Field of view (D x V x H)	77° x 43° x 70° (Cone)	70° x 46° x 59° (Cone)

POINT CLOUD DATA ACQUISITION

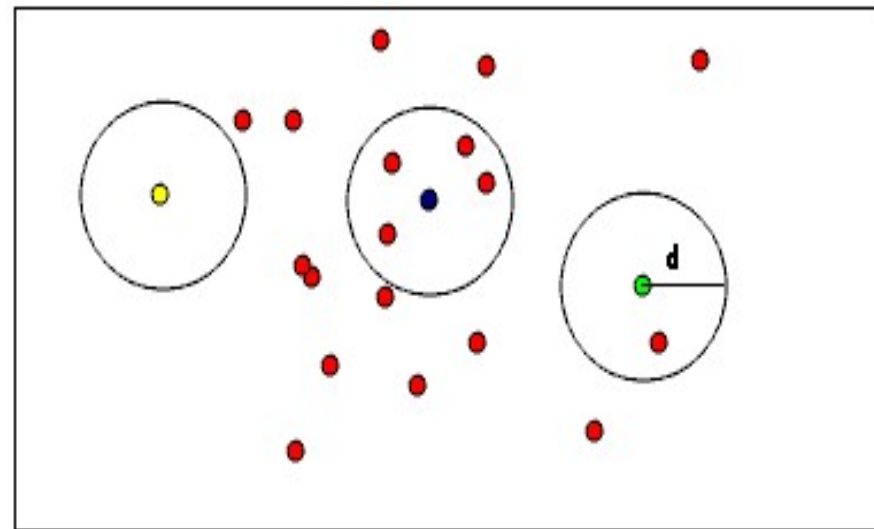
Our approach

- Using Kinect Grabber class to acquire Point cloud data.
- To store point cloud data in a vector of type POINTXYZRGBA and our hard disk.
- Capturing every frame is redundant.
- We use a timer to capture point cloud data after every fixed time interval.
- The interval can be set in our GUI

FILTERING ALONG X,Y AND Z AXIS

I. Radius outlier removal

- Removes points in a point cloud based on the number of neighbours they have.
- Iterates through the entire point cloud once and retrieves the number of neighbours with a certain radius.
- The point is considered an outlier if it has too few neighbours as determined by the function `setMinNeighbours()`
- The radius can be changed using `setRadiusSearch()` function



FILTERING ALONG X,Y AND Z AXIS

2. Pass-through filter

- Passes points in a cloud based on constraints for one particular field of the point type.
- Iterates through the entire input.
- Filters the points outside the interval defined by *setFilterLimits()*.
- Applies only to the field specified by *setFilterFieldName()*.

PLANAR REMOVAL

- Planar removal is the process of removing planes such as floors from the acquired point cloud data
- This can be done using algorithm based on RANSAC
- Since this was computationally heavy, we removed the planes analytically just by the filtering process itself

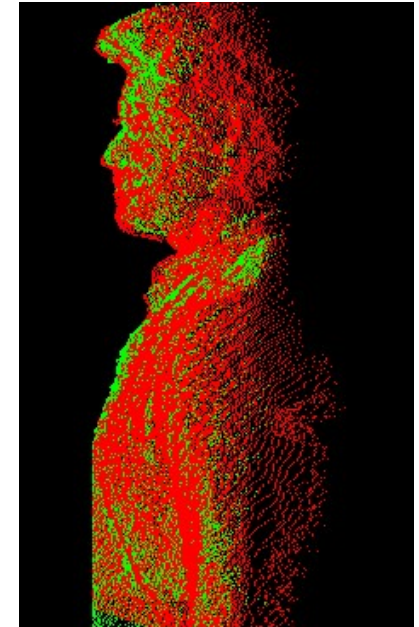
ITERATIVE CLOSEST POINT (ICP)

PRESENTED BY LEV KOLEZHUK AND WAJAHAT AKHTAR



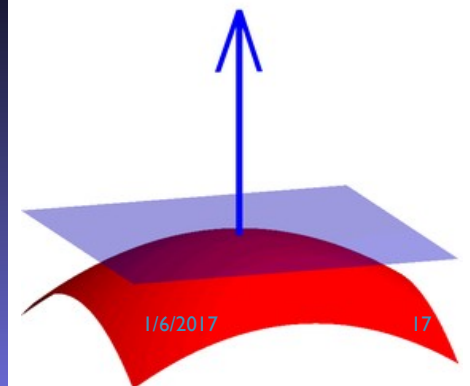
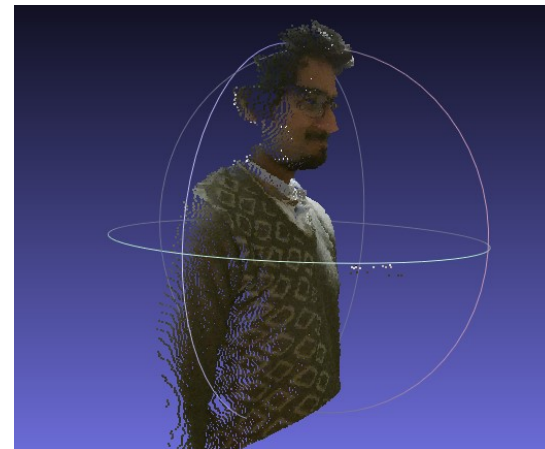
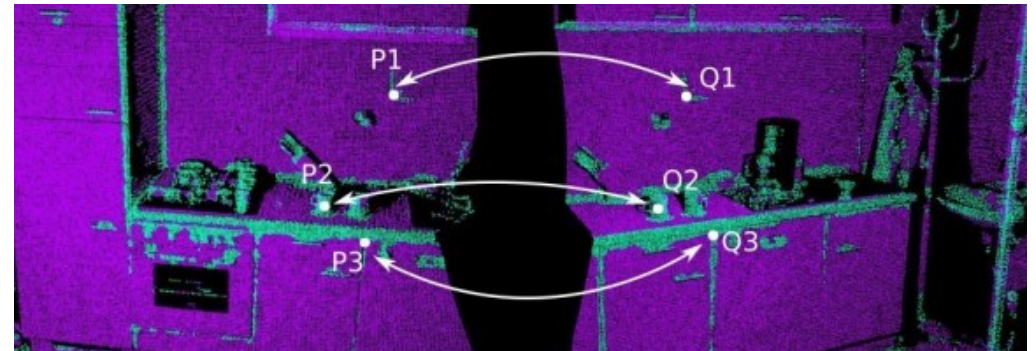
WHAT IS ICP? (ITERATIVE CLOSEST POINT)

- Aligning point clouds in order to recreate a certain object from multiple parts
- Optimization algorithm
- Iterative



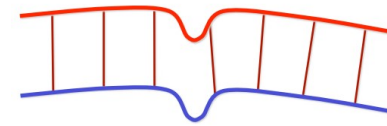
WHAT DATA DO WE WORK WITH?

- Point coordinates.
- Surfaces, normals.
- Correspondent points.
- Colors.

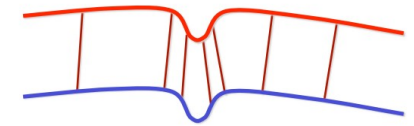


HOW TO SELECT POINTS

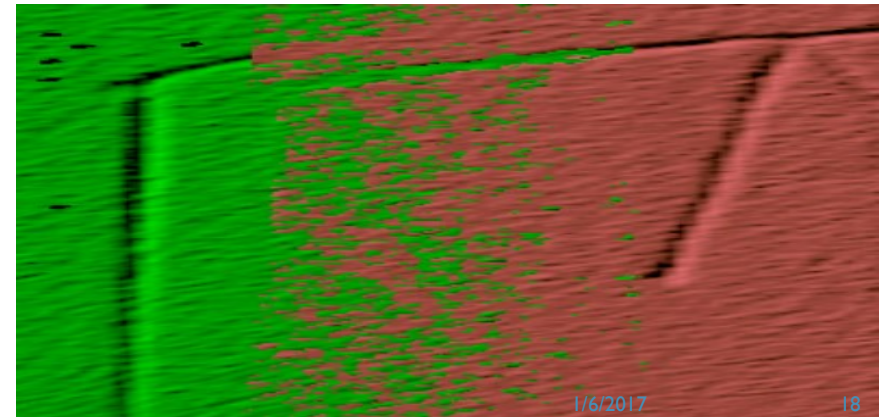
- Take random points (e.g 1000 points) – will decrease the processing time, but may cause some problems in highly detailed places
- Stable sampling and normal space sampling – clever sampling of points



Uniform Sampling

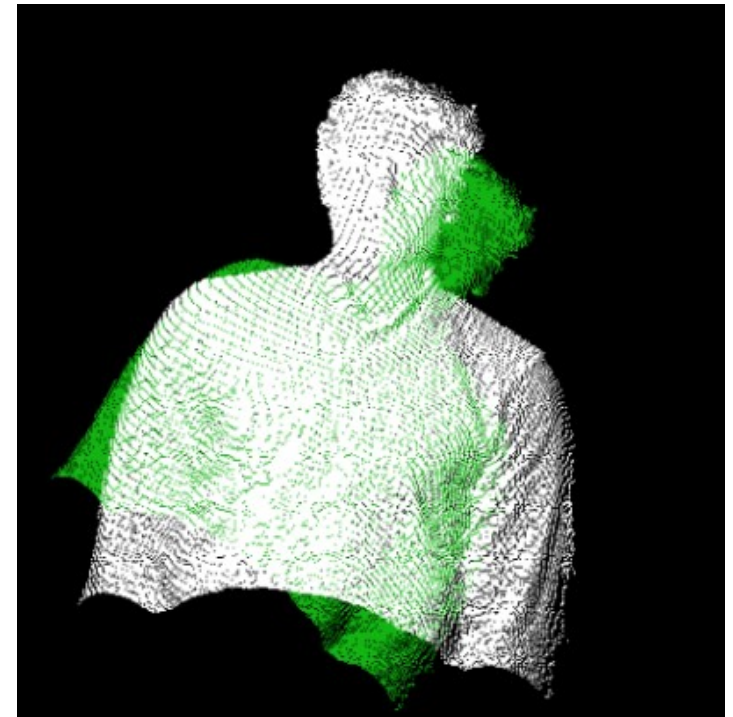


Stable Sampling



PREALIGNMENT

- If we know that our initial pointclouds are quite far from each other, we can provide a prealignment which might spend a lot of time compared to the iterative algorithms
- Based on centroid coordinates, estimate a transformation
- Based on known object rotation between scans



DOWNSAMPLING

- To reduce the amount of the processed points, we can downsample the pointcloud.
- We can estimate the transformation based on less points, but apply it on the initial pointcloud in order to increase speed of the algorithm but preserve the definition.
- If we downsample too much, we will lose accuracy of the ICP.



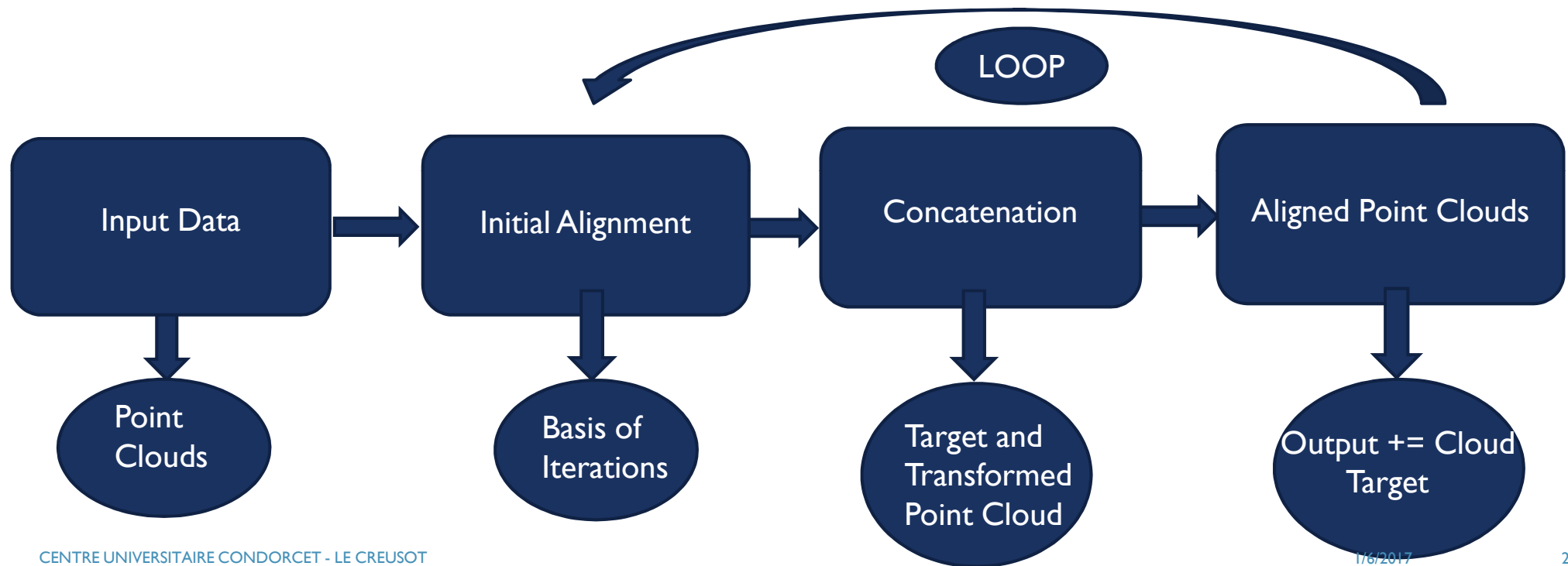
FILTERING AND SMOOTHING

In order to remove noise from input data as well as unnecessary objects and outliers, we have to use some kind of filtering before we start with ICP. Here are some filters we tried to use:

- Statistical outlier removal
- Radius outlier removal
- Color filtering
- Moving least squares



SCHEMATIC ICP



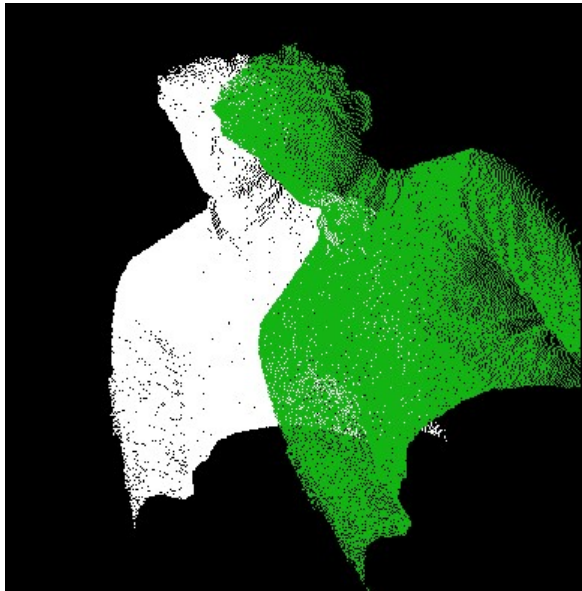
ICP (OUR APPROACH)

- Iterative Closest Point provides a base implementation of the Iterative Closest Point algorithm as defined.
- Aligning Point Clouds (Finding Optimal Rotation and Translation between Corresponding 3D Points) on the basis of iteration and Maximum CorrespondanceDistance between points of the Point Cloud.
- Register the Point Cloud after Initial alignment and Concatenation.
- It is done in a loop so the results of First two Point Clouds after Concatenation are taken as source and aligned with another Target and so on.
- It can be done locally or Globally.

ICP (CONDITIONS)

- The transformation is estimated based on Singular Value Decomposition (SVD).
- The algorithm has several termination criteria:
- Number of iterations has reached the maximum user imposed number of iterations (via setMaximumIterations)
- The epsilon (difference) between the previous transformation and the current estimated transformation is smaller than an user imposed value (via setTransformationEpsilon)
- Concatenating Point Cloud once the best transformation is been found.
- Filtering the Point Cloud by MLS .

SOURCE AND TARGET



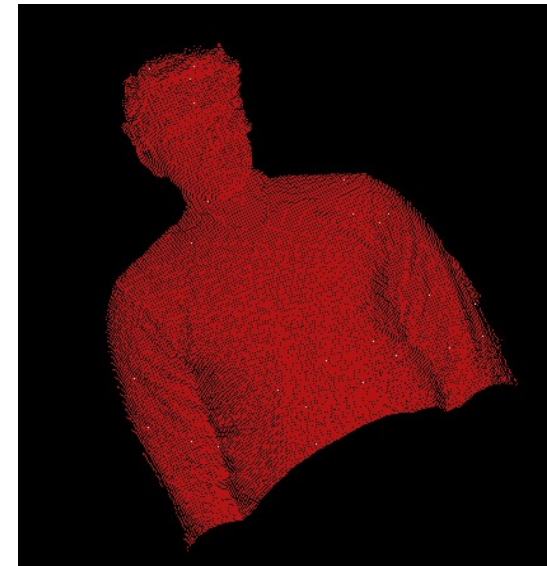
Iterations = 0 (Zero)



Iterations = 2 (Two)

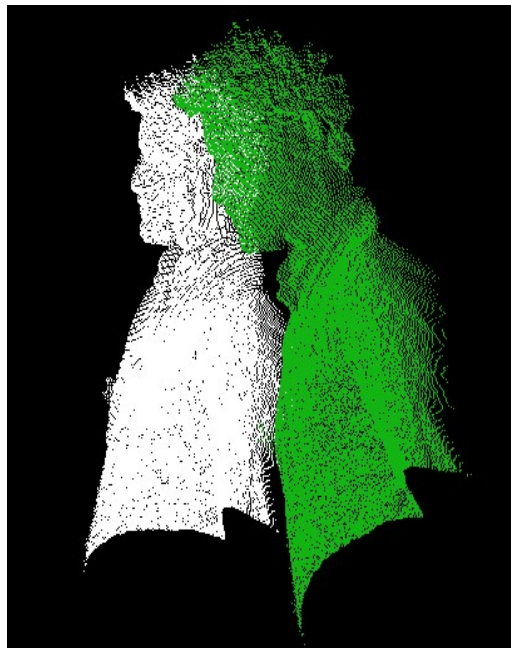
BEST TRANSFORMATION

- We get best Transformation between two point Clouds which can be checked by convergence value.
- There exist problem on shoulders, we get better Transformation on Front and back side of the Person Point Cloud as there is more overlapping but on shoulders we get worse results.



Iterations = 50 (Fifty)

SHOULDER PROBLEMS



Iterations = 0 (Zero)



Iterations = 200 (Two Hundred)

SOLUTION

- More datasets/Point Clouds on Shoulders > More overlap > More Correspondences and better transformation.
- Removing Noise before applying Transformation.
- The data should be taken with Better Resolution not further than 2m.
- Less movement of the person when taking data of the shoulders .
- Aligning or registration done locally.

FILTERING

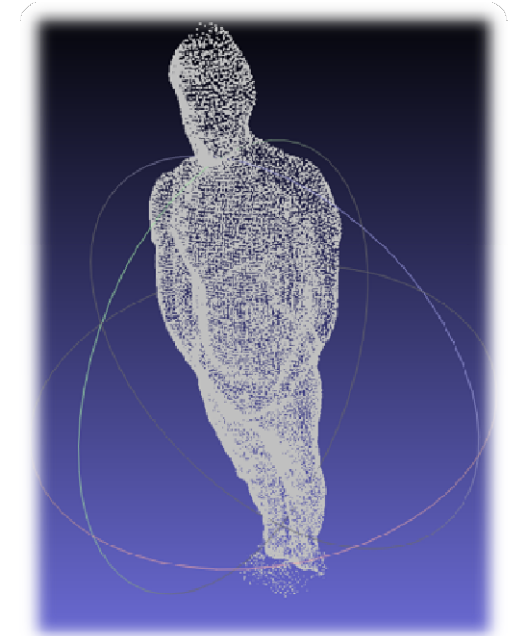
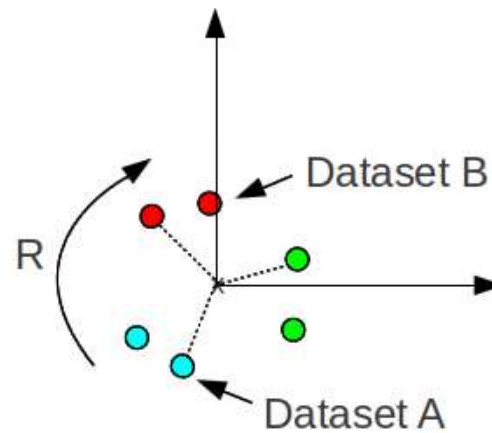
- Filtering in between Point Clouds before second Iteration .
- Filtering source and target Before Finding Transformation between them in every iteration
- Filtering After getting the Concatenated Point Cloud.
- Moving Least Squares (MLS).
- Removed outliers using a Statistical Outlier Removal filter.

MOVING LEAST SQUARES (MLS) AND STATISTICAL OUTLIER REMOVAL FILTER (SOR)

- Moving Least Squares (MLS) surface reconstruction method can be used to smooth and resample noisy data.
- SOR computes first the average distance of each point to its neighbours (considering k nearest neighbours for each - k is the first parameter).
- Then it rejects the points that are farther than the average distance plus a number of times the standard deviation (second parameter).

OUR THOUGHTS AND RESULTS

- Results can be improved by aligning Point Clouds on the basis of Centroids and finding the best Transformation with minimum error, but it didn't work for us.
- Normalization can be useful but it doesn't work when we used PCL algorithm.



3D RECONSTRUCTION CLASS

PRESENTED BY MARIA DEL CARMEN MORENO GENIS, OMAIR KHALID AND THOMAS DREVET



OVERVIEW OF THE CLASS

- Class contains 4 functions:
 - 3 « Tool » functions to perform the computations
 - Normal Estimation
 - Greedy Triangulation
 - Poisson Algorithm
 - 1 « Global » function, called by the GUI, which uses the « Tool » functions to perform the reconstruction

« GLOBAL » FUNCTION

- Developed to be used in the GUI
 - Need a Boolean as input to select the wanted reconstruction method (simple user choice)
 - Return a displayable and savable mesh
- How it works ?
 - Computes the normals of the point cloud stored in the database
 - According to the value of the Boolean (True = Poisson, False = GT), it calls the corresponding reconstruction function to be executed

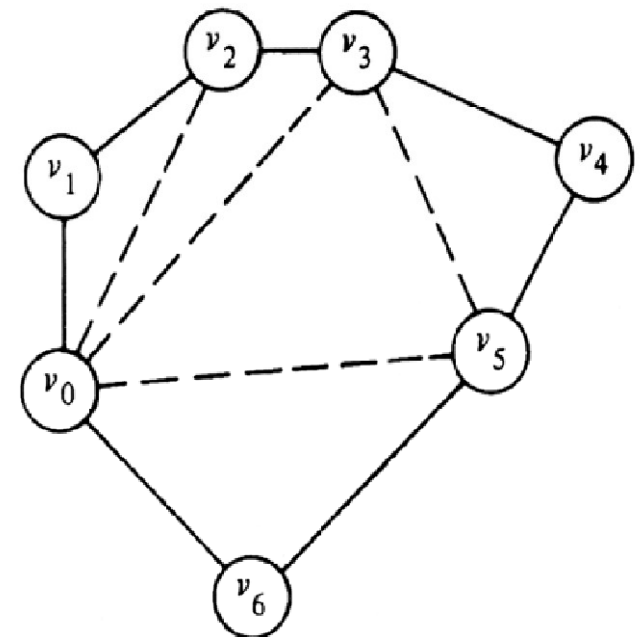
NORMAL ESTIMATION COMPUTATION

- Compute the normals from the Point cloud stored in the database
- The normals are needed to get the surfaces in our reconstruction methods (know the orientation)
- We use the functions from the NormalEstimationOMP class (from PCL)

WHAT IS GREEDY TRIANGULATION ?

Based on construction of triangles.

- Project the local neighbourhood of a point along its normal
- Adds the shortest “compatible” edge between 2 vertices.
 - Compatibility depends on the following parameters (explained in the next slide)



WHAT IS GREEDY TRIANGULATION ?

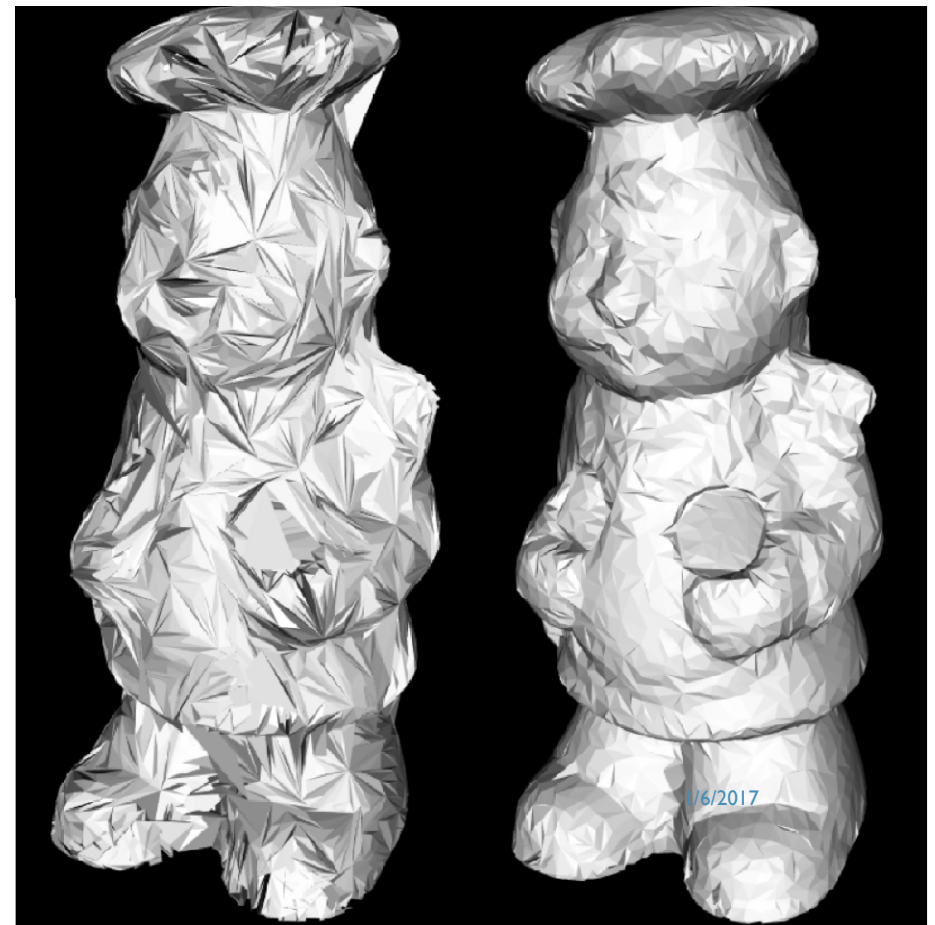
■ Parameters :

- **Search Radius:** *Max. length allowed for an edge of a triangle.*
- **Mu:** *Density parameter. Helps adjust the Search Radius depending on the density. New Search Radius equals distance of query point to nearest neighbour multiplied by Mu.*
- **Max. Nearest Neighbors:** *Max. Number of closest Neighbors to consider within the search radius.*
- **Max Surface Angle:** *the angle between the normals of the query point and subject point.*
- **Min Angle:** *Min angle in each triangle.*
- **Max.Angle:** *Max. angle in each triangle.*
- **Normal Consistency:** *We set this flag if we know that the normals are oriented consistently.*

WHAT IS GREEDY TRIANGULATION ?

Comparison between inappropriate and appropriate manipulation of the variables.

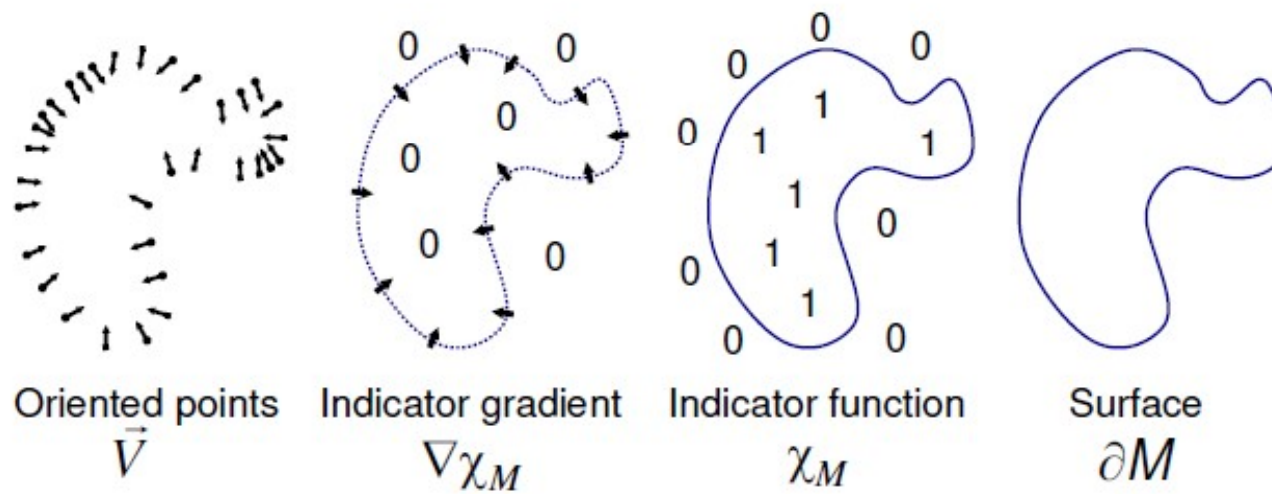
Used Library: GP3 from PCL



WHAT IS THE POISSON ALGORITHM ?

- At first the points with the oriented normals are taken and the gradient of the indicator function is approximated through the normal field.
- After that the indicator function, which is zero everywhere except near the surface, is derived from this gradient.

WHAT IS THE POISSON ALGORITHM ?



WHAT IS THE POISSON ALGORITHM ?

$$X_n = \alpha X + (1 - \alpha) X$$

$$X_m = X - X_g$$

$$\nabla \cdot \nabla X = \nabla \cdot \nabla X_m : \partial M$$

Problem: Size of Gaussian, Noisy Data

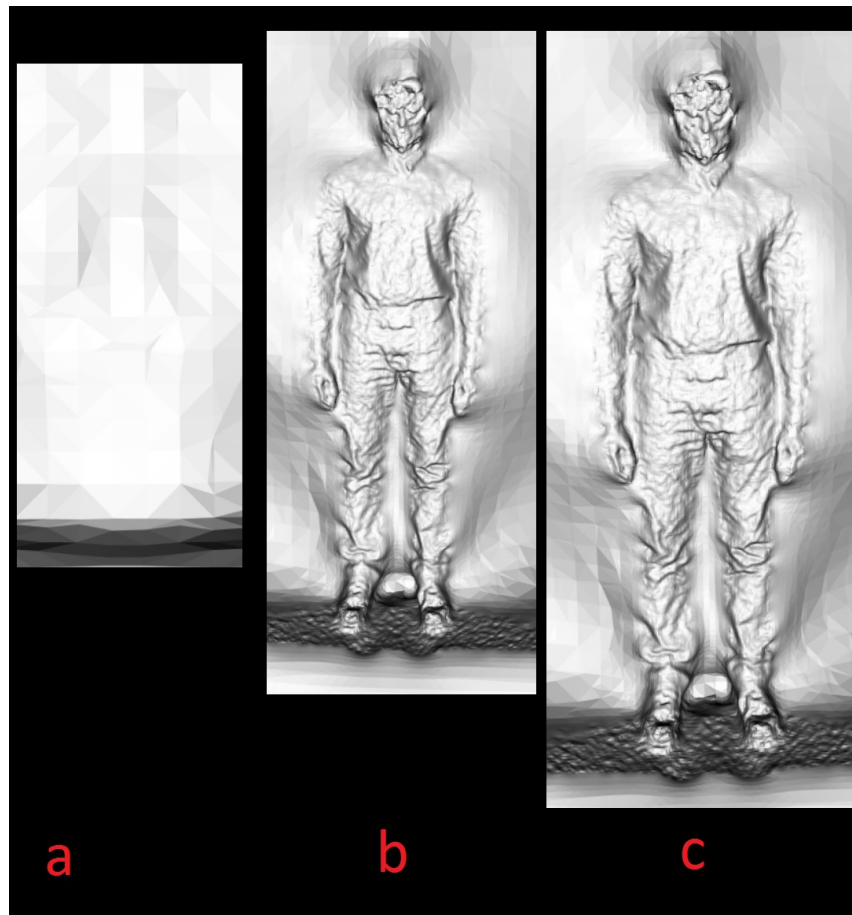
WHAT IS THE POISSON ALGORITHM ?

Depth of the Octree importance:

- a: 4
- b: 9
- c: 14

Used Library : Poisson from PCL

CENTRE UNIVERSITAIRE CONDORCET - LE CREUSOT



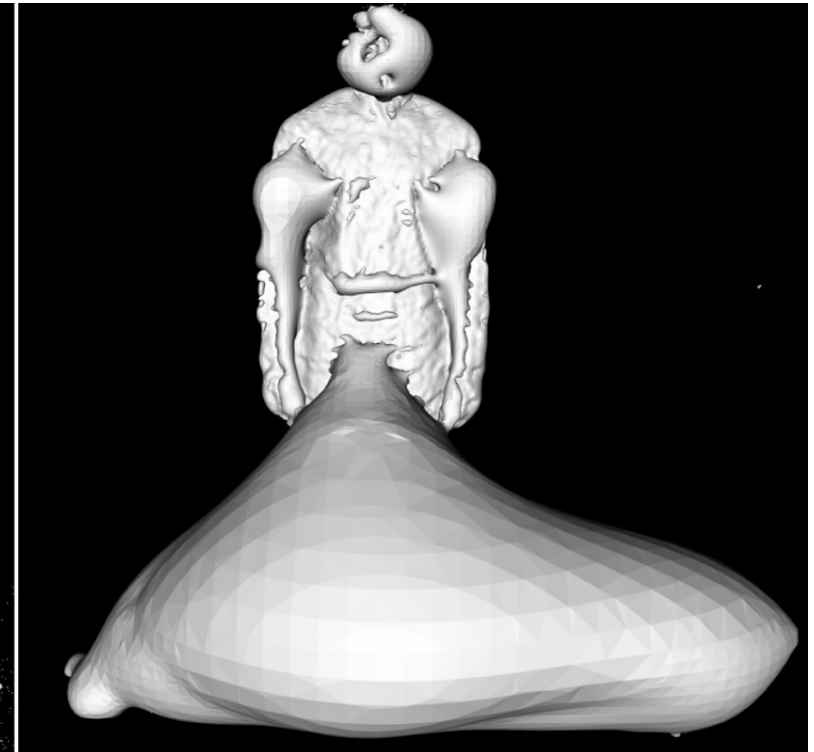
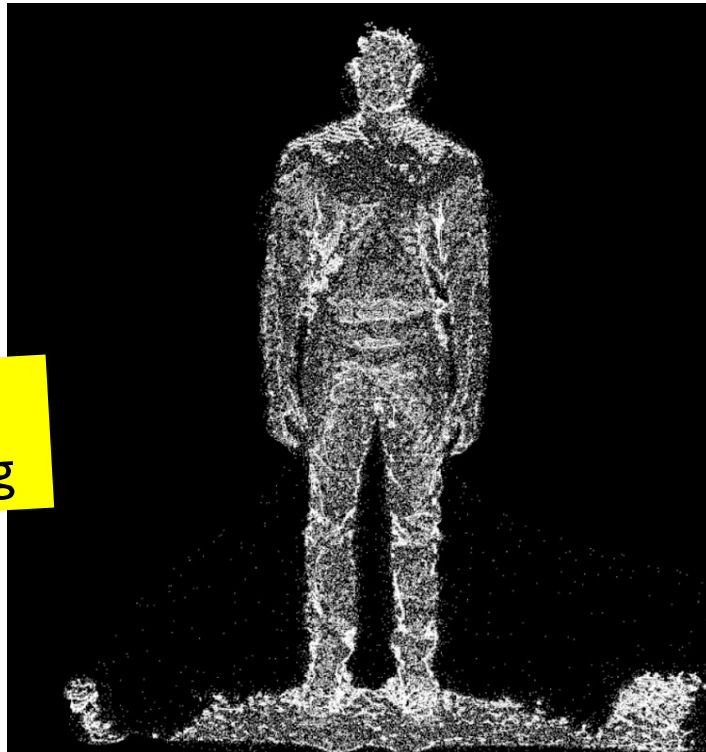
1/6/2017

42

WHAT IS THE POISSON ALGORITHM ?

Possible Errors

Noise data =
Over-Smoothing



WHAT IS THE POISSON ALGORITHM ?

Possible Errors

Narrow spaces =
Evaluated as single space



RESULTS AND LIMITS

Greedy Triangulation



Holes Problem

Poisson Algorithm



Over-Smoothing Problem

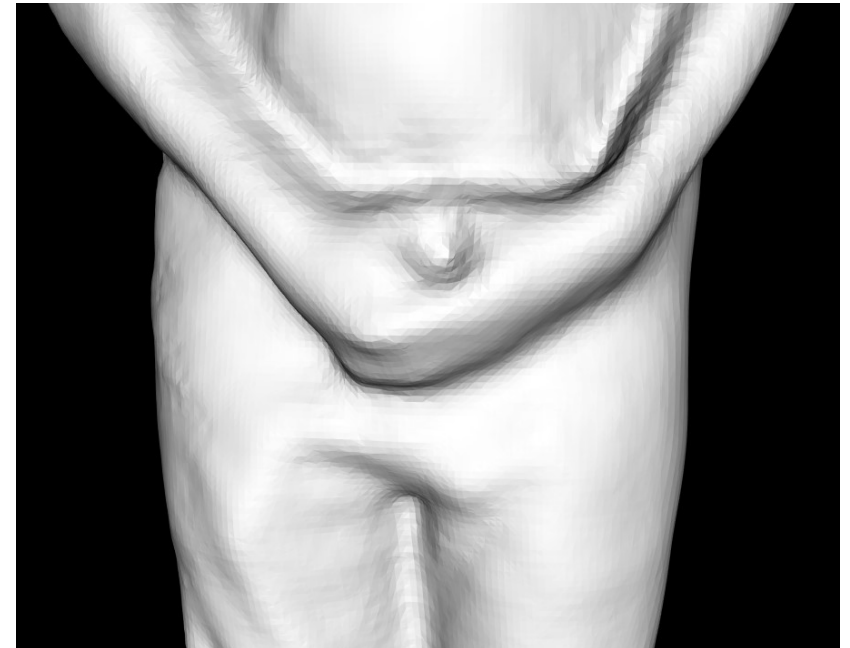
RESULTS AND LIMITS

Greedy Triangulation



CENTRE UNIVERSITAIRE CONDORCET - LE CREUSOT

Poisson Algorithm



1/6/2017

POSSIBLE IMPROVEMENTS

- Drawback of Greedy Triangulation: Holes
- Solution: Find Holes and apply Holes-Filling Algorithm
- To find holes:
 - Extract information of the polygons that is in the form "a b c", where a, b and c, respectively, are the index number of the three vertices of the triangle present in the same .PLY file.
 - Extract the three edges from the polygon.
 - Get the information of boundary edges (the edge which does not lend itself to two triangles)
 - Find connectivity of the edges to find the holes.

POSSIBLE IMPROVEMENTS

How to fill holes?

- Apply the Advancing Mesh Technique to fill the holes, as explained in the article "A robust hole-filling algorithm for triangular mesh" by Wei Zhao, Shuming Gao and Hongwei Lin.

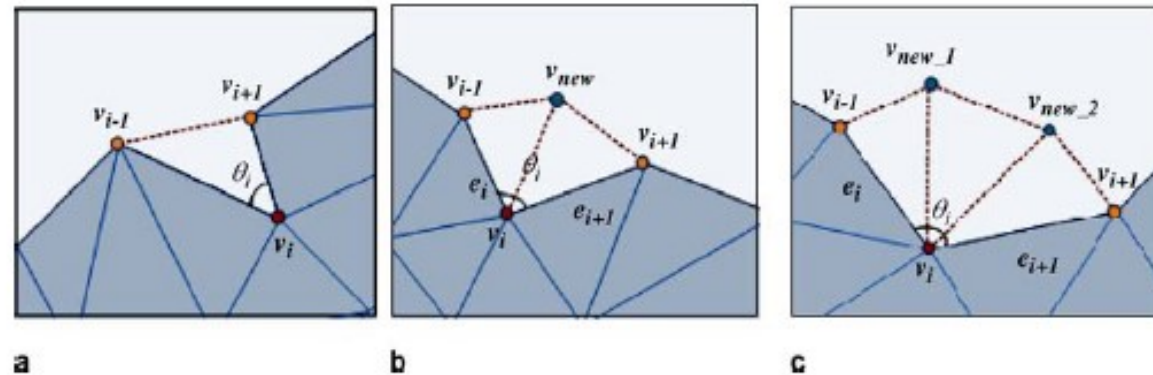


Fig. 3a–c. Rules for creating triangles: **a** $\theta_i \leq 75^\circ$; **b** $75^\circ < \theta_i \leq 135^\circ$; **c** $\theta_i > 135^\circ$

POSSIBLE IMPROVEMENTS

- Problem: Extra Surfaces (Poisson Algorithm)
- Solution: To improve the resulting mesh, we should develop a function that remove all the surfaces that are too far from the original point cloud.
- Holes-filling can be applied, if holes appear.

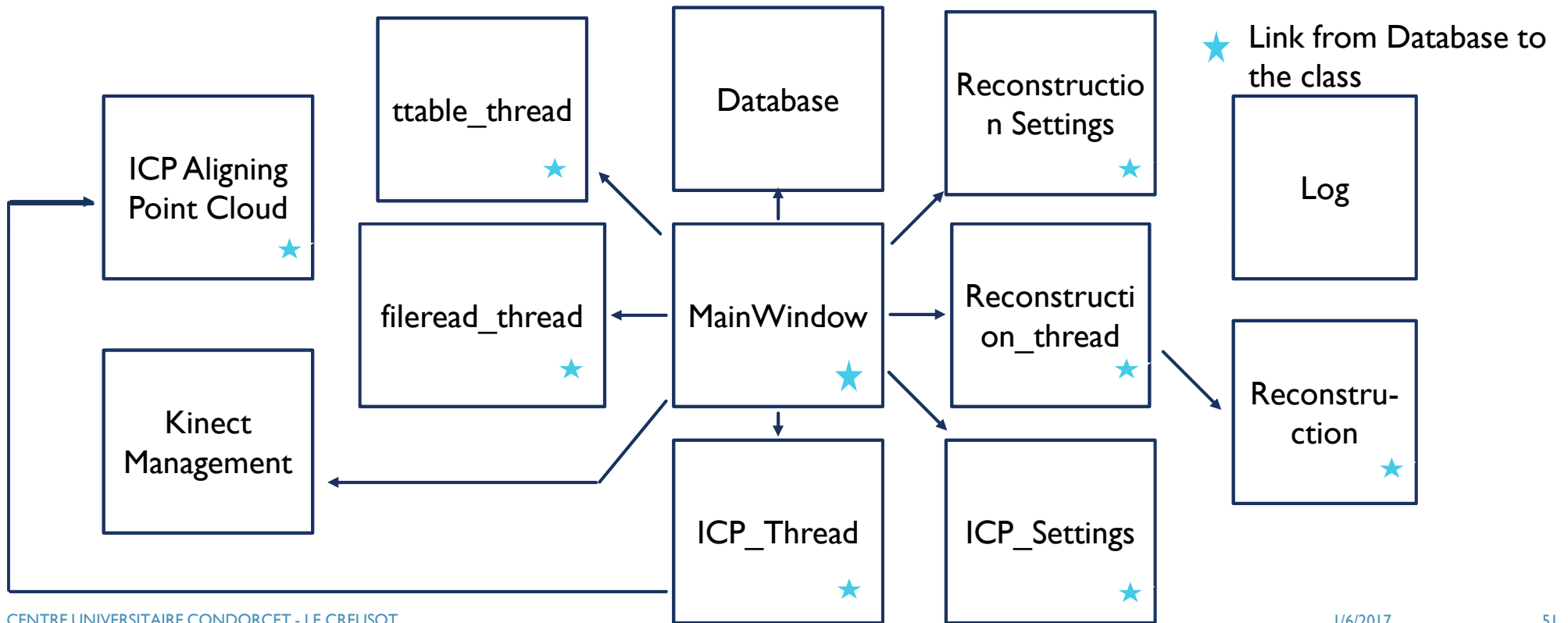
STRUCTURE OF THE CODE

PRESENTED BY MARC BLANCHON



Code less.
Create more.
Deploy everywhere.

STRUCTURE



OUR APPROACH

- Using a Database class as main actor
- All other classes act as function for each other or as mutators for database members
- MainWindow (UI) get all the data from database and display
- Optimal and understandable hierarchy of code
- Easy to manipulate the code
- Usage of standard of code

IMPROVEMENTS

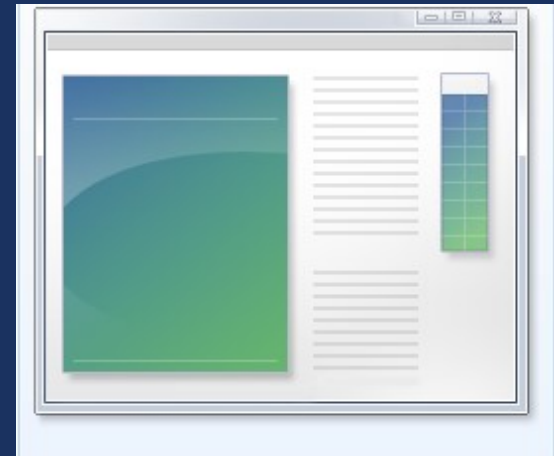
- Polymorphism for Kinect and R200
- Full usage standard of codes
- Threading process

ADVANTAGES OF OUR DESIGN

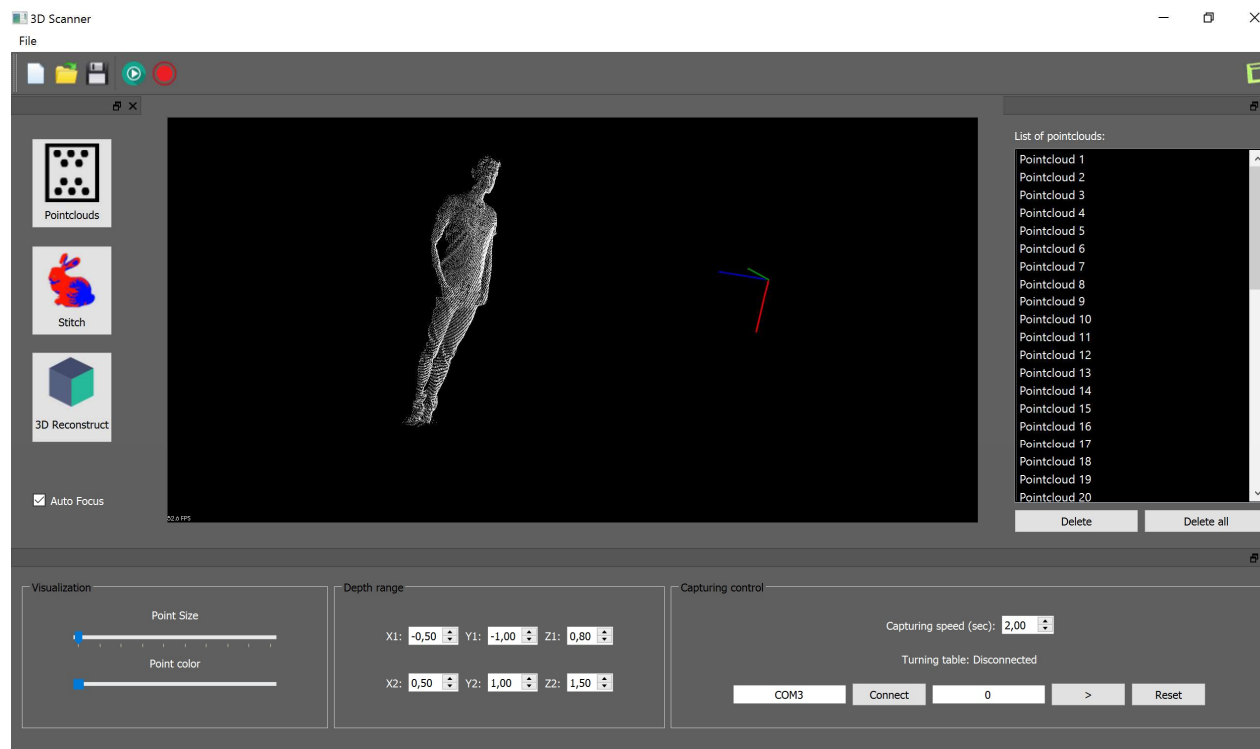
- Optimization
- Portability
- Readability
- Separation of tasks (divide and conquer)

GRAPHICAL USER INTERFACE

PRESENTED BY YAMID ESPINEL LOPEZ



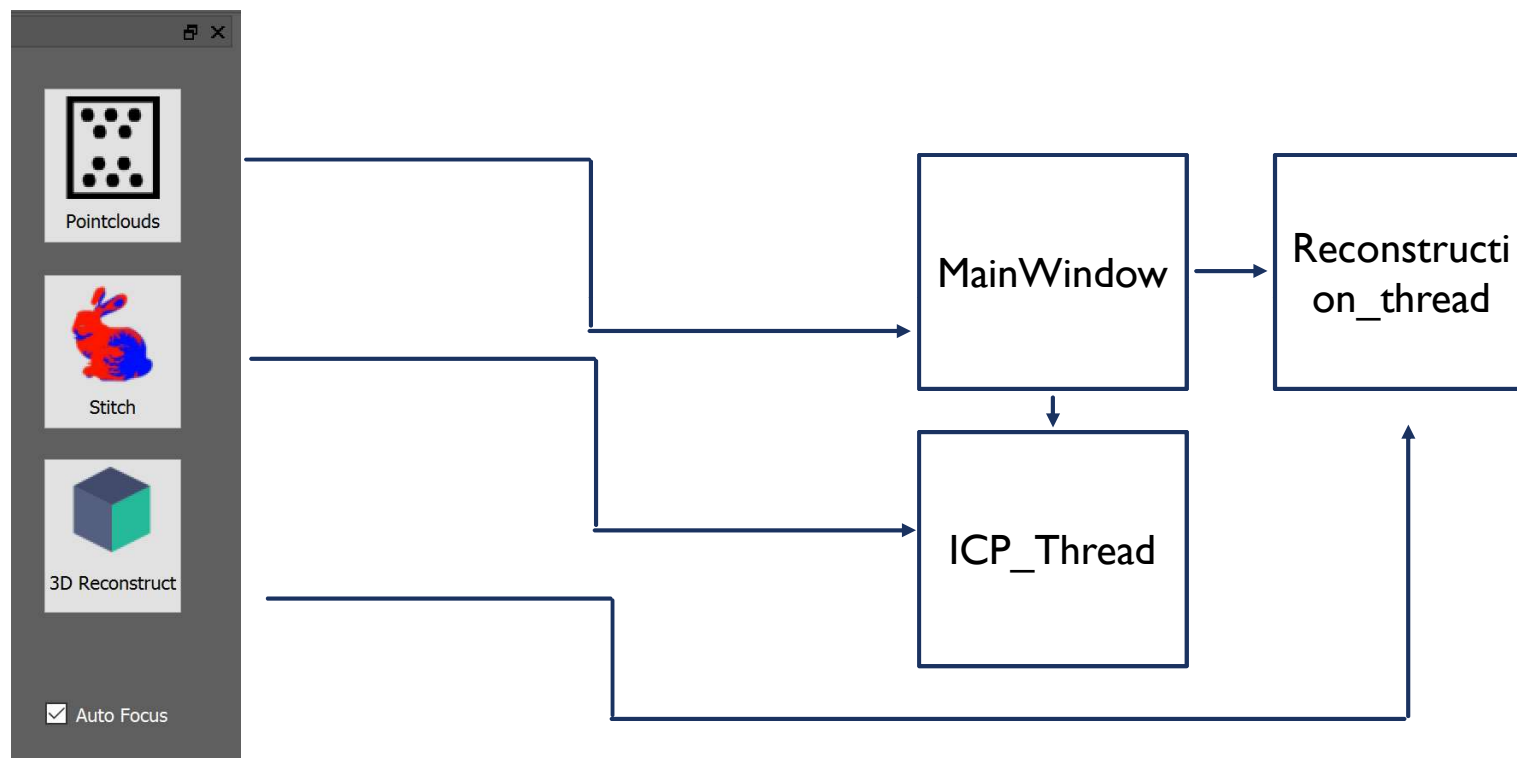
GUI COMPONENTS



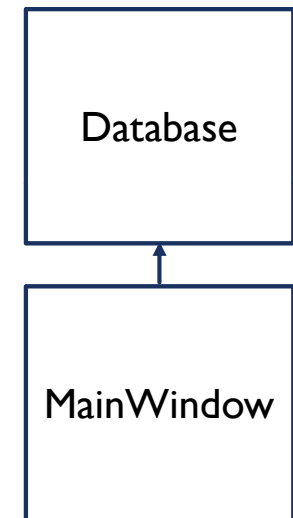
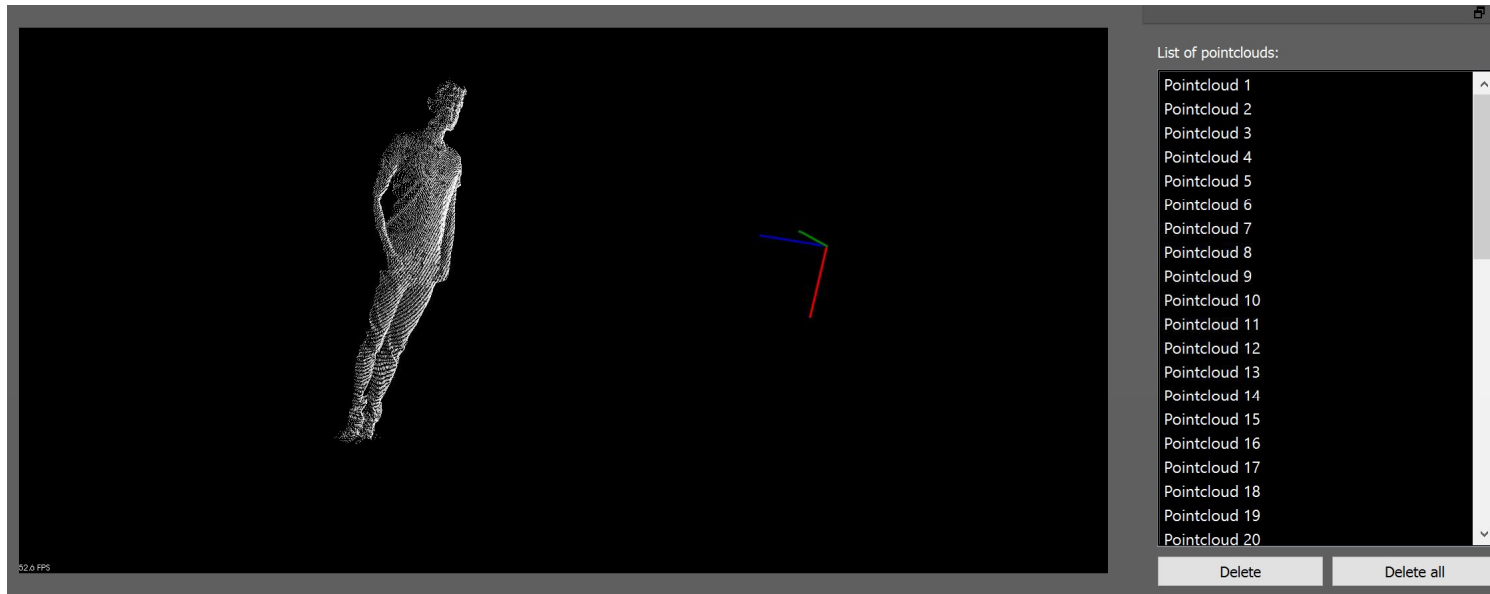
DESIGN

- User friendly
- Usage of layouts to have a clear and light interface
- Adaptability of the window
- Possibility to arrange the window ourselves (taking out the boxes)

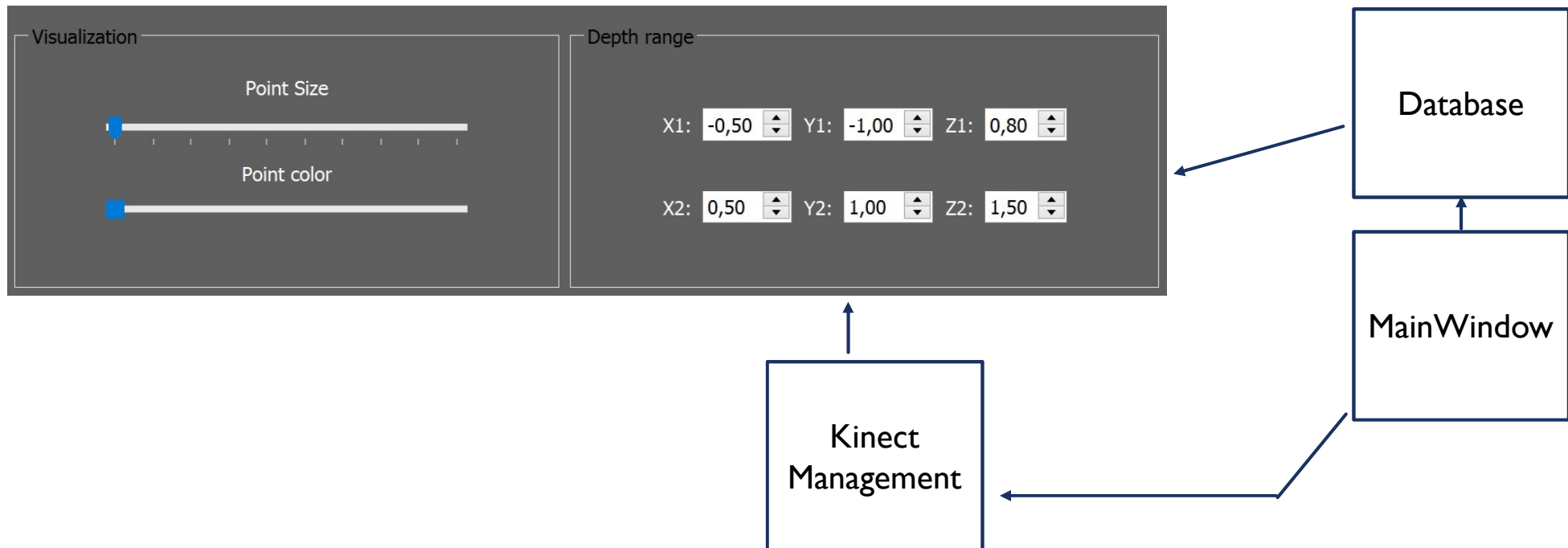
CONNECTIONS TO CLASSES



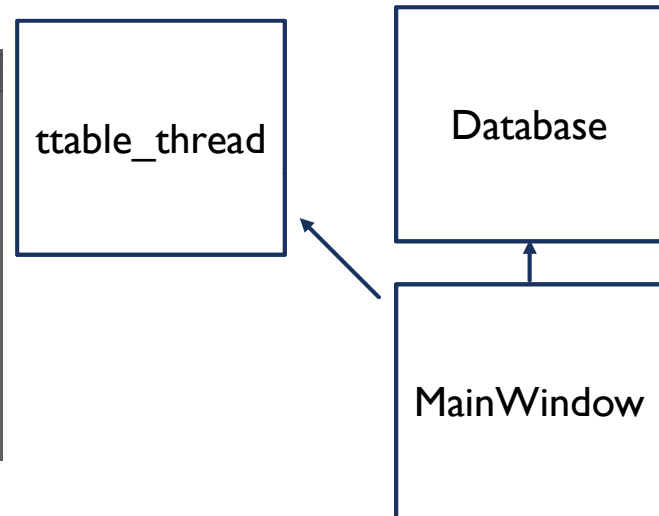
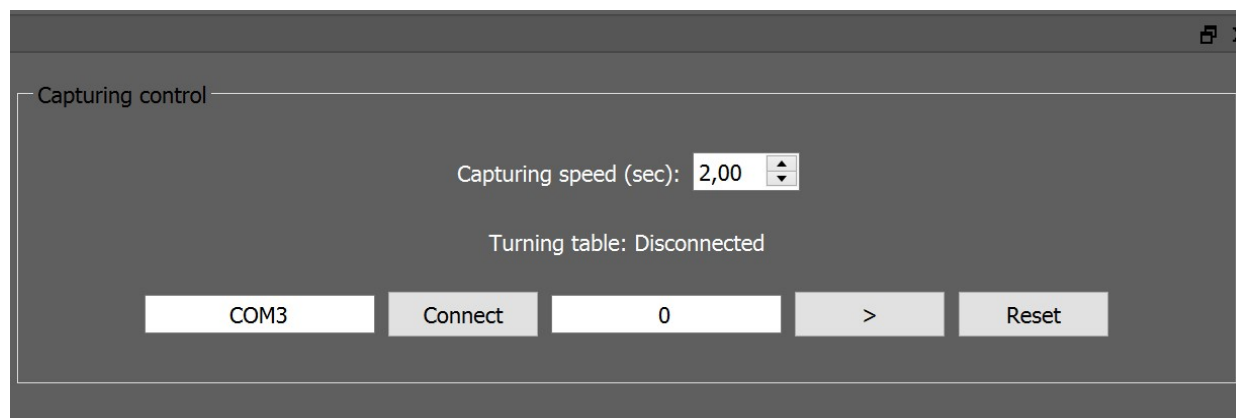
CONNECTIONS TO CLASSES



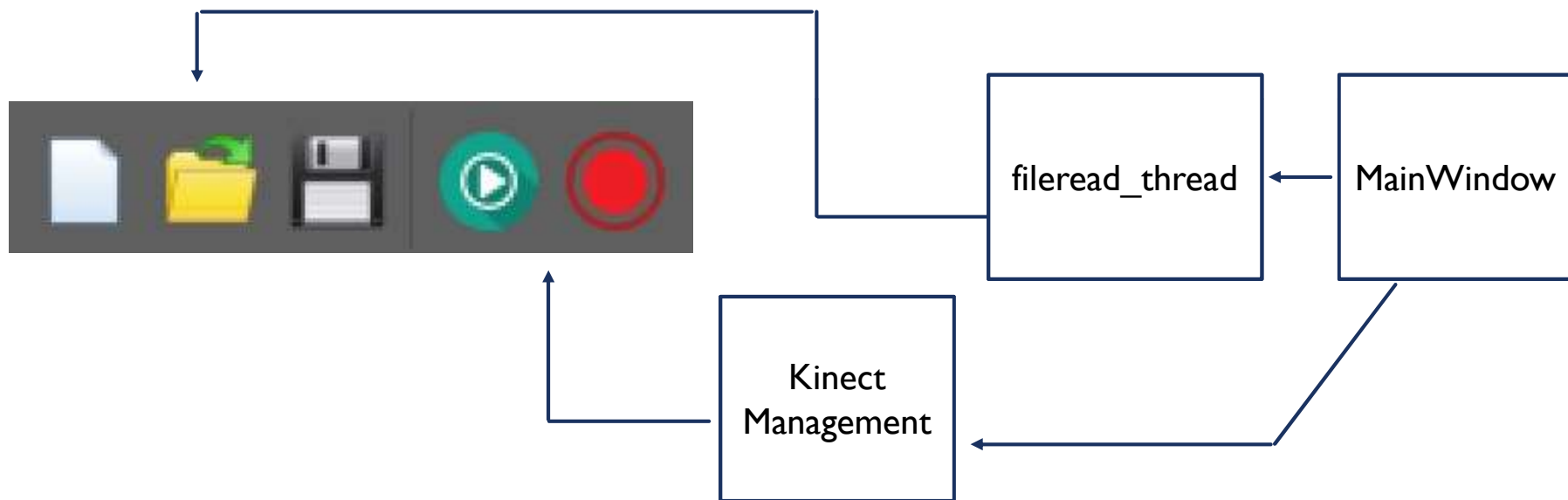
CONNECTIONS TO CLASSES



CONNECTIONS TO CLASSES



CONNECTIONS TO CLASSES



PROJECT MANAGEMENT

PRESENTED BY WAJAHAT AKHTAR



LATEX

PROJECT MANAGEMENT AND SKILLS

- Improvement in the Organization of Team Management during the Project.
 - Setting goals and deadlines.
 - Assigning Weekly Tasks by arranging Weekly meetings (Communication).
 - Weekly and Individual Progress Report to track the Performance of each member.
 - Managing GitHub in order to have a backup of the code with every member knowing about the Progress.
 - Forcing people to use Latex as it was a compulsion.
 - More communications (Facebook, WhatsApp)

FINAL TEAM ORGANIZATION

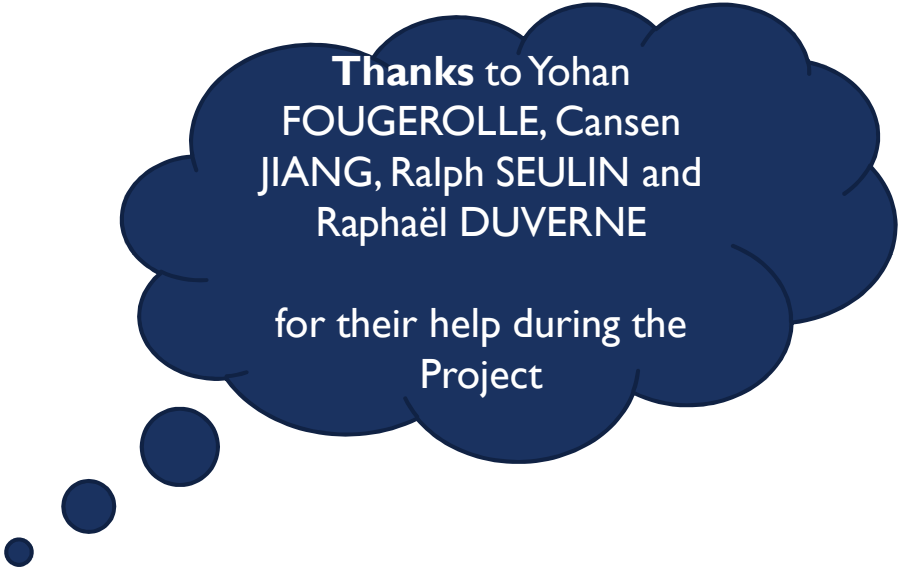
- Different Sub-Teams:
 - **ICP** : Lev and Wajahat
 - **3D reconstruction** : Maria del Carmen, Omair and Thomas
 - **GUI and Structure**: Yamid, Mohit and Marc
 - **Acquisition and Filtering**: Zain
 - **Research**: Yuliia and Utpal
- Prepare some backup plans (some were used)
 - Task not achieved on time -> temporary reorganisation
- Problems we had and how they were handled.
 - One laptop crashed

Head Manager:
Wajahat



CONCLUSION

- Main Goals are achieved (able to scan a full body and get its 3D mesh in a software)
- Results can be improved (smoothing, watertight mesh...)
 1. I.C.P(Noise because of occlusions)
 2. 3D Reconstruction(Holes, Smoothing)
- General improvement of the everyone's skills (coding, team work, tool using, facing problems...)



Thanks to Yohan
FOUGEROLLE, Cansen
JIANG, Ralph SEULIN and
Raphaël DUVERNE

for their help during the
Project