Shape Modeling and Geometry Processing

Exercise 6 – 3D Human Faces

INTERACTIVE GEOMETRY LAB

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Grouping

(1)

-Pascal Chang
-Nicolas Wicki
-Clemens Bachmann
-Franz Knobel
-Isaak Hanimann

(2)

-Yinwei DU
-Shaohui LIU
-Linfei PAN
-Mingyang SONG
-Weirong CHEN

(3)

-Martina Kessler
-Agon Serifi
-Lea Reichardt
-Andreas Aeberli
-Elham Amin Mansour

(4)

-Zhiyin Qian
-Zheyu Shi
-Le Chen
-Boyan Duan
-Chaoyu Du

(5)

Laurin Brandner, laurinb@student.ethz.ch
Jakub Kotal, jkotal@student.ethz.ch
Gilles Waeber, gwaeber@student.ethz.ch
Peter Haas, pehaas@student.ethz.ch
Semadeni Renato, renatos@student.ethz.ch

(6)

Hauksson	Haraldur Orri	hhauksson@student.ethz.ch
Pozdnyakov	Pavel	popavel@student.ethz.ch
Vogelsanger	Christopher	cvogelsa@student.ethz.ch
Zegarac Ana		ana.zegarac@math.ethz.ch
Kürsteiner	Michael	kumichae@student.ethz.ch

(7)

Zimmermann	Karl Walter	zkarl@student.ethz.ch
Terekhov	Mikhail	mterekhov@student.ethz.ch
Choutas Vasileios		vchoutas@student.ethz.ch
DanielettoGianluca Vincenzo Leone		danieleg@student.ethz.ch
Simic	Nicholas	nicsimic@student.ethz.ch

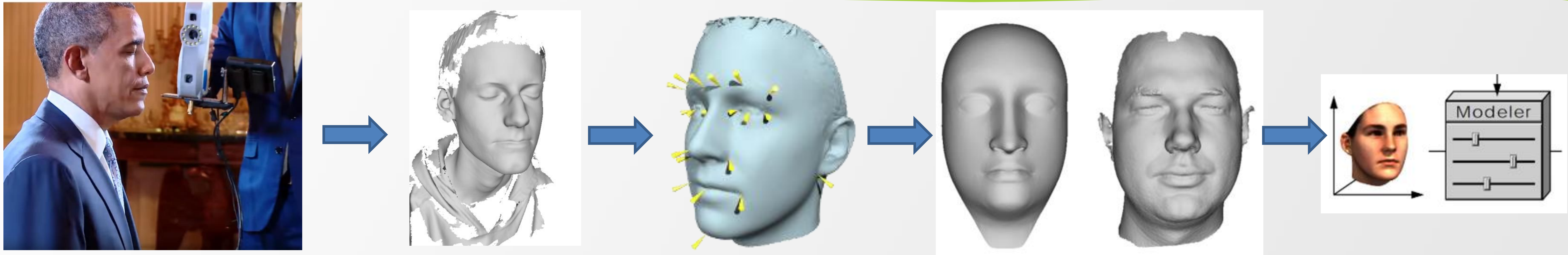
(8)

Dantas Pereira Nuno Alexandre	dnuno@student.ethz.ch
Dunskus Julian Thomas	jdunskus@student.ethz.ch
Nikic Marko	marko.nikic@inf.ethz.ch
Trinh Dinh Thuan-Henry	trinhhe@student.ethz.ch
Mengqi Wang	mewang@student.ethz.ch
Wülfroth Laura Nicola	laura.wuelfroth@inf.ethz.ch

Differences

- Group project: teamwork is both fun and challenging, all group members will have the same score, presentation for each group.
- Real dataset: apply geometry processing with a group of data instead of a single one
- No pre-defined solution: each group should come up with their own solution, and set up a git repository (no base code this time).
- Opportunity to explore geometric deep learning as a (global) bonus task.
- Less time (less than two and a half week)

Overview



Step 0: (Zoom) meet the group members and decide team leader

Step 1: Download 3D faces.

Step 2: Data preprocessing, face rigid and non-rigid alignment.

Step 3: Compute PCA faces and implement face morphing.

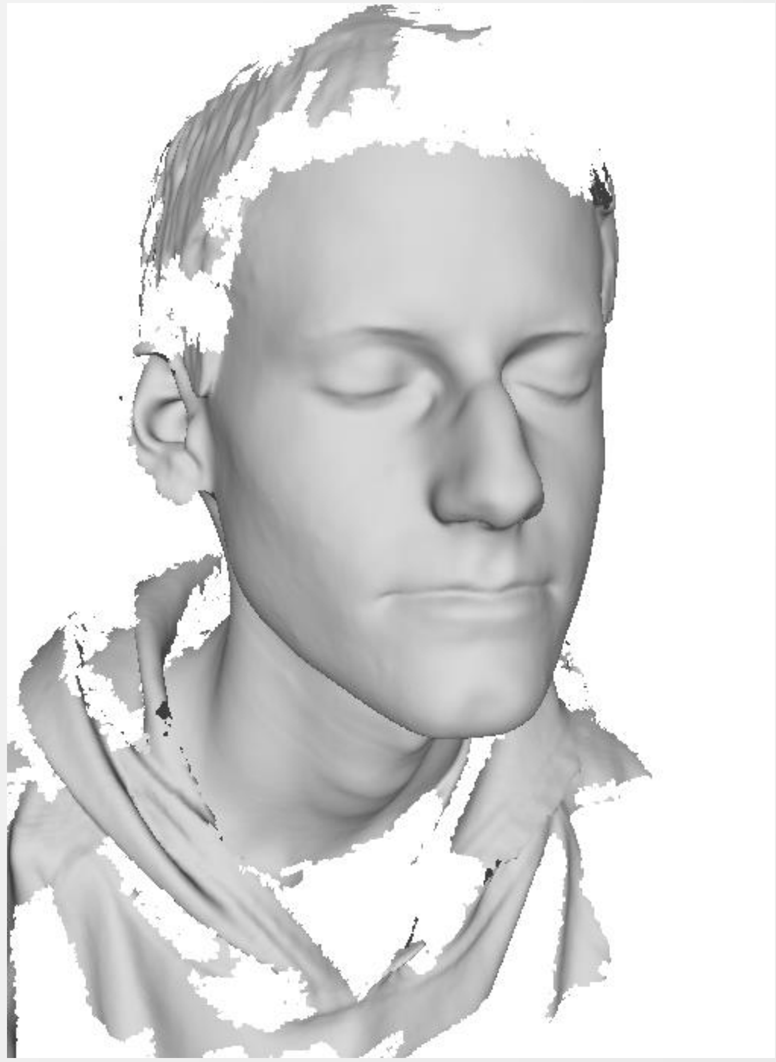
Step 4 (bonus): experiment with different 3D faces and shape space.

Step 5: Demo and final group presentation.

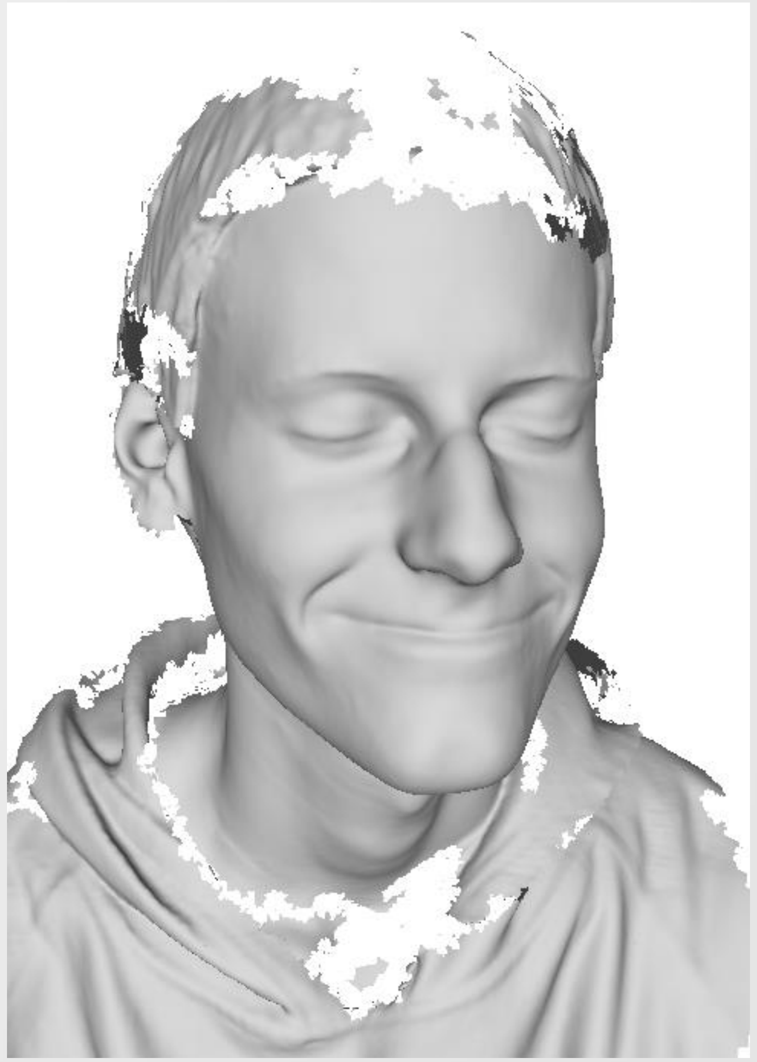
Note: we will provide some example data (scanned faces, face template, aligned faces) so that the implementation of each step should be independent.

Scanning & Preprocessing

- We provide 112 meshes from around 60 different faces, we have cleaned up and removed all the hair and neck parts.
- Better start from a small set
- Inter-group policy



Neutral

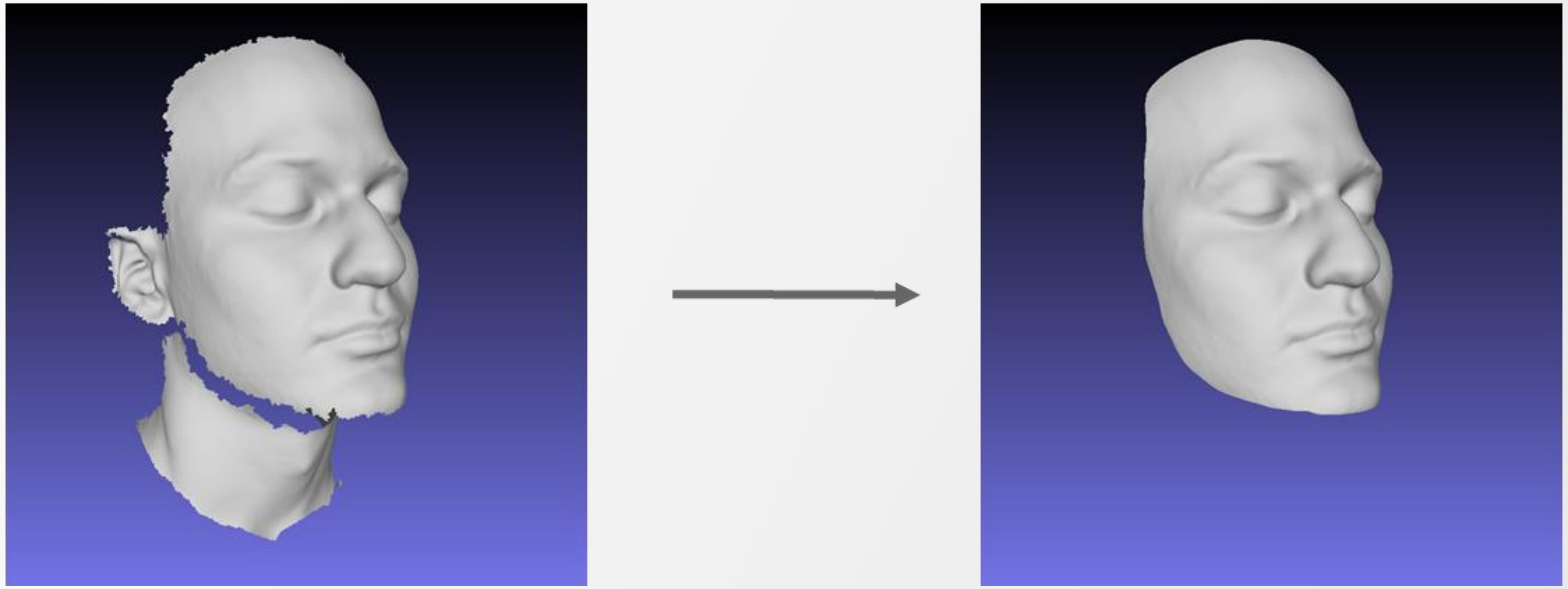


Smile

Scanning & Preprocessing

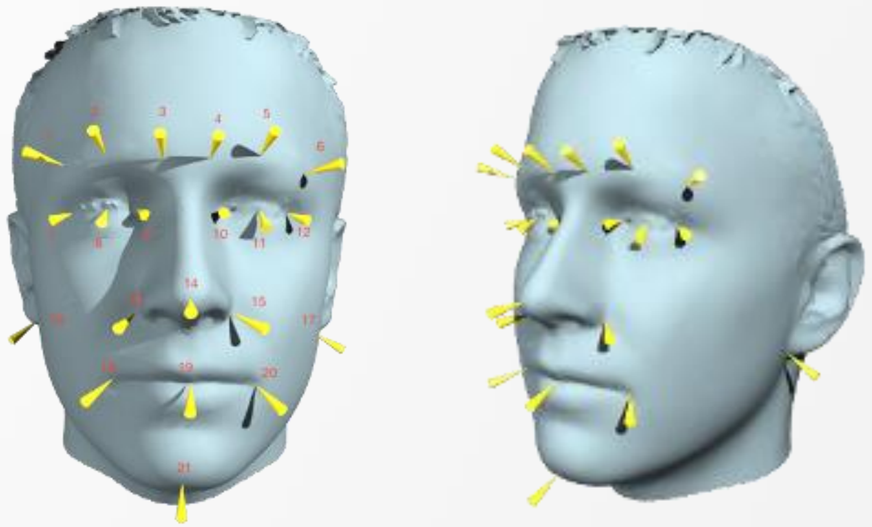
- Clean disconnected components
  (# connected components = 1)

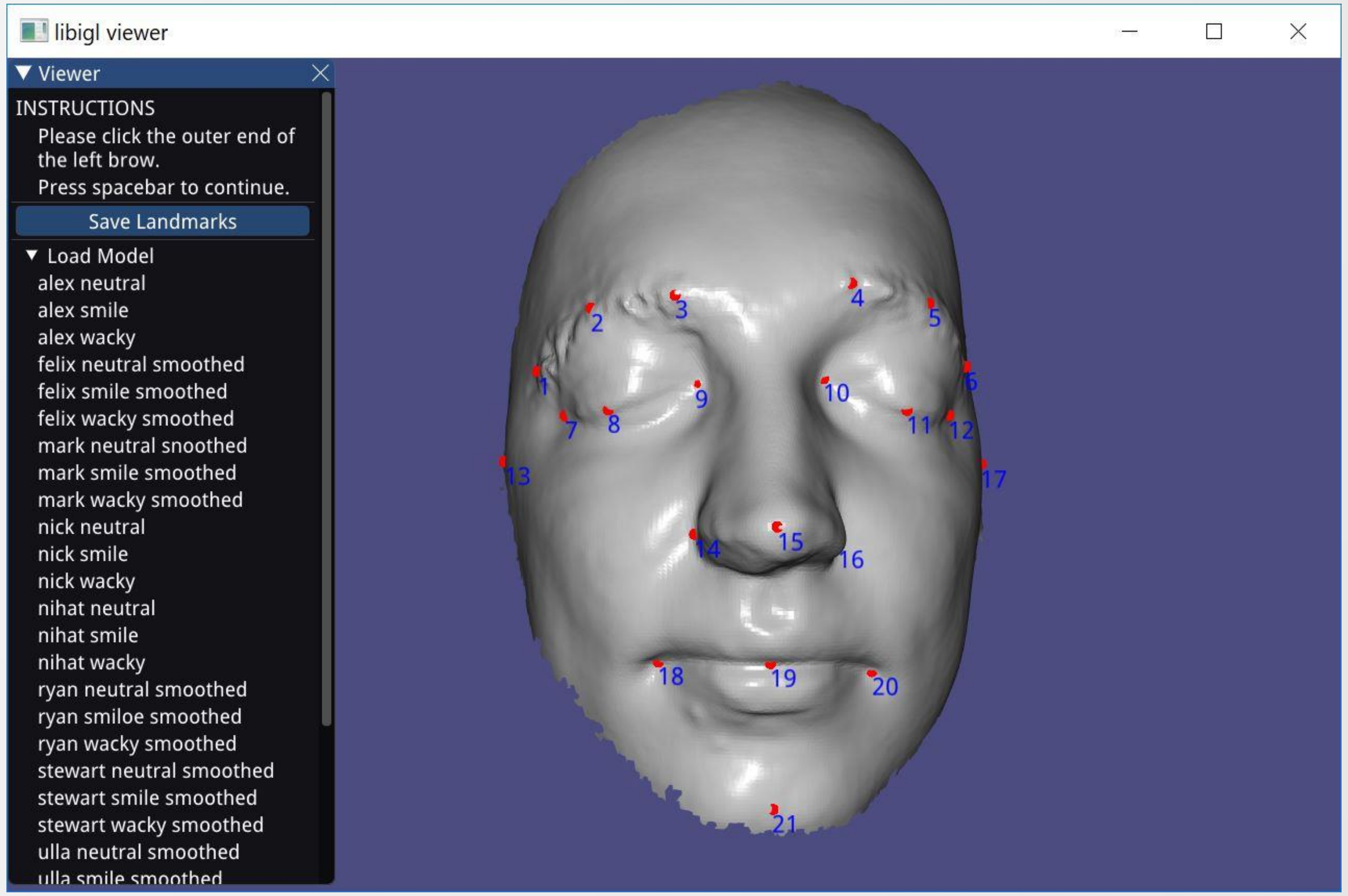
- Standardize face elements for every person
- Smooth mesh boundaries using cotangent Laplacian



Landmarks

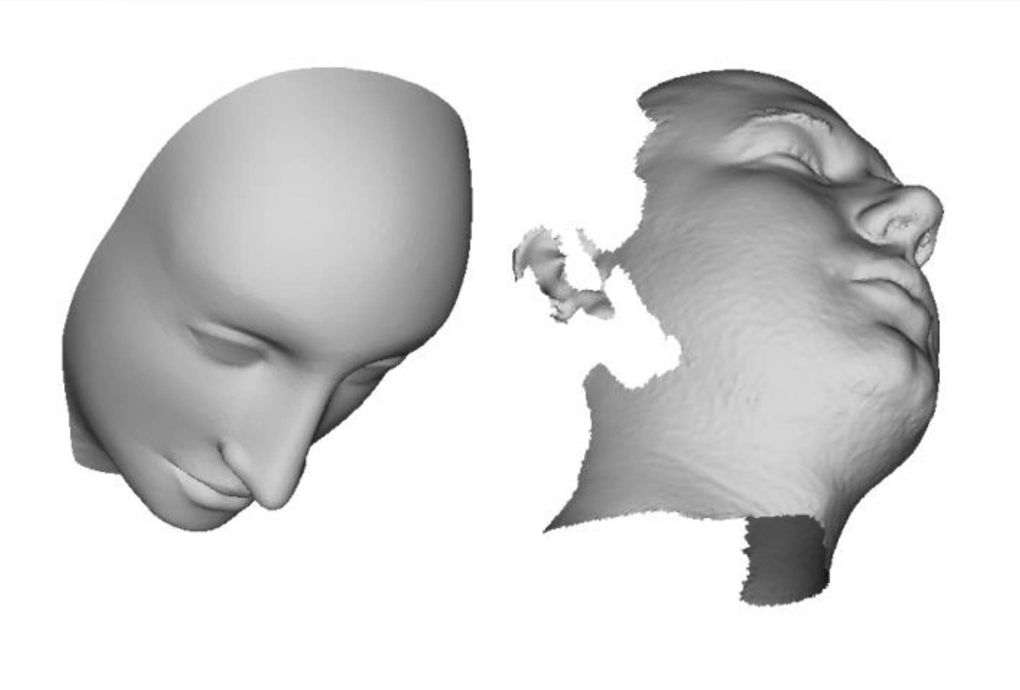
- Extract Landmarks
  - Proposed method: manual selection
  - Implement a tool that allows this





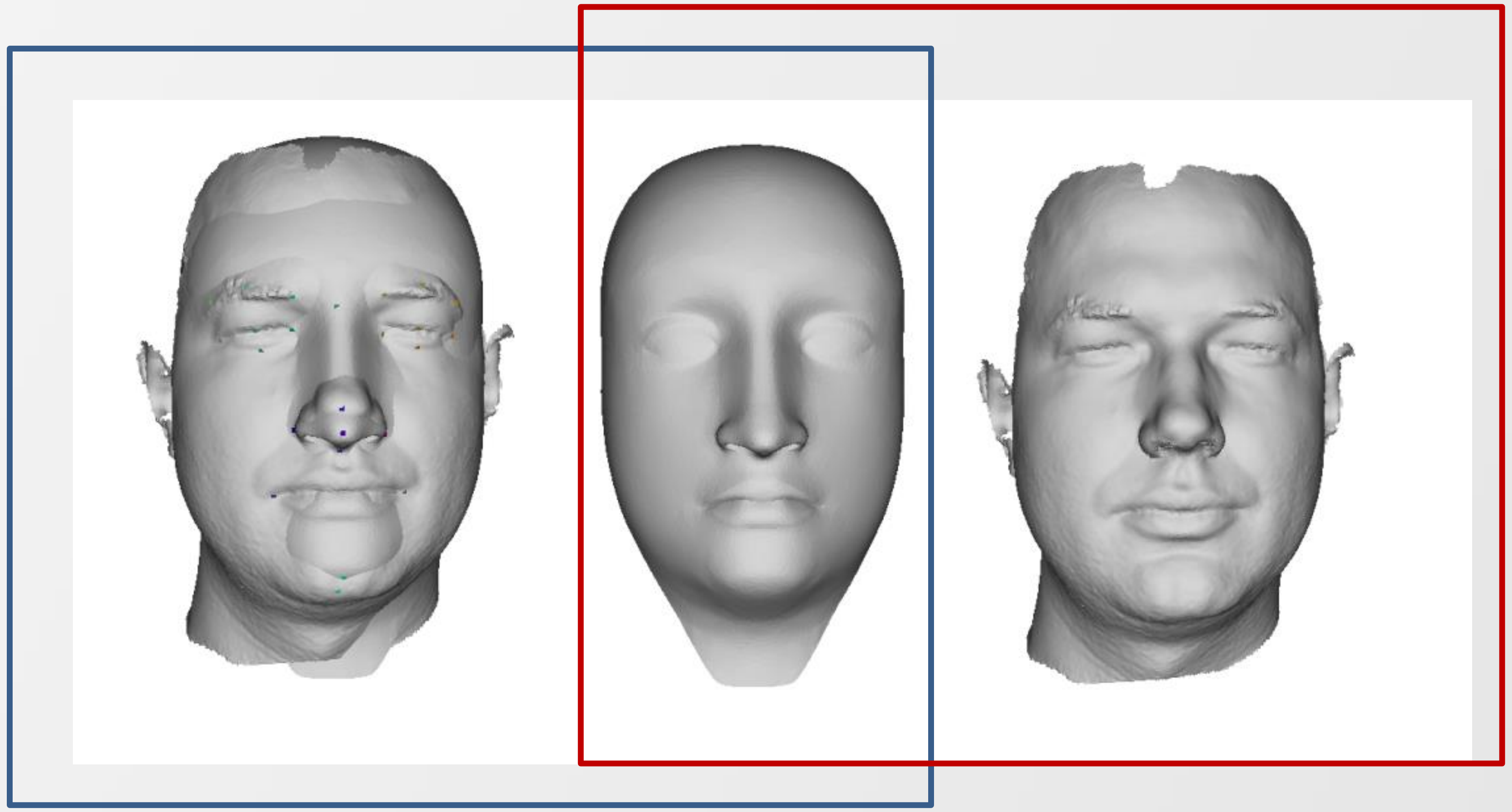
- Get a 3D position from a mouse click (see assignment 5), from libigl

Alignment



Template and target

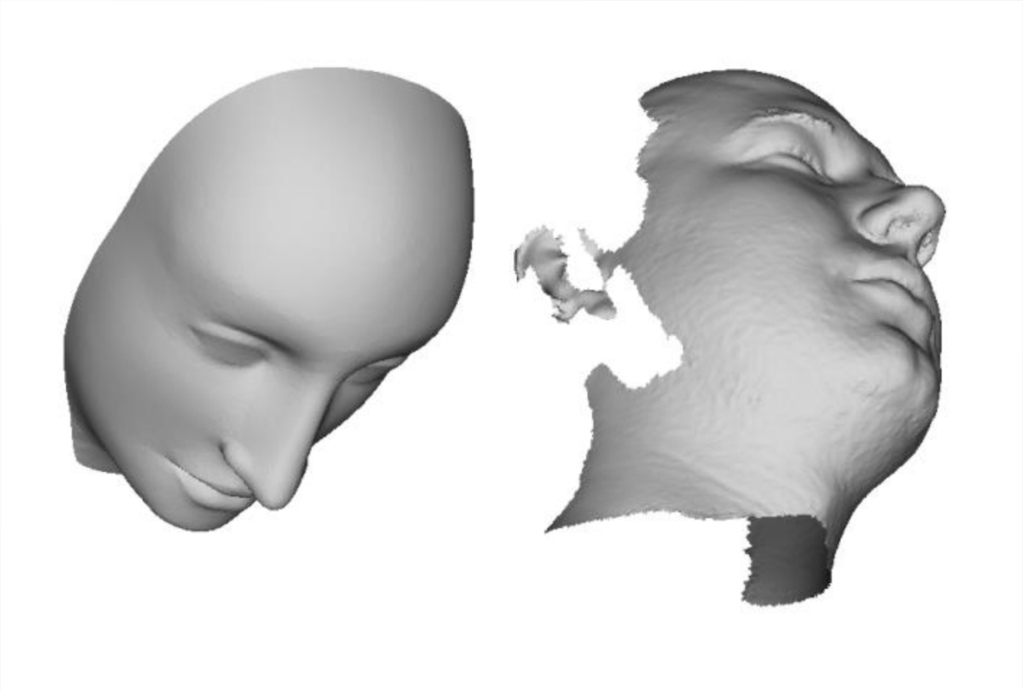
Step 2: non-rigid alignment
(or warping)



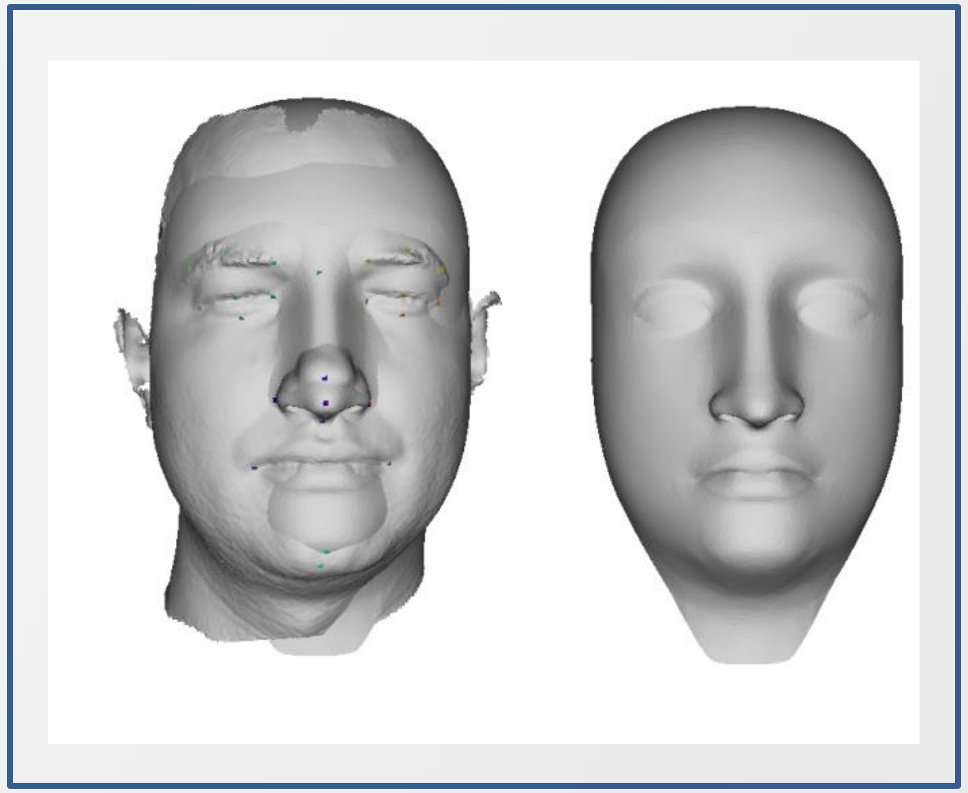
Step 1: Rigid alignment

Rigid Alignment

- Center template and target face meshes
- **Rescale** template to scan
- Rigidly align scan to template



Template and target



Step 1: Rigid alignment

Notes for Rigid Alignment

2. a) Rigid alignment

- Scale template to scan

  - If you rescale scan, make sure scale all scans by same factor.

  - Scale template s.t. the average distance to mean landmark is the same for scan and template.

  - Before scaling the template, translate it such that the mean of its vertices is (0,0,0)

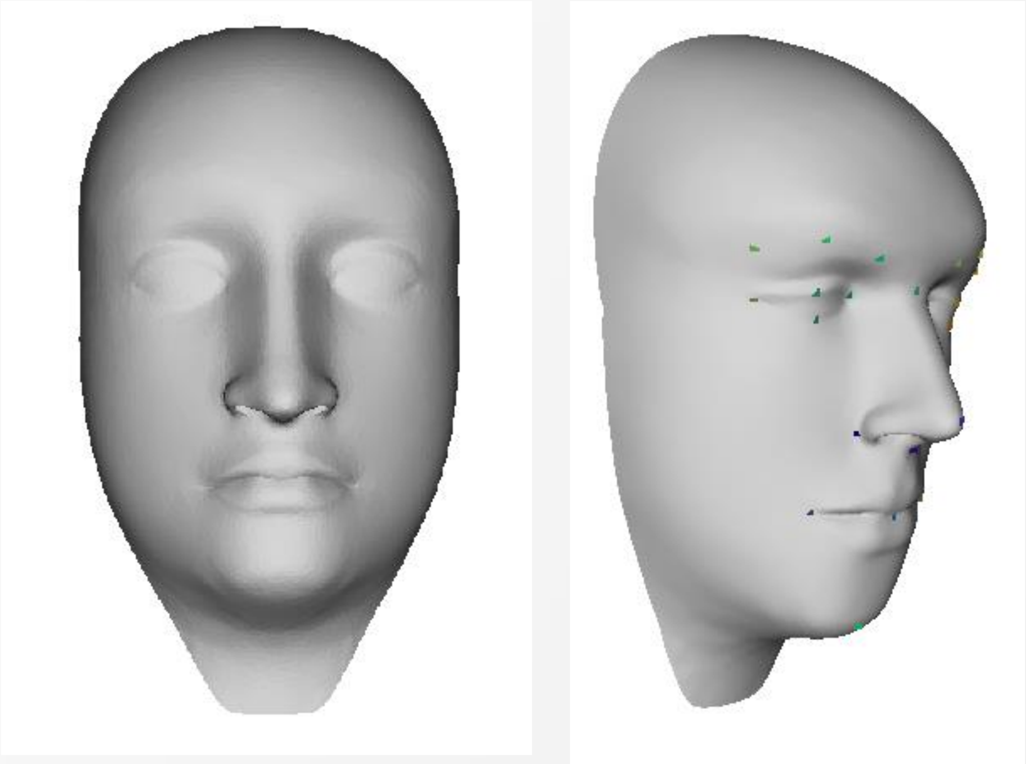
- Use the correspondences given by the landmarks to find a rigid alignment (e.g., rotation matrix computed via SVD).

Warping (non-rigid alignment)

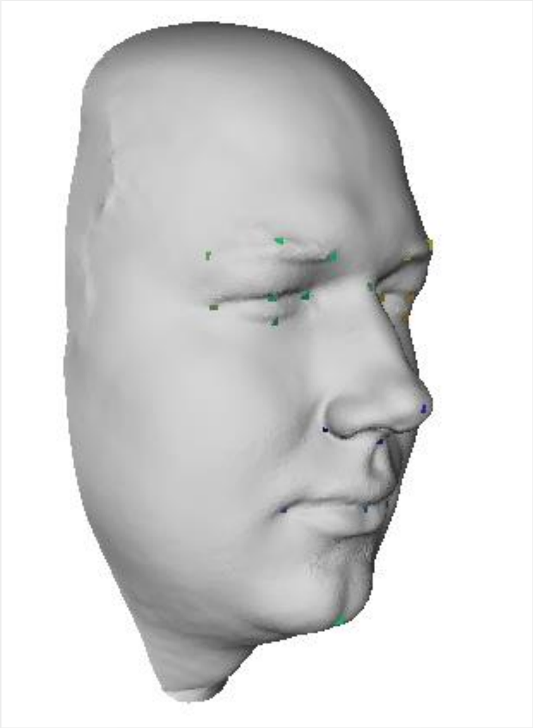
2 b). Warping

Goal: warp template to rigidly aligned scan, which provides the common triangulation

* Similar to assignment 5, but here we consider multiple constraints simultaneously



Non-rigid warp to match landmarks



After four warping iterations

Note: find a good resolution (number of faces) of the template face model

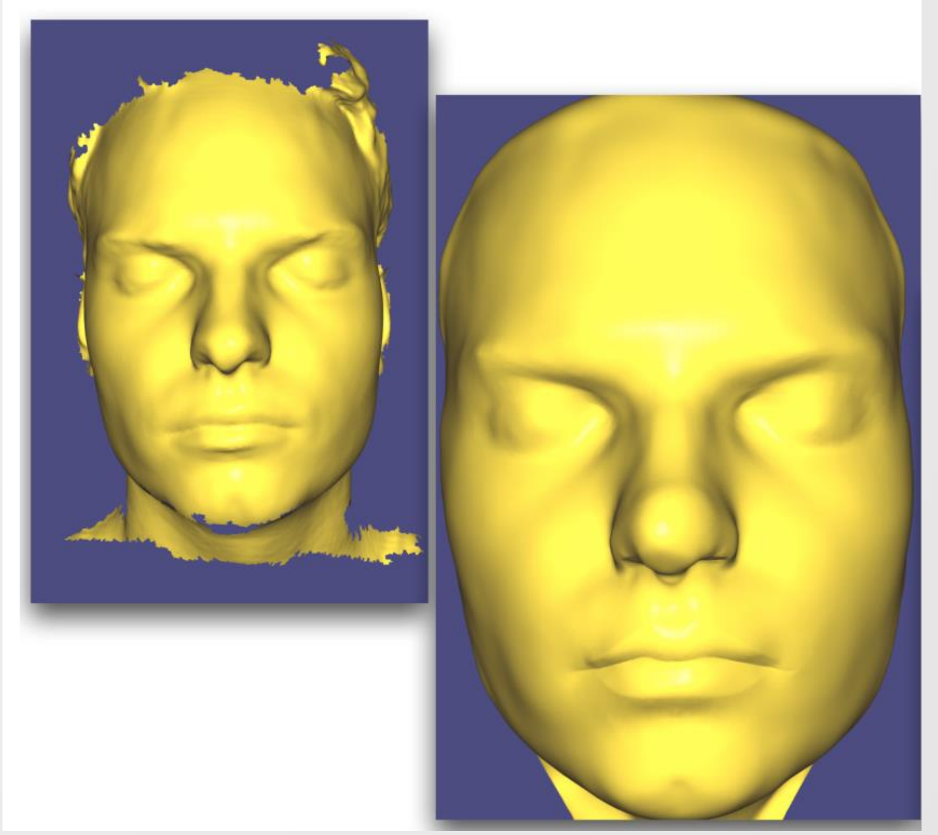
Warping

2.b) Warping: **suggest** method

Use Laplacian as a smoothness constraint

$$E_{warp} = ||Lx' - Lx||^2 + \lambda||Id|_{constr}x' - c||^2$$

X': unknown warped positions,
L: your favorite Laplacian (e.g. cotan weights on boundaries)
Id|const: selected positions to be constrained
c: target positions

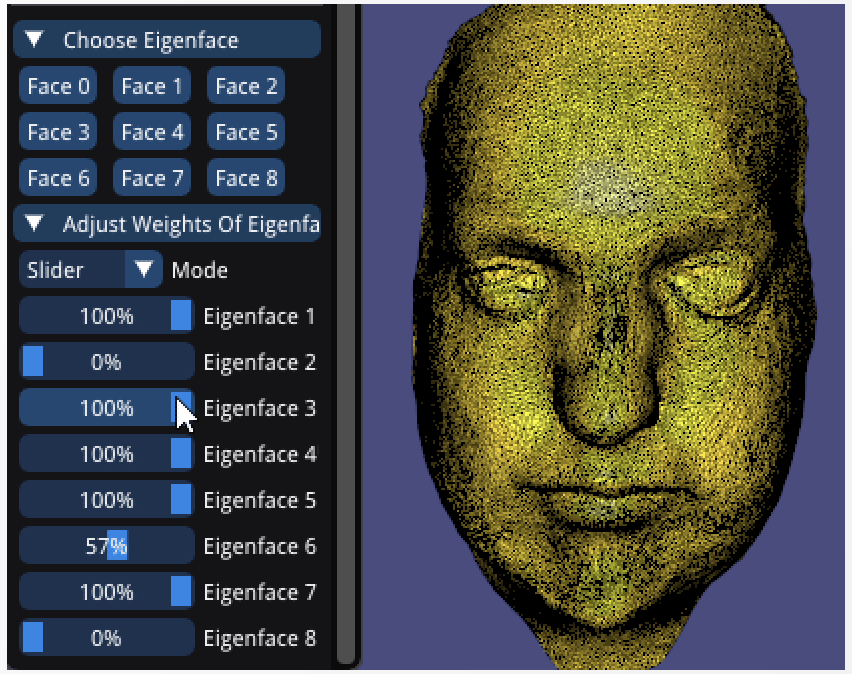


Note:

- Target and source should be rigidly aligned first!
- Keep the boundary fixed (it might also work to use the boundaries as soft constraints, but it is better to have a common boundary loops for the PCA part)
- Consider other and dynamic constraints like those vertices close enough to target face
- Please show in your presentation/report the results after each iteration

PCA face (unsupervised learning)

- PCA on face vectors to find our dominant variation / average mesh
- Morphing on eigen-space
  - for example, smile-to-neutral, personA-to-personB



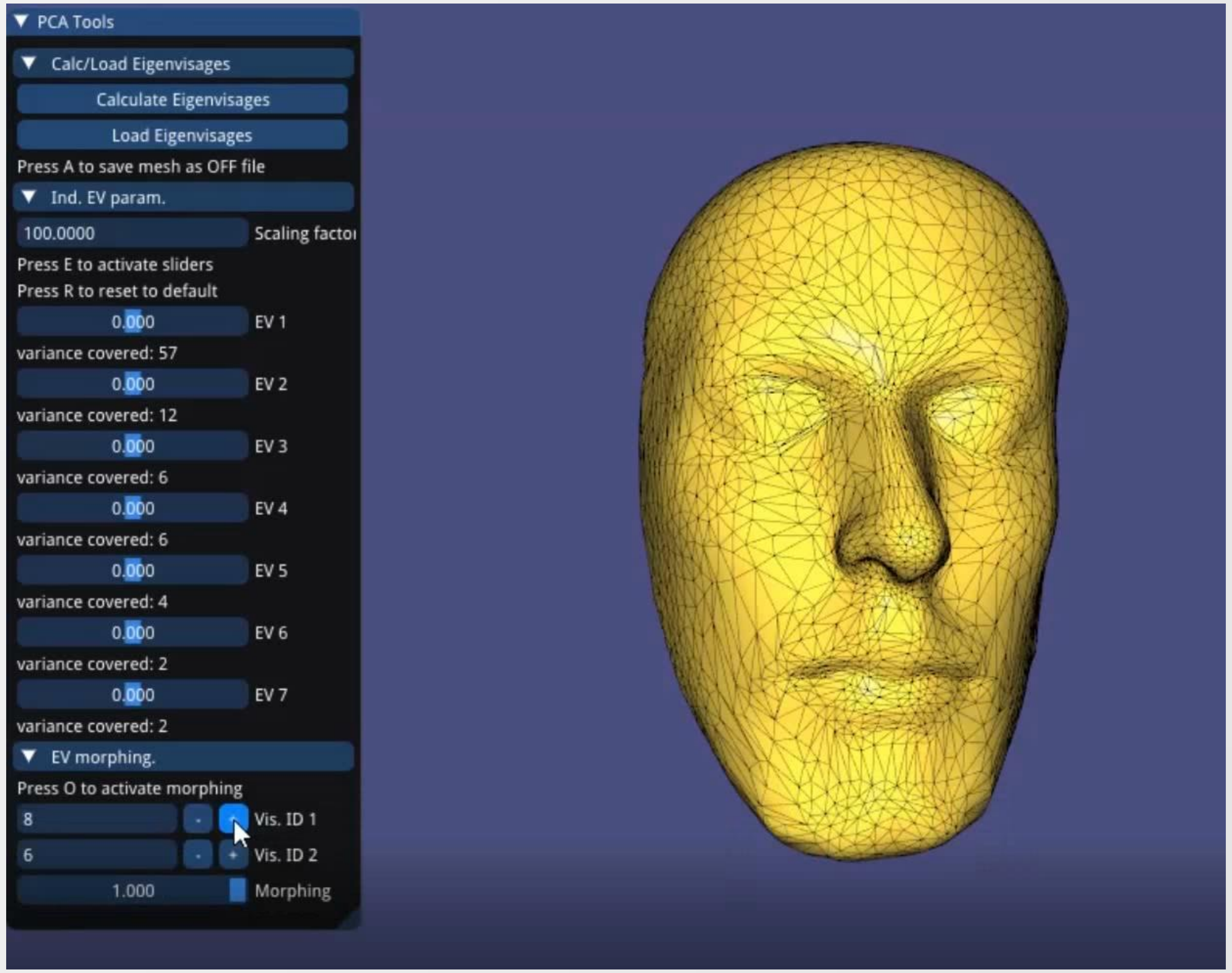
$$f_r = F_m + \sum_{i=0}^{n} e_i * w_i$$

PCA face morphing

- Some literature about PCA 3D face modeling:
  1) A Morphable Model For The Synthesis Of 3D Faces
  2) Singular Value Decomposition, Eigenfaces
  3) 3D Eigenfaces for Face Modeling.
  4) PCA and Face Recognition

$$w_i = (f_i - F_m)^T e_i$$

$$f_{morph} = F_m + \sum_{i=0}^{n} (w_i^{f_1} - m * (w_i^{f_1} - w_i^{f_2})) * e_i$$



Data to get started

- A template and some scanned faces.
  headtemplate.obj
- Landmarks of the scanned faces:
  File Format: vertexIndex<int> labelName<string>\n
- Rigidly aligned of one face:
  template_rigid_aligned.obj, peter_rigid_aligned.obj
  To get started with non-rigid warping and rigid alignment at once.
- Scanned data from last year
- Basel face model (if additional data needed)

Here is the link to the data: https://www.dropbox.com/t/fyHENhyO6rQiXAlY or https://www.dropbox.com/sh/bu975ppo5thn4jq/AADCufEVjXlPok0HUnOYl5HIa?dl=0

Disclaimer: we do **not** hold copyrights of these data, please use them for study in this class exclusively!

Define Input/Output APIs

- Output (Scanning):

  Colored obj files.
- Output (Landmark extraction):

  Vertex indices, Label

  As a txt format.

  E.g: label_name<oneword/int> vertex1<int> vertex2<int> vertex3<int> bari1<float> bari2<float> bari3<float>
- Output (common triangulation via template registration)

  Colored obj files, where vertices and faces enumerated the same way
- Result (Applications):

  Interactive GUI allowing to explore Morphing/ Dominant PCA, allow to save STLs or objs of the results.

Tipps: use whatsapp or slack for communication, use google doc to define the APIs, and merge early.

Grading

15% : Landmark selection
40% : Face alignment (rigid and non-rigid)
15% : The PCA of faces
10% : The UI (eigenface mixture weight tuning) and interpolation quality
20% : Project presentation and demo
Bonus (20%): Personalized 3D faces and Leaning-based face space

A package per group:

- Code (mainly written with libigl) and data
- Slides, a short report with screenshots and **work division**

Presentation: 02.06.2021, 10:15
Package submission deadline (no more new feature): 04.06.2021, 11:00

Bonus 1: use your own 3D faces for morphing

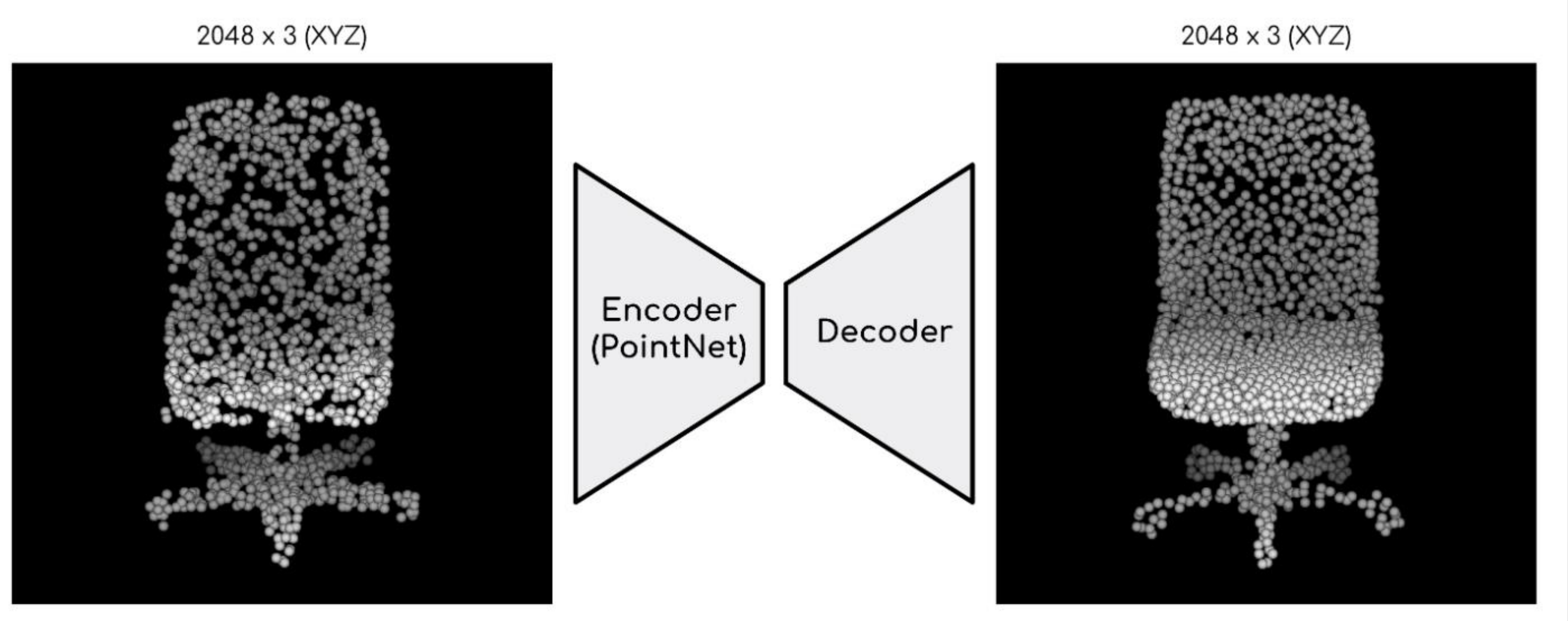
Possible solution 1:

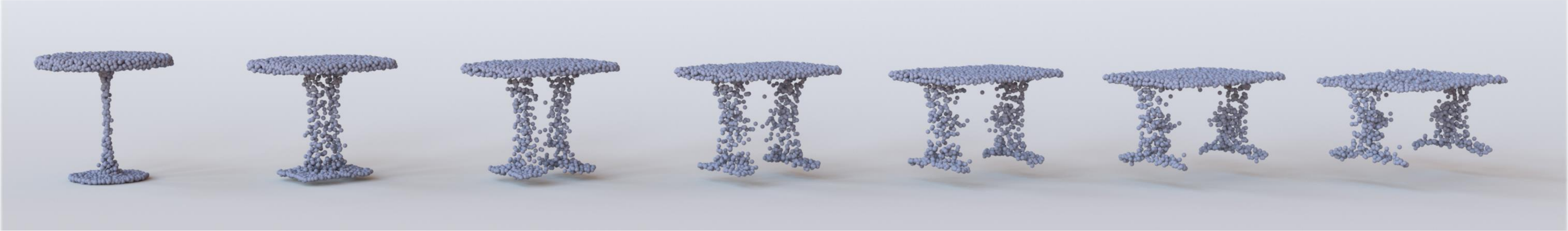
- Use existing mobile app (e.g., Capture) to obtain 3D faces

Possible solution 2:

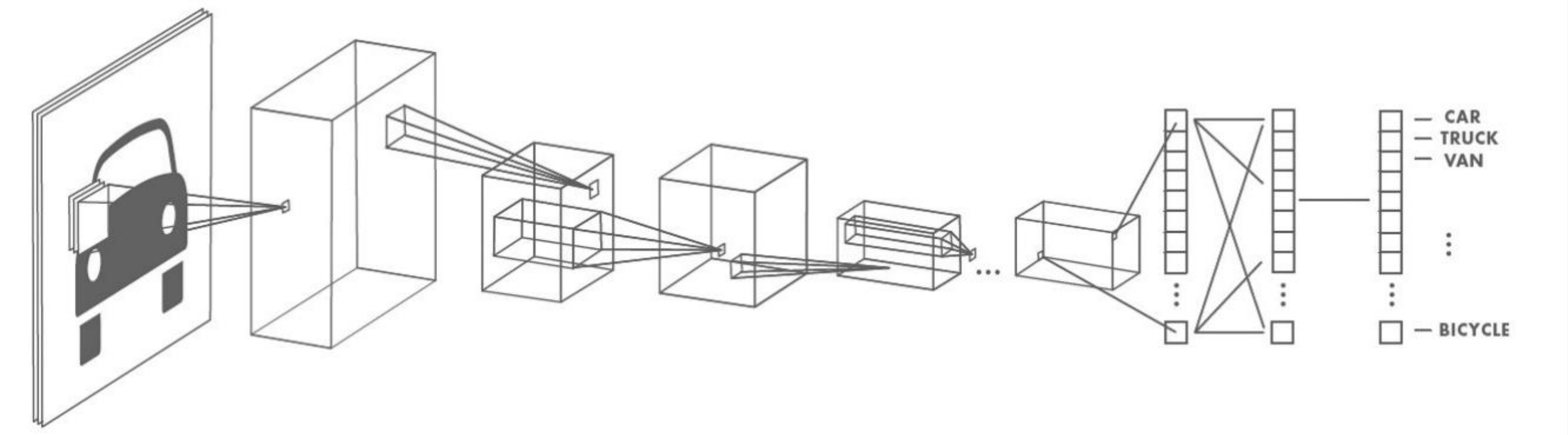
- Take some photos of your face
- Use photogrammetry code or software to reconstruct 3D faces from the photos, such as SMVS, Colmap, openMVG, or VisualSVM

Bonus 2: Learning-based face space





Bonus 2: Learning-based face space

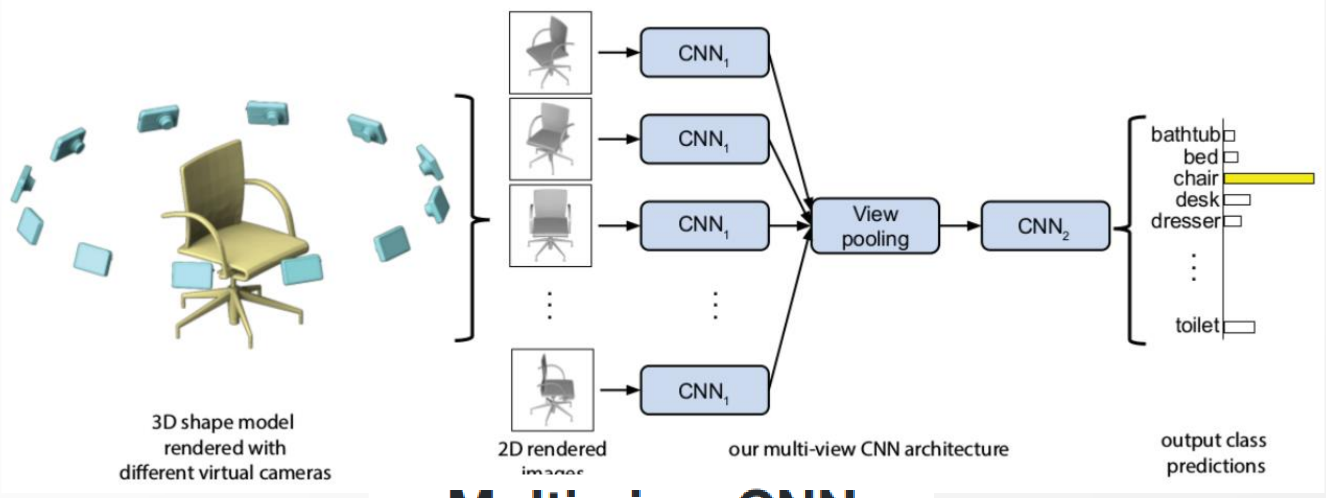


LeCun et al. 1989

Convolutional filters (Translation invariance+self-similarity)

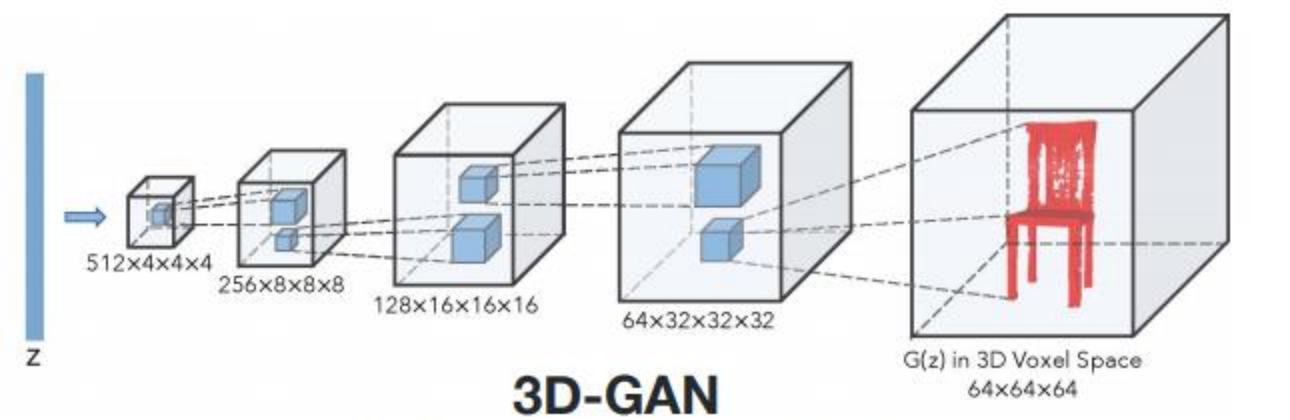
Bonus 2: Learning-based face space

Image-based



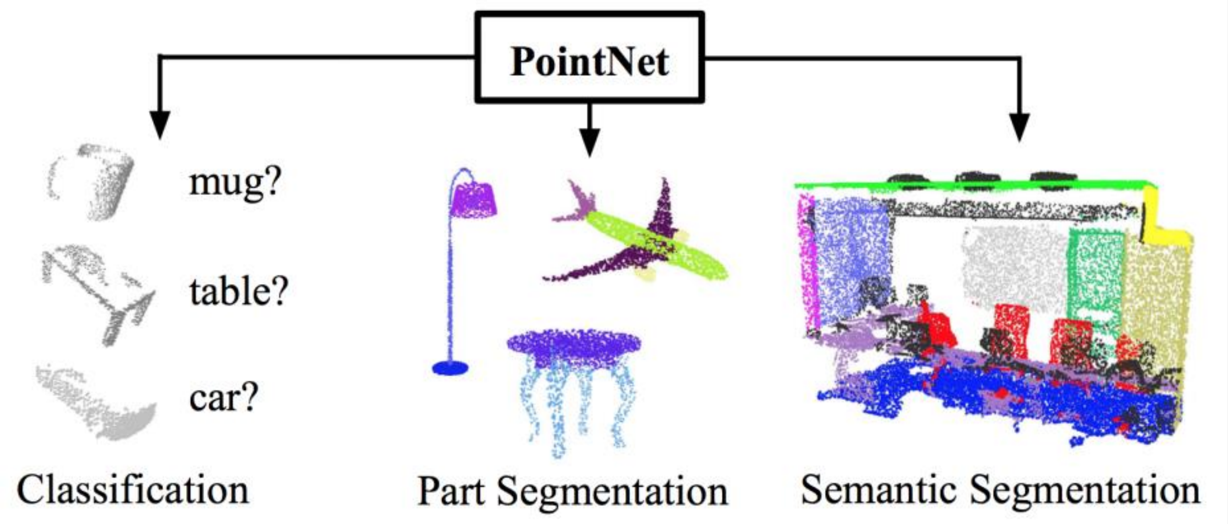
Multi-view CNN
Hang Su et al., ICCV 2015

Volumetric

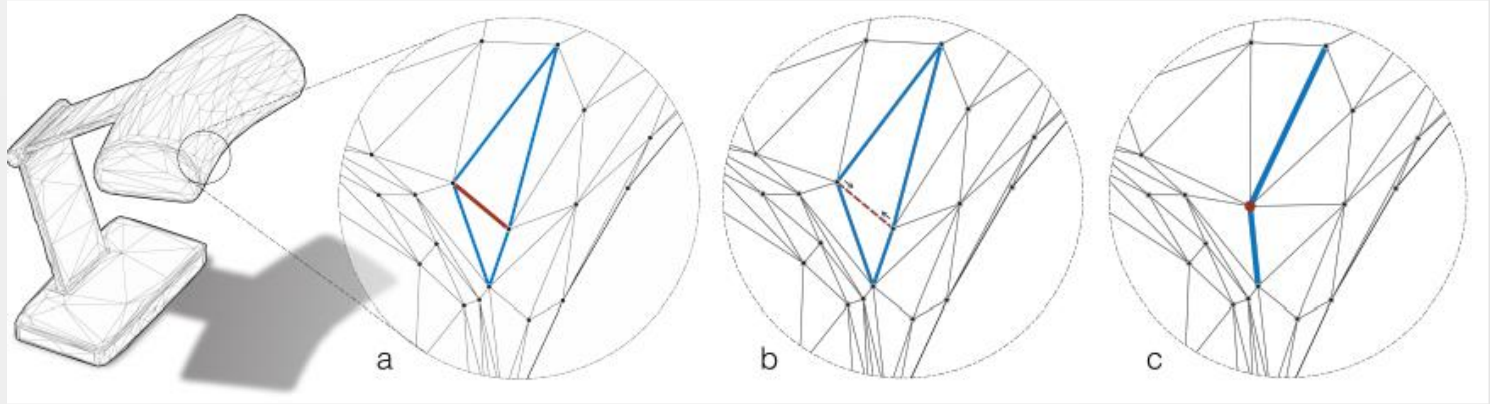


3D-GAN
J. Wu et al., NIPS 2016

Point-based



Mesh or Graph based



MeshCNN
Rana et al. SIGGRAPH 2019

Bonus 2: Learning-based face space

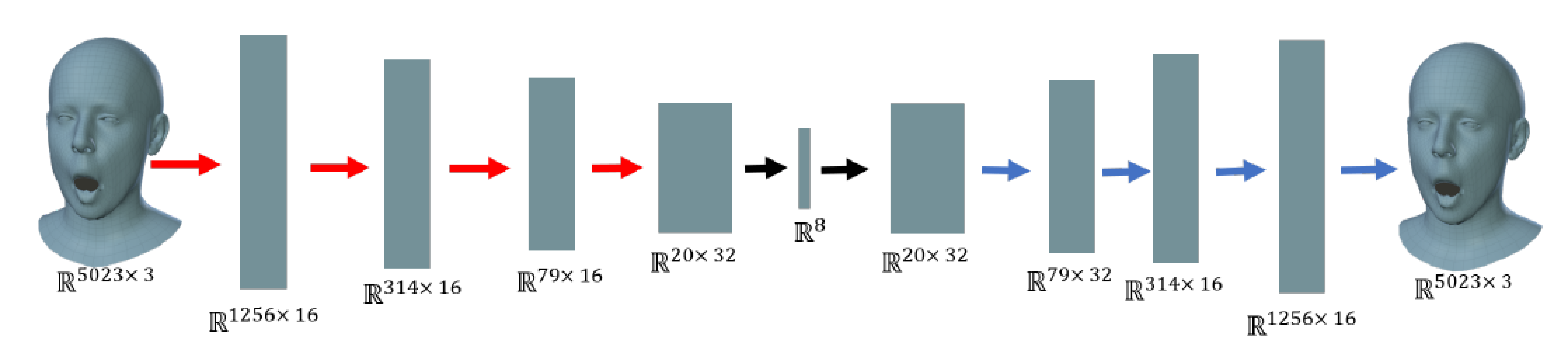
Useful geometric learning libraries:

- Pytorch: easier to prototype compared with tensor-flow
- Pytorch Geometric: the implementation of Edge Convolution and PointNet can be useful for this task
- Pytorch3D: the implementation of Mesh R-CNN can be inspiring

Bonus 2: Learning-based face space

Example codes and papers:

- Learning representations and generative models for 3d point clouds
- PointNet implementation in pytorch geometric
- Generating 3D faces using Convolutional Mesh Autoencoders
- Semantic Deep Face Models co-authored by Thabo Beeler (guest lecturer on 26.05.21)



Questions?

Thank you!