Statistical / AI Shape Modeling for Computer-Assisted Medicine

Marek BUCKI – Twinsight / UGA
marek.bucki@univ-grenoble-alpes.fr

Université Grenoble Alpes

twinsight
twinsight-medical.com

3 Statistical / AI Shape Modeling 2r Computer-Assisted Medicine 1

Marek BUCKI – Twinsight / UGA
marek.bucki@univ-grenoble-alpes.fr

At the end of this lecture you will:

- Understand Estimators, Covariances, Correlations
- Understand **Principal Components Analysis**
- Be able to compute a **Linear Statistical Shape Model**
- Be ready to tackle the **Data Challenge!**

Université Grenoble Alpes

twinsight
twinsight-medical.com

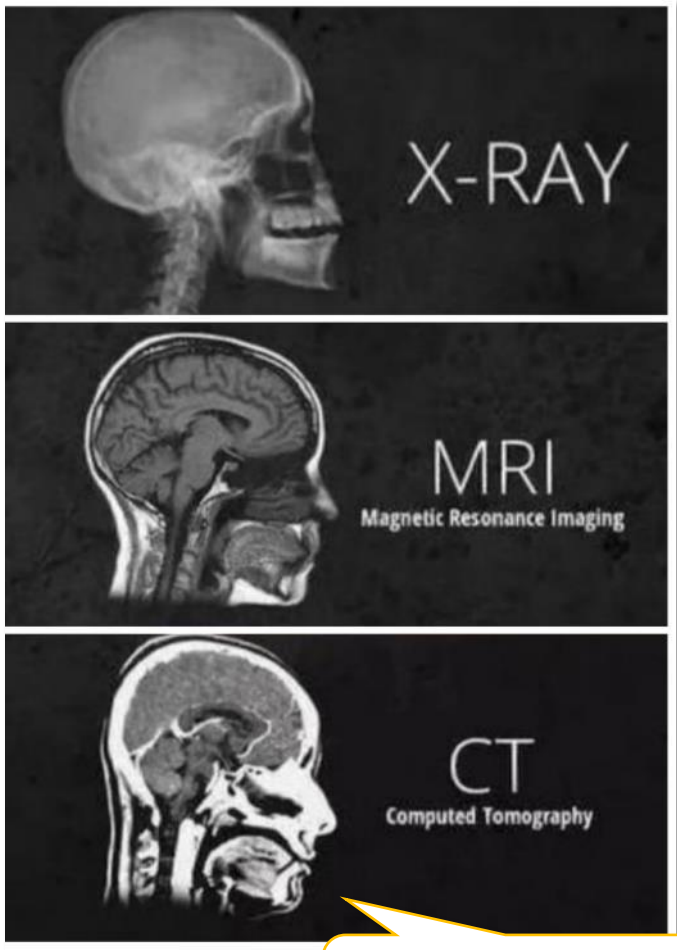
Statistical / AI Shape Modeling for Computer-Assisted Medicine

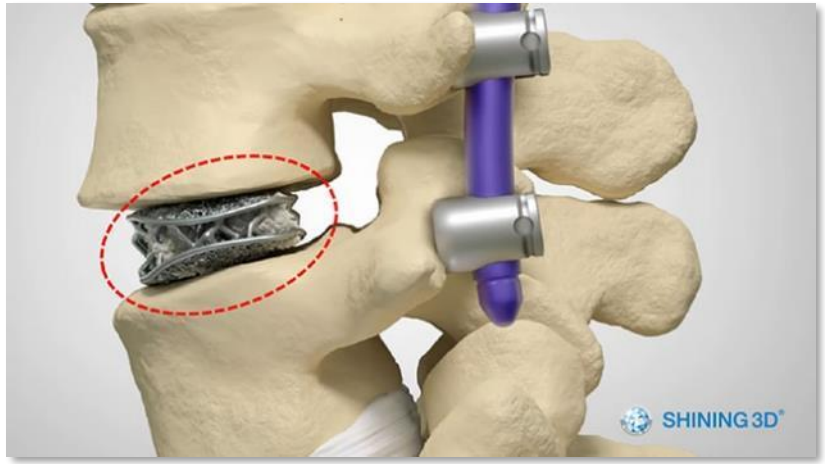
Université Grenoble Alpes

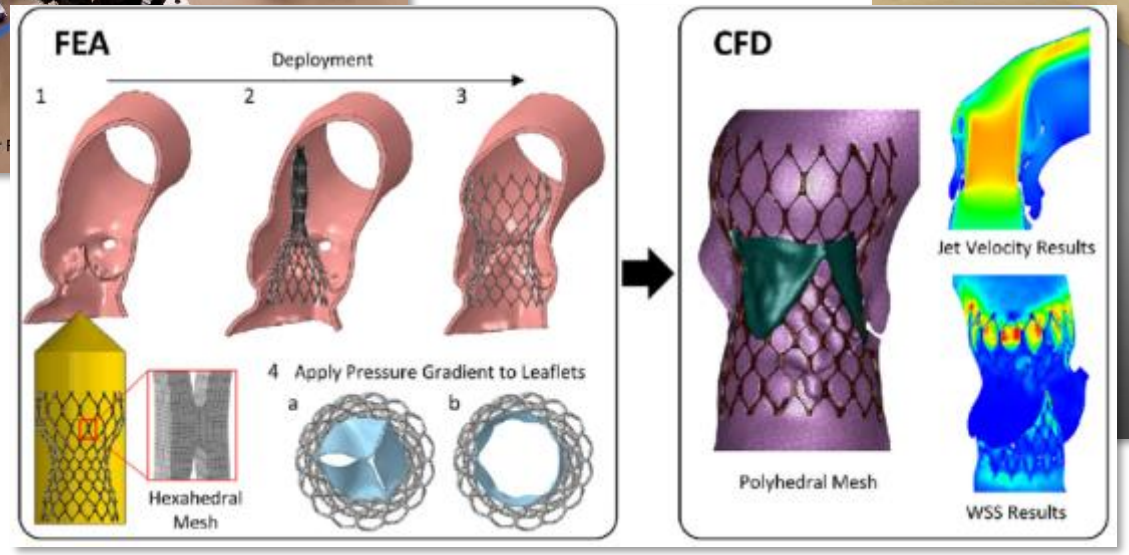
Marek BUCKI – marek.bucki@univ-grenoble-alpes.fr

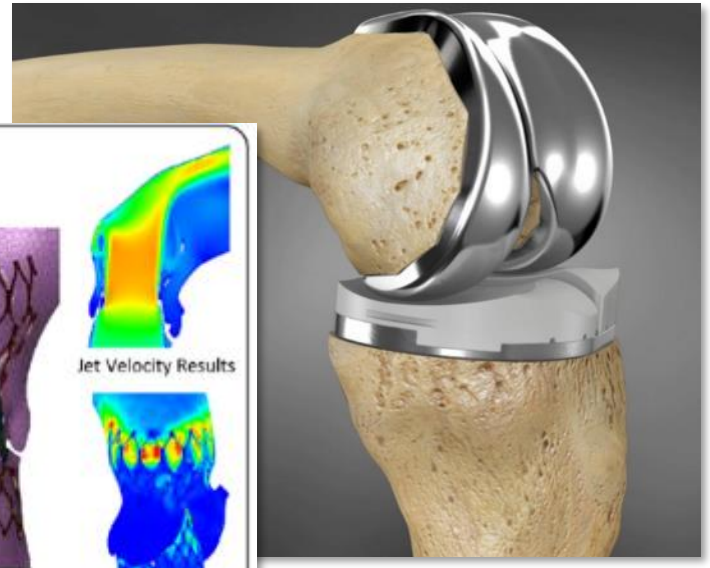
twinsight
twinsight-medical.com

Statistical / AI Shape Modeling for Computer-Assisted Medicine









Soft tissues vs **Hard** tissues

Organic complexity

Marek BUCKI – marek.bucki@univ-grenoble-alpes.fr

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Grenoble = *France's Silicon Valley* ☺

- → Surgiqual Insititute (robotics, imaging, regulatory affairs)
- → Koelis (urology: prostate biopsy, early cancer detection)
- → Blue Ortho (orthopedic surgery: knee, shoulder, ankle)
- → Pixyl (neurology, medical imaging + AI)
- → Microlight 3D (medical 3D printing)
- → Diabeloop (closed loop diabetes management + AI)
- → eCential Robotics (robotics and medical imaging)
- → UroMems (urology: artificial sphincter)
- → Imactis (medical imaging: interventional radiology)
- → Texisense (pressure ulcer prevention)
- → Twinsight (digital twins for orthopedic surgery)
- → …

Laboratories

- → CEA (Commissariat à l'Energie Atomique)
- → CNRS (Centre National de la Recherche Scientifique)
- → INRIA (Institut National de Recherche en Sciences et Technologies du Numérique)
- → …

Marek BUCKI – marek.bucki@univ-grenoble-alpes.fr

Statistical / AI Shape Modeling for Computer-Assisted Medicine

GENERAL PROBLEM STATEMENT in Computer Assisted Medicine: Therapy / Surgery / Rehabilitation / Prevention / …

1 – A patient has a **health problem**: trauma, blood pressure, Parkinson's, cancer, diabetes, osteoarthritis, …

2 – Clinical investigation is hard… **we need to "look inside" the patient!**

→ **Invasive**: coloscopy, arthroscopy, biopsy, … (anything that would "hurt")

→ **Non-invasive**: medical imaging

- Ultrasound (US)
- Magnetic Resonance Imaging (MRI)
- Computed Tomography (CT)
- Single Photon Emission Computed Tomography (SPECT)
- …

CRITICAL SOFTWARE

High quality requirements

3 – Analyzing medical images is hard… **we need algorithms to help us with the diagnosis!**

- Classical image processing
- AI (works extremely well)
- Mathematical modeling
- …

4 – Okay, now we know what's wrong with the patient, **we want to cure them!**

→ Off the shelf solution (standard medical treatment, "one size fits all")

→ Personalized medicine (personalized medicine, personalized treatment)

→ Surgical treatment… But surgery is hard… **we need assistance to navigate the body!**

Statistical / AI Shape Modeling for Computer-Assisted Medicine

… a **MODEL** can be **"understood"** and **"reasoned"** about …

1 – Mathematical **MODELING**

Classical inference

- Experiments / Data / Observations → **INDUCTIVE REASONING** → (updated) **MODEL** of reality (= idealization)
- Observations + **MODEL** → **DEDUCTIVE REASONING** → **Predictions** about reality

AI inference

- Experiments / Data / Observations → **… GPU MAGIC + kW …** → Set of weights in a **NN** architecture…
- Observations + **NN** → **Basic linear algebra** → **Predictions** about reality

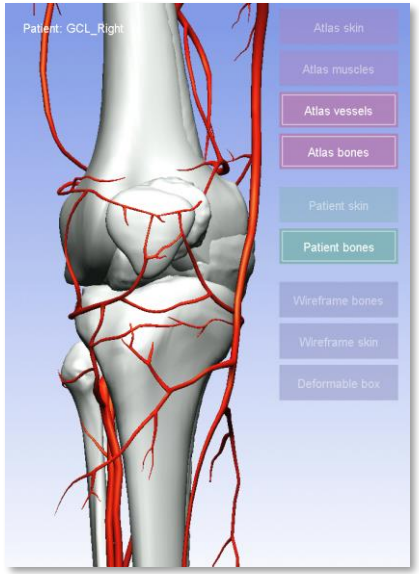
AI is **great**, but …

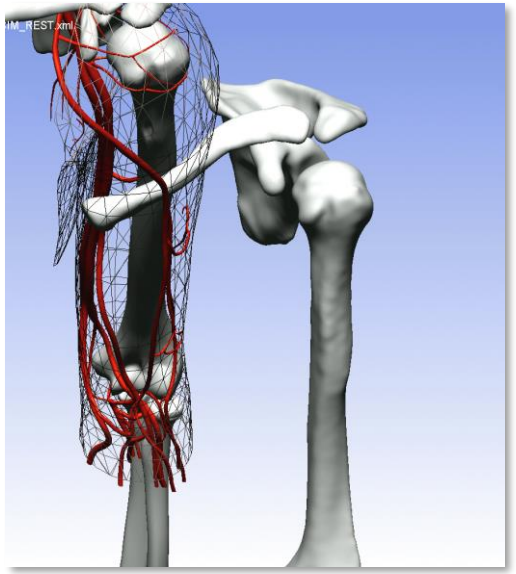
… a **NN can't**

2 – Anatomical **SHAPES**

We are all made the **SAME (Anatomy)** but …
… we are all **DIFFERENT (Morphology)**

→ **3D** representations of a patient's
INDIVIDUAL MORPHOLOGY





Université Grenoble Alpes

Marek BUCKI – marek.bucki@univ-grenoble-alpes.fr

twinsight
twinsight-medical.com

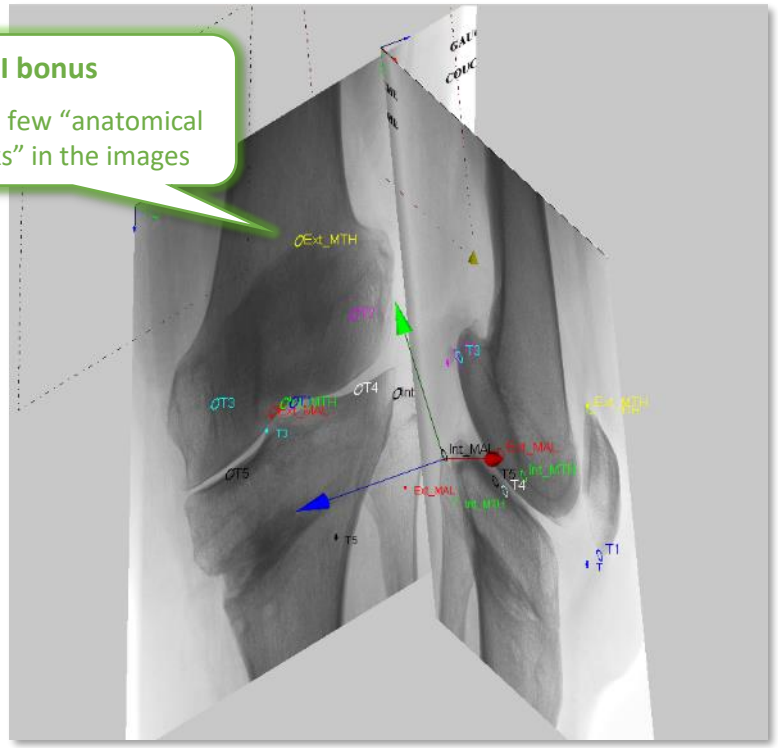
Statistical / AI Shape Modeling for Computer-Assisted Medicine

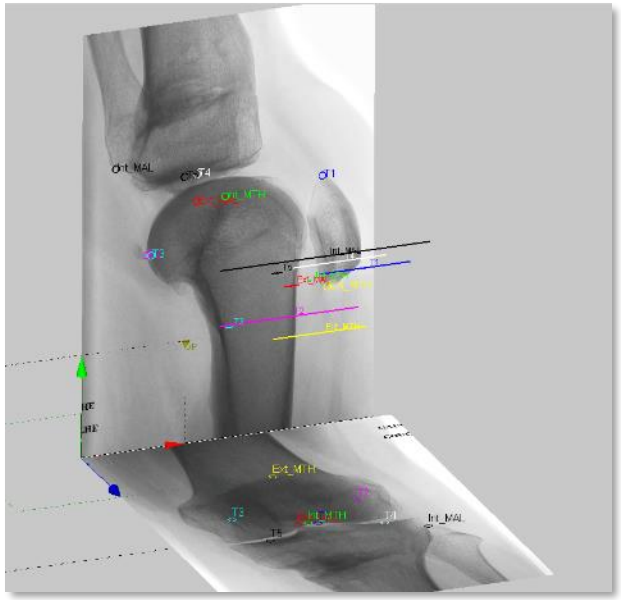
Real-world example: **Shape Modeling** in **Computer Assisted Medicine**

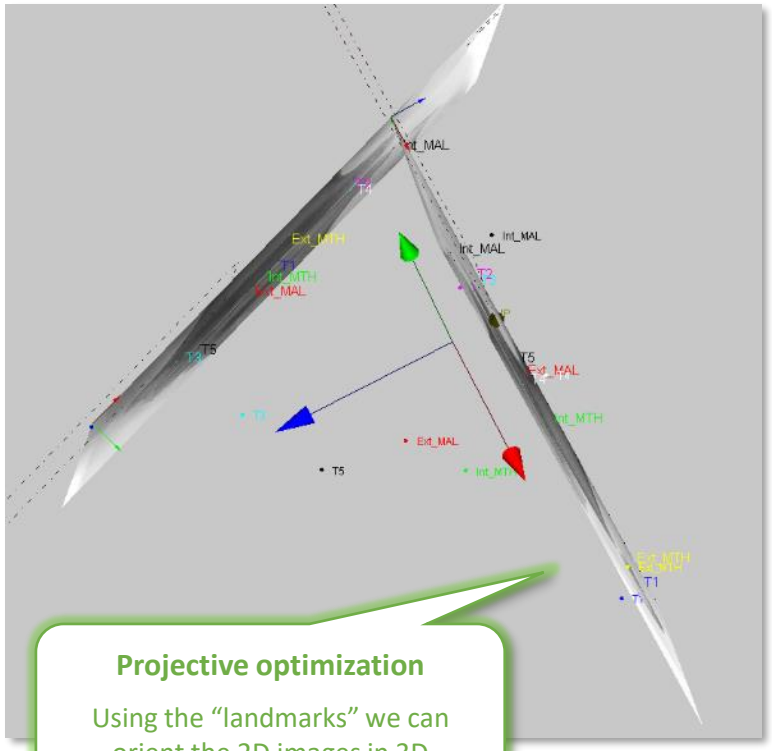
Problem: We want to treat in 3D a knee injury … but we **only have 2D X-rays** of the patient's knee

AI bonus

We have a few "anatomical landmarks" in the images







Projective optimization

Using the "landmarks" we can orient the 2D images in 3D

Marek BUCKI – marek.bucki@univ-grenoble-alpes.fr

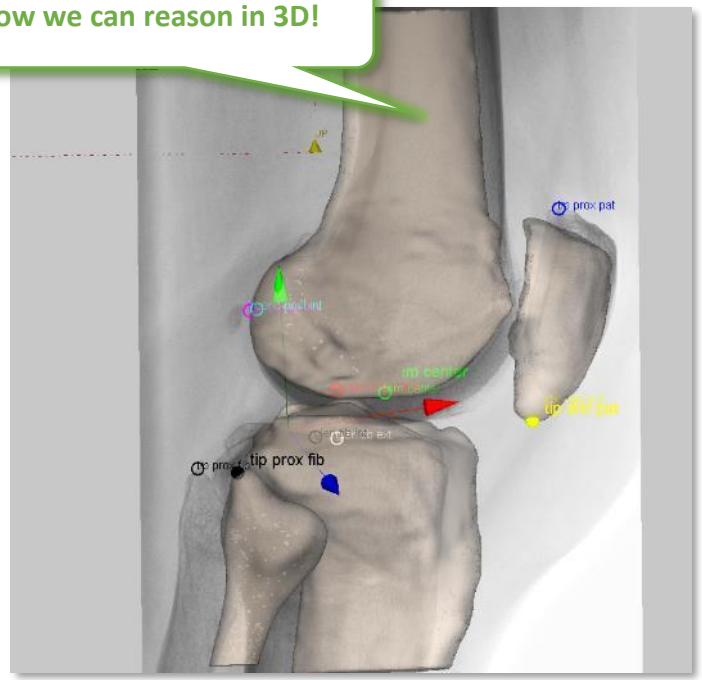
Statistical / AI Shape Modeling for Computer-Assisted Medicine

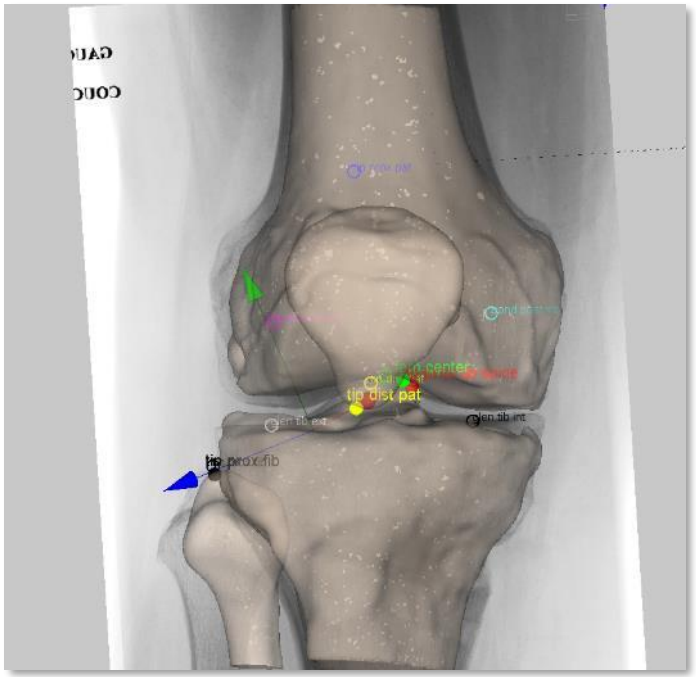
Real-world example: **Shape Modeling** in **Computer Assisted Medicine**

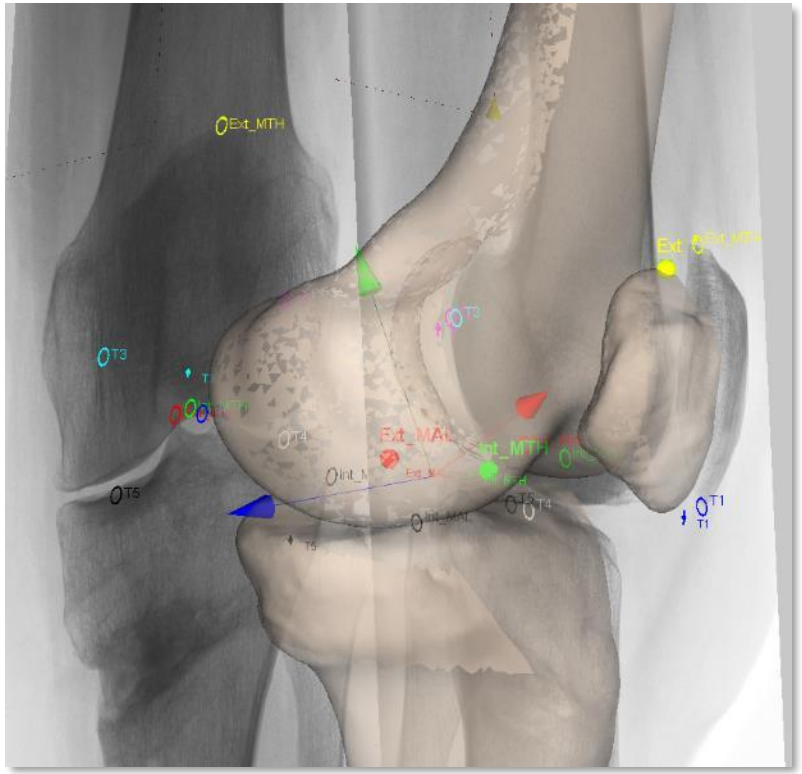
Problem: We want to treat in 3D a knee injury … but we **only have 2D X-rays** of the patient's knee

Solution: AI + classical maths ➔ 3D mathematical model of the patient's knee

Now we can reason in 3D!





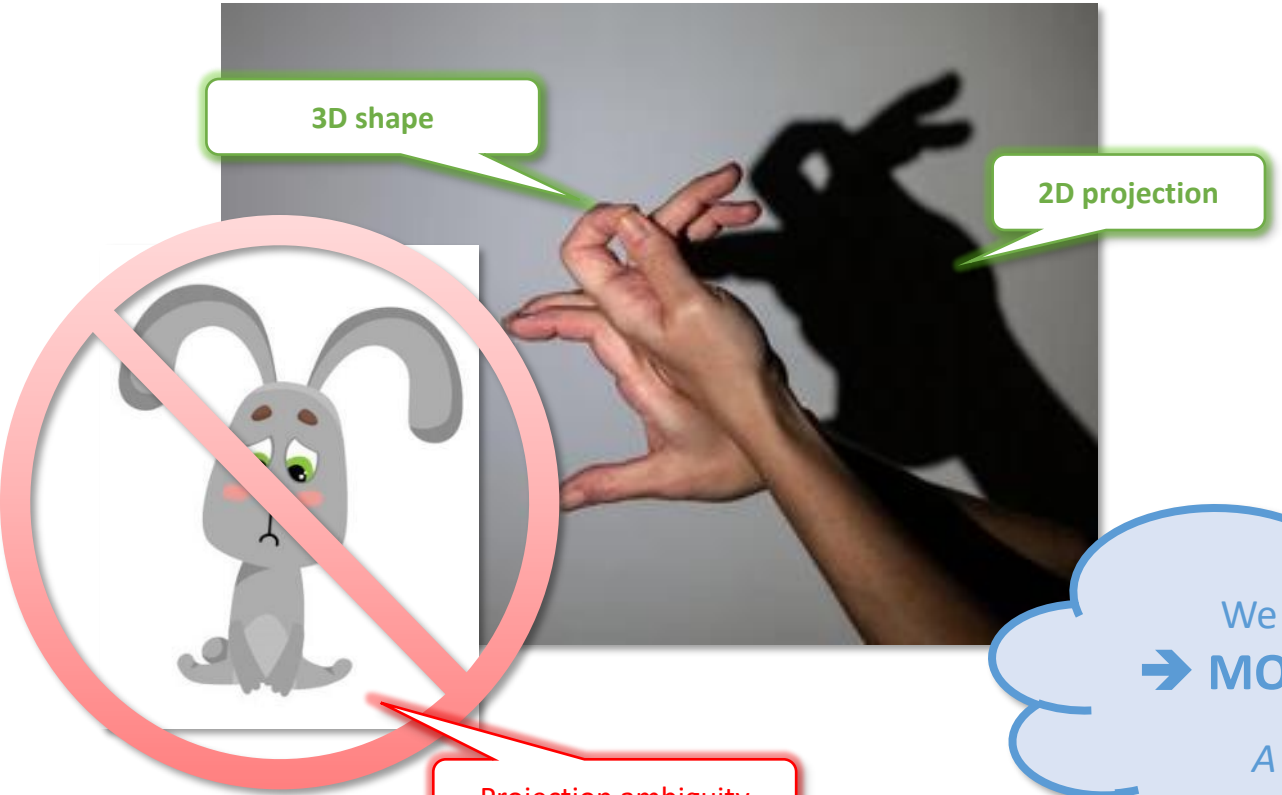


Marek BUCKI – marek.bucki@univ-grenoble-alpes.fr

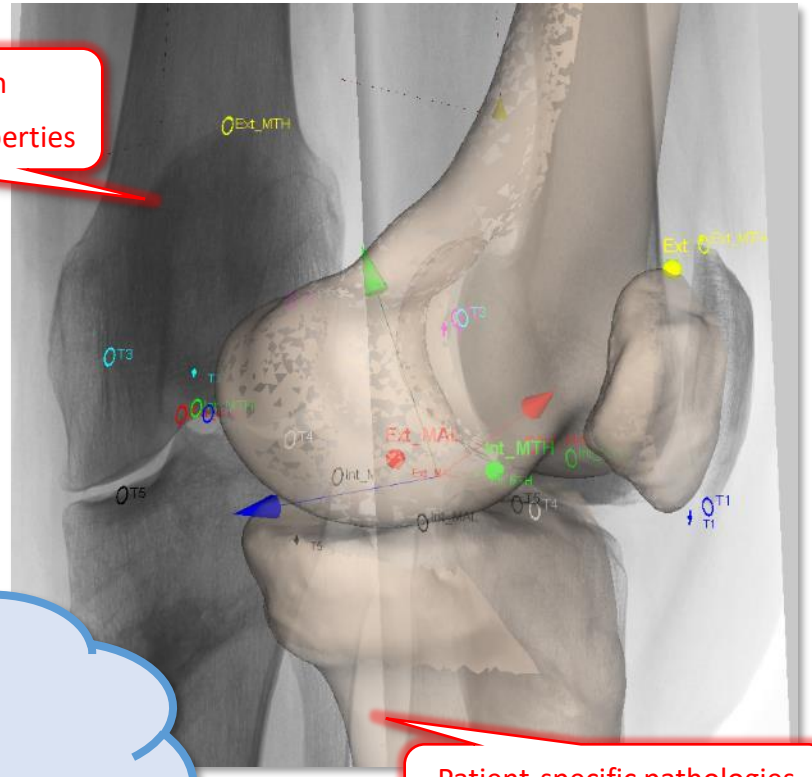
Statistical / AI Shape Modeling for Computer-Assisted Medicine

Real-world example: **Shape Modeling** in **Computer Assisted Medicine**

Question: How **accurate** is this reconstruction?



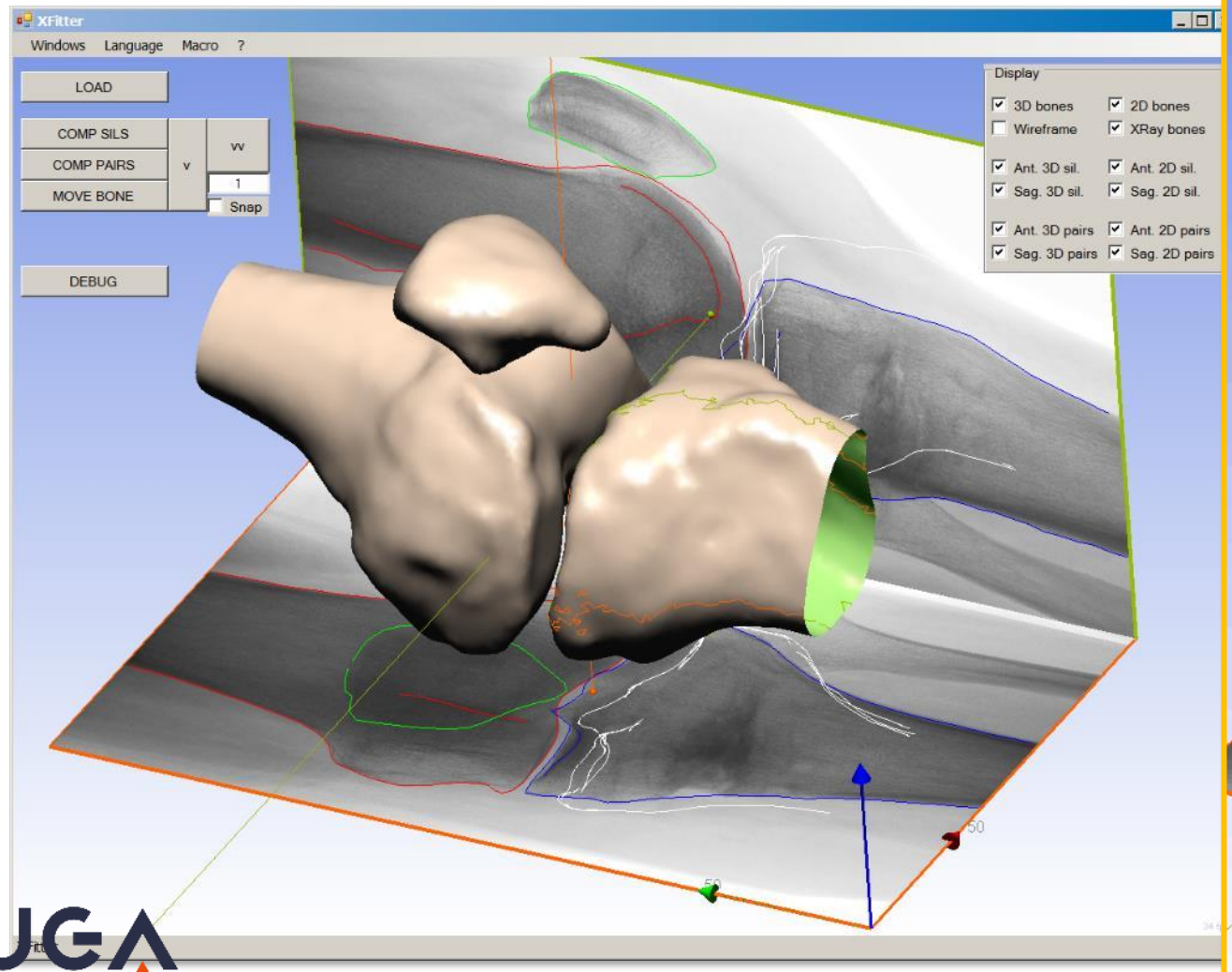
We need some kind of
➔ **MODEL of REALITY** ➔
A priori knowledge

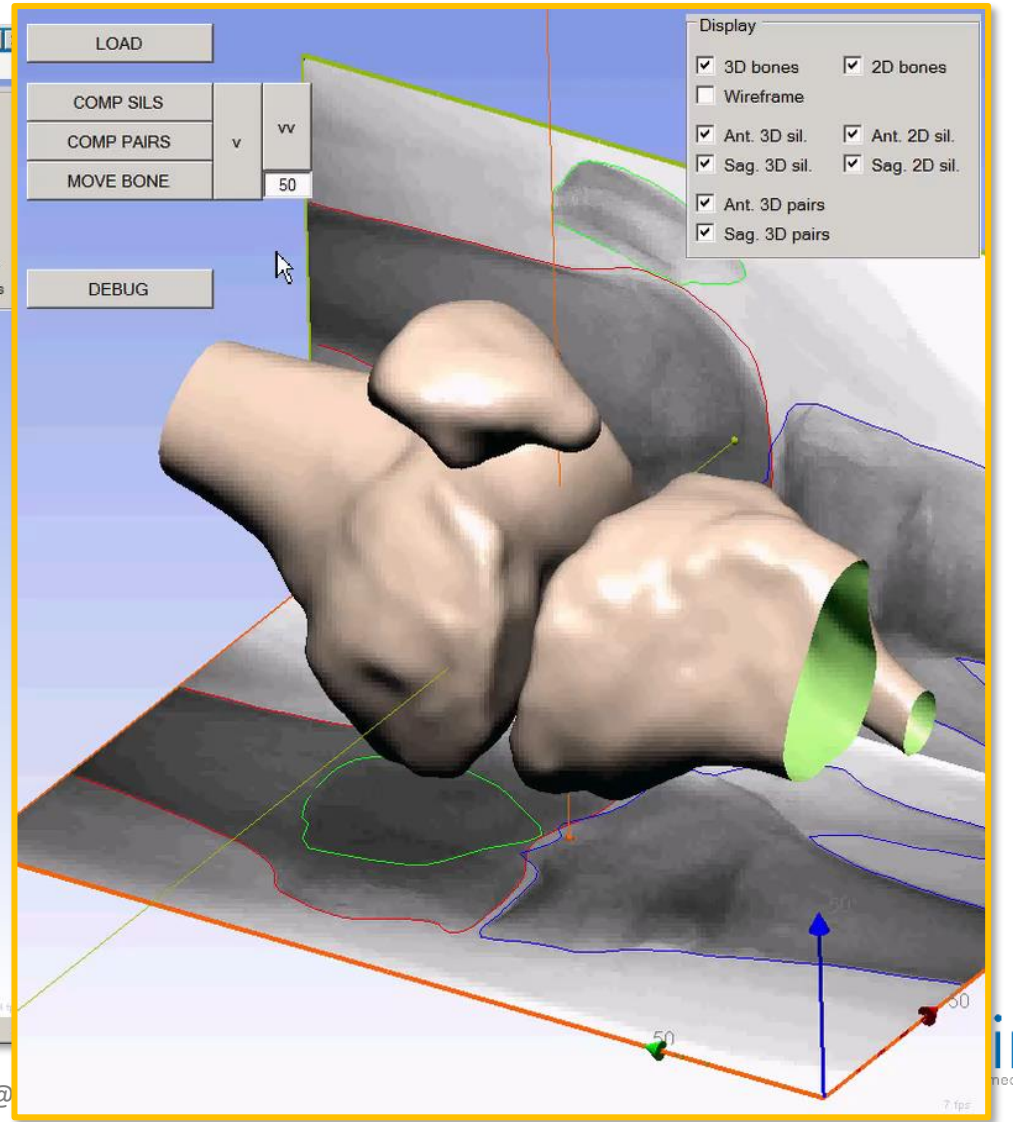


Marek BUCKI – marek.bucki@univ-grenoble-alpes.fr

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Real-world example: **Shape Modeling** in **Computer Assisted Medicine**





Statistical / AI Shape Modeling for Computer-Assisted Medicine

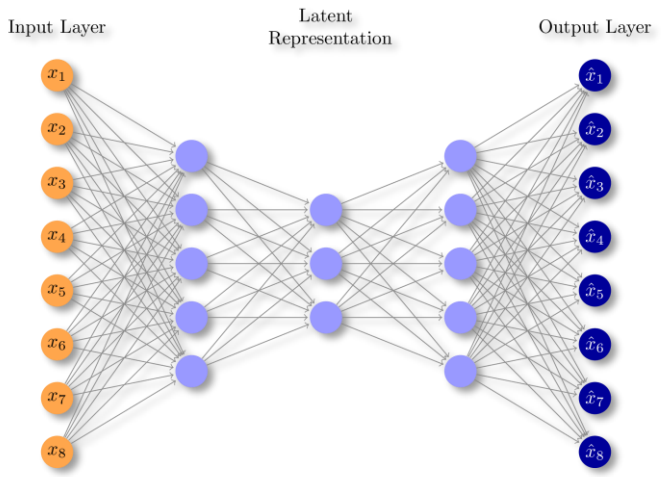
Real-world example: **Shape Modeling** in **Computer Assisted Medicine**

Question: How **STATISTICALLY LIKELY** is this reconstruction?

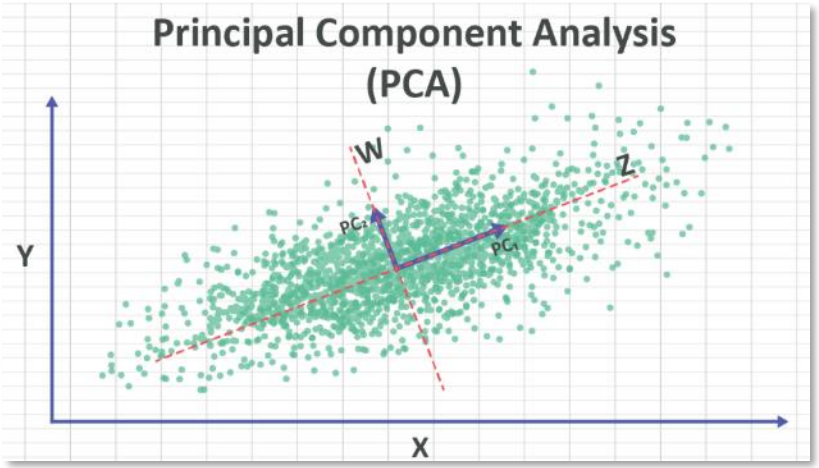
Possible ways to go

Classical stats
- Mean shape, variance / covariance analysis, principal deformation modes ➔ **Principal Component Analysis**
- Non-linear hypothesis (because of course **it's non-linear**!) ➔ **Kernel / Non-linear - Principal Component Analysis**

AI magic
- Learned dimension reduction, latent space representation ➔ **Autoencoders (built-in non-linearity)**



VS



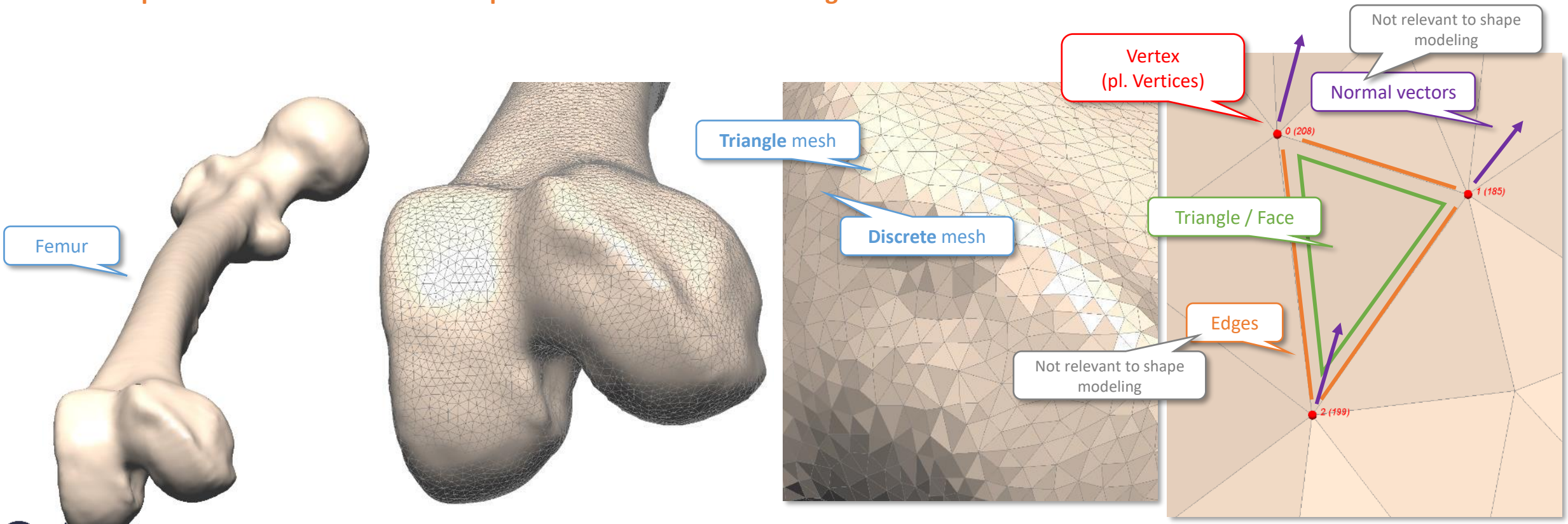
Marek BUCKI – marek.bucki@univ-grenoble-alpes.fr

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Next: an introduction to statistical **Shape Modeling** using **PCA**

kPCA, Autoencoders, … → *Left for **you** to explore during the challenge!*

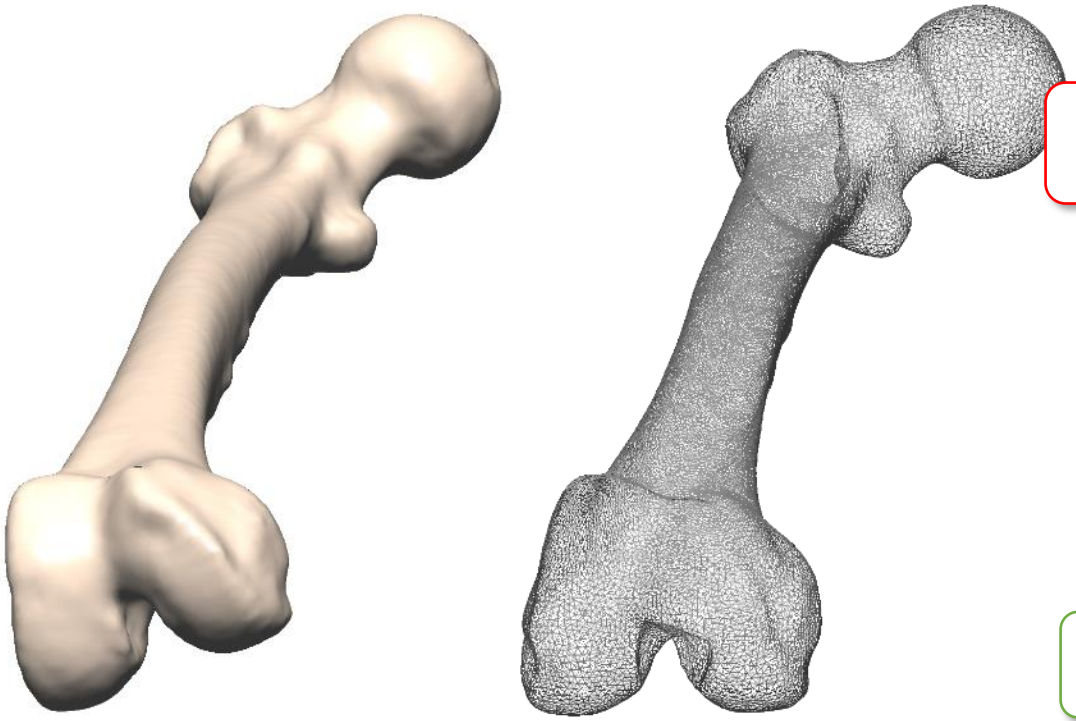
But first … → **Representation of anatomical shapes** + **Overview of the Challenge Dataset**

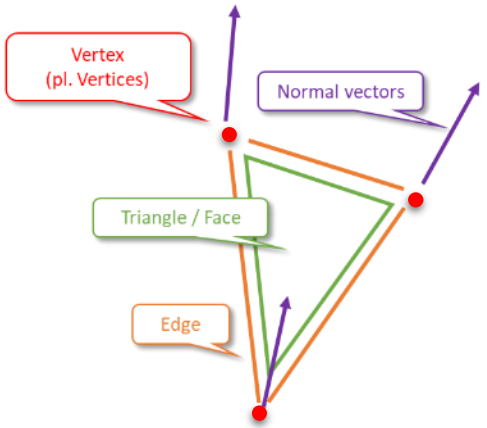


Marek BUCKI – marek.bucki@univ-grenoble-alpes.fr

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Representation of anatomical shapes





Wavefront OBJ
Index-based format

GEOMETRY / MORPHOLOGY
Specific to each "subject"

```
# 18291 vertice(s)
v -212.68 27.92 9.63
v -212.68 28.02 6.49
v -212.72 28.80 7.71
v -212.68 27.13 7.87
...
```

```
# 18291 normal(s)
vn -0.99 -0.04 0.03
vn -0.99 -0.01 -0.09
vn -0.99 0.03 -0.02
vn -0.99 -0.08 -0.00
...
```

TOPOLOGY / ANATOMY
Shared by ALL "subjects"

```
# 36578 triangle(s)
f 14//14 43//43 18//18
f 43//43 14//14 36//36
f 36//36 14//14 24//24
...
```

Indices in tables
OBJ: indices start with **1**

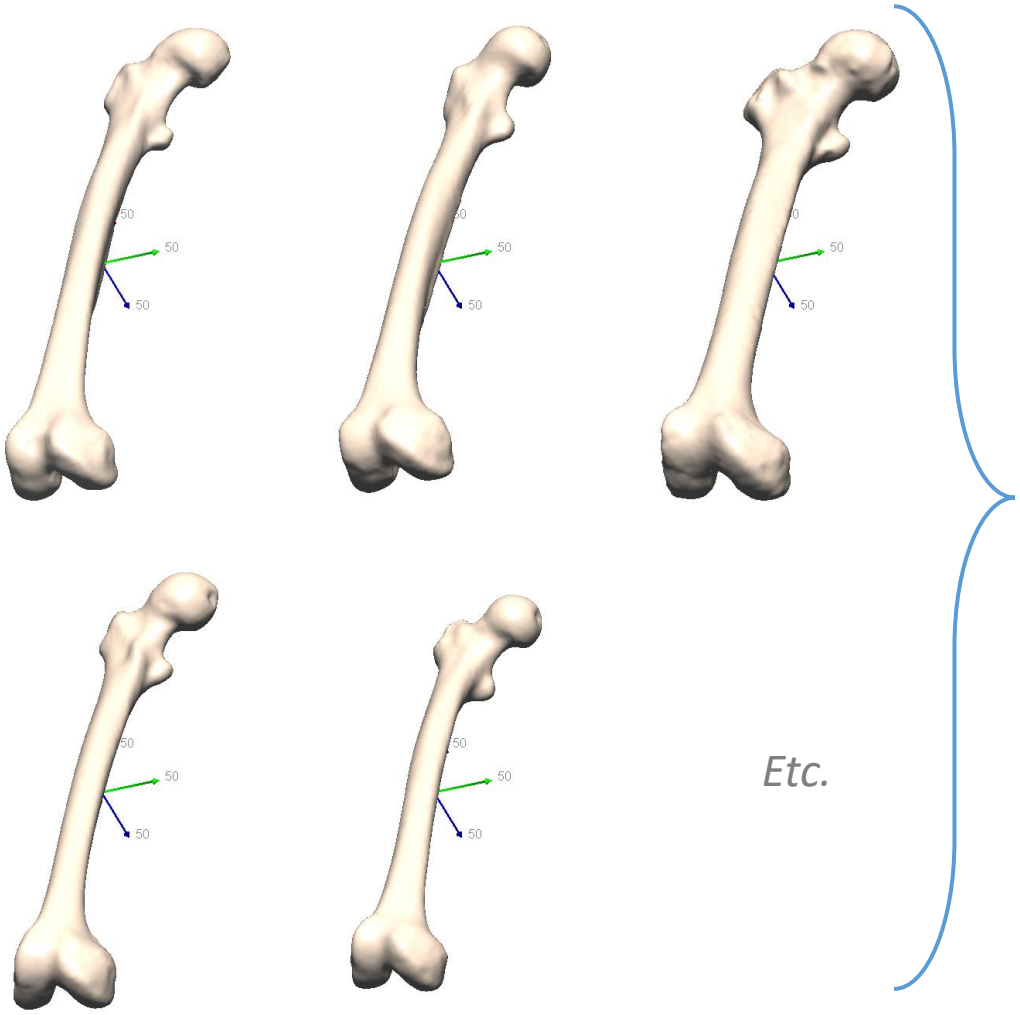
f **14**//14 **43**//43 **18**//18

Marek BUCKI – marek.bucki@univ-grenoble-alpes.fr

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Overview of the Challenge Dataset : 24 “subjects” = 24 “aligned” femurs

L_Femur_11____EJ.obj.FINAL.obj	3 425 Ko	3D Object
L_Femur_13_DECIM.obj.FINAL.obj	3 426 Ko	3D Object
L_Femur_14_DECIM.obj.FINAL.obj	3 426 Ko	3D Object
L_Femur_15_DECIM.obj.FINAL.obj	3 426 Ko	3D Object
L_Femur_16_DECIM.obj.FINAL.obj	3 426 Ko	3D Object
L_Femur_19_DECIM.obj.FINAL.obj	3 427 Ko	3D Object
L_Femur_20_DECIM.obj.FINAL.obj	3 427 Ko	3D Object
L_Femur_21_DECIM.obj.FINAL.obj	3 426 Ko	3D Object
L_Femur_23_DECIM.obj.FINAL.obj	3 427 Ko	3D Object
L_Femur_24_DECIM.obj.FINAL.obj	3 427 Ko	3D Object
R_Femur_01_DECIM.obj.FINAL.obj	3 427 Ko	3D Object
R_Femur_02_DECIM.obj.FINAL.obj	3 426 Ko	3D Object
R_Femur_03_DECIM.obj.FINAL.obj	3 426 Ko	3D Object
R_Femur_04_DECIM.obj.FINAL.obj	3 426 Ko	3D Object
R_Femur_05_DECIM.obj.FINAL.obj	3 425 Ko	3D Object
R_Femur_06_DECIM.obj.FINAL.obj	3 426 Ko	3D Object
R_Femur_07_DECIM.obj.FINAL.obj	3 426 Ko	3D Object
R_Femur_08_DECIM.obj.FINAL.obj	3 425 Ko	3D Object
R_Femur_09_DECIM.obj.FINAL.obj	3 426 Ko	3D Object
R_Femur_10_DECIM.obj.FINAL.obj	3 426 Ko	3D Object
R_Femur_12_DECIM.obj.FINAL.obj	3 425 Ko	3D Object
R_Femur_17_DECIM.obj.FINAL.obj	3 426 Ko	3D Object
R_Femur_18_DECIM.obj.FINAL.obj	3 427 Ko	3D Object
R_Femur_22_DECIM.obj.FINAL.obj	3 426 Ko	3D Object



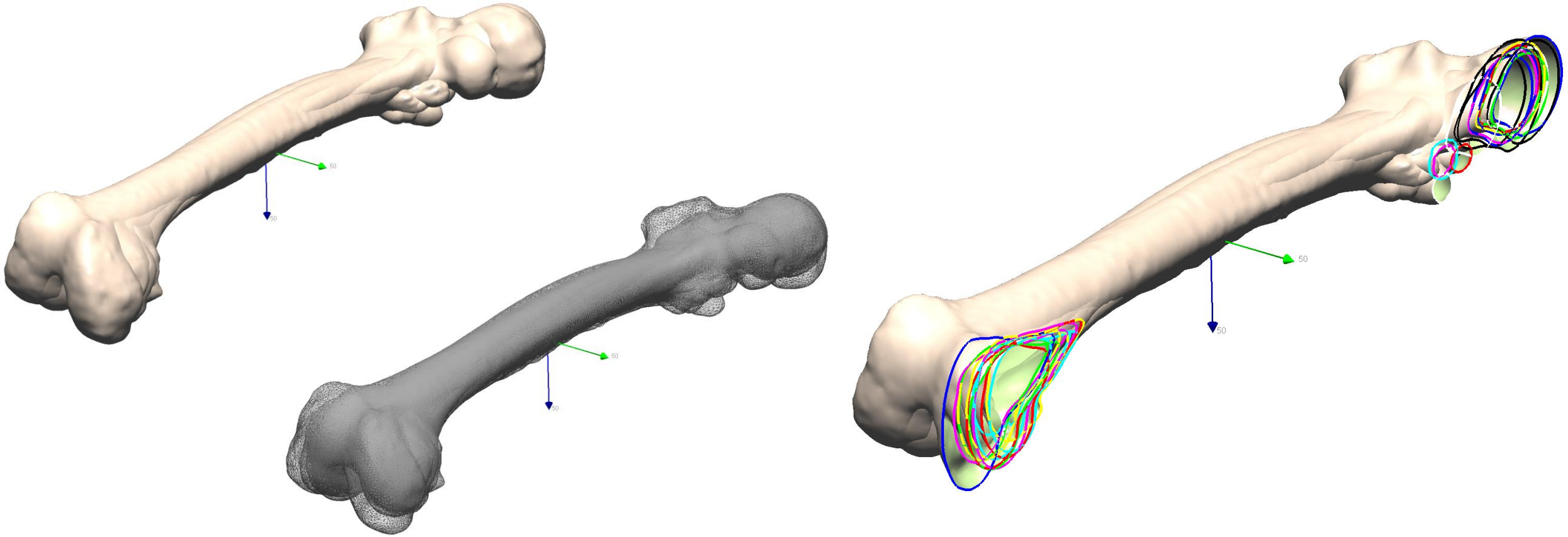
- All subjects = **RIGHT** femurs
- Male / female, all ages… (no extreme pathologies)
- All aligned in the same “World” referential
- **18,291 vertices**
- **36,578 triangles**
- **18,291 normals** (unused for PCA)

➔ Same **TOPOLOGY** (Anatomy)
➔ Different **GEOMETRIES** (Morphologies)

Marek BUCKI – marek.bucki@univ-grenoble-alpes.fr

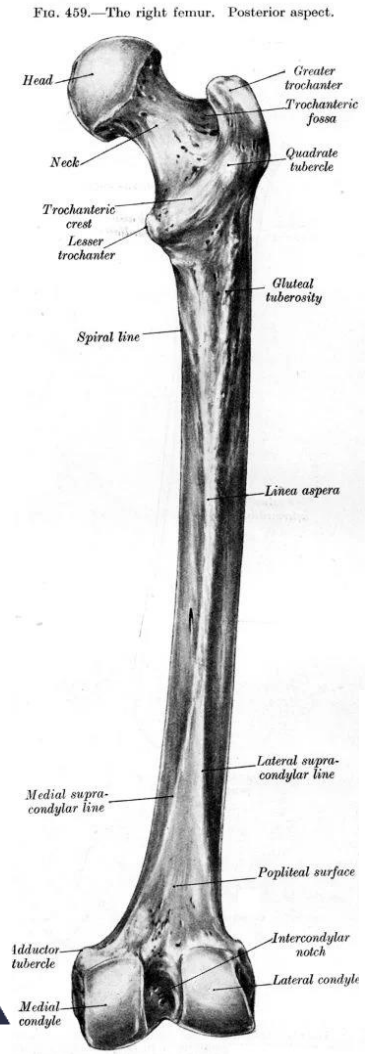
Statistical / AI Shape Modeling for Computer-Assisted Medicine

Overview of the Challenge Dataset : 24 “subjects” = 24 “aligned” femurs



Statistical / AI Shape Modeling for Computer-Assisted Medicine

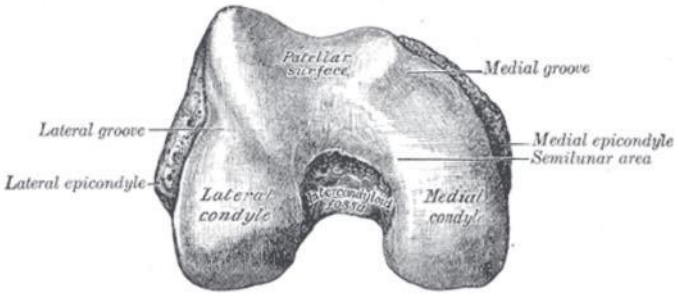
Overview of the Challenge Dataset : 24 “subjects” = 24 “aligned” femurs

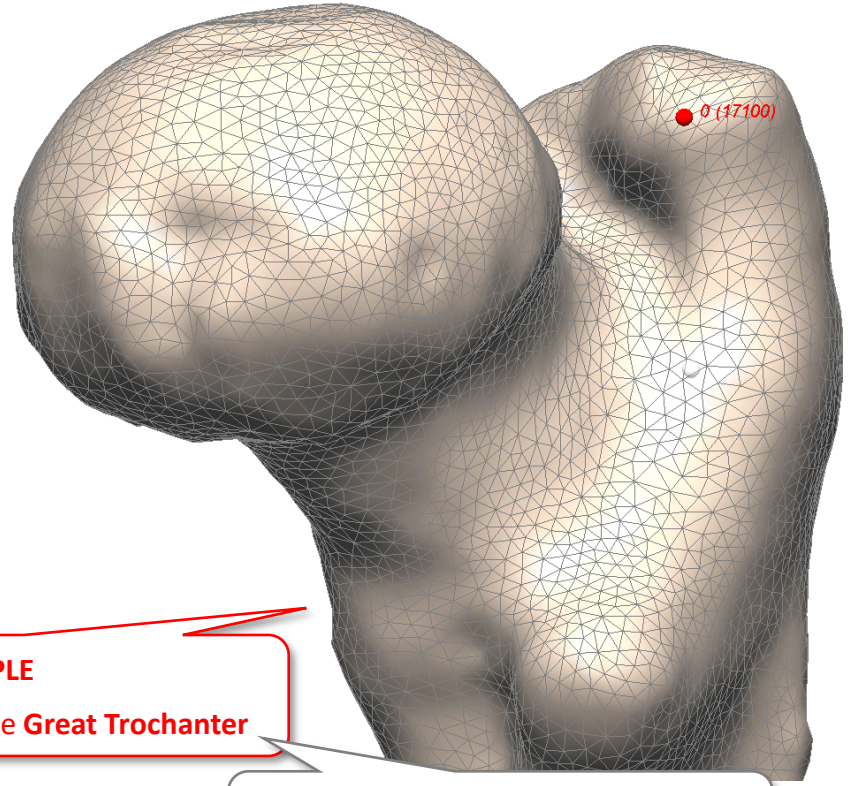


“**Aligned**”, “**Registered**” means each vertex always corresponds to the **SAME SPECIFIC ANATOMICAL POINT** in **ALL SUBJECTS**

All subjects share the same **VERTEX SEMANTICS** (meaning) ➔ **Anatomy**

… but they have different **VERTEX POSITIONS** ➔ **Morphology**





EXAMPLE

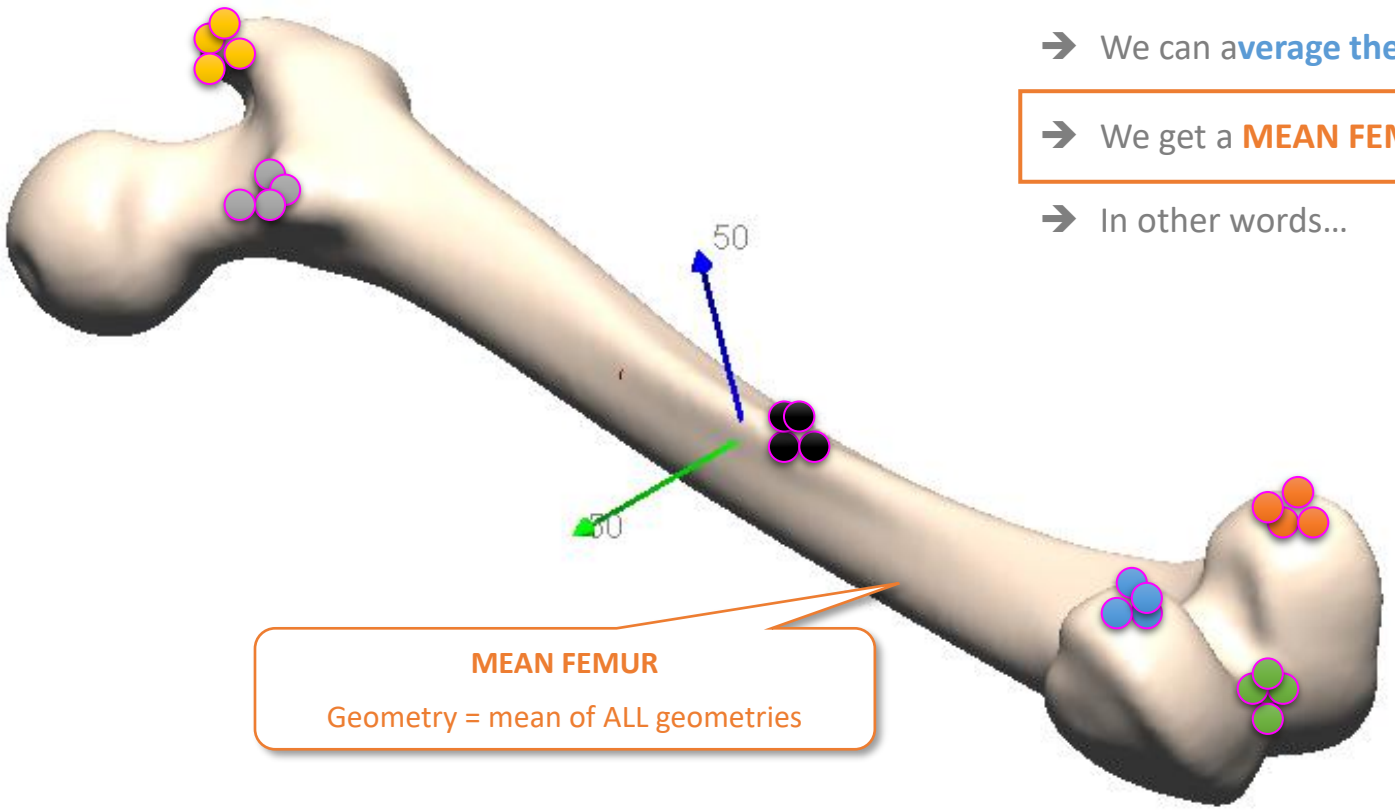
Vertex **17100** = tip of the **Great Trochanter**

… you **don’t need to KNOW EXACTLY** what each one (of the 18,291 vertices) means ☺

Marek BUCKI – marek.bucki@univ-grenoble-alpes.fr

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Statistical **Shape Modeling** using **PCA**



MEAN FEMUR

Geometry = mean of ALL geometries

Since **ALL VERTICES** define the **SAME** anatomical points

➔ We can a**verage their positions** (it does make sense!)

➔ We get a **MEAN FEMORAL SHAPE** (we're already doing shape statistics at this point ☺)

➔ In other words…

$$\forall v \in [0\,..\,18290],$$

$$[P_v^X]_{\text{mean}} = \frac{1}{24}\sum_{f=1}^{24}[P_v^X]_f$$

$$[P_v^Y]_{\text{mean}} = \frac{1}{24}\sum_{f=1}^{24}[P_v^Y]_f$$

$$[P_v^Z]_{\text{mean}} = \frac{1}{24}\sum_{f=1}^{24}[P_v^Z]_f$$

We don't care about the normals …

Your favorite 3D rendering lib (OpenGL, Three.js, …) can easily recompute them for you

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Let's clarify a few key concepts …

- **Discrete**: coin toss, dice, cards, …
- **Continuous**: height of a person, room temperature, …

- **Random Variable (RV)** = **Function**: input = { space of "experiments" } ; output = { random stuff }
  *e.g., **Coin toss**: { A coin in the air… } → { Heads or Tails }*

- **Discrete RVs**: binomial, uniform, … (anything countable)
- **Continuous RVs**: Normal, Poisson, uniform, …

- **Probability density function (PDF) of a RV** = Defines how a RV behaves: { probabilities for each possible outcome }

- **PDF parameters** = "Moments of the RV" = { mean, standard deviation, … } → Entirely define the behavior of the RV

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

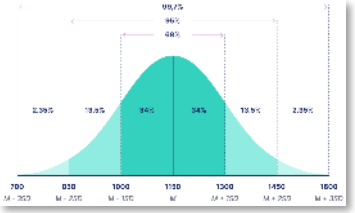
PROBABILITIES – *Theoretical World*

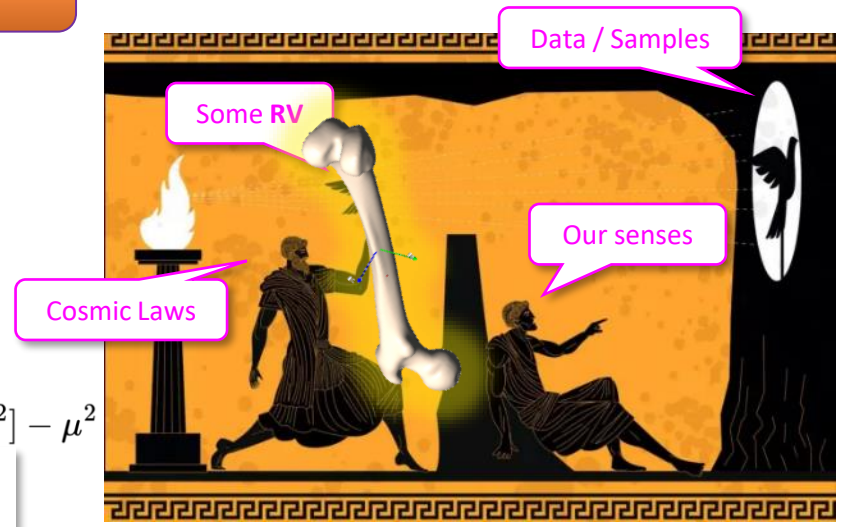
***Inaccessible, True** parameters*

$$E[X] = \mu = \int_{-\infty}^{+\infty} x \times f(x)dx$$

$$E[X^2] = \int_{-\infty}^{+\infty} x^2 \times f(x)dx$$

$$\sigma^2[X] = E[(X - E[X])^2] = E[X^2] - (E[X])^2 = E[X^2] - \mu^2$$





Plato's allegory of the cave

STATISTICS – *Practical World*

***Parameters Estimated** through experiments*

Unbiased estimators

$$\hat{E}[X] = \hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\hat{\sigma}^2[X] = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \hat{\mu})^2$$

$$\hat{\sigma}[X] = \sqrt{\frac{1}{N-1.5}\sum_{i=1}^{N}(x_i - \hat{\mu})^2}$$

Lim N → +∞ = **true value**

Marek BUCKI – marek.bucki@univ-grenoble-alpes.fr

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Statistical **Shape Modeling** using **PCA**

Beyond the mean shape → study of **VARIATIONS**: we need **some stats** …

Definitions

- 1 subject (femur) = **1 REALIZATION** / **SAMPLE** of a **RANDOM VARIABLE** living in a "morphological space"
- 1 realization = 1 mesh = (1 set of **18,291** 3D points) = (1 set of **3 x 18,291** coordinates) = (1 **VECTOR** in dimension **54,873**)
- We are working in a **HIGH DIMENSIONAL** (D=54,873) space … using **VERY FEW** (N=24) samples!
- We will "pack" the coordinates X, Y, and Z and use a **single vector representation** of our 24 femur shapes

→ All these 50k values do not change **AT RANDOM** in this high dim space …
… there is a **STRUCTURE** that **maintains shape consistency across all 55k dimensions!**

$$[F]_f = \begin{pmatrix} u_f^0 \\ u_f^1 \\ u_f^2 \\ .. \\ .. \\ u_f^{54872} \end{pmatrix}$$

Big questions

- *How can we describe variability in a **HIGH DIM SPACE** using only a **FEW PARAMETERS**?*
- *How do we discover this **STRUCTURE**?*

Before studying variations in D=58,873 … **let's see how PCA works in 2D**

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Statistical **Shape Modeling** using **PCA** … **in 2D**

Let's consider N samples ("femurs") in 2D

$$\forall f \in [1\,..\,N], [F]_f = \begin{pmatrix} u_f^0 \\ u_f^1 \end{pmatrix} = \begin{pmatrix} u_f \\ v_f \end{pmatrix}$$

First, let's get rid of the constant bias, by centering our random vars U and V

$$[F']_f = \begin{pmatrix} u'_f \\ v'_f \end{pmatrix}$$

$$u'_f = u_f - \frac{1}{N}\sum_{g=1}^{N} u_g$$

$$v'_f = v_f - \frac{1}{N}\sum_{g=1}^{N} v_g$$

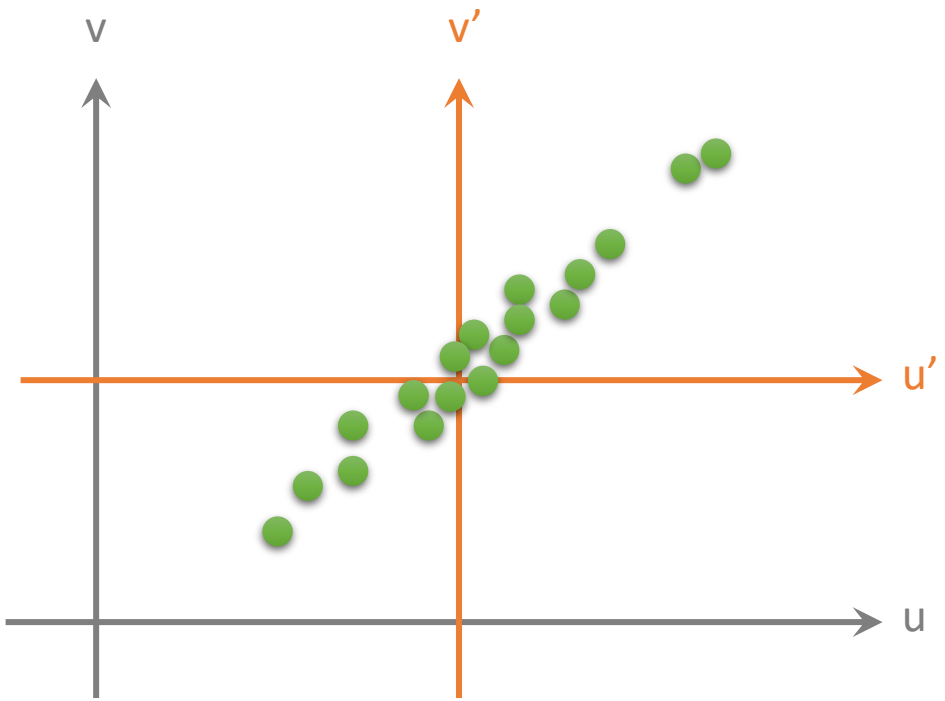
The **U'** and **V'** random variables are now **CENTERED** → They have a **0 mean**

E(u') = E(v') = 0

$$\boxed{\begin{aligned} u'_f &= u_f - \bar{u} \\ v'_f &= v_f - \bar{v} \end{aligned}}$$

Our first referential change

One more to come…



Statistical / AI Shape Modeling for Computer-Assisted Medicine

Statistical **Shape Modeling** using **PCA … in 2D**

Clearly, we can se that IN THIS NEW SPACE, our "high dim" (2D…) **samples** can **MOSTLY** be described by a **single 1D parameter λ** … the **REMAINING DETAILS** are captured by a smaller parameter **μ**

Visually it is obvious that **if we know u'** then we can **almost certainly predict v'**

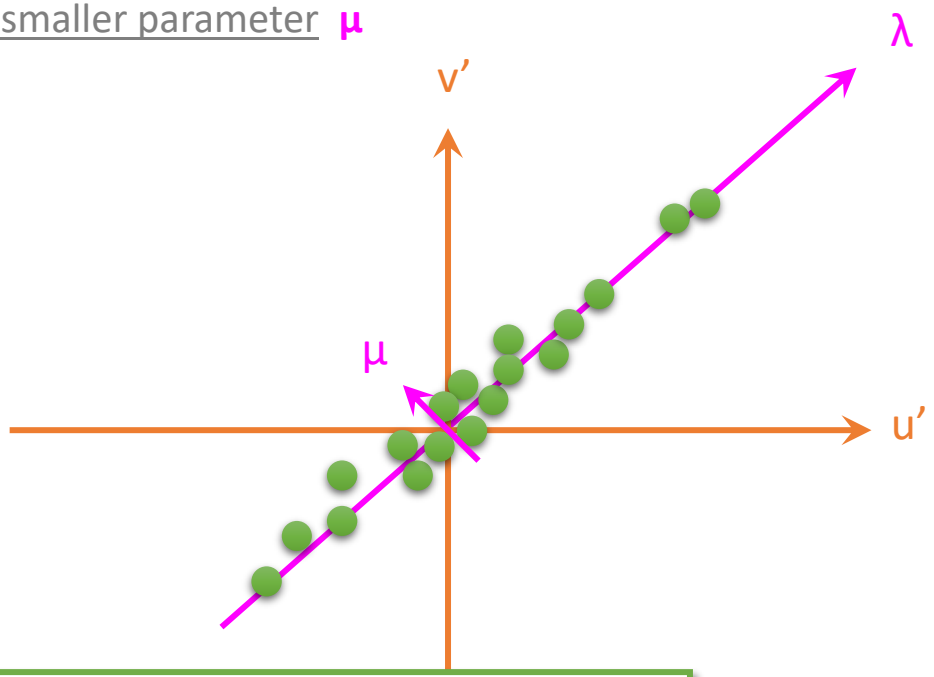
→ **v' ≈ K x u'**

So, the **1D parametrization** = { **u' = λ; v' ≈ K x λ** }

This simple relation is possible because u' and v' are **HIGHLY CORRELATED**

→ Knowing **ONE** gives you a **LOT** of information about the **OTHER**

u' and v' **VARY TOGETHER**, they thus have a **HIGH COVARIANCE**



In **HIGH DIM** problems (our femurs) we can identify **GROUPS OF VARIABLES** that **VARY TOGETHER** and **LUMP** them into **ONE PARAMETER** … reducing the **ORDER OF THE MODEL** … from D≈**55,000** to D≈**10**

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Statistical **Shape Modeling** using **PCA … in 2D**

Highly correlated values can be automatically identified using the **(CROSS-) COVARIANCE MATRIX** of the dataset

The entries of the matrix are **COVARIANCES** of all possible **2-value combinations**:

- **Cov**(U' ; U') = **Var**(U') = **σ²**(U')
- **Cov**(U' ; V') = **Cov**(V' ; U')
- **Cov**(V' ; V') = **Var**(V') = **σ²**(V')

About **VARIANCE** and **COVARIANCE**:

- **VARIANCE** = a number measuring how much a RV varies **around its mean**
  This is true because E(U') = E(V') = 0 (our RVs are CENTERED)

$$\begin{cases} Var(U') = \boxed{E[(U' - E(U'))^2] = E[U'^2]} \\ Var(V') = E[(V' - E(V'))^2] = E[V'^2] \end{cases}$$

This definition **MAKES SENSE!**

This one is **LESS INTUITIVE…**

- **COVARIANCE** = a number measuring how much **<u>2 RVs</u> vary together**

$$\begin{cases} Cov(U';V') = \boxed{E[(U' - E(U'))(V' - E(V'))] = E[U'V']} \\ Cov(V';U') = Cov(U';V') \\ Cov(U';U') = E[(U' - E(U'))^2] = Var(U') \\ Cov(V';V') = E[(V' - E(V'))^2] = Var(V') \end{cases}$$

- **ESTIMATORS** =

$$\widehat{Var}(U') = \frac{1}{N-1}\sum_{i=1}^{N} u_i'^2$$

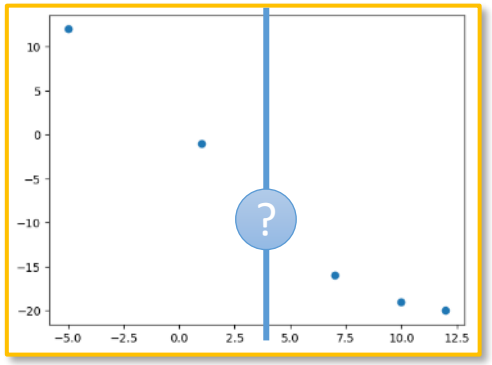
$$\widehat{Cov}(U';V') = \frac{1}{N-1}\sum_{i=1}^{N} u_i' \times v_i'$$

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Statistical **Shape Modeling** using **PCA** … **in 2D**

Assuming **X** and **Y** are **ALREADY CENTERED**

Let's **develop an intuition** about the **COVARIANCE** and its **ESTIMATOR** … using **X** and **Y**, as our RVs … (for clarity)

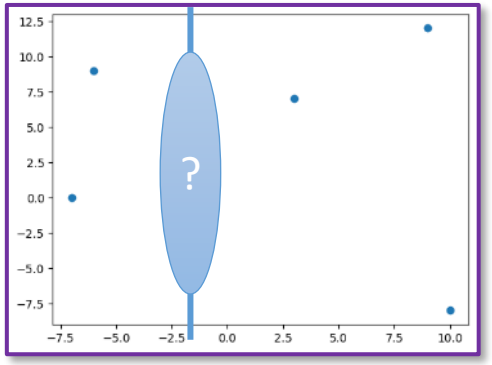


Example 1

Samples = { (x1=+10, y1=-19), (x2=-5, y2=+12), (x3=1, y3=-1), (x4=+7, y4=-16), (x5=+12, y5=-20), **(x6=+3, y6= …?)** }

Something's going on here… **pattern**: **Y ≈ -2 X**

Cov(X, Y) = 1/(5 - 1) * (10*(-19) + (-5)*12 + 1*(-1) + 7*(-16) + 12*(-20)) = **-150.75**



Example 2

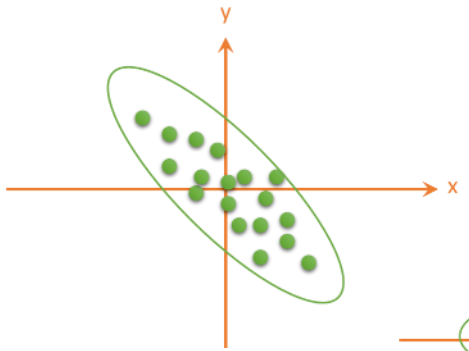
Samples = { (x1=+10, y1=-8), (x2=+9, y2=+12), (x3=-7, y3=0), (x4=-6, y4=9), (x5=+3, y5=+7), **(x6=-2, y6= …?)** }

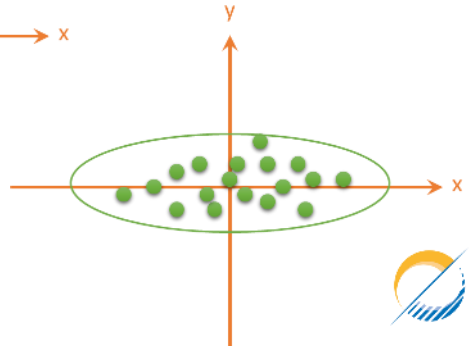
NO CLEAR **pattern**… **knowing X doesn't tell us much about Y**…

Cov(X, Y) = 1/(5 - 1) * (10*(-8) + 9*12 + (-7)*0+ (-6)*9 + 3*7) = **-1.25**

Cov(X, Y) = 1/(N - 1) * (10*[(-8) + 3 + (-2) + 5 + …] + (-6)*[……] + …) = **~ 0**

~ 0 ~ 0



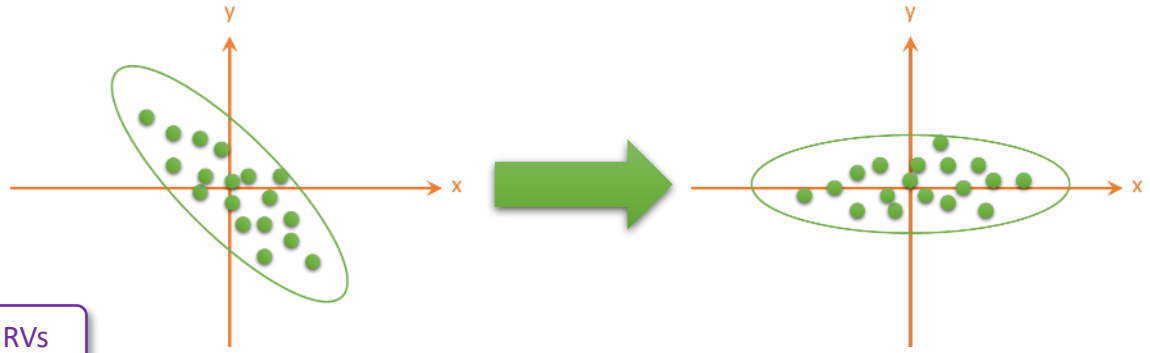


Example 1: **X** and **Y** carry **redundant information** → Wasteful description of reality

Example 2: **X** and **Y** seem "unrelated", they are **independent** → Efficient modeling

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Statistical **Shape Modeling** using **PCA** … **in 2D**



We want to **reduce redundancy** in our dataset, without losing data

Go from the situation in **Example 1** to the situation in **Example 2**

"**Quintessential**" RVs

In other words, go from **DEPENDENT RVs** to **INDEPENDENT RVs** … describing the same **REALITY** but in a **DIFFERENT FRAME**

Problem statement

- We want to convert **X** and **Y** to some new variables **X'** and **Y'** = **LINEAR COMBINATIONS** OF **X** and **Y**
- We want **X'** and **Y'** to be **INDEPENDENT**, i.e., Cov(**X'**; **Y'**) = Cov(**Y'**; **X'**) = 0

$$COV(X;Y) = \begin{pmatrix} Cov(X;X) = Var(X) & Cov(X;Y) \\ Cov(Y;X) & Cov(Y;Y) = Var(Y)) \end{pmatrix} \Rightarrow COV(X';Y') = \begin{pmatrix} Cov(X';X') = Var(X') & 0 \\ 0 & Cov(Y';Y') = Var(Y')) \end{pmatrix}$$

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Statistical **Shape Modeling** using **PCA** … **in 2D**

Let's make sur we can **ACTUALLY DIAGONALIZE** **COV(X;Y)** … and that this operation **solves our problem**

Data / samples matrix

$$COV(X;Y) = \frac{1}{N-1}\begin{pmatrix} \sum_{i=1}^{N} x_i^2 & \sum_{i=1}^{N} x_i y_i \\ \sum_{i=1}^{N} x_i y_i & \sum_{i=1}^{N} y_i^2 \end{pmatrix} = \frac{1}{N-1}\underbrace{\boxed{\begin{pmatrix} x_1 & x_2 & x_3 & .. & x_N \\ y_1 & y_2 & y_3 & .. & y_N \end{pmatrix}}}_{D} \times \underbrace{\boxed{\begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ .. & .. \\ x_N & y_N \end{pmatrix}}}_{D^T} = \frac{1}{N-1} D \times D^T$$

COV(X;Y) is obviously **SYMMETRICAL**

COV(X;Y) is also **POSITIVE DEFINITE** → $\forall Z \in \mathbb{R}^2$ $Z^T \times COV(X;Y) \times Z = \frac{1}{N-1}\{Z^T \times D \times D^T \times Z\} = \frac{1}{N-1}\{(D^T \times Z)^T \times (D^T \times Z)\} = \frac{1}{N-1}||D^T \times Z||^2 \geq 0$

→ **COV(X;Y)** can be **DIAGONALIZED**

$$\exists R \in \mathbb{R}^{2\times 2}, R^{-1} = R^T$$

$$COV(X;Y) = \boxed{R \times \Lambda \times R^T} = \boxed{R} \times \boxed{\begin{pmatrix} \lambda & 0 \\ 0 & \mu \end{pmatrix}} \times \boxed{R^T}$$

R is a rotation matrix
= **BASIS CHANGE** matrix

What is this matrix?

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Statistical **Shape Modeling** using **PCA** … **in 2D**

Could it be that **Λ = COV(X';Y')**? … If yes, then what are **X'** and **Y'**?

$$COV(X;Y) = R \times \Lambda \times R^T$$

$$\Lambda = R^T \times COV(X;Y) \times R \quad \Rightarrow \quad \Lambda = \frac{1}{N-1}\{R^T \times D \times D^T \times R\} \quad \Rightarrow \quad \Lambda = \frac{1}{N-1}\{(R^T \times D) \times (R^T \times D)^T\}$$

$$\boxed{\Lambda = \frac{1}{N-1} D' \times D'^{\,T}}$$

What is **D'**?

If we define **X'** and **Y'** as a linear combination of **X** and **Y** …

$$\begin{pmatrix} X' \\ Y' \end{pmatrix} = R^T \times \begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} r_{11} & r_{21} \\ r_{12} & r_{22} \end{pmatrix} \times \begin{pmatrix} X \\ Y \end{pmatrix}$$

Then **D'** is the **DATA MATRIX** associated to **X'** and **Y'** … and **Λ = COV(X';Y')**

$$D' = R^T \times D = R^T \times \begin{pmatrix} x_1 & x_2 & x_3 & .. & x_N \\ y_1 & y_2 & y_3 & .. & y_N \end{pmatrix} = \begin{pmatrix} r_{11} & r_{21} \\ r_{12} & r_{22} \end{pmatrix} \times \begin{pmatrix} x_1 & x_2 & x_3 & .. & x_N \\ y_1 & y_2 & y_3 & .. & y_N \end{pmatrix} = \begin{pmatrix} x'_1 & x'_2 & x'_3 & .. & x'_N \\ y'_1 & y'_2 & y'_3 & .. & y'_N \end{pmatrix}$$

Statistical / AI Shape Modeling for Computer-Assisted Medicine

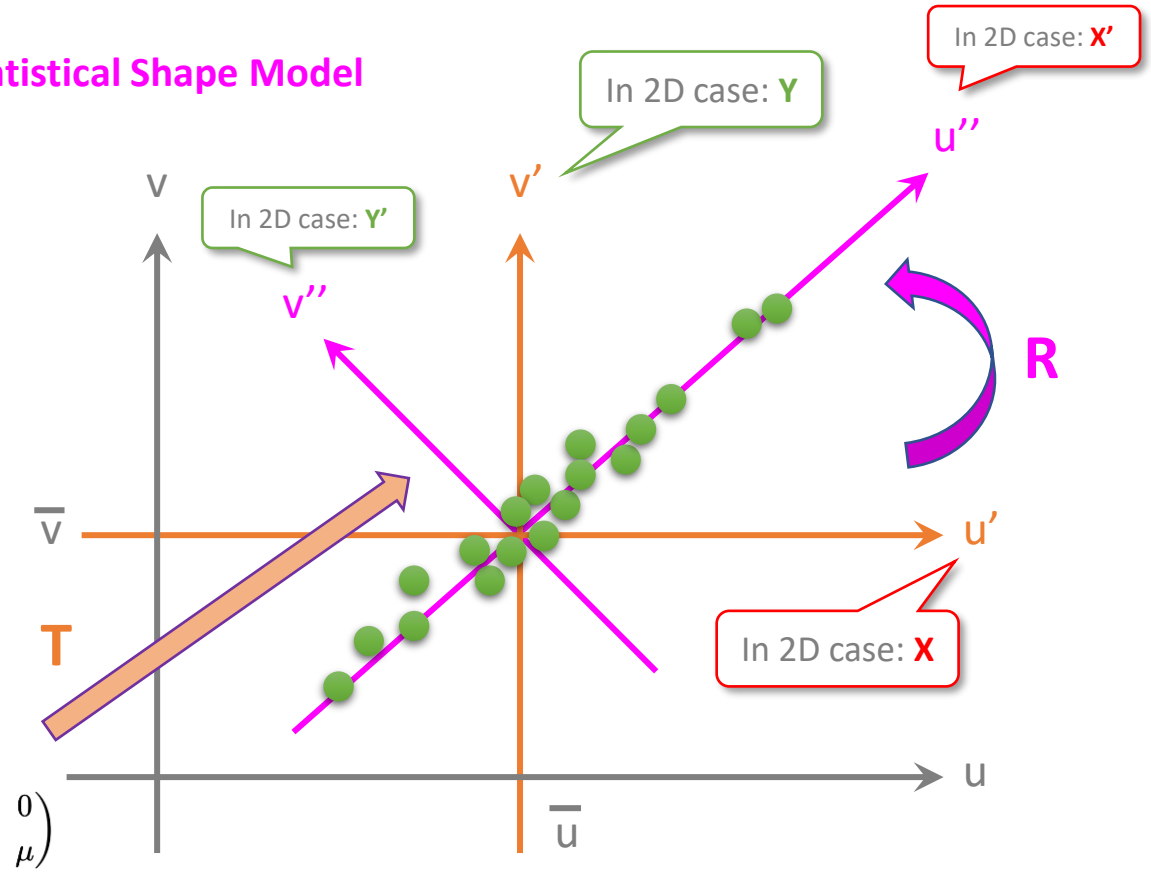
Statistical **Shape Modeling** using **PCA** … **in 2D**

Finally, let's analyze the **meaning of the Eigenvalues** … and build our first **2D Statistical Shape Model**

First $\forall f \in [1\,..\,N], [\mathbf{F}]_f = \begin{pmatrix} u_f^0 \\ u_f^1 \end{pmatrix} = \begin{pmatrix} u_f \\ v_f \end{pmatrix}$

Then $[\mathbf{F}']_f = \begin{pmatrix} u'_f \\ v'_f \end{pmatrix}$ where $\boxed{\begin{aligned} u'_f &= u_f - \bar{u} \\ v'_f &= v_f - \bar{v} \end{aligned}}$ Translation **T**

And finally $[\mathbf{F}'']_f = \boxed{R^T} \times [\mathbf{F}']_f$ Rotation **R**



Statistical Shape Model

1 – In the **eigen referential**, the **original samples** are described by **INDEPENDENT RVs**: **u''** and **v''** acting along the column **eigenvectors** from **R**

2 – The **variances** of **u''** and **v''** are the **eigenvalues λ, μ** of **Λ=COV(u''; v'')**: $\Lambda = \begin{pmatrix} \lambda & 0 \\ 0 & \mu \end{pmatrix}$

3 – **SSM**: **New realistic shapes** can produced by **drawing realizations of F''** within reasonable **standard deviations** and then working backwards to the **original referential**:

$$[\mathbf{F}]_f = R \times [\mathbf{F}'']_f + \bar{\mathbf{F}}$$

Assuming **Normal** distribution

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Example of a sample generation … in 2D

Let's assume that $[\bar{\mathbf{F}}] = \begin{pmatrix} \bar{u} \\ \bar{v} \end{pmatrix} = \begin{pmatrix} 2.3 \\ -9.8 \end{pmatrix}$.

The centered samples are: $[\mathbf{F}']_f = \begin{pmatrix} u'_f \\ v'_f \end{pmatrix} = \begin{pmatrix} u_f - \bar{u} \\ v_f - \bar{v} \end{pmatrix}$.

The corresponding random vector $[\mathbf{F}']$ and random coordinates u' and v' are:

$$[\mathbf{F}'] = \begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} u - \bar{u} \\ v - \bar{v} \end{pmatrix}$$

Now, let's assume that:

$$COV(u'; v') = \begin{pmatrix} 1.5 & 0.4 \\ 0.4 & 0.8 \end{pmatrix}$$

If we diagonalize this matrix with numpy:

```
COV_up_vp = np.array([[1.5, 0.4], [0.4, 0.8]])
lambda_mu, R = np.linalg.eig(COV_up_vp)
```

we get the following:

$$\Lambda = \begin{pmatrix} \lambda = 1.68150729 & 0 \\ 0 & \mu = 0.61849271 \end{pmatrix}$$

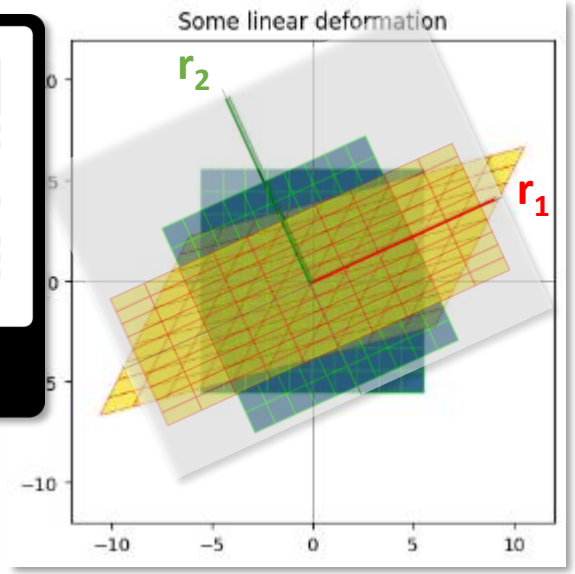
and:

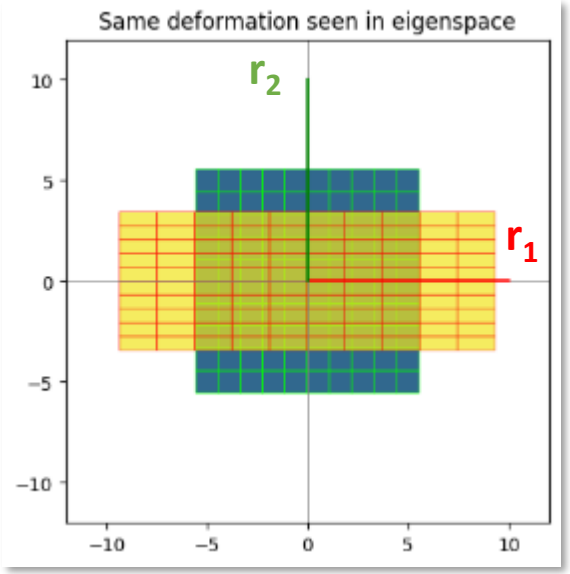
$$R = \begin{pmatrix} \boxed{\begin{matrix} 0.91063291 \\ 0.41321628 \end{matrix}} & \boxed{\begin{matrix} -0.41321628 \\ 0.91063291 \end{matrix}} \end{pmatrix} = \begin{pmatrix} \vec{\mathbf{r}_1} & \vec{\mathbf{r}_2} \end{pmatrix}$$

such that:

$$COV(u'; v') = R \times \Lambda \times R^T$$

SCAN ME





Let's define the "eigen random coordinates" u'' and v'' as:

$$\begin{pmatrix} u'' \\ v'' \end{pmatrix} = R^T \times \begin{pmatrix} u' \\ v' \end{pmatrix}$$

u'' and v'' are **CENTERED** and **INDEPENDENT**

In the eigen referential $(0, \vec{\mathbf{r}_1}, \vec{\mathbf{r}_2})$, the random variables u'' and v'' are normally distributed, and $u'' \sim N(0, \lambda)$ and $v'' \sim N(0, \mu)$.

Let's draw realizations of u'' and v'' using numpy's random engine:

```
rng = np.random.default_rng()
lambda_0 = rng.normal( 0, np.sqrt(1.68150729) )  # use standard deviation, not variance!
mu_0 = rng.normal( 0, np.sqrt(0.61849271) )      # use standard deviation, not variance!
```

Then, the corresponding realization in the eigen referential $(0, \vec{\mathbf{r}_1}, \vec{\mathbf{r}_2})$ is $[\mathbf{F}'']_0 = \lambda_0 \times \vec{\mathbf{r}_1} + \mu_0 \times \vec{\mathbf{r}_2}$.

The $[\mathbf{F}'']_0$ vector cannot be "unpacked" into a mesh. We need to convert it back to the original "mesh" frame:

$$[\mathbf{F}]_0 = R \times [\mathbf{F}'']_0 + [\bar{\mathbf{F}}] = \begin{pmatrix} 0.91063291 & -0.41321628 \\ 0.41321628 & 0.91063291 \end{pmatrix} \times \begin{pmatrix} \lambda_0 \times 0.91063291 + \mu_0 \times -0.41321628 \\ \lambda_0 \times 0.41321628 + \mu_0 \times 0.91063291 \end{pmatrix} + \begin{pmatrix} 2.3 \\ -9.8 \end{pmatrix}$$

The resulting vector can be "unpacked" into a mesh representing our new shape.

UGA Université Grenoble Alpes
vinsight
t-medical.com

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Statistical **Shape Modeling** using **PCA** in a nutshell

Let's generalize the procedure from **dimension D=2** … to **any dimension D**

Numerical Recipe for a Linear PCA-based Statistical Shape Model

- **F** is a random vector in $\mathbf{R}^D$
- **F** is a formed by D scalar random variables in **R**: $F^1, F^2, \ldots, F^D$

1) Collect N samples of **F**
2) Center the samples on the estimated mean: $\mathbf{F' = F - \overline{F}}$
3) Compute the DxD covariance matrix **COV(F'; F')**
4) Diagonalize this matrix using the DxD rotation matrix **R**
5) Define the new random vector $\mathbf{F'' = R^T \times F'}$
6) Use the **eigenvectors** (columns of **R**) and **variances** (diagonal elements of **COV(F''; F'')**) to produce **M statistically realistic samples of F''**: $[\mathbf{F''}]_1, [\mathbf{F''}]_2, \ldots, [\mathbf{F''}]_M$
7) Go back to the **original referential**, where the numerical values actually **form a "shape"**:

$$[\mathbf{F}]_f = R \times [\mathbf{F''}]_f + \overline{\mathbf{F}}$$

$$[\mathbf{F}] = \begin{pmatrix} F^1 \\ F^2 \\ F^3 \\ .. \\ .. \\ F^D \end{pmatrix} \qquad [\mathbf{F}]_f = \begin{pmatrix} [F^1]_f \\ [F^2]_f \\ [F^3]_f \\ .. \\ .. \\ [F^D]_f \end{pmatrix} = \begin{pmatrix} u_f^1 \\ u_f^2 \\ u_f^3 \\ .. \\ .. \\ u_f^D \end{pmatrix} \qquad [\overline{\mathbf{F}}] = \begin{pmatrix} \overline{u^1} \\ \overline{u^2} \\ \overline{u^3} \\ .. \\ .. \\ \overline{u^D} \end{pmatrix}$$

$$[\mathbf{F'}] = \begin{pmatrix} F'^1 \\ F'^2 \\ F'^3 \\ .. \\ .. \\ F'^D \end{pmatrix} = \begin{pmatrix} F^1 - \bar{F}^1 \\ F^2 - \bar{F}^2 \\ F^3 - \bar{F}^3 \\ .. \\ .. \\ F^D - \bar{F}^D \end{pmatrix} \qquad [\mathbf{F'}]_f = \begin{pmatrix} [F'^1]_f \\ [F'^2]_f \\ [F'^3]_f \\ .. \\ .. \\ [F'^D]_f \end{pmatrix} = \begin{pmatrix} u_f'^1 \\ u_f'^2 \\ u_f'^3 \\ .. \\ .. \\ u_f'^D \end{pmatrix} = \begin{pmatrix} u_f^1 - \overline{u^1} \\ u_f^2 - \overline{u^2} \\ u_f^3 - \overline{u^3} \\ .. \\ .. \\ u_f^D - \overline{u^D} \end{pmatrix}$$

$$[\mathbf{F'}]_f = [\mathbf{F}]_f - [\overline{\mathbf{F}}]$$

$$COV(\mathbf{F'};\mathbf{F'}) = \begin{pmatrix} Var(F'^1) & Cov(F'^1;F'^2) & Cov(F'^1;F'^3) & .. & Cov(F'^1;F'^D) \\ Cov(F'^2;F'^1) & Var(F'^2) & Cov(F'^2;F'^3) & .. & Cov(F'^2;F'^D) \\ Cov(F'^3;F'^1) & Cov(F'^3;F'^2) & Var(F'^3) & .. & Cov(F'^3;F'^D) \\ .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \\ Cov(F'^D;F'^1) & Cov(F'^D;F'^2) & Cov(F'^D;F'^3) & .. & Var(F'^D) \end{pmatrix}$$

$$\widehat{Var}(F'^i) = \frac{1}{N-1}\sum_{f=1}^{N}(u_f'^i)^2 = \frac{1}{N-1}\sum_{f=1}^{N}(u_f^i - \bar{u}^i)^2$$

$$\widehat{Cov}(F'^i;F'^j) = \frac{1}{N-1}\sum_{f=1}^{N} u_f'^i \times u_f'^j = \frac{1}{N-1}\sum_{f=1}^{N}(u_f^i - \bar{u}^i)(u_f^j - \bar{u}^j)$$

$$\widehat{Var}(F'^i) = \widehat{Cov}(F'^i;F'^i)$$

Marek BUCKI – marek.bucki@univ-grenoble-alpes.fr

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Example of a sample generation in any dimension D

Let's say we want to use the **first 3 modes**, with respective variances: **120**, **56**, and **9**

a) Assuming normal distributions → $F''^1 \sim \boldsymbol{N}(\mu = \mathbf{0}, \sigma^2 = \mathbf{120})$, $F''^2 \sim \boldsymbol{N}(\mu = \mathbf{0}, \sigma^2 = \mathbf{56})$, and $F''^3 \sim \boldsymbol{N}(\mu = \mathbf{0}, \sigma^2 = \mathbf{9})$

b) Using a random engine, we get the (scalar) samples $[F''^1]$, $[F''^2]$ and $[F''^3]$

1st column of **R** | 2nd column | 3rd column

c) We assemble the sample vector $[\mathbf{F''}]$ in the eigen-frame as: $[\mathbf{F''}] = [F''^1] \times [\mathbf{R}_{:1}] + [F''^2] \times [\mathbf{R}_{:2}] + [F''^3] \times [\mathbf{R}_{:3}]$

d) Finally we convert the sample in eigen-frame (meaningless as a "mesh") into the original frame (where "meshes" are encoded): $[\mathbf{F}] = \mathbf{R} \times [\mathbf{F''}] + [\bar{\mathbf{F}}]$

e) We **unpack** the vector $[\mathbf{F}]$ as (X, Y, Z) vertices coordinates and **replace the point coordinates** in the **mesh template** (shared topology)

f) Done!

Statistical / AI Shape Modeling for Computer-Assisted Medicine

Going back to our **N=24** femurs in **D=54,873** …

Let's consider the **SHAPES** of the matrices … and the **NUMERICAL METHODS** needed

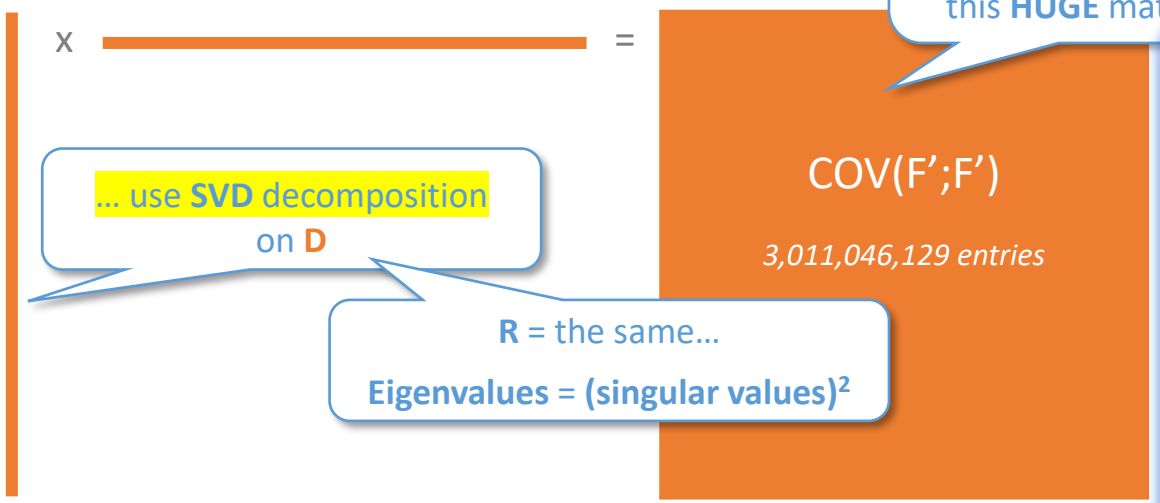
In the case of our **FEMURS** …

D is **54,873 x 24** … **COV(F'; F')**, **R** and **COV(F''; F'')** are **54,873 x 54,873**

$$\begin{pmatrix} x_1 & x_2 & x_3 & .. & x_N \\ y_1 & y_2 & y_3 & .. & y_N \end{pmatrix} \times \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ .. & .. \\ x_N & y_N \end{pmatrix}$$

D **D^T**

Visually … D x D^T =



Life hack

Do **NOT** Eigen-Jacobi this **HUGE** matrix …

… use **SVD** decomposition on **D**

R = the same…

Eigenvalues = **(singular values)^2**

FINAL COMMENTS

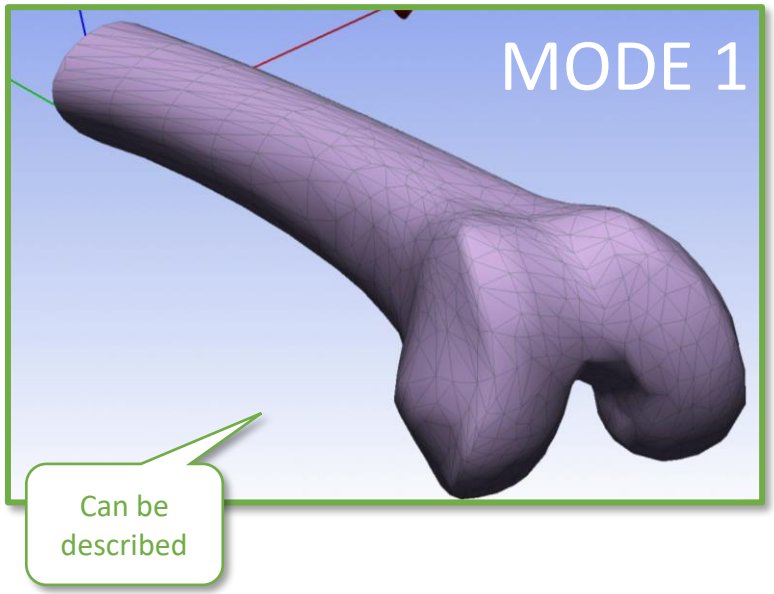
- The rank of **COV(F'; F')** is **< N=24** … **why?**
- There are **only 24 non-0 eigenvalues**… out of 54,873…
- SVD should sort them in **decreasing order**:
  - 1st ones are the MOST MEANINGFUL → Largest variance
  - Last ones are the LEAST MEANINGFUL → Noise…
- Each **eigenvalue** defines a **DEFORMATION MODE**
- Now our samples live in **D=24** … and not **D=54,873**

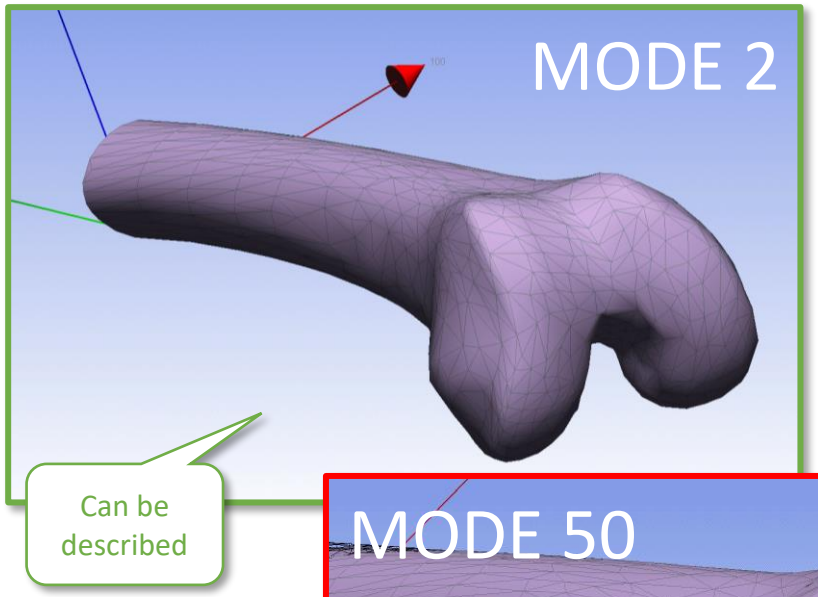
ROM

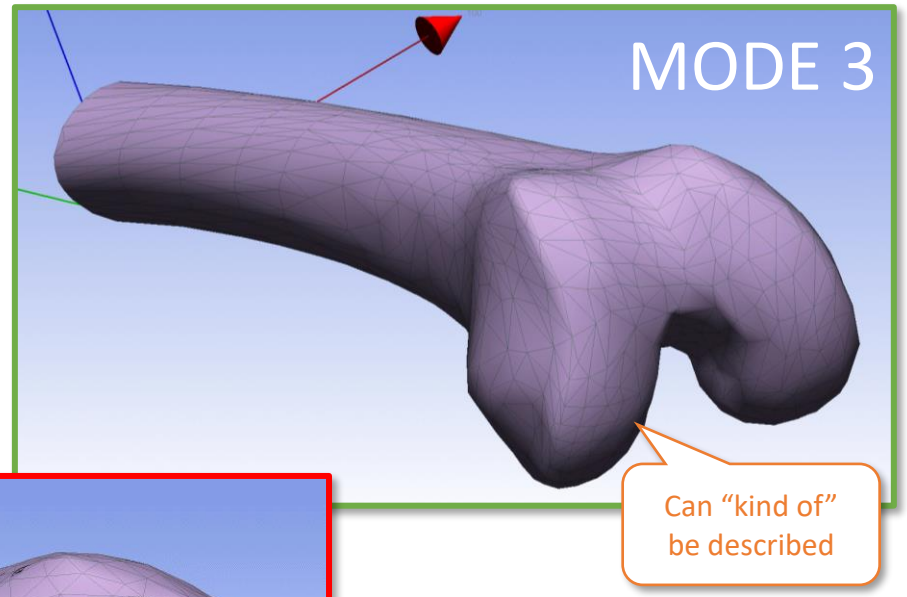
Reduced Order Modeling

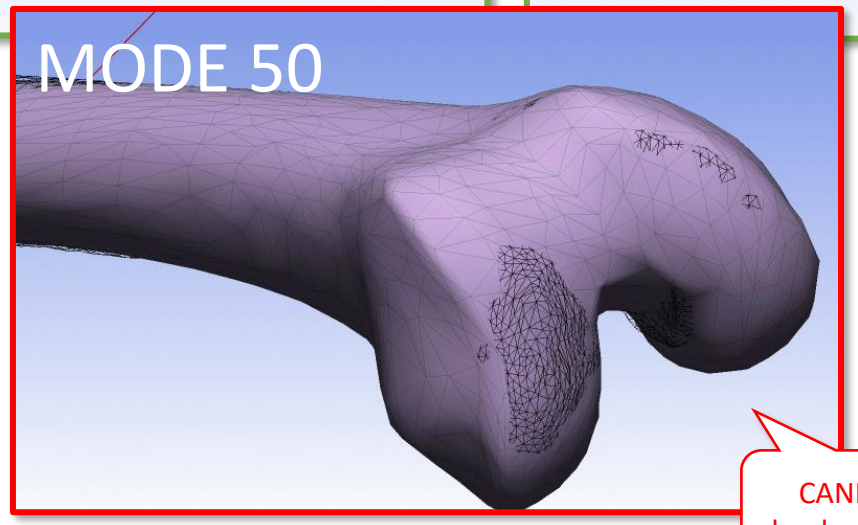
Statistical / AI Shape Modeling for Computer-Assisted Medicine

Results (on a different dataset)









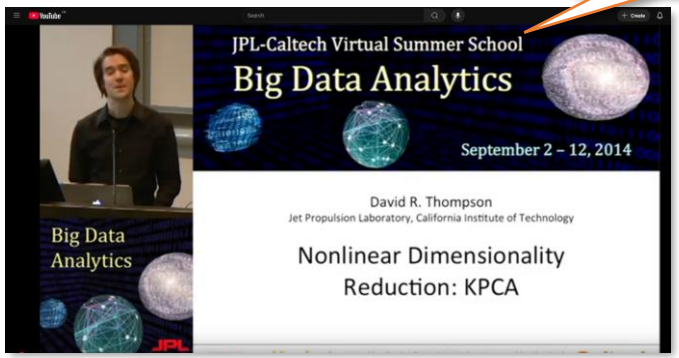
Statistical / AI Shape Modeling for Computer-Assisted Medicine

YouTube

Data Challenge – **Resources**

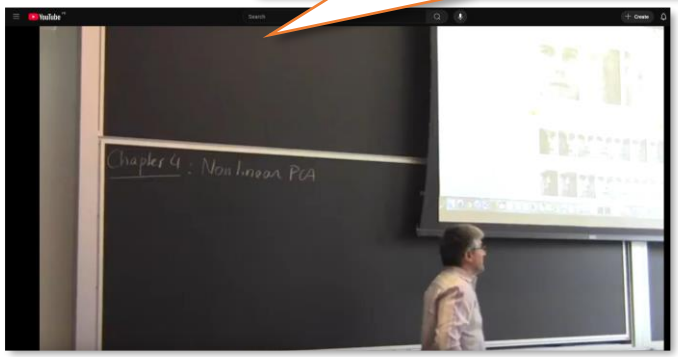
kPCA →

Caltech / JPL



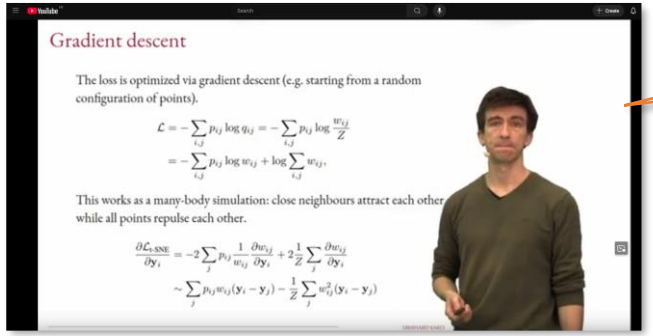
https://www.youtube.com/watch?v=HbDHohXPLnU

Johns Hopkins Biomedical Engineering



https://www.youtube.com/watch?v=-zTW0ZfQDTI

Manifold learning → …



Tübingen Machine Learning

https://www.youtube.com/watch?v=MnRskV3NY1k

24 femurs

Autoencoders → *Seminar to come!*

SCAN ME

SCAN ME

SCAN ME

SCAN ME

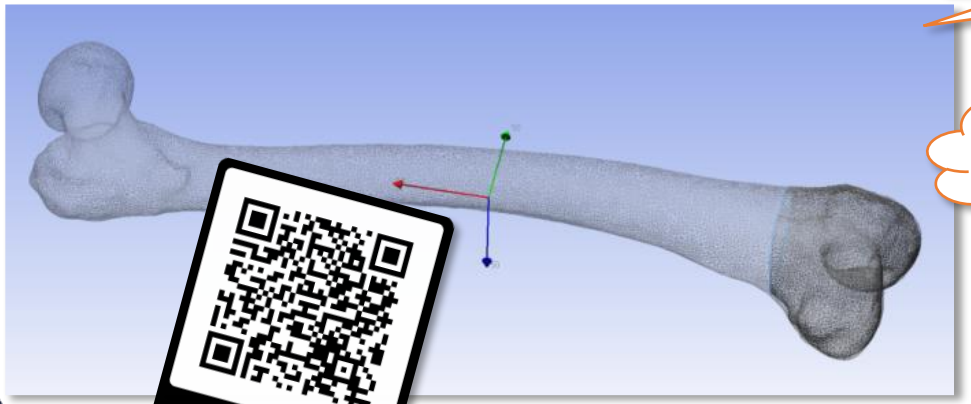
Statistical / AI Shape Modeling for Computer-Assisted Medicine

Enjoy your projects! … There are tons of ideas to explore ☺

Contact = marek.bucki@univ-grenoble-alpes.fr

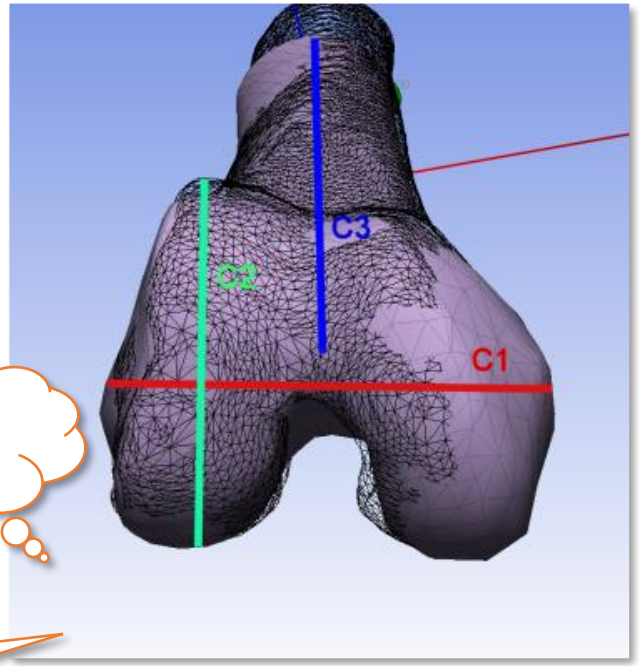
Interesting side quests…

How to reconstruct a **REALISTIC FULL BONE** from a **PARTIAL SHAPE**?



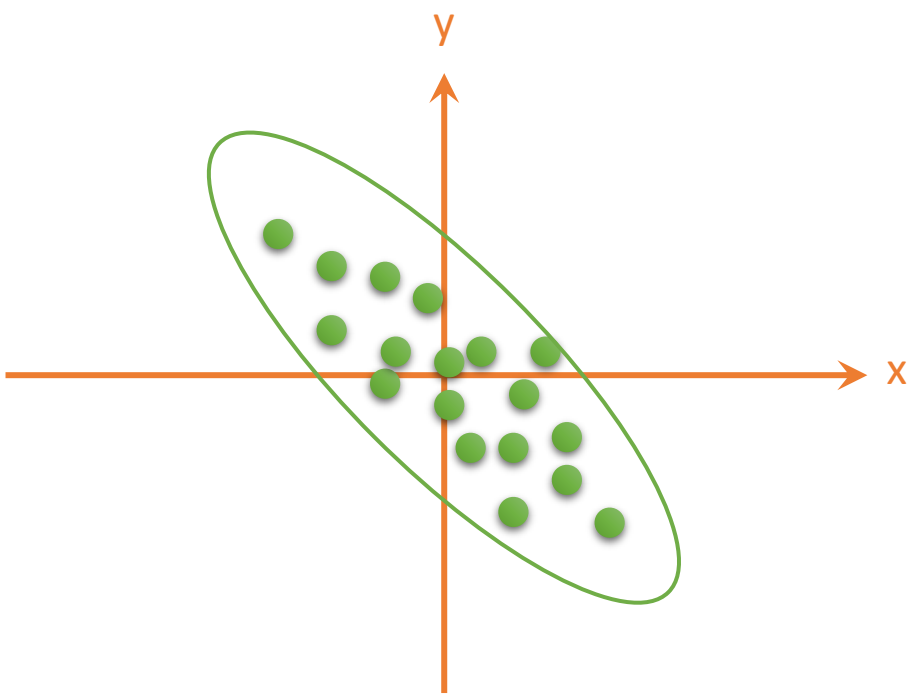
How to "align" meshes?

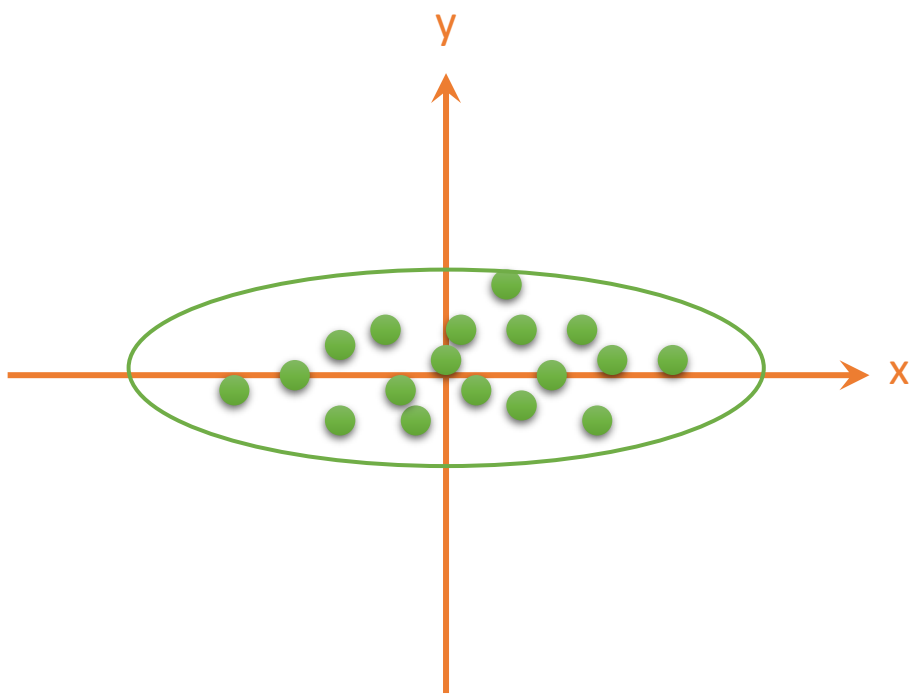
Is an SSM "objective"?



How to reconstruct a **REALISTIC BONE** from a set of **SCALAR MEASUREMENTS / CONSTRAINTS** ?

SCAN ME

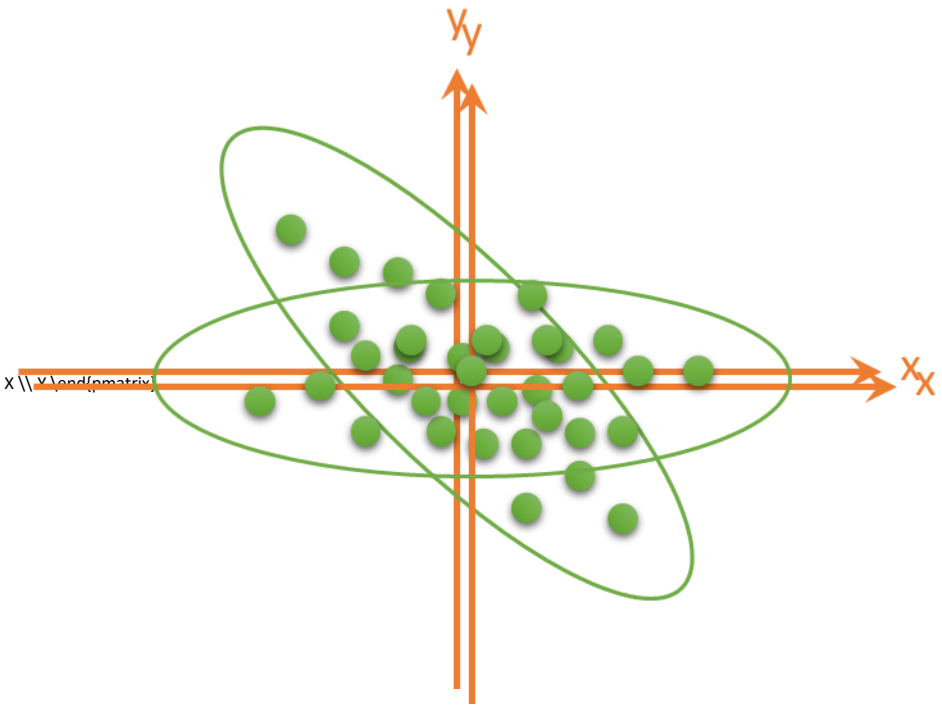

y
x


y
x

```
\newcommand{\isep}{\mathrel{{.}\,{.}}}

\forall f \in [1 \isep N], [\mathbf{F}]_f = \begin{pmatrix}
 u_f^0 \\
 u_f^1 \\
\end{pmatrix} =
\begin{pmatrix}
 u_f \\
 v_f \\
\end{pmatrix}
\\
\\
[\mathbf{F'}]_f = \begin{pmatrix}
 u'_f \\
 v'_f \\
\end{pmatrix}
\\
u'_f = u_f - \frac{1}{N}\sum_{g=1}^{N} u_g \\
v'_f = v_f - \frac{1}{N}\sum_{g=1}^{N} v_g
\\
\\
u'_f = u_f - \bar{u} \\
v'_f = v_f - \bar{v} \\
\\
\\
[\mathbf{F''}]_f = R^T \times [\mathbf{F'}]_f = R^T \times \lbrace \mathbf{F} - \mathbf{\bar{F}} \rbrace \\
\\
\\
[\mathbf{F}]_f = R \times [\mathbf{F''}]_f + \mathbf{\bar{F}} \\
\\
\\
\begin{pmatrix} X' \\ Y' \end{pmatrix} = R^T \times \begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} r_{11} & r_{21} \\ r_{12} & r_{22} \end{pmatrix} \times \begin{pmatrix} X \\ Y \end{pmatrix}
\\
\\
D' = R^T \times D = R^T \times \begin{pmatrix}
 x_1 & x_2 & x_3 & \isep & x_N\\
 y_1 & y_2 & y_3 & \isep & y_N\\
\end{pmatrix} = \begin{pmatrix} r_{11} & r_{21} \\ r_{12} & r_{22} \end{pmatrix} \times \begin{pmatrix}
 x_1 & x_2 & x_3 & \isep & x_N\\
 y_1 & y_2 & y_3 & \isep & y_N\\
\end{pmatrix} = \begin{pmatrix}
 x'_1 & x'_2 & x'_3 & \isep & x'_N\\
 y'_1 & y'_2 & y'_3 & \isep & y'_N\\
\end{pmatrix}

\\
\\
COV(X;Y) = \frac{1}{N-1}\begin{pmatrix}
 \sum_{i=1}^{N} x_i^2 & \sum_{i=1}^{N} x_i y_i \\
 \sum_{i=1}^{N} x_i y_i & \sum_{i=1}^{N} y_i^2 \\
\end{pmatrix}
= \frac{1}{N-1}\begin{pmatrix}
 x_1 & x_2 & x_3 & \isep & x_N\\
 y_1 & y_2 & y_3 & \isep & y_N\\
\end{pmatrix} \times \begin{pmatrix}
 x_1 & y_1 \\
 x_2 & y_2 \\
 x_3 & y_3 \\
\isep & \isep \\
 x_N & y_N \\
\end{pmatrix} = \frac{1}{N-1} D \times D^T
\\
\\
```



$$\widehat{Var}(F'^i) = \frac{1}{N - 1} \sum_{f=1}^N (u'^i_f)^2 = \frac{1}{N - 1} \sum_{f=1}^N (u^i_f - \bar{u^i})^2 \\$$

$$\widehat{Cov}(F'^i;F'^j) = \frac{1}{N - 1} \sum_{f=1}^N u'^i_f \times u'^j_f = \frac{1}{N - 1} \sum_{f=1}^N (u^i_f - \bar{u^i})(u^j_f - \bar{u^j}) \\$$

$$\widehat{Var}(F'^i) = \widehat{Cov}(F'^i;F'^i) \\$$

$$[\mathbf{\overline{F}}] = \begin{pmatrix}
 \overline{u^1} \\
 \overline{u^2} \\
 \overline{u^3} \\
 \isep \\
 \isep \\
 \overline{u^D}
\end{pmatrix} \\$$

$$[\mathbf{F'}]_f = [\mathbf{F}]_f - [\mathbf{\overline{F}}]$$

Let's assume that $[\bar{\mathbf{F}}] = \begin{pmatrix} \bar{u} \\ \bar{v} \end{pmatrix} = \begin{pmatrix} 2.3 \\ -9.8 \end{pmatrix}$.

The centered samples are: $[\mathbf{F'}]_f = \begin{pmatrix} u'_f \\ v'_f \end{pmatrix} = \begin{pmatrix} u_f - \bar{u} \\ v_f - \bar{v} \end{pmatrix}$.

The corresponding random vector $[\mathbf{F'}]$ and random coordinates u' and v' are:

$$[\mathbf{F'}] = \begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} u - \bar{u} \\ v - \bar{v} \end{pmatrix}$$

Now, let's assume that:

$$COV(u'; v') = \begin{pmatrix} 1.5 & 0.4 \\ 0.4 & 0.8 \end{pmatrix}$$

If we diagonalize this matrix with numpy:

```
COV_up_vp = np.array([[1.5, 0.4], [0.4, 0.8]])
lambda_mu, R = np.linalg.eig(COV_up_vp)
```

we get the following:

$$\Lambda = \begin{pmatrix} \lambda = 1.68150729 & 0 \\ 0 & \mu = 0.61849271 \end{pmatrix}$$

and:

$$R = \begin{pmatrix} 0.91063291 & -0.41321628 \\ 0.41321628 & 0.91063291 \end{pmatrix} = \begin{pmatrix} \vec{\mathbf{r_1}} & \vec{\mathbf{r_2}} \end{pmatrix}$$

such that:

$$COV(u'; v') = R \times \Lambda \times R^T$$

Let's define the "eigen random coordinates" u'' and v'' as:

$$\begin{pmatrix} u'' \\ v'' \end{pmatrix} = R^T \times \begin{pmatrix} u' \\ v' \end{pmatrix}$$

Let's define the "eigen random coordinates" u'' and v'' as:

$$\begin{pmatrix} u'' \\ v'' \end{pmatrix} = R^T \times \begin{pmatrix} u' \\ v' \end{pmatrix}$$

In the eigen referential $(0, \vec{\mathbf{r_1}}, \vec{\mathbf{r_2}})$, the random variables u'' and v'' are normally distributed, and $u'' \sim N(0, \lambda)$ and

Let's draw realizations of u'' and v'' using numpy's random engine:

```
rng = np.random.default_rng()
lambda_0 = rng.normal( 0, np.sqrt(1.68150729) )  # use standard deviation, not varianc
mu_0 = rng.normal( 0, np.sqrt(0.61849271) )      # use standard deviation, not varianc
```

Then, the corresponding realization in the eigen referential $(0, \vec{\mathbf{r_1}}, \vec{\mathbf{r_2}})$ is $[\mathbf{F''}]_0 = \lambda_0 \times \vec{\mathbf{r_1}} + \mu_0 \times \vec{\mathbf{r_2}}$.

The $[\mathbf{F''}]_0$ vector cannot be "unpacked" into a mesh. We need to convert it back to the original "mesh" frame:

$$[\mathbf{F}]_0 = R \times [\mathbf{F''}]_0 + [\bar{\mathbf{F}}] = \begin{pmatrix} 0.91063291 & -0.41321628 \\ 0.41321628 & 0.91063291 \end{pmatrix} \times \begin{pmatrix} \lambda_0 \times 0.91063291 + \mu_0 \times -0.4132162 \\ \lambda_0 \times 0.41321628 + \mu_0 \times 0.9106329 \end{pmatrix}$$

The resulting vector can be "unpacked" into a mesh representing our new shape.

Let's assume that $[\mathbf{\bar{F}}] = \begin{pmatrix} \bar{u} \\ \bar{v} \end{pmatrix} = \begin{pmatrix} 2.3 \\ -9.8 \end{pmatrix}$.

The centered samples are: $[\mathbf{F'}]_f = \begin{pmatrix} u'_f \\ v'_f \end{pmatrix}
= \begin{pmatrix} u_f - \bar{u} \\ v_f - \bar{v} \end{pmatrix}$.

The corresponding random vector $[\mathbf{F'}]$ and random coordinates u' and v' are:

$$[\mathbf{F'}] = \begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} u - \bar{u} \\ v - \bar{v} \end{pmatrix}$$

Now, let's assume that:

$$COV(u'; v') = \begin{pmatrix} 1.5 & 0.4 \\ 0.4 & 0.8 \end{pmatrix} $$

If we diagonalize this matrix with numpy:
```
  COV_up_vp = np.array([[1.5, 0.4], [0.4, 0.8]])
  lambda_mu, R = np.linalg.eig(COV_up_vp)
```

we get the following:

$$\Lambda = \begin{pmatrix} \lambda = 1.68150729 & 0 \\ 0 & \mu = 0.61849271 \end{pmatrix} $$

and:

$$R = \begin{pmatrix} 0.91063291 & -0.41321628 \\ 0.41321628 & 0.91063291 \end{pmatrix} = \begin{pmatrix} \vec{\mathbf{r_1}} & \vec{\mathbf{r_2}} \end{pmatrix}$$

such that:

$$ COV(u'; v') = R \times \Lambda \times R^T $$

Let's define the "eigen random coordinates" u'' and v'' as:

$$ \begin{pmatrix} u'' \\ v'' \end{pmatrix} = R^T \times \begin{pmatrix} u' \\ v' \end{pmatrix}$$

In the eigen referential $(0, \vec{\mathbf{r_1}}, \vec{\mathbf{r_2}})$, the random variables u'' and v'' are normally distributed, and $u'' \sim N(0, \lambda)$ and $v'' \sim N(0, \mu)$.

Let's draw realizations of u'' and v'' using numpy's random engine:
```
  rng = np.random.default_rng()
  lambda_0 = rng.normal( 0, np.sqrt(1.68150729) )  # use standard deviation, not variance!
  mu_0 = rng.normal( 0, np.sqrt(0.61849271) )      # use standard deviation, not variance!
```

Then, the corresponding realization in the eigen referential $(0, \vec{\mathbf{r_1}}, \vec{\mathbf{r_2}})$ is $[\mathbf{F''}]_0 = \lambda_0 \times \vec{\mathbf{r_1}} + \mu_0 \times \vec{\mathbf{r_2}}$.

The $[\mathbf{F''}]_0$ vector cannot be "unpacked" into a mesh. We need to convert it back to the original "mesh" frame:

$$ [\mathbf{F}]_0 = R \times [\mathbf{F''}]_0 + [\mathbf{\bar{F}}]
= \begin{pmatrix} 0.91063291 & -0.41321628 \\ 0.41321628 & 0.91063291 \end{pmatrix}
\times \begin{pmatrix}
\lambda_0 \times 0.91063291 + \mu_0 \times -0.41321628 \\
\lambda_0 \times 0.41321628 + \mu_0 \times 0.91063291 \end{pmatrix} + \begin{pmatrix} 2.3 \\ -9.8 \end{pmatrix}$$

The resulting vector can be "unpacked" into a mesh representing our new shape.