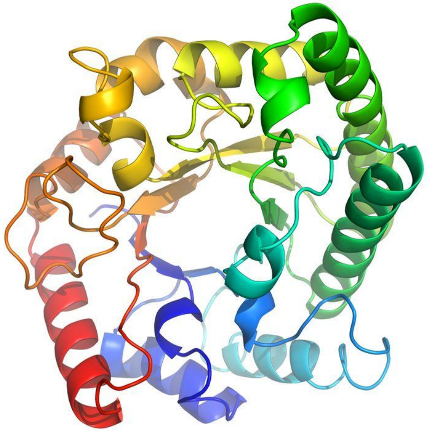


3D Poses Recovery in Single-Particle Cryo-EM from Learned Pairwise Projection Distances

Supervisors:
Laurène Donati (BIG)
Michaël Defferrard (LTS2)

Professor:
Michaël Unser (BIG)

Student:
Jelena Banjac

Section:
SC, Data Science

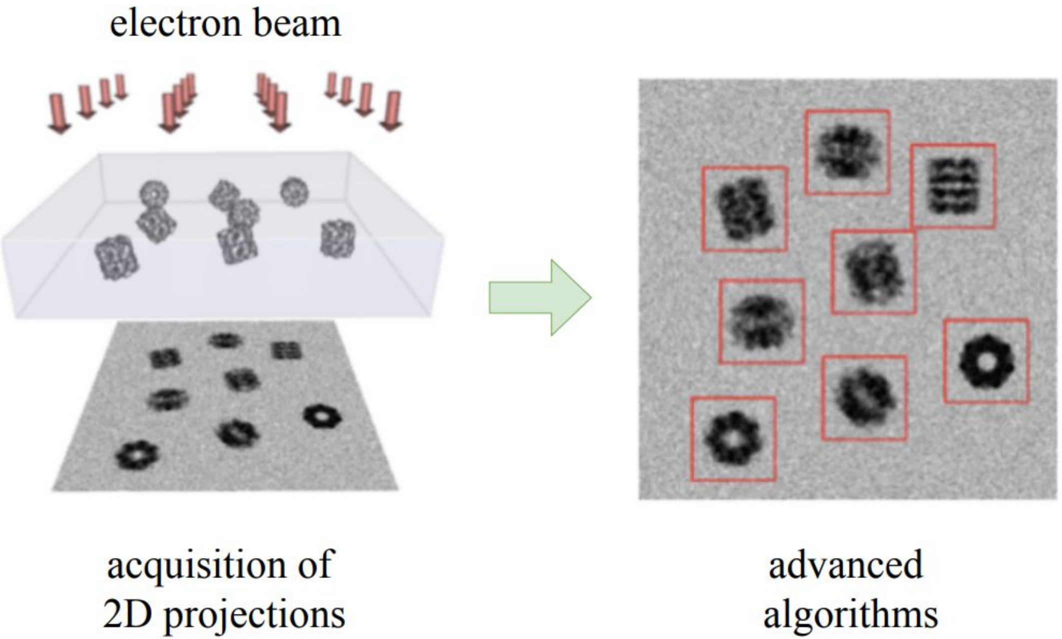
Project type:
Optional semester project (8 cr.)

Lausanne
17th January 2020

Image credits: biology stack exchange question

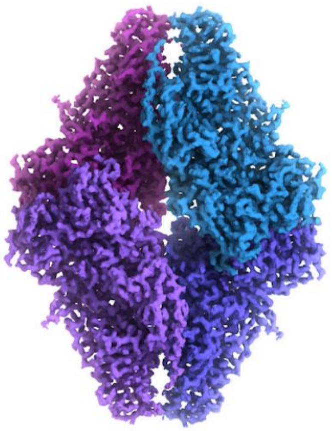
Problem Statement

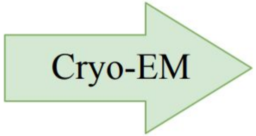
Single-particle Cryo-EM



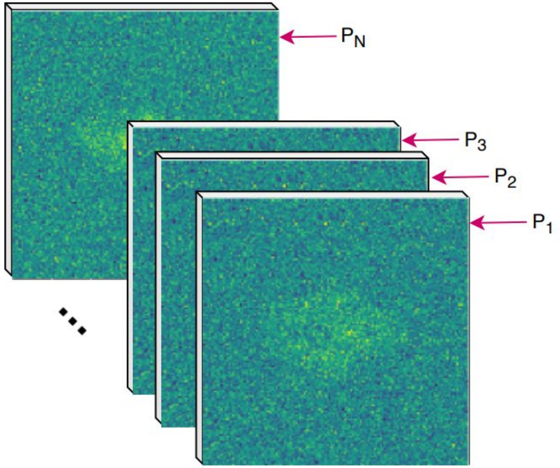
Single-particle Cryo-EM

3D volume of the protein (N copies)





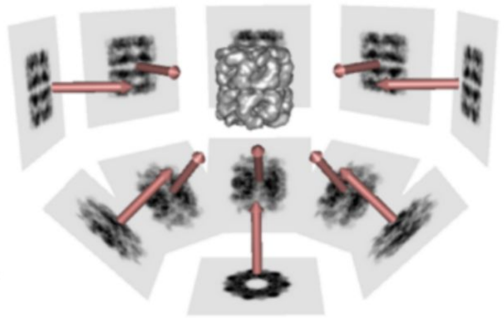
2D projections of the protein (N copies)



$$\mathbf{p}_i = \mathbf{C}_\varphi \mathbf{S}_\mathbf{t} \mathbf{P}_{\theta_i} \mathbf{x} + \mathbf{n}$$

$\mathbf{x} \in \mathbb{R}^V$	- 3D density map
$\mathbf{P}_{\theta_i} : \mathbb{R}^V \to \mathbb{R}^M$	- projection operator
$\mathbf{S}_\mathbf{t} : \mathbb{R}^M \to \mathbb{R}^M$	- shift operator
$\mathbf{C}_\varphi : \mathbb{R}^M \to \mathbb{R}^M$	- convolution operator (with CTF)
$\mathbf{n} \in \mathbb{R}^M$	- additive noise

Imaging Challenge and Project Goal

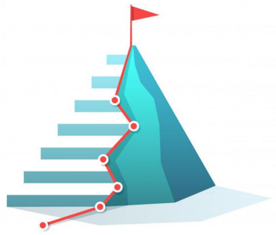


reconstruction pipeline

To **reconstruct** the protein we need to know the **angles** of the projections.

Problem:

Those projection angles are unknown in single-particle cryo-EM.

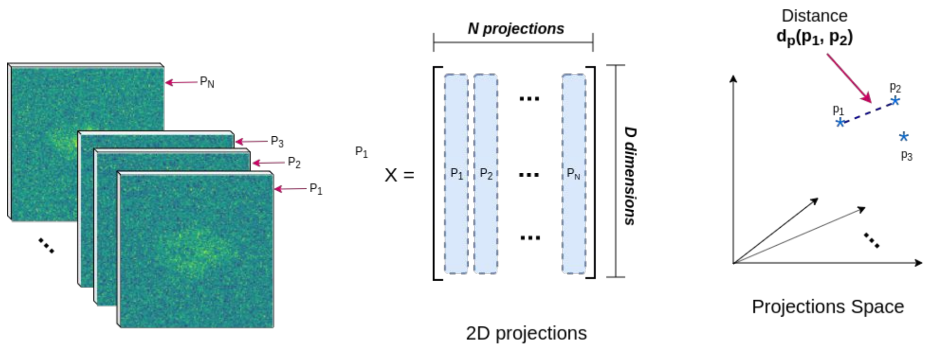


Goal of the Project:

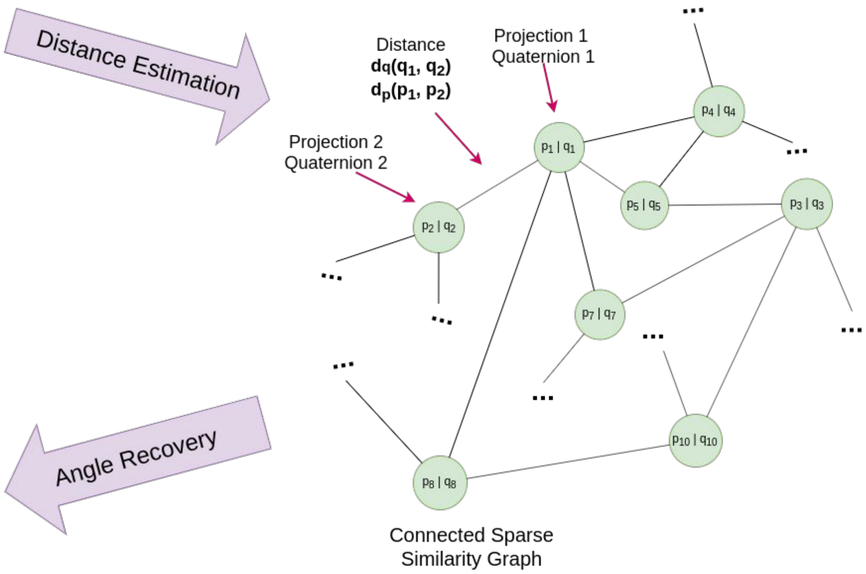
Angles recovery directly from the projections

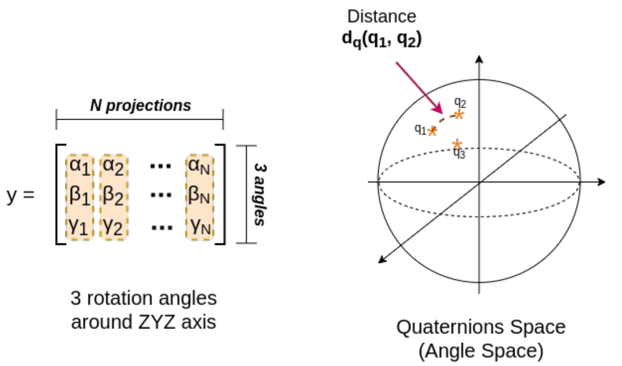
Proposed Method

General Flow



Two-Steps Method





Key assumption:

$$d_p \sim d_q$$

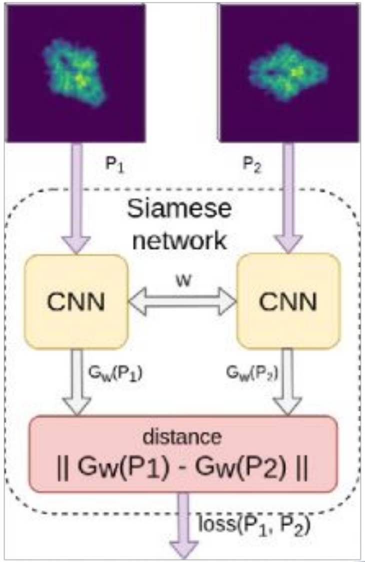
e.g. small d_p => small d_q

1 Distance estimation *(offline)*

$$\hat{d}_p = \arg\min_{d_p} \sum_{i,j} |\boxed{d_p(p_i, p_j)} - d_q(q_i, q_j)|^2$$

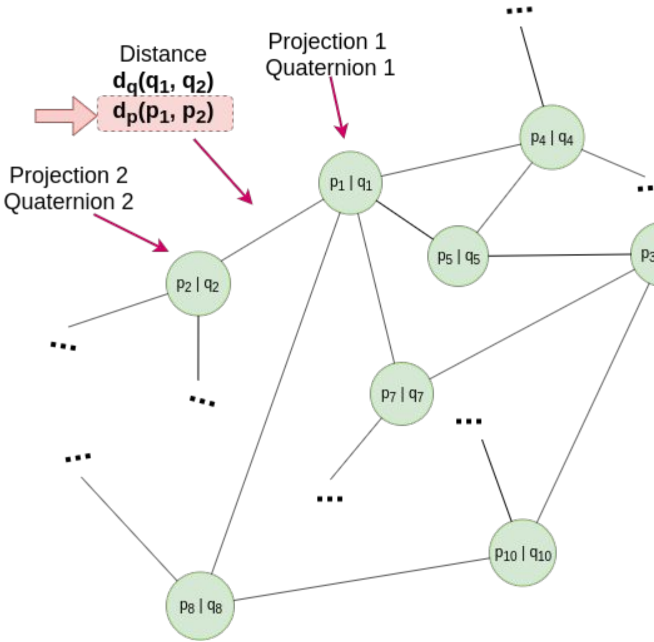
Distance between projections d_p

- Euclidean distance
- Learned distance



Siamese Network trained on:

proteins database
+
simulated measurements



2 Angle recovery *(online)*

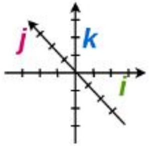
$$\{\hat{q}_i\}_{i=1}^{N} = \arg \min_{\{q_i\}_{i=1}^{N}} \sum_{i\ j} |d_p(p_i, p_j) - d_q(q_i, q_j)|^2$$

Quaternion distance d_q

$$d_q(q_i, q_j) = 2arccos(| < q_i, q_j > |)$$

What is quaternion?

$$q = \underbrace{a}_{\text{Real Part}} + \underbrace{b\,\boldsymbol{i} + c\,\boldsymbol{j} + d\,\boldsymbol{k}}_{\text{Imaginary Part}}$$

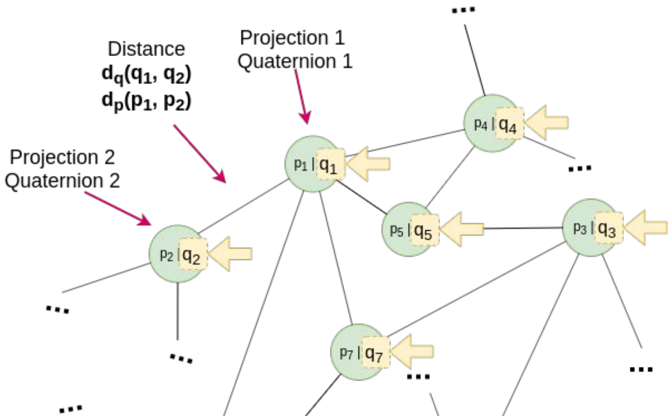


$$i^2 = j^2 = k^2$$
$$ij = -ji = k$$
$$ki = -ik = j$$
$$jk = -kj = i$$

Unit quaternion:

$$i^2 + j^2 + k^2 = 1$$

Assuming we have the distance between the projections, can we estimate their positions on SO(3)?



Relevant Literature (euclidean context): I. Dokmanic, R. Parhizkar, J. Ranieri, and M. Vetterli, "Euclidean distance matrices: essential theory, algorithms, and applications,"IEEE Signal Processing Magazine, vol.32, no.6, pp.12–30, 2015.

Results

Angle Recovery with Perfect Distances

Question:

Is it possible to recover the angles from the perfect distances?

How:

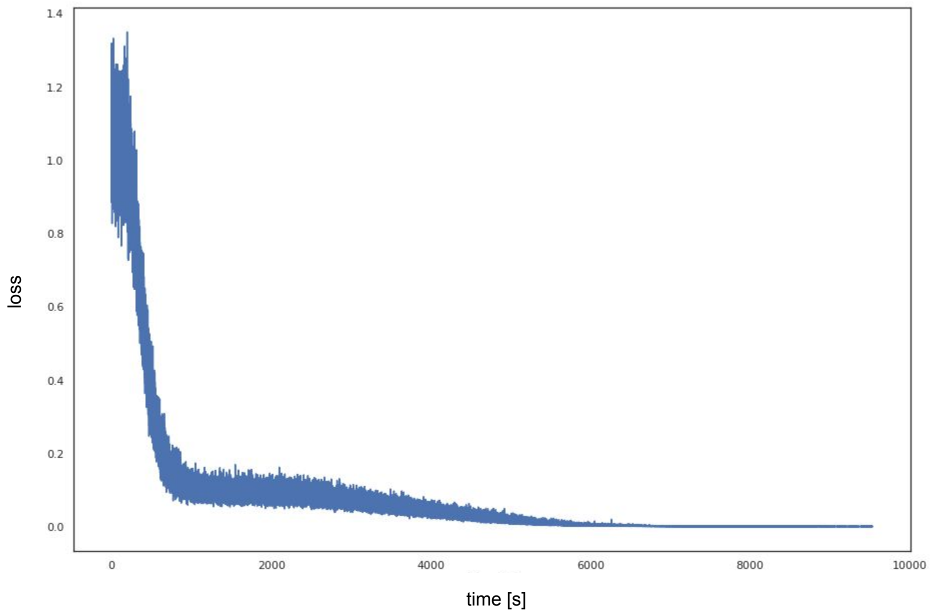
$$\{\hat{q}_i\}_{i=1}^N = \arg \min_{\{q_i\}_{i=1}^N} \sum_i |\underbrace{d_p(p_i, p_j)}_{\text{perfect distance estimation}} - d_q(\hat{q}_i, \hat{q}_j)|^2$$

$$d_p(p_i, p_j) = d_q(q_i, q_j)$$

Experiment 1

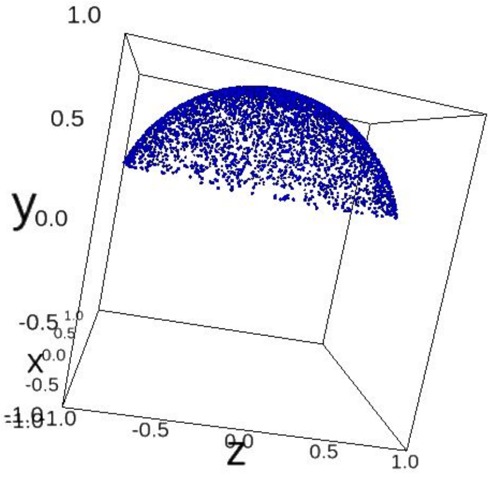
Angle Recovery with Perfect Distances

Optimization result:



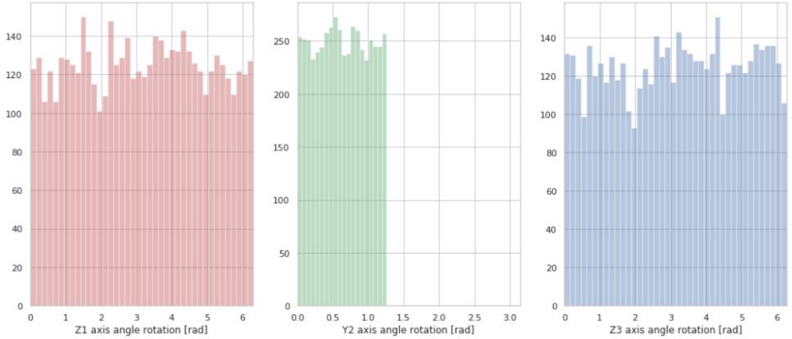
Result:
- optimization loss: 5.23e-04

Sphere coverage:

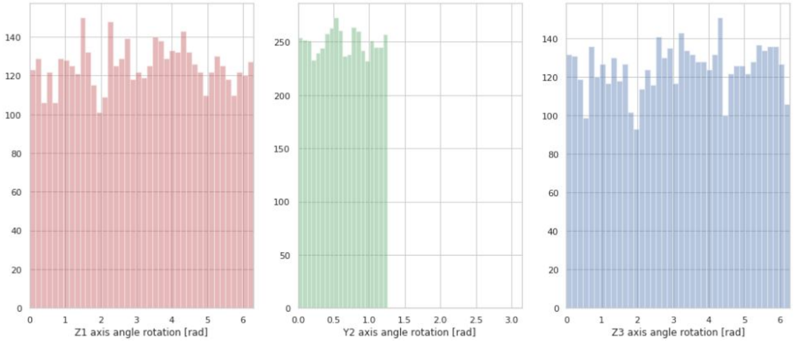


Phase 1: Angle Recovery with Perfect Distances

True angles count



Predicted angles count



Angle Recovery with Perfect Distances

Question:

Is it possible to recover the angles from the perfect distances?

Observation:

It is possible to recover angles from the distances!

Note: This equation will be used as a measure of success (**GT loss***) in following experiments.

Experiment 1

*GT loss = ground truth loss

Angle Recovery with Euclidean Distance

Question:

Is an *Euclidean* d_p a good estimation for d_q?

$$d_p(p_i, p_j) = C \cdot d_q(q_i, q_j)$$

Is angle recovery possible in this setting?

How:

$$\{\hat{q}_i\}_{i=1}^N = \arg\min_{\{q_i\}_{i=1}^N} \sum_i |\underbrace{d_p(p_i, p_j)}_{\text{Euclidean distance estimation (baseline)}} - d_q(\hat{q}_i, \hat{q}_j)|^2$$

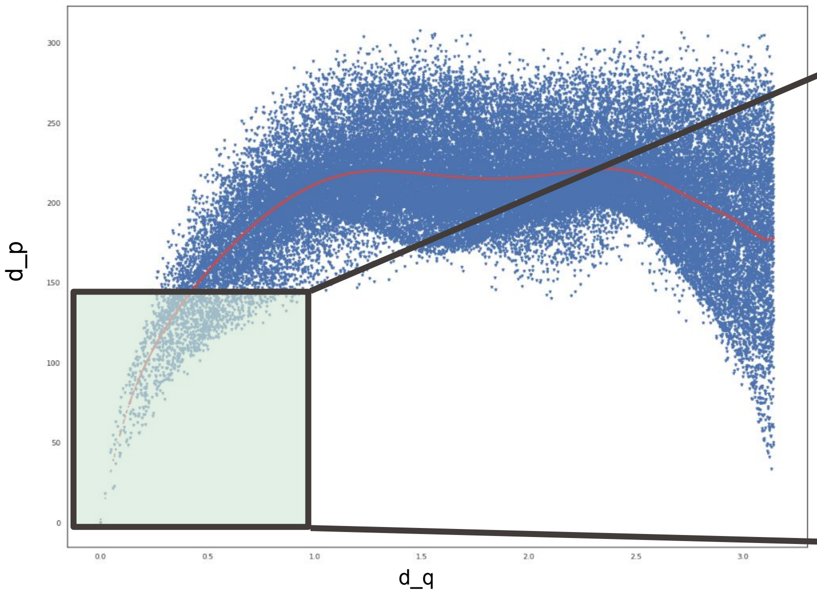
$$d_p(p_i, p_j) = \sqrt{\sum_{k=1}^{n} (p_{i_k} - p_{j_k})^2}$$

Experiment 2

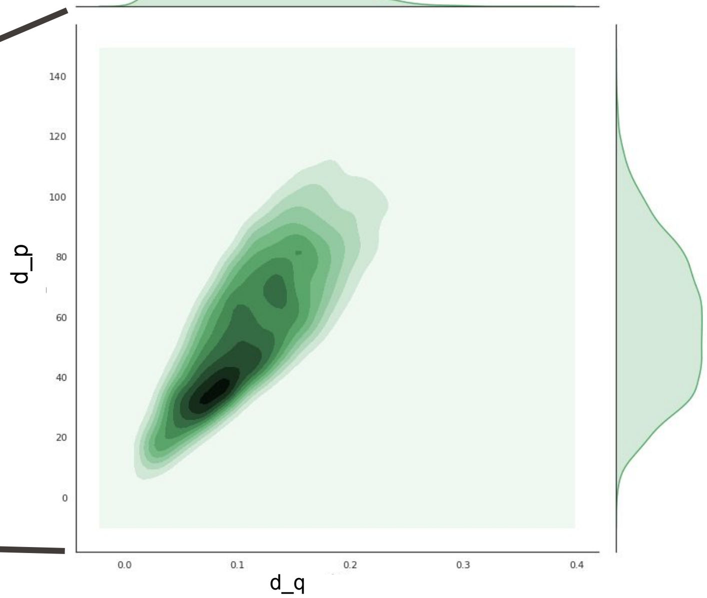
Euclidean Distance as Distance Metric

Question: Is an *Euclidean* d_p a good estimation for d_q?

10 projection distances to all the others:



kNN closest distances density plot:

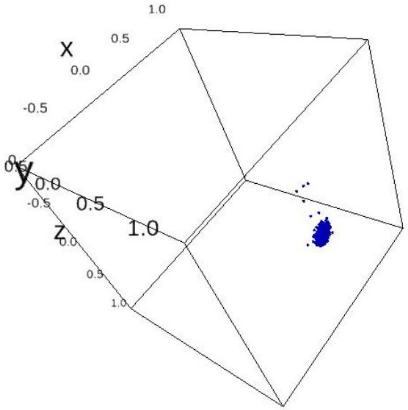


Observation:

Linear relation between $\mathbf{d_p}$ and $\mathbf{d_q}$ only valid for small angle distances!

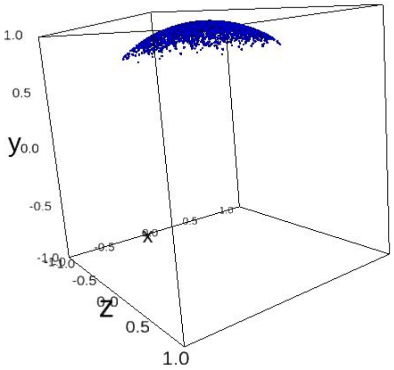
Angle Recovery with Euclidean Distance

Question: What is the effect of sampling strategy in angle recovery?



Settings:
- sampling: **kNN closest**

Result:
- optimization loss: 3.27e-01
- GT loss: 1.4478



Settings:
- sampling: **half kNN closest + half random**

Result:
- optimization loss: 2.20e+00
- GT loss: 1.2948

Observation: Projections are concentrating in angle space!

Angle Recovery with Estimated Distances

Question:

Is an *Euclidean* d_p a good estimation for d_q?

$$d_p(p_i, p_j) = C \cdot d_q(q_i, q_j)$$

Observation:

Linear relation between $\mathbf{d_p}$ and $\mathbf{d_q}$ only valid for small angle distances!

Projections are concentrating in angle space!

Experiment 2

Distance Estimation with Siamese Neural Network

Question:

Is an *SiameseNN* d_p a good approximation of d_q?

$$d_p(p_i, p_j) \approx d_q(q_i, q_j)$$

Is angle recovery possible in this setting?

How:

$$\{\hat{q}_i\}_{i=1}^N = \arg\min_{\{q_i\}_{i=1}^N} \sum_i |\underbrace{d_p(p_i, p_j)}_{\text{SiameseNN distance estimation}} - d_q(\hat{q}_i, \hat{q}_j)|^2$$

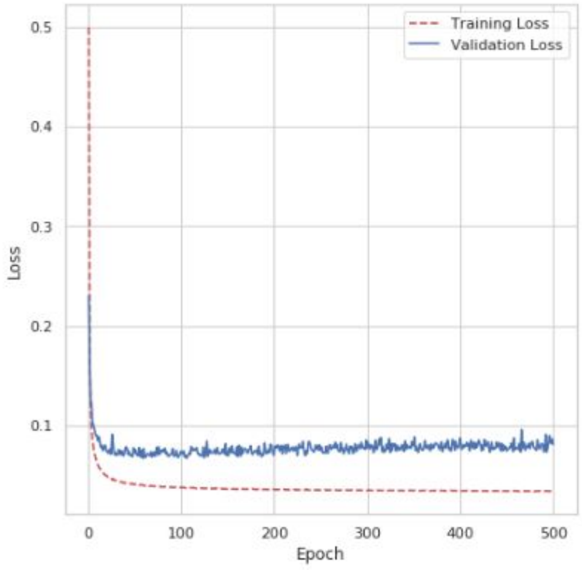
$$d_p(p_i, p_j) = \text{SiameseNN model function}$$

Experiment 3

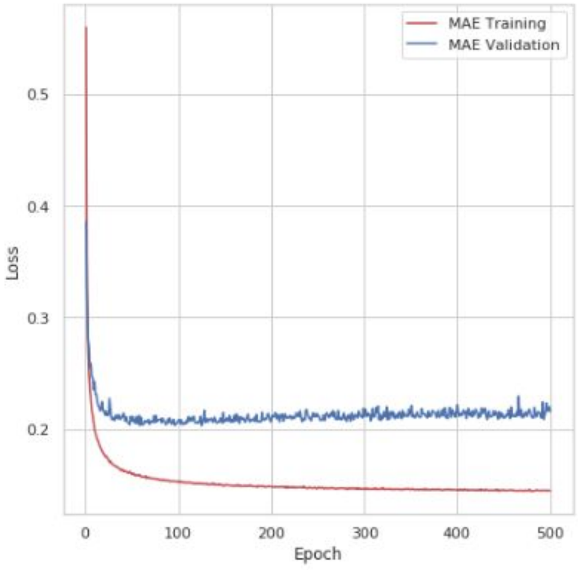
Distance Estimation with Siamese Neural Network

Question: Is it possible to learn distance metric?

MSE error



MAE error



Settings:

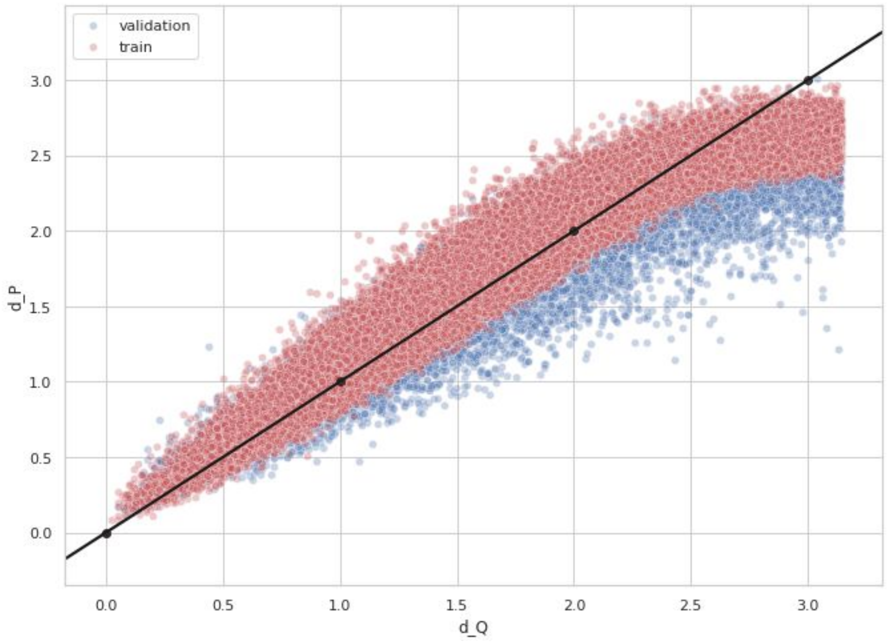
- epochs: 500
- batch size: 256
- # projections: 3K
- # pairs: 60K
- learning rate: 0.001

Observation: Metric learning works, though it overfits.

Distance Estimation with Siamese Neural Network

Question: Is an *Siamese NN* d_p a good approximation of d_q?

Training & validation set projection distances ratio



Observation: linear relation between $\mathbf{d_p}$ and $\mathbf{d_q}$ is valid for small and big distances!

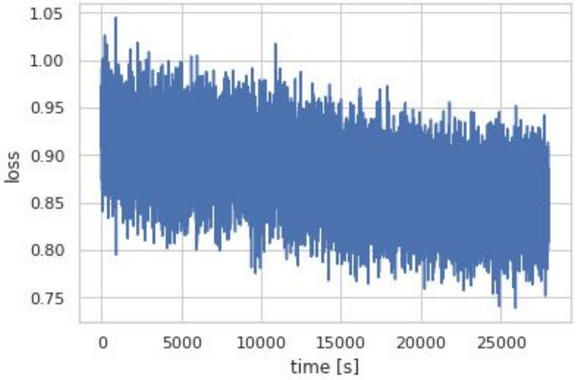
Angle Recovery with Siamese Neural Network

Question: Is angle recovery possible?

Optimization result:

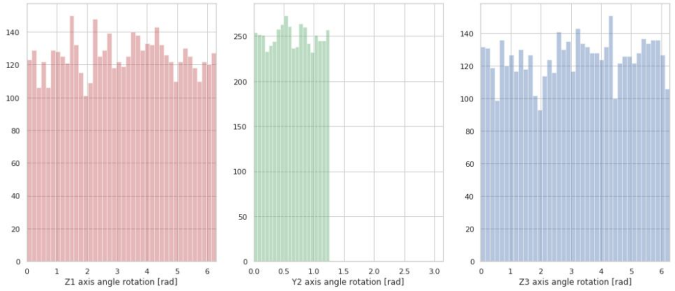
GT Loss *before* angle recovery: **1.0909**

Angle recovery loss: **9.13e-01**

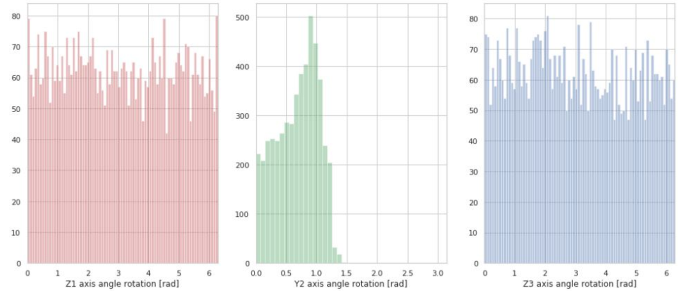


GT Loss *after* angle recovery: **1.0042**

True angles



Predicted angles



Observation: Angle recovery very noisy. GT loss not intuitive/informative.

Distance Estimation with Siamese Neural Network

Question:

Is an *SiameseNN* d_p a good approximation of d_q?

$$d_p(p_i, p_j) \approx d_q(q_i, q_j)$$

Is angle recovery possible in this setting?

Observation:

Metric learning works, though it overfits.

Linear relation between $\mathbf{d_p}$ and $\mathbf{d_q}$ is valid for small and big distances!

Angle recovery very noisy.

Experiment

3

Summary of the Results

1. Two steps work independently
   a. *angle recovery* can recover the angles from perfect distances
   b. *distance estimation* from projections alone is possible
2. We could *not* estimate angles from approximate distance estimations
   a. using Euclidean
   b. using SiameseNN

Future Work

Future work

- Make the angle estimation work with approximate distances
  - better distance estimation
  - more robust angle estimation
- Test on realistic data (noise, CTF, etc.)
  - robustness to noise
  - robustness to unseen protein volumes
  - faithfulness of transfer function representing the projection shift, CTF, noise, etc.
  - final goal to test on real data

Thank you

Questions?

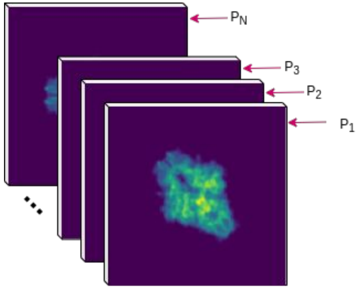
Project Timeline (3 phases)

Phase 1

Angle recovery

d_p metric:
L2 distance

simulated data:
without noise



Phase 2

Distance estimation

d_p metric:
L2 distance

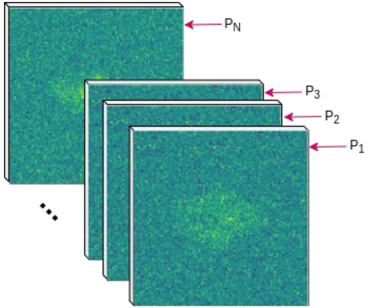
→

d_p metric:
learned distance

Phase 3

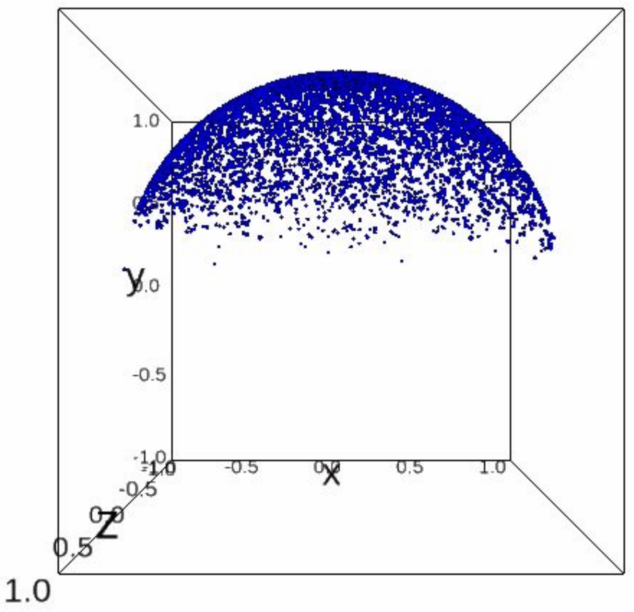
Performance characterization

w.r.t. noise, CPF, positioning, volumes, etc.



Phase 2.2: Angle Recovery with Siamese Neural Network

Sphere coverage:



Observation: We have a half-sphere coverage, but GT loss is not really good. Can we do more?

Angle estimation error metric:

$$\arg\min_{R} \frac{1}{N} \sum_{i=1}^{N} |d_q(q_i, R\hat{q}_i)|$$

R - global rotation quaternion

Phase 2.2: Angle Recovery with Siamese Neural Network

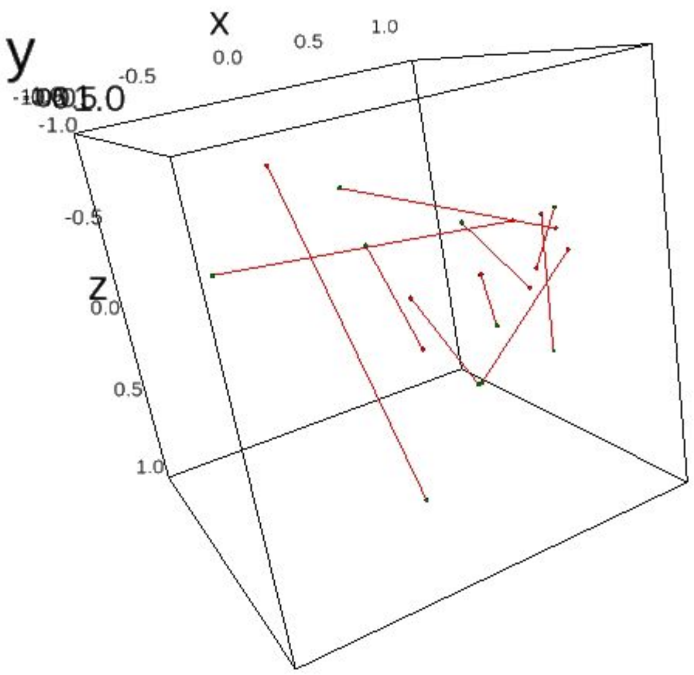
Angle estimation error metric:

$$\arg\min_{R} \frac{1}{N}\sum_{i=1}^{N} \left|d_q(q_i, \boxed{R}\hat{q}_i)\right|$$

R - global rotation quaternion

- true angles

- predicted angles



Before rotation values:

- GT loss: **1.009**
- Angle estimation error: **1.848 rad (~105.9°)**

Phase 2.2: Angle Recovery with Siamese Neural Network

Angle estimation error metric:

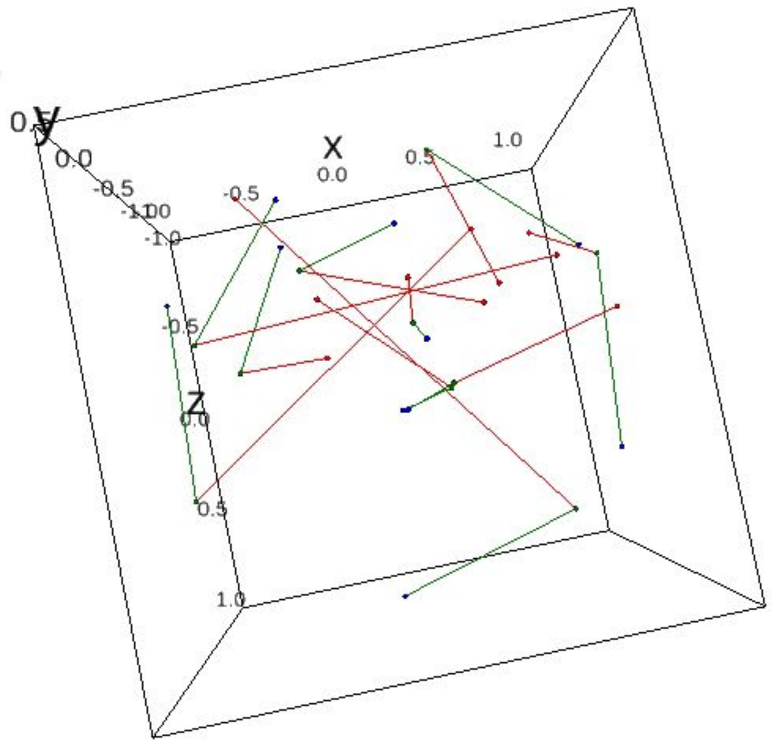
$$\arg\min_{R} \frac{1}{N}\sum_{i=1}^{N} |d_q(q_i, \boxed{R}\hat{q}_i)|$$

R - global rotation quaternion

- true angles

- rotated predicted angles

- initial predicted angles



After rotation values:

- GT loss: **1.004**
- Angle estimation error: **1.845 rad (~105.7°)**

Phase 2.2: Angle Recovery with Siamese Neural Network

Estimated angles initialized as: **true angles**

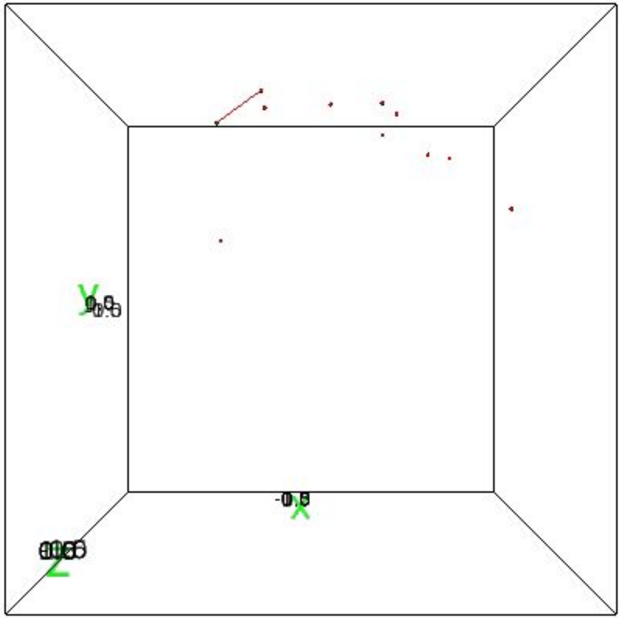
Angle estimation error metric:

$$\arg\min_{R} \frac{1}{N}\sum_{i=1}^{N} |d_q(q_i, \boxed{R}\hat{q}_i)|$$

R - global rotation quaternion

- true angles

- predicted angles



Before rotation values:

- GT loss: **0.32**
- Angle estimation error: **0.1867 rad (~10.69°)**

Phase 2.2: Angle Recovery with Siamese Neural Network

Estimated angles initialized as: **true angles**

Angle estimation error metric:

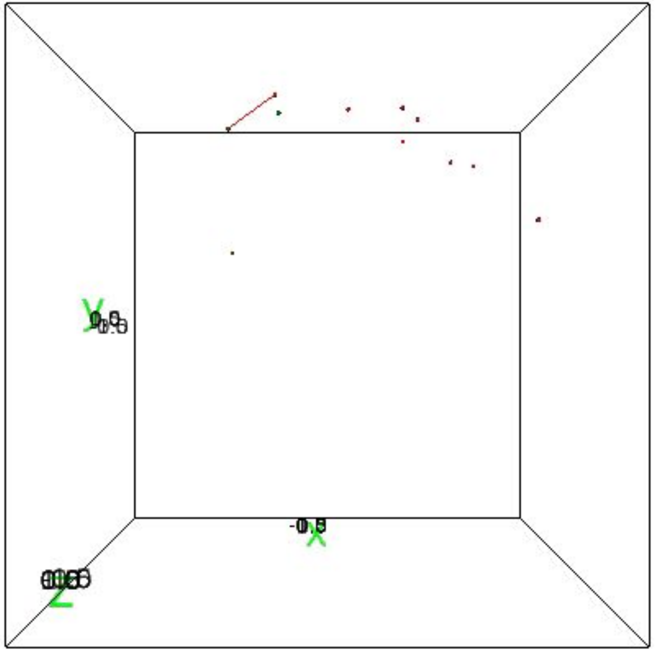
$$\arg\min_{R} \frac{1}{N}\sum_{i=1}^{N} |d_q(q_i, \boxed{R}\hat{q}_i)|$$

R - global rotation quaternion

- true angles

- rotated predicted angles

- initial predicted angles



After rotation values:

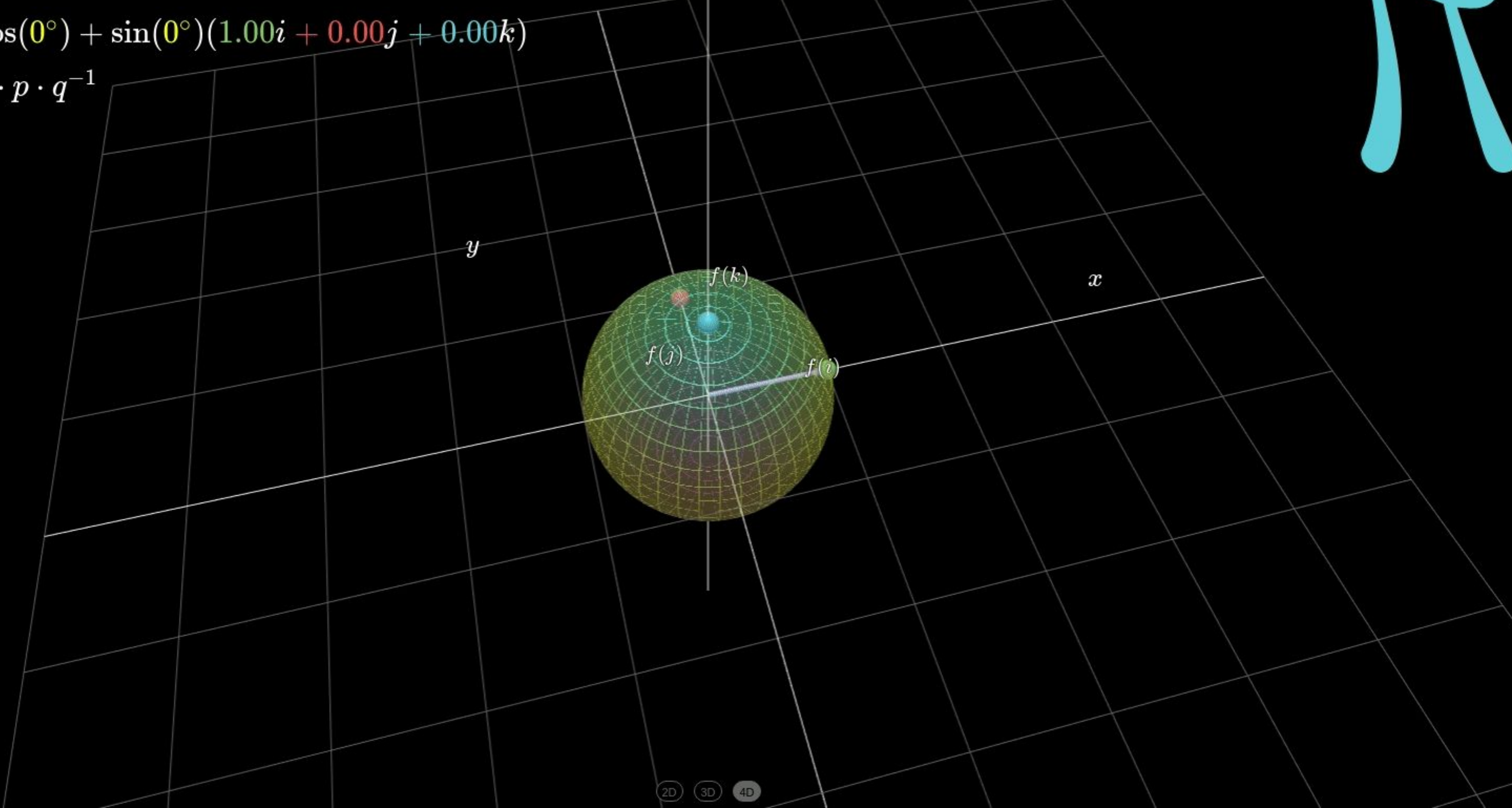
- GT loss: **0.324**
- Angle estimation error: **0.188 rad (~10.77°)**

$$q = \cos(0^\circ) + \sin(0^\circ)(1.00i + 0.00j + 0.00k)$$

$p =$ i × j × k × ijk sphere ×

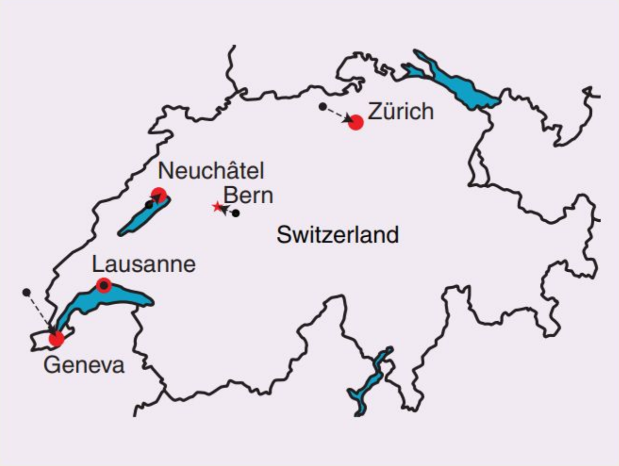
$$q^{-1} = \cos(0^\circ) + \sin(0^\circ)(1.00i + 0.00j + 0.00k)$$

$$f(p) = q \cdot p \cdot q^{-1}$$

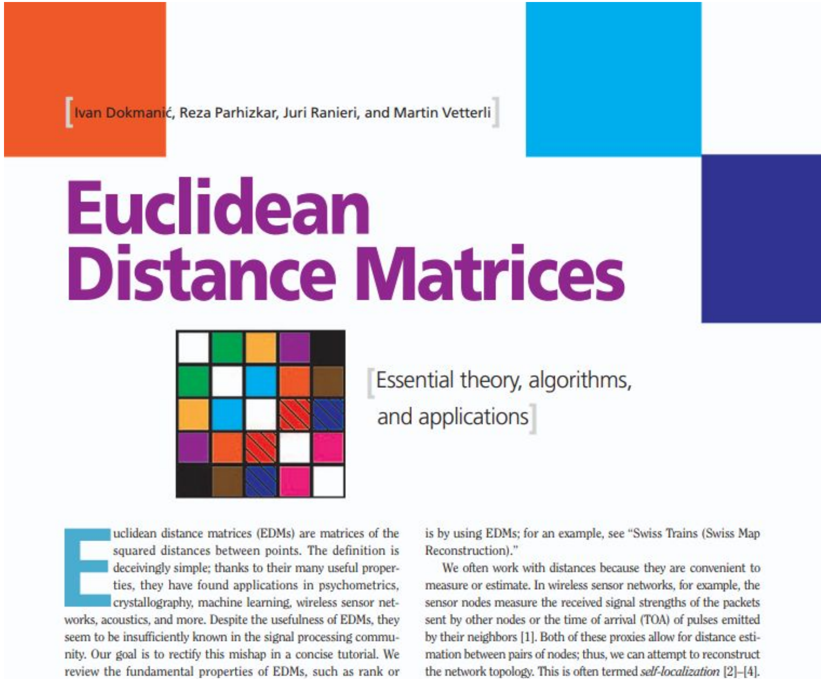


Relevant Literature

$$
\begin{array}{l}
 \\
\text{Lausanne} \\
\text{Geneva} \\
\text{Zürich} \\
\text{Neuchâtel} \\
\text{Bern}
\end{array}
\begin{array}{c}
\begin{array}{ccccc} \text{L} & \text{G} & \text{Z} & \text{N} & \text{B} \end{array} \\
\begin{pmatrix}
0 & 33 & 128 & 40 & 66 \\
33 & 0 & 158 & 64 & 101 \\
128 & 158 & 0 & 88 & 56 \\
40 & 64 & 88 & 0 & 34 \\
66 & 101 & 56 & 34 & 0
\end{pmatrix}.
\end{array}
$$



[FIGS1] A map of Switzerland with the true locations of five cities (red) and their locations estimated by using classical MDS on the train schedule (black).



[Ivan Dokmanić, Reza Parhizkar, Juri Ranieri, and Martin Vetterli]

Euclidean Distance Matrices

[Essential theory, algorithms, and applications]

Euclidean distance matrices (EDMs) are matrices of the squared distances between points. The definition is deceivingly simple; thanks to their many useful properties, they have found applications in psychometrics, crystallography, machine learning, wireless sensor networks, acoustics, and more. Despite the usefulness of EDMs, they seem to be insufficiently known in the signal processing community. Our goal is to rectify this mishap in a concise tutorial. We review the fundamental properties of EDMs, such as rank or is by using EDMs; for an example, see "Swiss Trains (Swiss Map Reconstruction)."

We often work with distances because they are convenient to measure or estimate. In wireless sensor networks, for example, the sensor nodes measure the received signal strengths of the packets sent by other nodes or the time of arrival (TOA) of pulses emitted by their neighbors [1]. Both of these proxies allow for distance estimation between pairs of nodes; thus, we can attempt to reconstruct the network topology. This is often termed *self-localization* [2]–[4].

Image credits: Euclidean Distance Matrices, [Ivan Dokmanic, Reza Parhizkar, Juri Ranieri, and Martin Vetterli], IEEE IEEE SIGNAL PROCESSING MAGAZINE IEEE SIGNAL PROCESSING MAGAZINE, 2015

Future work

Done:

- ✓ Data without noise
- ✓ Euclidean distance for projections
- ✓ Working in quaternion and projection spaces
- ✓ Using kNN to create sparse connected graphs
- ✓ Output of Siamese network used as a projections distance
- ✓ New angle estimation error metric

Optimizations

$$\arg\min \sum_{i,j} |\underbrace{d_p}_{1}(p_i, p_j) - \underbrace{d_q(q_i, q_j)}_{2}|^2$$

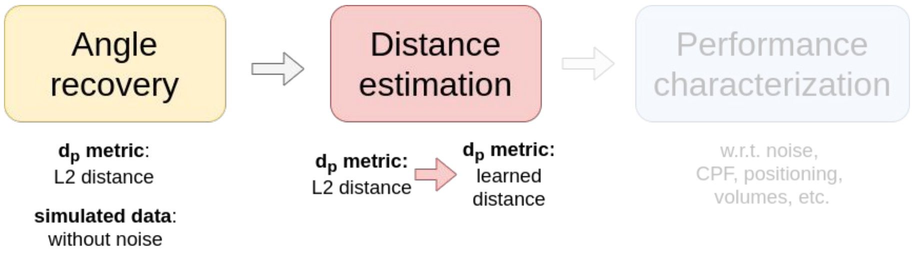
1

Distance estimation
(offline)

2

Angle recovery
(online)

p_i, p_j - i^{th} and j^{th} projections
q_i, q_j - i^{th} and j^{th} quaternions (projection angles)
d_p - distance between projections
d_q - distance between quaternions (projection angles)



Angle recovery
Distance estimation
Performance characterization
d_p metric: L2 distance
simulated data: without noise
d_p metric: L2 distance
d_p metric: learned distance
w.r.t. noise, CPF, positioning, volumes, etc.