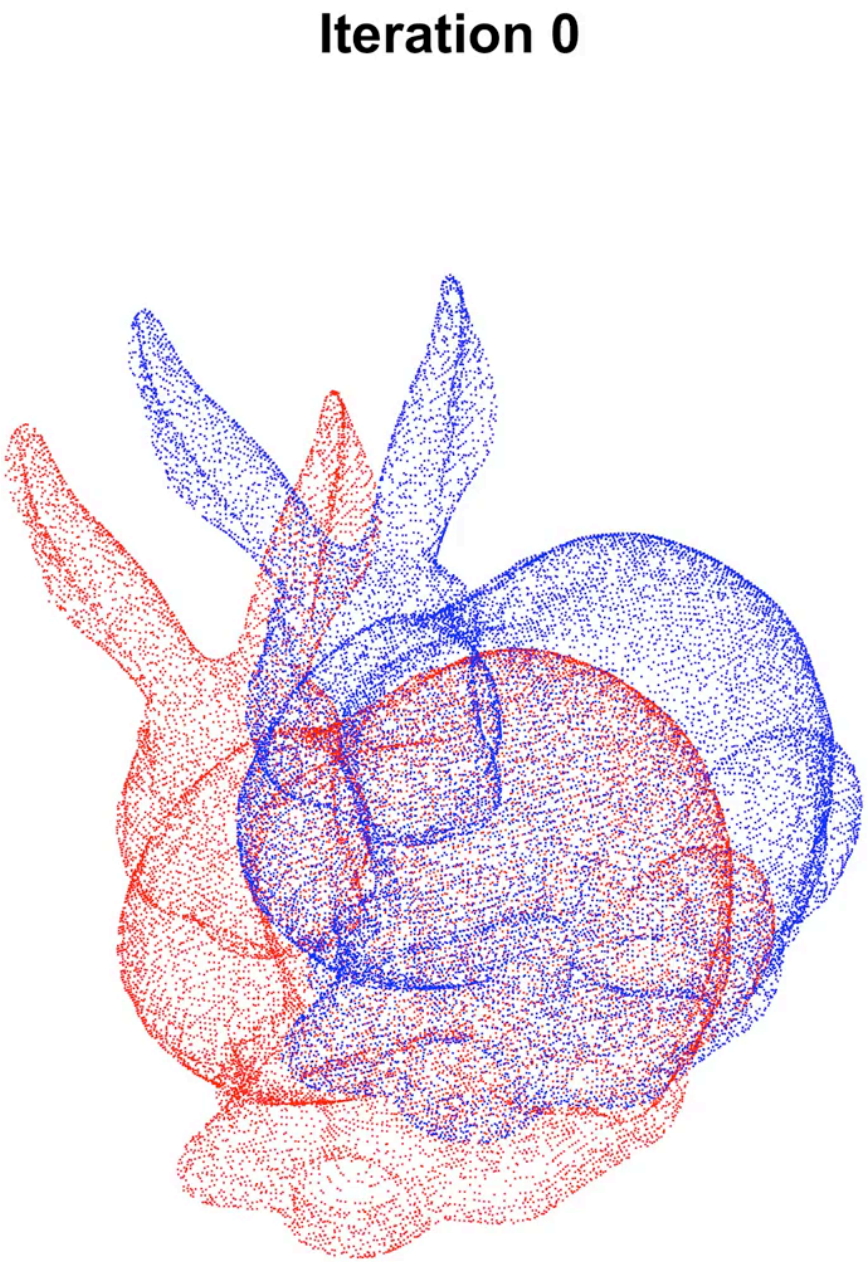
Photogrammetry & Robotics Lab

Point Cloud Registration Part 3: using Non-Linear Least Squares

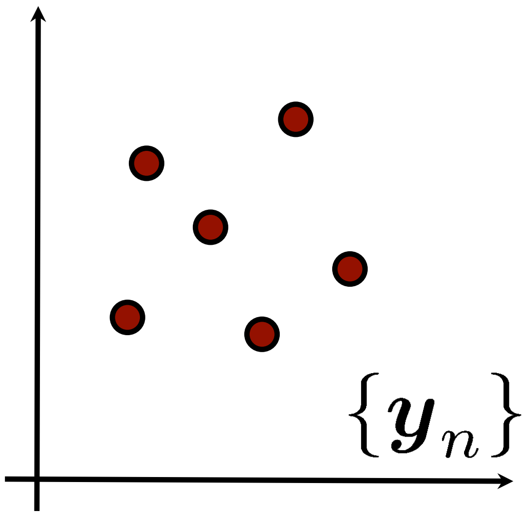
Cyrill Stachniss

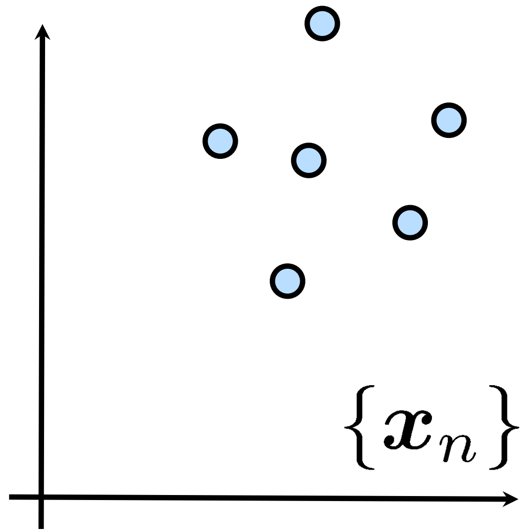
3D Point Cloud Registration Example



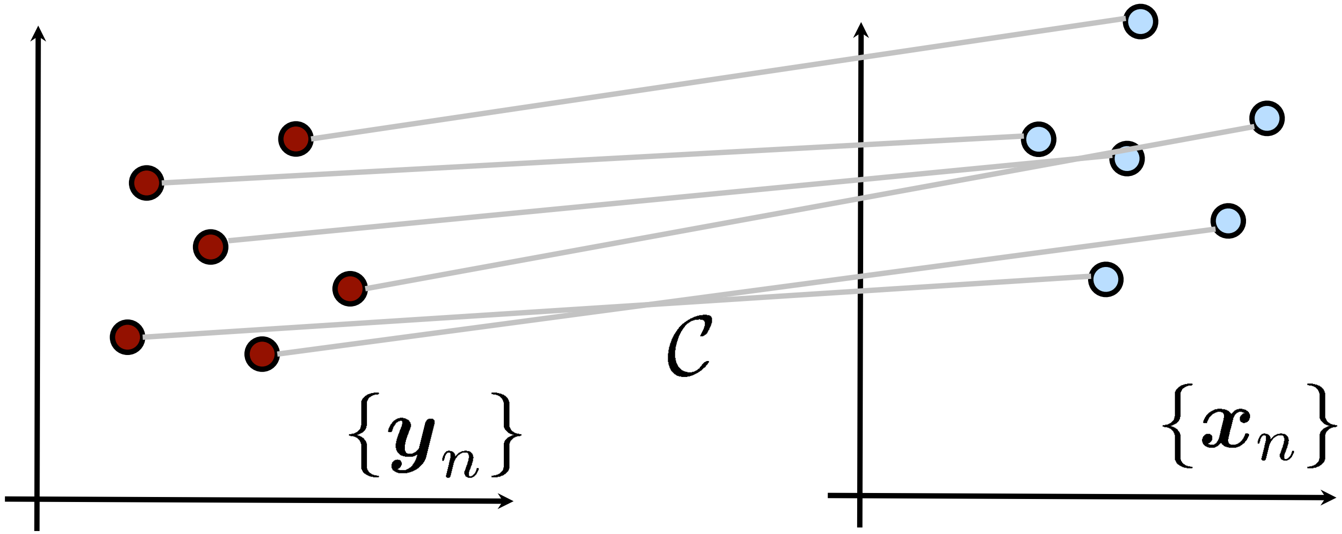
[Video courtesy: P. Glira]

Simple Form of Point Cloud Registration



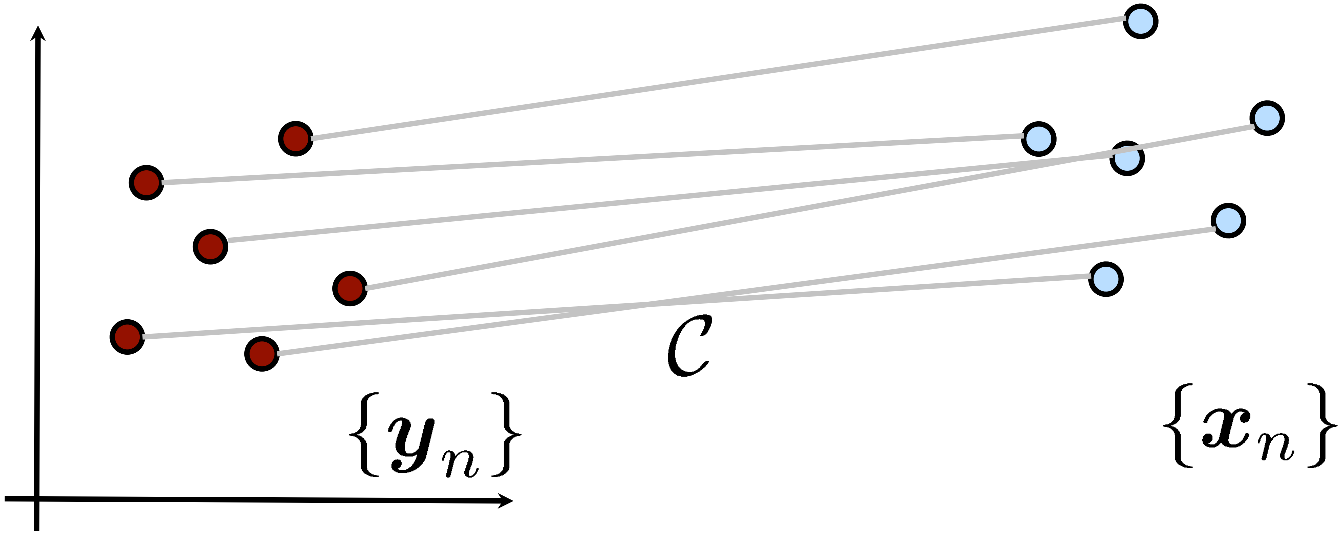


Simple Form of Point Cloud Registration



Simple Form of Point Cloud Registration

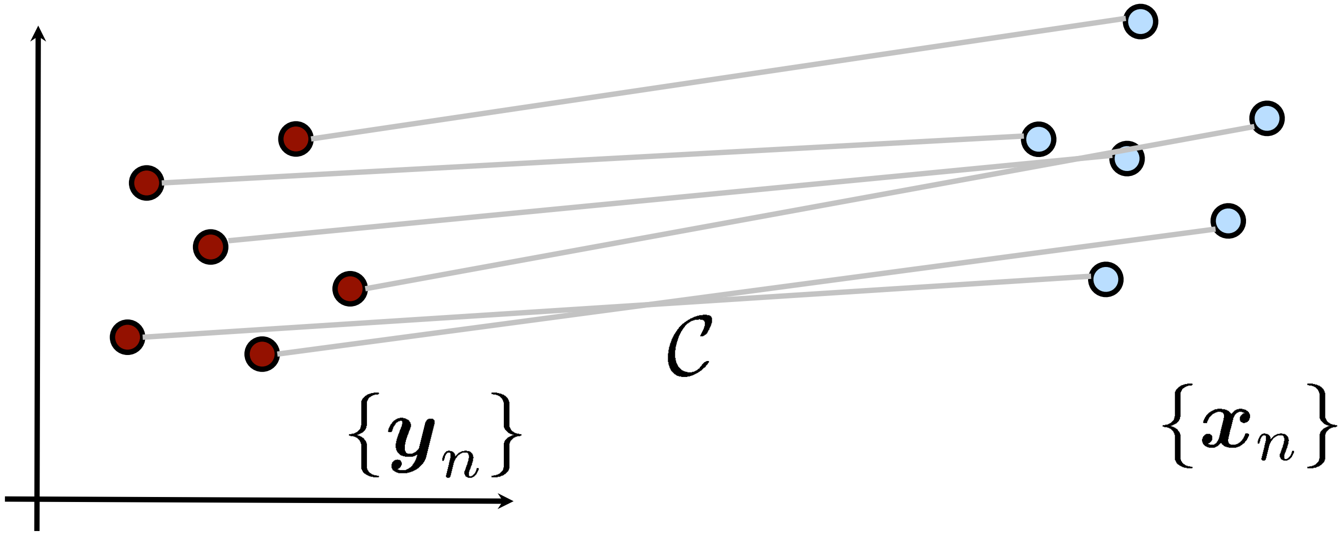
$$\bar{\boldsymbol{x}}_n = R\boldsymbol{x}_n + \boldsymbol{t}$$



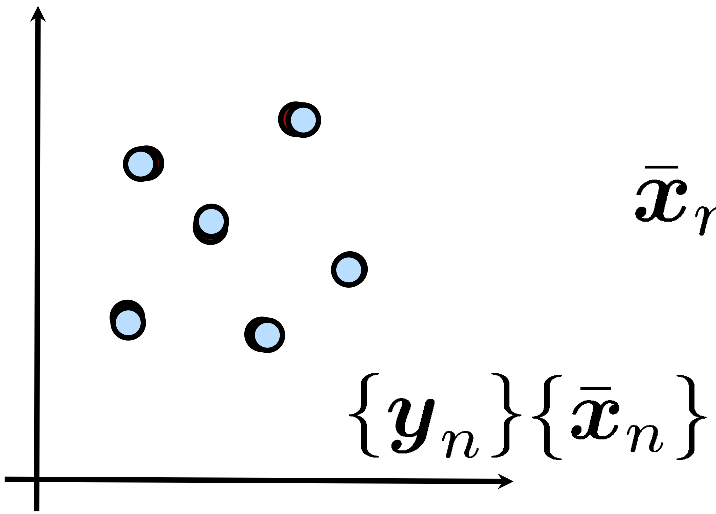
Simple Form of Point Cloud Registration

$$\bar{\boldsymbol{x}}_n = R\boldsymbol{x}_n + \boldsymbol{t}$$

$$\sum ||\boldsymbol{y}_n - \bar{\boldsymbol{x}}_n||^2 \rightarrow \min$$



Simple Form of Point Cloud Registration

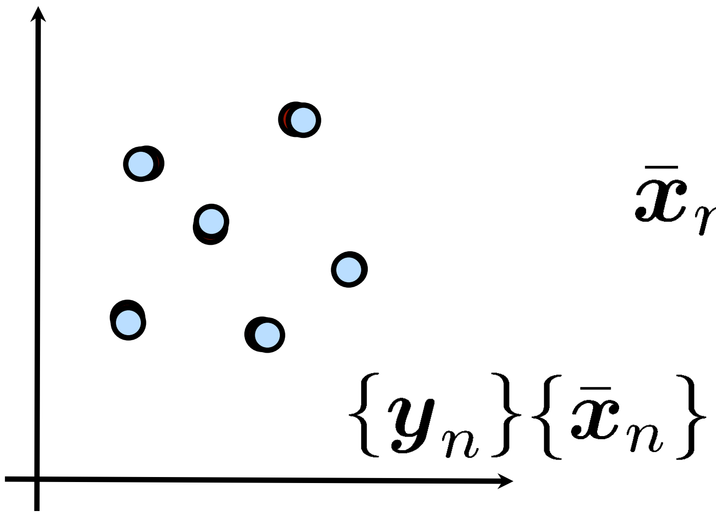


$$\bar{\boldsymbol{x}}_n = R\boldsymbol{x}_n + \boldsymbol{t}$$

Simple Form of Point Cloud Registration

$$\sum ||\boldsymbol{y}_n - \bar{\boldsymbol{x}}_n||^2 \rightarrow \min$$

least squares solution!



$$\bar{\boldsymbol{x}}_n = R\boldsymbol{x}_n + \boldsymbol{t}$$

Registration of 3D Data Points

- **Goal:** find the parameters of the transformation that best align corresponding data points
- Optimization / search for parameters
  - Iterative closest point (ICP w/ SVD)
  - Robust **least squares** approaches **(#3)**
- Known (#1) vs. estimated (#2) correspondences

Reminder

Part 1
Point Cloud Registration with Known Data Association

We have derived an efficient to compute, optimal, direct solution

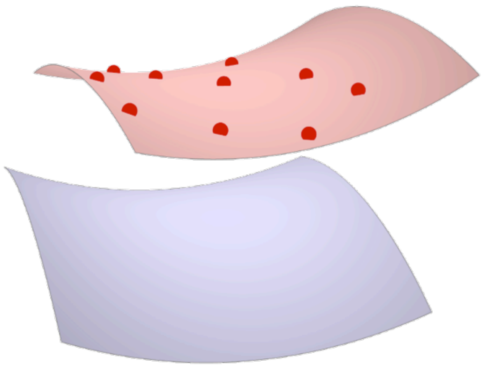
Reminder

Part 2
Point Cloud Registration with Unknown Data Association

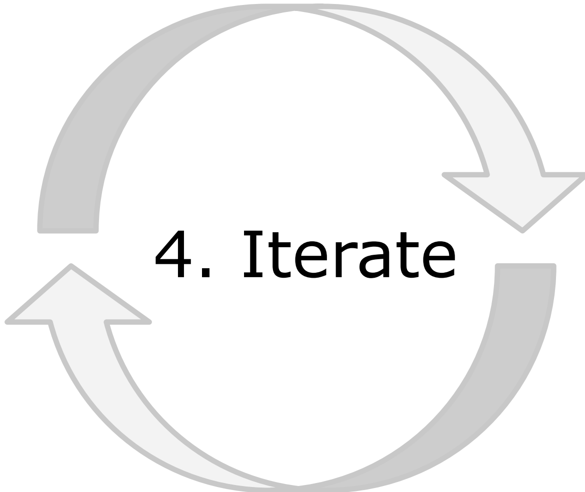
No direct and optimal solution exists but we can register clouds using iterative approaches estimating correspondences

Reminder

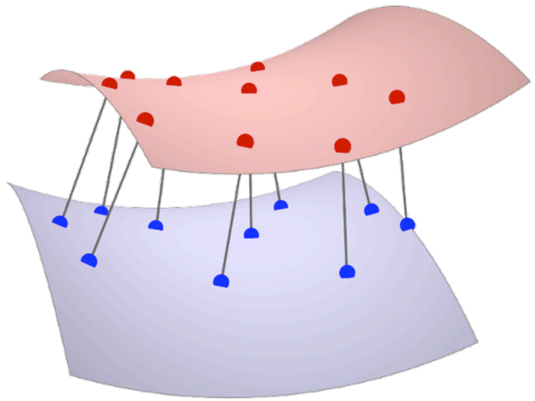
ICP Illustrated



1. Select points on one mesh or point cloud

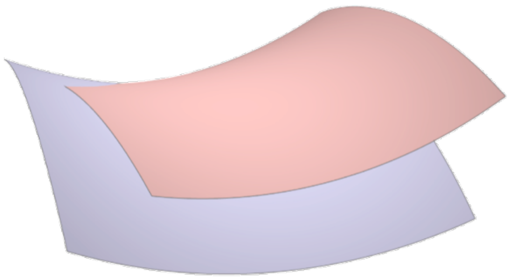


4. Iterate



2. Find closest on other mesh or point cloud

3. Minimize distances



[Courtesy of Rusinkiewicz]

ICP Variants

Reminder

Variants on the following stages of ICP have been proposed:

1. Consider point subsets
2. Different data association strategies
3. Weight the correspondences
4. Reject potential outlier point pairs

Finding Correspondences

Reminder

- There a various different ways to find correspondences
- Investing into a good data association is key obtaining good results
- Exploit any initial guess
- Normal-based metrics often better than standard point-to-point metric
- Outlier rejection is important, especially in dynamic environments

Part 3
Point Cloud Registration using Non-Linear Least Squares

Why a Least Squares Approach?

- SVD solution assumes point-to-point correspondences
- More complex error functions require a more general least squares approach
- LS approach can better consider uncertainties (3D point covariances)
- Often solved via an iterative Gauss Newton-based minimization

Start with Least Squares for 2D Point-to-Point Registration

Gauss Newton Minimization

- Example in 2D for point-to-point
- Objective: $\Phi(t_x, t_y, \theta) = \sum ||\bar{\boldsymbol{x}}_n - \boldsymbol{y}_n||^2 \rightarrow \min$
- Error vector: $\boldsymbol{e}_n = \bar{\boldsymbol{x}}_n - \boldsymbol{y}_n$
- Expands to: $\boldsymbol{e}_n = R\boldsymbol{x}_n + \boldsymbol{t} - \boldsymbol{y}_n$
- Parameters: $[t_x, t_y, \theta]^\top$
- Explicitly: $\boldsymbol{e}_n(t_x, t_y, \theta) = R(\theta)\,\boldsymbol{x}_n + [t_x, t_y]^\top - \boldsymbol{y}_n$
- Linearize the non-linear error function

How does the Jacobian looks like?

Jacobian for 2D Points

- Computing the Jacobian

$$
\begin{aligned}
J_n(\boldsymbol{x}) &= \frac{\partial \boldsymbol{e}_n}{\partial \boldsymbol{x}} \\
&= \left[\frac{\partial \boldsymbol{e}_n}{\partial t_x}, \frac{\partial \boldsymbol{e}_n}{\partial t_y}, \frac{\partial \boldsymbol{e}_n}{\partial \theta}\right] \\
&= \begin{bmatrix} \frac{\partial \boldsymbol{e}_n^x}{\partial t_x}, & \frac{\partial \boldsymbol{e}_n^x}{\partial t_y}, & \frac{\partial \boldsymbol{e}_n^x}{\partial \theta} \\ \frac{\partial \boldsymbol{e}_n^y}{\partial t_x}, & \frac{\partial \boldsymbol{e}_n^y}{\partial t_y}, & \frac{\partial \boldsymbol{e}_n^y}{\partial \theta} \end{bmatrix}
\end{aligned}
$$

- leads to a 2 x 3 matrix in our case

Jacobian for 2D Points

- Computing the Jacobian for the error vector $\boldsymbol{e}_n(t_x, t_y, \theta) = R(\theta)\, \boldsymbol{x}_n + [t_x, t_y]^\top - \boldsymbol{y}_n$

$$J_n = \left[\frac{\partial \boldsymbol{e}_n}{\partial t_x}, \frac{\partial \boldsymbol{e}_n}{\partial t_y}, \frac{\partial \boldsymbol{e}_n}{\partial \theta}\right] = \left[I; R'_\theta \boldsymbol{x}_n\right]$$

$$= \begin{bmatrix} 1 & 0 & -\sin\theta\ x_n - \cos\theta\ y_n \\ 0 & 1 & \cos\theta\ x_n - \sin\theta\ y_n \end{bmatrix}$$

- as

$$R'_\theta = \frac{\partial}{\partial\theta} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} -\sin\theta & -\cos\theta \\ \cos\theta & -\sin\theta \end{bmatrix}$$

Gauss Newton Minimization

- Example in 2D for point-to-point
- Objective: $\Phi = \sum ||\bar{\boldsymbol{x}}_n - \boldsymbol{y}_n||^2 \rightarrow \min$
- Error vector: $\boldsymbol{e}_n = R\boldsymbol{x}_n + \boldsymbol{t} - \boldsymbol{y}_n$
- Jacobian: $J_n = \begin{bmatrix} 1 & 0 & -\sin\theta \; x_n - \cos\theta \; y_n \\ 0 & 1 & \cos\theta \; x_n - \sin\theta \; y_n \end{bmatrix}$

Todo

- Compute normal equation matrix
- Compute right-hand side
- Solve resulting linear system

Gauss Newton Minimization

- Jacobian: $J_n = \begin{bmatrix} 1 & 0 & -\sin\theta\ x_n - \cos\theta\ y_n \\ 0 & 1 & \cos\theta\ x_n - \sin\theta\ y_n \end{bmatrix}$
- Matrix: $H_n = J_n^\top J_n$
- Right-hand side: $\boldsymbol{b}_n = J_n^\top \boldsymbol{e}_n$
- Compute terms over all points

$$H = \sum_n H_n = \sum_n J_n^\top J_n \qquad \boldsymbol{b} = \sum_n \boldsymbol{b}_n = \sum_n J_n^\top \boldsymbol{e}_n$$

- Solve $H\Delta\boldsymbol{x} = -b$ via $\Delta\boldsymbol{x} = -H^{-1}\boldsymbol{b}$
- Update parameters $\boldsymbol{x} \leftarrow \boldsymbol{x} + \Delta\boldsymbol{x}$
- Iterative until convergence

2D Least Squares Example

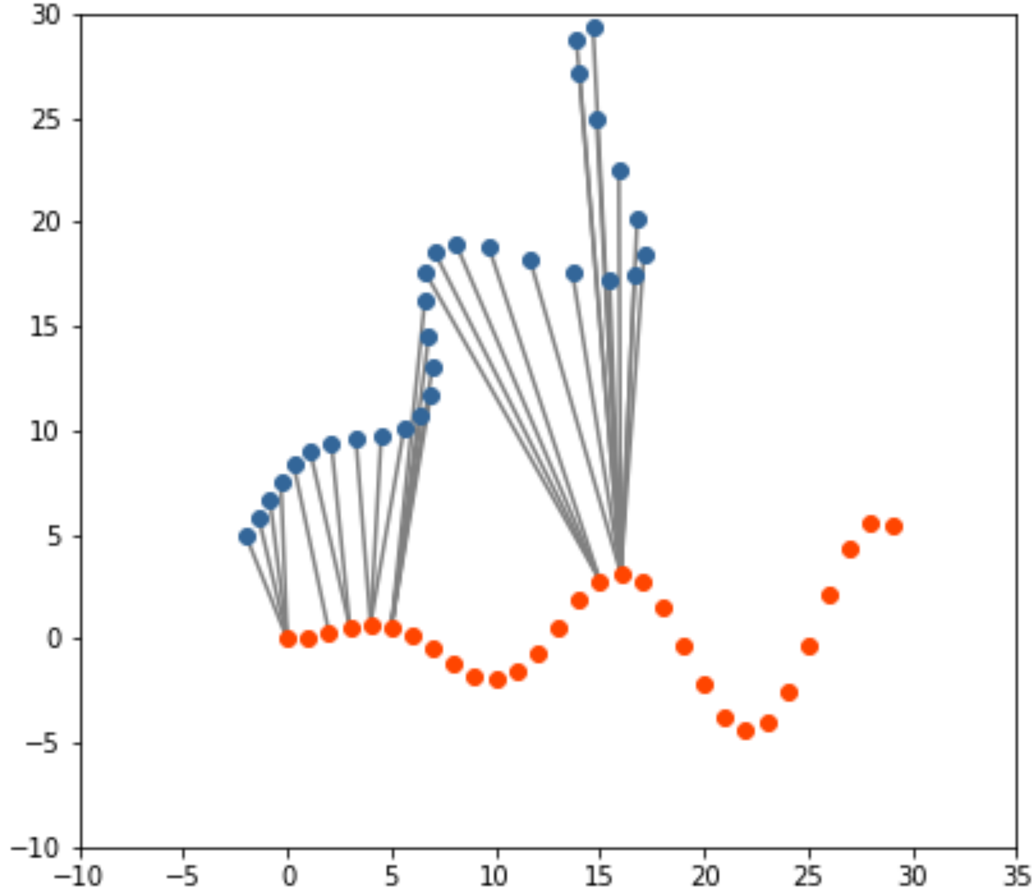


Image courtesy: Bogoslavskyi

2D Least Squares Example

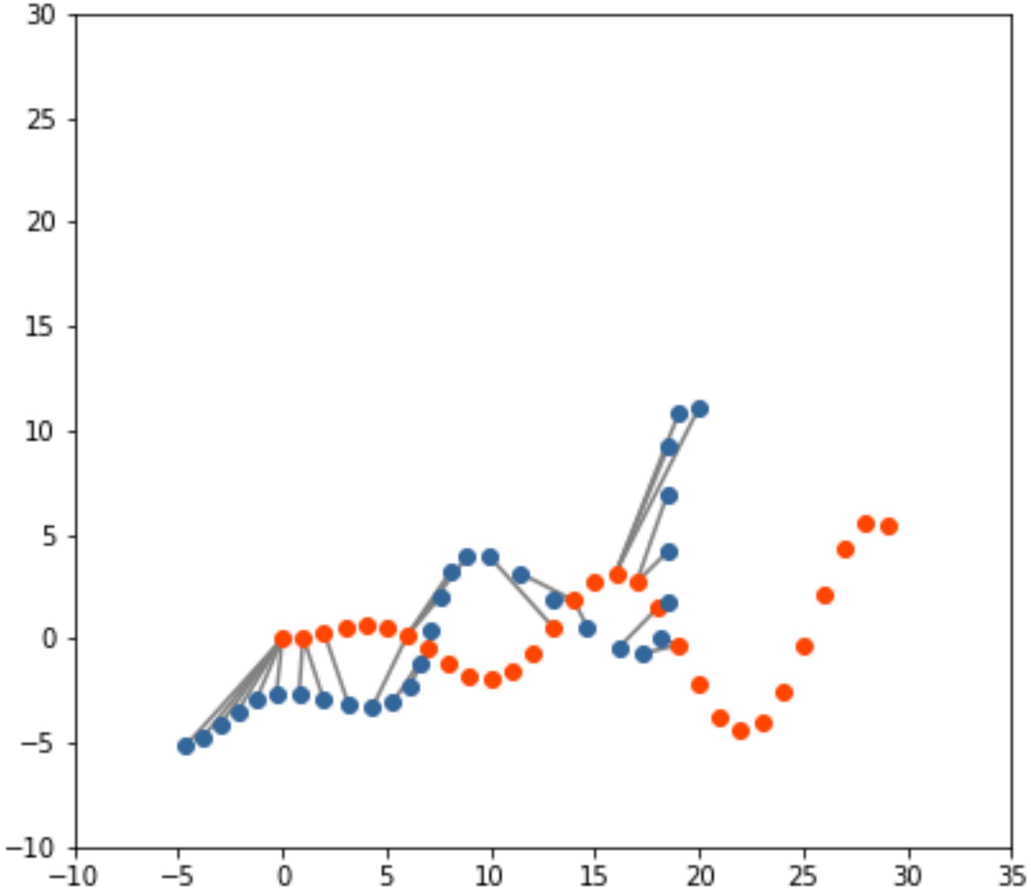


Image courtesy: Bogoslavskyi

2D Least Squares Example

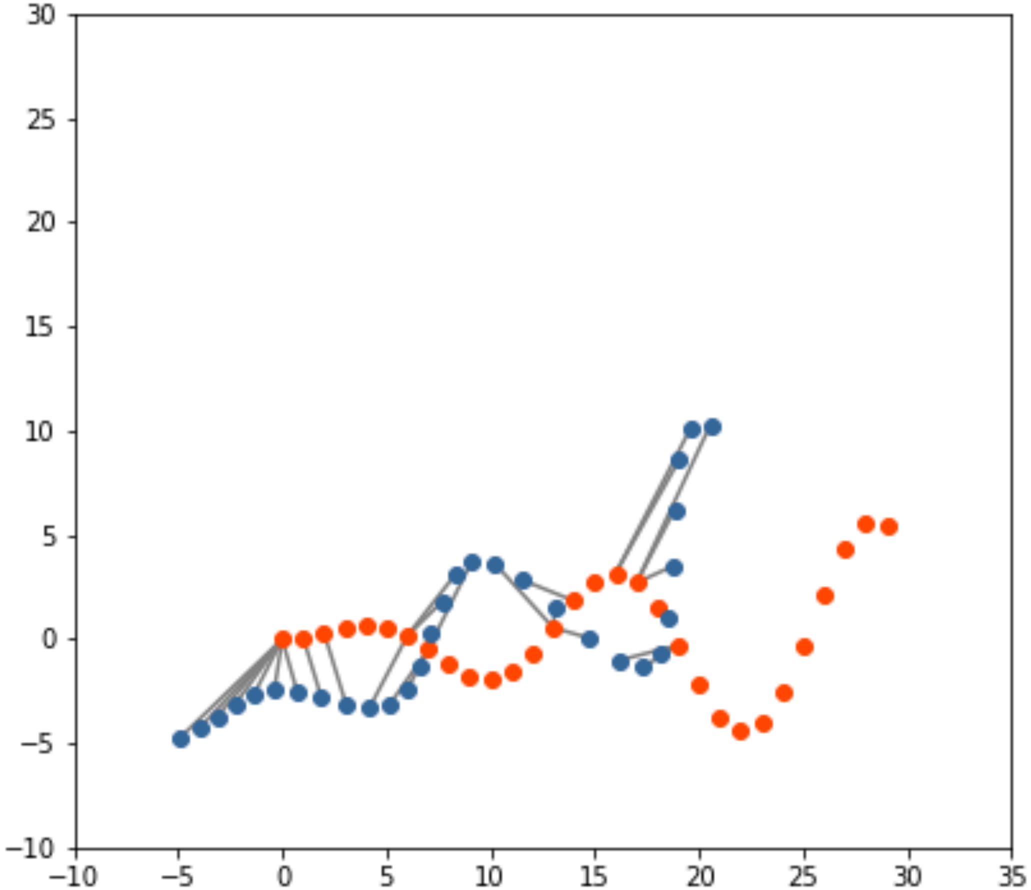


Image courtesy: Bogoslavskyi

2D Least Squares Example

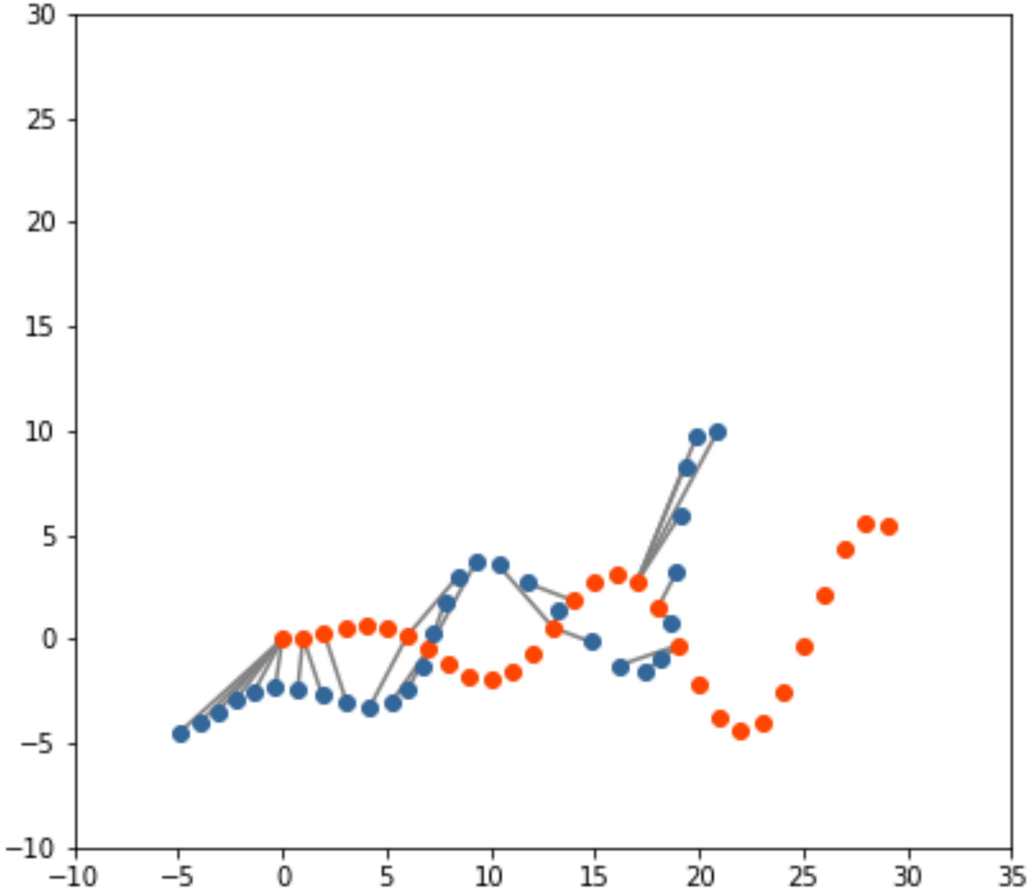


Image courtesy: Bogoslavskyi

2D Least Squares Example

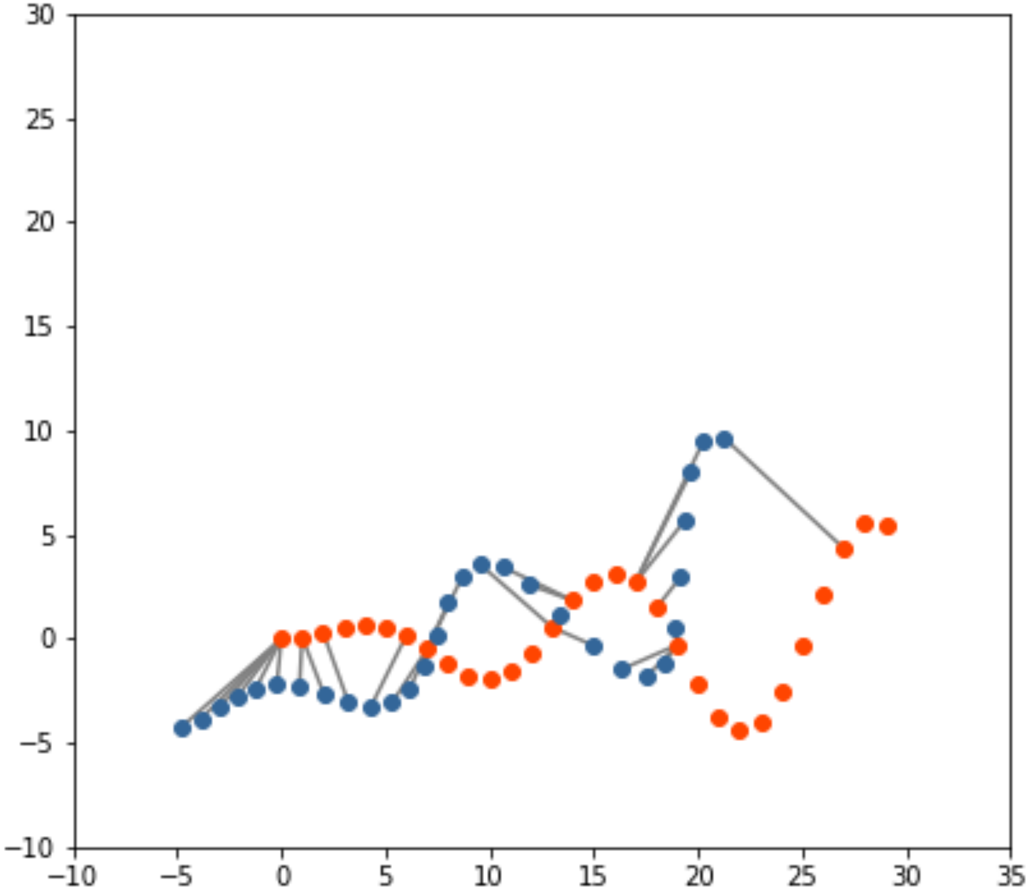


Image courtesy: Bogoslavskyi

2D Least Squares Example

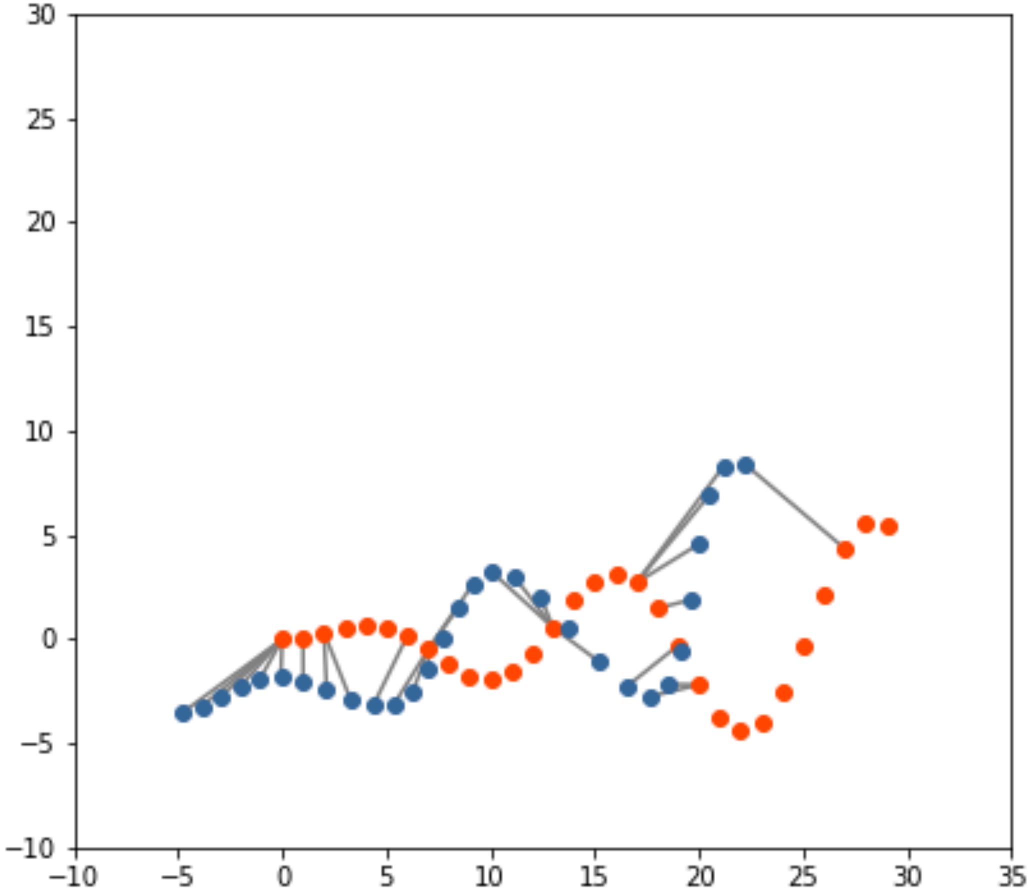


Image courtesy: Bogoslavskyi

2D Least Squares Example

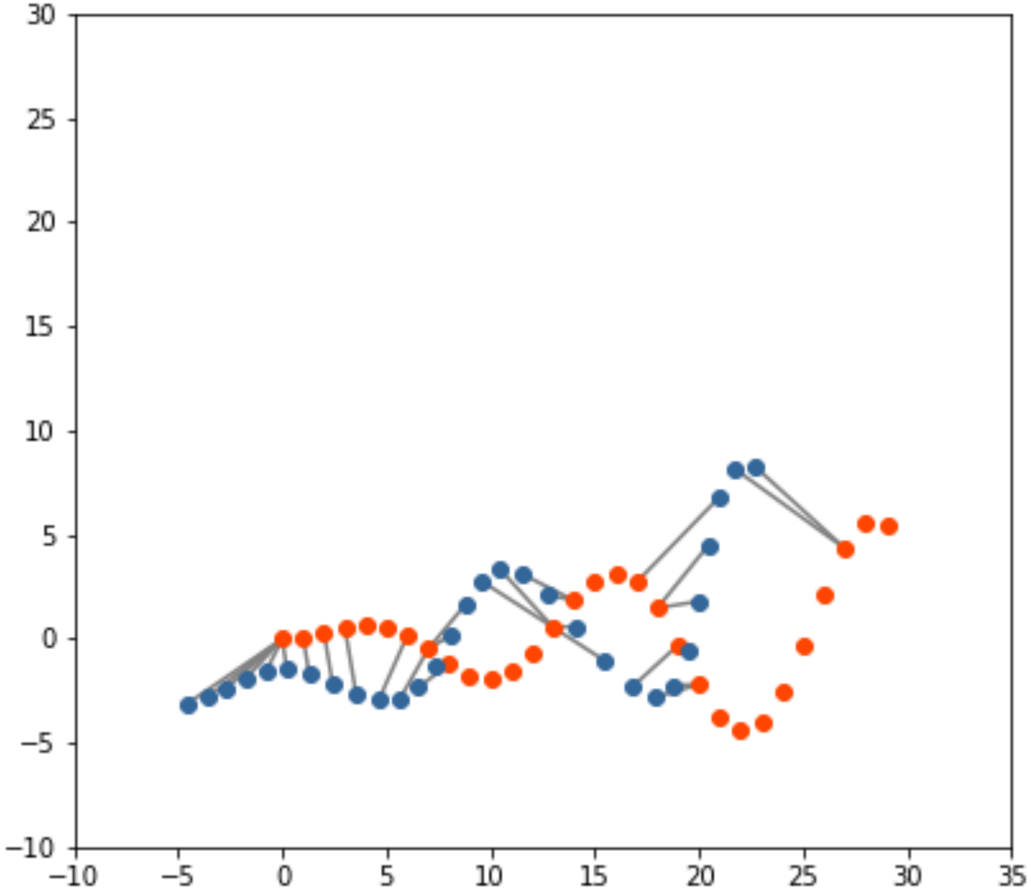


Image courtesy: Bogoslavskyi

2D Least Squares Example

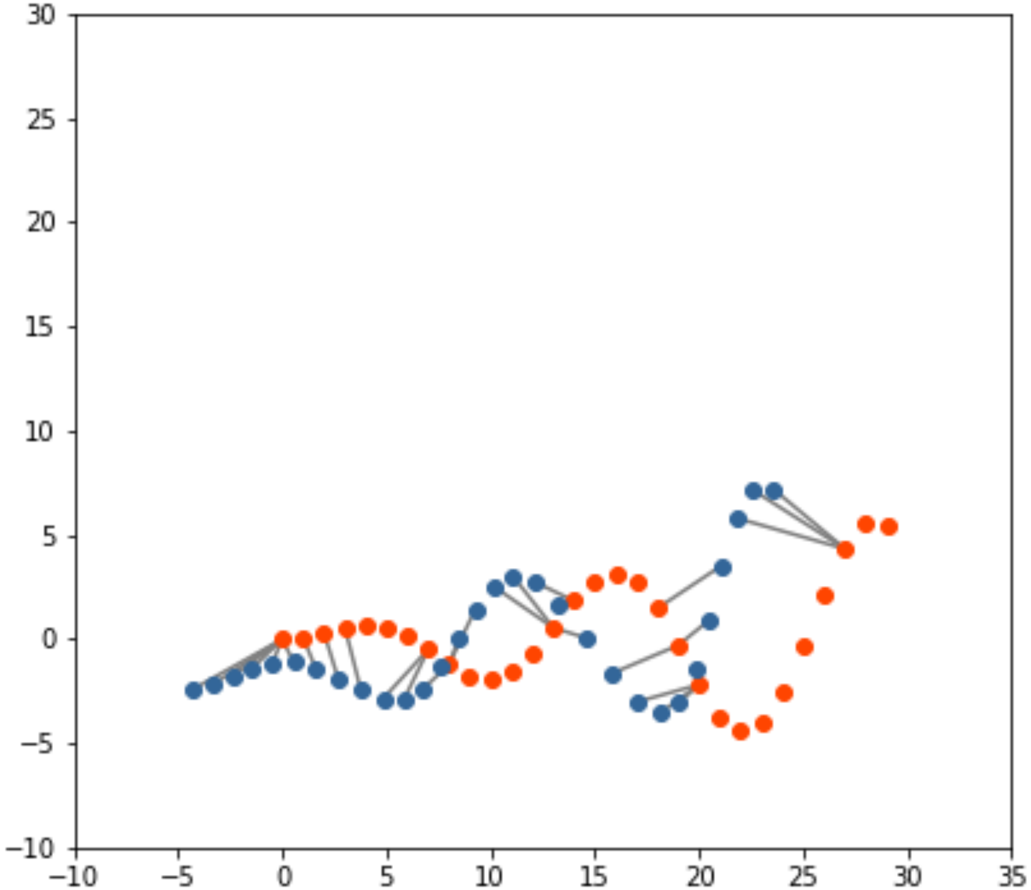


Image courtesy: Bogoslavskyi

2D Least Squares Example

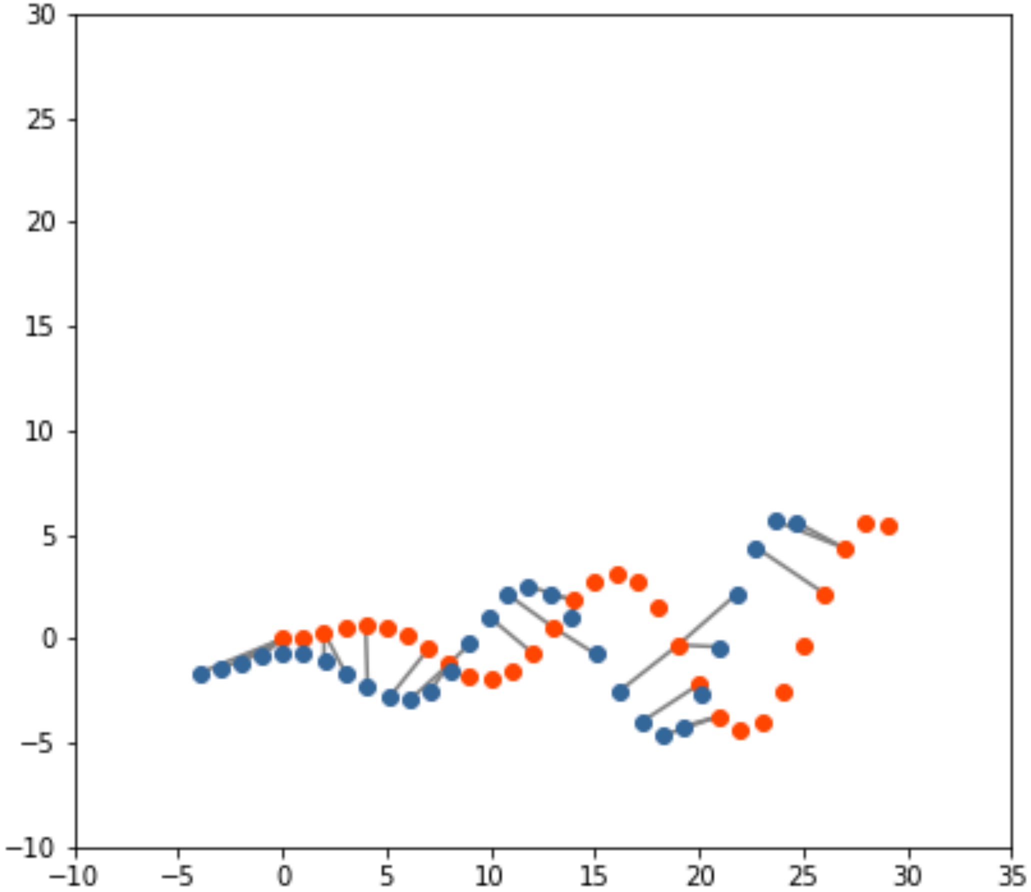


Image courtesy: Bogoslavskyi

2D Least Squares Example

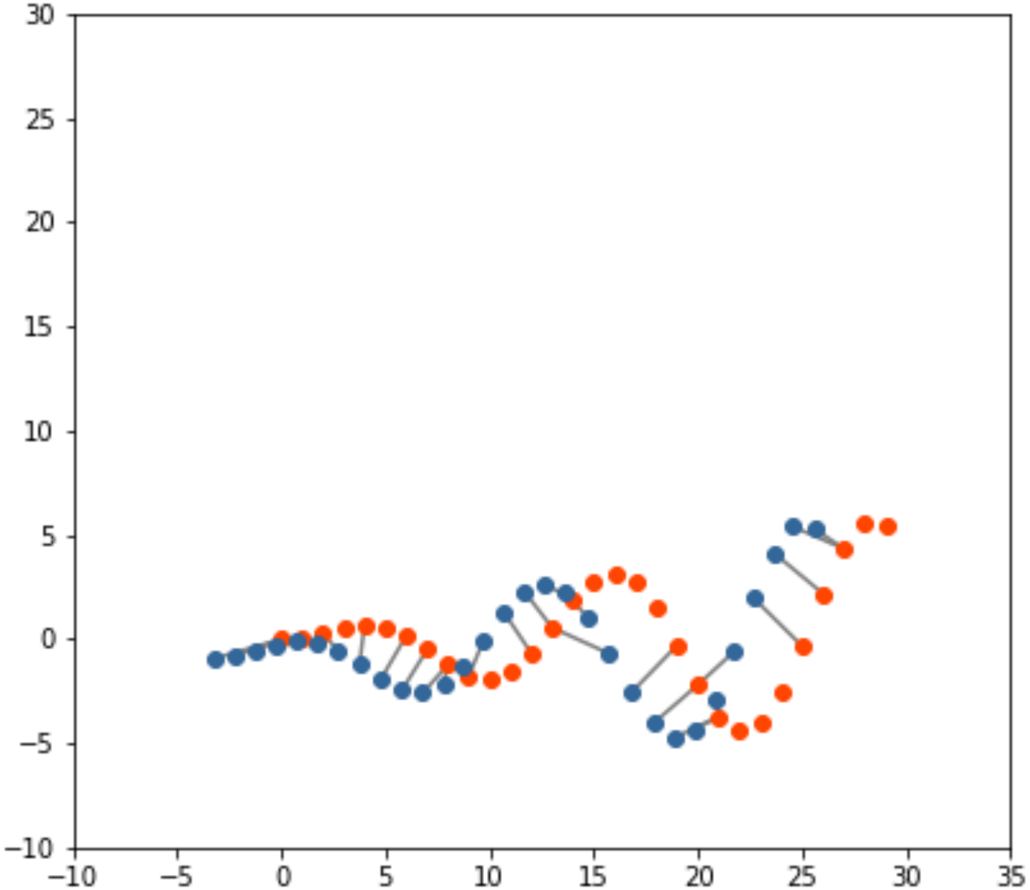


Image courtesy: Bogoslavskyi

2D Least Squares Example

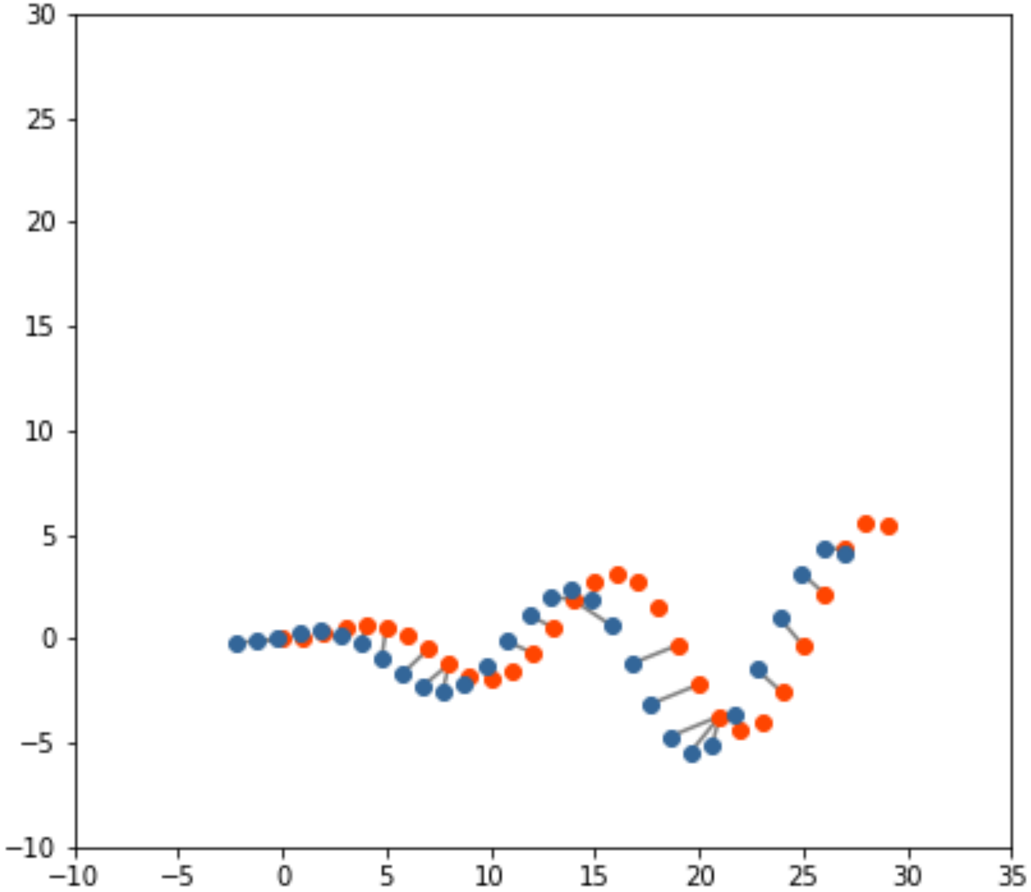


Image courtesy: Bogoslavskyi

2D Least Squares Example

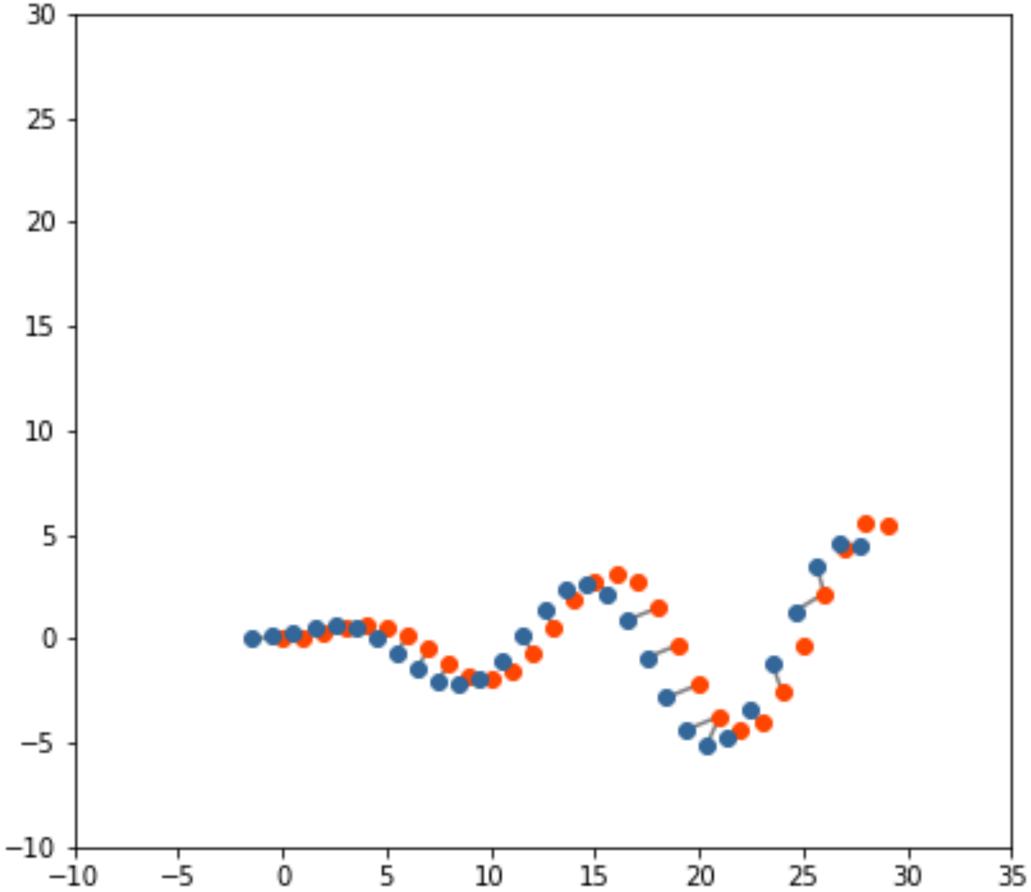


Image courtesy: Bogoslavskyi

2D Least Squares Example

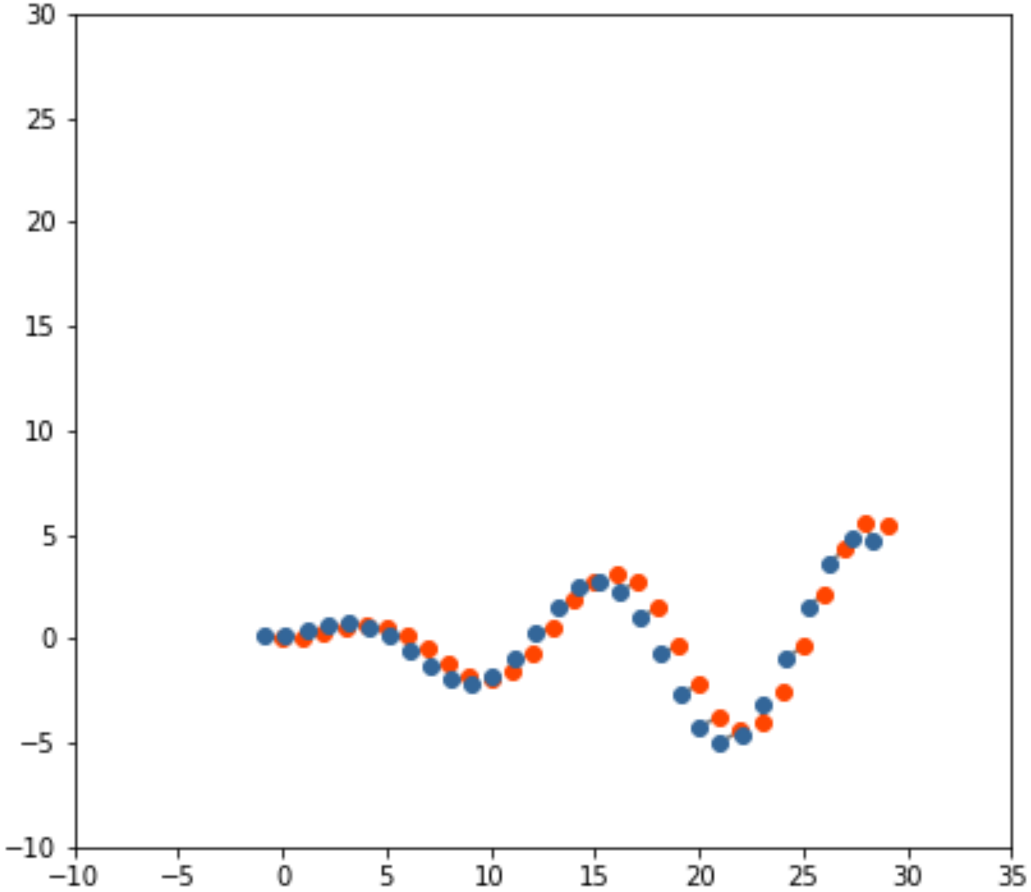


Image courtesy: Bogoslavskyi

2D Least Squares Example

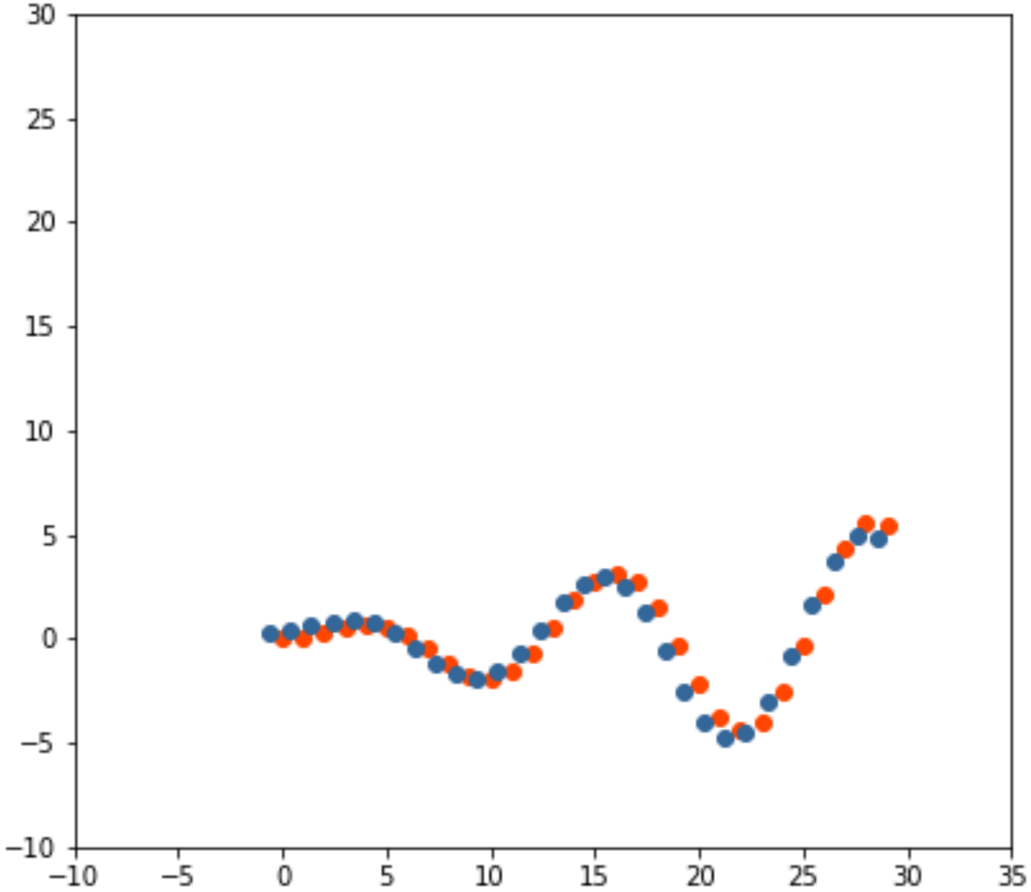


Image courtesy: Bogoslavskyi

2D Least Squares Example

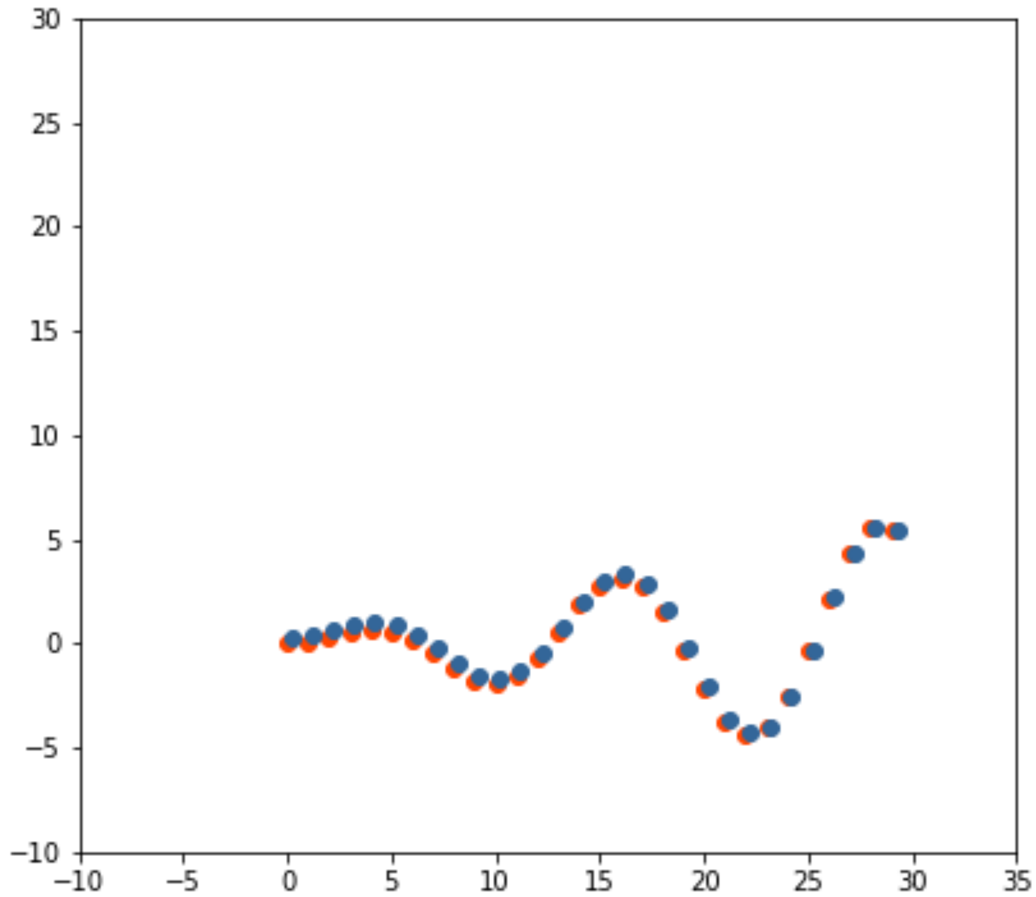


Image courtesy: Bogoslavskyi

2D Least Squares Example

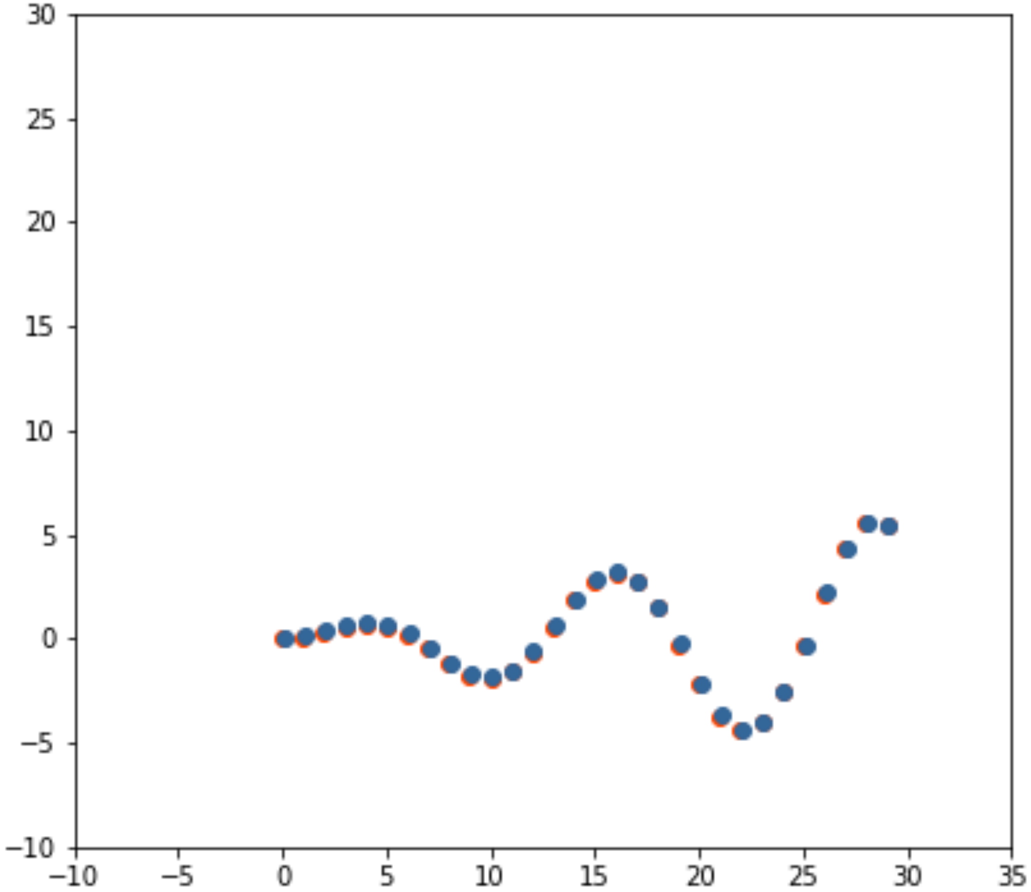


Image courtesy: Bogoslavskyi

2D Least Squares Example

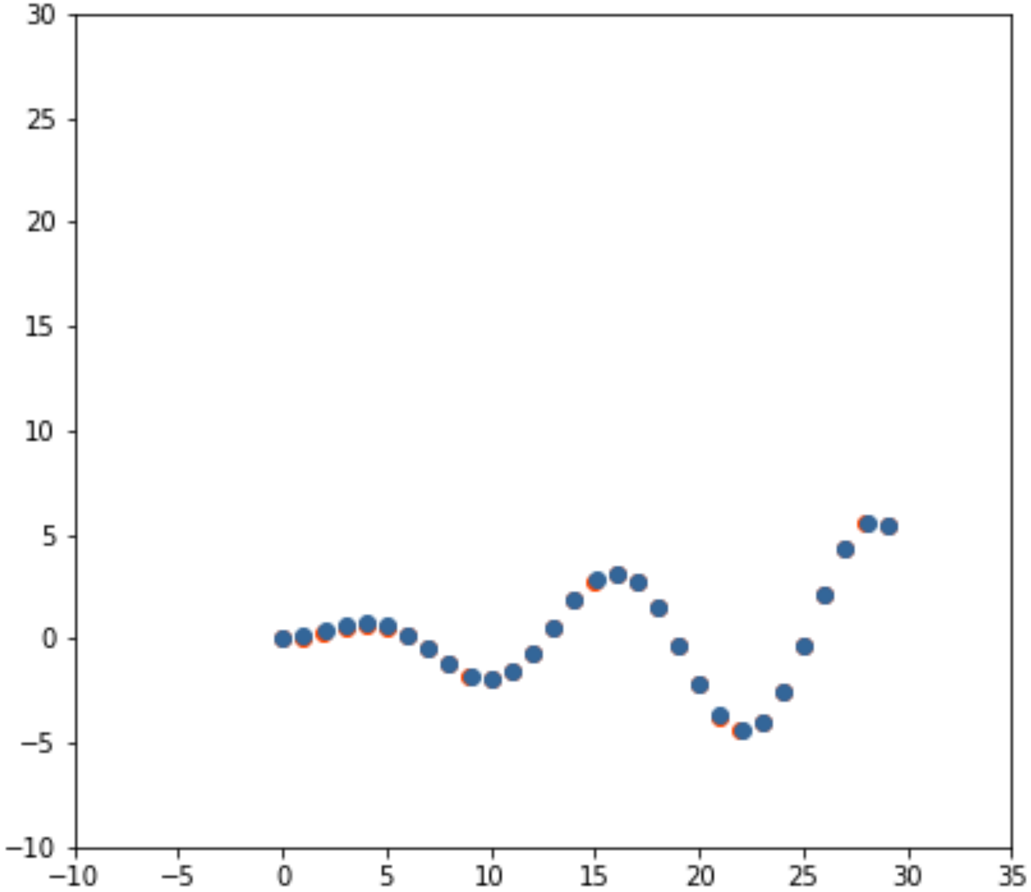


Image courtesy: Bogoslavskyi

2D Least Squares Example

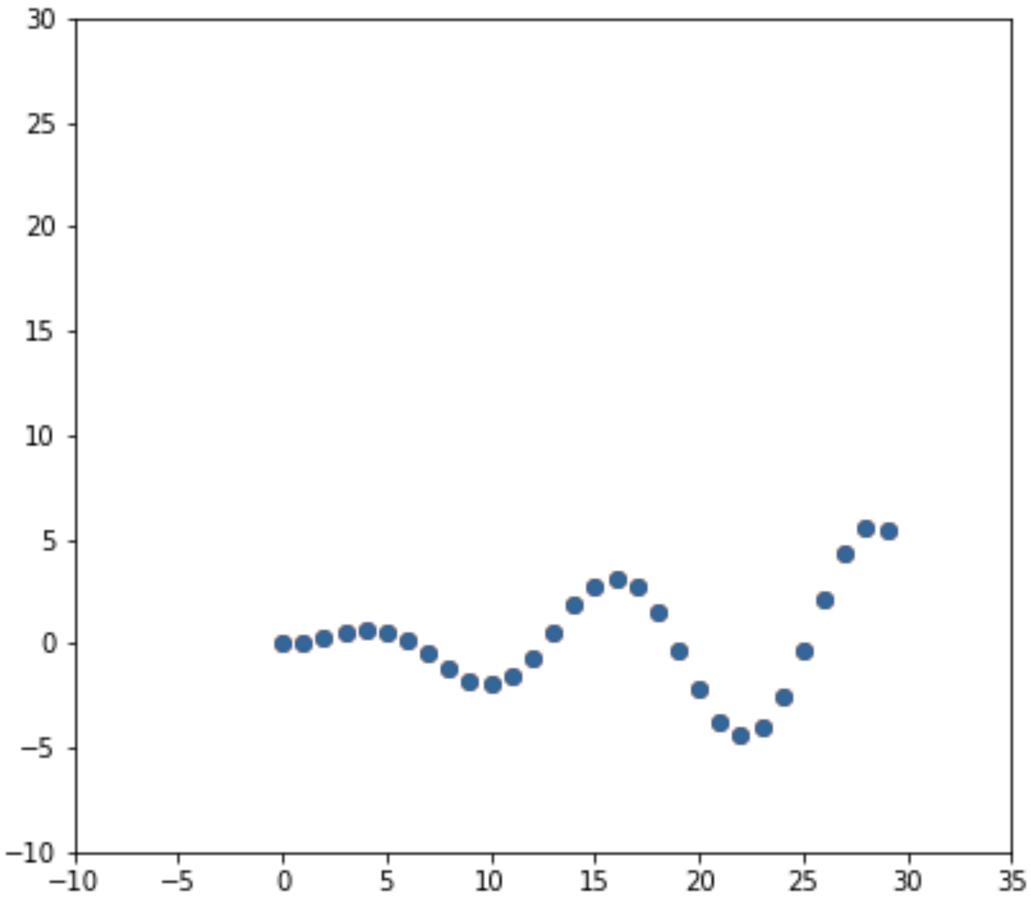


Image courtesy: Bogoslavskyi

2D Least Squares Example

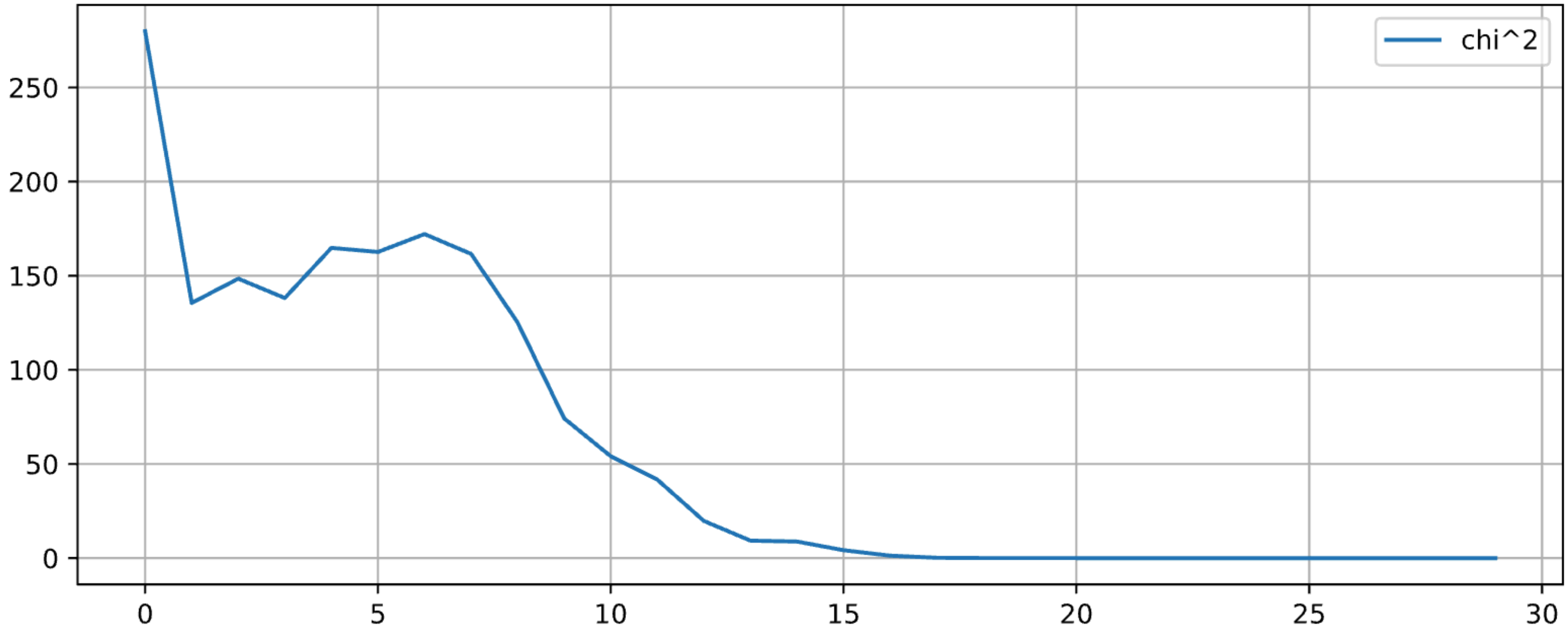
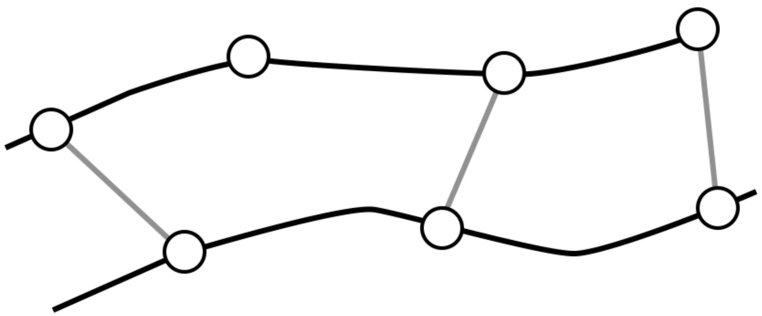


Image courtesy: Bogoslavskyi

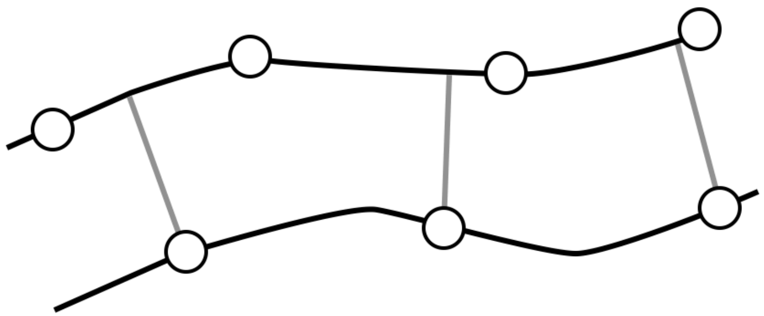
Least Squares Registration using Point-to-Plane Metric

Point-to-Plane Error

- Idea: still find the closest points
- Error = project point-to-point onto the direction of the normal, shot from the found point



point-to-point

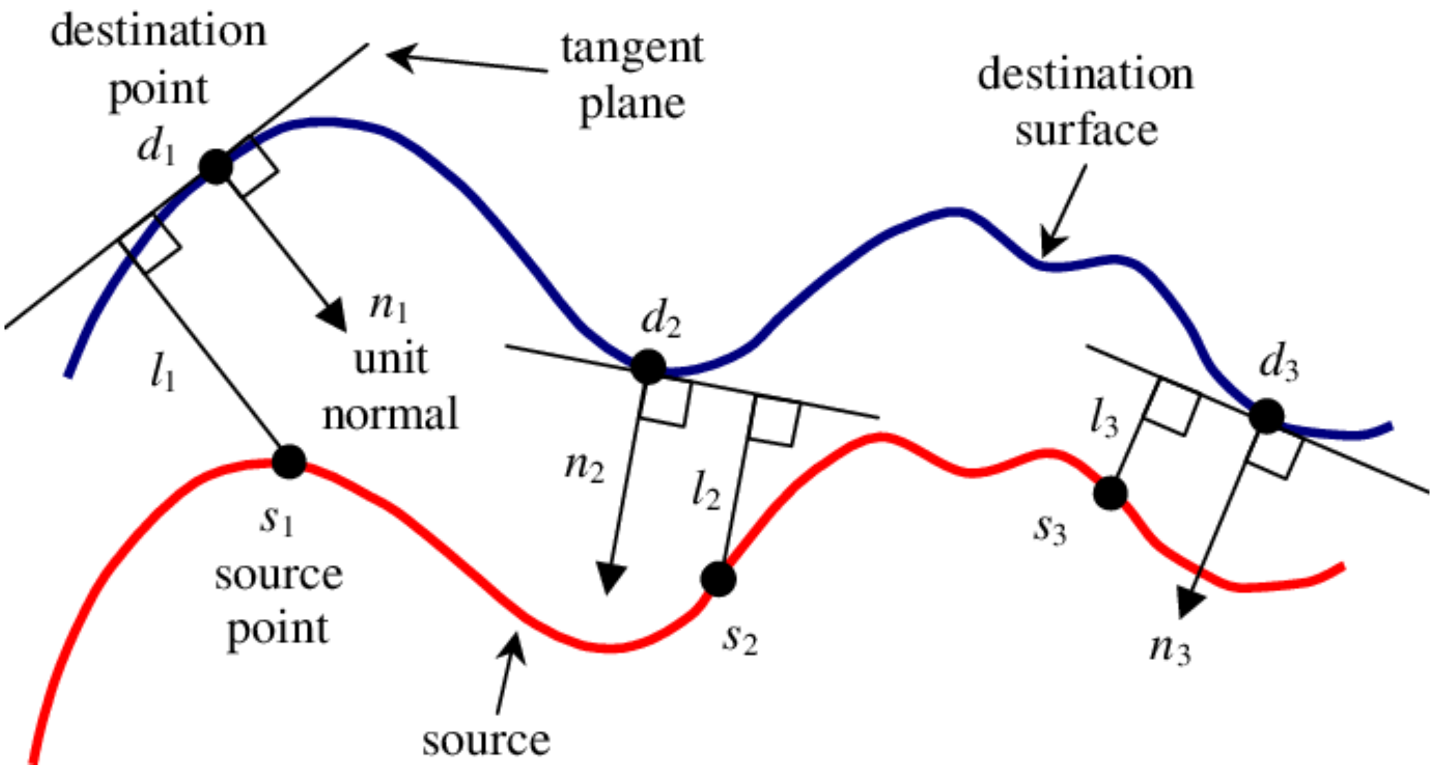


point-to-plane

Point-to-Plane Error

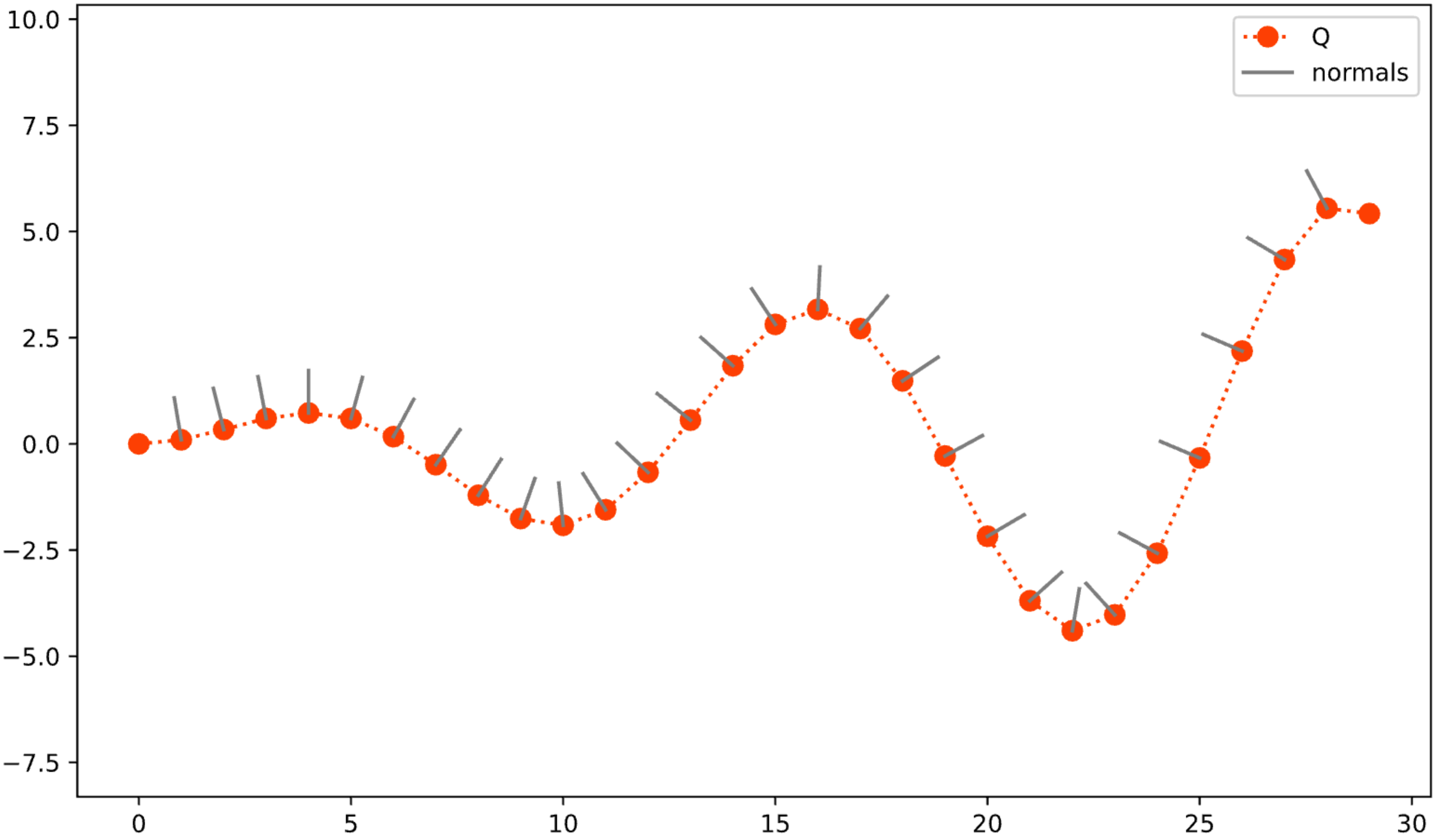
- Error = project point-to-point onto the direction of the normal, shot from the found point

$$\Phi(t_x, t_y, \theta) = \sum ||\boldsymbol{n}_n \cdot \boldsymbol{e}_n||^2$$



[Image courtesy: Low]

Simple Normals from Neighbors



Point-to-Plane Metric

- Objective

point-to-point
error vector

$$\Phi(t_x, t_y, \theta) = \sum ||\boldsymbol{n}_n \cdot \boldsymbol{e}_n||^2$$
$$= \sum (\boldsymbol{n}_n \cdot (R(\theta)\,\boldsymbol{x}_n + [t_x, t_y]^\top - \boldsymbol{y}_n))^2$$

Different Jacobian

- A changes objective leads to a different Jacobian

$$\Phi(t_x, t_y, \theta) = \sum \underbrace{(\boldsymbol{n}_n \cdot (R(\theta)\,\boldsymbol{x}_n + [t_x, t_y]^\top - \boldsymbol{y}_n))}_{\text{1D error}}{}^2$$

$$J_n(\boldsymbol{x}) = \left[\frac{\partial e_n}{\partial t_x}, \frac{\partial e_n}{\partial t_y}, \frac{\partial e_n}{\partial \theta}\right]$$

1D error

Different Jacobian

- A changes objective leads to a different Jacobian

$$\Phi(t_x, t_y, \theta) \quad = \quad \sum \left(\boldsymbol{n}_n \cdot (R(\theta)\, \boldsymbol{x}_n + [t_x, t_y]^\top - \boldsymbol{y}_n)\right)^2$$

1D error

$$J_n(\boldsymbol{x}) \quad = \quad \left[\frac{\partial e_n}{\partial t_x}, \frac{\partial e_n}{\partial t_y}, \frac{\partial e_n}{\partial \theta}\right]$$

$$J_n(\boldsymbol{x}) \quad = \quad \left[\boldsymbol{n}_x, \boldsymbol{n}_y, \boldsymbol{n}_x(-t_x \sin(\theta) - t_y \cos(\theta)) + \boldsymbol{n}_x(t_x \cos(\theta) - t_y \sin(\theta))\right]$$

2D Point-to-Plane Example

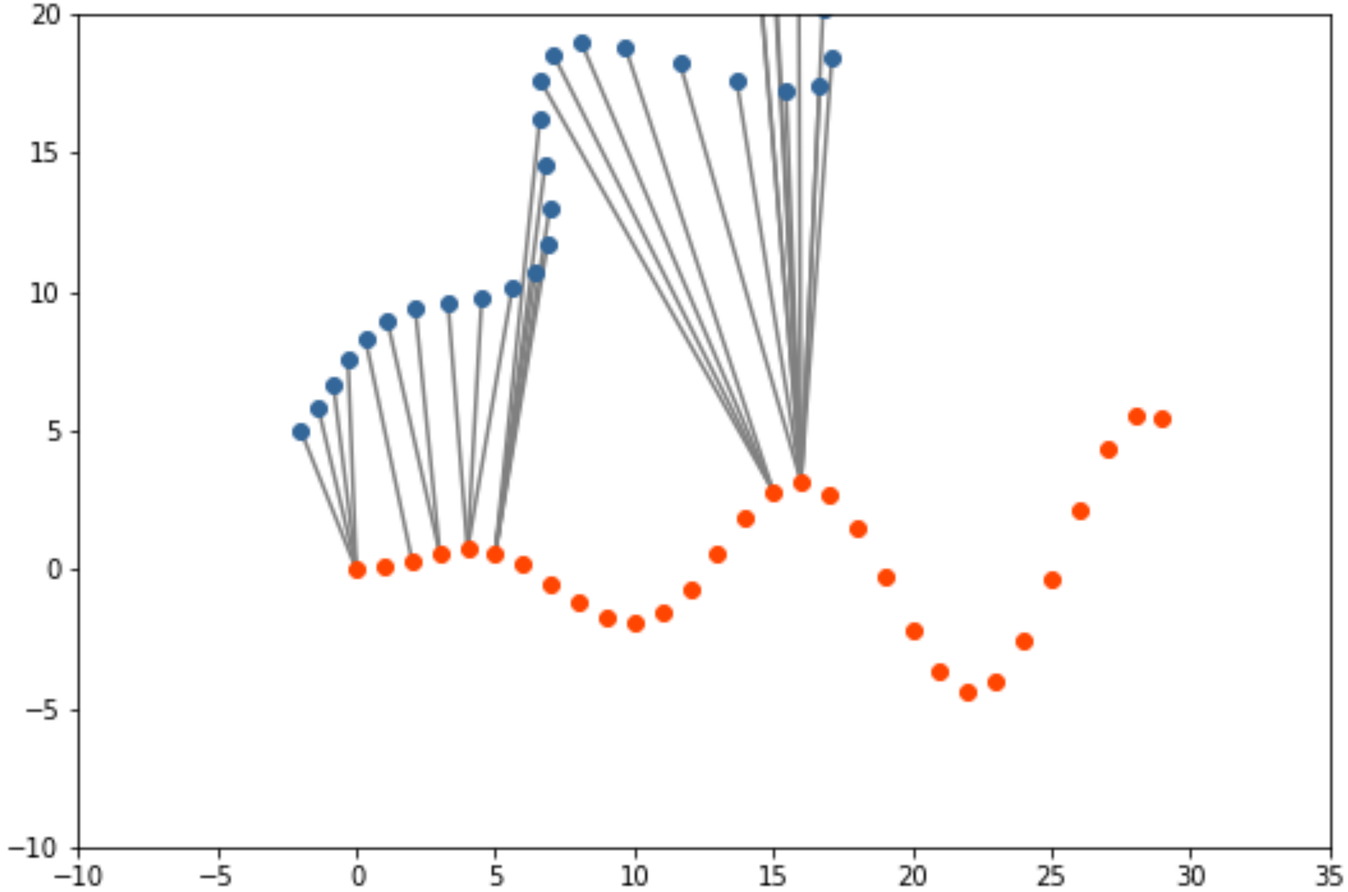


Image courtesy: Bogoslavskyi

2D Point-to-Plane Example

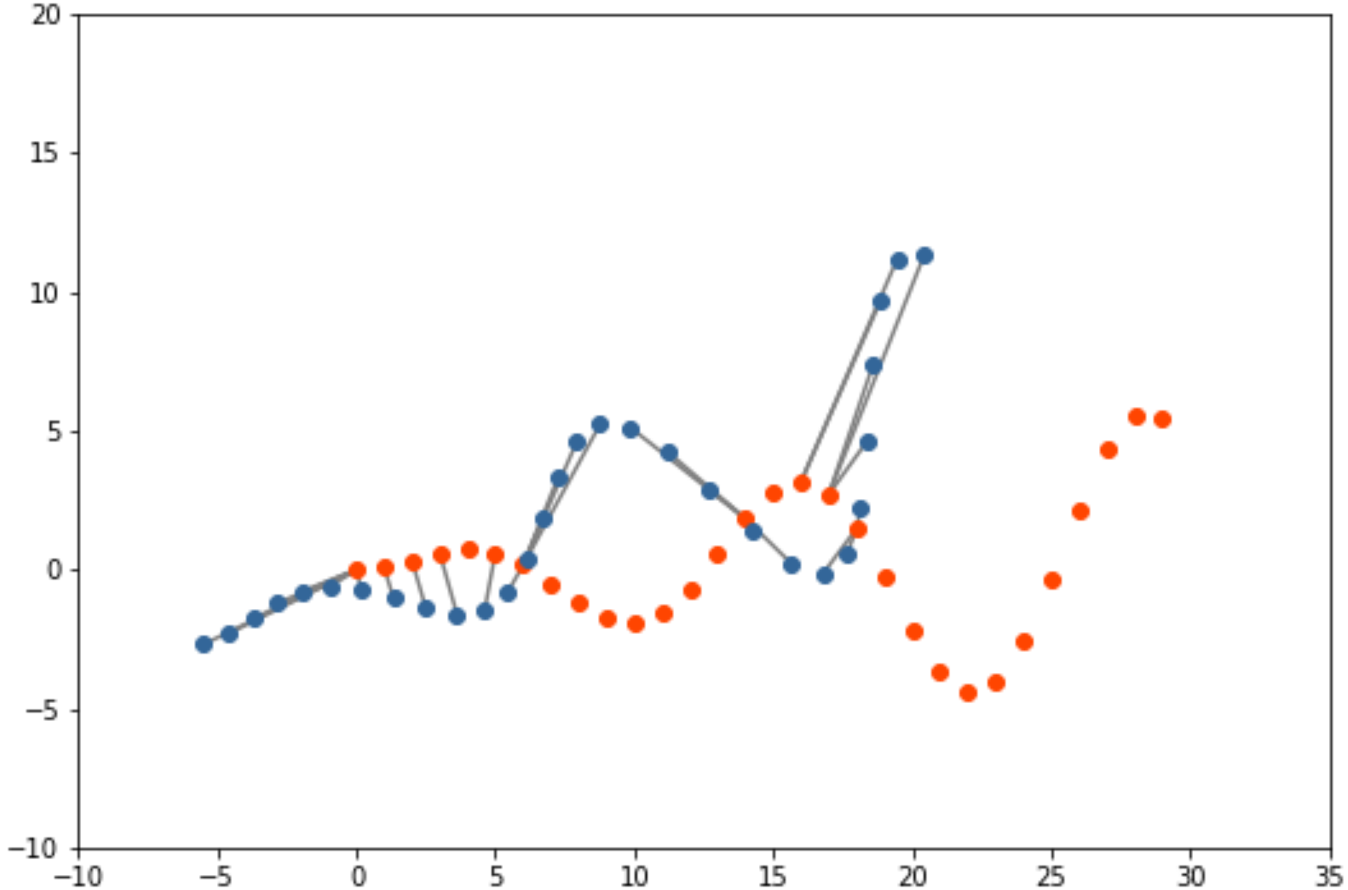


Image courtesy: Bogoslavskyi

2D Point-to-Plane Example

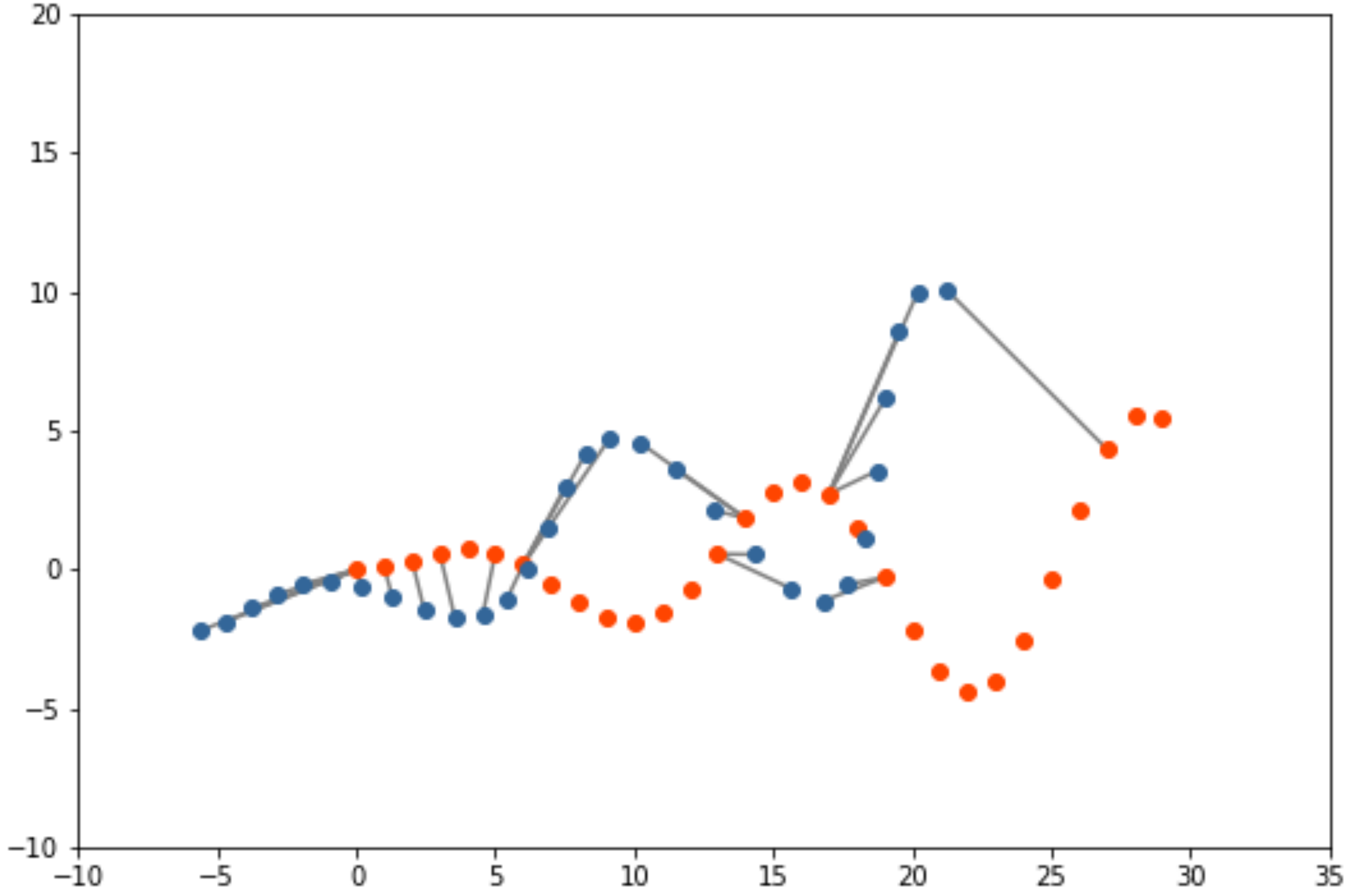


Image courtesy: Bogoslavskyi

2D Point-to-Plane Example

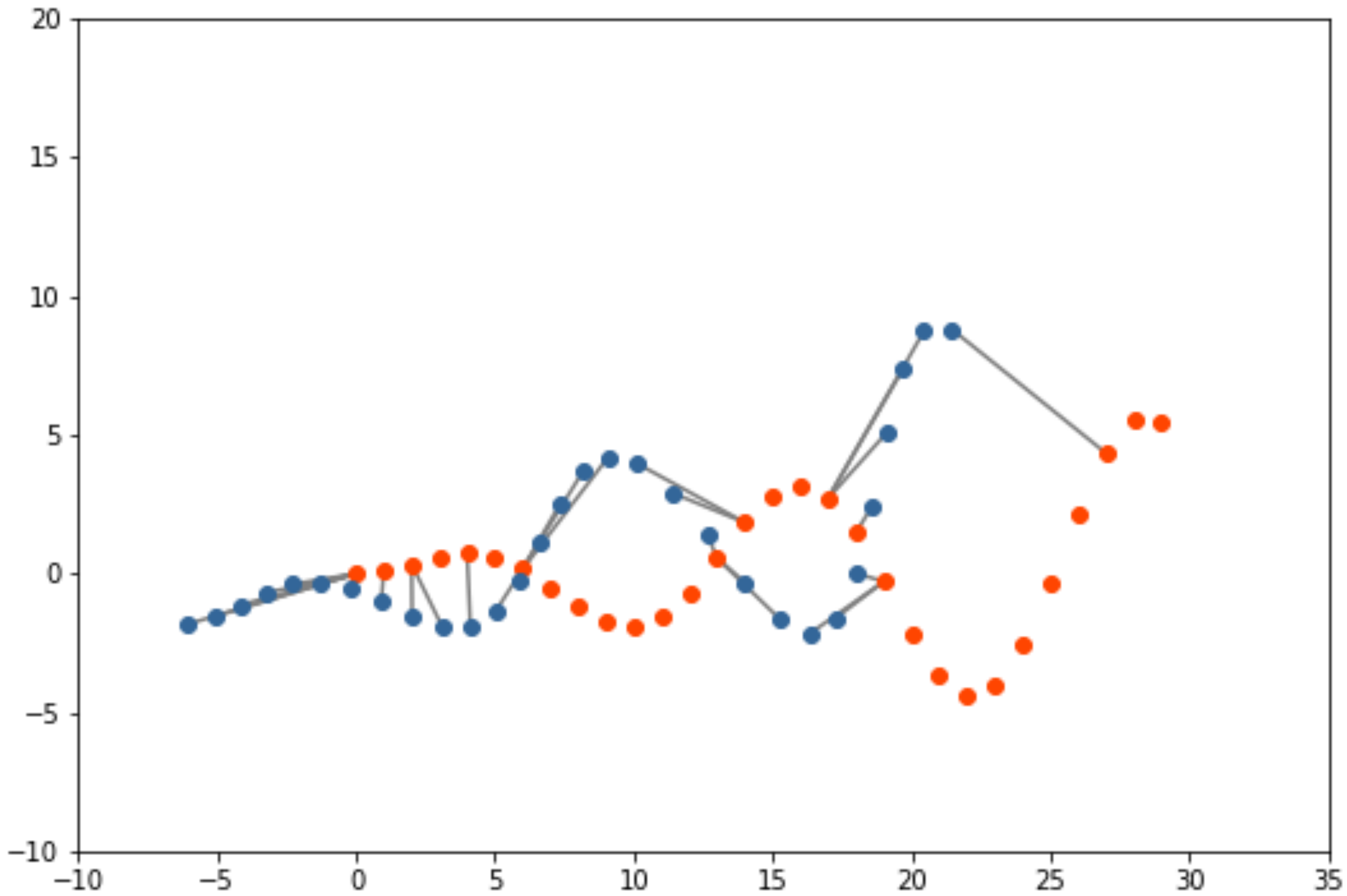


Image courtesy: Bogoslavskyi

2D Point-to-Plane Example

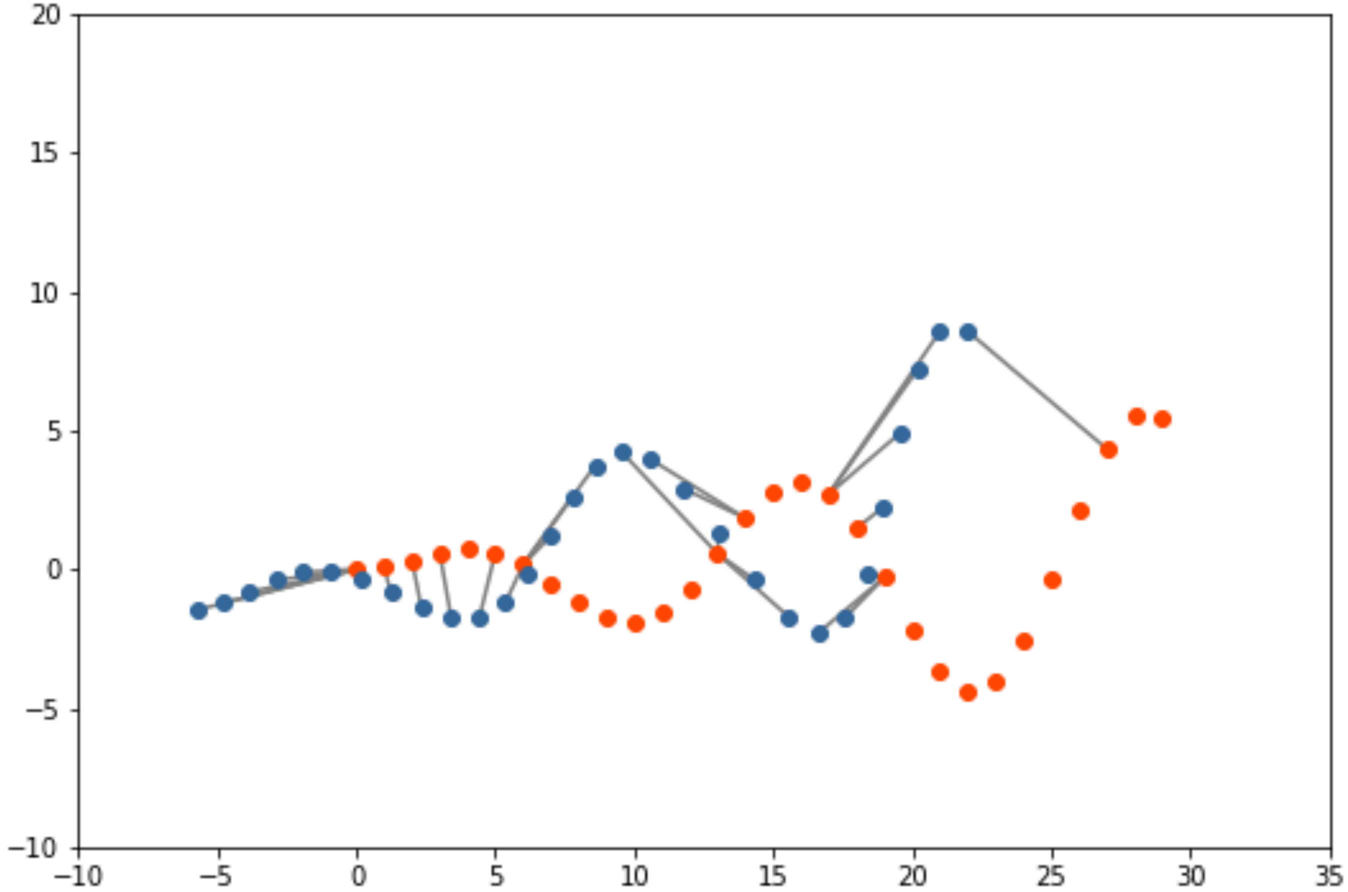


Image courtesy: Bogoslavskyi

2D Point-to-Plane Example

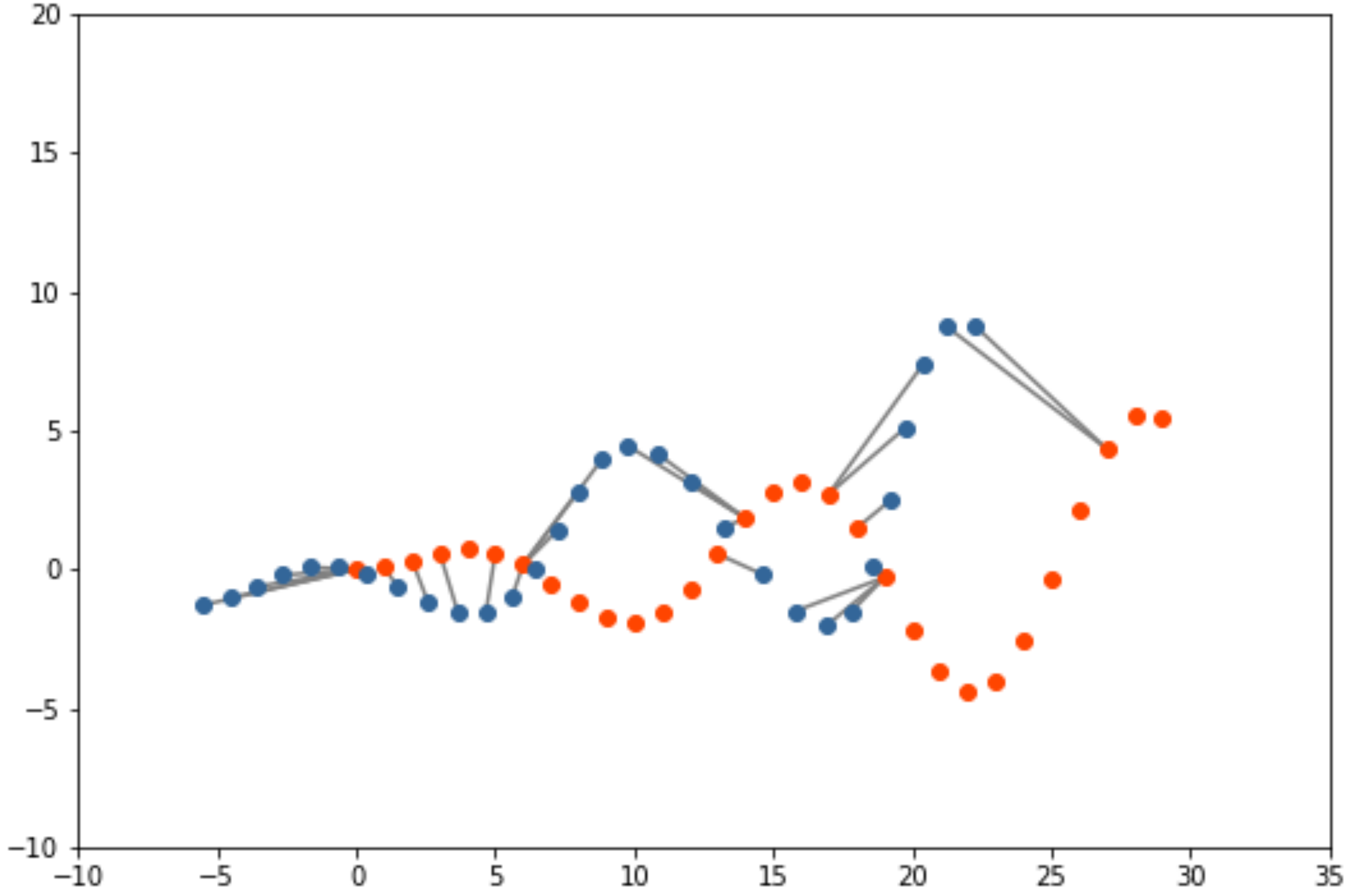


Image courtesy: Bogoslavskyi

2D Point-to-Plane Example

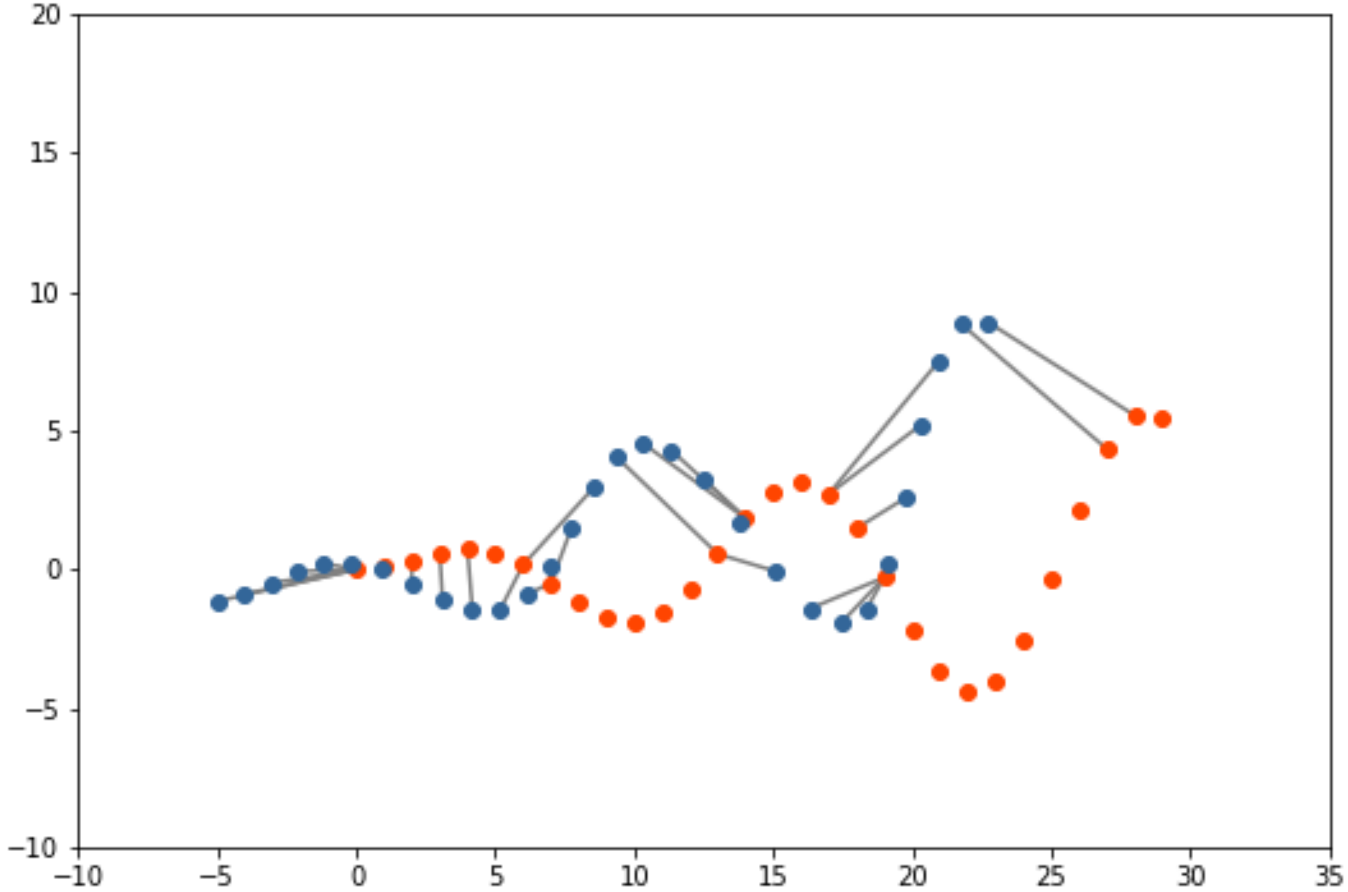


Image courtesy: Bogoslavskyi

2D Point-to-Plane Example

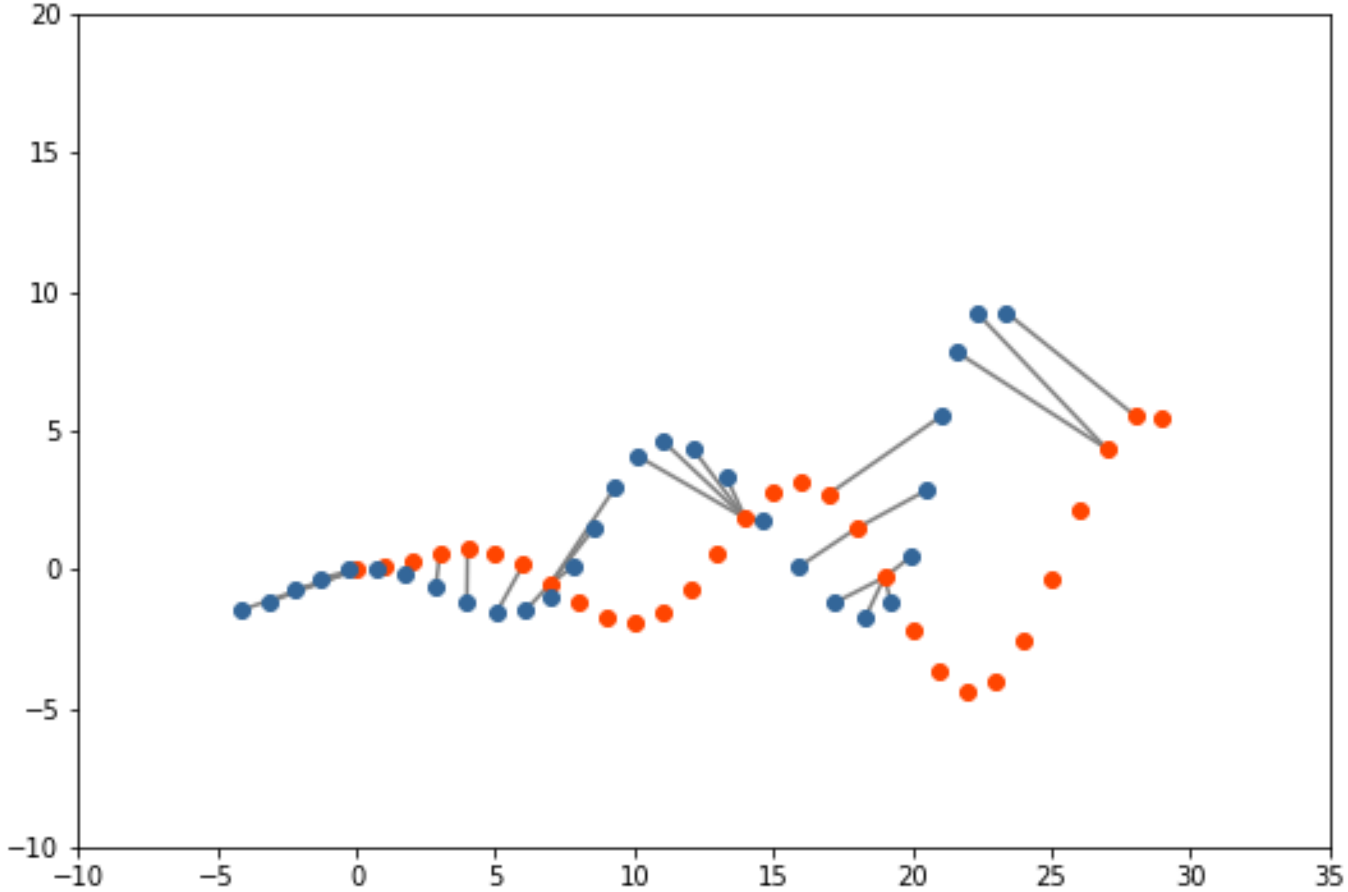


Image courtesy: Bogoslavskyi

2D Point-to-Plane Example

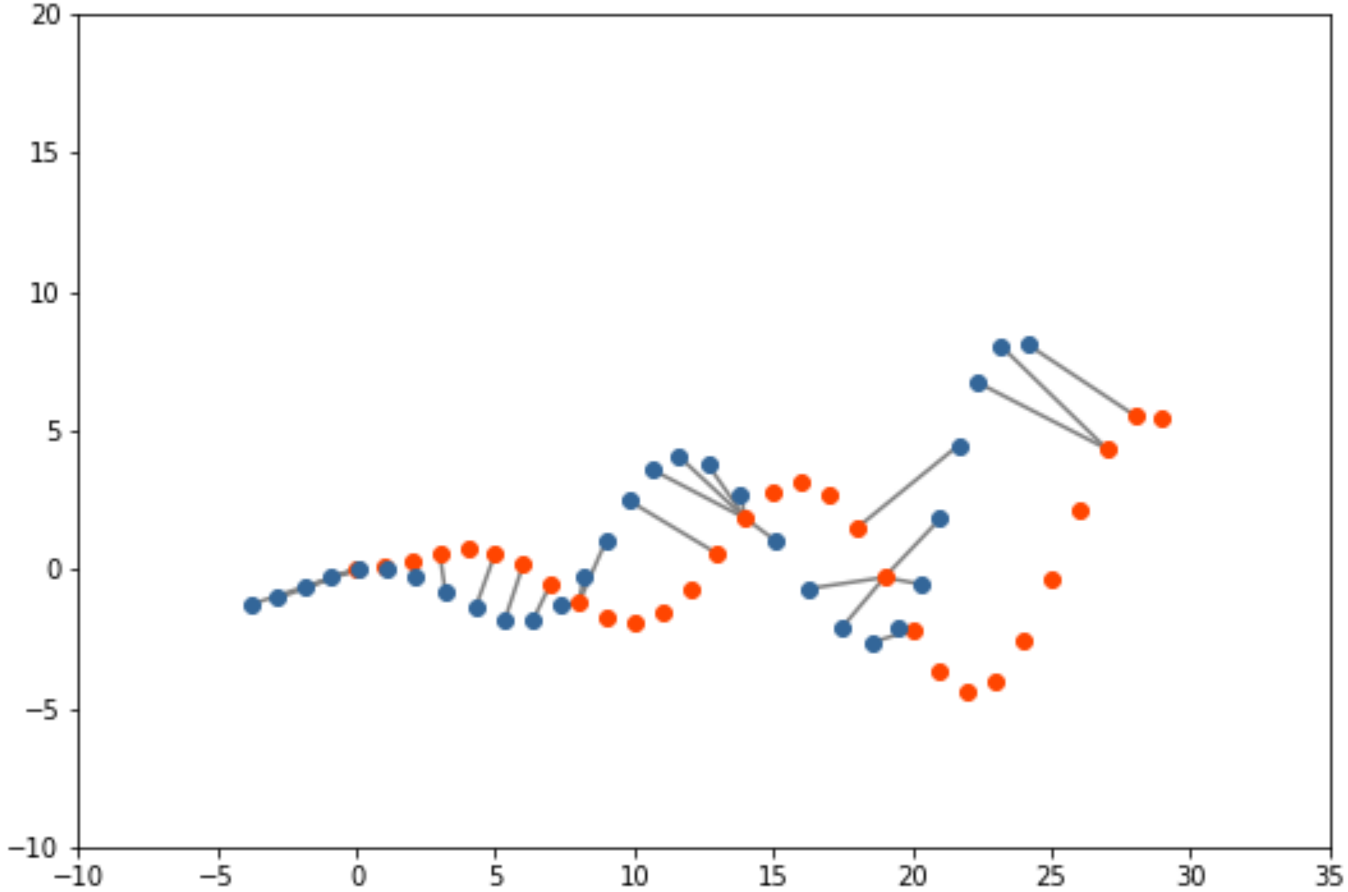


Image courtesy: Bogoslavskyi

2D Point-to-Plane Example

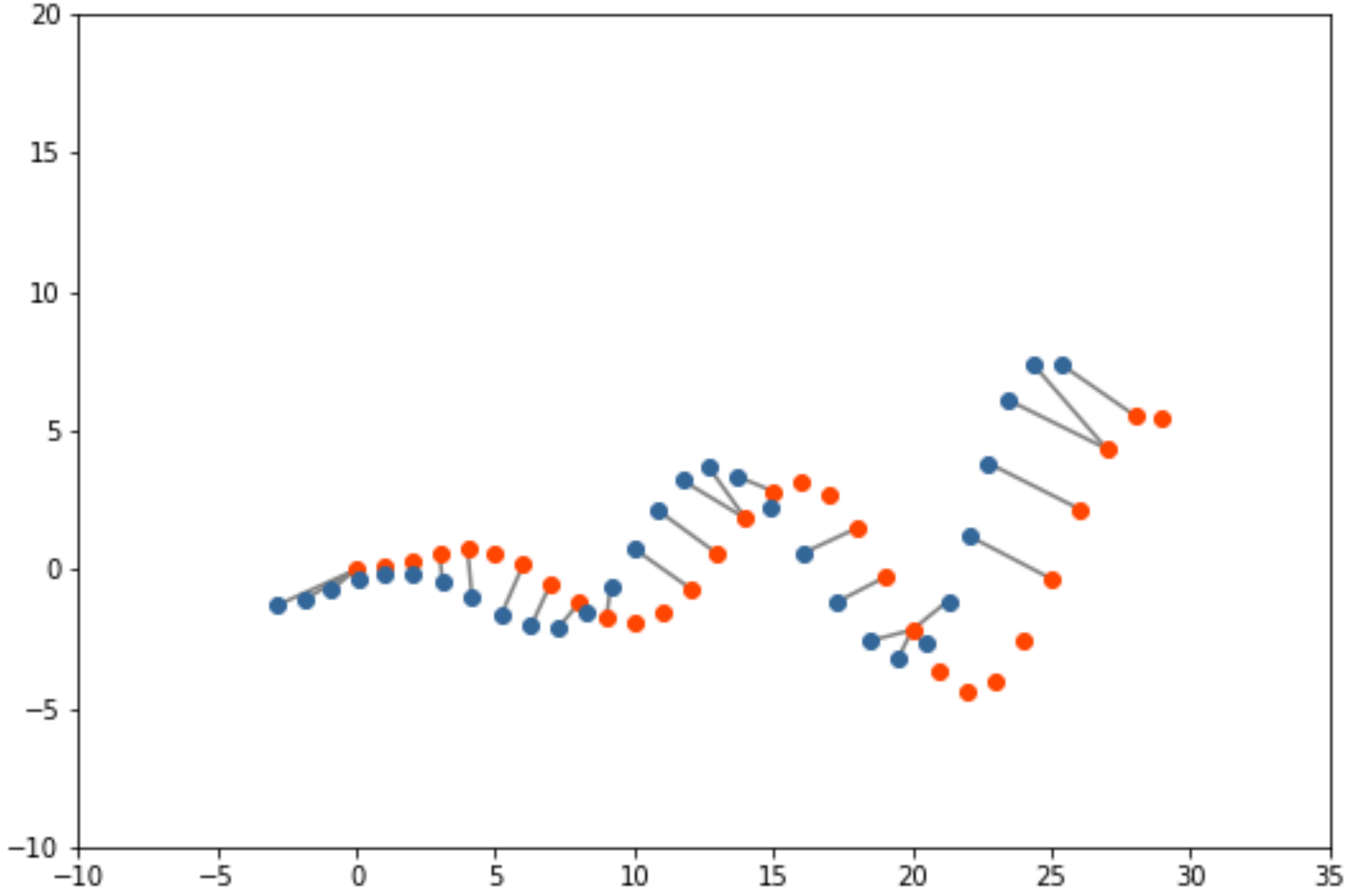


Image courtesy: Bogoslavskyi

2D Point-to-Plane Example

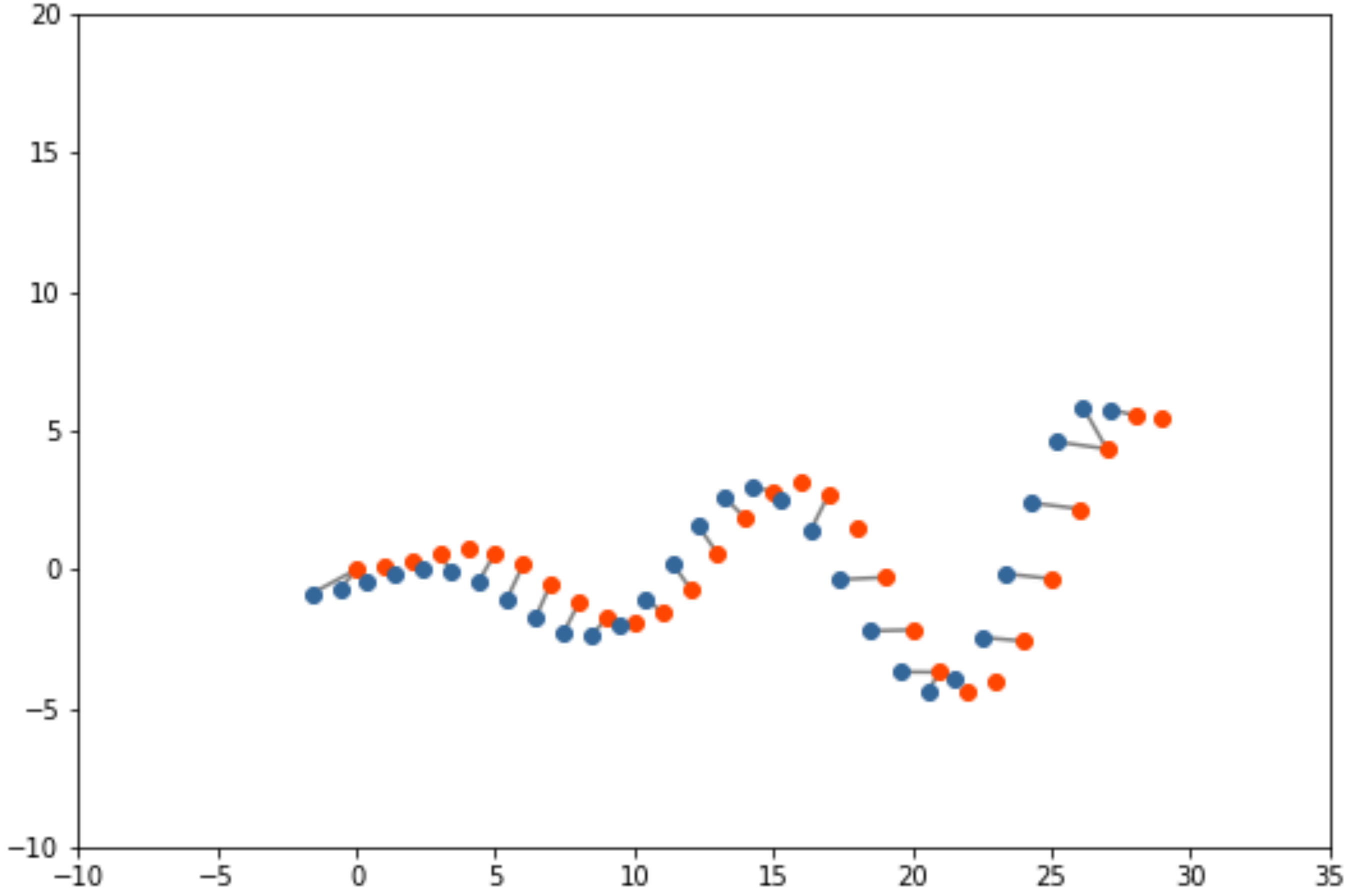


Image courtesy: Bogoslavskyi

2D Point-to-Plane Example

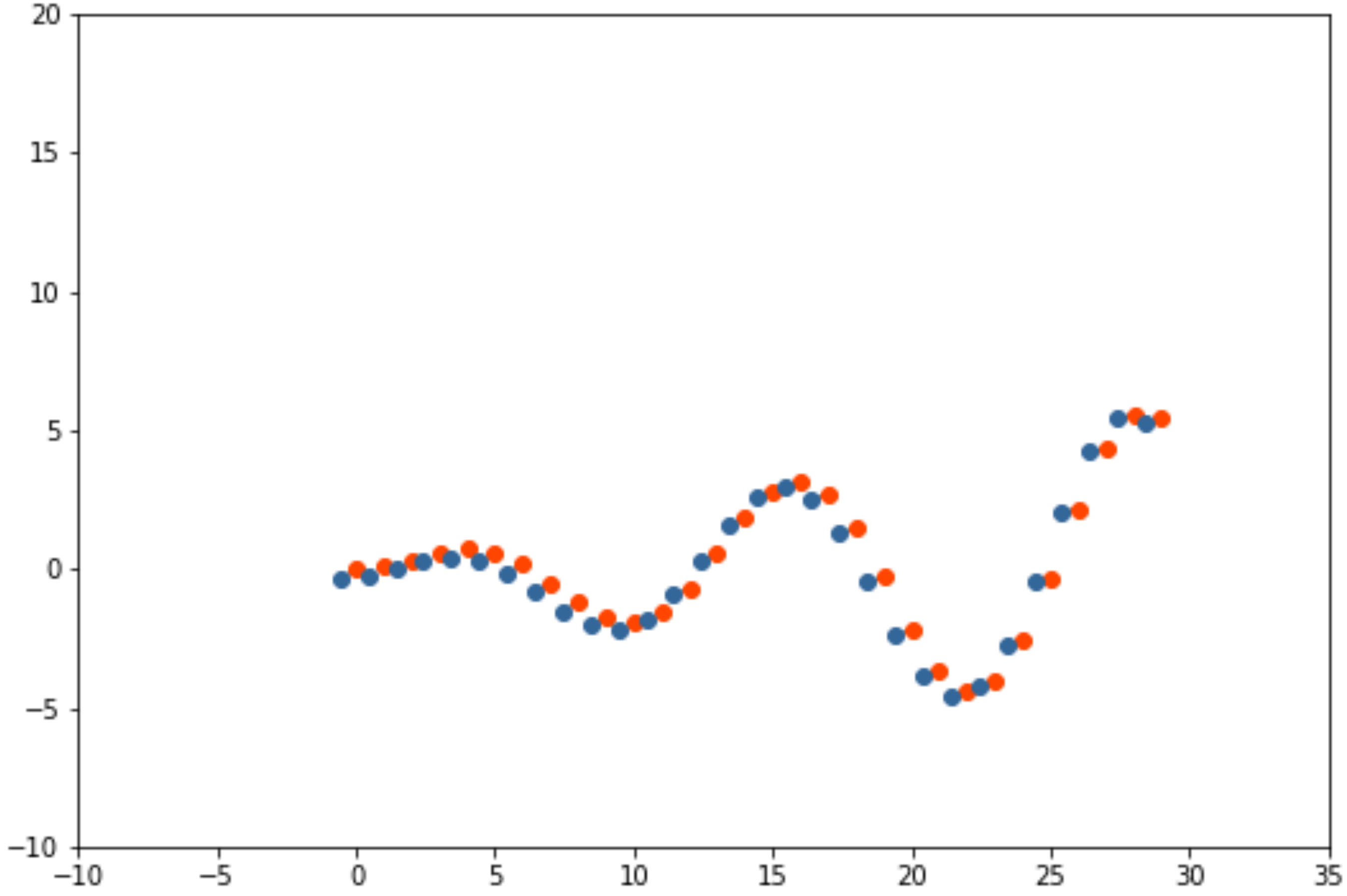


Image courtesy: Bogoslavskyi

2D Point-to-Plane Example

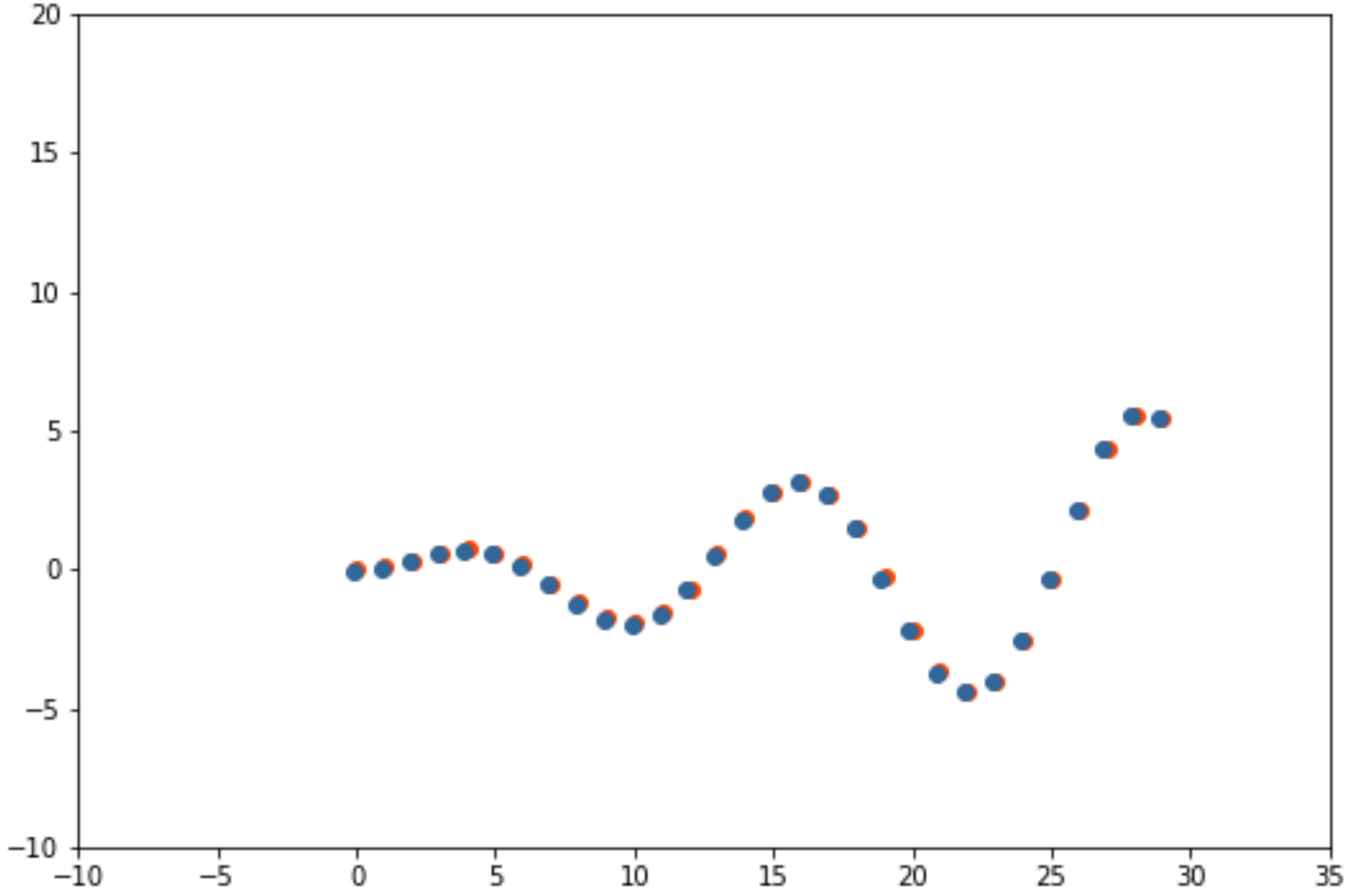


Image courtesy: Bogoslavskyi

2D Point-to-Plane Example

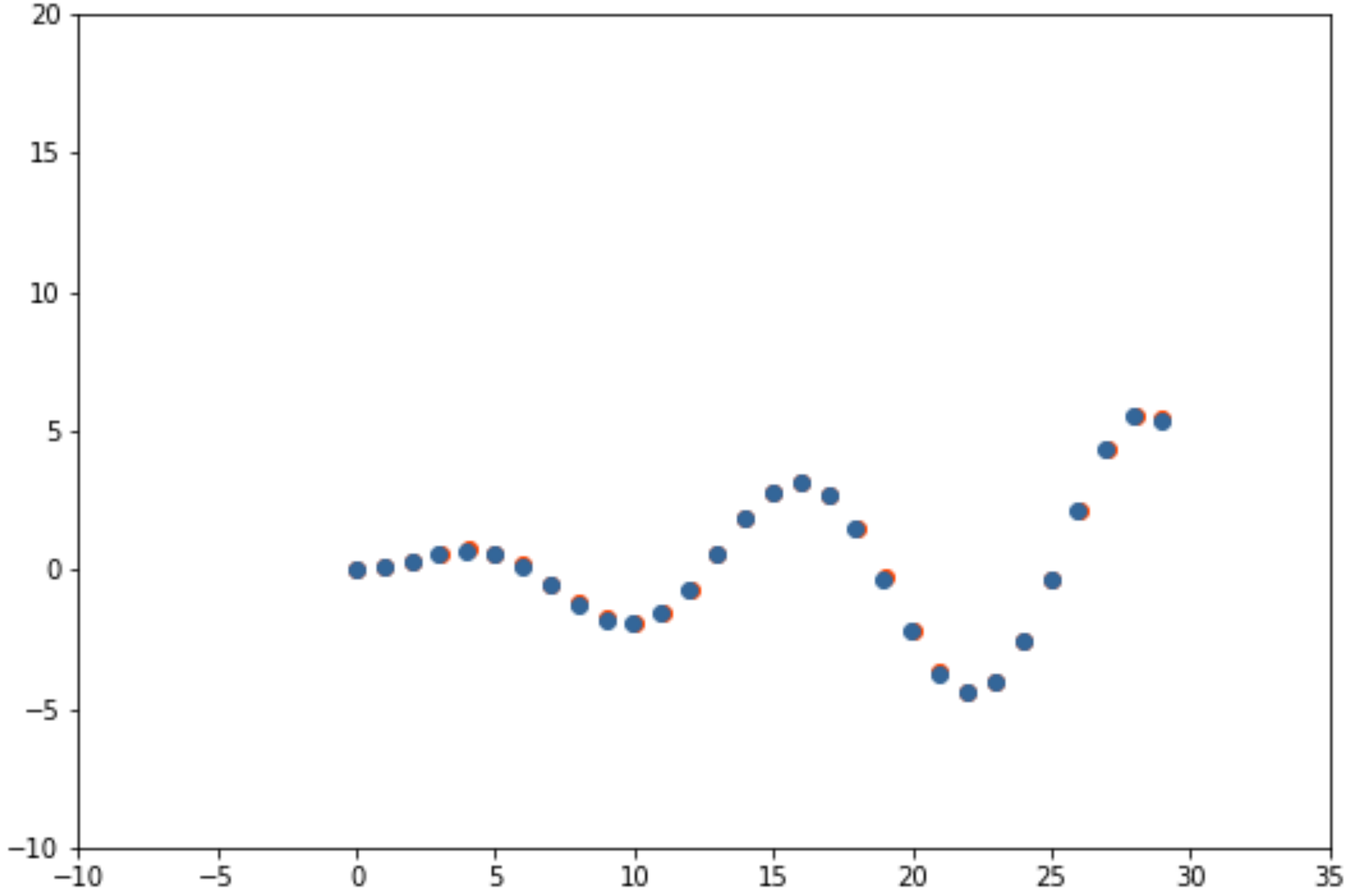


Image courtesy: Bogoslavskyi

2D Point-to-Plane Example

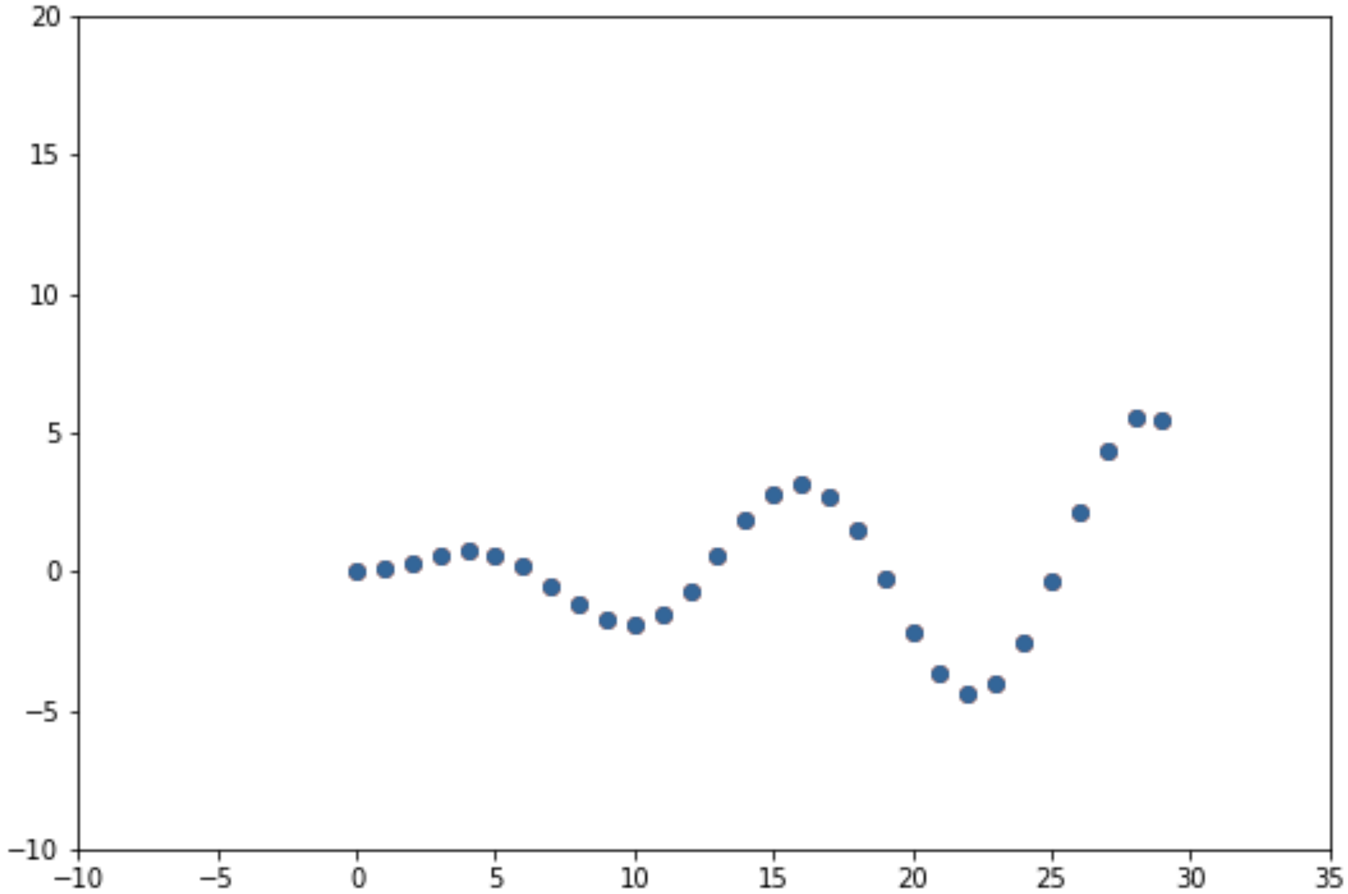


Image courtesy: Bogoslavskyi

2D Point-to-Plane Example

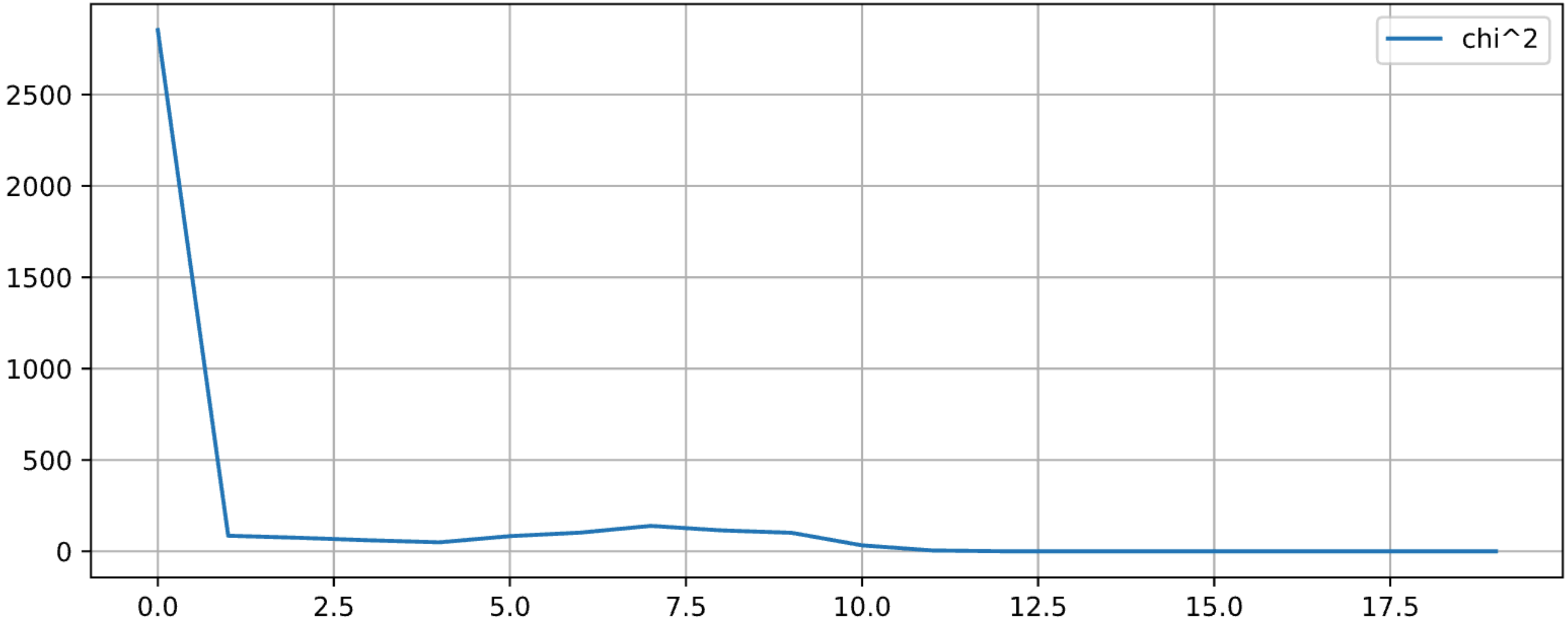
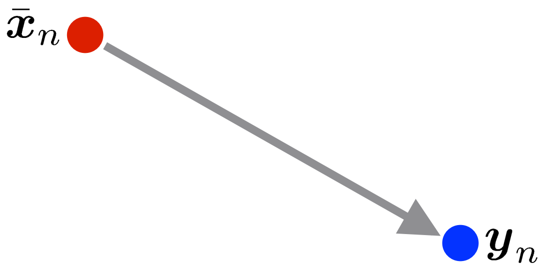


Image courtesy: Bogoslavskyi

Symmetric Point-to-Plane

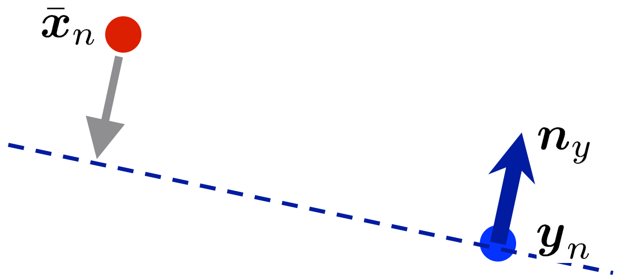
- Point-to-plane metric is not symmetric

point-to-point



$$\min \sum ||\boldsymbol{y}_n - \bar{\boldsymbol{x}}_n||^2$$

point-to-plane



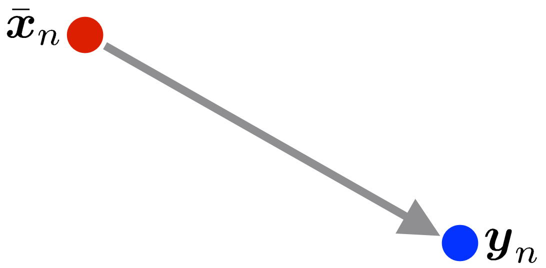
$$\min \sum \left((\boldsymbol{y}_n - \bar{\boldsymbol{x}}_n) \cdot \boldsymbol{n}_y\right)^2$$

Image courtesy: Rusinkiewicz

Symmetric Point-to-Plane

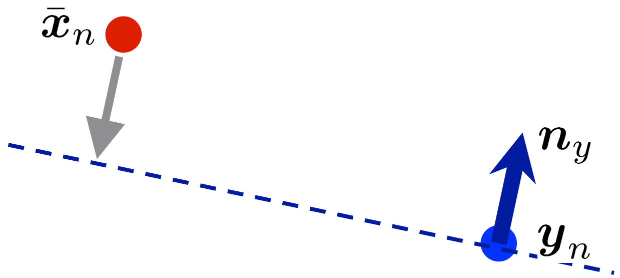
- Point-to-plane metric is not symmetric

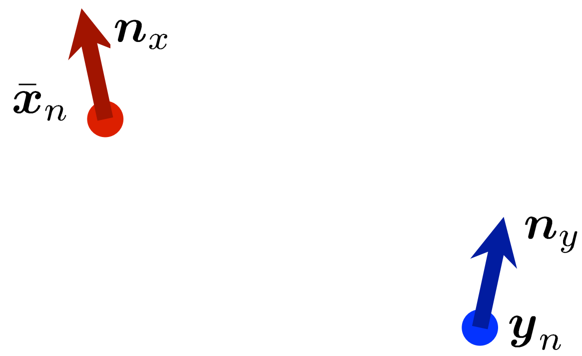
point-to-point



$$\min \sum ||\boldsymbol{y}_n - \bar{\boldsymbol{x}}_n||^2$$

point-to-plane





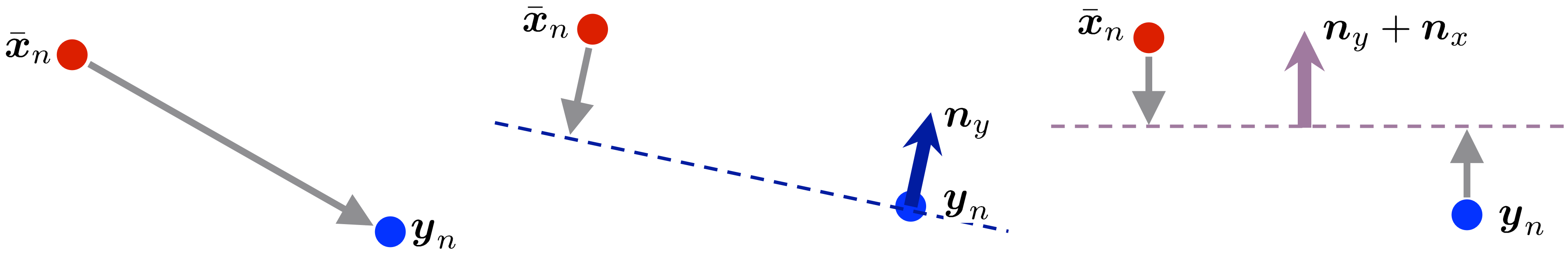
$$\min \sum \left((\boldsymbol{y}_n - \bar{\boldsymbol{x}}_n) \cdot \boldsymbol{n}_y \right)^2$$

Image courtesy: Rusinkiewicz

Symmetric Point-to-Plane

- Point-to-plane metric is not symmetric
- We can easily combine normals from both surfaces to obtain symmetry

point-to-point — point-to-plane — symmetric



$$\min \sum ||\boldsymbol{y}_n - \bar{\boldsymbol{x}}_n||^2 \qquad \min \sum ((\boldsymbol{y}_n - \bar{\boldsymbol{x}}_n) \cdot \boldsymbol{n}_y)^2 \qquad \min \sum ((\boldsymbol{y}_n - \bar{\boldsymbol{x}}_n) \cdot (\boldsymbol{n}_y + \boldsymbol{n}_x))^2$$

Image courtesy: Rusinkiewicz

Symmetric Point-to-Plane

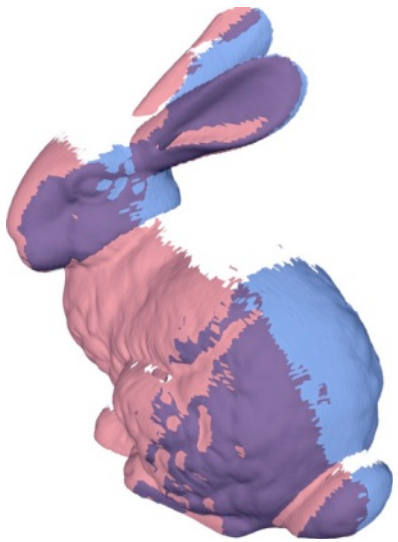
- Point-to-plane metric is not symmetric
- We can easily combine normals from both surfaces to obtain symmetry

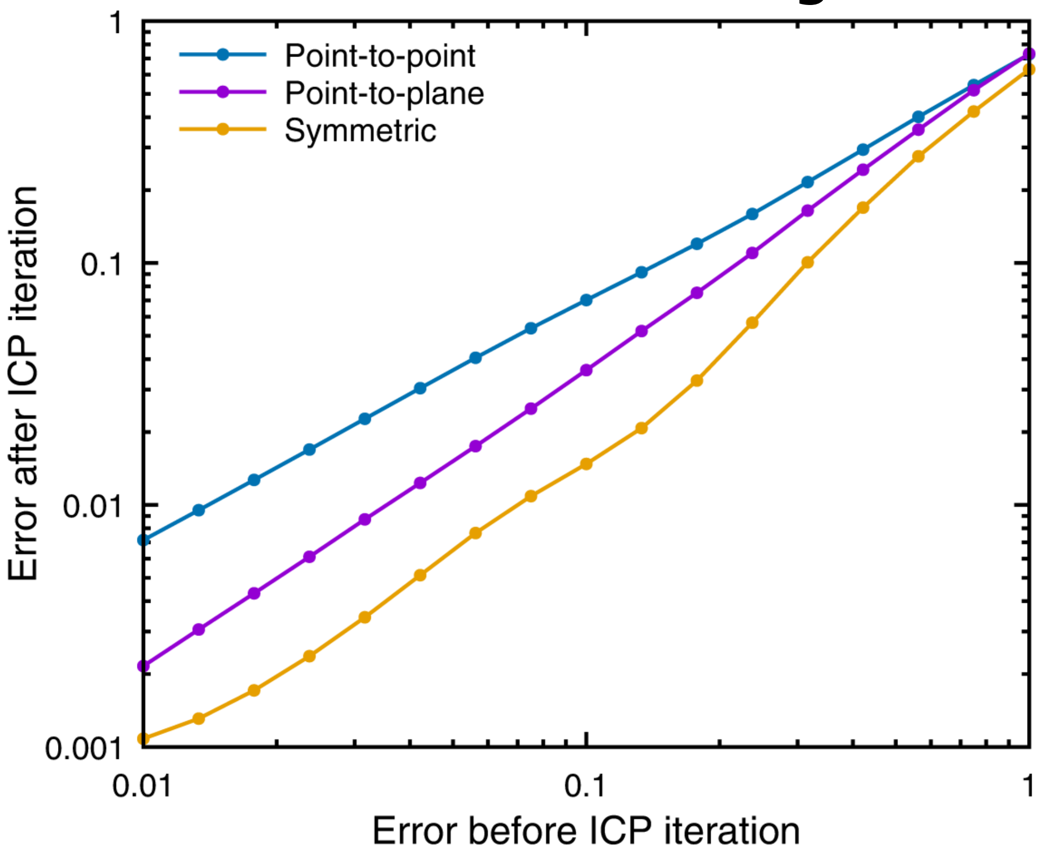
$$\min \sum \left((\boldsymbol{y}_n - \bar{\boldsymbol{x}}_n) \cdot (\boldsymbol{n}_y + \boldsymbol{n}_x) \right)^2$$

Additional work: requires computing the normals in both clouds (originally in one)

Comparison of Metrics (Bunny dataset)

Note: log scale





Symmetric metric performs best

Image courtesy: Rusinkiewicz

Symmetric Point-to-Plane

Combine normals from both surfaces to obtain a symmetric metric

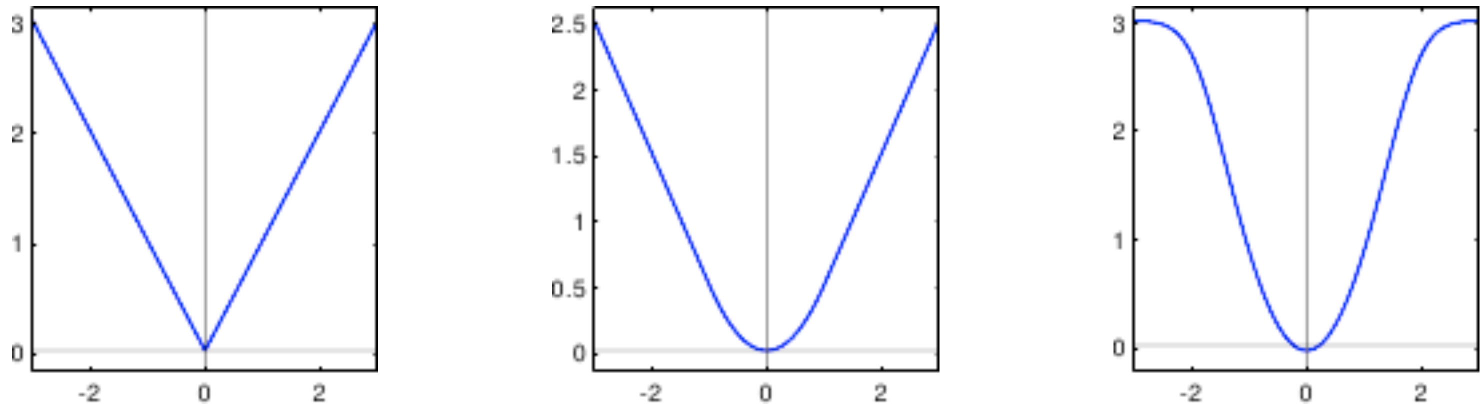
$$\min \sum \left((\boldsymbol{y}_n - \bar{\boldsymbol{x}}_n) \cdot (\boldsymbol{n}_y + \boldsymbol{n}_x) \right)^2$$

A simple change that leads to an improved performance of ICP (speed, basin of convergence)

Robust Least Squares

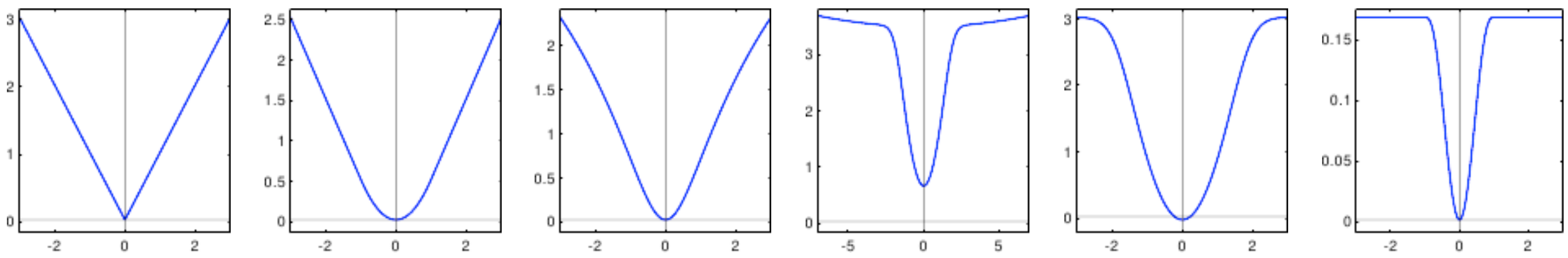
Robust Least Squares

- Data association outliers strongly impact the least squares result
- Robust kernels / M-estimators aims at down-weighing the impact of outliers
- Function that changes the error function depending on its magnitude



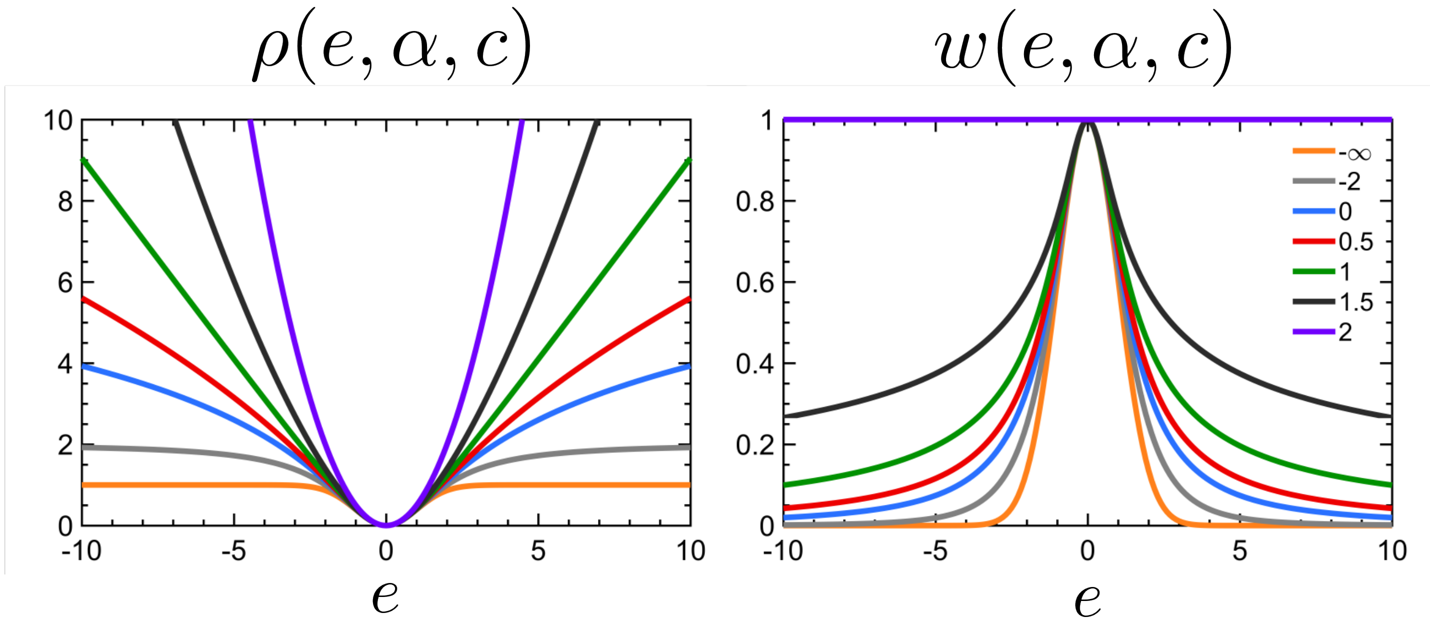
Robust Least Squares

- Weighted least squares approach to realize robust least squares estimation
- Each kernel yields a specific weight
- The kernel will impact the Jacobians
- The rest stays the same
- The choice of the kernel must align with the outlier distribution



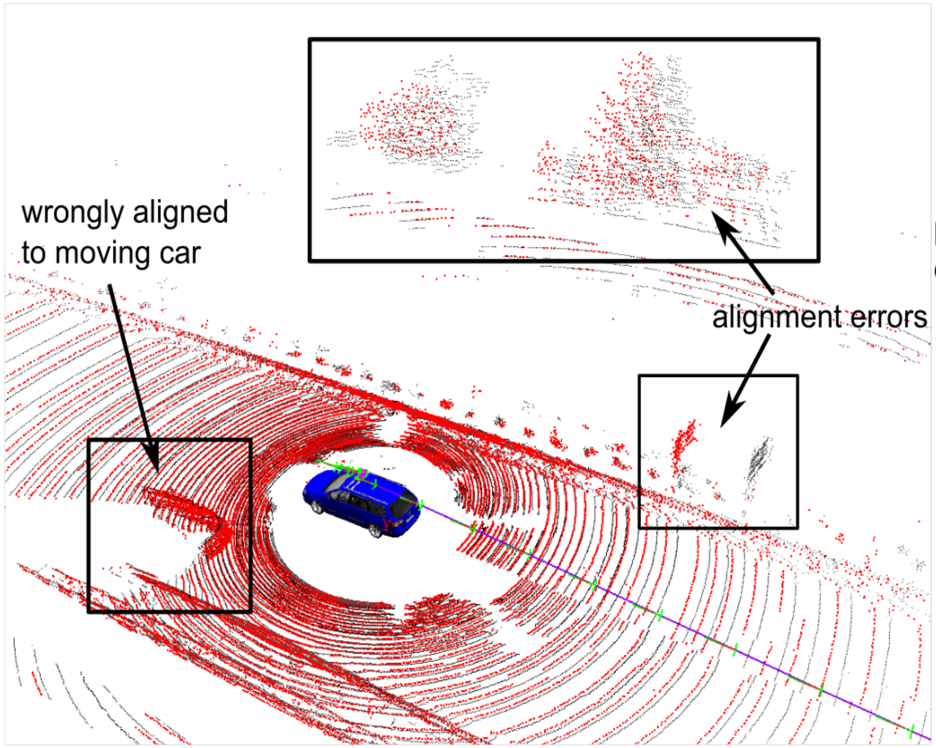
Kernel for Outlier Rejection

- Apply robust kernels to down-weigh the impact of potential outliers
- Kernel parameter can be adjusted

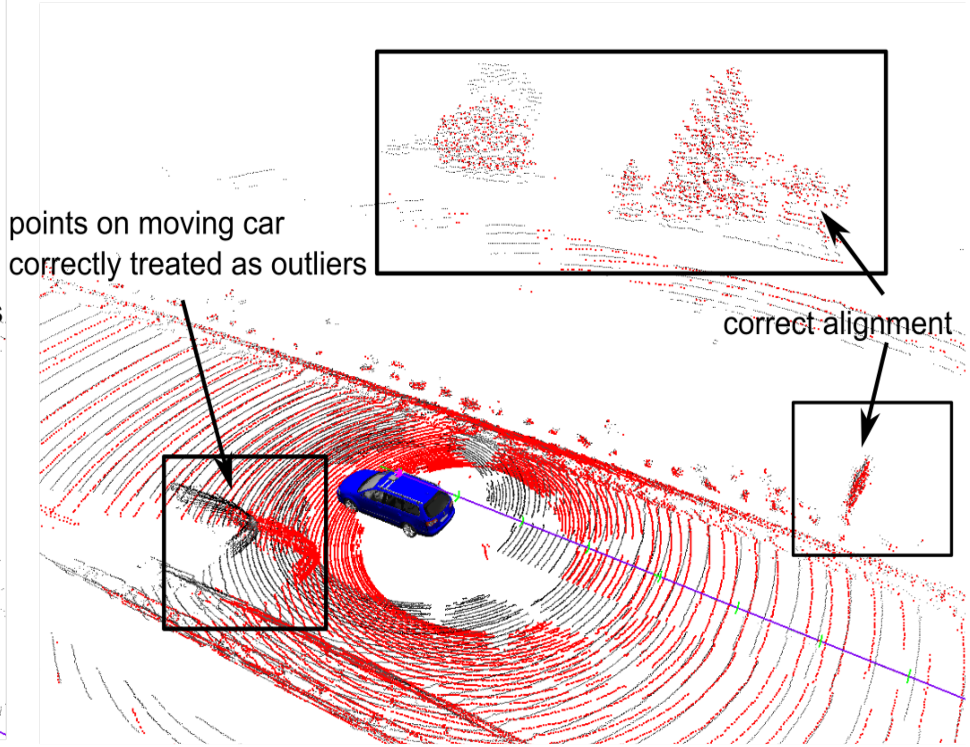


See: Chebrolu, Läbe, Vysotska, Behley, Stachniss: "Adaptive Robust Kernels for Non-Linear Least Squares Problems"

Robust Kernels in Action



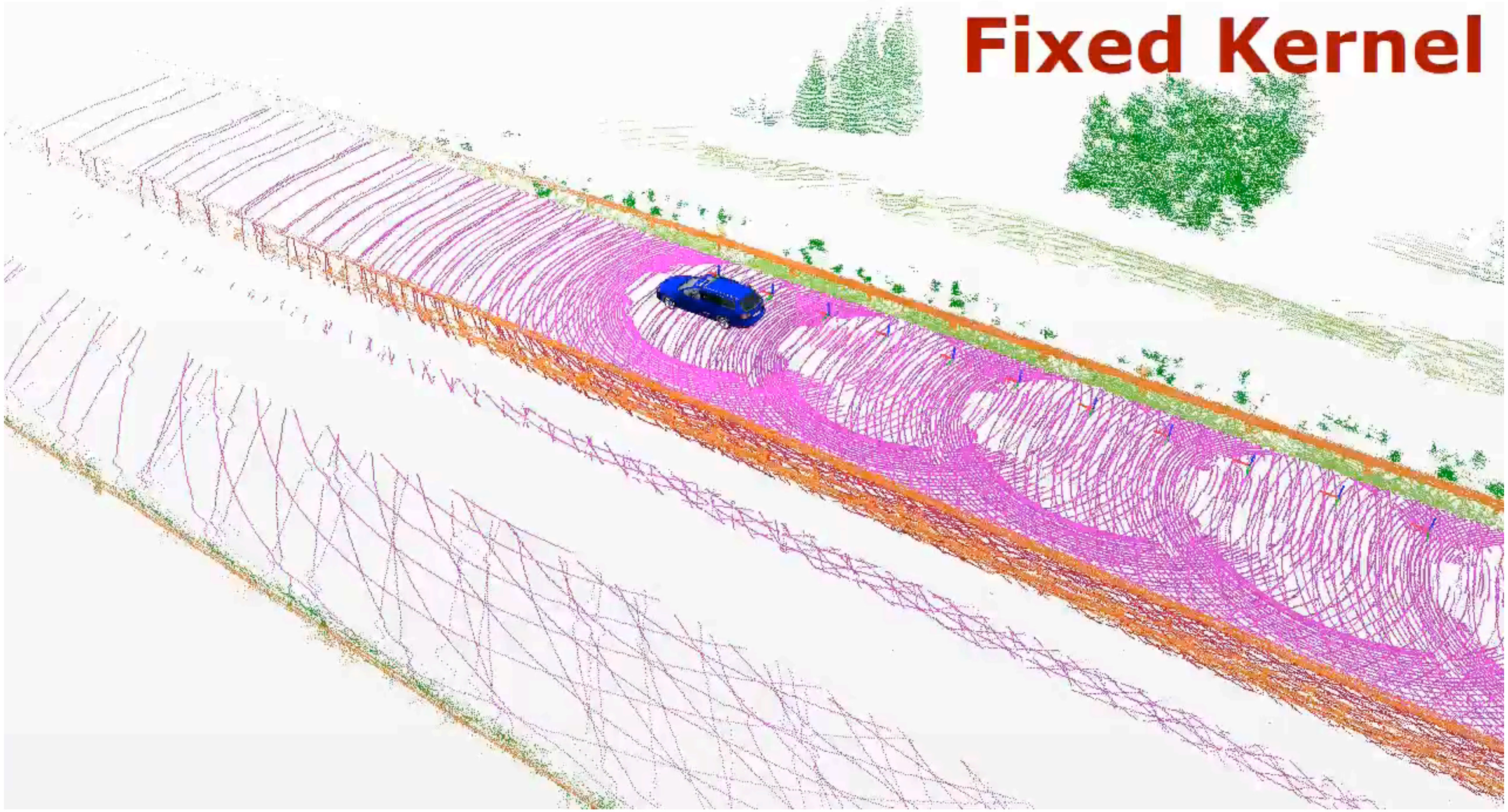
Huber Loss



Adaptive Robust Loss

Outlier rejection in presence of dynamic objects

Adaptive Robust Kernels



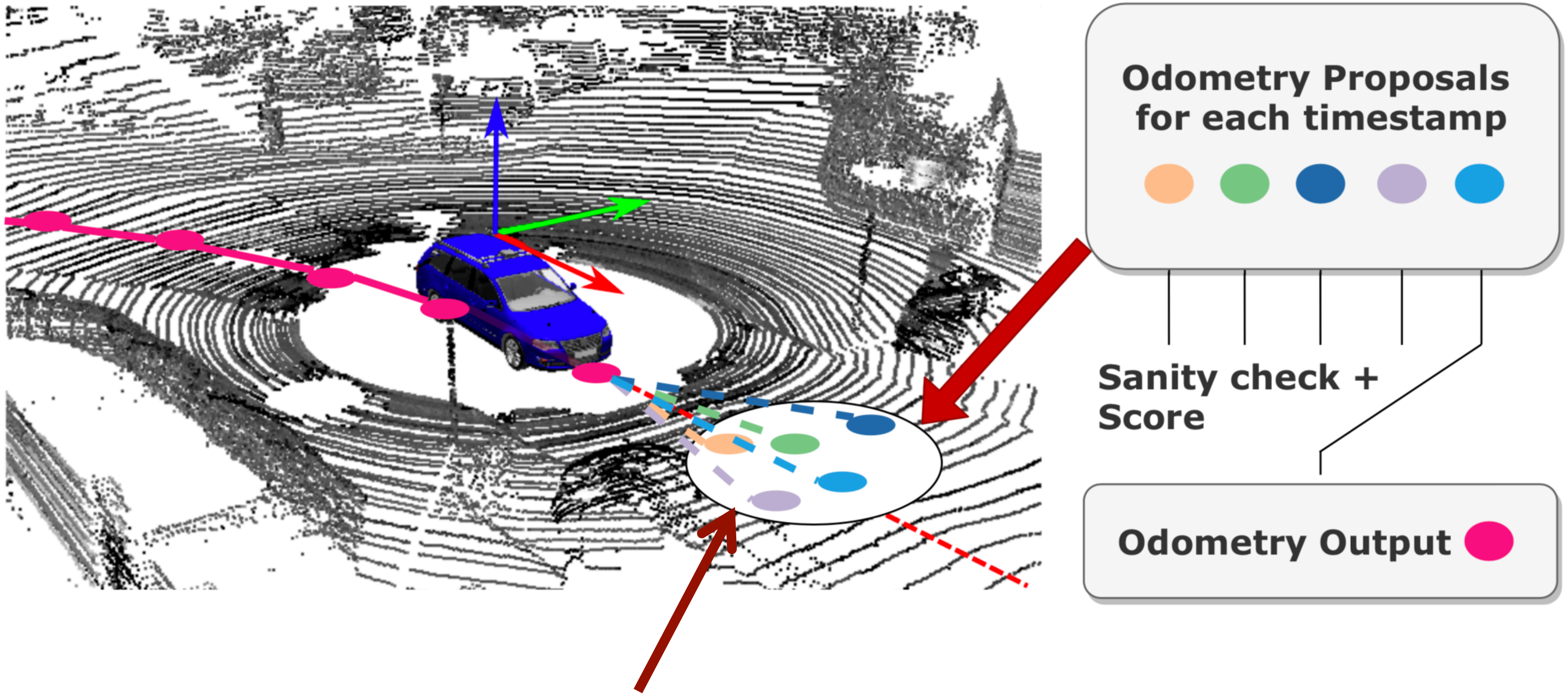
Chebrolu, Läbe, Vysotska, Behley, Stachniss: "Adaptive Robust Kernels for Non-Linear Least Squares Problems"

Outlier Rejection is Key

- Finding the correct data association is key for robust registration
- Approaches often also use heuristics as an initial guess for associations
- Example questions:
  - Are there some well-identifiable points?
  - Do we know something about potentially moving objects in the scene?
  - Can we exploit ego-motion estimates?

See also Part 2 of the lecture!

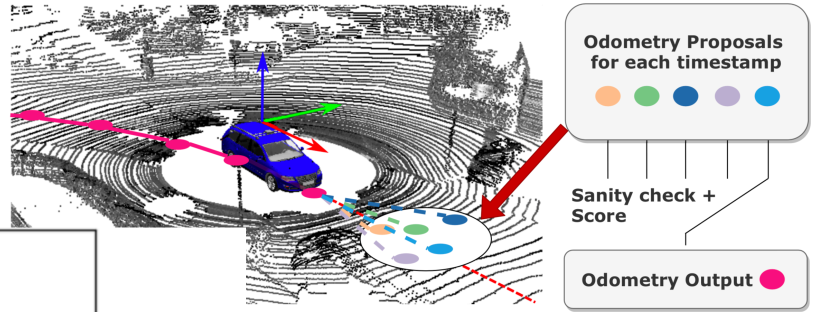
Redundant Odometry

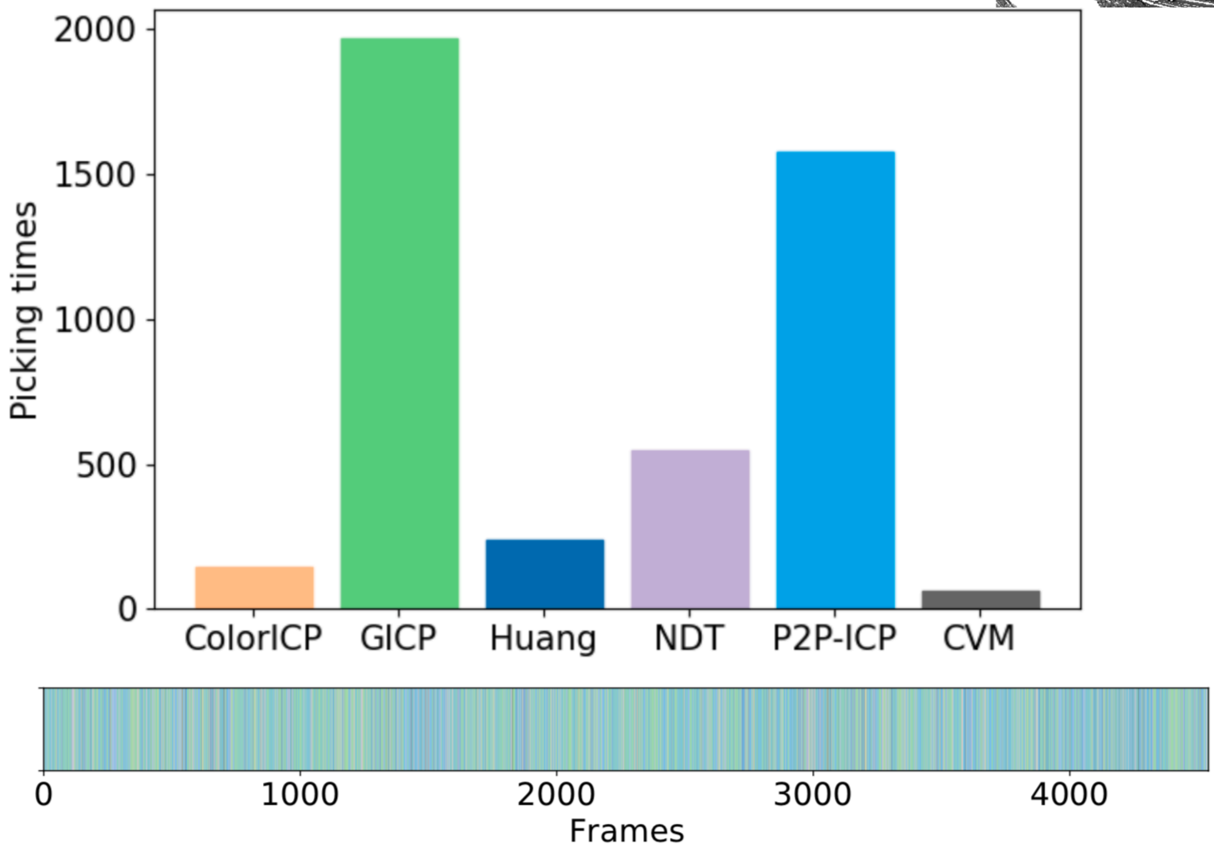


multiple motion estimates

Reinke, Chen, Stachniss: "Simple But Effective Redundant Odometry for Autonomous Vehicles", ICRA 2021

Different Approaches Win in Different Situations





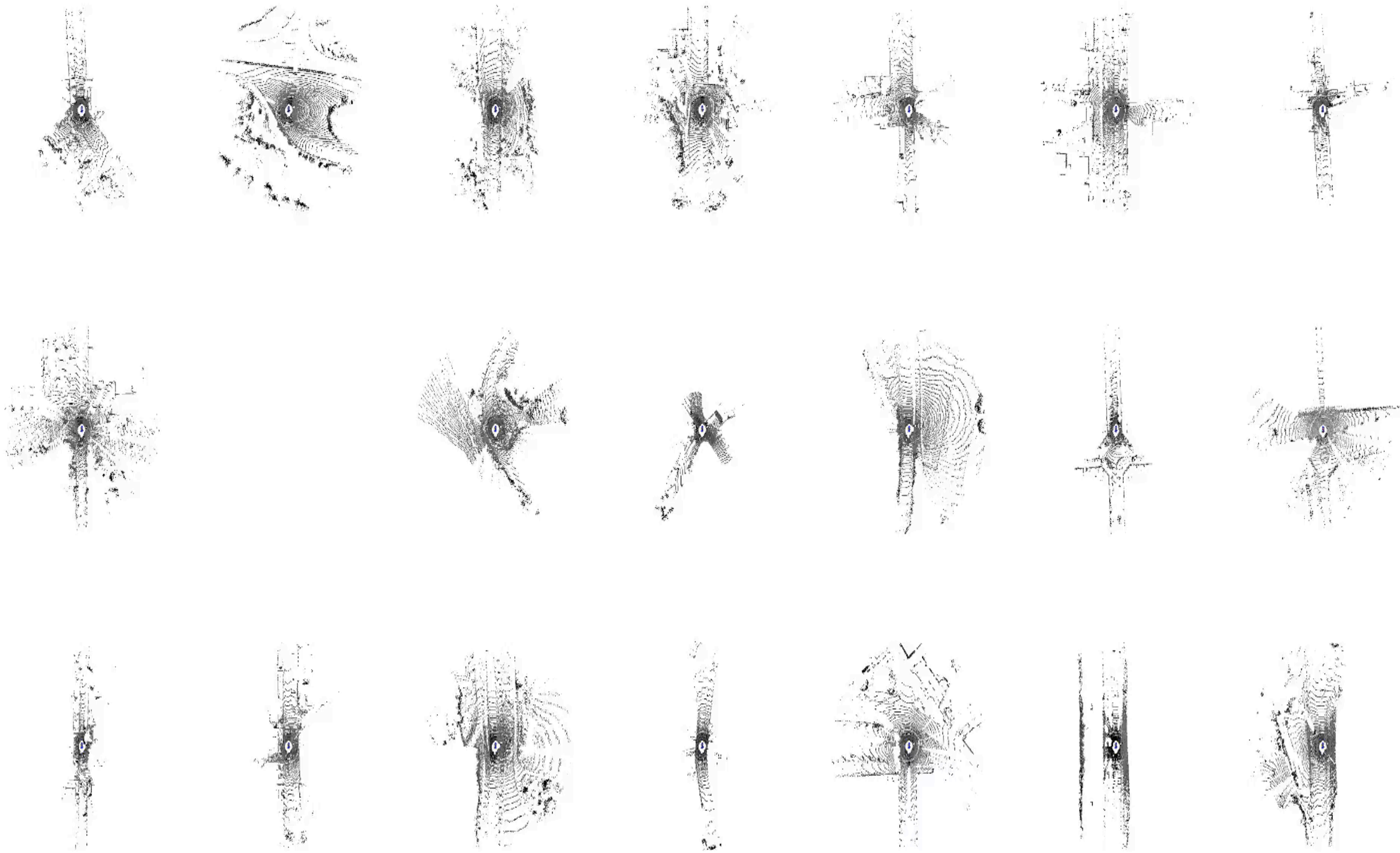
Remarks from Practice

- Always exploit an initial guess (odometry, constant velocity, ...)
- Normal-based metrics often better than standard point-to-point metric
- Symmetric metric often performs well
- Exploit informed outlier rejection if possible/available
- Adaptive kernels adapt to the outlier situation for each scan pair

Remarks from Practice

- For “sensor odometry” estimation, exploit multiple sensors
- Sanity check to detect failures
  - Vehicle constraints
  - Dynamic constraints
  - ...
- At some point, SLAM with loop closing and global optimization is needed
- Remark: proper point uncertainties are often tricky to estimate

SuMa: LiDAR-based SLAM



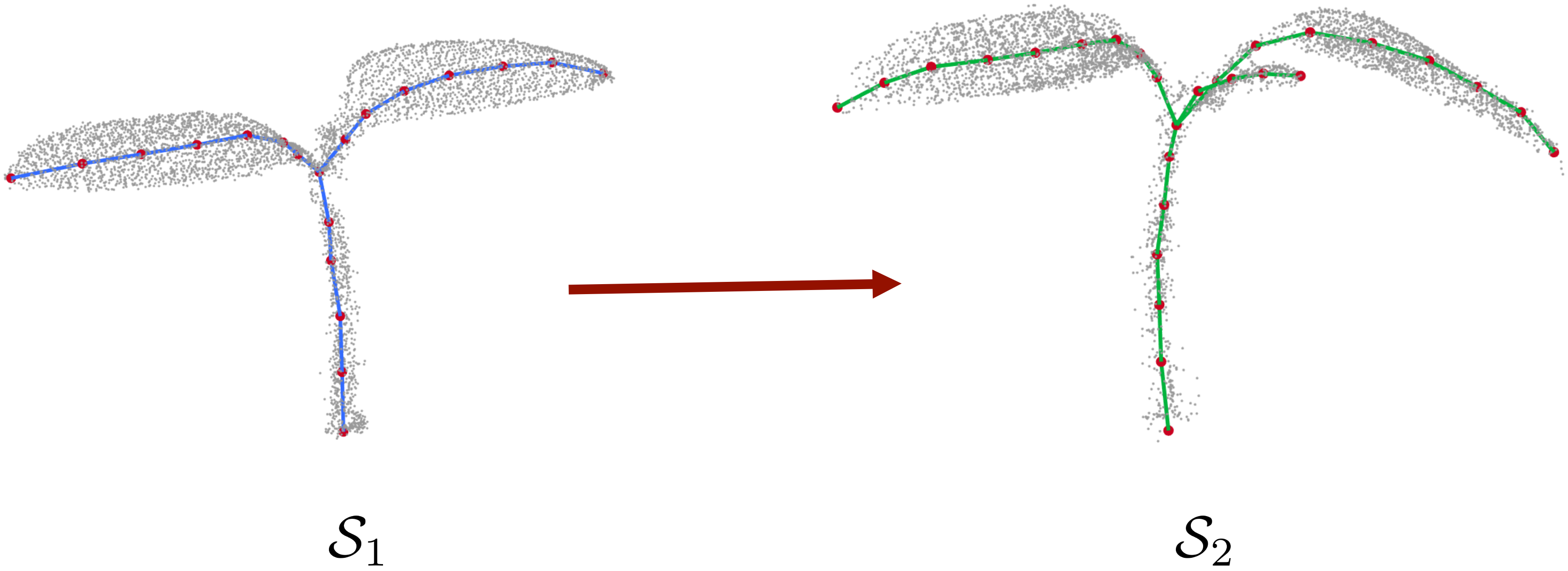
Going a Step Further: Non-Rigid Registration

Non-Rigid Registration

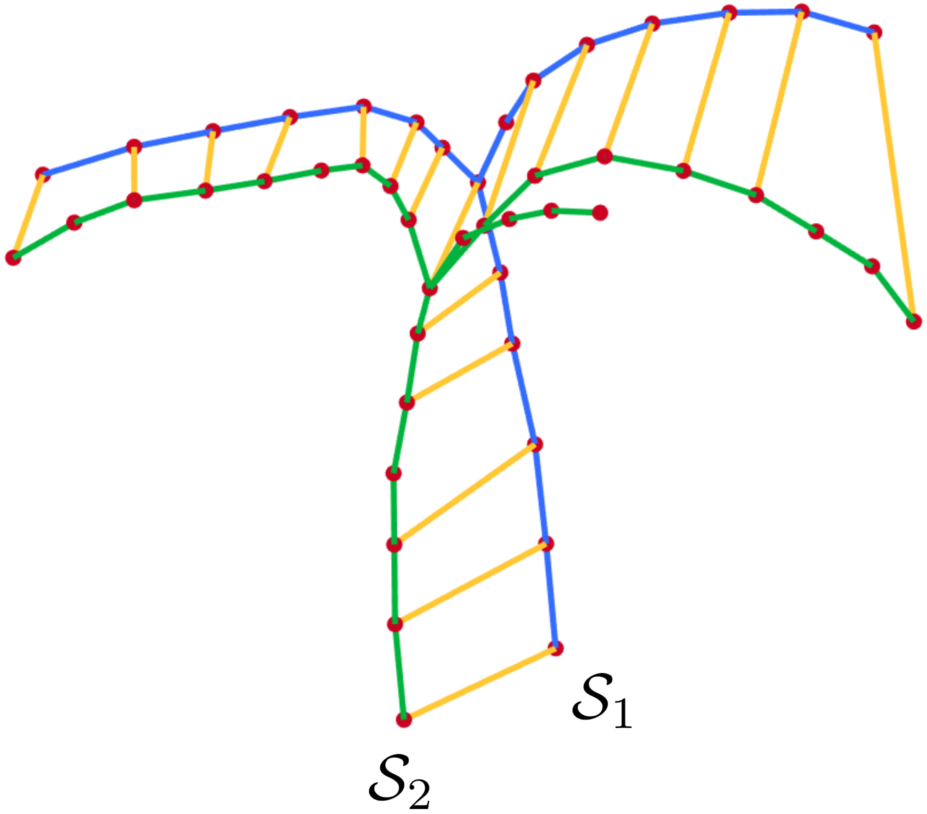
- What happens when the objects are non-rigid and can be deformed?
- Location-specific transformations
- Object deformations often encoded via an additional cost term
- Leads to least squares methods with more complex cost functions

Non-Rigid Registration Example: Time Series of 3D Point Clouds

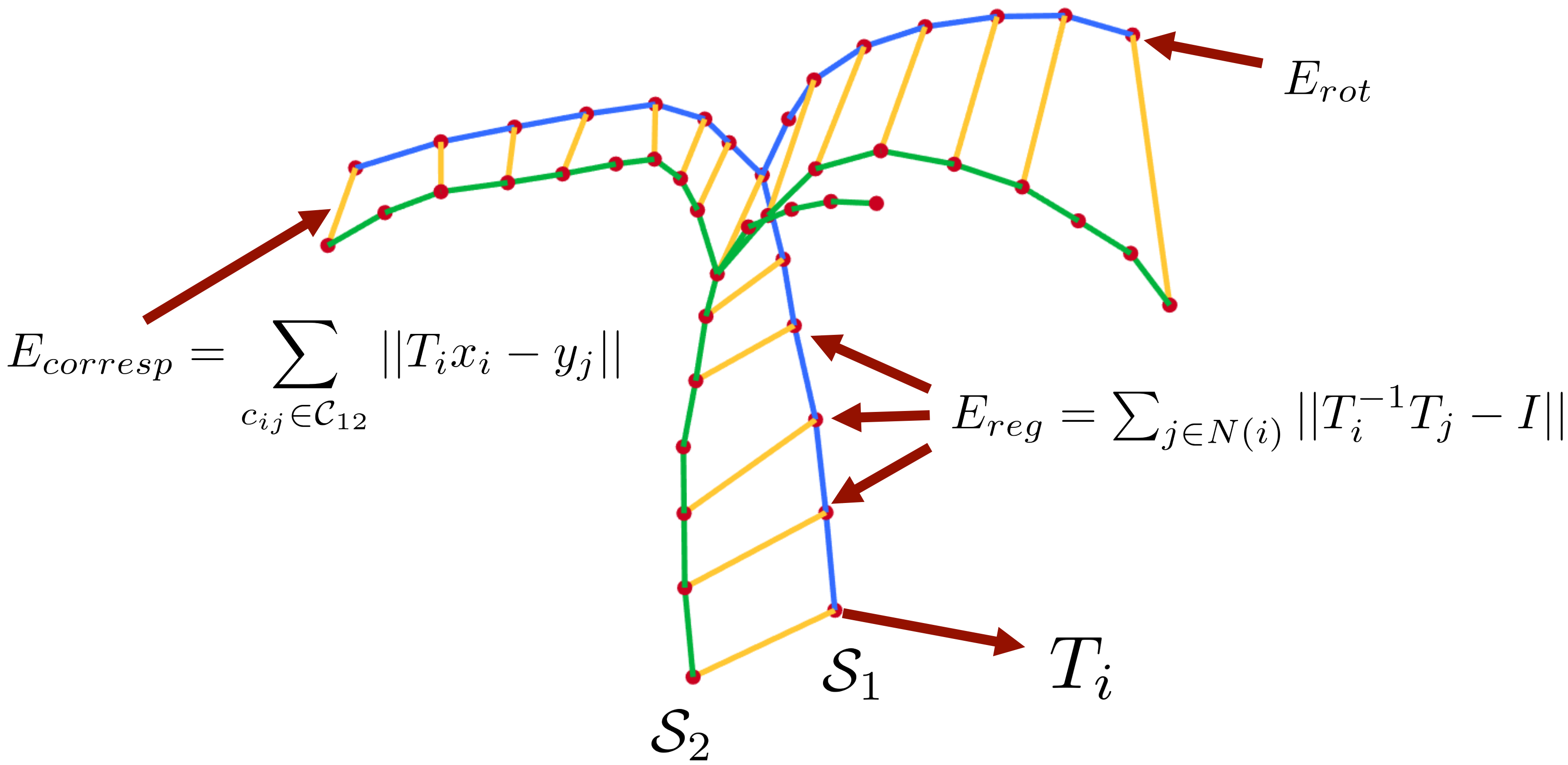
Non-Rigid Registration Example: Simplified Data Association via Skeleton Matching



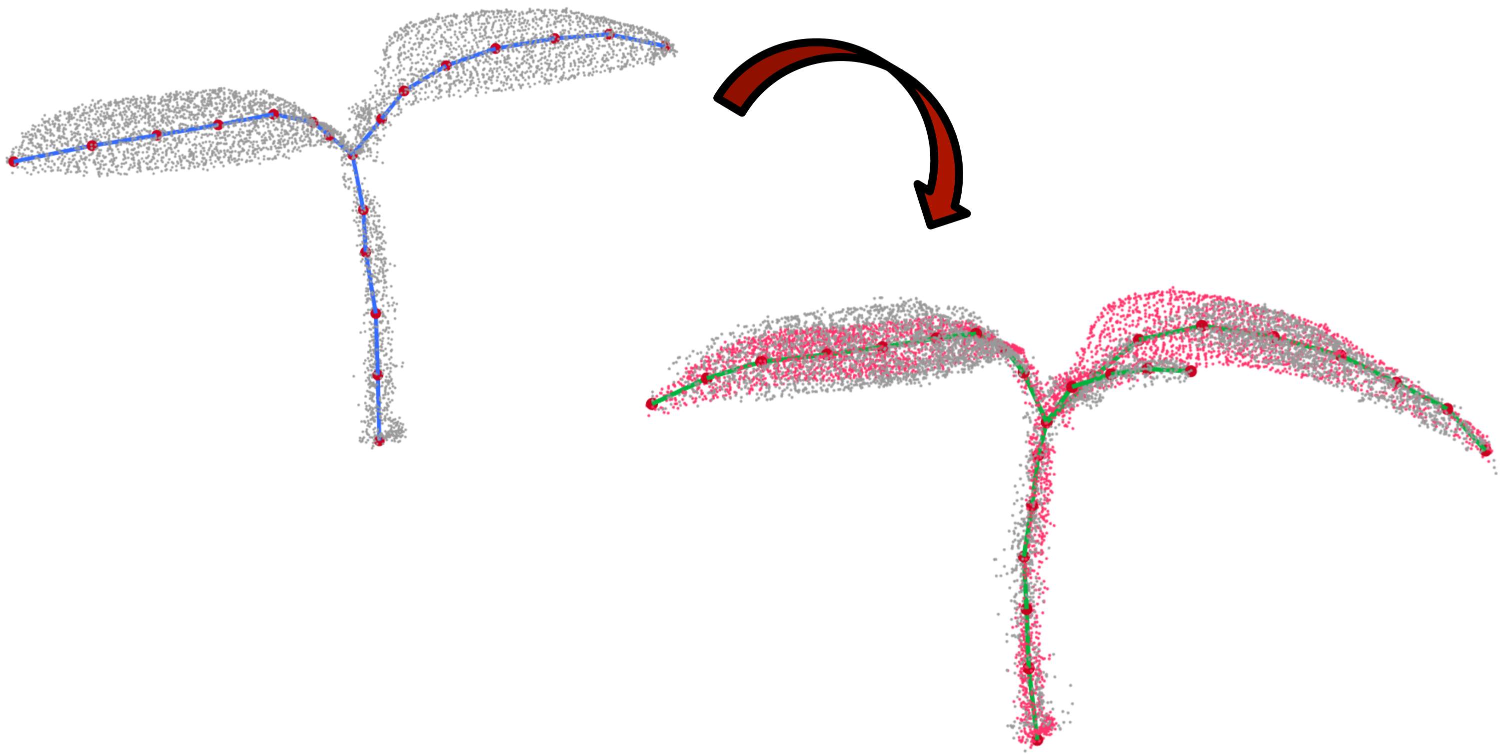
Non-Rigid Registration Example: Estimating Correspondences



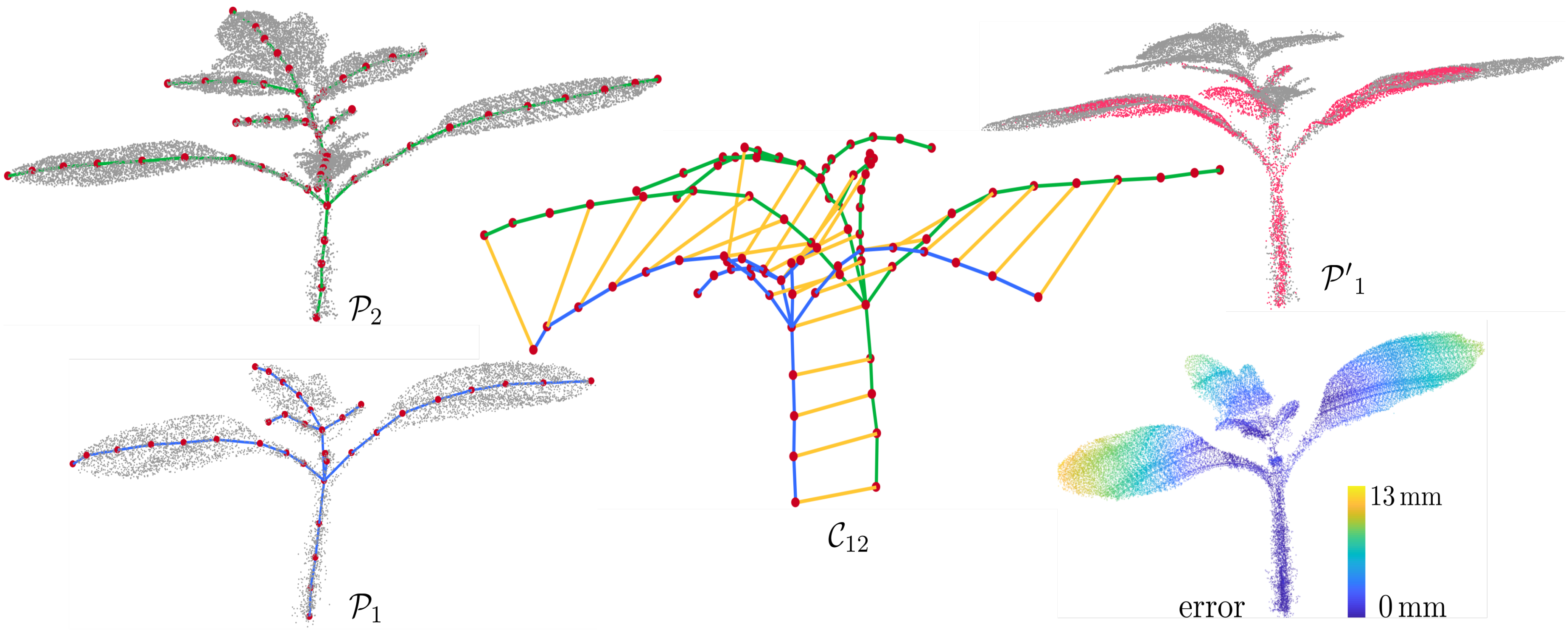
Non-Rigid Registration Example: Skeleton Deformation



Non-Rigid Registration Example: Back to the Point Clouds

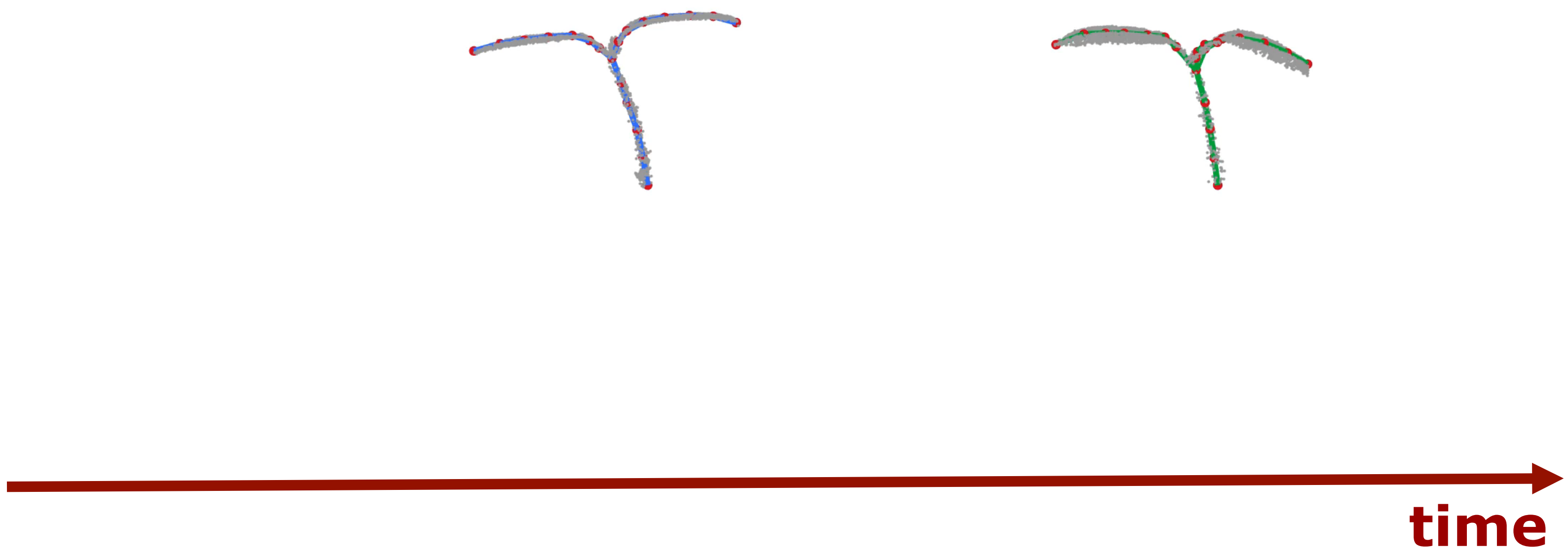


Non-Rigid Registration Example: Registration Results



Day 6 vs. Day 10

Non-Rigid Registration Example: Timeline Interpolation



Registering Humans

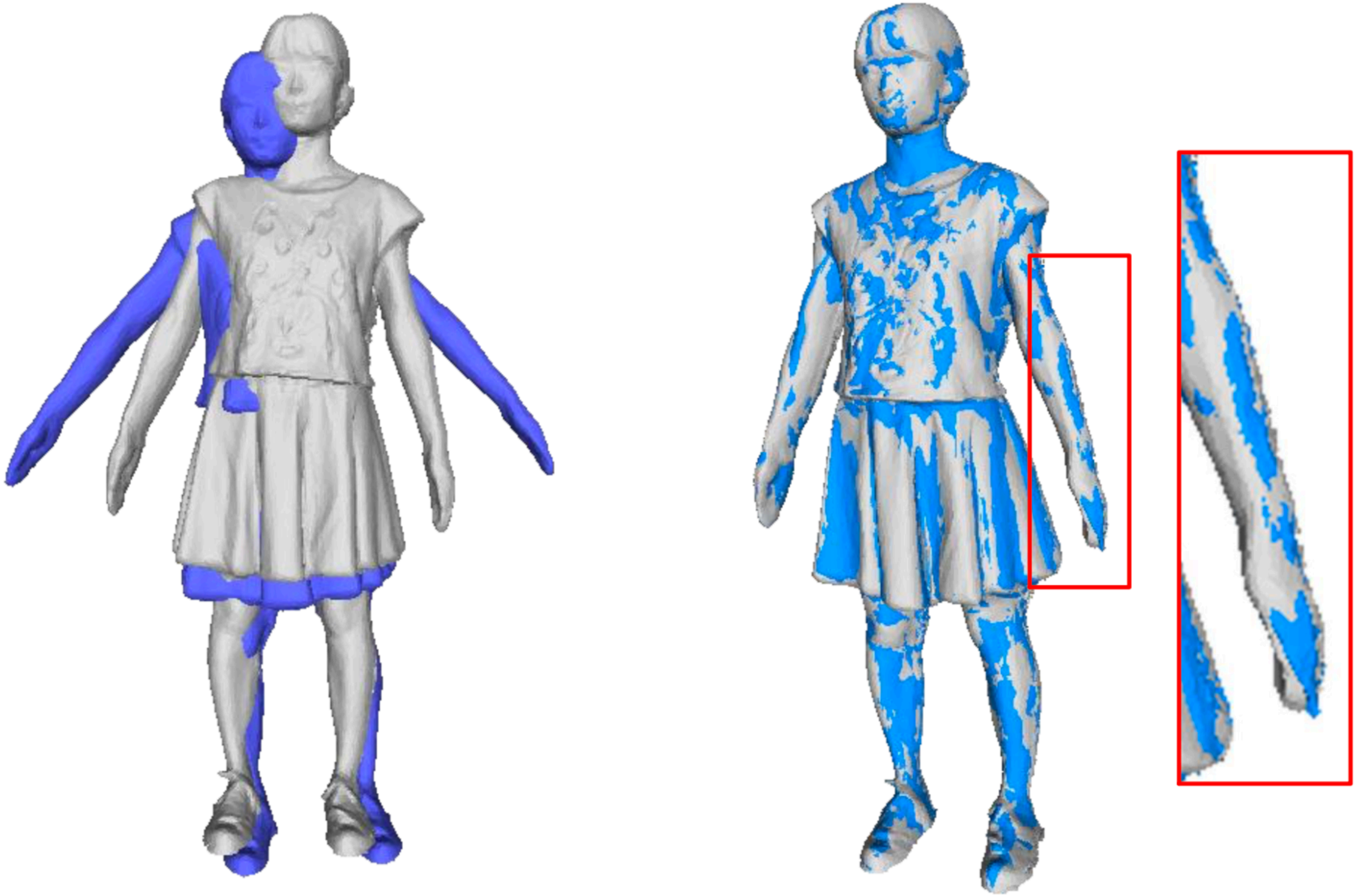


Image courtesy: Li, Yang, Lai, Guo

Resources

Notebook by Igor Bogoslavskyi

jupyter nbviewer

JUPYTER FAQ

notebooks / icp.ipynb

ICP

This notebook is all about ICP and it's different implementations. It should be visual and self - descriptive.

Contents:

- Overview
- ICP based on SVD
- Non linear Least squares based ICP
- Using point to plane metric with Least Squares ICP
- Dealing with outliers

Overview

Having two scans $P = \{p_i\}$ and $Q = \{q_i\}$ we want to find a transformation (rotation R and translation t) to apply to P to match Q as good as possible. In the remainder of this notebook we will try to define what does "as good as possible mean" as well as ways to find such a transformation.

In [1]:
```
import sys
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import animation, rc
from math import sin, cos, atan2, pi
from IPython.display import display, Math, Latex, Markdown, HTML
```

The way we will plot the data

In [2]:
```
def plot_data(data_1, data_2, label_1, label_2, markersize_1=8, markersize_2=8):
```

https://nbviewer.jupyter.org/github/niosus/notebooks/blob/master/icp.ipynb

Open3D – A Popular Library

http://www.open3d.org/

Open3D
A Modern Library for 3D Data Processing

Home Blog Documentation Code Help

Introduction

Open3D is an open-source library that supports rapid development of software that deals with 3D data. The Open3D frontend exposes a set of carefully selected data structures and algorithms in both C++ and Python. The backend is highly optimized and is set up for parallelization. Open3D was developed from a clean slate with a

RECENT NEWS

The best present: Open3D 0.12.0
December 24, 2020

Further Reading

- Jupyter notebook by I. Bogolslavskyi (highly recommended)
- SimpleICP by P. Glira: https://github.com/pglira/simpleICP
- Arun et al. “Least-Squares Fitting of Two 3D Point Sets”
- Besl & McKay “Registration of 3-D shapes”
- Pomerleau et al. “Review of Point Cloud Registration”
- Rusinkiewicz et al. “Efficient Variants of ICP” ···
- Rusinkiewicz: “A Symmetric Objective Function for ICP”
- Pomerleau et al. “Comparing ICP Variants”
- Serafin & Grisetti: “Normal-ICP”
- Segal et al. “Generalized ICP”
- Yang et al. “Go-ICP”
- Chenbrolu et al. “Adaptive Kernels”
- Agamennoni et al. "Self-tuning M-estimators”
- Chen et al. “Moving object Segmentation”
- Landry et al. “CELLO-3D: Covariances for ICP”
- Babin et al. “Analysis of Robust Functions for ICP”
- Della Corte et al. “Photometric point cloud registration”
- Behley & Stachniss “SuMa: Projective ICP in LiDAR SLAM”

Summary

- Registration of point clouds is an important task in perception
- ICP is the standard algorithm for point cloud alignment/scan matching
- Estimates translation and rotation between clouds/scans
- Given data associations between clouds, the transformation can be computed efficiently

Summary

- **The major problem is to determine the correct data associations**
- Iterative approach (DA & alignment)
- Several variants exist
- Initial guess is needed for robust data association
- **Often:** least squares approach with a plane-based metric, data association heuristics, and outlier rejection

5 Minute Summary…

https://www.youtube.com/watch?v=QWDM4cFdKrE