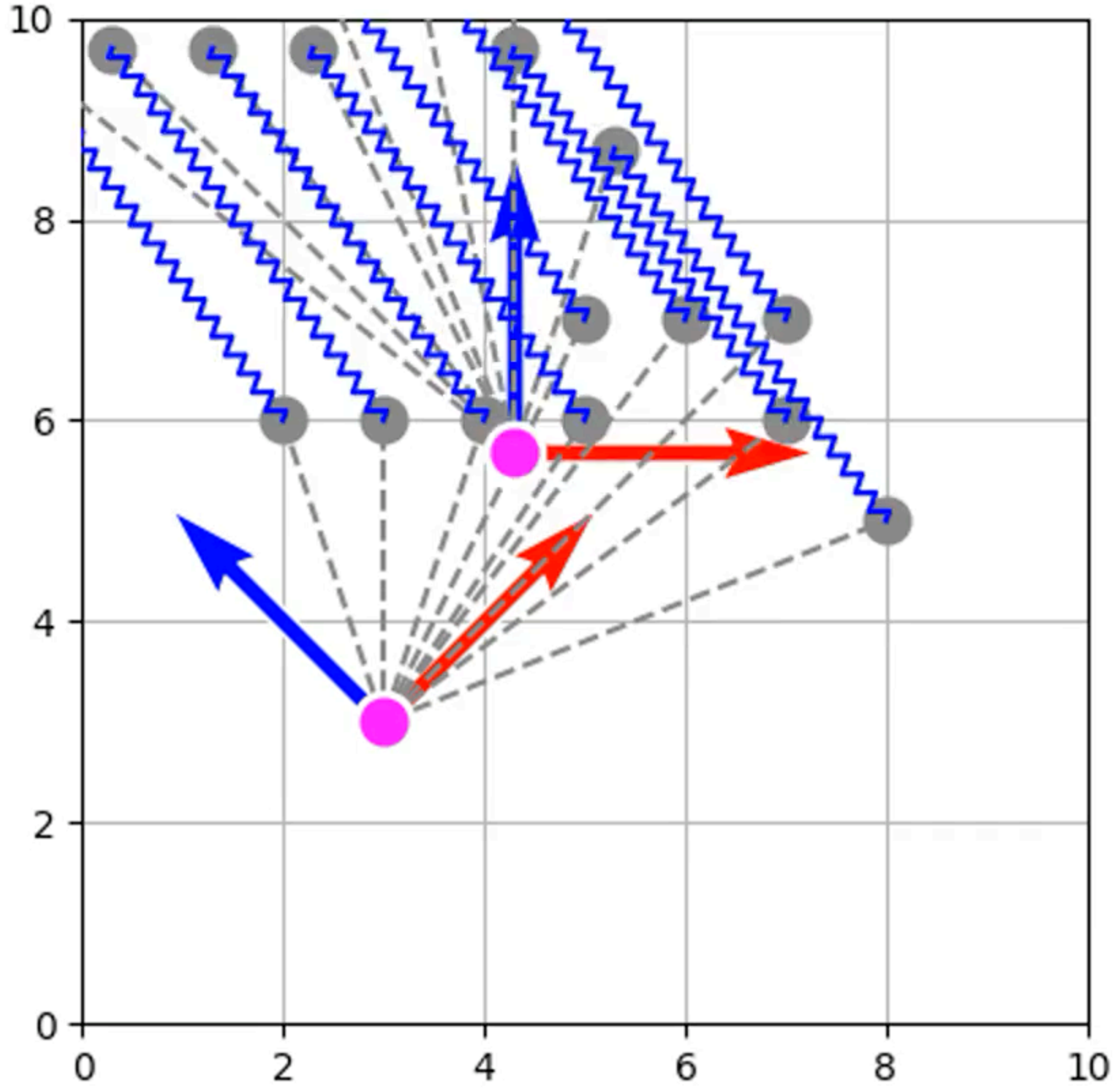
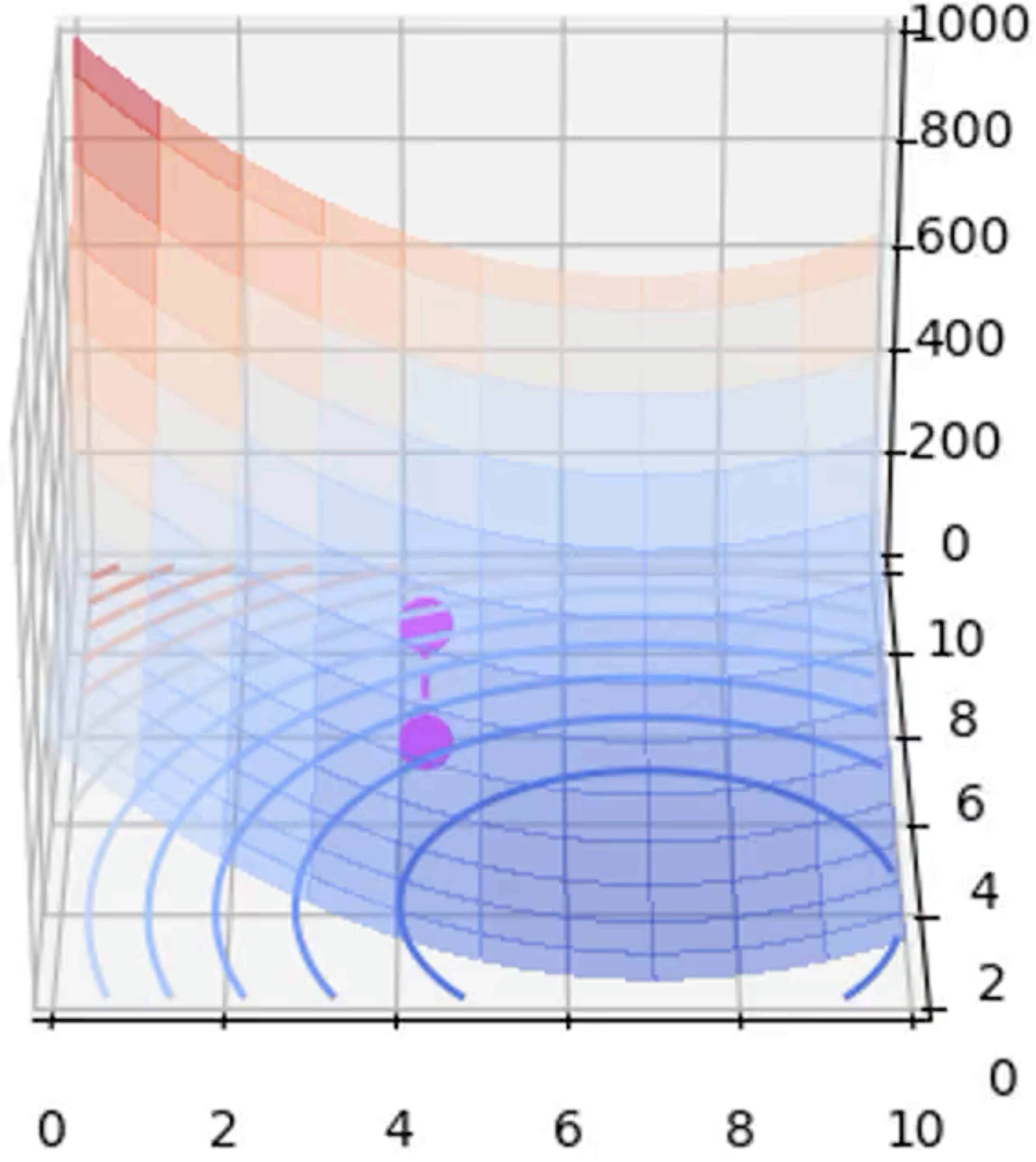
Robust regression: from ICP to RANSAC

Karel Zimmermann

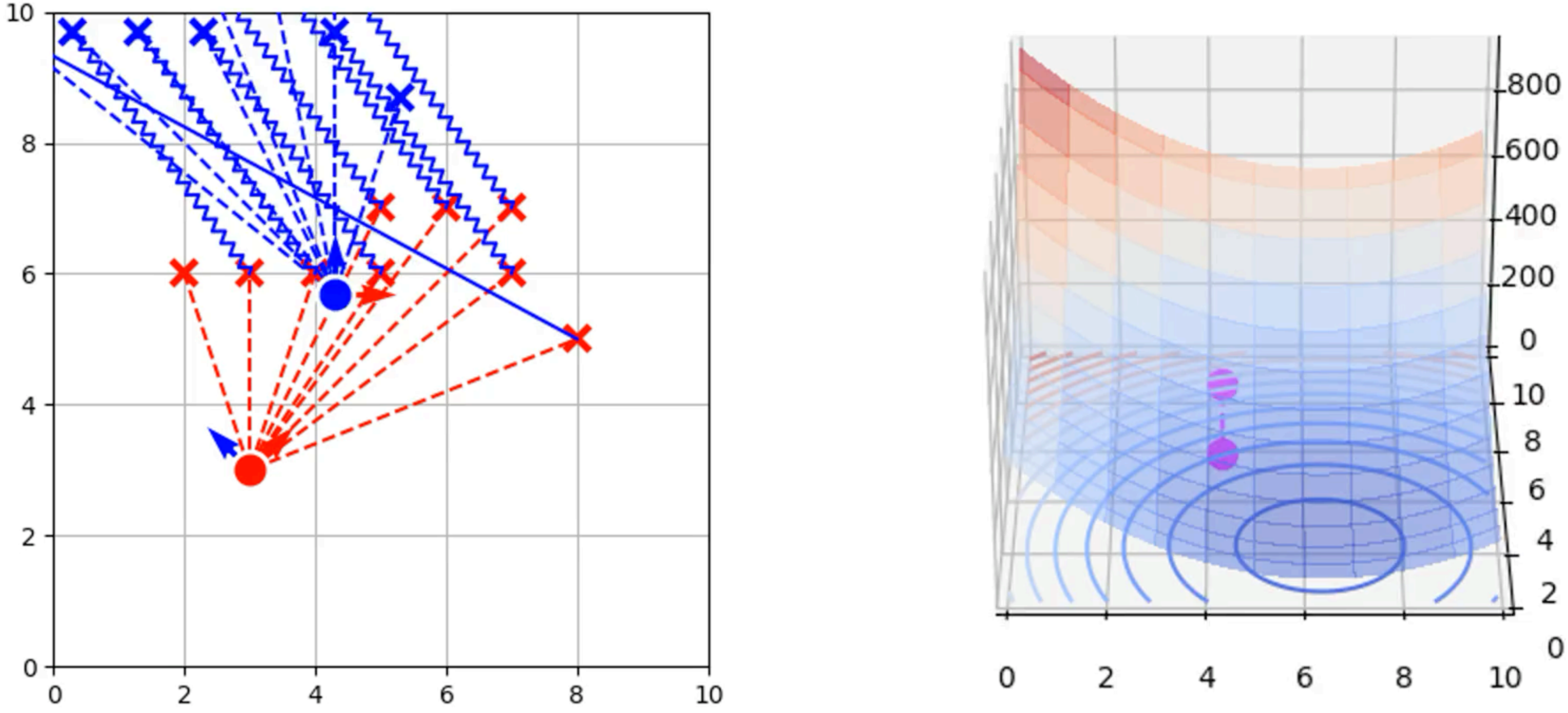
L2 loss only inliers





Ground truth position [7,2]

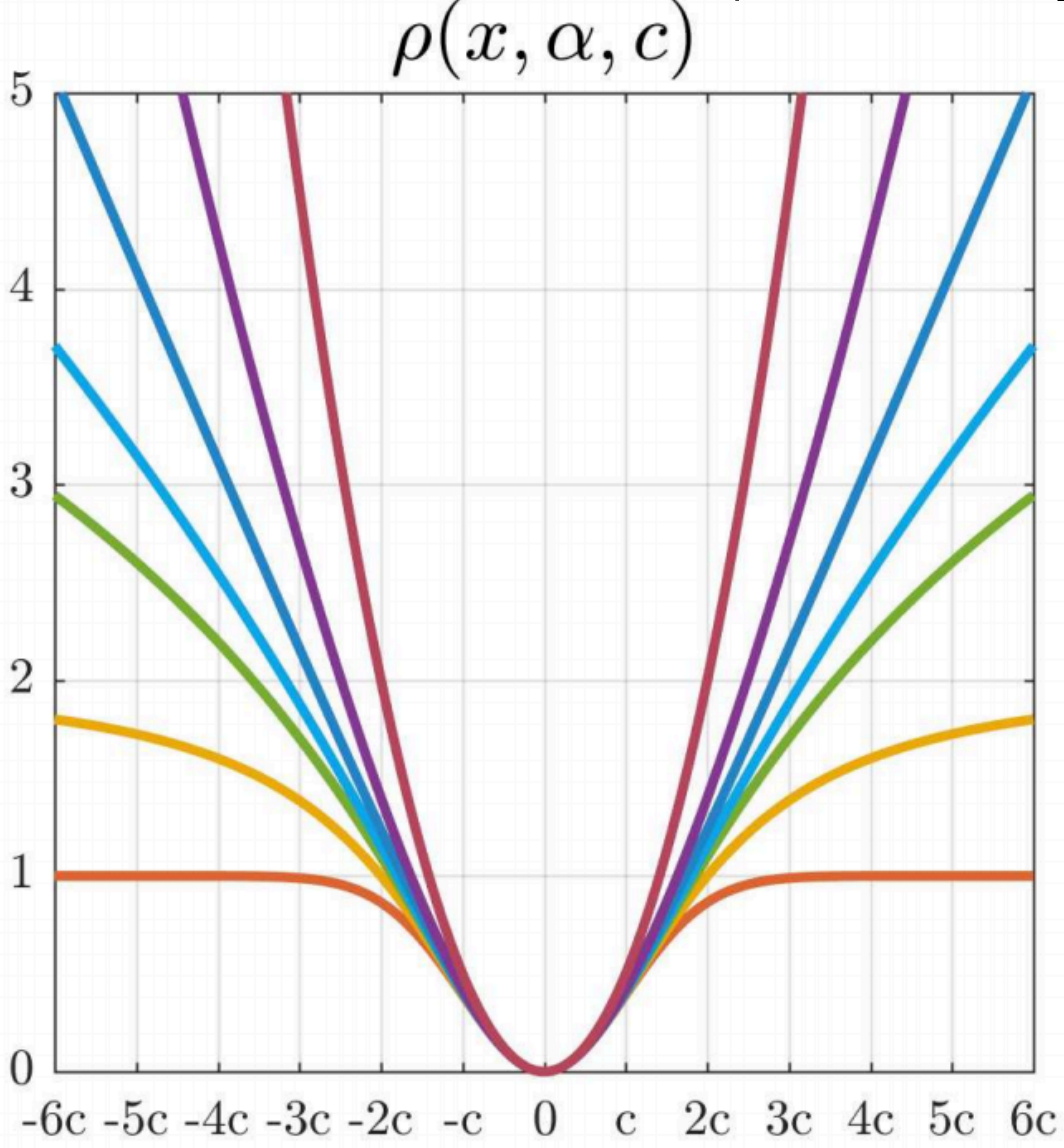
L2 loss with outlier



Result deviated by 1m

Shape of robust regression functions [Barron CVPR 2019]

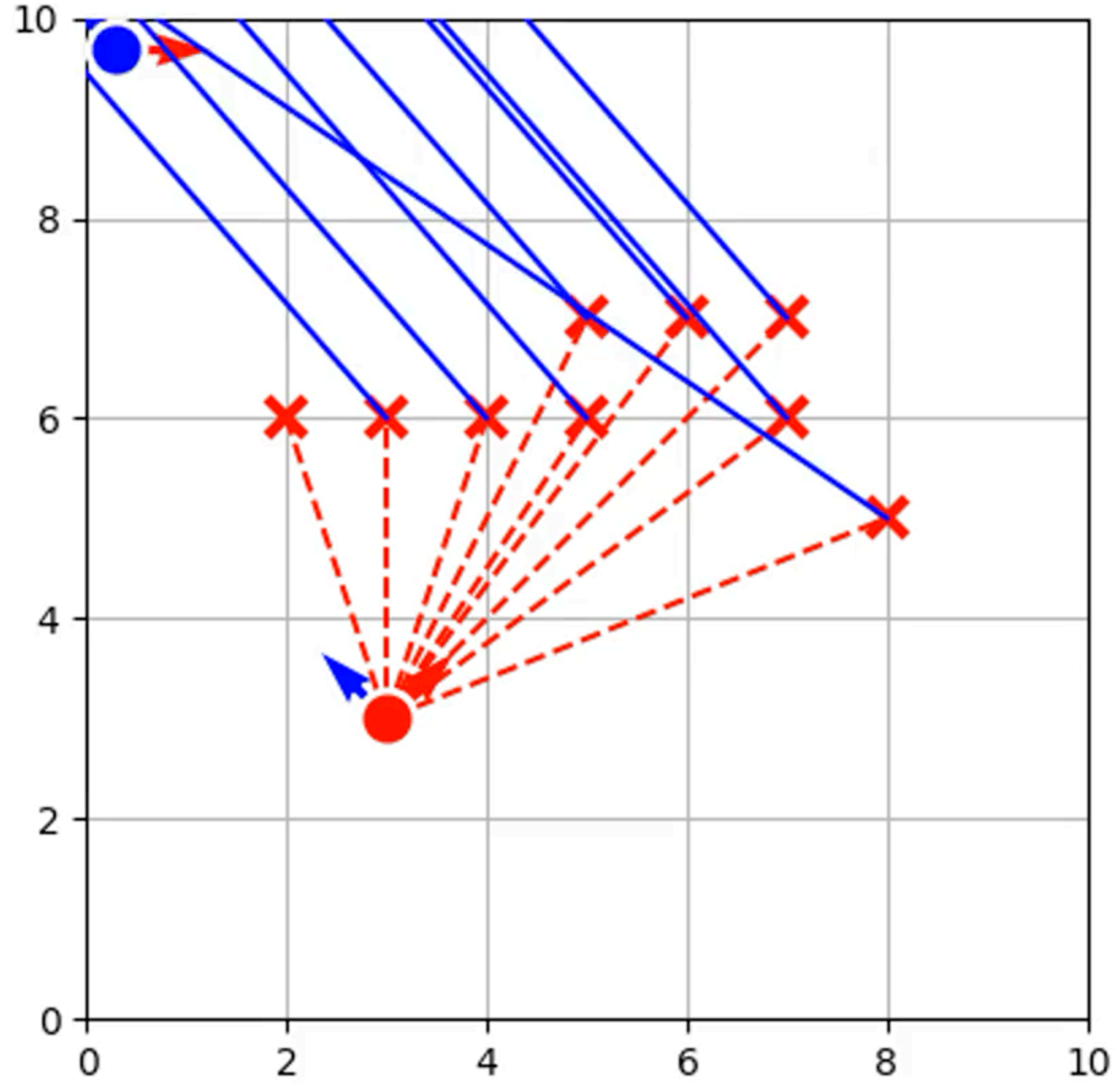
https://arxiv.org/abs/1701.03077

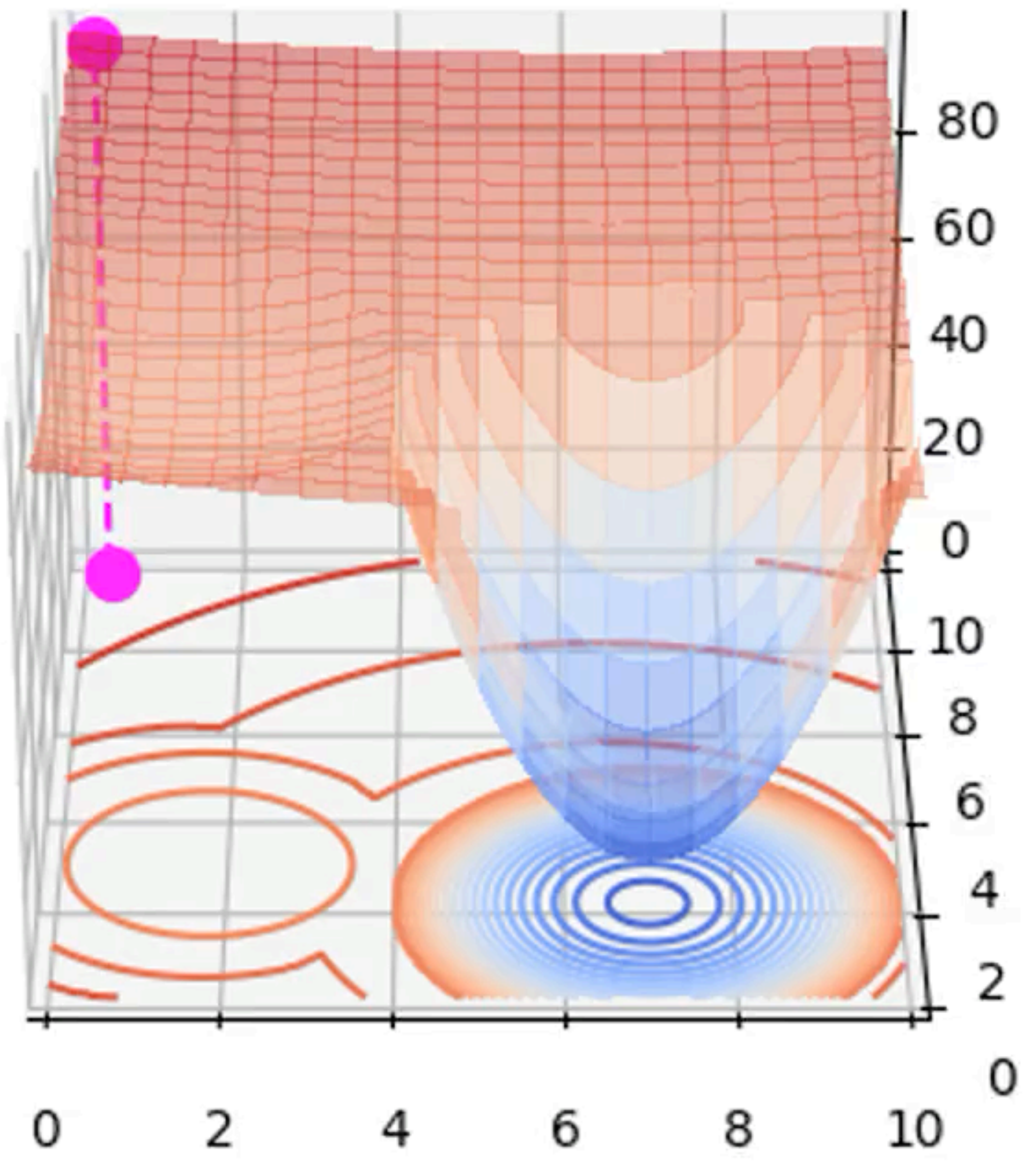


$$\rho(x, \alpha, c) = \frac{|\alpha - 2|}{\alpha}\left(\left(\frac{\left({}^{x}/_{c}\right)^2}{|\alpha - 2|} + 1\right)^{{}^{\alpha}/_{2}} - 1\right)$$

Robust loss with outlier

Is it a free lunch?

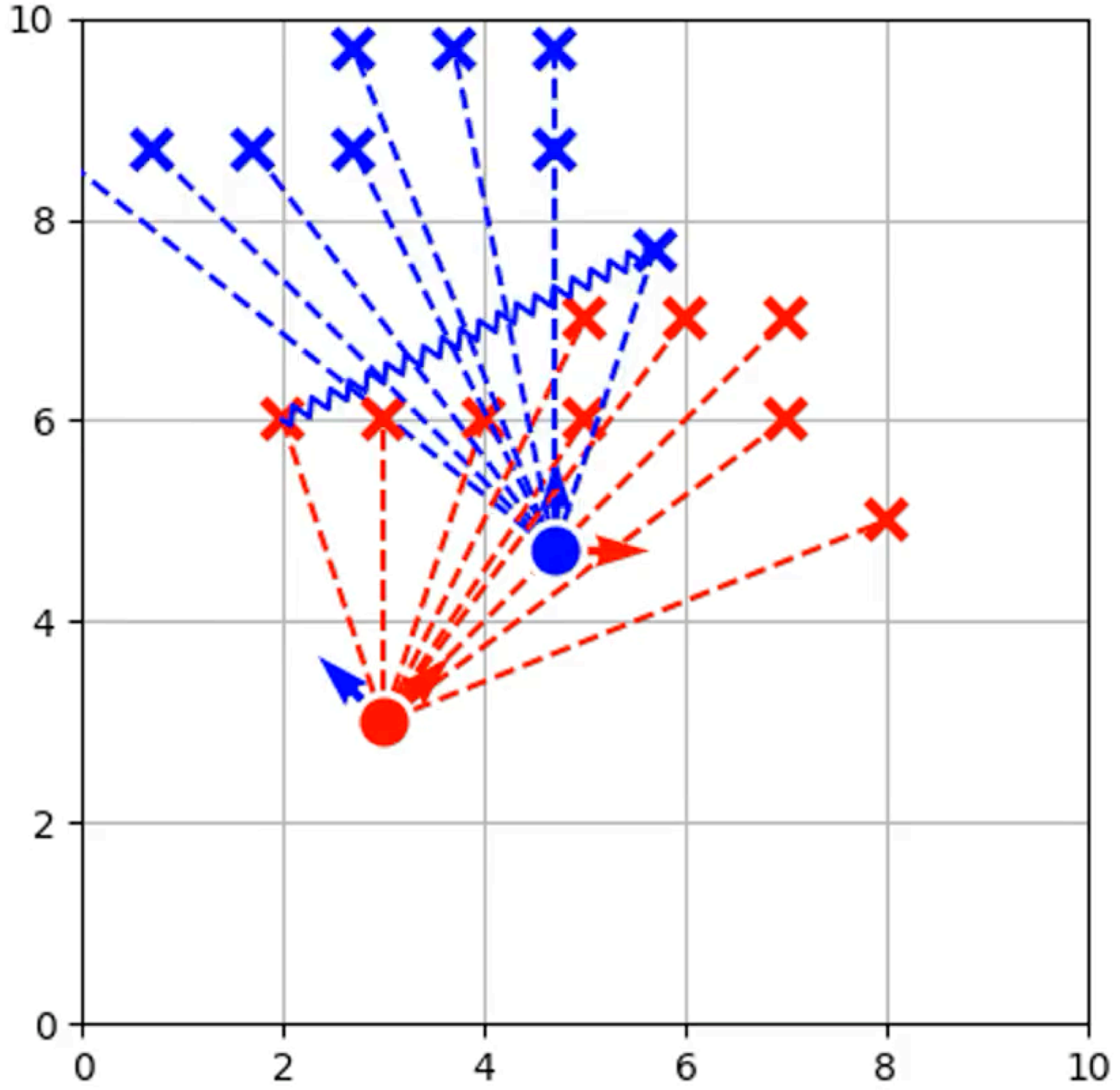


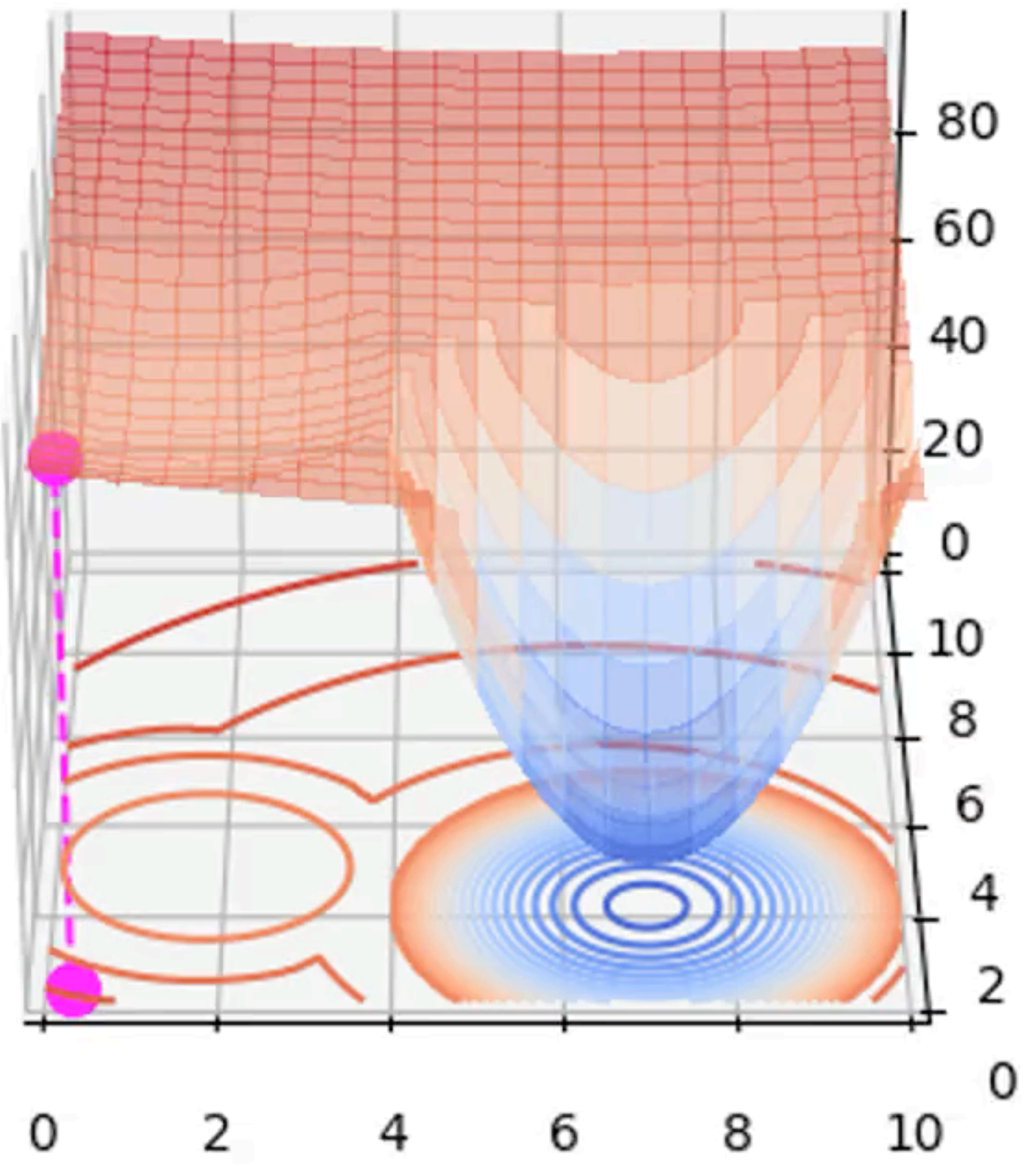


Correct result achieved

Robust loss with outlier

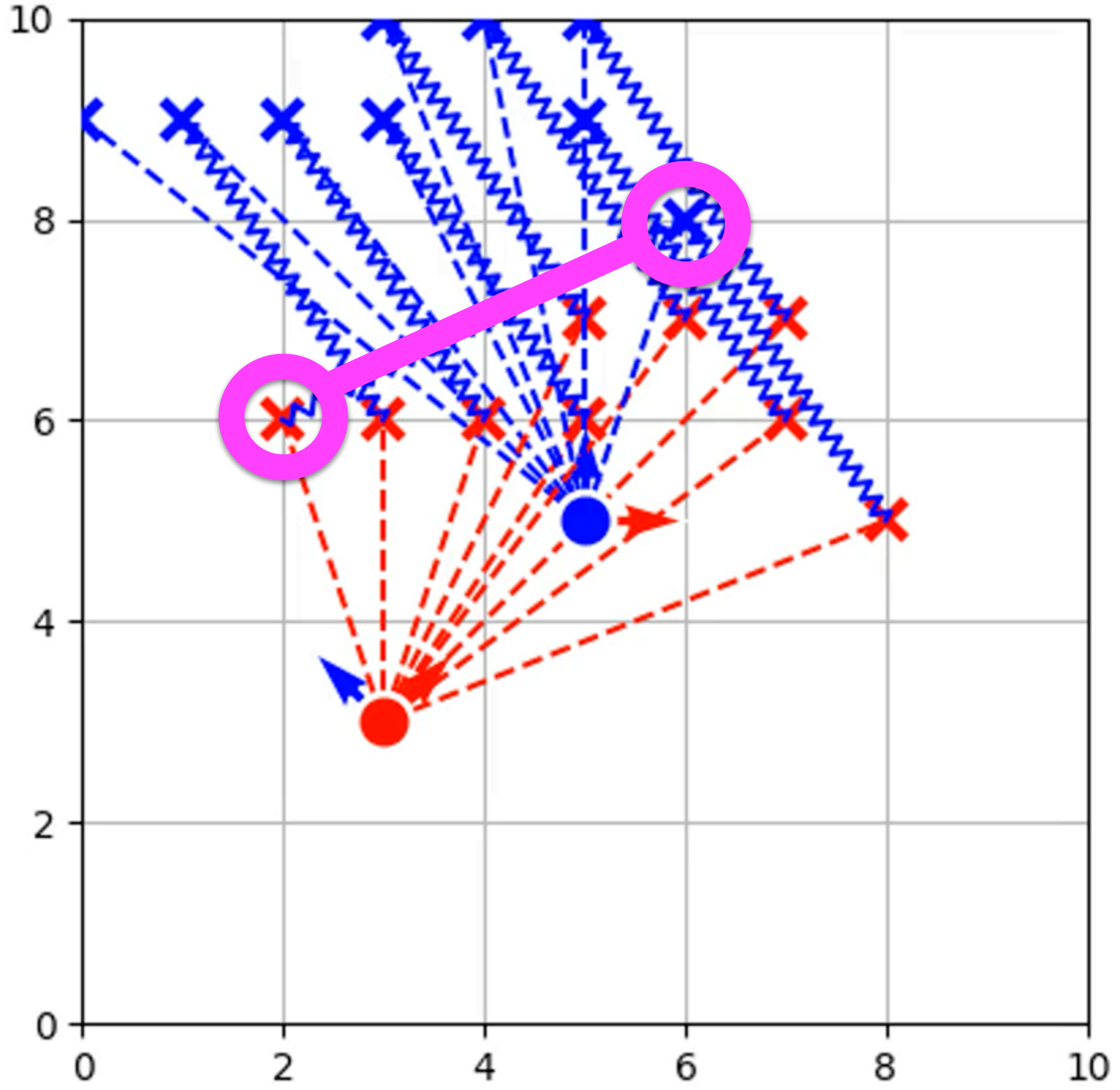
Let's push the loss into extreme

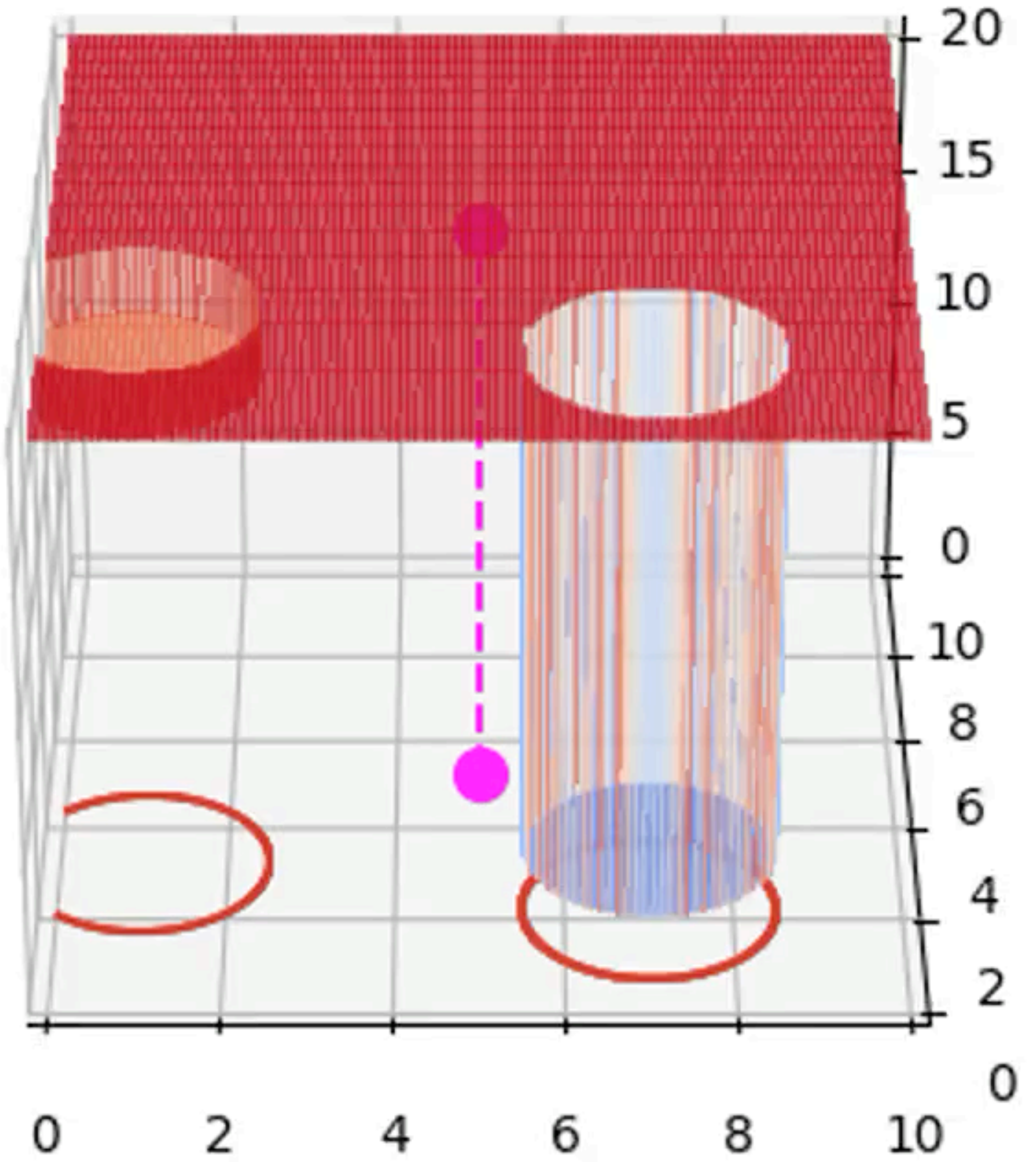




The prize we pay are local optima corresponding to outliers

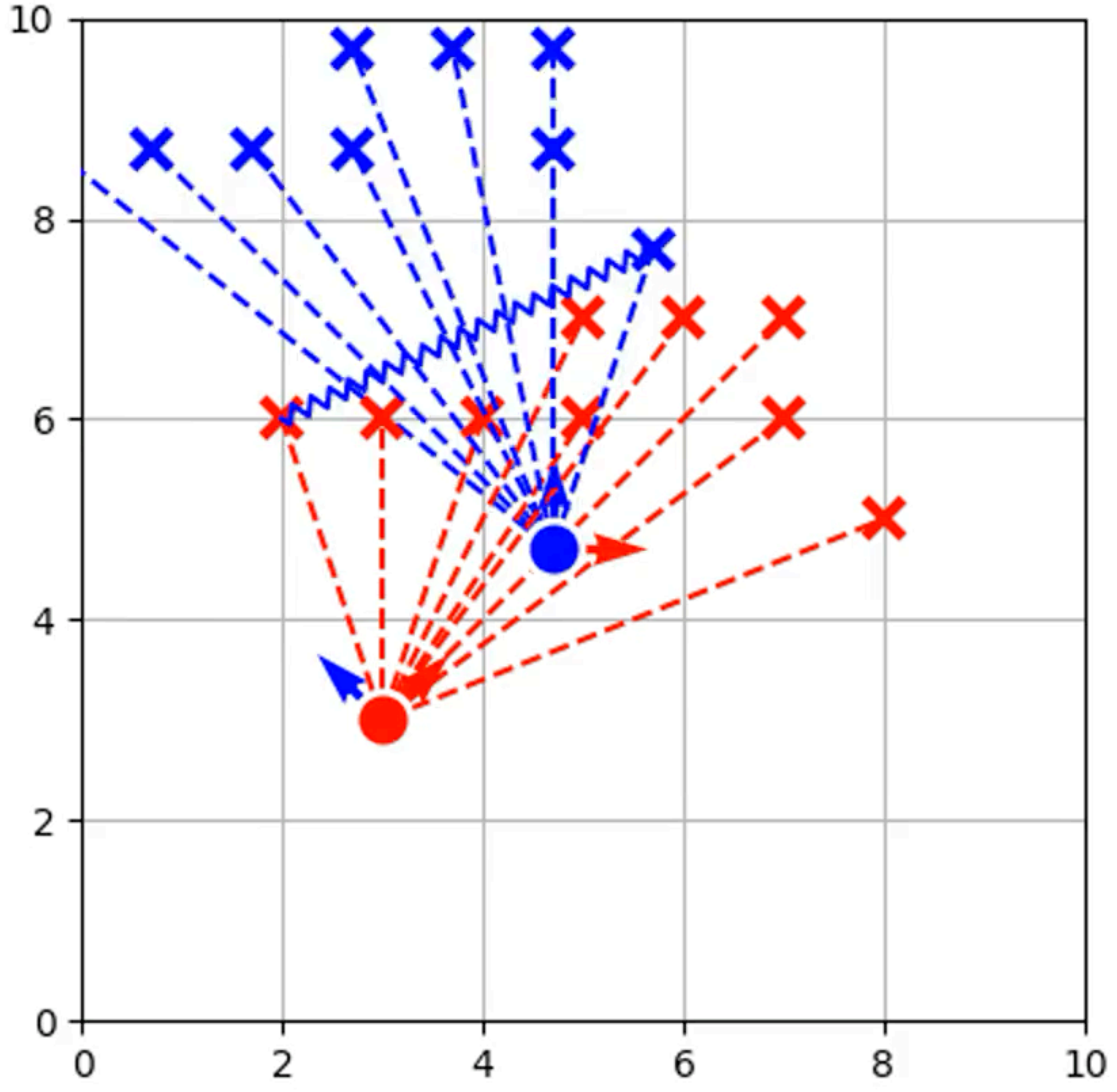
1. sample one tuple of corresponding points at random

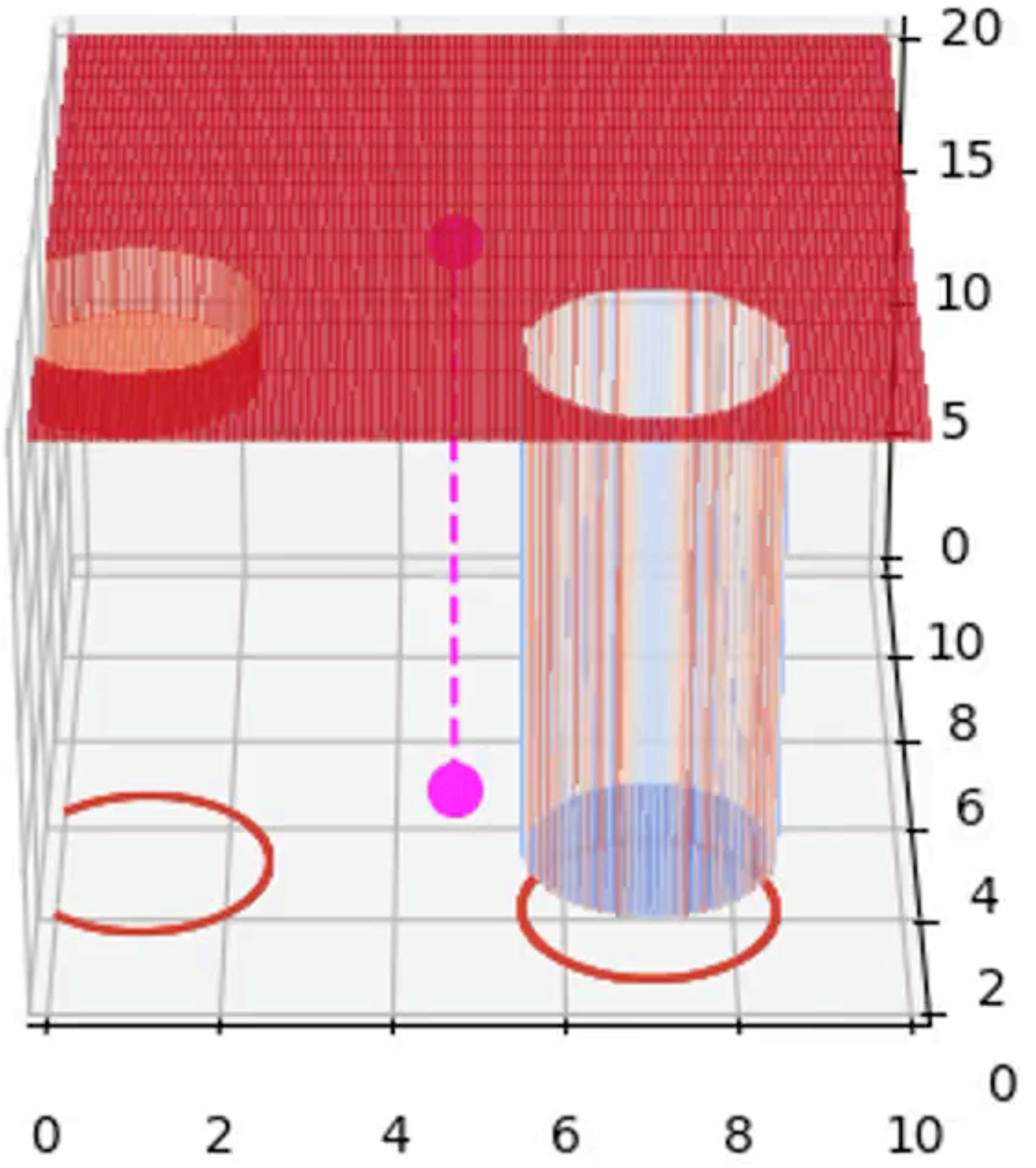




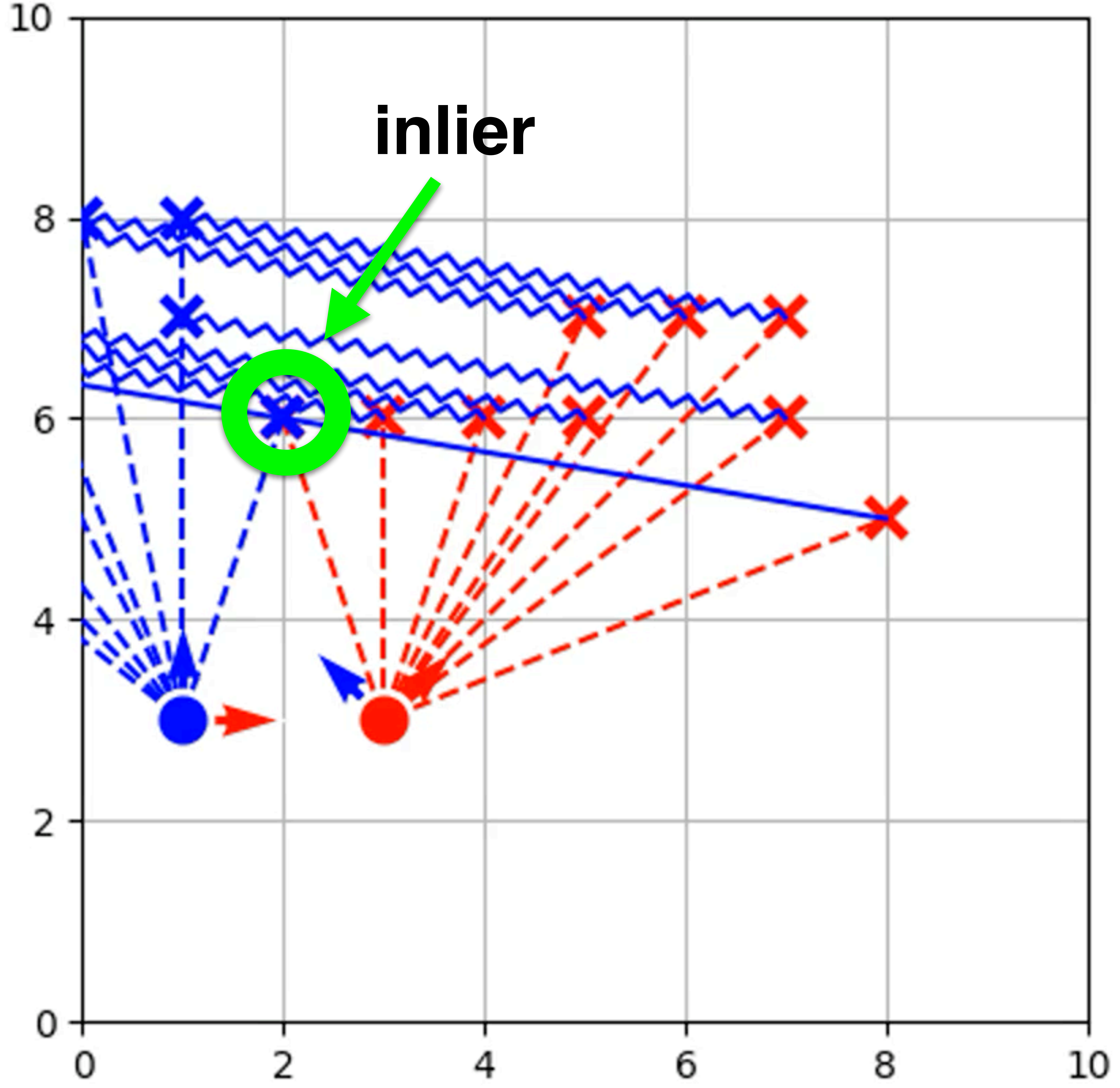
Is there any way to optimize it?

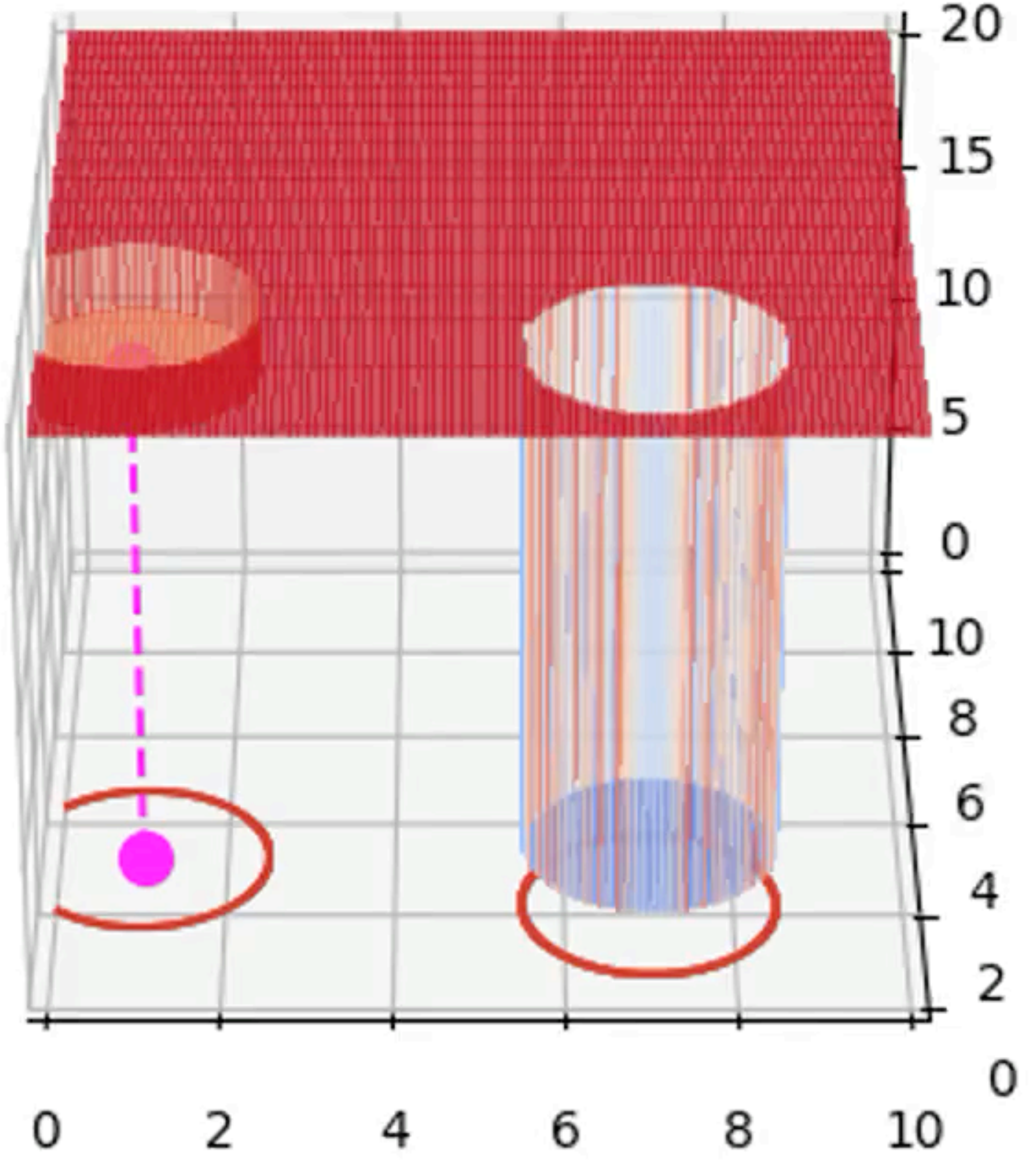
1. **sample one tuple of corresponding points at random**
2. **align it by minimizing L2-norm**



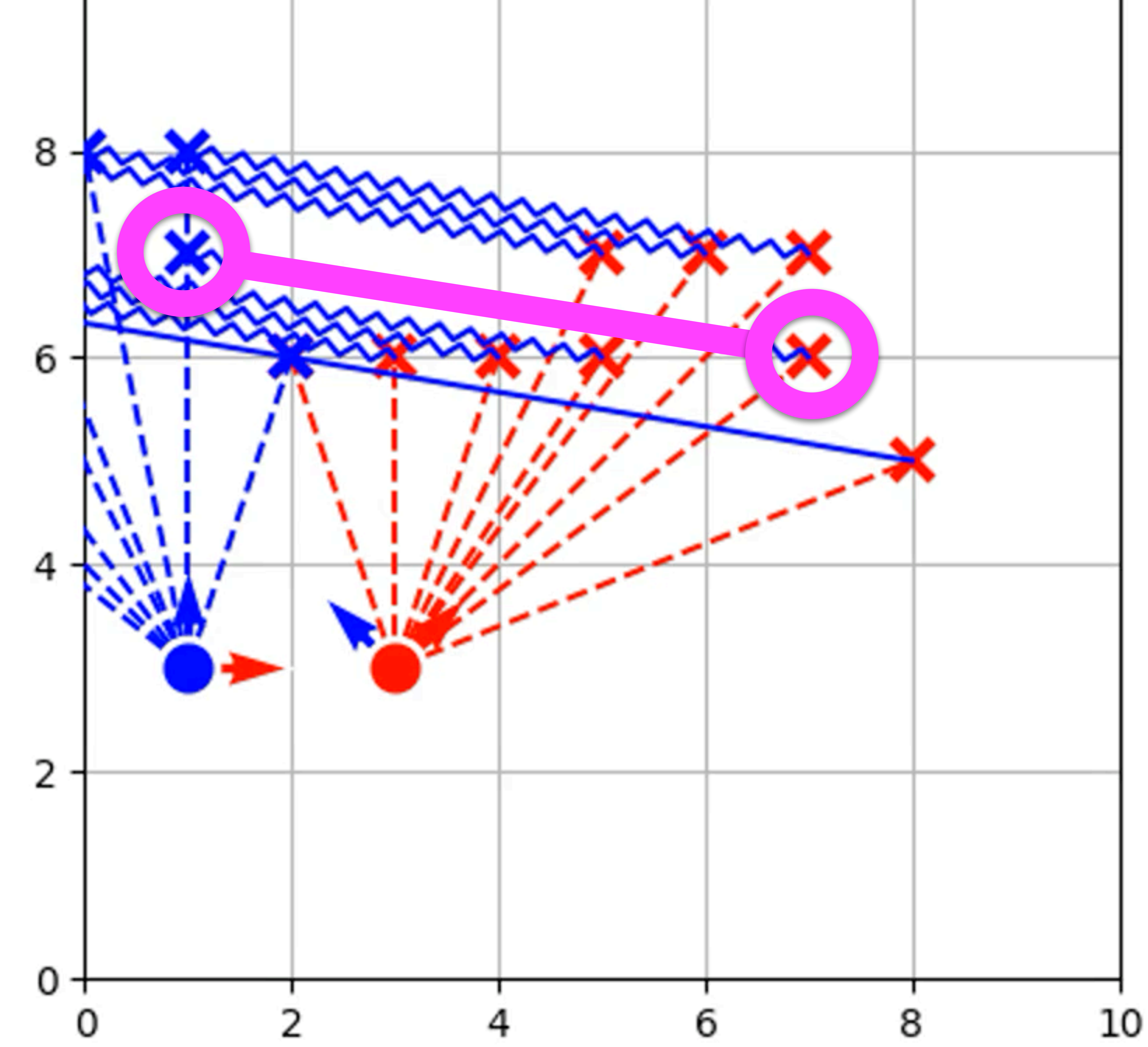


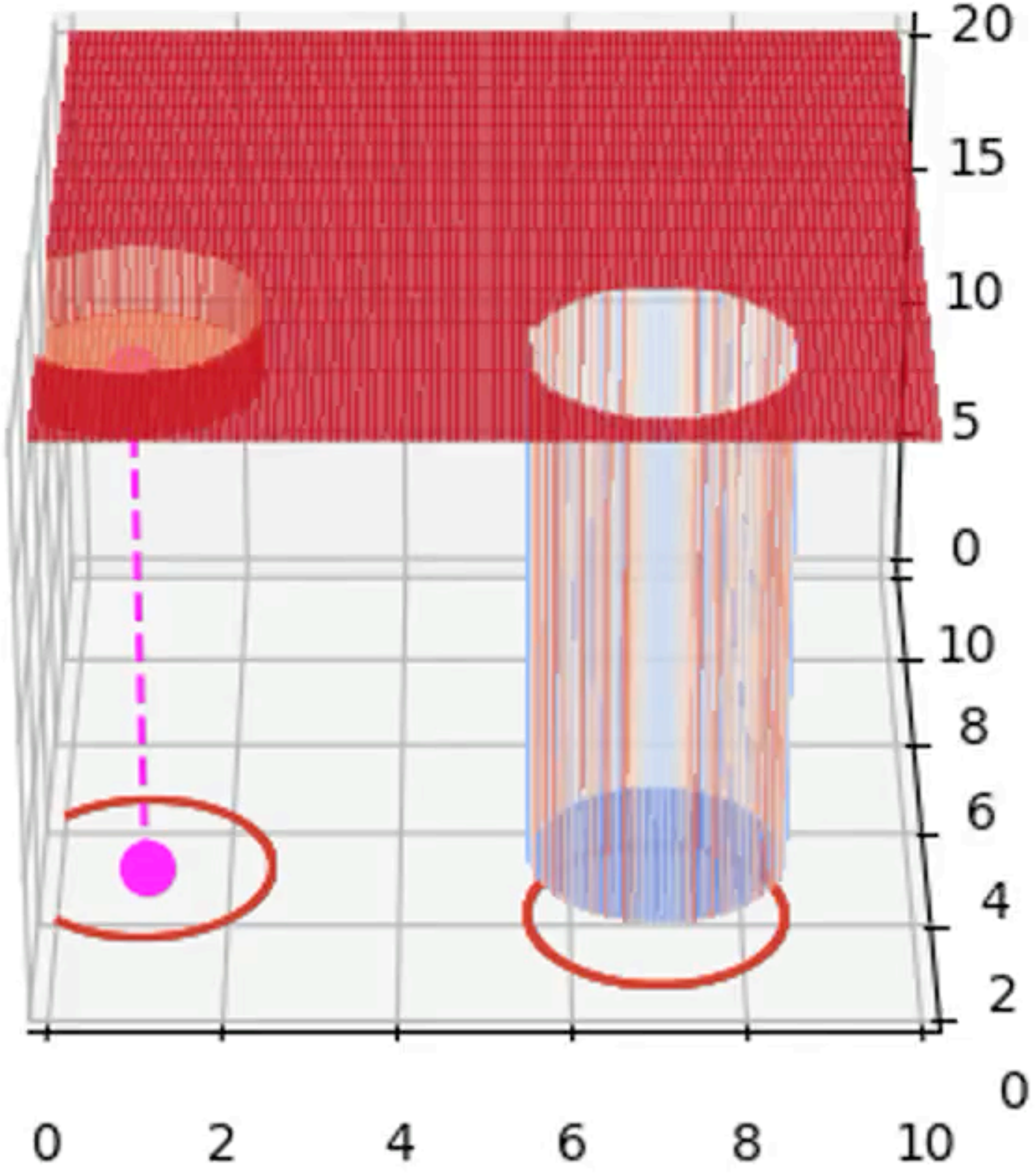
1. sample one tuple of corresponding points at random
2. align it by minimizing L2-norm
3. compute number of inliers for this hypothesis (inliers=1)



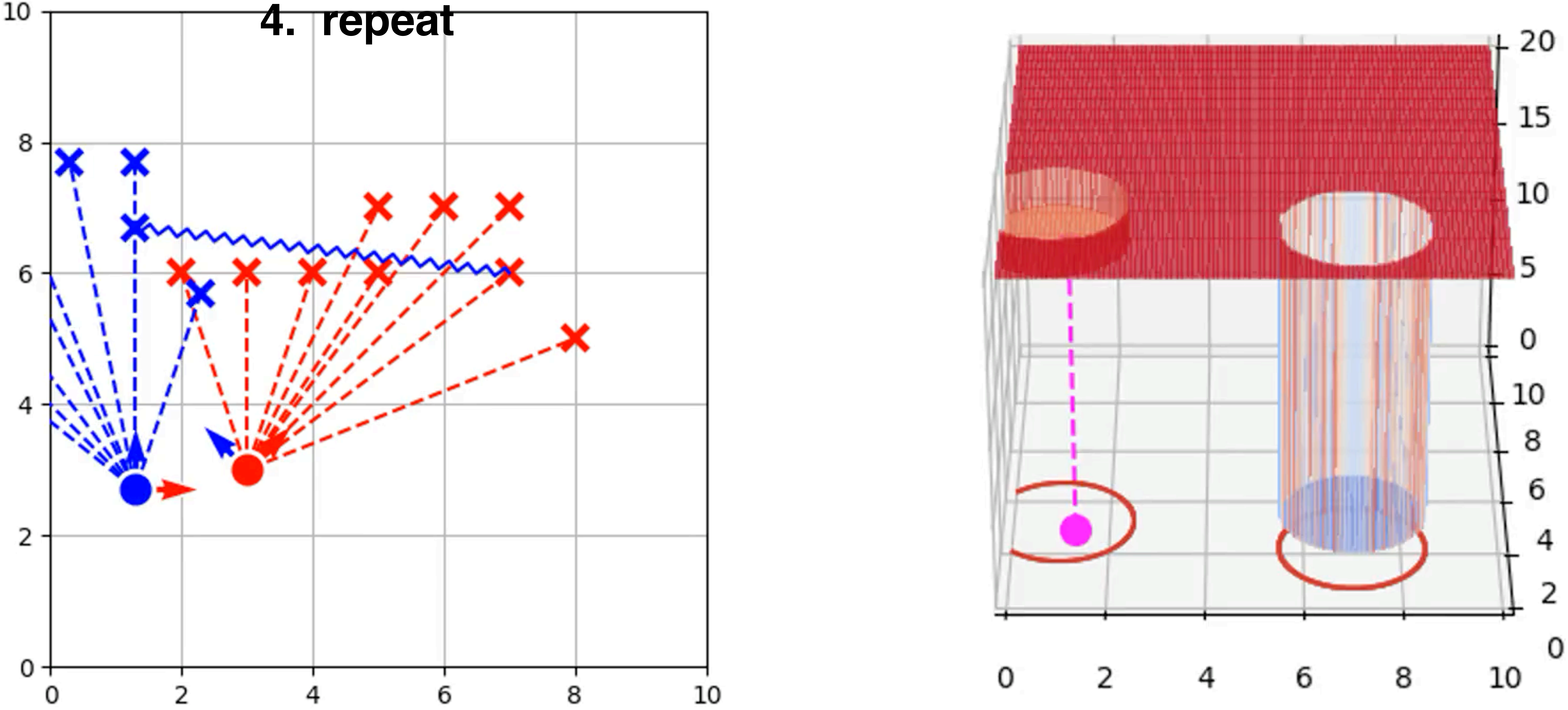


1. sample one tuple of corresponding points at random
2. align it by minimizing L2-norm
3. compute number of inliers for this hypothesis (inliers=1)
4. repeat

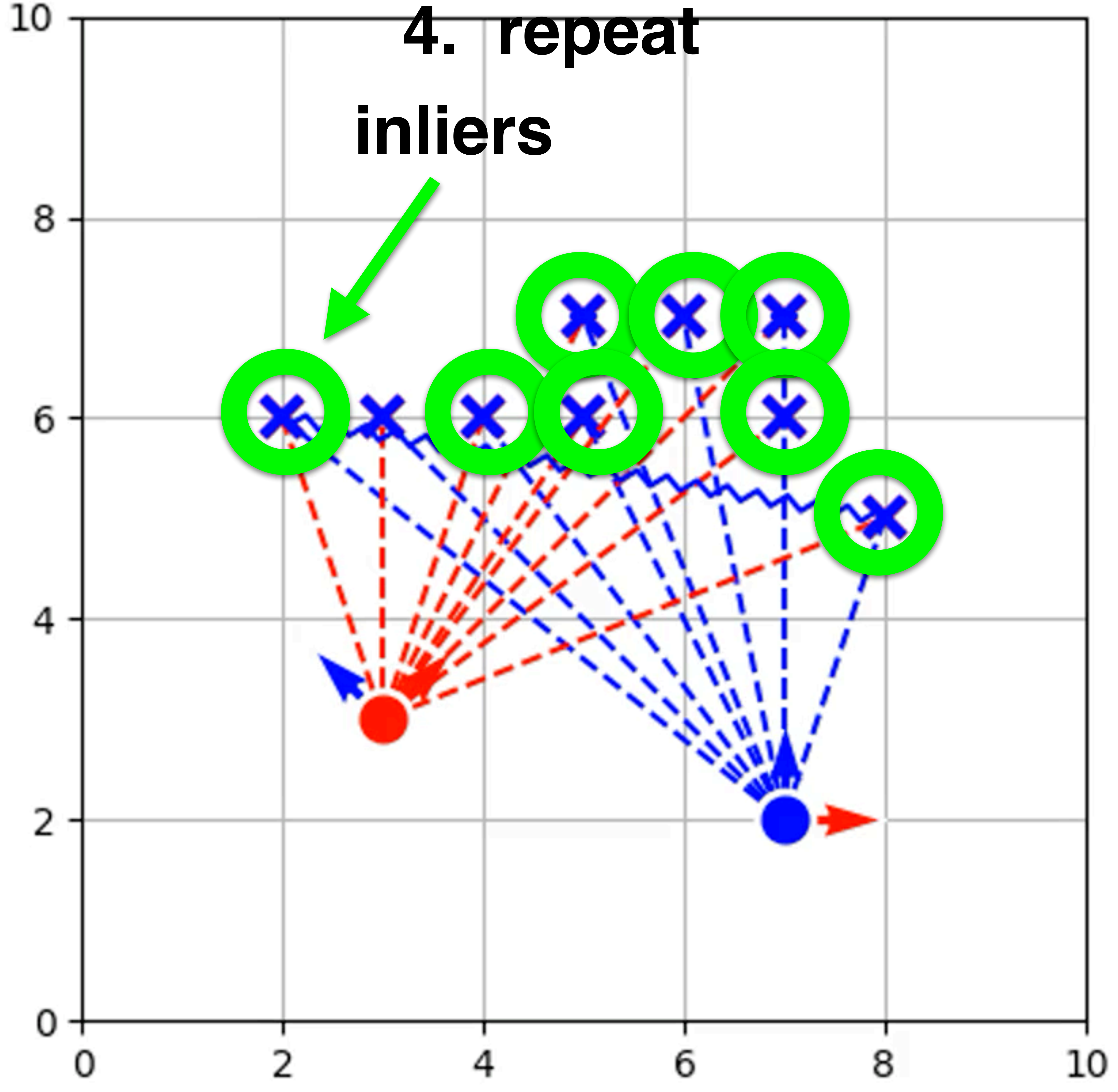


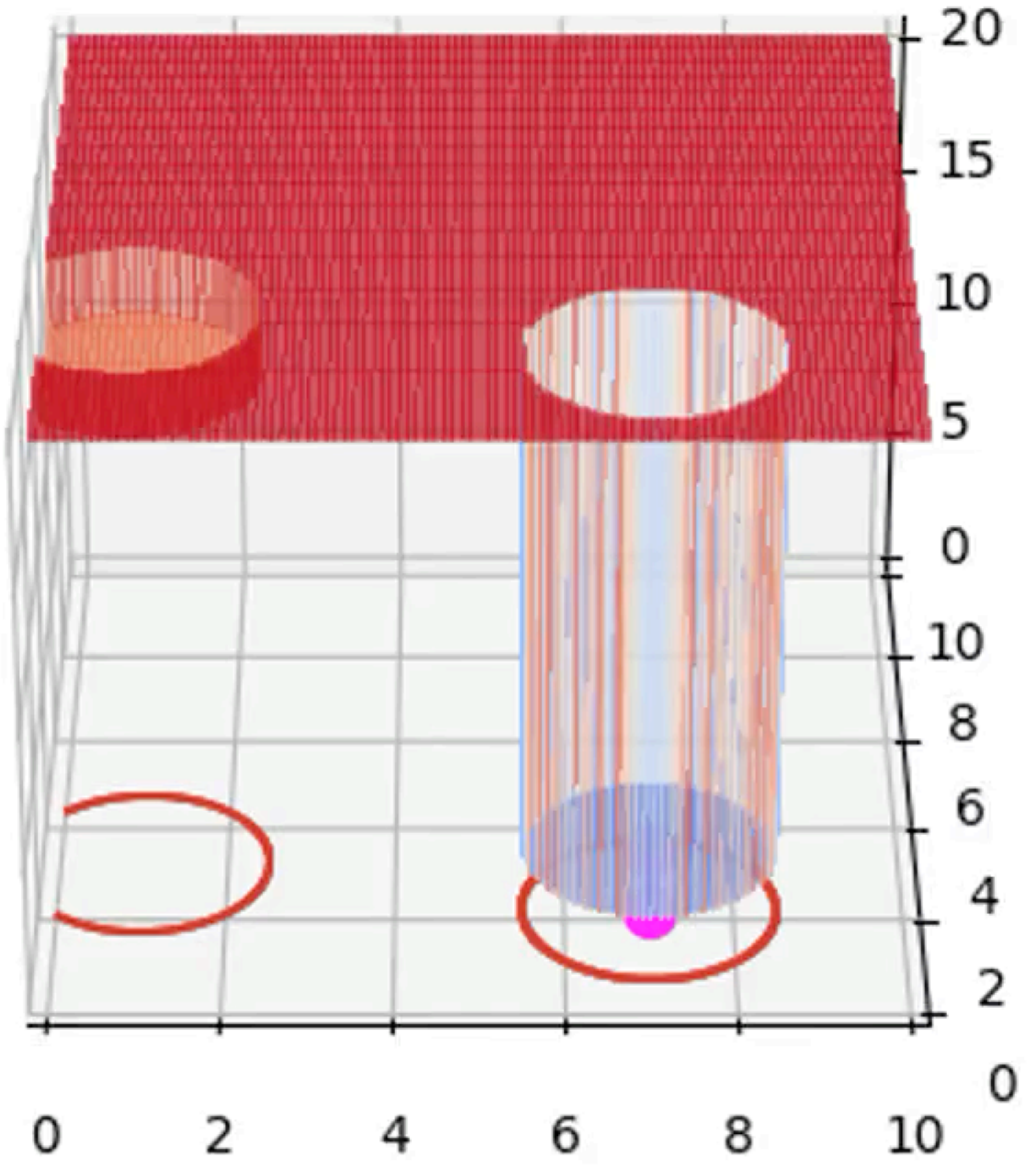


1. **sample one tuple of corresponding points at random**
2. **align it by minimizing L2-norm**
3. **compute number of inliers for this hypothesis (inliers=1)**
4. **repeat**

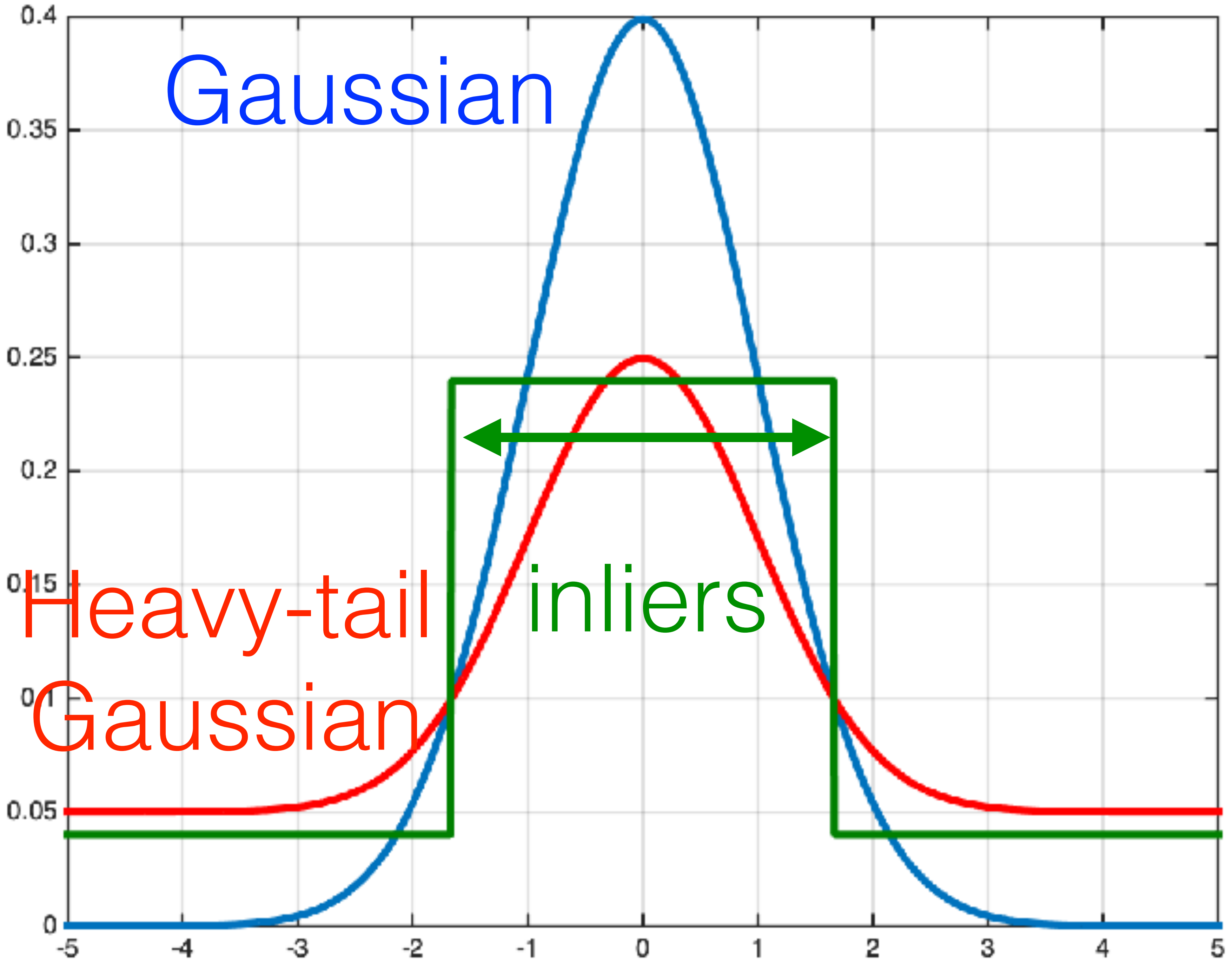


1. sample one tuple of corresponding points at random
2. align it by minimizing L2-norm
3. compute number of inliers for this hypothesis (inliers=8)
4. repeat

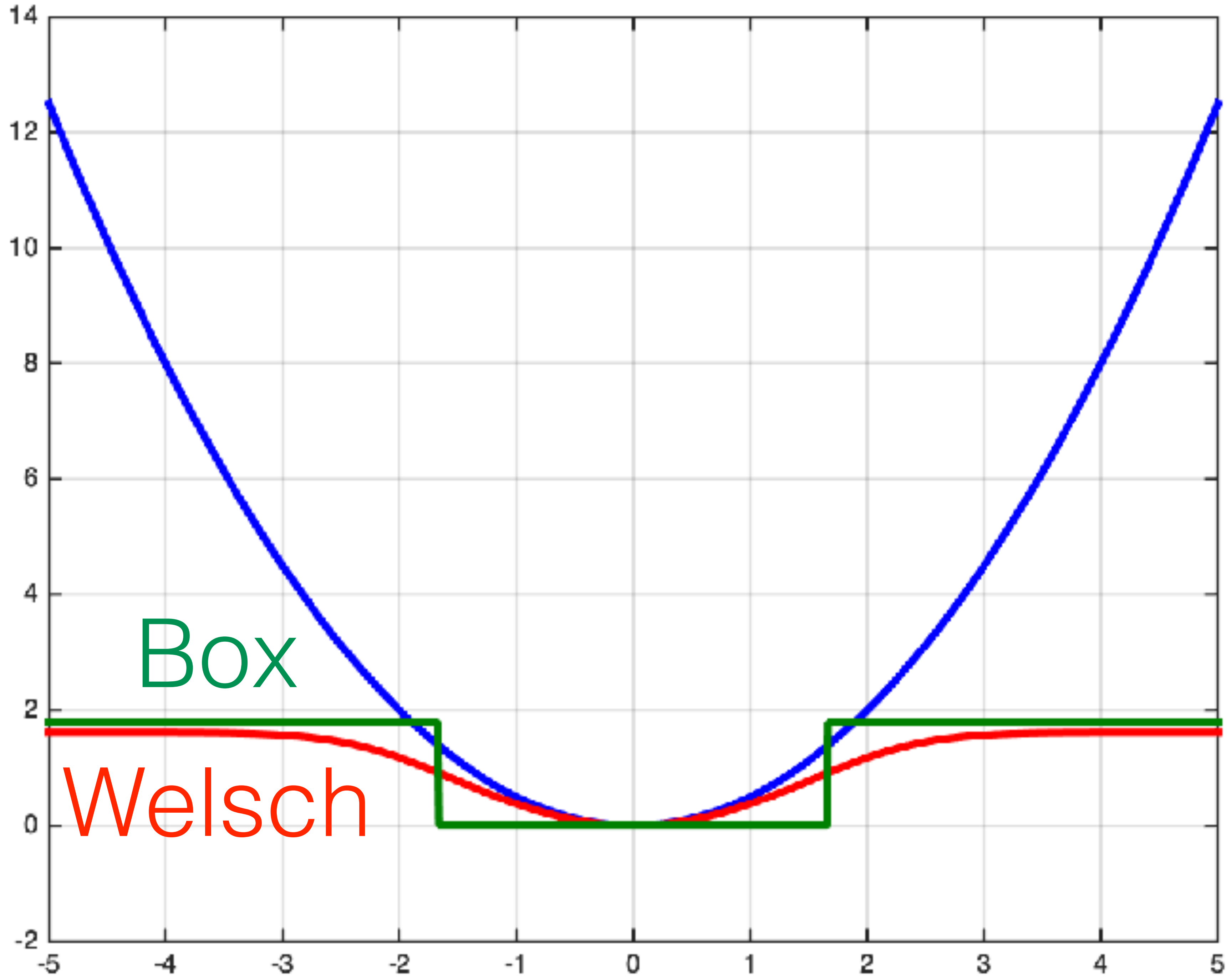




ICP SLAM - outlier detection procedure



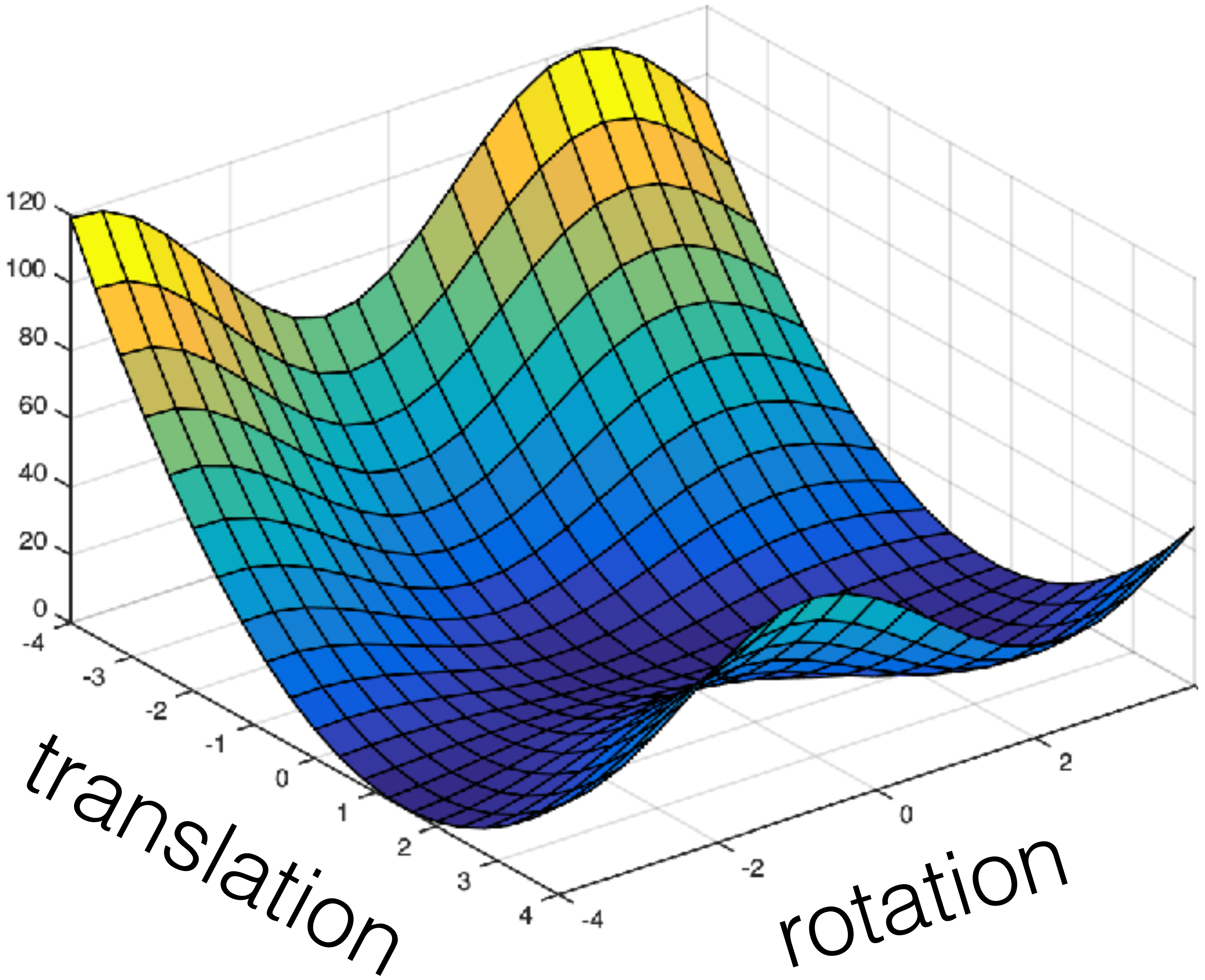
Probability distributions



Corresponding losses

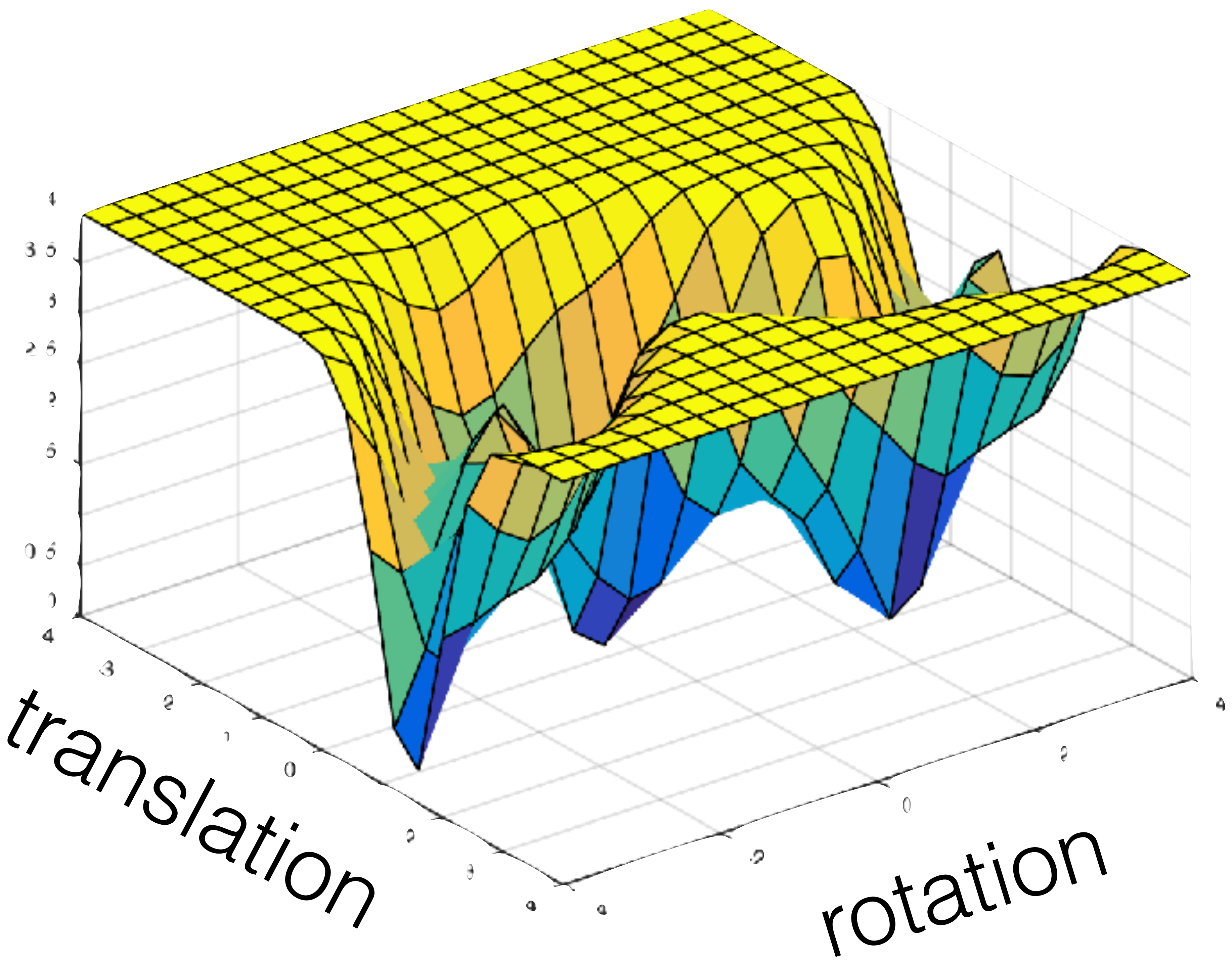
ICP SLAM - gradient optimisation of robust loss

L2 landscape



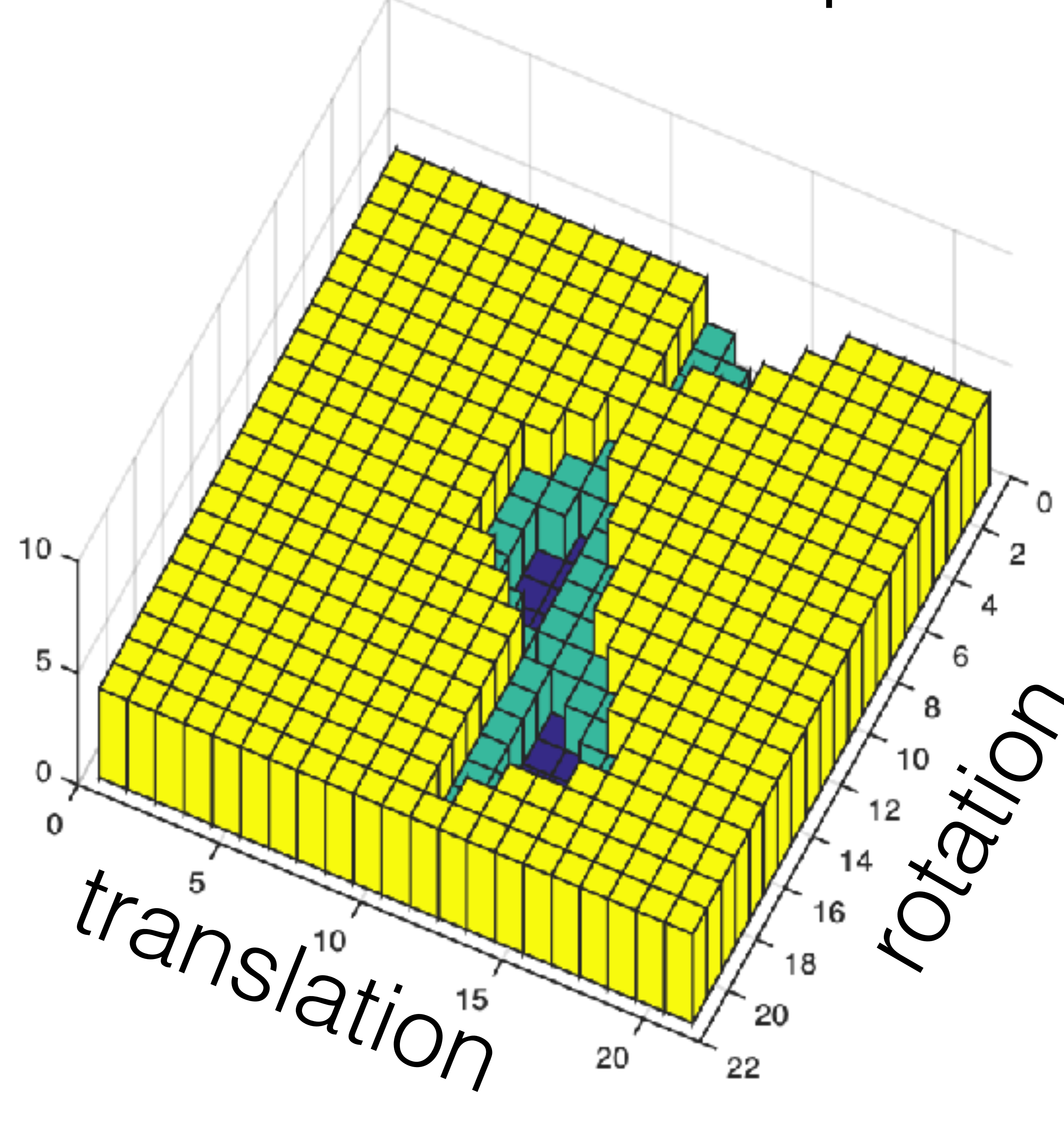
- Convex in translation space
- Non-convex but smooth in SO3

Welsch landscape



- Non-convex+Large narrow plateaus with zero gradient
- Any gradient optimization requires good initialization

Box loss landscape

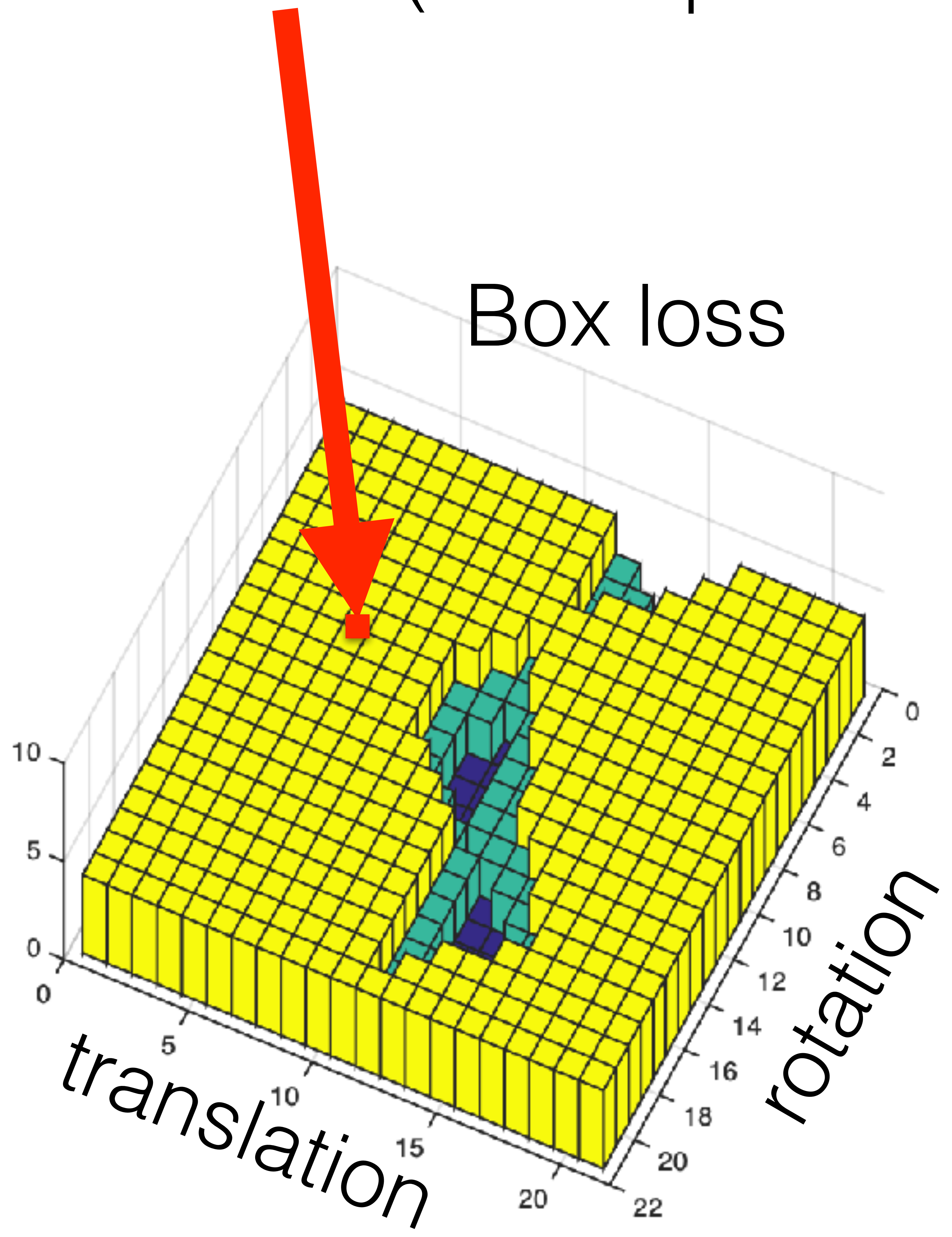


- Zero gradients
- Combinatorial optimization

Optimizing box-loss

Naive optimization algorithm:

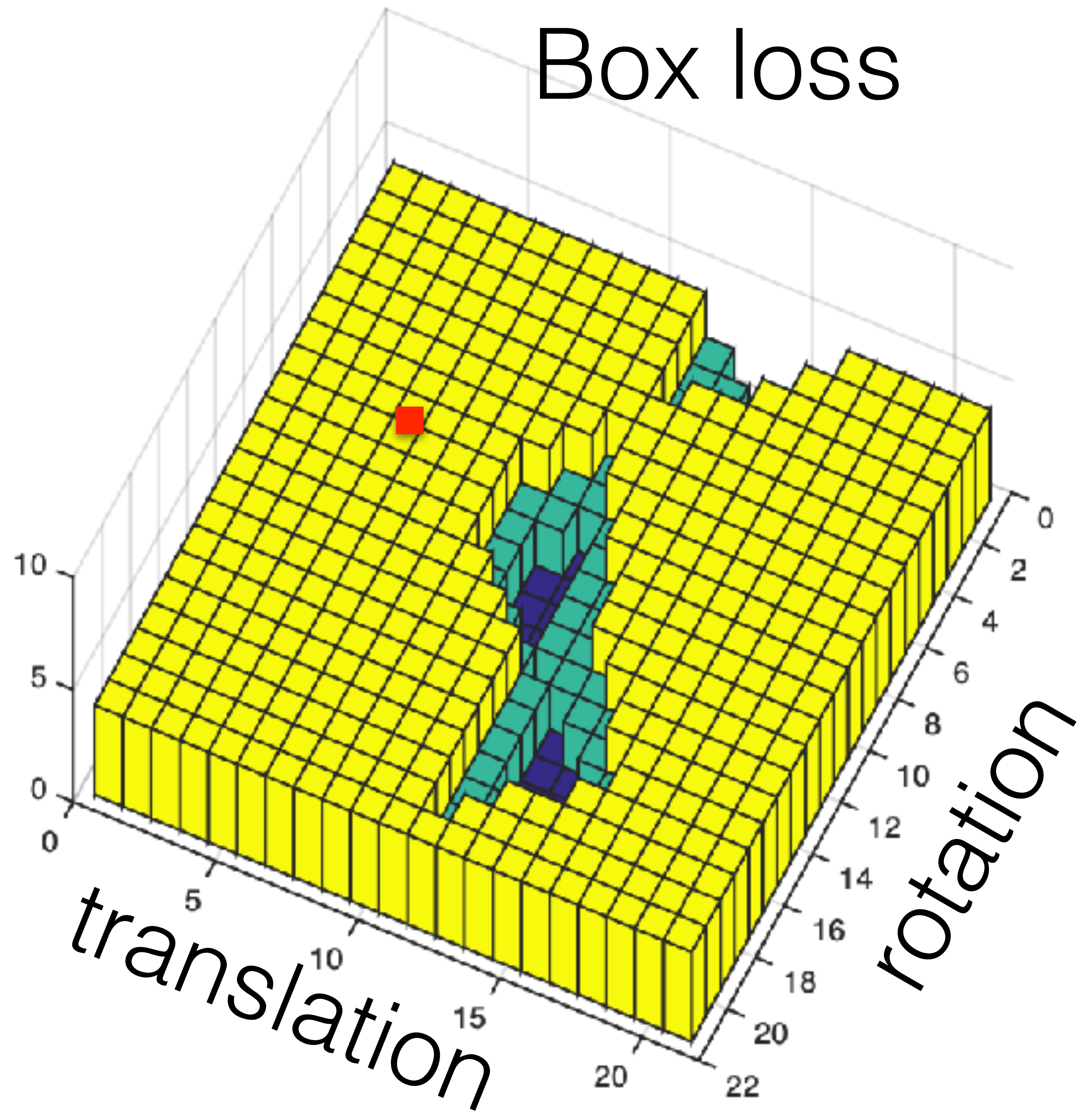
1. Sample hypothesis (R,t) at random
2. Evaluate value of the box-loss function (at this point R,t)



Optimizing box-loss

Naive optimization algorithm:

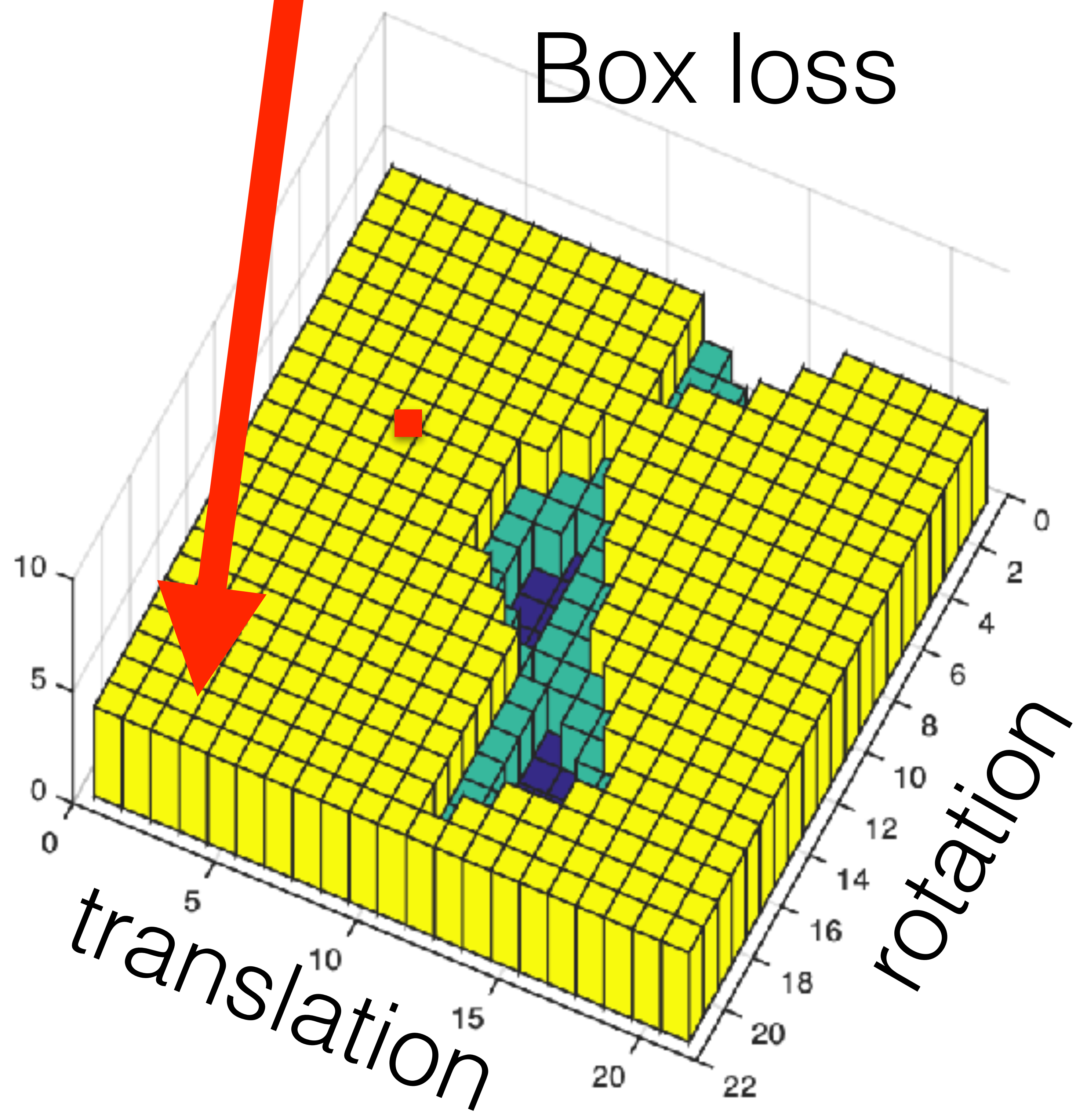
1. Sample hypothesis (R,t) at random
2. Evaluate value of the box-loss function (at this point R,t)
3. Remember the lowest value so far
4. repeat K times



Optimizing box-loss

Naive optimization algorithm:

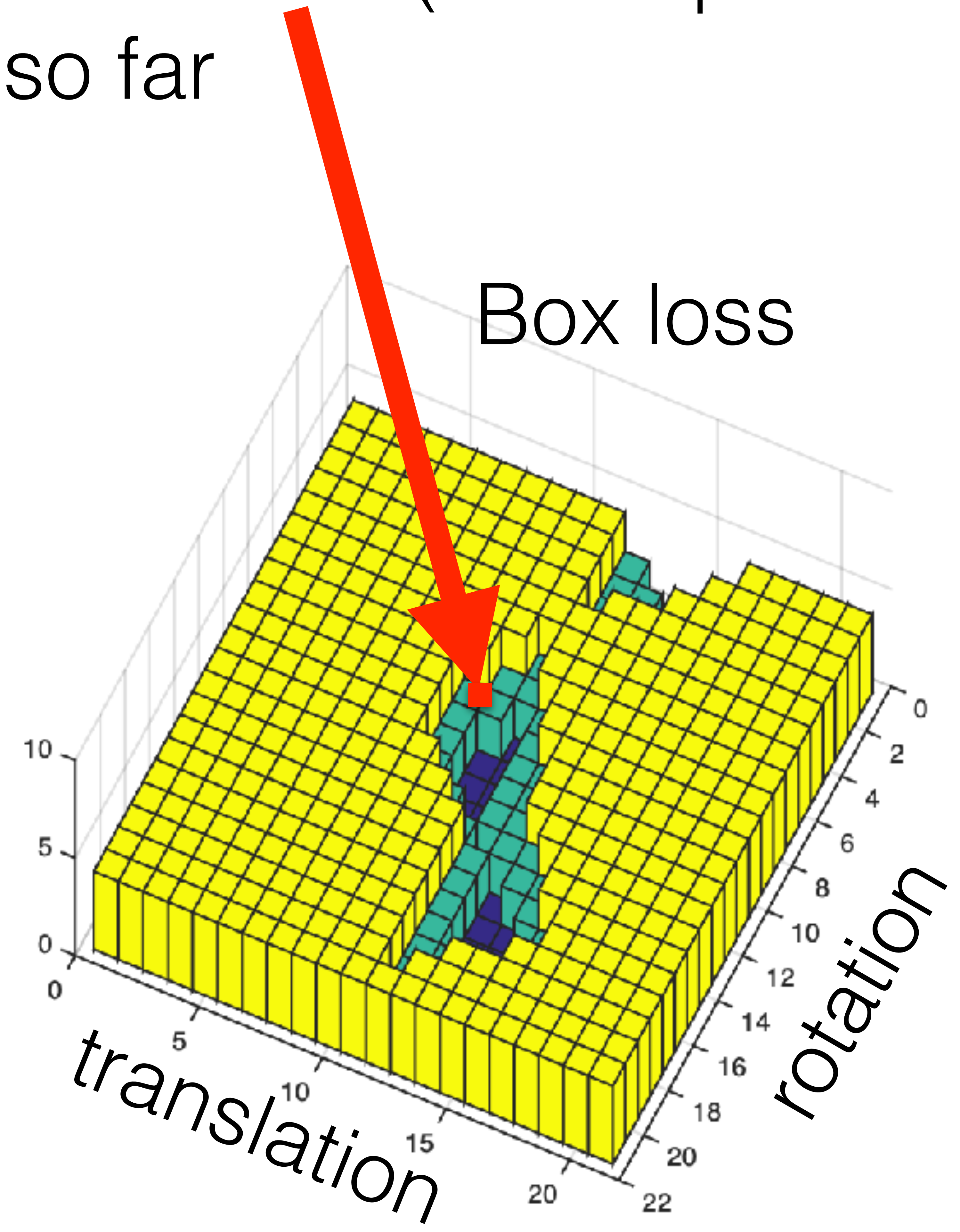
1. Sample hypothesis (R,t) at random
2. Evaluate value of the box-loss function (at this point R,t)
3. Remember the lowest value so far
4. repeat K times



Optimizing box-loss

Naive optimization algorithm:

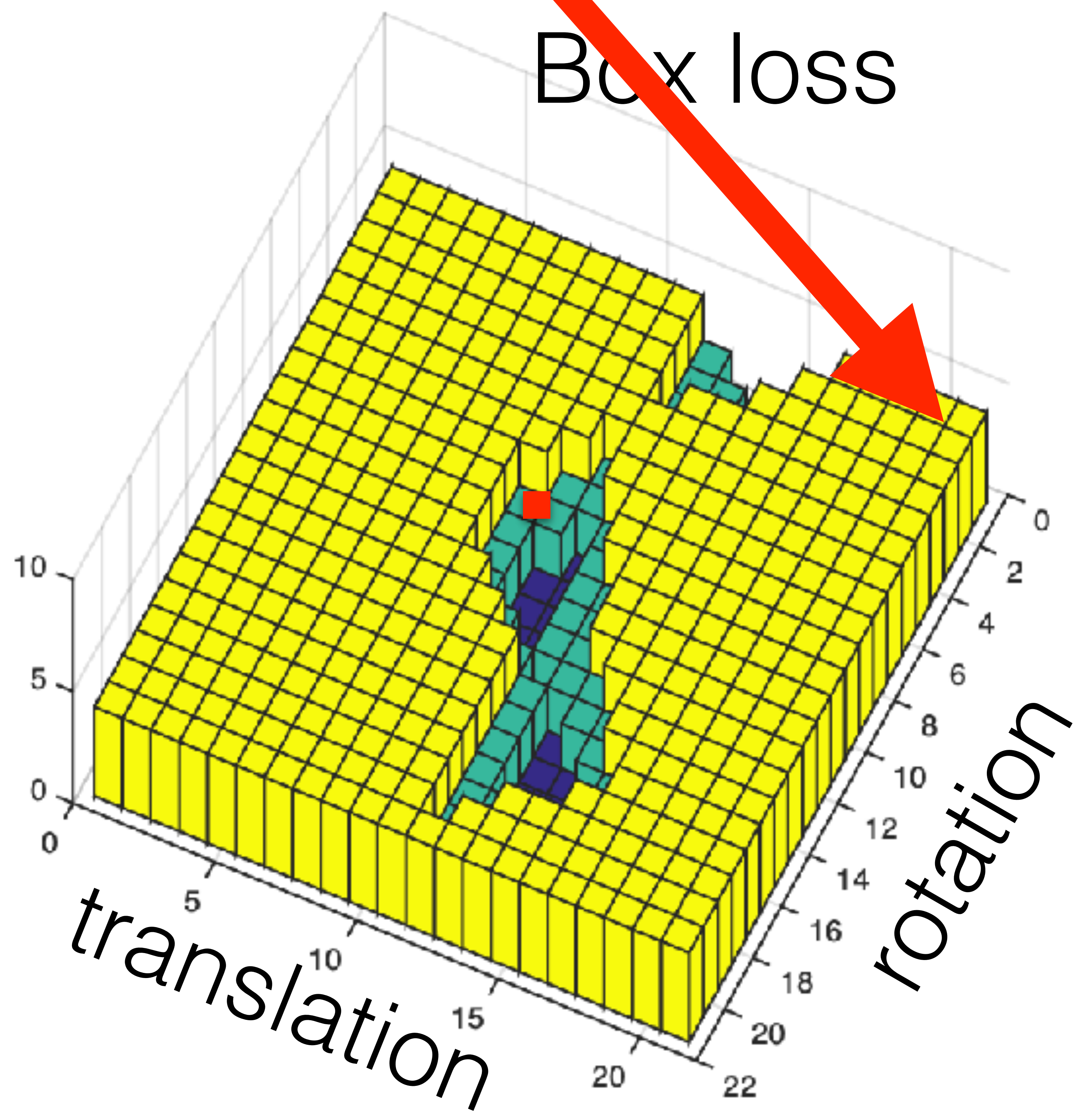
1. Sample hypothesis (R,t) at random
2. Evaluate value of the box-loss function (at this point R,t)
3. Remember the lowest value so far
4. repeat K times



Optimizing box-loss

Naive optimization algorithm:

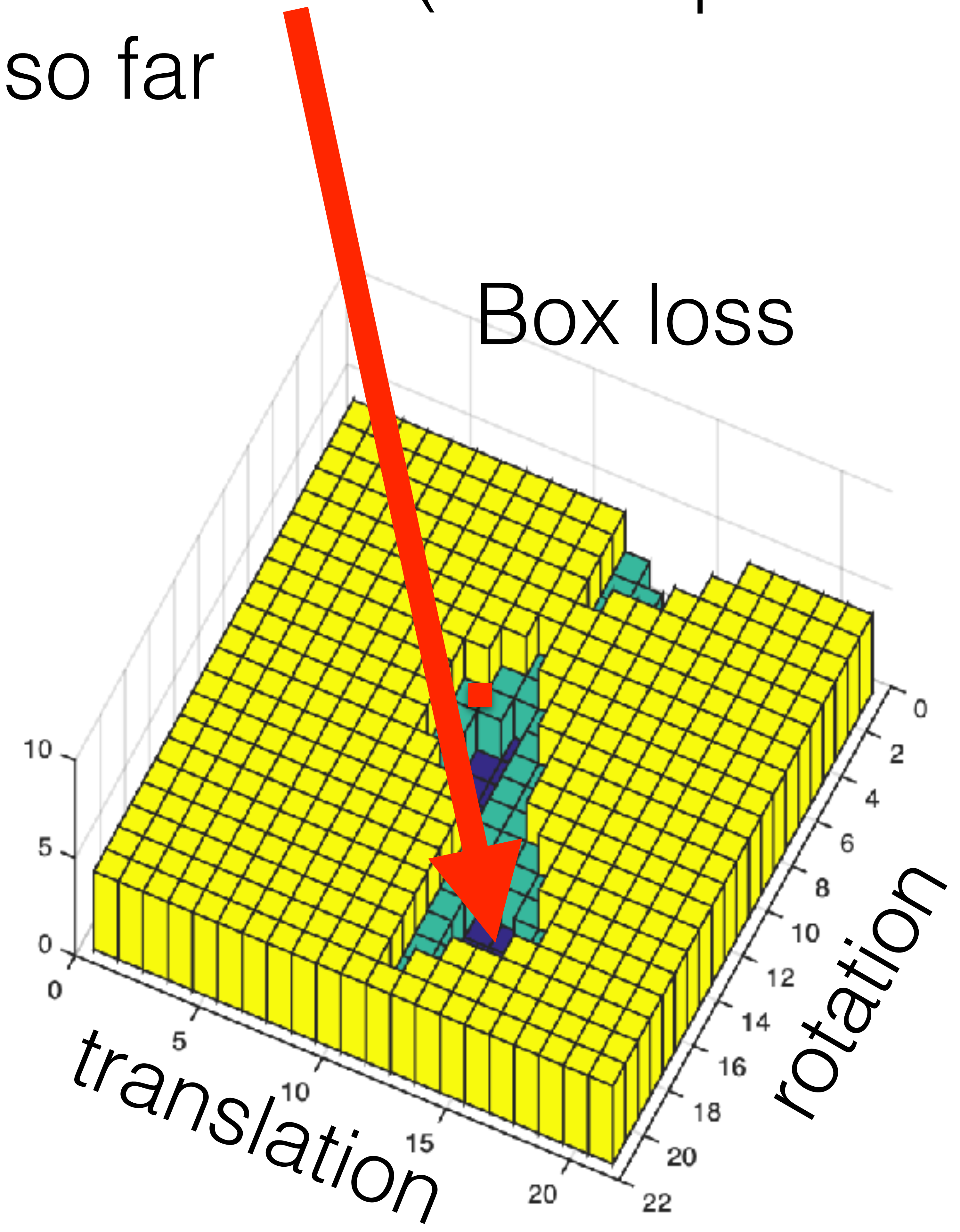
1. Sample hypothesis (R,t) at random
2. Evaluate value of the box-loss function (at this point R,t)
3. Remember the lowest value so far
4. repeat K times



Optimizing box-loss

Naive optimization algorithm:

1. Sample hypothesis (R,t) at random
2. Evaluate value of the box-loss function (at this point R,t)
3. Remember the lowest value so far
4. repeat K times

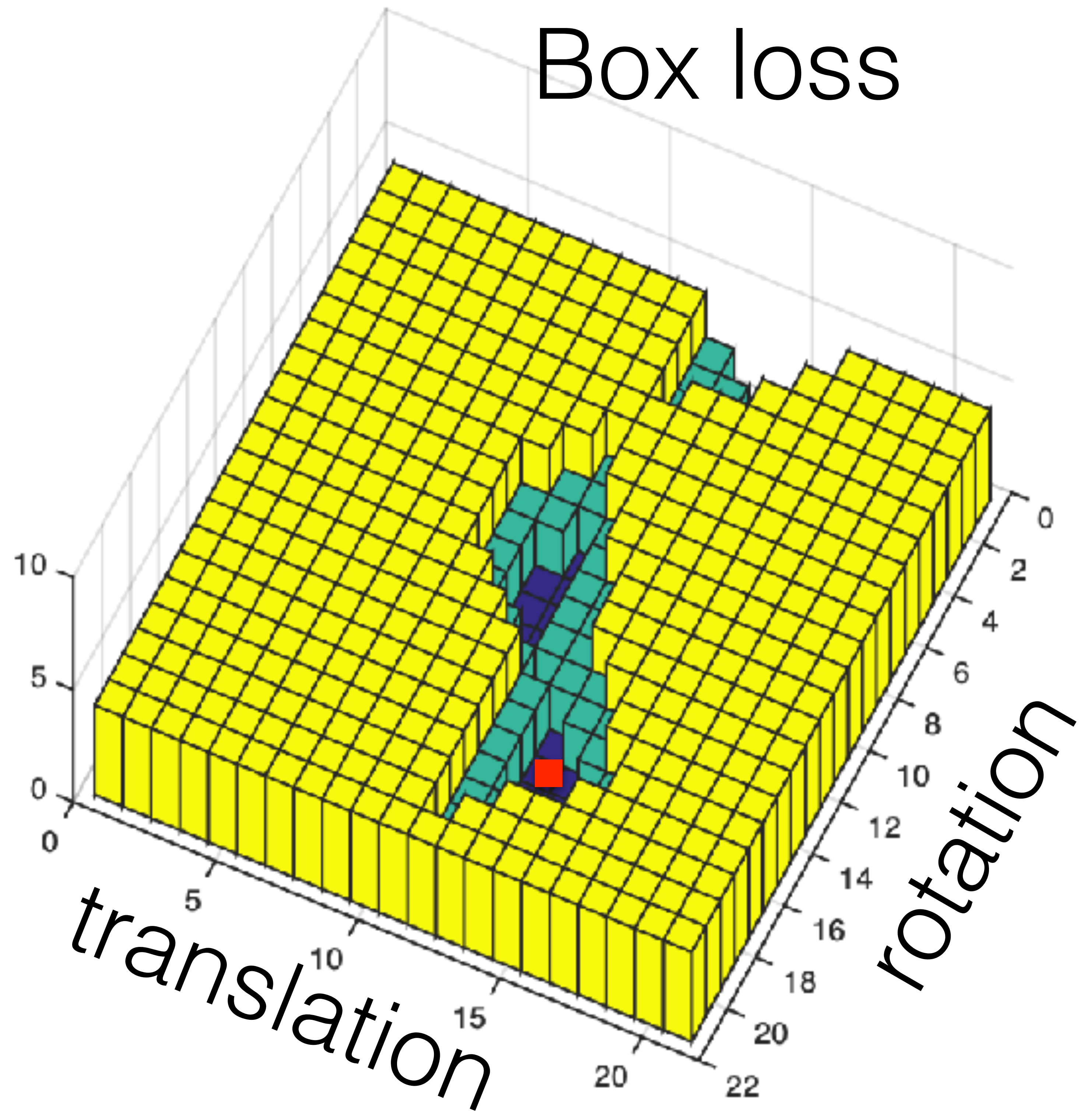


Optimizing box-loss

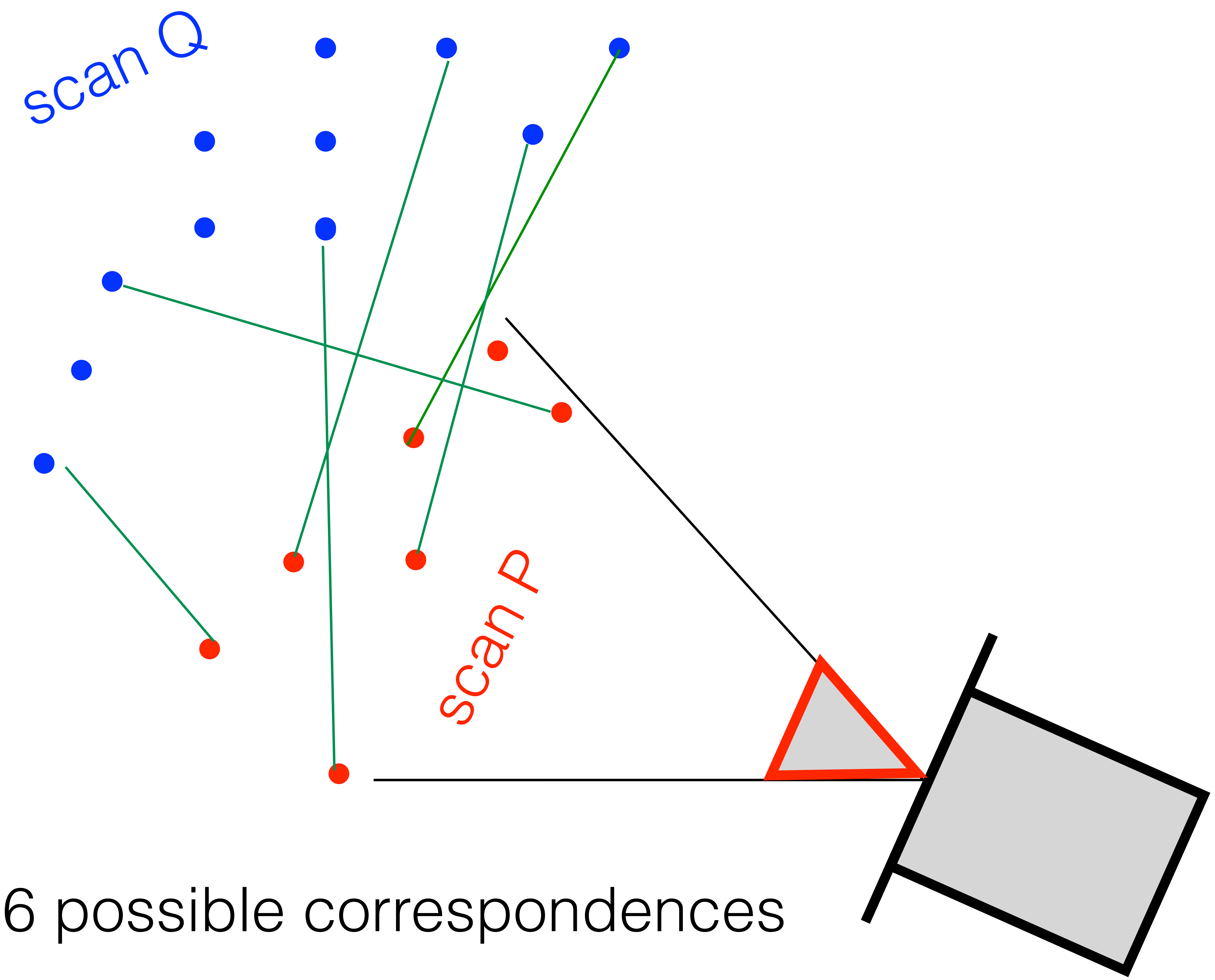
Naive optimization algorithm:

1. Sample hypothesis (R,t) at random
2. Evaluate value of the box-loss function (at this point R,t)
3. Remember the lowest value so far
4. repeat K times

if K is huge and you are lucky



RANSAC (RAndom SAmple Consensus)
Minimize number of outliers

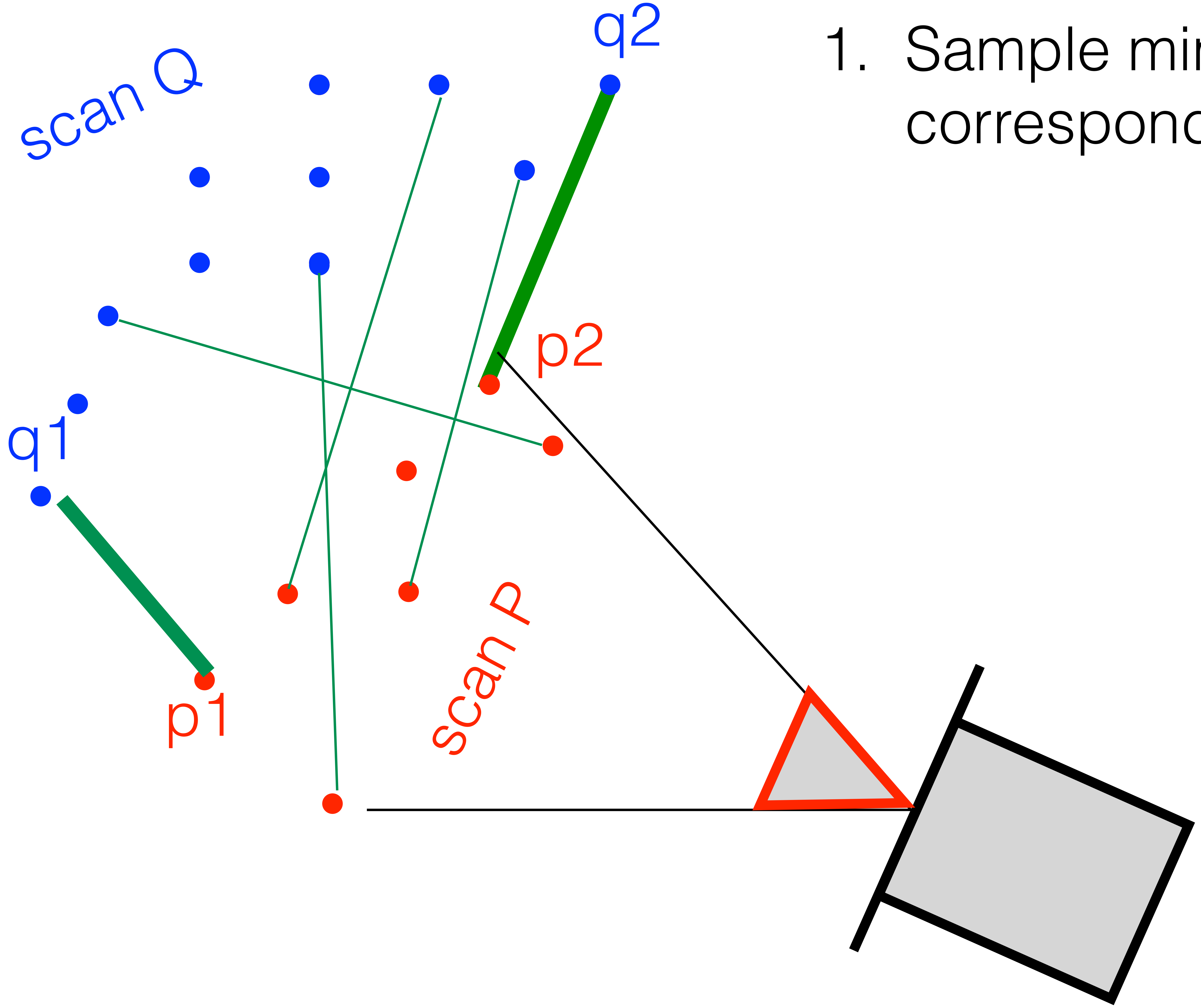


6 possible correspondences

N … total number of correspondences (N=6)

RANSAC (RAndom SAmple Consensus)
Minimize number of outliers

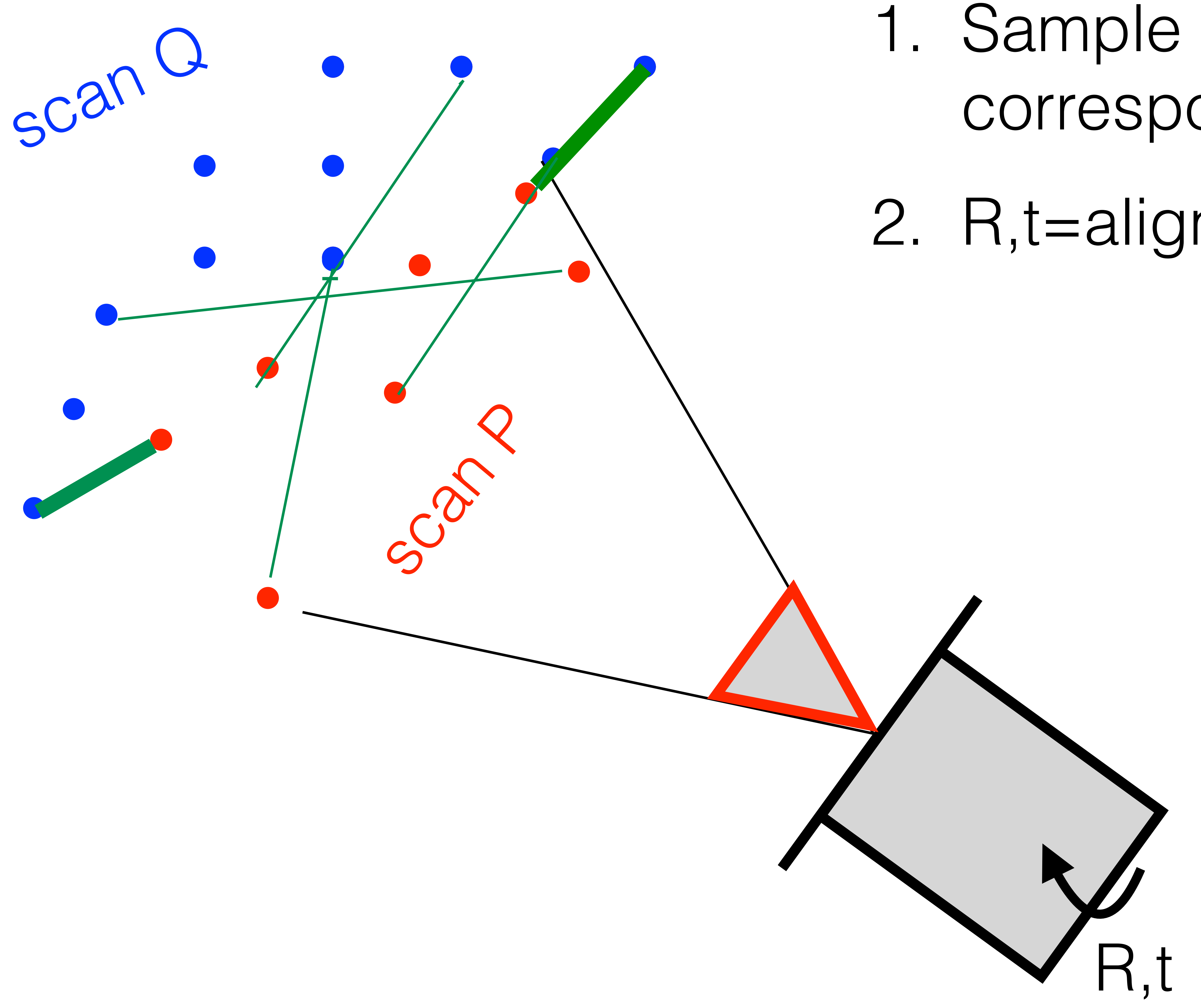
1. Sample minimal subset of correspondences (p, q).



s … size of |S| (s=2)

N … total number of correspondences (N=6)

RANSAC (RAndom SAmple Consensus)
Minimize number of outliers

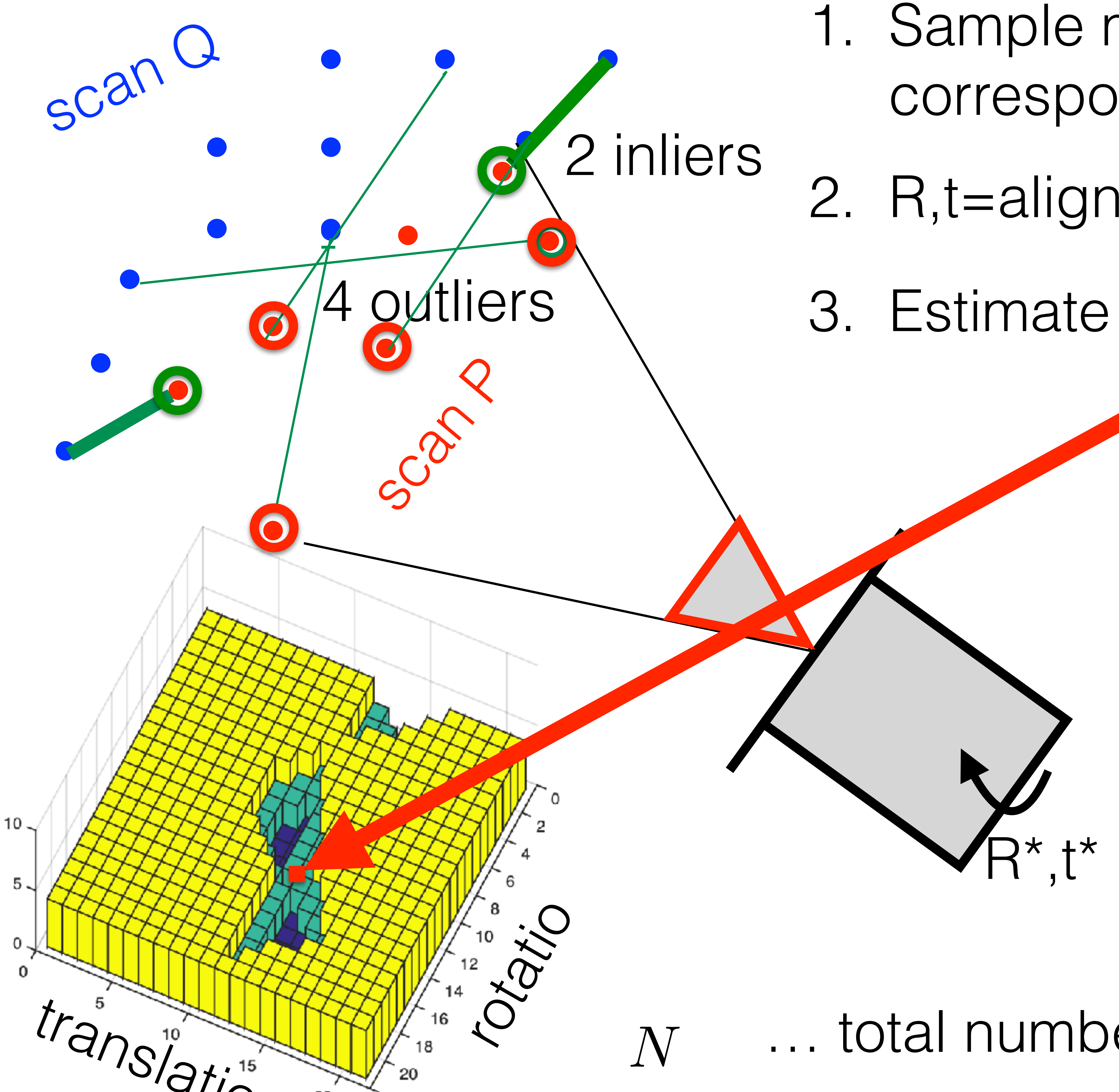


1. Sample minimal subset of correspondences (p, q).
2. R,t=align_L2(p, q)

s … size of |S| (s=2)

N … total number of correspondences (N=6)

RANSAC (RAndom SAmple Consensus)
Minimize number of outliers

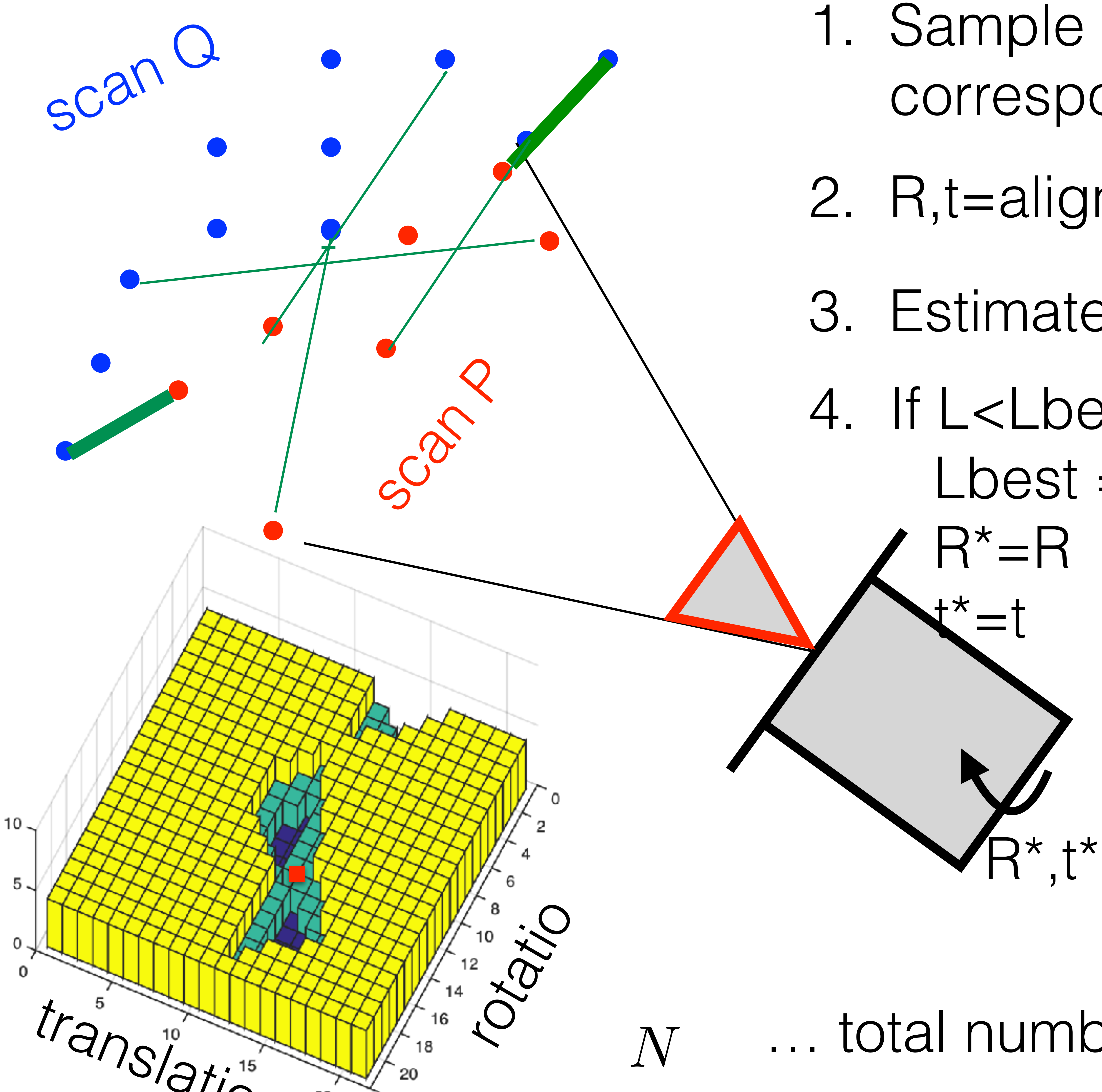


1. Sample minimal subset of correspondences (p, q).
2. R,t=align_L2(p, q)
3. Estimate loss **L=4**

s … size of |S| (s=2)

N … total number of correspondences (N=6)

RANSAC (RAndom SAmple Consensus)

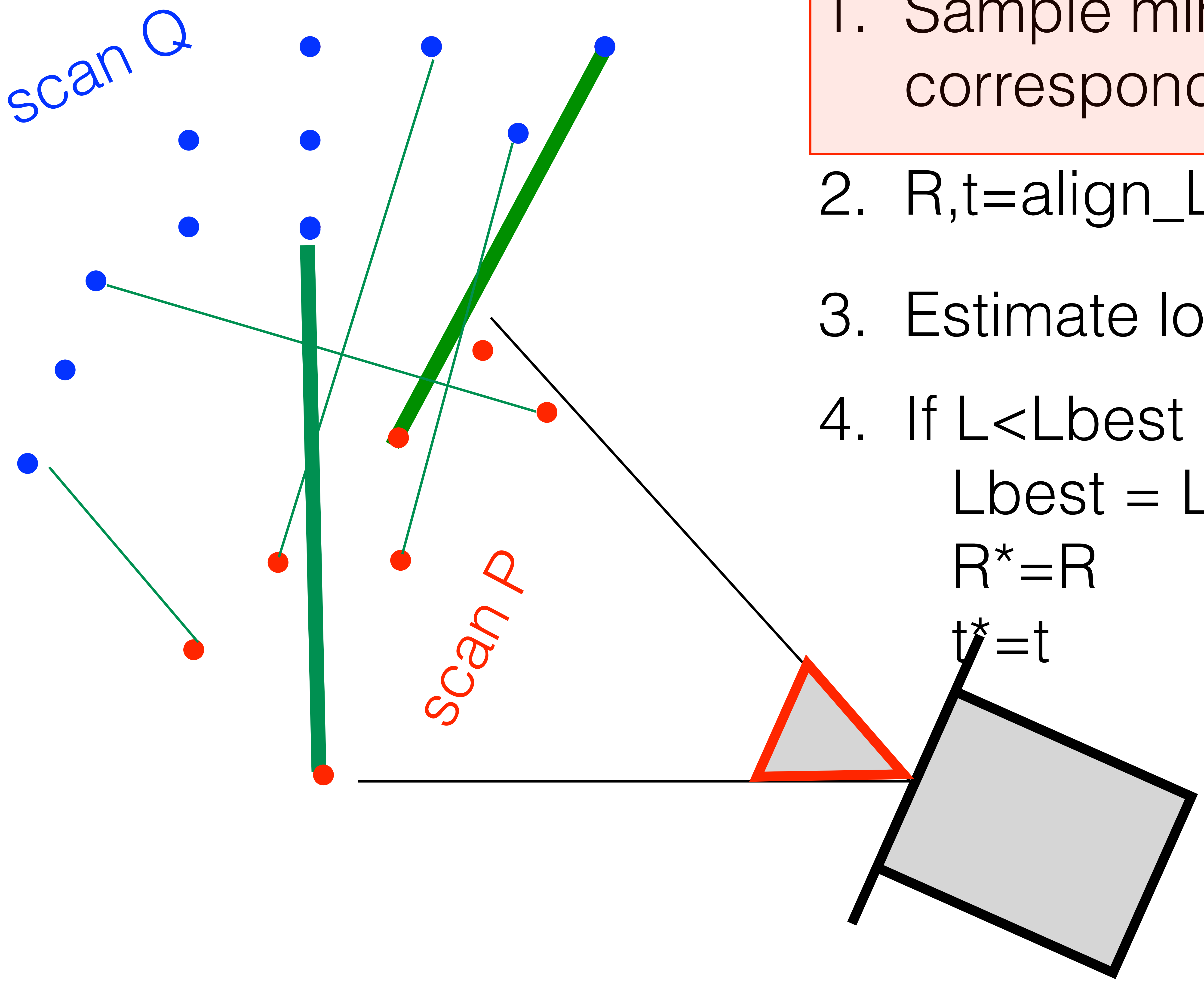


1. Sample minimal subset of correspondences (p, q).
2. R,t=align_L2(p, q)
3. Estimate loss
4. If L<Lbest
   Lbest = L
   R*=R
   t*=t

s … size of |S| (s=2)

N … total number of correspondences (N=6)

RANSAC

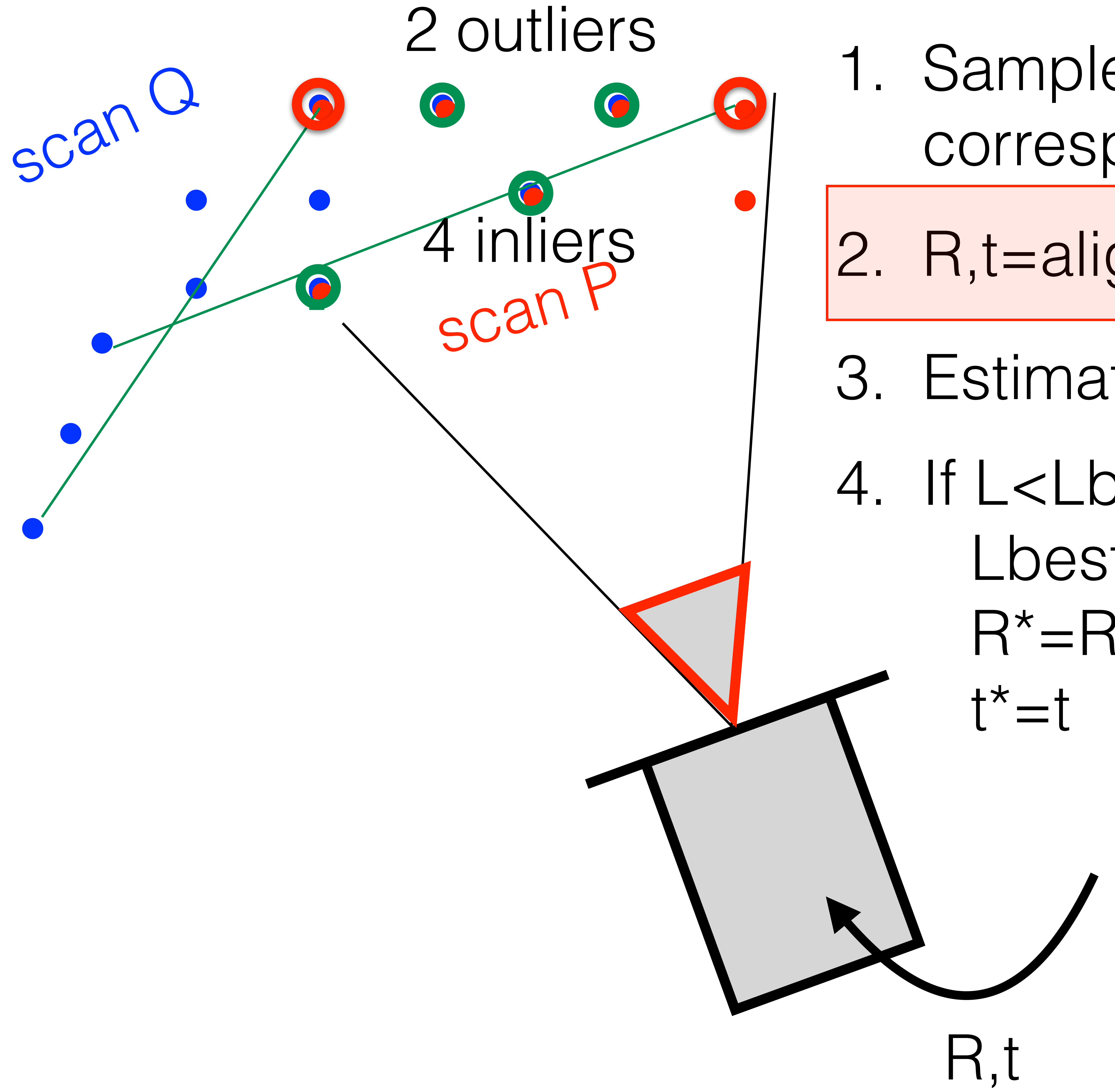


1. Sample minimal subset of correspondences (p, q).
2. R,t=align_L2(p, q)
3. Estimate loss
4. If L<Lbest
   Lbest = L
   R*=R
   t*=t

s … size of |S| (s=2)

N … total number of correspondences (N=6)

RANSAC (RAndom SAmple Consensus)

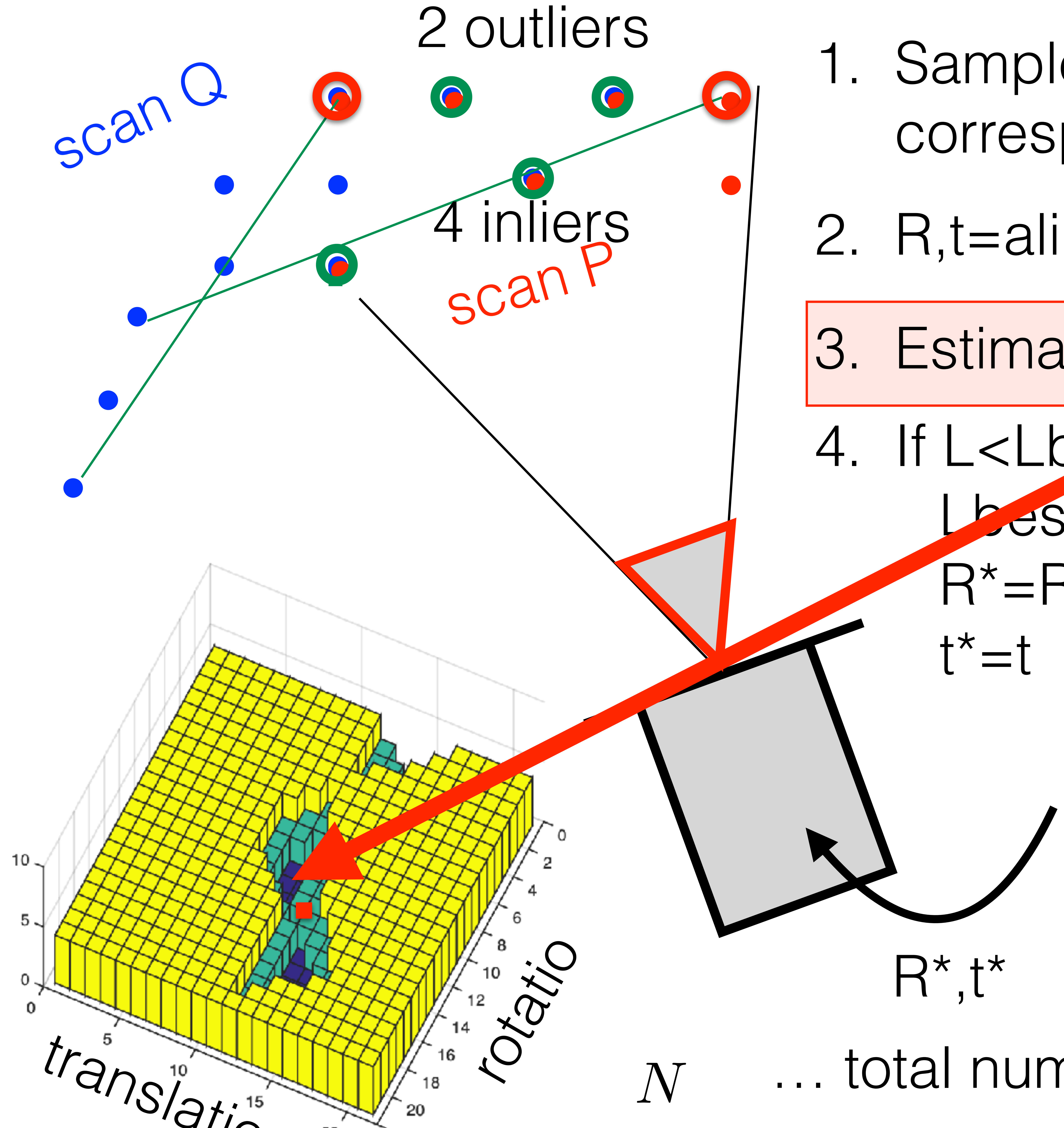


1. Sample minimal subset of correspondences (p, q).
2. R,t=align_L2(p, q)
3. Estimate loss
4. If L<Lbest
   Lbest = L
   R*=R
   t*=t

s … size of |S| (s=2)

N … total number of correspondences (N=6)

RANSAC (RAndom SAmple Consensus)



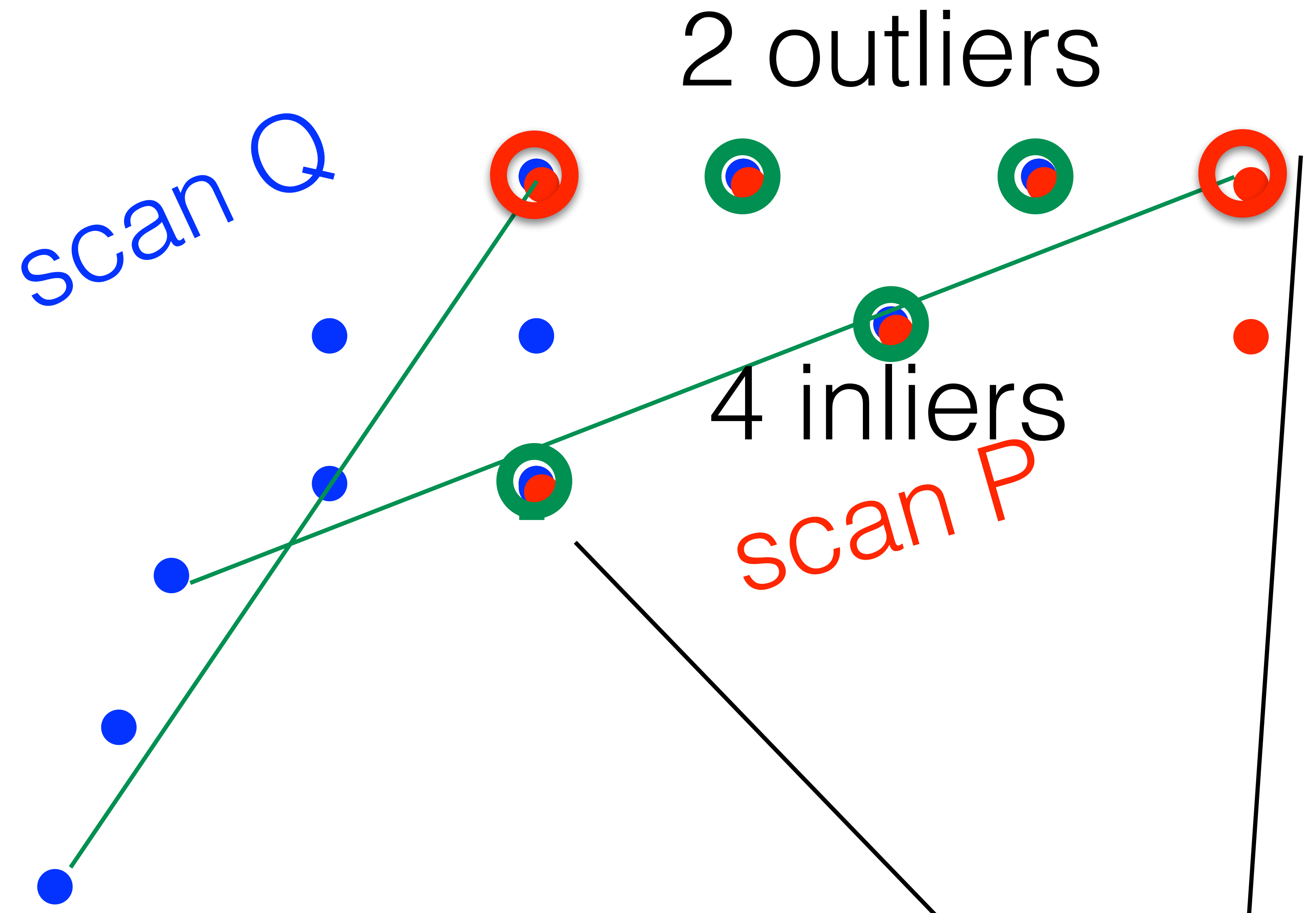
1. Sample minimal subset of correspondences (p, q).
2. R,t=align_L2(p, q)
3. Estimate loss **L=2**
4. If L<Lbest
   Lbest = L
   R*=R
   t*=t

R*,t*

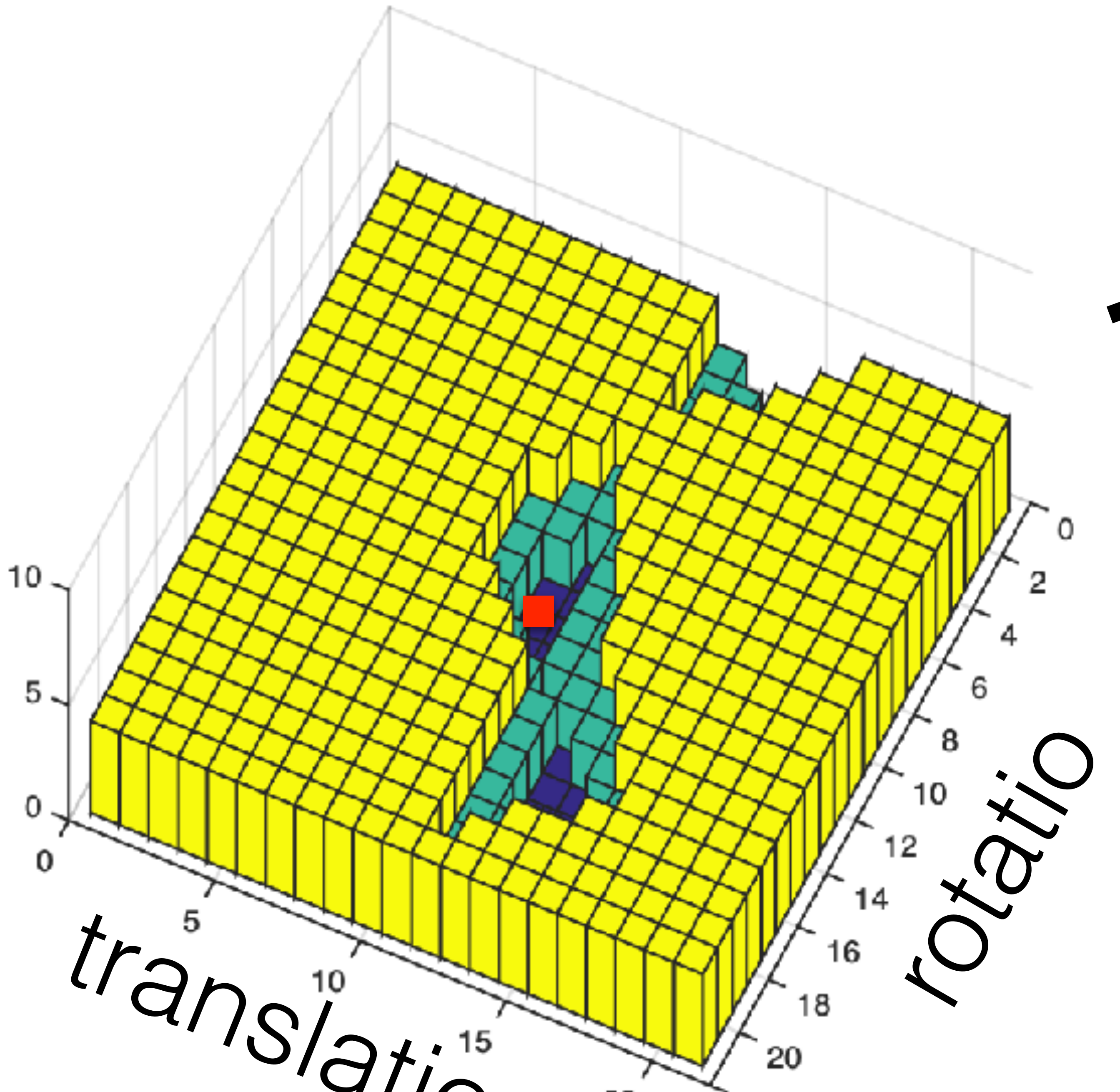
s … size of |S| (s=2)

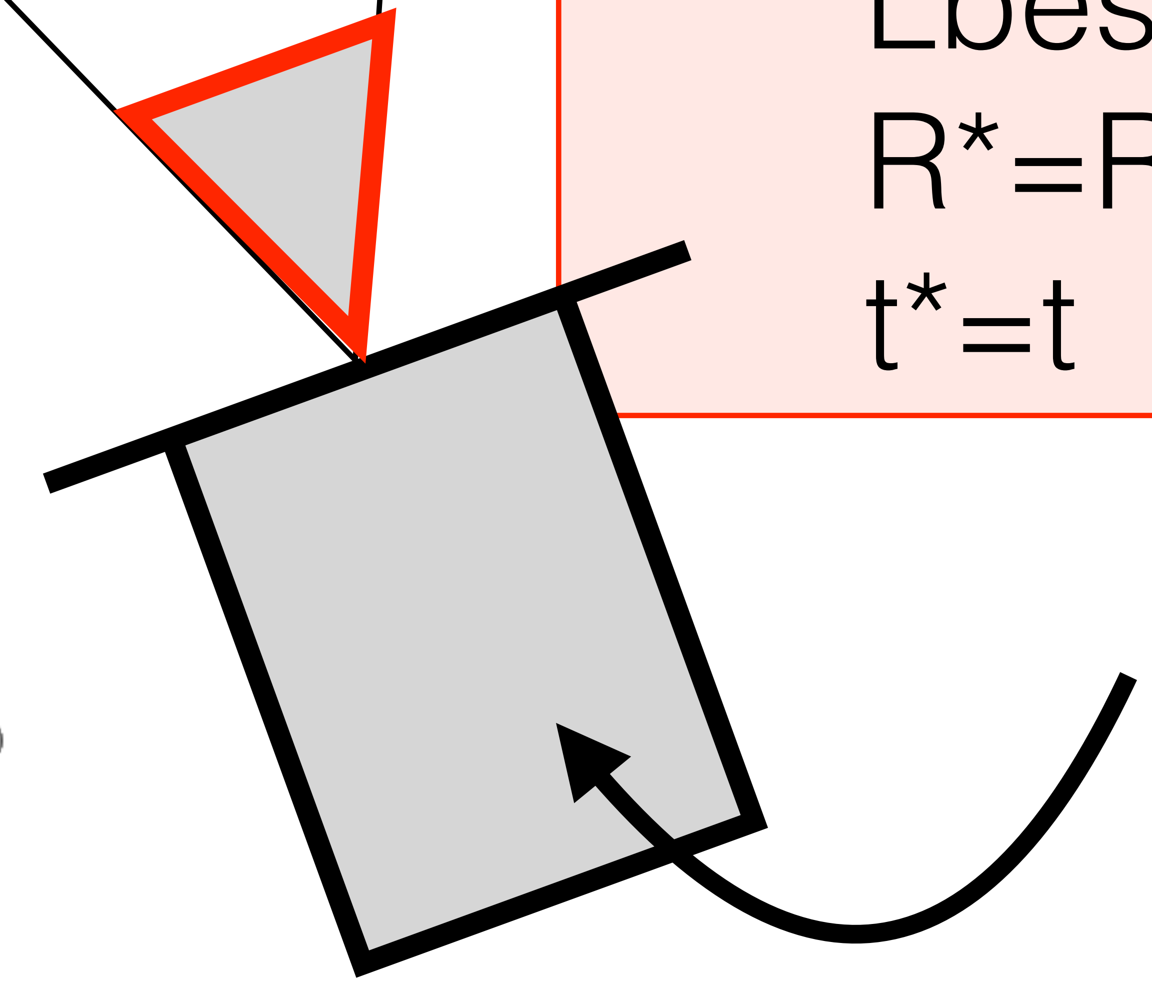
N … total number of correspondences (N=6)

RANSAC (RAndom SAmple Consensus)



1. Sample minimal subset of correspondences (p, q).
2. R,t=align_L2(p, q)
3. Estimate loss **L=2**
4. If L<Lbest
   Lbest = L
   R*=R
   t*=t



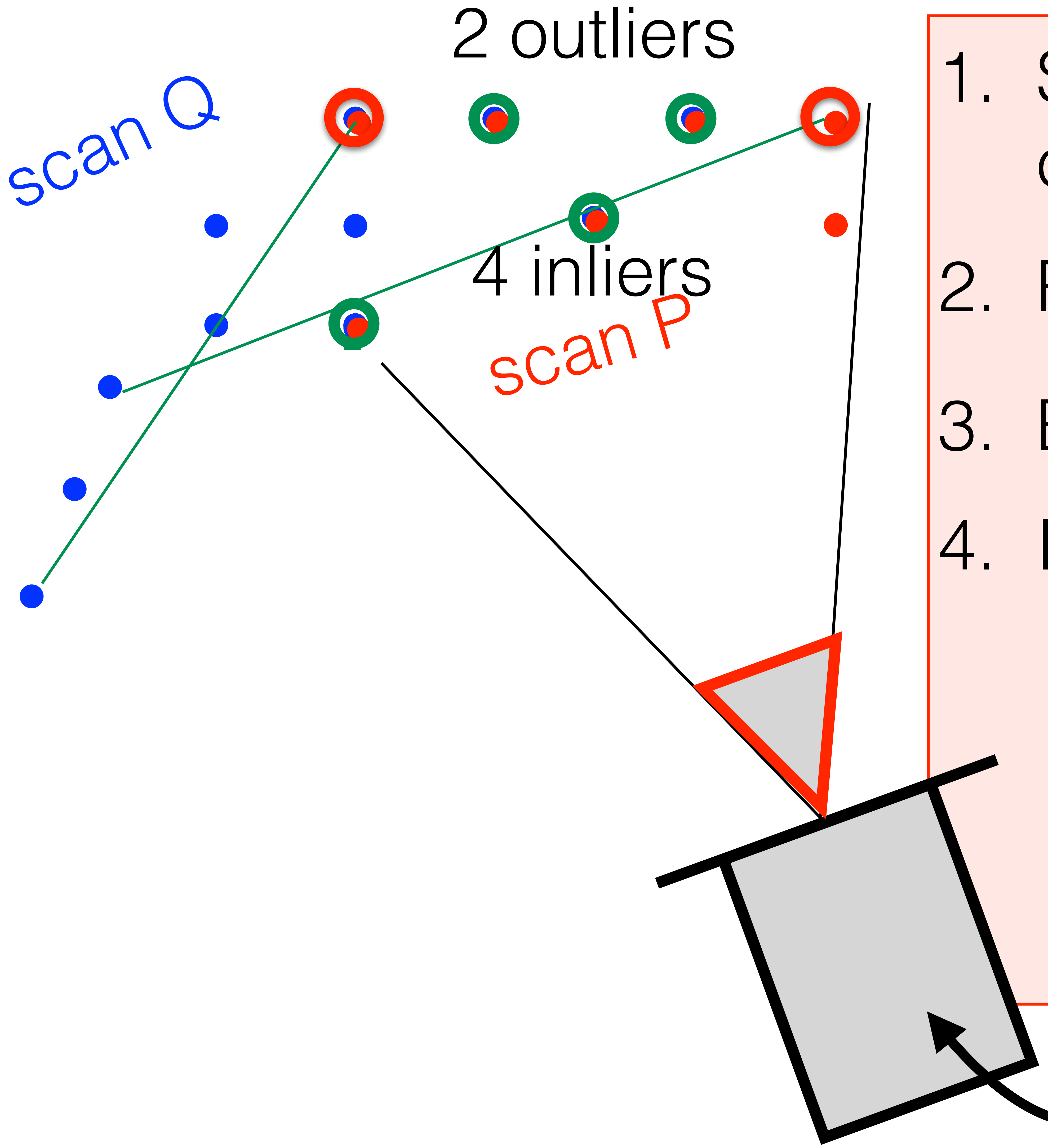


R*,t*

s … size of |S| (s=2)

N … total number of correspondences (N=6)

RANSAC (RAndom SAmple Consensus)



1. Sample minimal subset of correspondences (p, q).
2. R,t=align_L2(p, q)
3. Estimate loss **L=2**
4. If L<Lbest
   Lbest = L
   R*=R
   t*=t

repeat K-times

R*,t*

w … fraction of inliers (w =2/6 = 0.33)

s … size of |S| (s=2)

N … total number of correspondences (N=6)

RANSAC (RAndom SAmple Consensus)

N … total number of correspondences (N=6)

w … fraction of inliers (w = 2/6 = 0.33)

s … size of |S| (s=2)

p … probability, that we have selected a clean sample at least once out of K trials (prob. of success). (p=0.99)

K … number of trials/iterations

$$K = \frac{\log(1-p)}{\log(1-w^s)} = \frac{\log(1-0.99)}{\log(1-0.33^2)} \approx 39.94$$

Summary

- Minimizing **L2-loss** on unclean correspondences (with **outliers**) yields **biased pose** estimate (and pointcloud alignment).
- Minimizing **robust norms** (Welsch) yields **complicated optimization** due to large plateaus with almost zero gradients.
- When **motion** between successive frames is sufficiently **small** (self-driving cars), odom-initialized **gradient minimization** of a robust loss is quite **OK**.
- When **motion is large** and **correspondences unclean** inlier detection method **RANSAC**, which randomly sample reasonable hypothesis (R,t).
- RANSAC is often used for 2D-2D correspondences and large motions (e.g. reconstruction of 3D world from collection of unordered RGB images).
- **Takehome message:** When designing the loss function always think about:
  A. Underlying probability distribution
  B. Optimization of the resulting landscape

Useful references

- SLAM implementations:
  - Nvidia Issac SLAM: https://github.com/NVIDIA-ISAAC-ROS/isaac_ros_visual_slam
  - ORB SLAM (RGBD SLAM): https://github.com/alsora/ros2-ORB_SLAM2
  - GTSAM (modular factorgraph SLAM implementation in C++) https://gtsam.org/
  - PyPose (modular factorgraph SLAM implementation in Python/Pytorch) https://pypose.org/
- Datasets, benchmarks and challenges:
  - Waymo https://waymo.com/intl/en_us/dataset-download-terms/
  - Kitti http://www.cvlibs.net/datasets/kitti/