Super 4PCS

Fast Global Pointcloud Registration via Smart Indexing

Nicolas Mellado[1], Dror Aiger[2], Niloy J. Mitra[1]

[1] University College London [2] Google Inc.

Problem Statement

- Estimate rigid transformation **tr**
  - Large search space (6DoF)

P

+

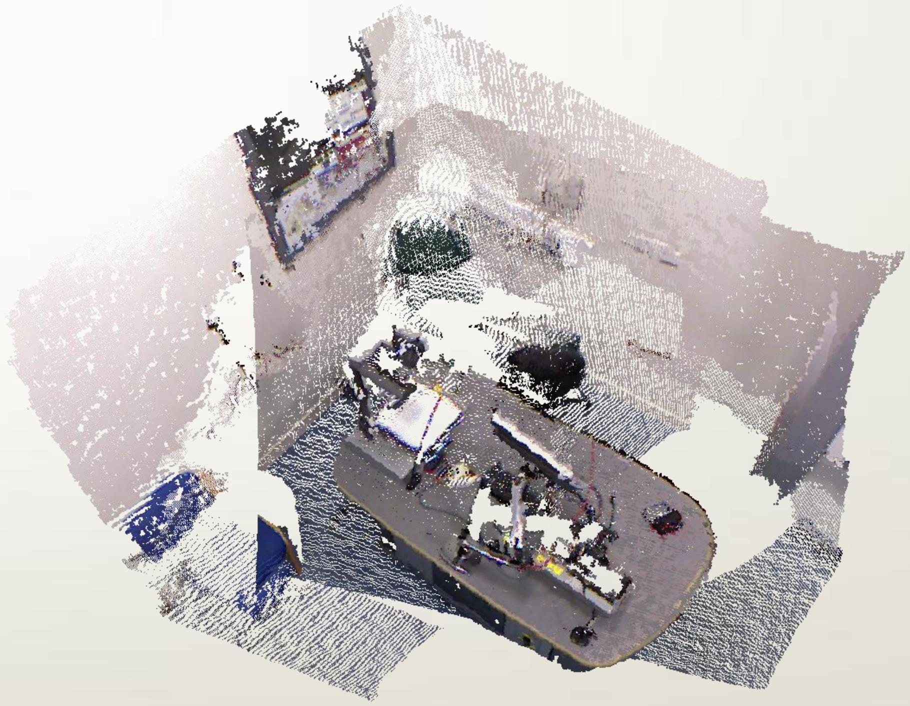
Q

=

P + tr(Q)

Problem Statement

- Estimate rigid transformation **tr**
  - **Local** registration: from an input pose
    - **ICP** [BM92], [CM92], [RL01], [MGPG04]
    - Sparse ICP [BTP13]
    - Kinect Fusion [IKH*11]



P + tr(Q)

Problem Statement

- Estimate rigid transformation **tr**
  - **Local** registration: from an input pose
    - **ICP** [BM92], [CM92], [RL01], [MGPG04]
    - Sparse ICP [BTP13]
    - Kinect Fusion [IKH*11]
  - **Global** registration: arbitrary input pose
    - RANSAC [FB81], [IR96], [CH99]
    - and variants [GMGP05], [PB09], [PB11], [ART10], [RABT13]
    - 4 Point Congruent Set [AMCO08]

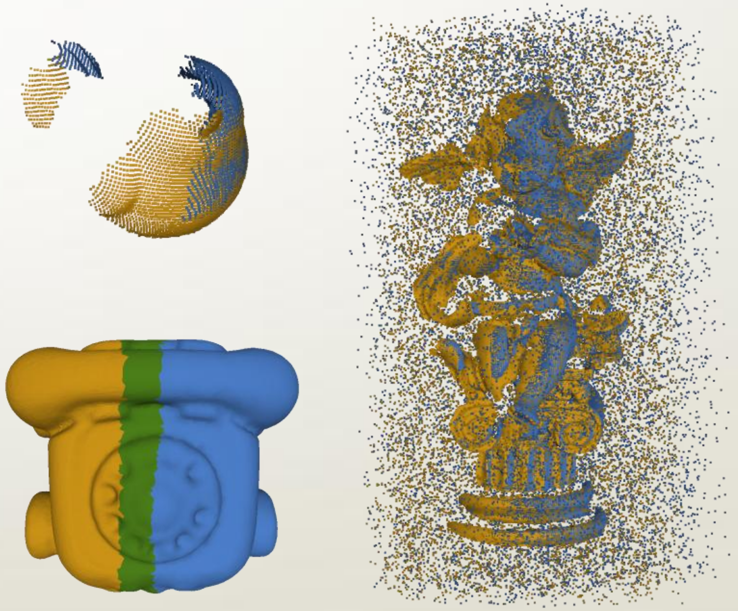
3 pairs of corresponding points are sufficient to define a rigid transformation

Problem Statement

RANSAC: $O(n^3)$

4 Point Congruent Set (4PCS) : $O(n^2)$

Difficult cases



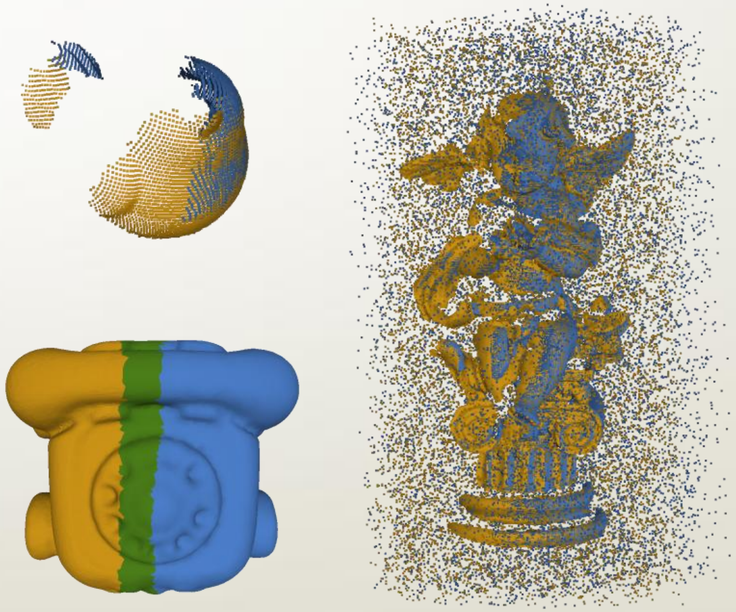
Problem Statement

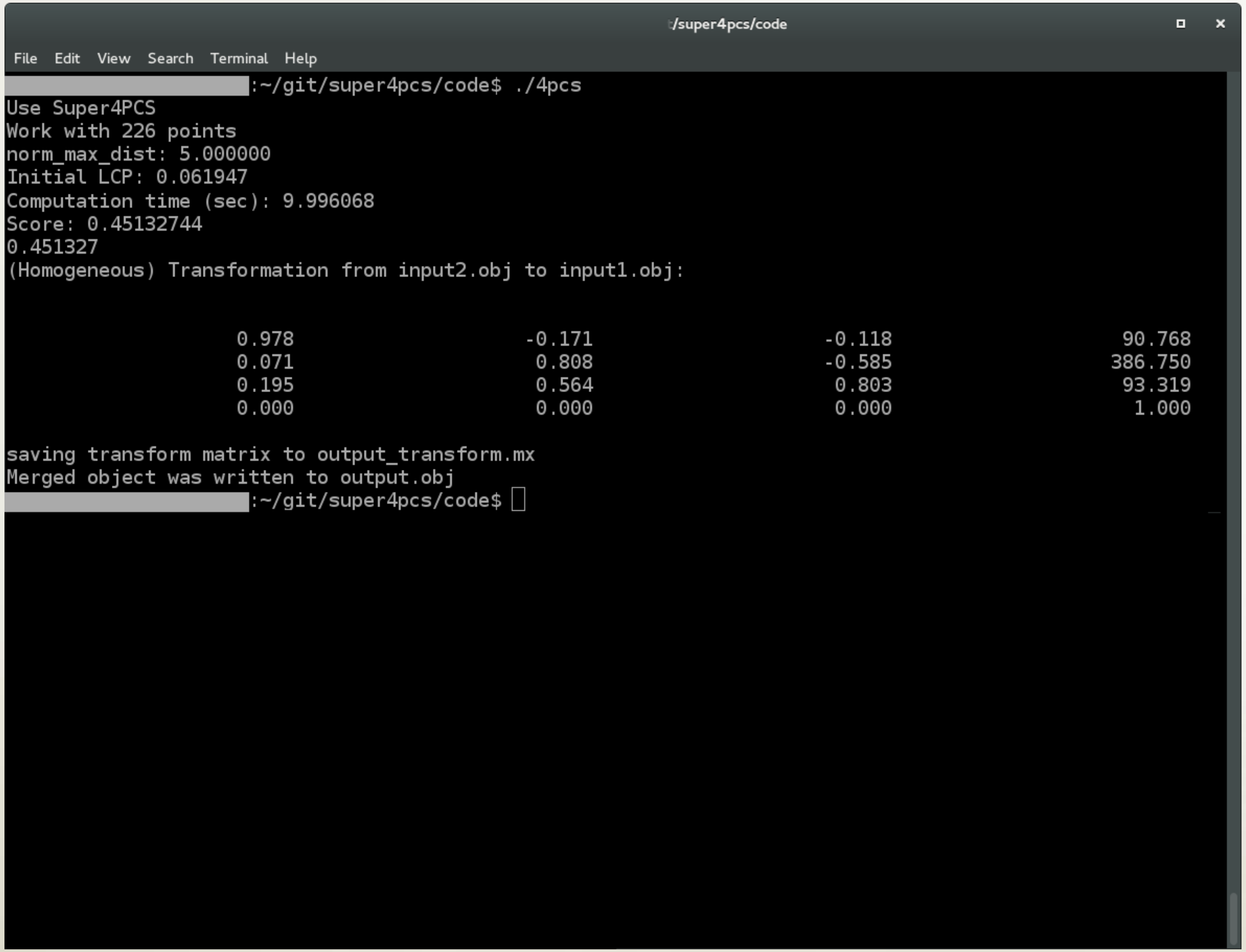
RANSAC: $O(n^3)$

4 Point Congruent Set (4PCS) : $O(n^2)$

Our approach (Super 4PCS): $O(n)$

Difficult cases

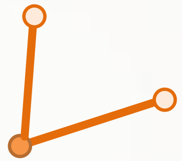



~/super4pcs/code
File Edit View Search Terminal Help
[redacted]:~/git/super4pcs/code$ ./4pcs
Use Super4PCS
Work with 226 points
norm_max_dist: 5.000000
Initial LCP: 0.061947
Computation time (sec): 9.996068
Score: 0.45132744
0.451327
(Homogeneous) Transformation from input2.obj to input1.obj:
0.978 -0.171 -0.118 90.768
0.071 0.808 -0.585 386.750
0.195 0.564 0.803 93.319
0.000 0.000 0.000 1.000
saving transform matrix to output_transform.mx
Merged object was written to output.obj
[redacted]:~/git/super4pcs/code$

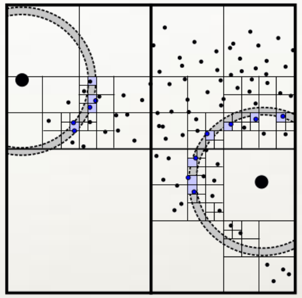
Demo

Overview

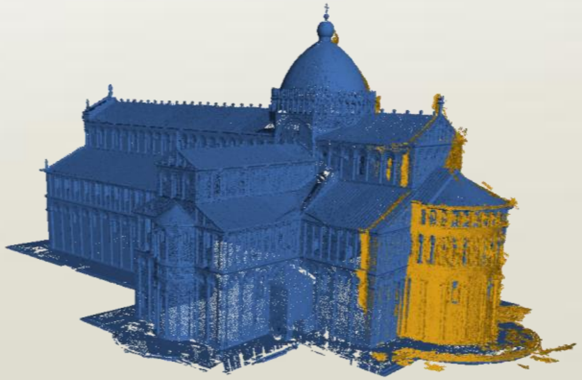
- 4PCS



- Super4PCS

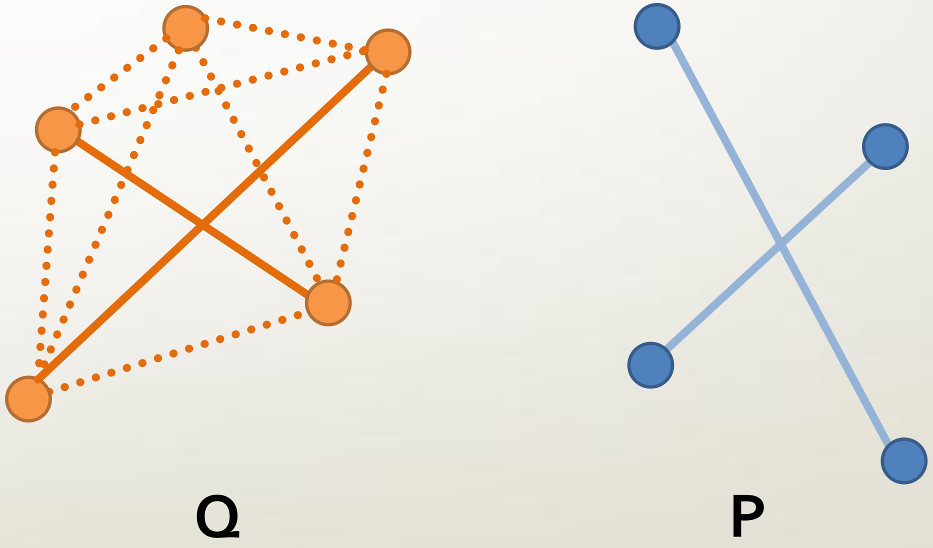


- Results



4 Point Congruent Set

- Use planar 4-points basis in **P**
- Find congruent 4-points in **Q**

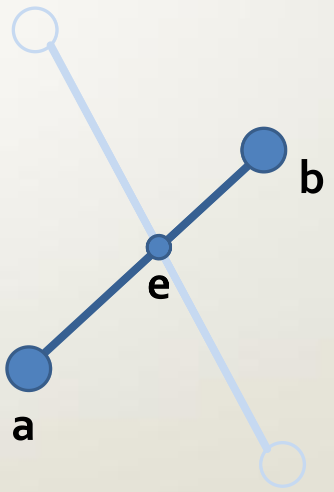


4-points Congruent Sets for Robust Surface Registration
Dror Aiger, Niloy J. Mitra, Daniel Cohen-Or
SIGGRAPH 2008

4 Point Congruent Set

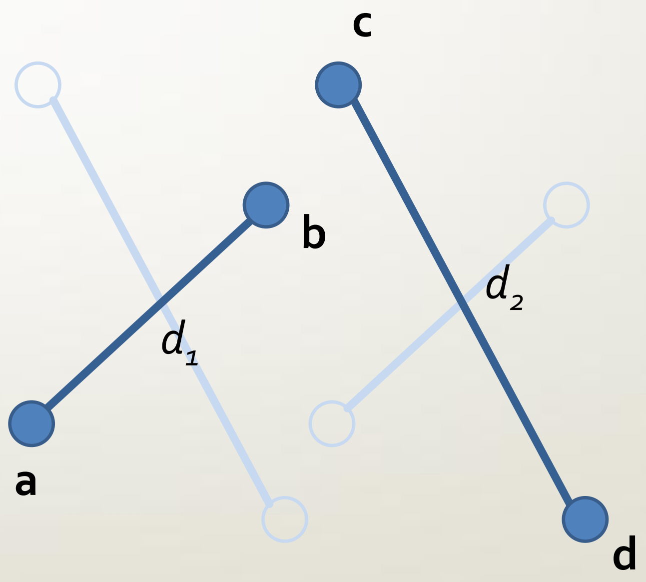
- What does congruent mean ?
- Similar under a given transformation class
  - Ratios r_1 and r_2

r_1 = || **a** - **e** || / || **a** - **b** ||



4 Point Congruent Set

- What does congruent mean ?
- Similar under a given transformation class
  - Ratios r_1 and r_2
  - Distances d_1 , d_2

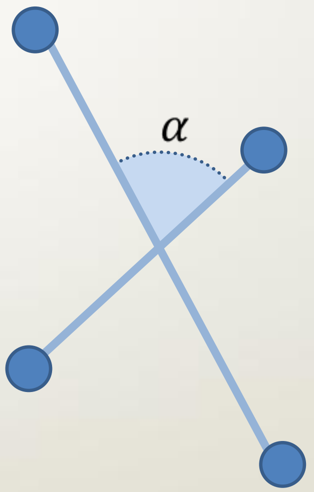


4 Point Congruent Set

- What does congruent mean ?

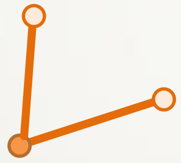
- Similar under a given transformation class
  - Ratios r_1 and r_2
  - Distances d_1 , d_2
  - Angle α

Rigid transformation

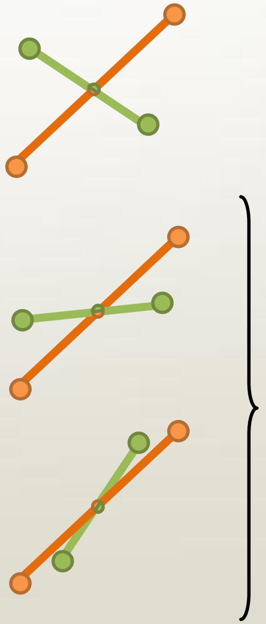


4 Point Congruent Set

Select Base → Extract pairs → Extract congruent superset → Filter congruent set → Verify and update transformation

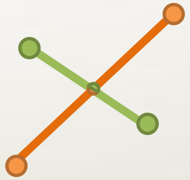


Ratios
Distances



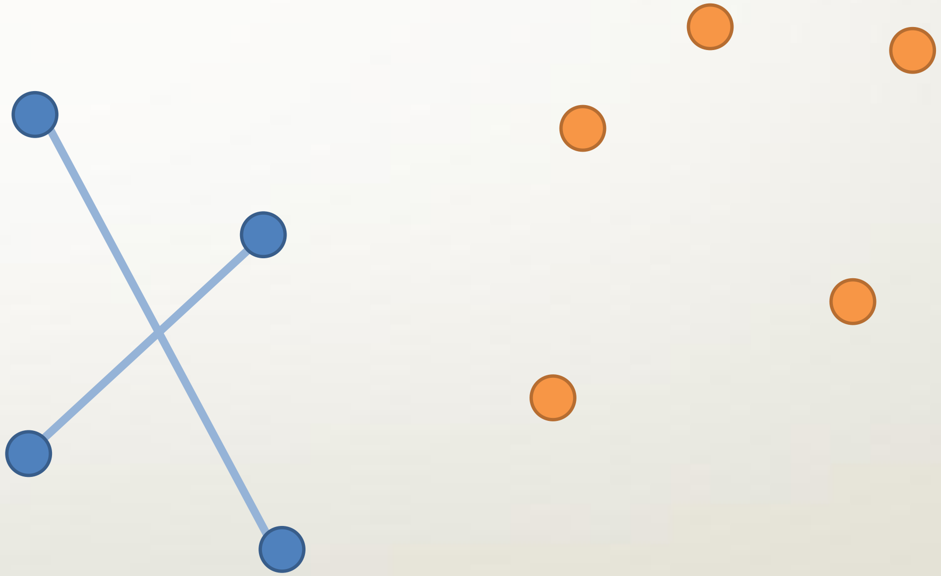
False positives (non congruents)

Ratios
Distances
Angles



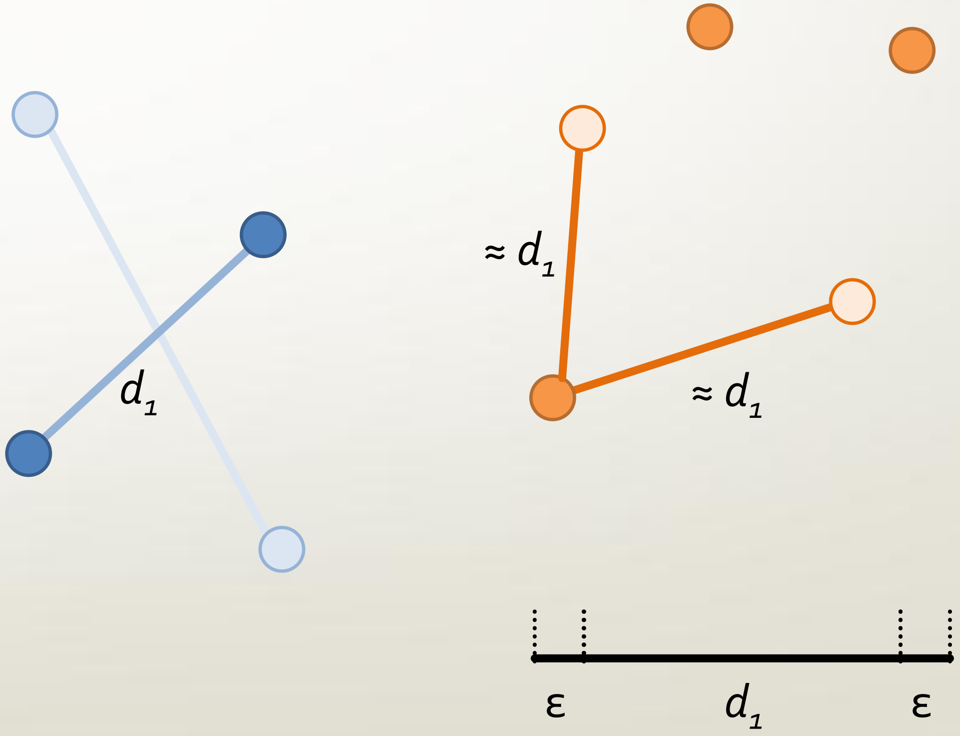
4 Point Congruent Set

- Extract pairs



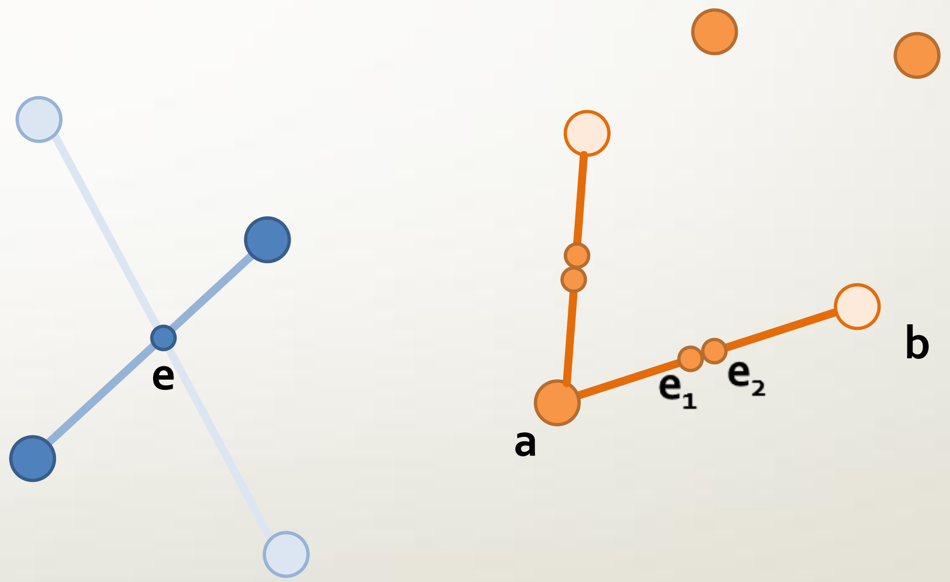
4 Point Congruent Set

- Extract pairs



4 Point Congruent Set

- Extract pairs

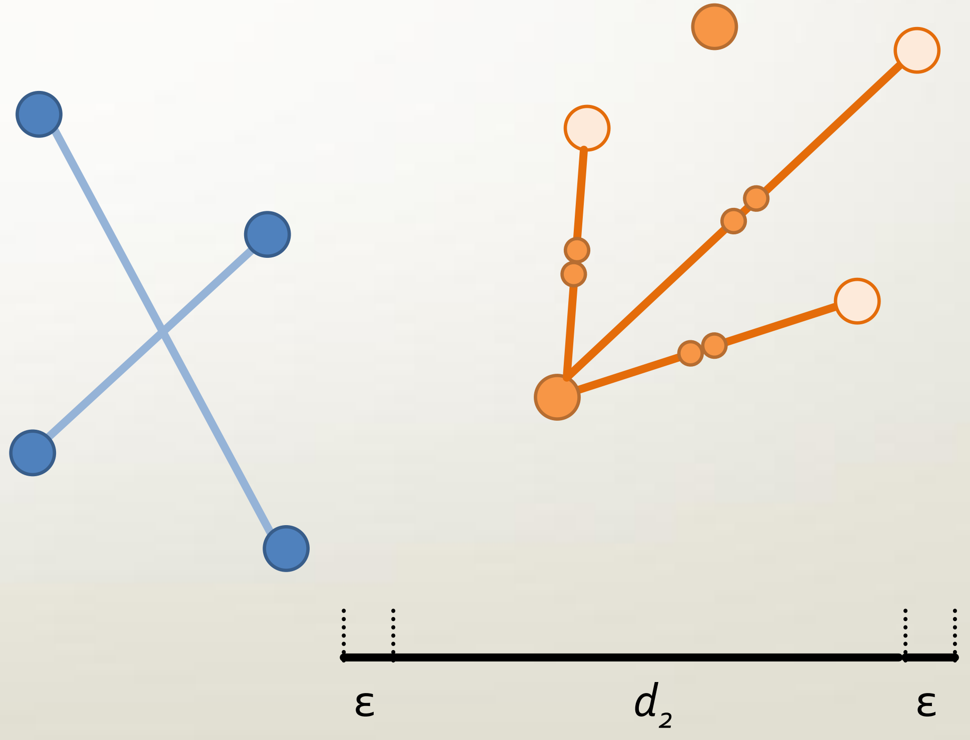


$$\mathbf{e}_1 = \mathbf{a} + r_1(\mathbf{b} - \mathbf{a})$$

$$\mathbf{e}_2 = \mathbf{a} + r_2(\mathbf{b} - \mathbf{a})$$

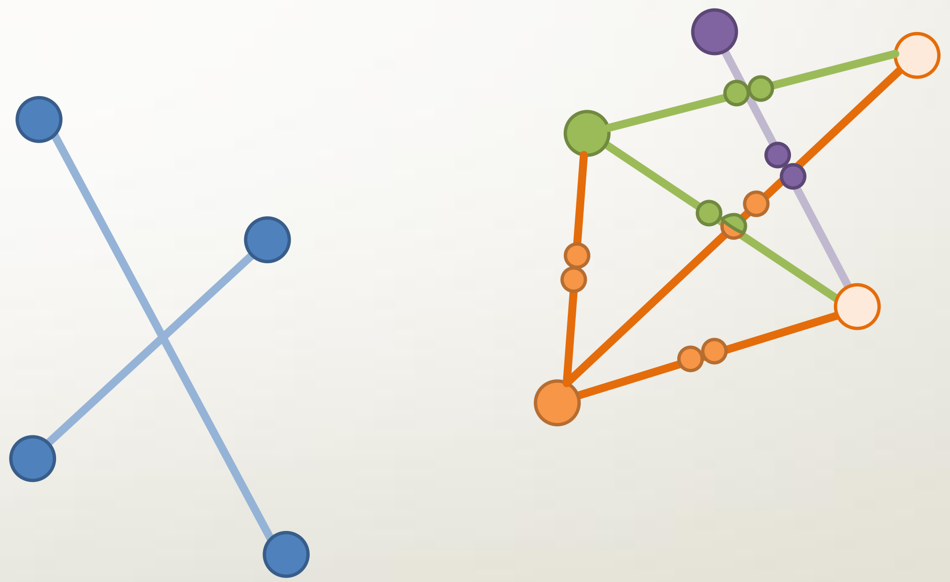
4 Point Congruent Set

- Extract pairs



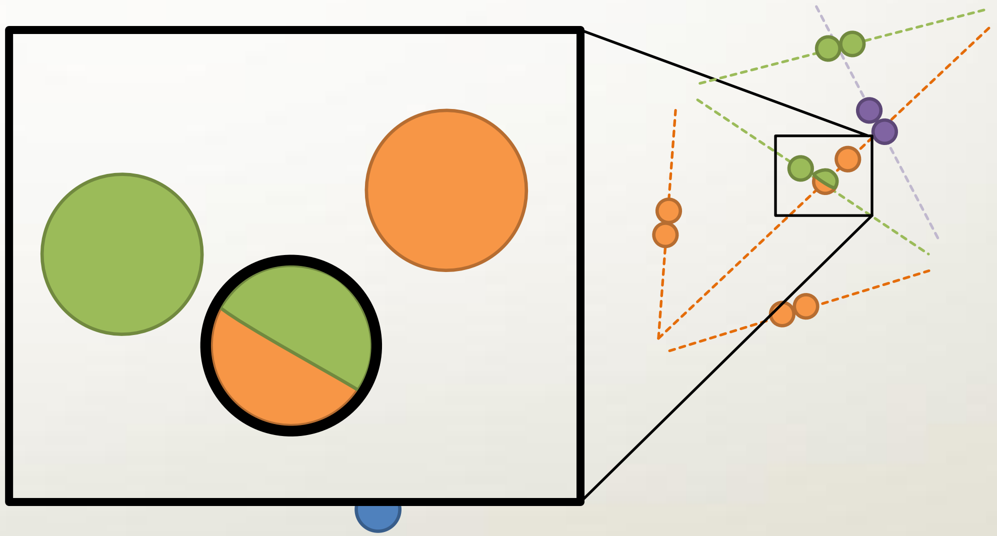
4 Point Congruent Set

- Extract pairs



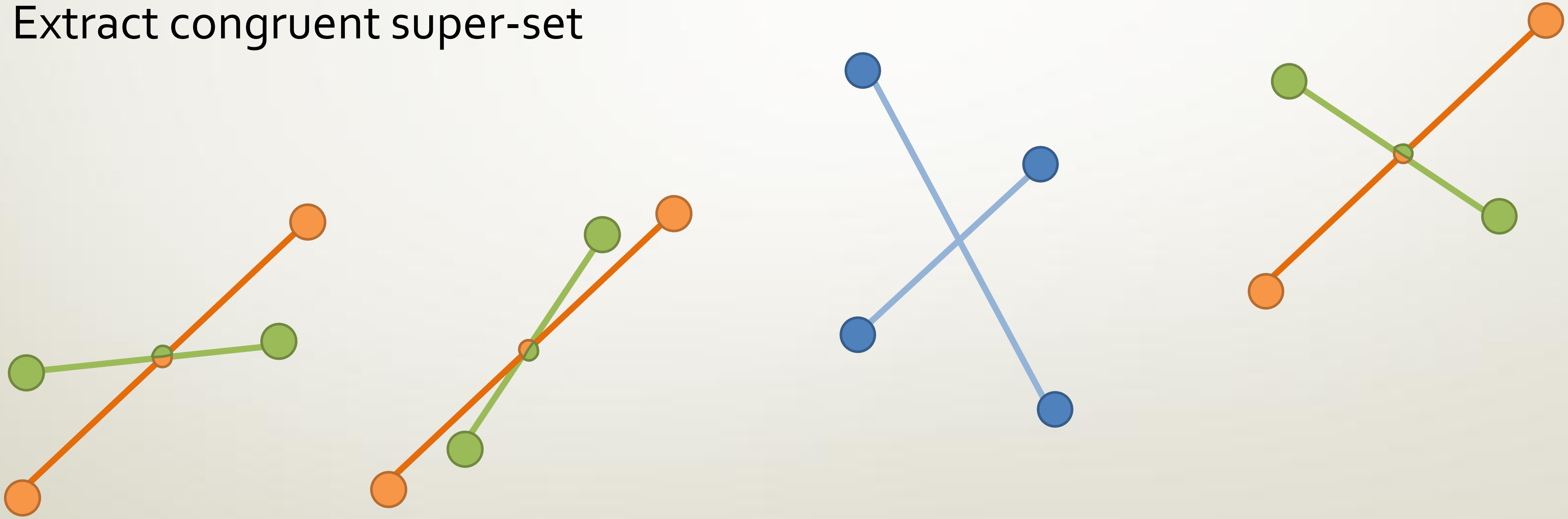
4 Point Congruent Set

- Extract pairs
- Extract congruent super-set



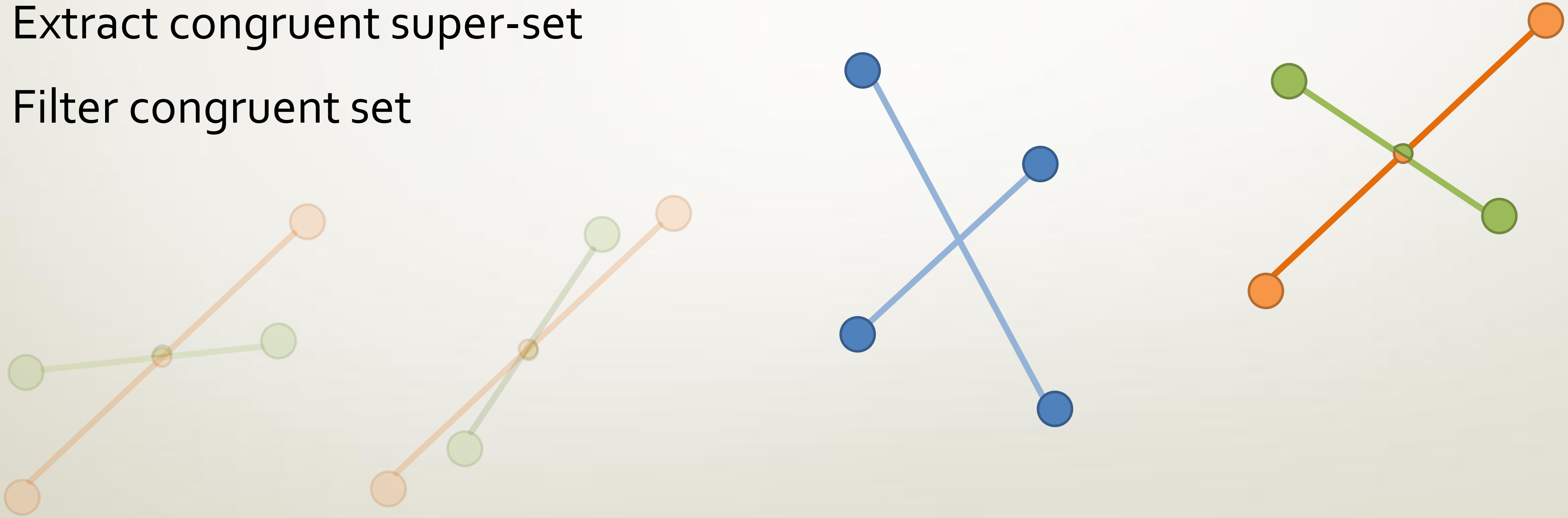
4 Point Congruent Set

- Extract pairs
- Extract congruent super-set



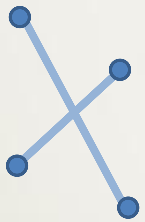
4 Point Congruent Set

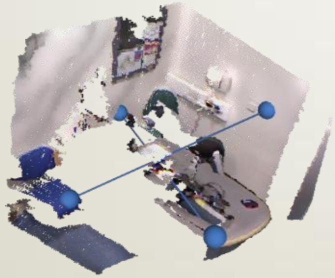
- Extract pairs
- Extract congruent super-set
- Filter congruent set

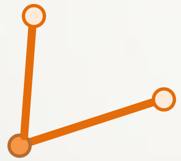


4 Point Congruent Set

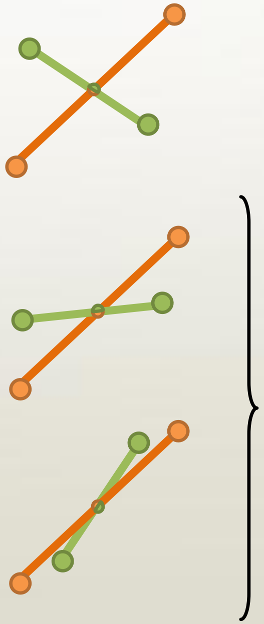
	$O(n^2)$	$O(m\log(m))$	$O(k)$	
Select Base	Extract pairs	Extract congruent superset	Filter congruent set	Verify and update transformation





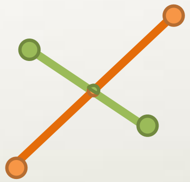


Ratios
Distances



False positives (non congruents)

Ratios
Distances
Angles



n points
m pairs
k congruent sets

Super 4PCS

~~$O(n^2)$~~

$O(n)$

~~$O(m\log(m) + k)$~~

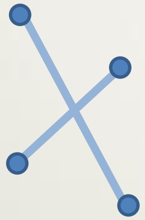
$O(m+k)$

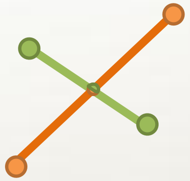
Select Base

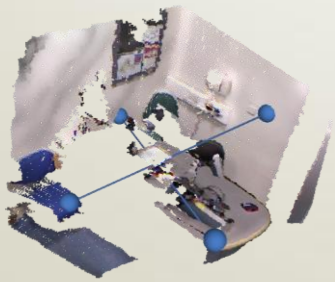
Extract pairs

Extract congruent set

Verify and update transformation







No more false positive

n points

m pairs

k congruent sets

Super 4PCS

$\cancel{O(n^2)}$ $\cancel{O(m\log(m) + k)}$

$O(n)$ $O(m+k)$

on

Our smart indexing techniques
produce the **<u>same</u>** congruent set as 4PCS
but in linear time

n points
m pairs
k congruent sets

Super 4PCS

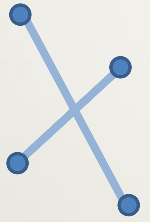
$\cancel{O(n^2)}$

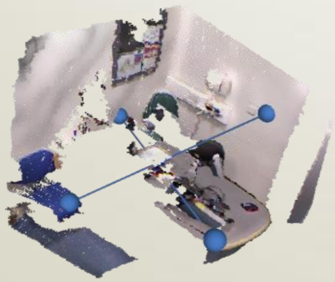
$O(n)$

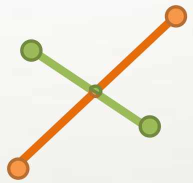
$\cancel{O(m\log(m) + k)}$

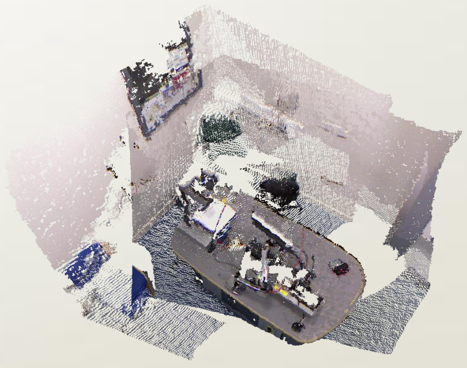
$O(m+k)$

Select Base → Extract pairs → Extract congruent set → Verify and update transformation





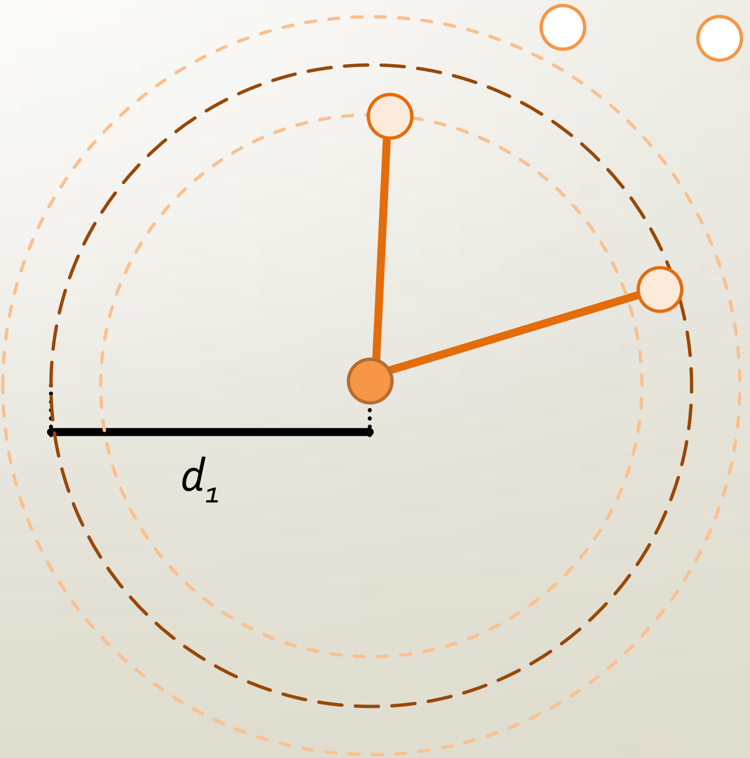




n points
m pairs
k congruent sets

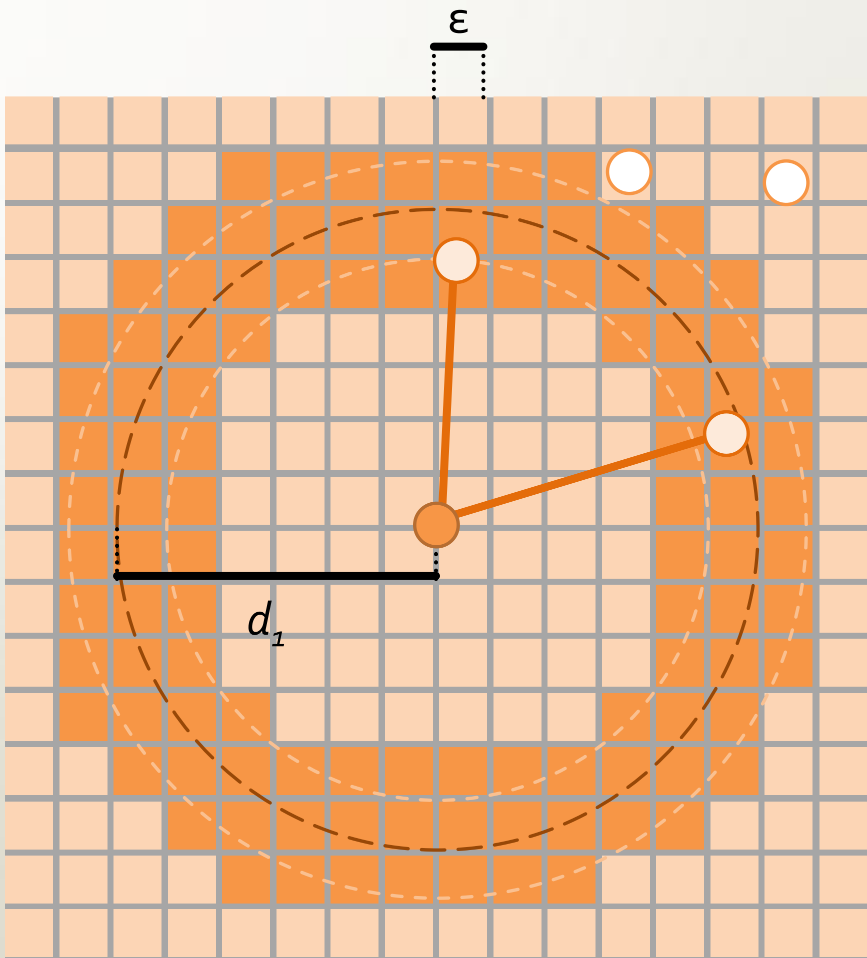
Pair extraction

- Reporting incidences: all valid pairs generated from a sphere



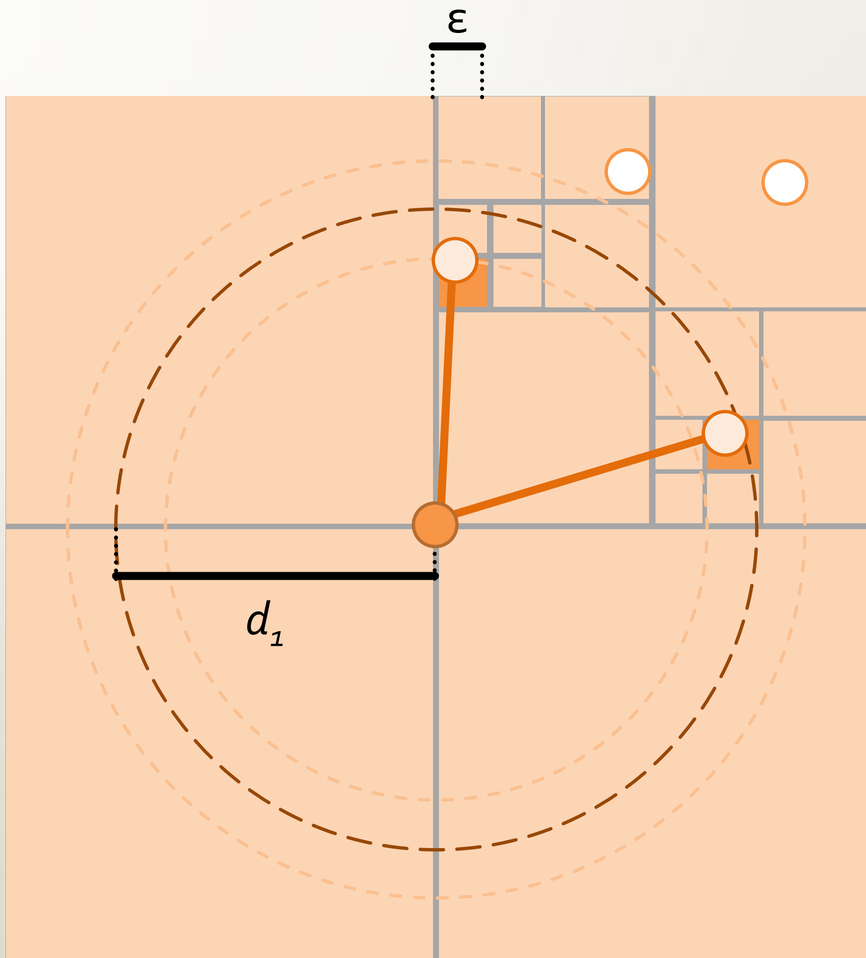
Pair extraction

- Reporting incidences using sphere rasterization
  - Complexity depends only on ε and d_1



Pair extraction

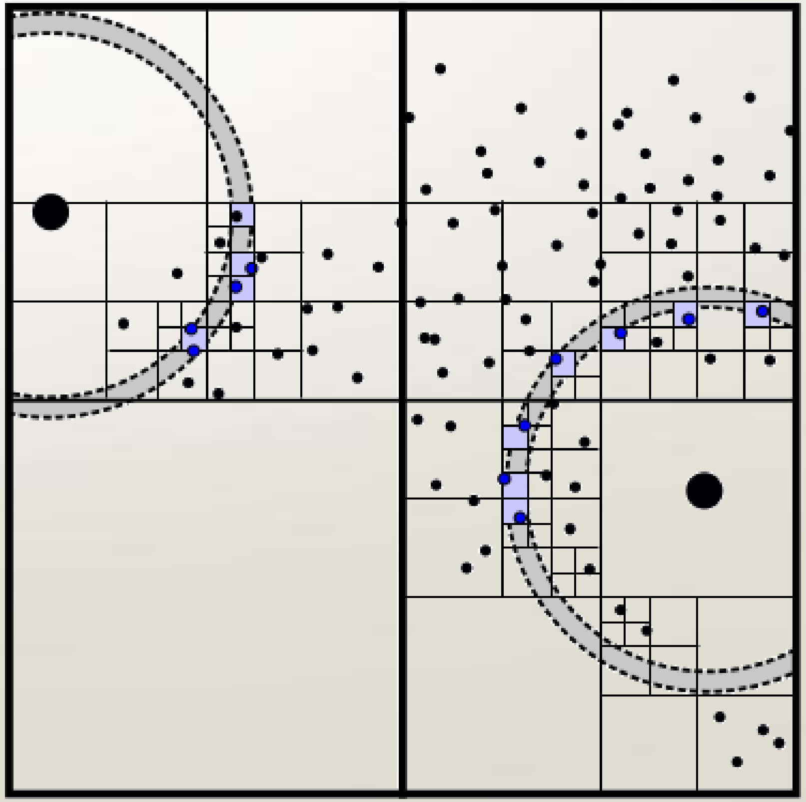
- Reporting incidences using sphere rasterization
- With an adaptative grid



Note: using a pre-computed tree is not optimal

Pair extraction

- Reporting incidences using sphere rasterization
- With an adaptative grid
- Simultaneously for all points

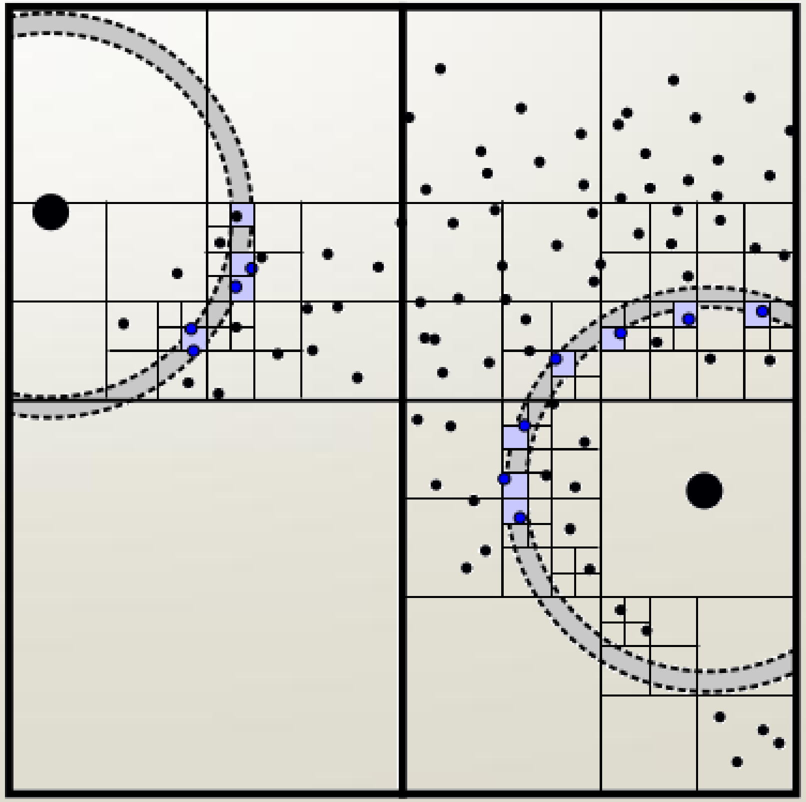


Pair extraction

- Reporting incidences using sphere rasterization
- With an adaptative grid
- Simultaneously for all points

- Theoretical complexity: *O(n)*

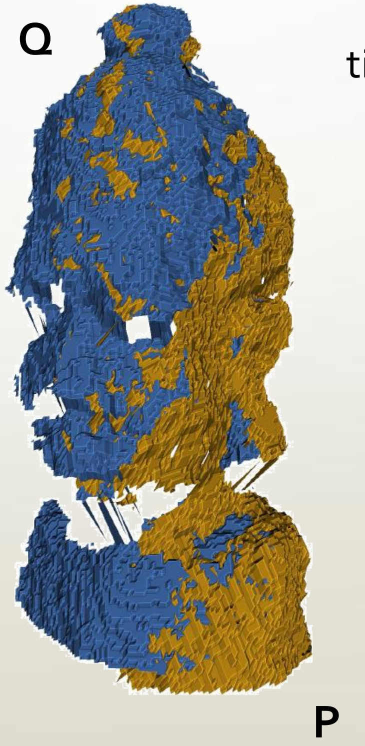
  (see details in the paper)

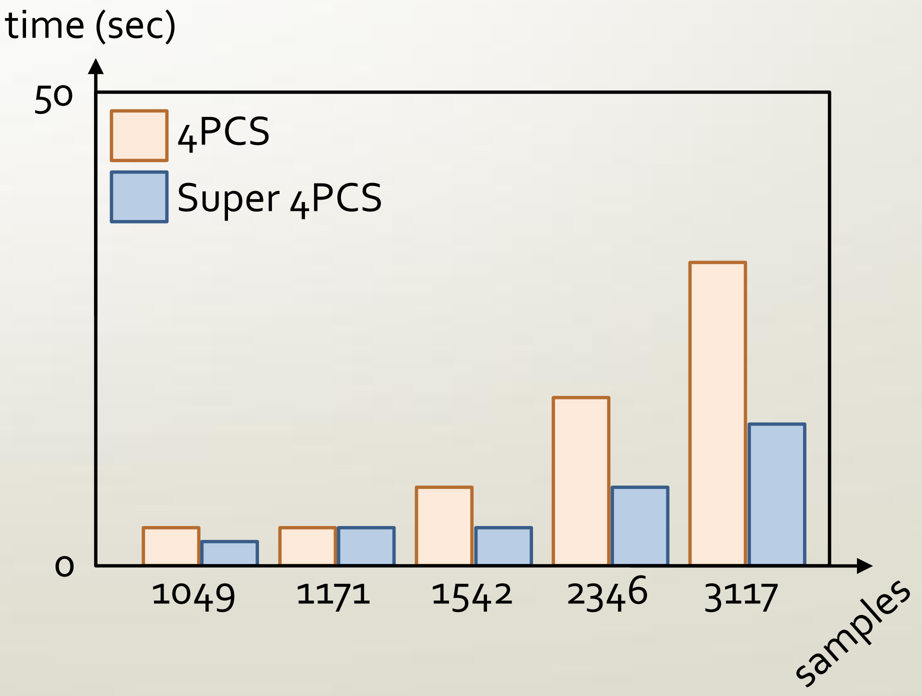


Pair extraction

- Reporting incidences using sphere rasterization
- With an adaptative grid
- Simultaneously for all points

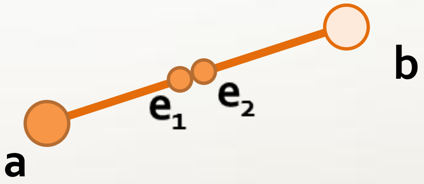
- Theoretical complexity: *O(n)*
- In practice
  - Runtime: linear
  - Minimal memory overhead

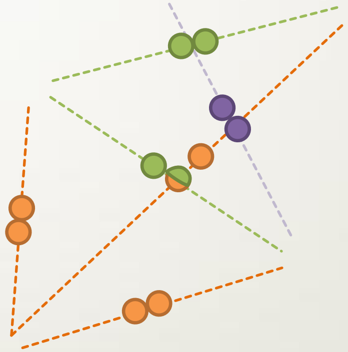




4PCS
Congruent set extraction

- Original approach
  - Represent a pair by 2 invariants

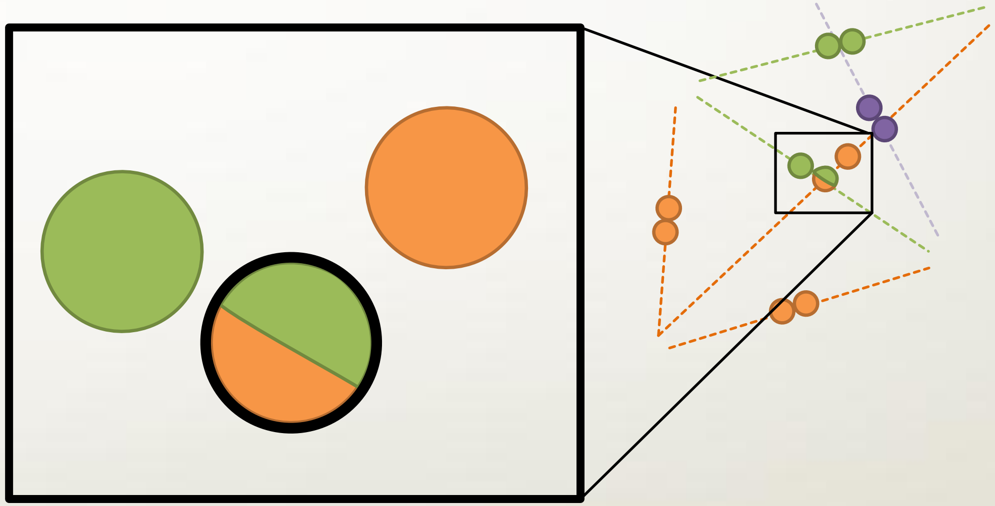




m pairs
k congruent sets

4PCS
Congruent set extraction

- Original approach
  - Represent a pair by 2 invariants
  - Use **kd-tree** to find closest invariants

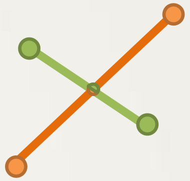


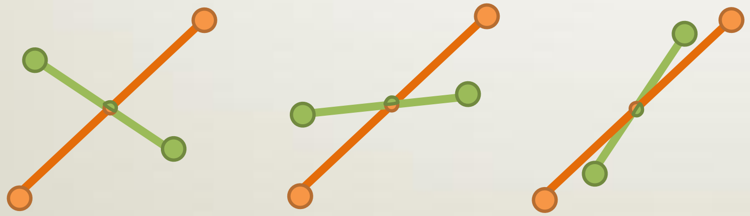
O(mlog(m)   )

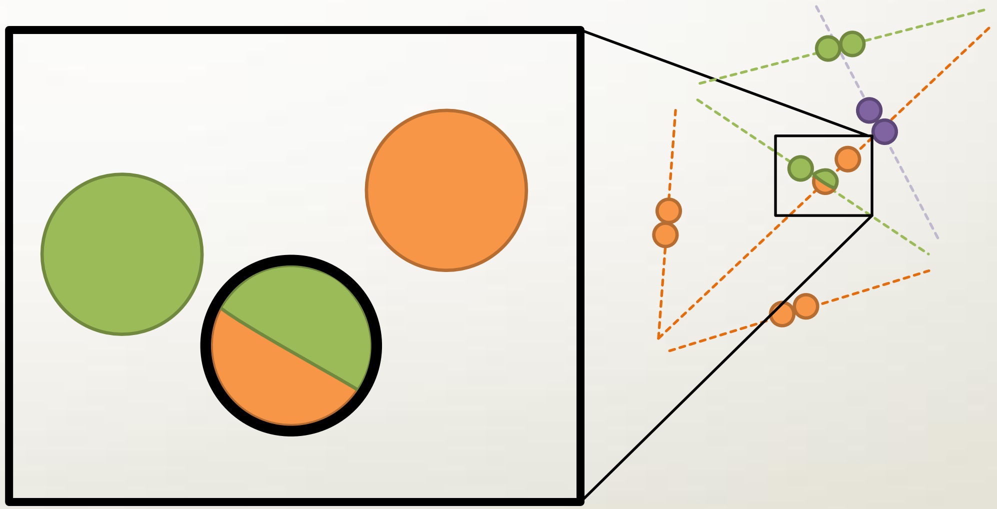
m pairs
k congruent sets

4PCS
Congruent set extraction

- Original approach
  - Represent a pair by 2 invariants
  - Use **kd-tree** to find closest invariants
  - Filter non congruent quadriplets





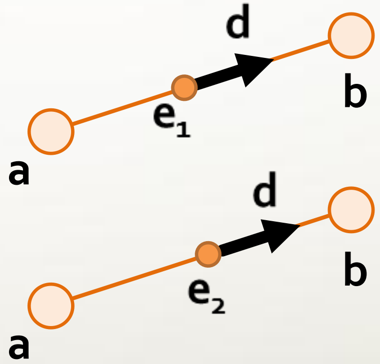


$O(m\log(m) + k)$

m pairs
k congruent sets

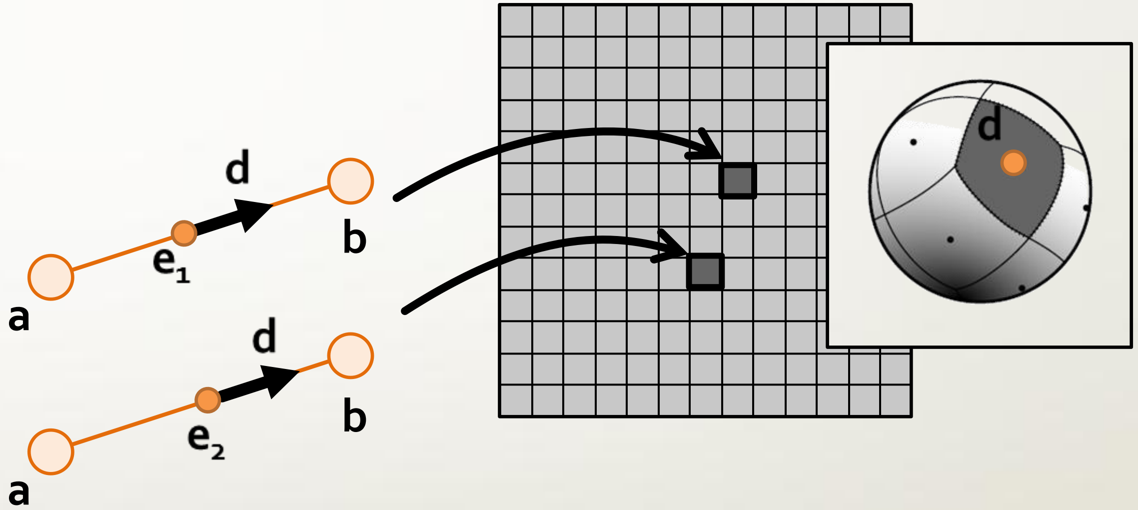
Congruent set extraction

- Efficient indexing
  - Represent pairs as invariant + direction



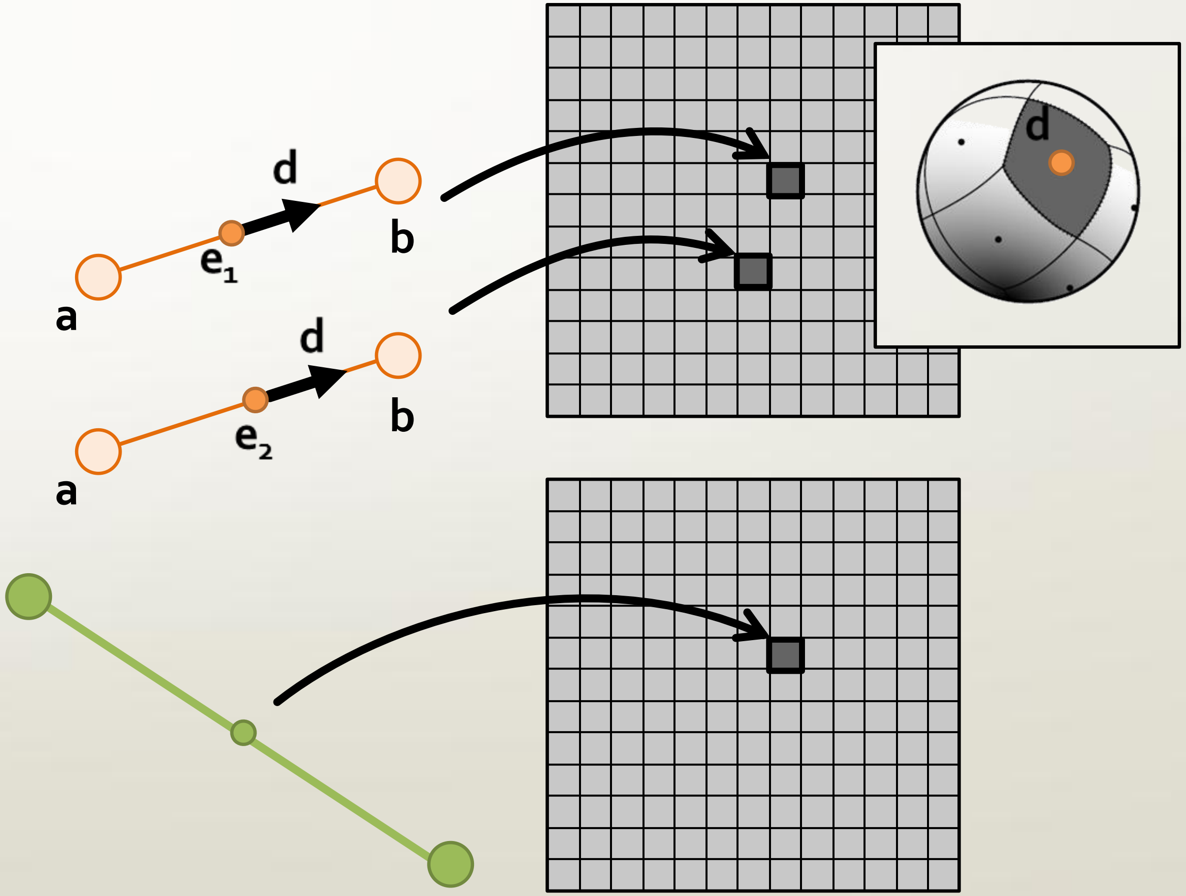
Congruent set extraction

- Efficient indexing
  - Represent pairs as invariant + direction
  - Hash pairs by position and direction



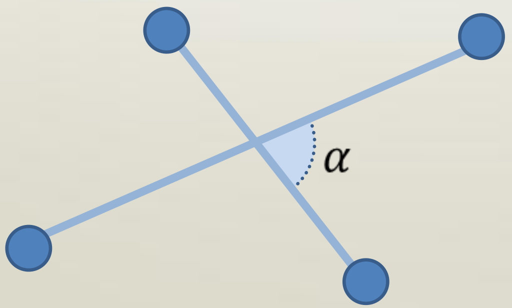
Congruent set extraction

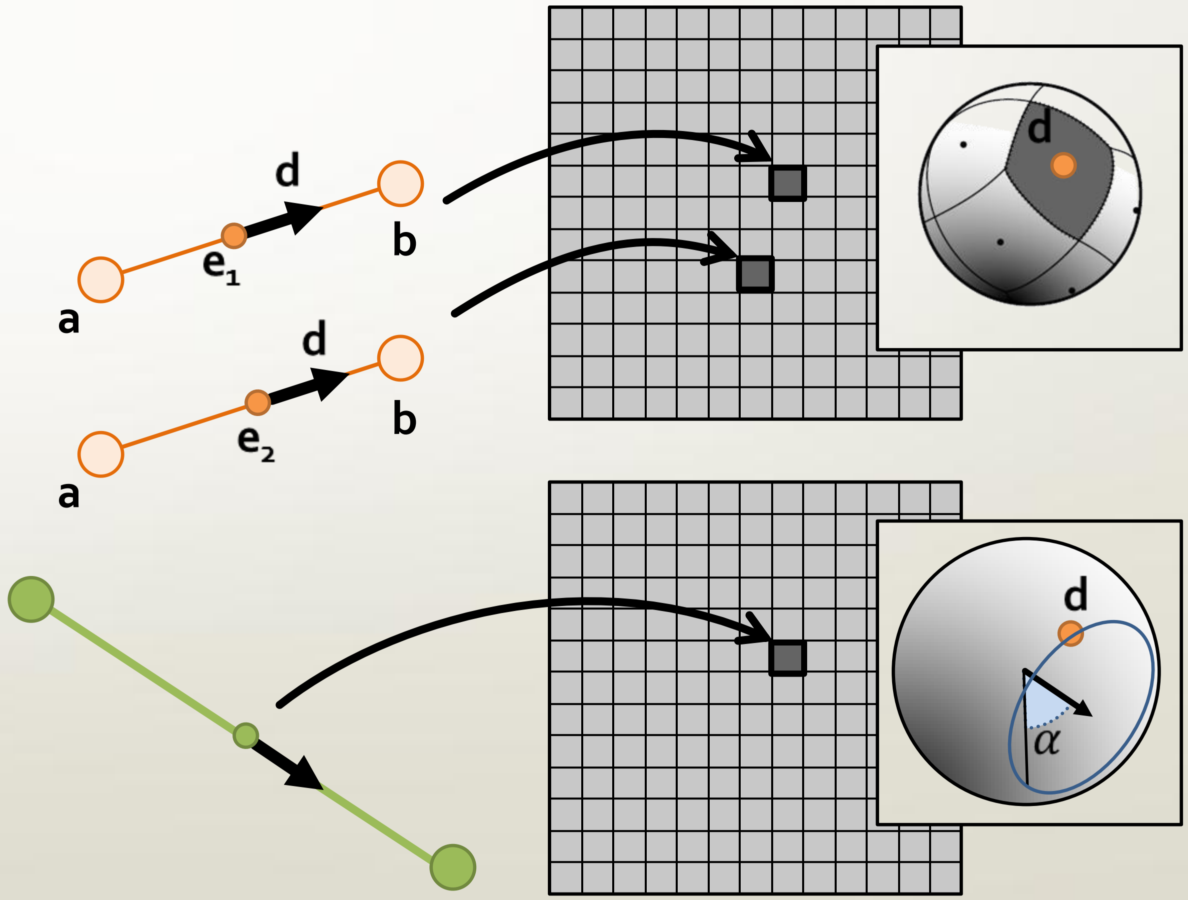
- Efficient indexing
  - Represent pairs as invariant + direction
  - Hash pairs by position and direction
- Query
  - Hash positions (closest invariants)



Congruent set extraction

- Efficient indexing
  - Represent pairs as invariant + direction
  - Hash pairs by position and direction
- Query
  - Hash positions (closest invariants)

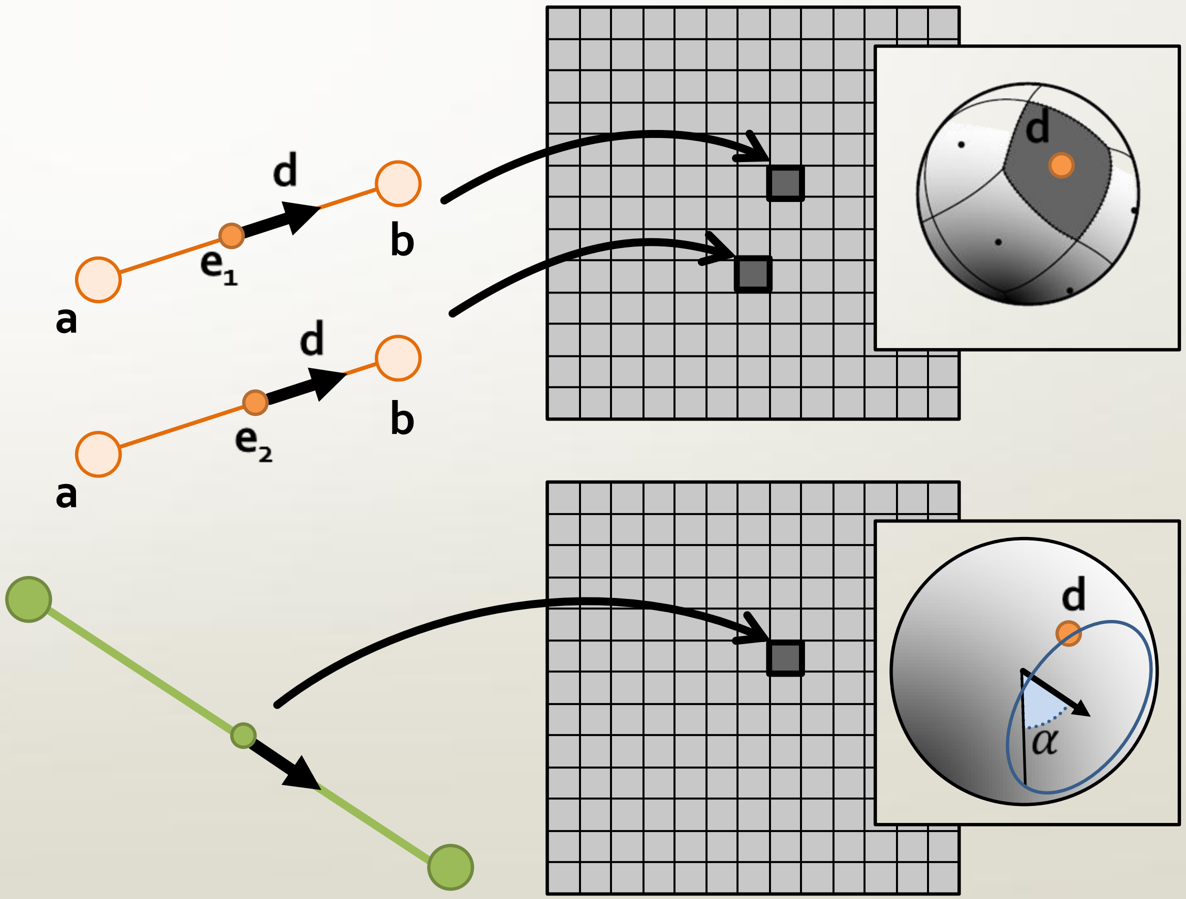




Congruent set extraction

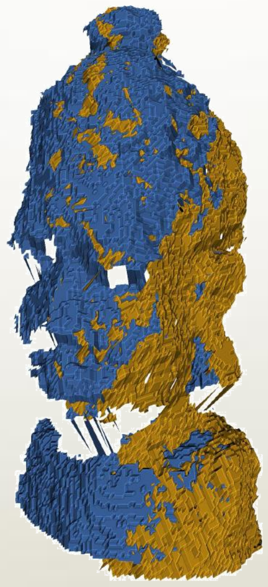
- Efficient indexing
  - Represent pairs as invariant + direction
  - Hash pairs by position and direction
- Query
  - Hash positions (closest invariants)
  - Theoretical complexity: *O(n)*

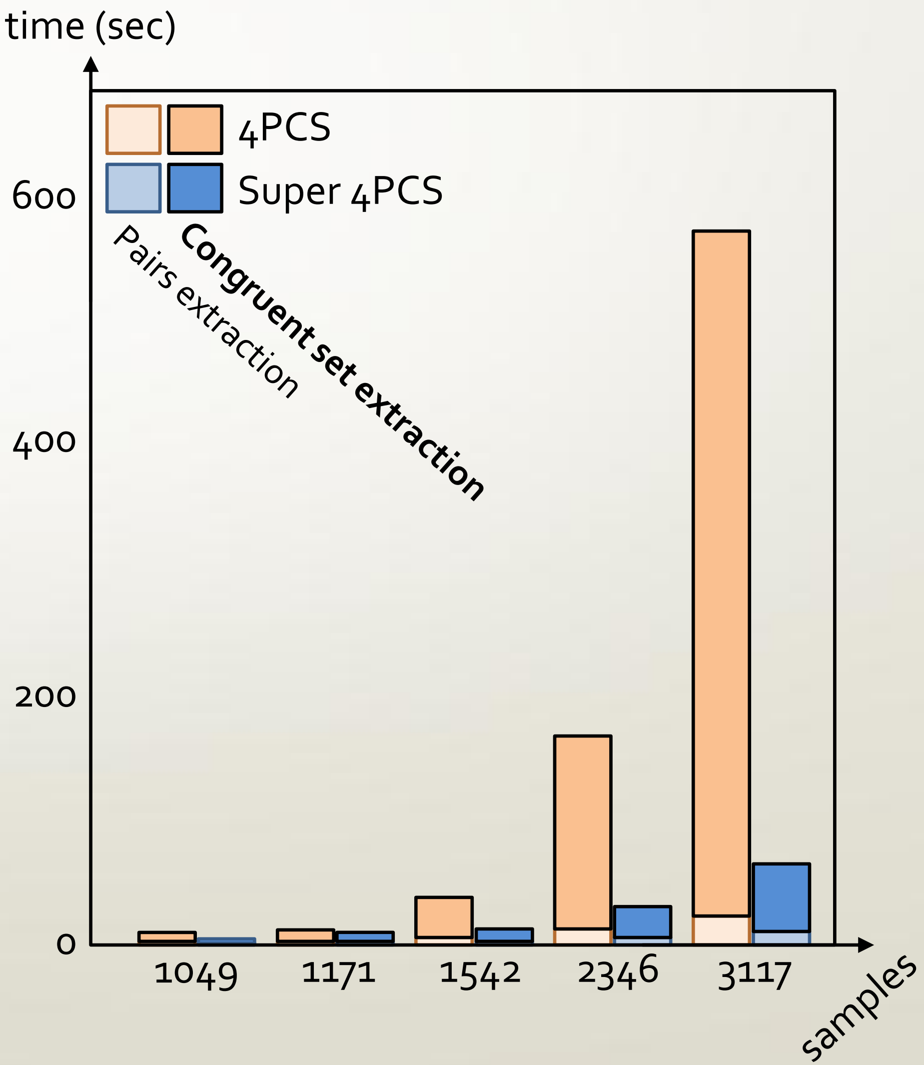
    (see details in the paper)



Congruent set extraction

- Efficient indexing
  - Represent pairs as invariant + direction
  - Hash pairs by position and direction
- Query
  - Hash positions (closest invariants)
  - Theoretical complexity: *O(n)*
  - In practice
    - Runtime: linear
    - Memory overhead: similar to kd-tree





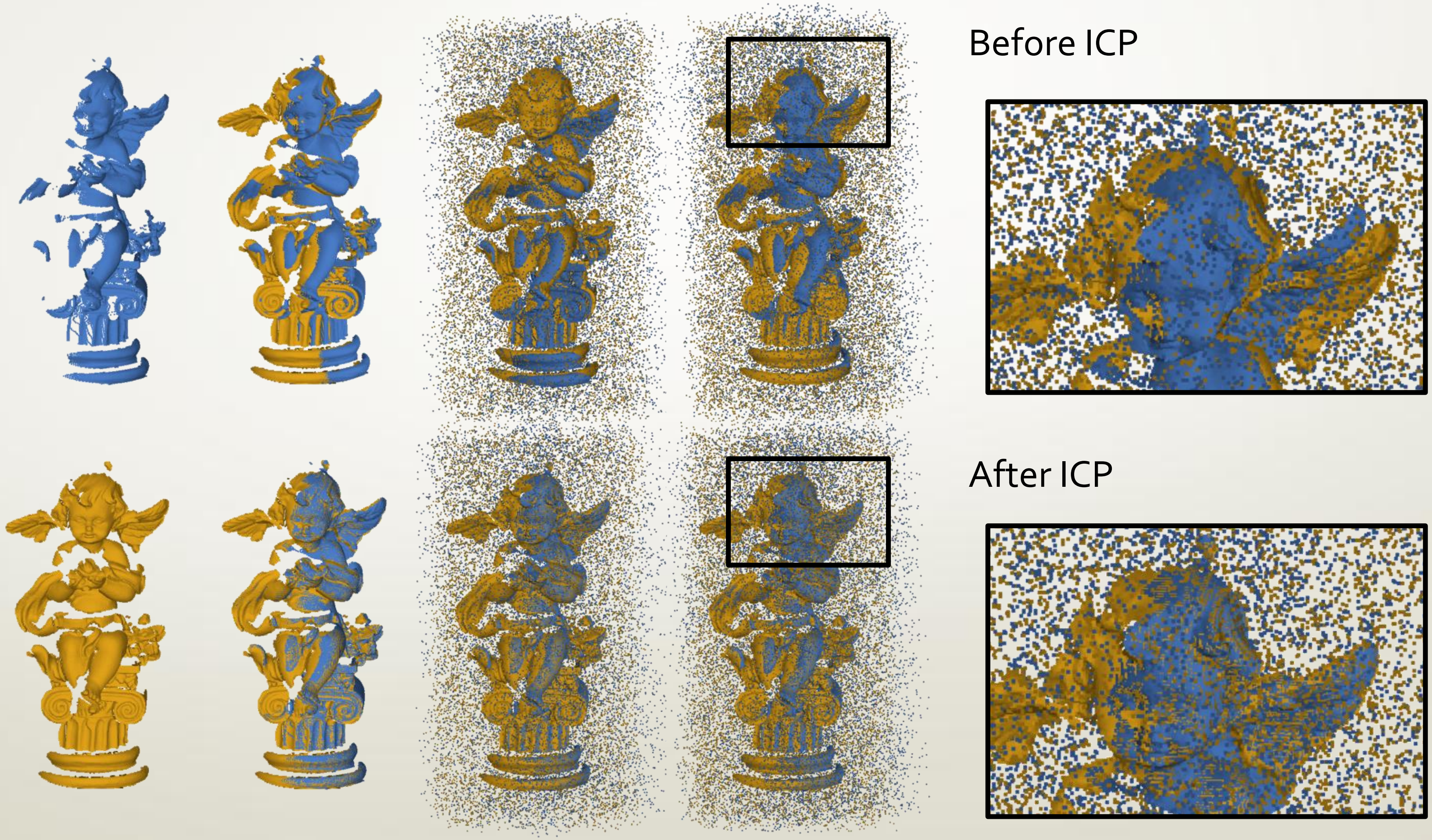
Results

Outliers

35k points

Before ICP

After ICP

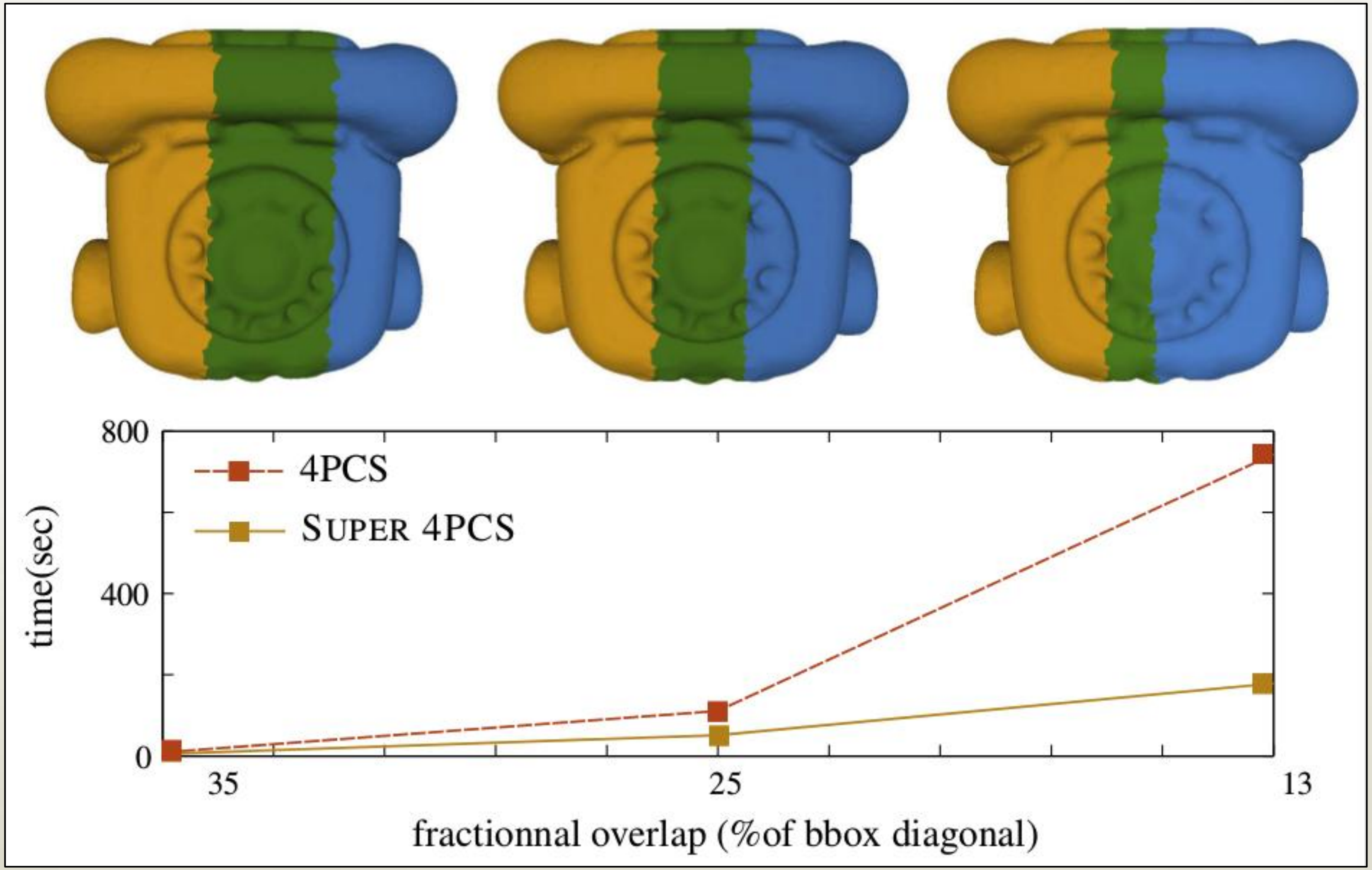


Input 1.4 sec 15 sec 30 sec

Low overlap

(no ICP)

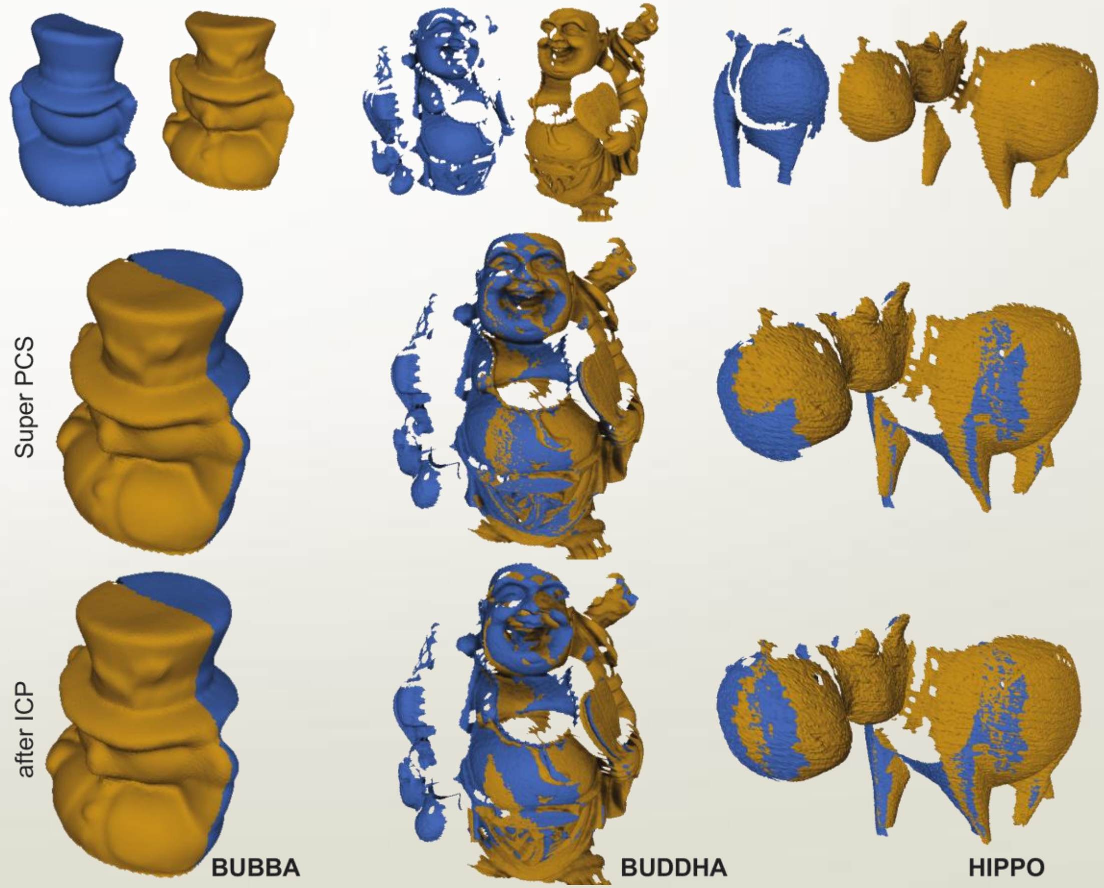
32k points



Low overlap

- Other examples

Model	Points (x1000)	Overlap (%)	4PCS (in sec)	Super 4PCS (in sec)
Bubba	10	5	5	2.5
Buddha	37	20	63	37
Hippo	32	25	11	0.5



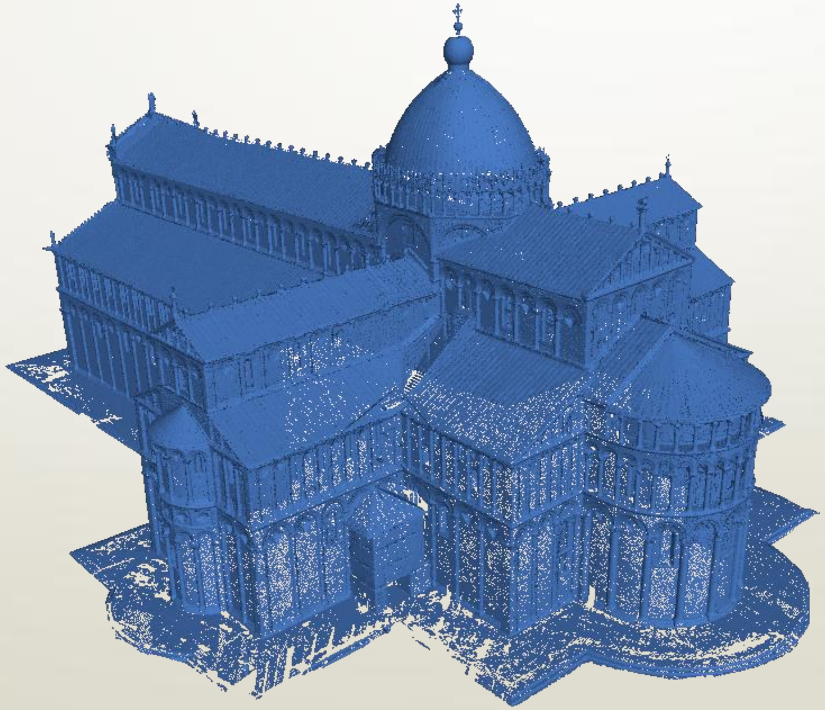
10k points 37k points 32k points

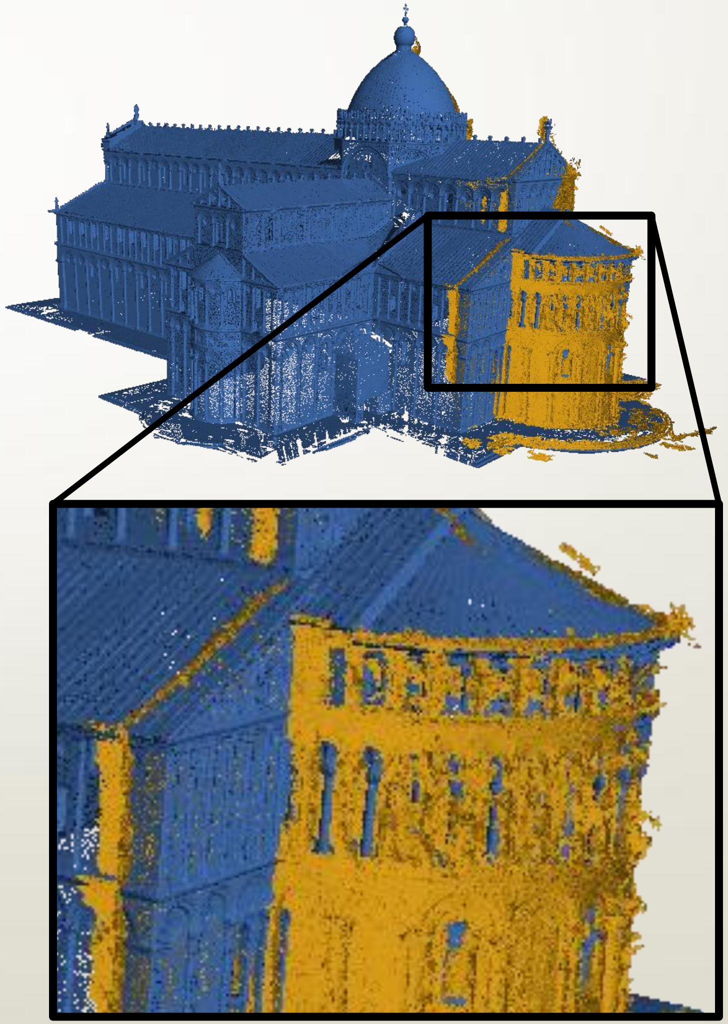
Low overlap + featureless

2 500 points

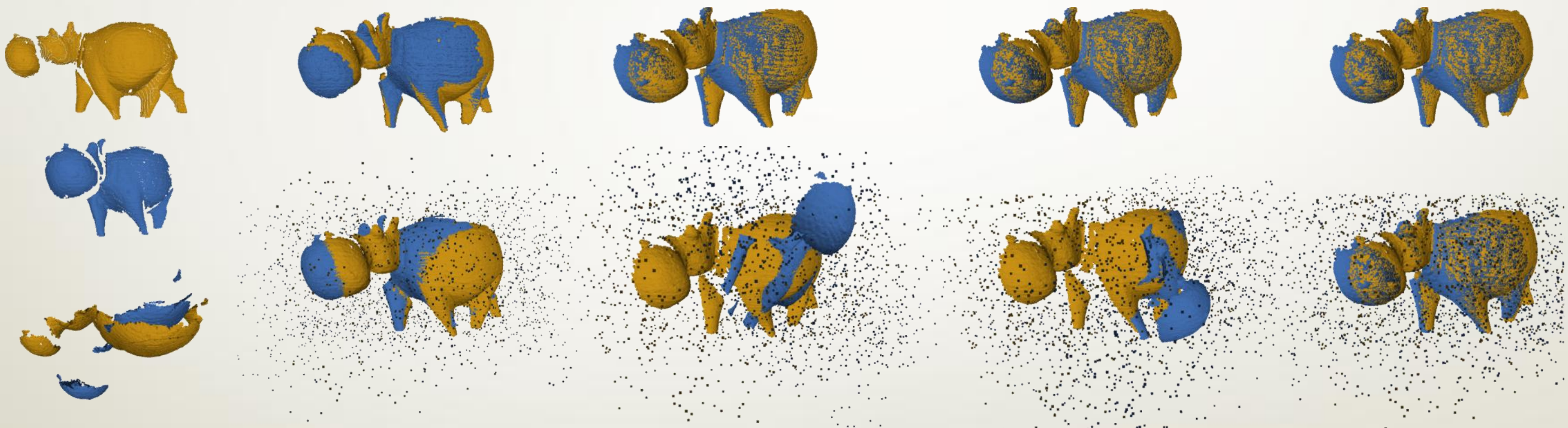
Multi-modal models

- **P**: 2.5M points, multiview stereo
- **Q**: 2.5M points, LIDAR





Range Images alignment



Top: Input
Bottom: initial pose

SUPER 4PCS, no ICP
Top: 0.5 sec
Bottom: 11 sec

ICP
Top: 16 sec
Bottom: 28 sec

Sparse-ICP ($p = 1$)
Top: 12 sec
Bottom: 43 sec

Sparse-ICP ($p = 0.5$)
Top: 71 sec
Bottom: 60 sec

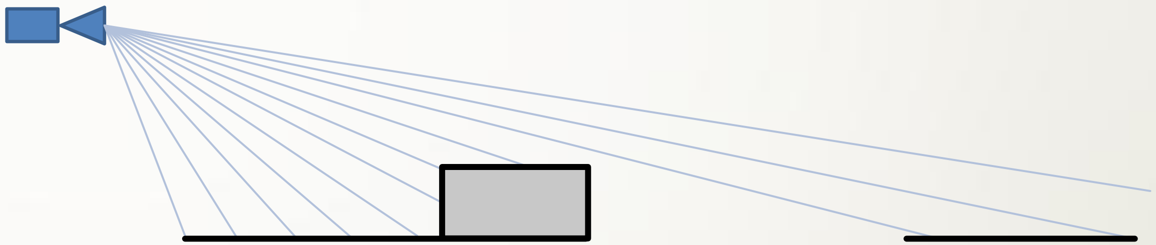
Kinect scans

- Chaining pairwise registration (5/6 frames)

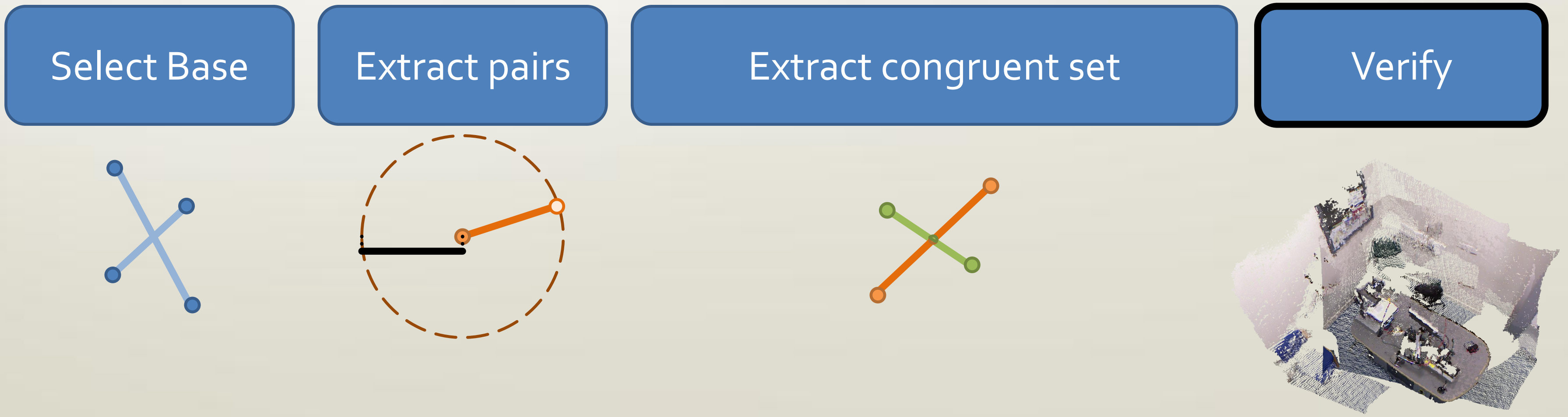


Limitations

- Sampling sensitivity
  - Region Of Interest (ROI)
  - Cannot match *between* the points



- Metric



Conclusion

- Global matching algorithm
  - Running in linear time
  - Unstructured point clouds without normals
  - Can be combined with local descriptors
- Future work
  - Real-time using GPGPU programming
  - Alternative to Kinect Fusion

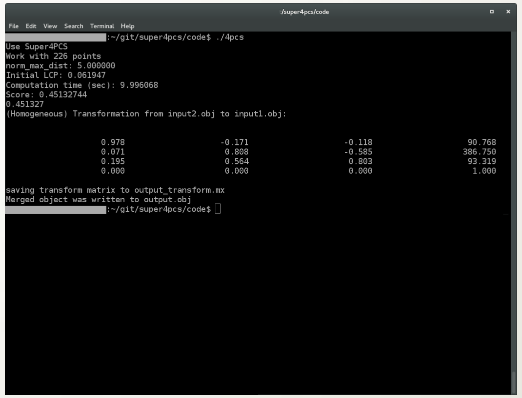
Thank you for your attention

- Super 4PCS
  - Global matching running in linear time

- Acknowledgements
  - Feedback&discussion: Duygu Ceylan, Aron Monszpart
  - SparseICP data and comparisons: Sofien Bouaziz, Andrea Tagliasacchi
  - Pisa dataset: Matteo Dellepiane

- Funding
  - Marie Curie Career Integration Grant
  - ERC Starting Grant SmartGeometry
  - Adobe Research

Code and data goo.gl/uQrhJU



github.com/smartgeometry-ucl/Super4PCS

Features:

- C++
- Based on Eigen
- Structures implemented in arbitrary dimensions