Point Set Alignment

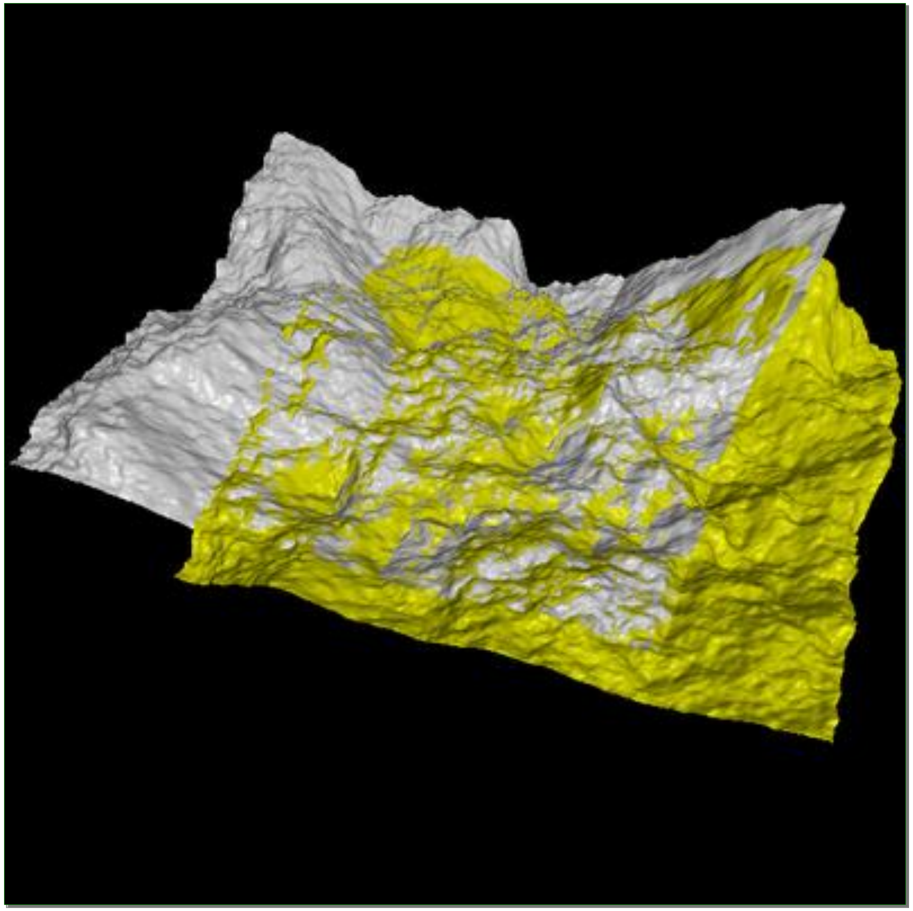
Thomas Funkhouser

COS 526, Fall 2014

Motivation

Point sets to be aligned

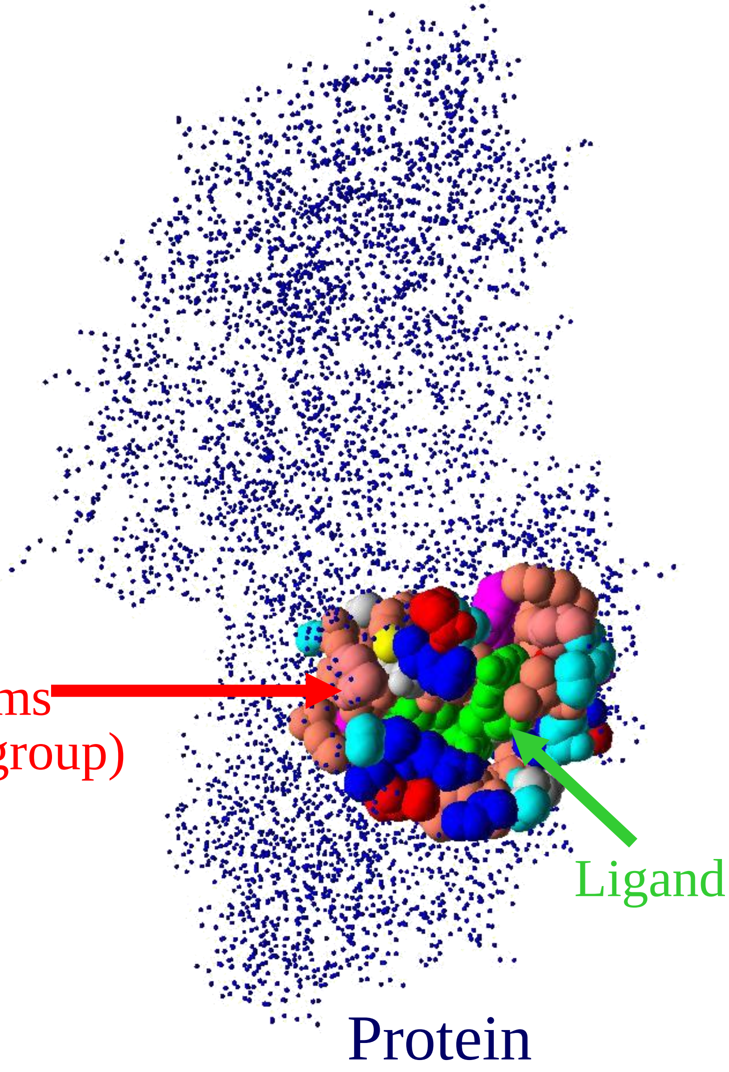
- ➢Range scans
- Image features
- Molecules
- etc.



Motivation

Point sets to be aligned

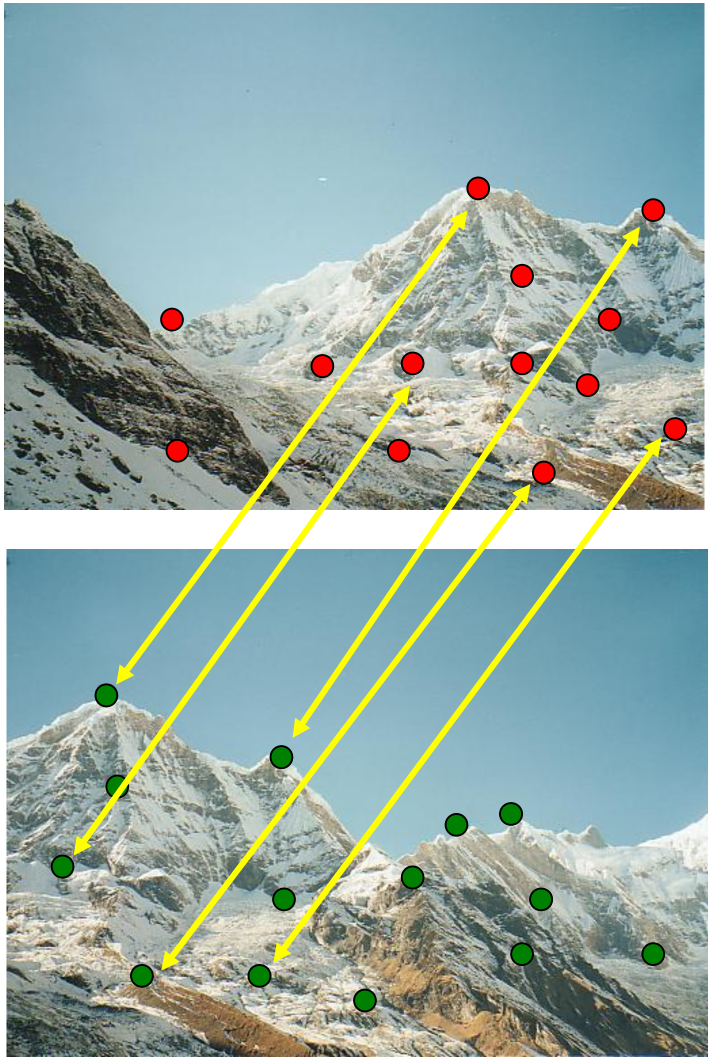
- Range scans
- Image features
- ➢Molecules
- etc.



Motivation

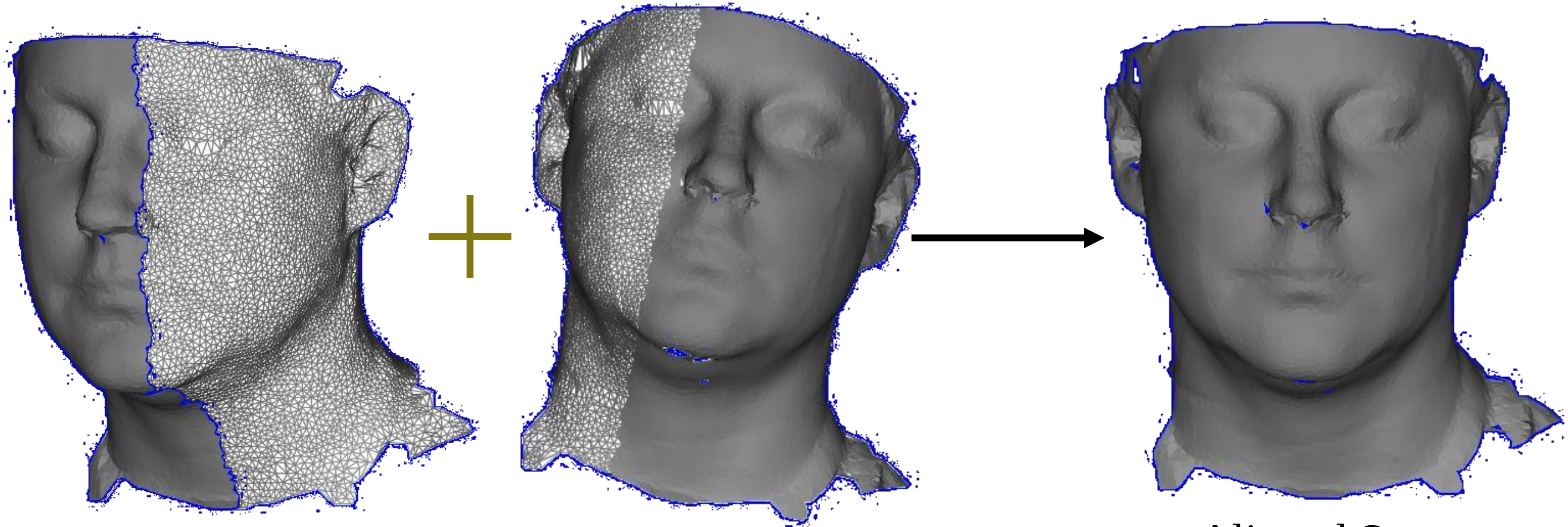
Point sets to be aligned

- Range scans
- ➢Image features
- Molecules
- etc.



Goal

Given two partially overlapping point sets, compute the transformation that merges the two

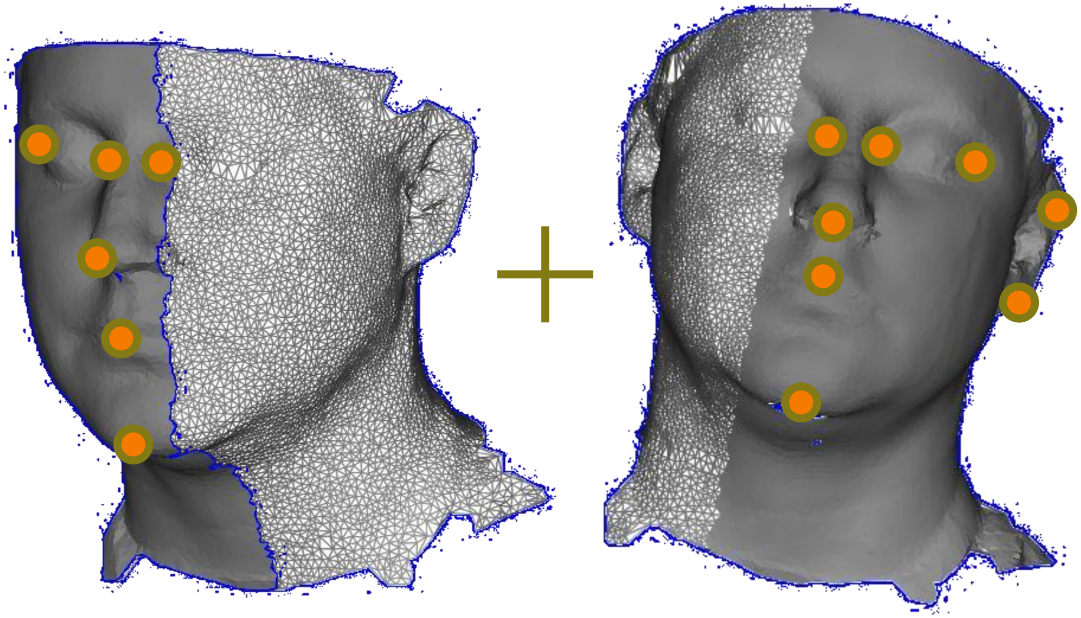


Partially Overlapping Scans

Aligned Scans

General Approach

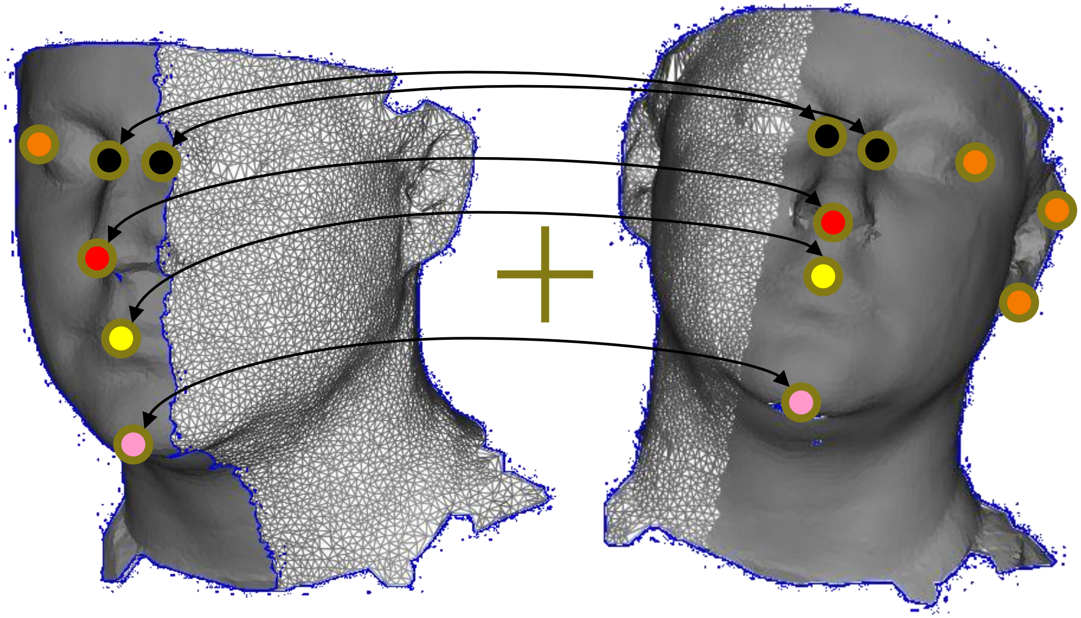
1. Find feature points



Partially Overlapping Scans

General Approach

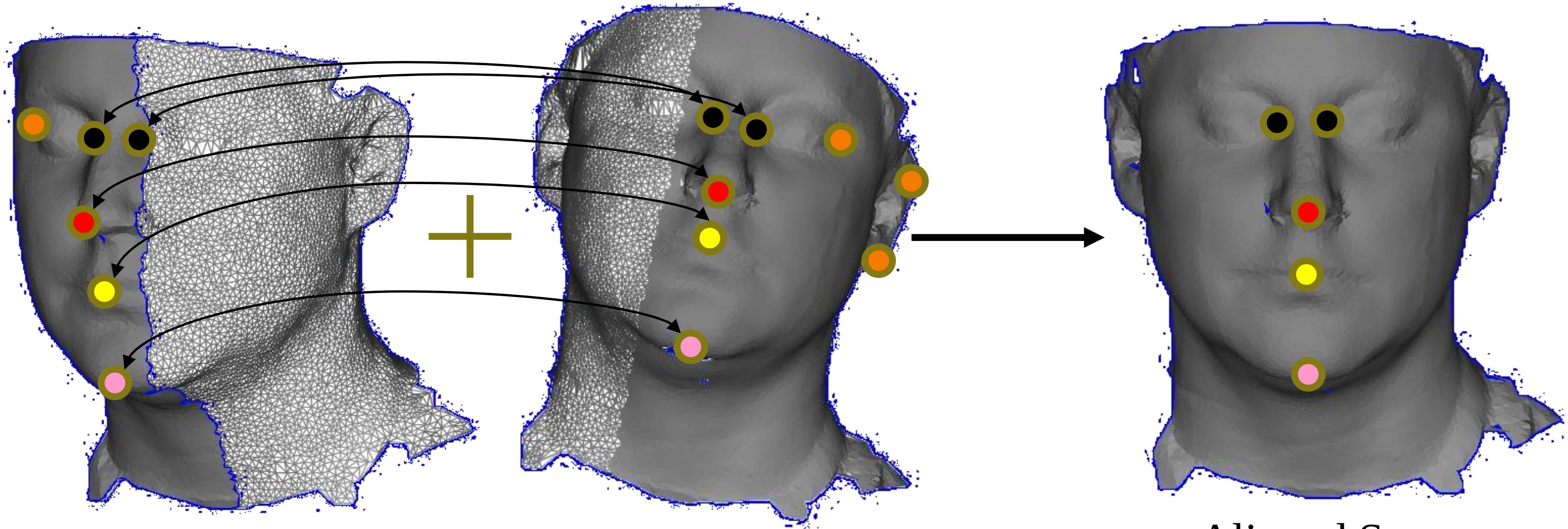
1. Find feature points
2. Establish correspondences



Partially Overlapping Scans

General Approach

1. Find feature points
2. Establish correspondences
3. Compute the aligning transformation

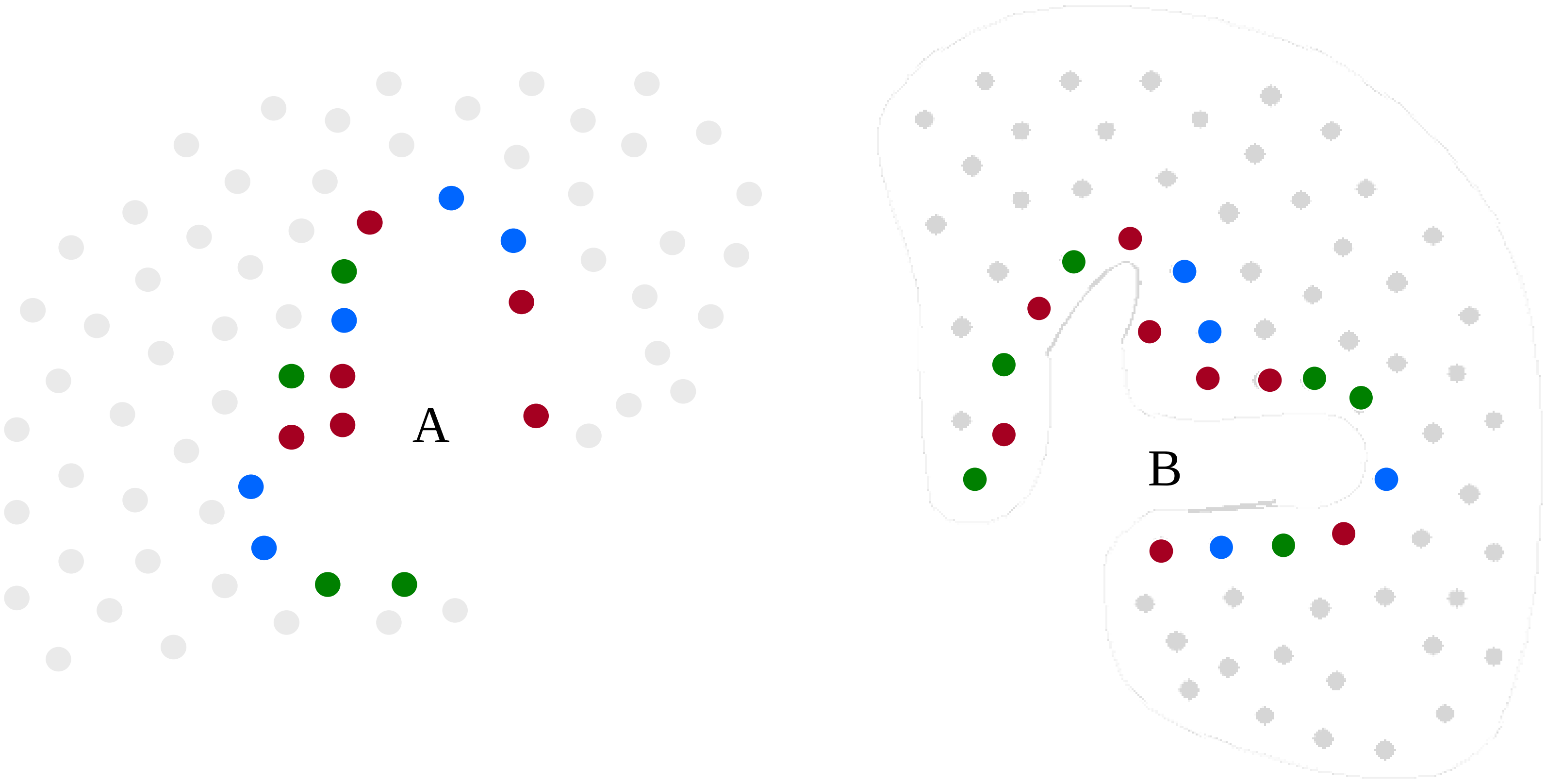


Partially Overlapping Scans

Aligned Scans

Problem

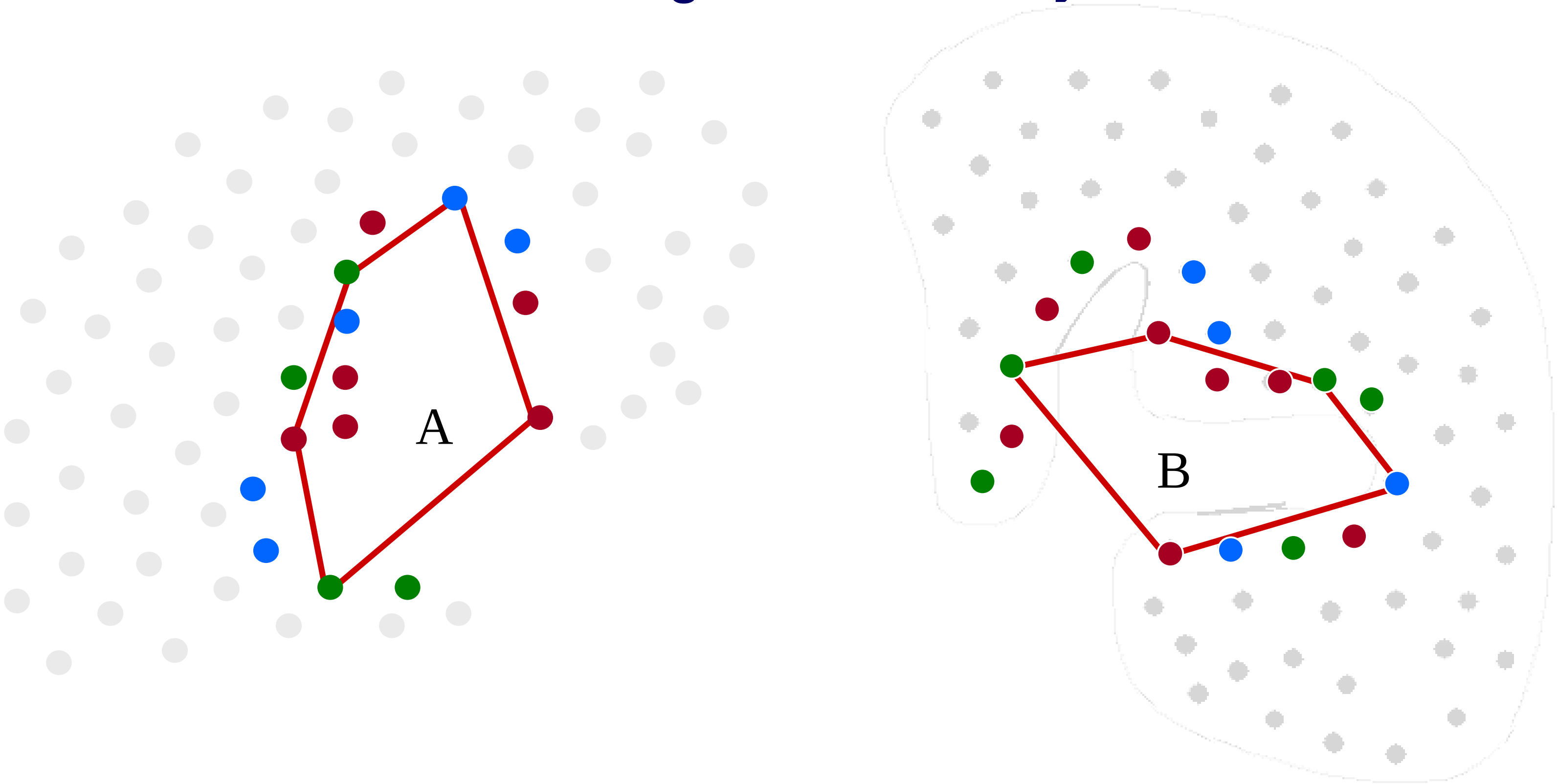
Most problems require aligning a subset of features



Problem

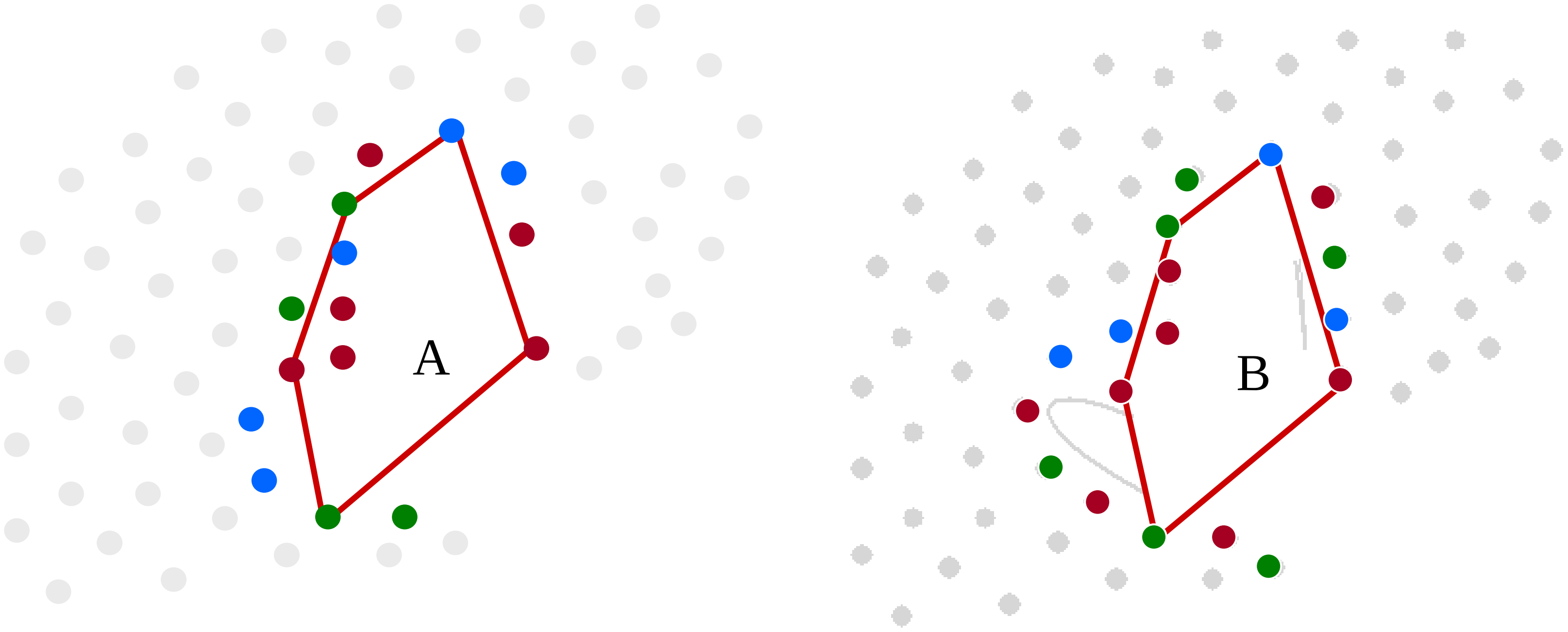
Most problems require aligning a subset of features

- Find the maximal subsets of points that align with error E
- Find the minimum misalignment for any subset of a size S



Observation I

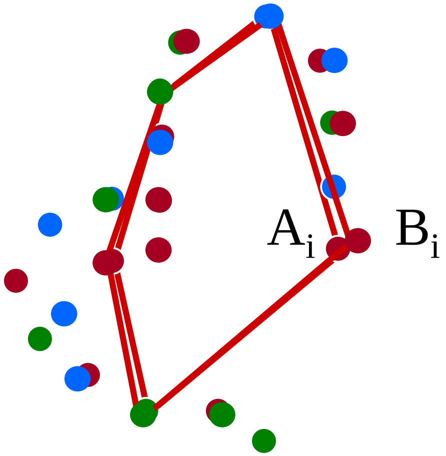
Calculating the aligning transformation is usually easy if correspondences are known (proposed)



$$RMSD(A,B) = \sqrt{\sum_{i=1}^{N}(A_i - B_i)^2}$$

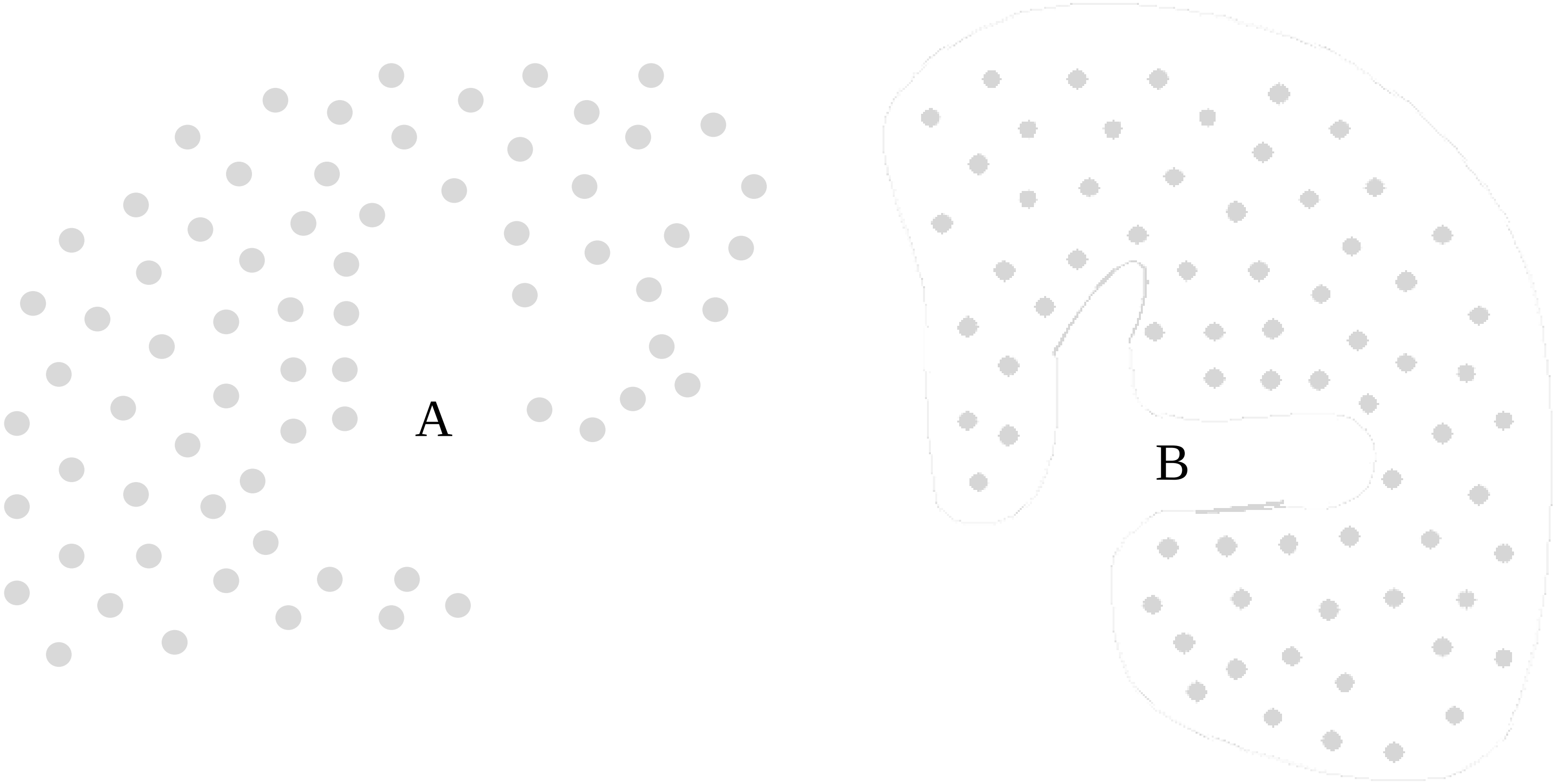
Observation II

Calculating the correspondences is usually easy if the aligning transformation is known (proposed)



Challenge

The challenge is to discover the correspondences and aligning transformation together



Outline

Introduction

Point set matching

- ➢ Brute force search
- RANSAC
- Geometric hashing
- Assocation graphs
- Generalized Hough transform
- Iterative closest point

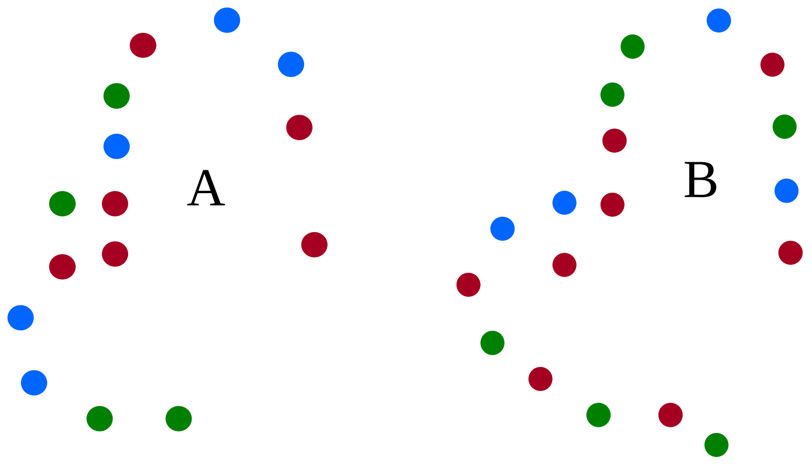
Methods used for RGB-D scanning

Discussion

Brute Force Search

Simple method:

- Try all possible sets of point correspondences
- Score the alignment for each one



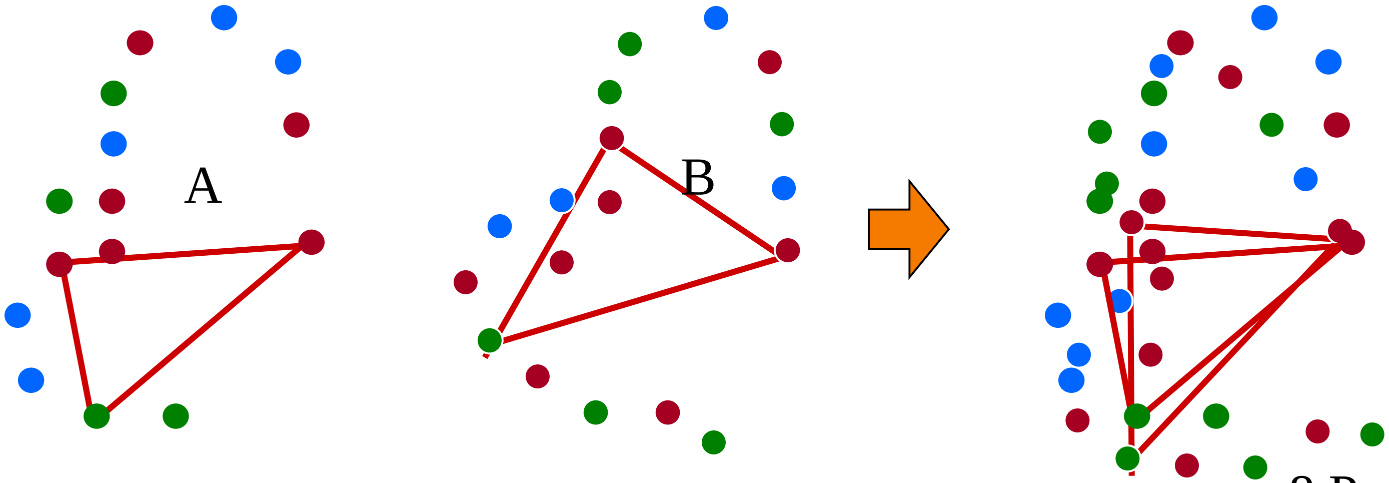
Problem:

- $O(n^m)$ possible sets of m correspondences among n points

Brute Force Search

Simple method:

- Try all possible sets of point correspondences
- Score the alignment for each one



8 Point aligned
RMSD = 3.1

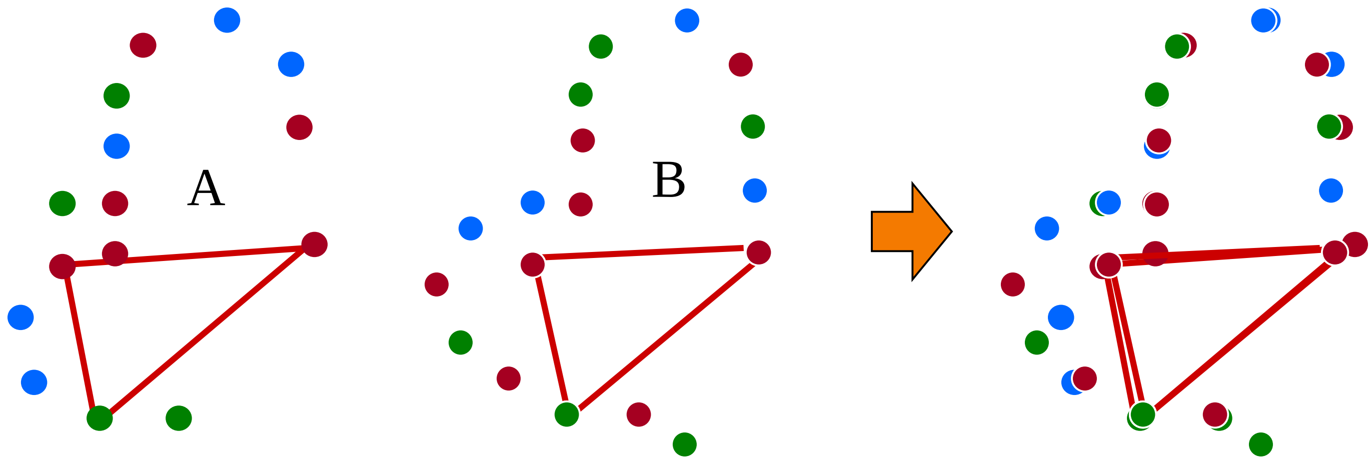
Problem:

- O(n^m) possible sets of m correspondences among n points

Brute Force Search

Simple method:

- Try all possible sets of point correspondences
- Score the alignment for each one (e.g., RMSD)



All points aligned
RMSD = 0.2

Problem:

- $O(n^m)$ possible sets of m correspondences among n points

Outline

Introduction

Point set matching

- Brute force search
- ➢RANSAC
- Geometric hashing
- Assocation graphs
- Generalized Hough transform
- Iterative closest point

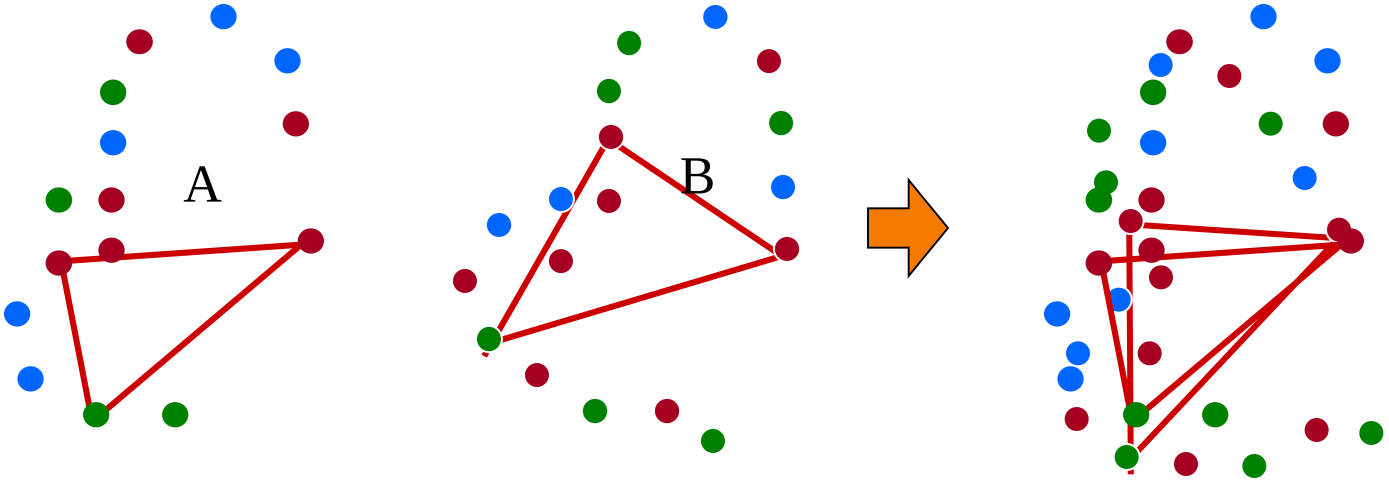
Methods used for RGB-D scanning

Discussion

RANSAC

Randomly sample set of possible correspondences

- Randomly generate a small set of point correspondences
- Compute the aligning transformation for correspondences
  Score how well other points align after that transformation
- Remember the best transformation



RANSAC

RANSAC loop:

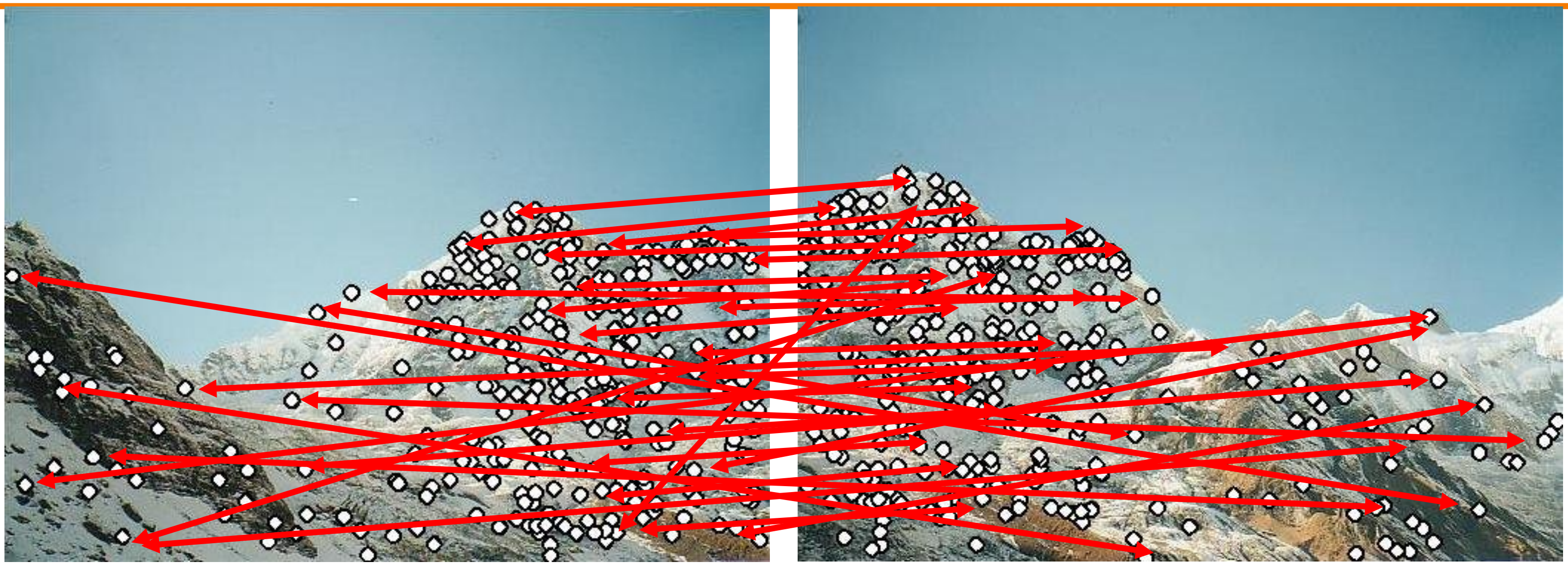
1. Select k matches (at random)
2. Compute transformation T aligning those matches
3. Find *inlier matches* where $d(p_i', \mathbf{T}p_i) < \varepsilon$
4. Re-compute T to align on all of its inliers
5. Re-find *inlier matches* where $d(p_i', \mathbf{T}p_i) < \varepsilon$
6. T*=T if has T largest set of inliers seen so far

Warp image by T*

Composite images

Slide credit: Steve Seitz

RANSAC for Image Mosaics



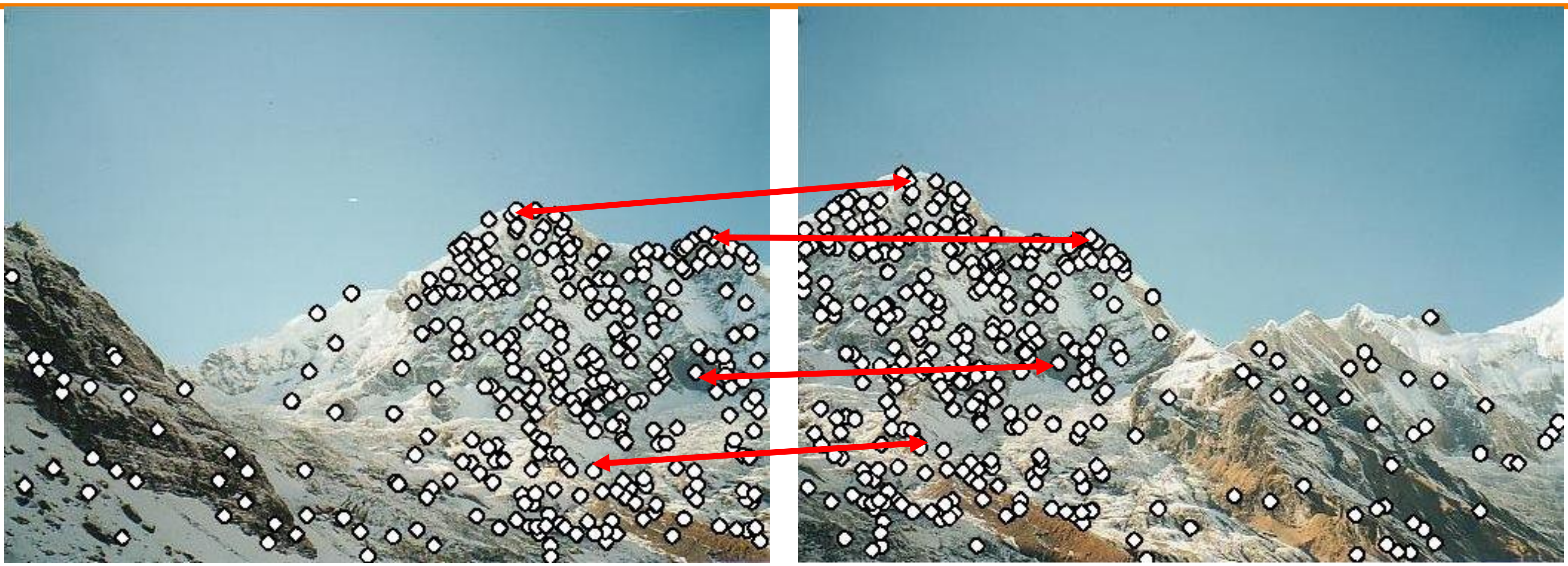
RANSAC loop:

1. Select four matches (at random)
2. Compute homography H aligning those matches
3. Find *inlier matches* where $d(p_i', \mathbf{H}p_i) < \varepsilon$
4. Re-compute H to align on all of its inliers (least squares)
5. Re-find *inlier matches* where $d(p_i', \mathbf{H}p_i) < \varepsilon$
6. H*=H if has H largest set of inliers seen so far

Warp image by H* and composite images

Source: L. Lazebnik

RANSAC for Image Mosaics



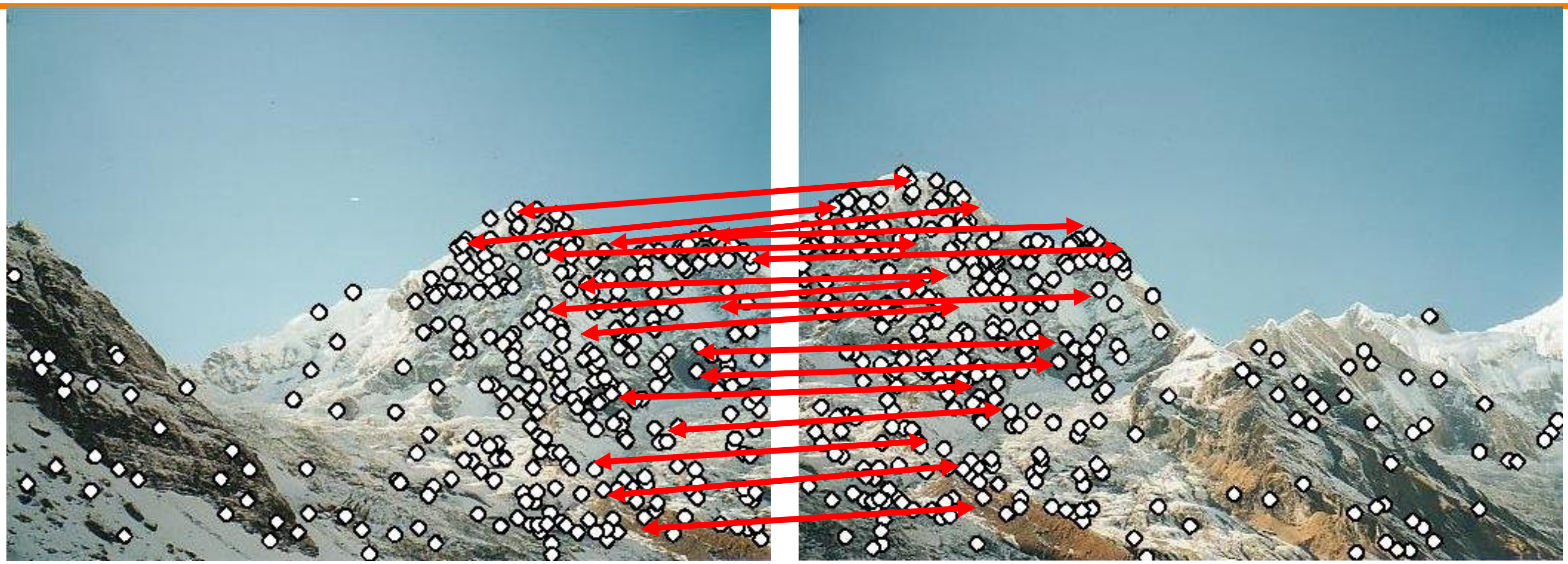
RANSAC loop:

1. Select four matches (at random)
2. Compute homography H aligning those matches
3. Find *inlier matches* where $d(p_i', \mathbf{H}p_i) < \varepsilon$
4. Re-compute H to align on all of its inliers (least squares)
5. Re-find *inlier matches* where $d(p_i', \mathbf{H}p_i) < \varepsilon$
6. H*=H if has H largest set of inliers seen so far

Warp image by H* and composite images

Source: L. Lazebnik

RANSAC for Image Mosaics



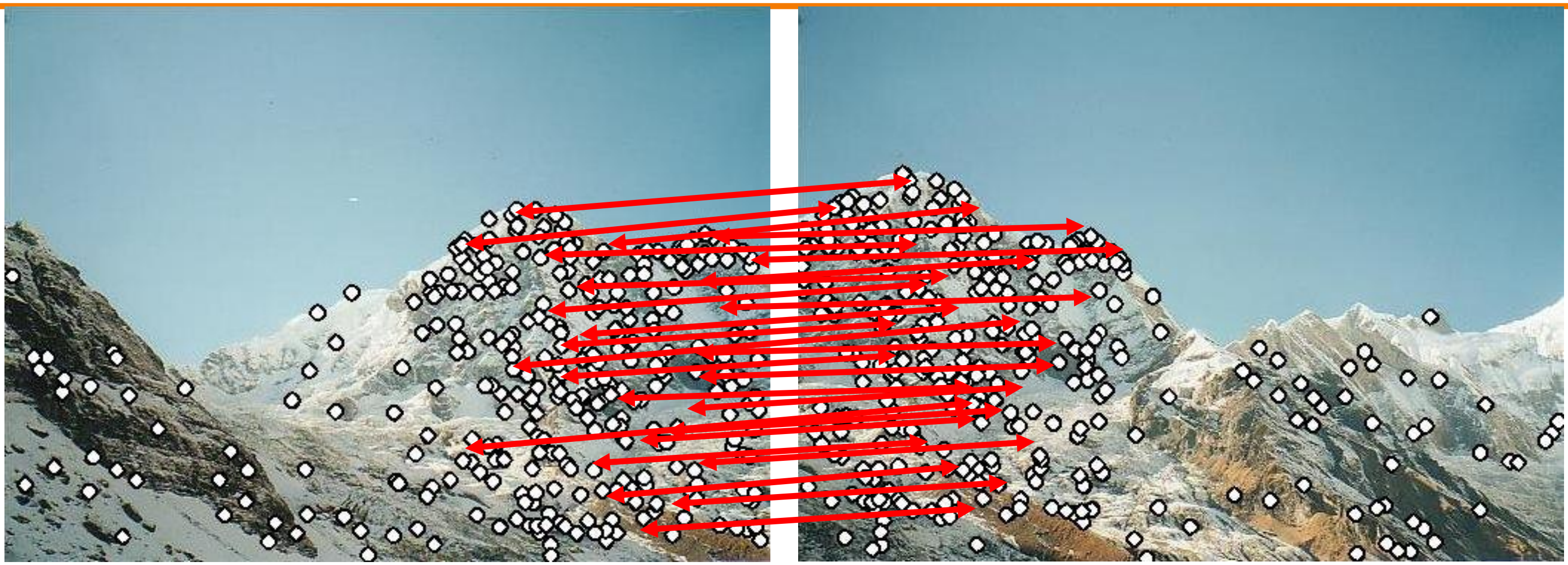
RANSAC loop:

1. Select four matches (at random)
2. Compute homography H aligning those matches
3. Find *inlier matches* where $d(p_i', \mathbf{H}p_i) < \varepsilon$
4. Re-compute H to align on all of its inliers (least squares)
5. Re-find *inlier matches* where $d(p_i', \mathbf{H}p_i) < \varepsilon$
6. H*=H if has H largest set of inliers seen so far

Warp image by H* and composite images

Source: L. Lazebnik

RANSAC for Image Mosaics



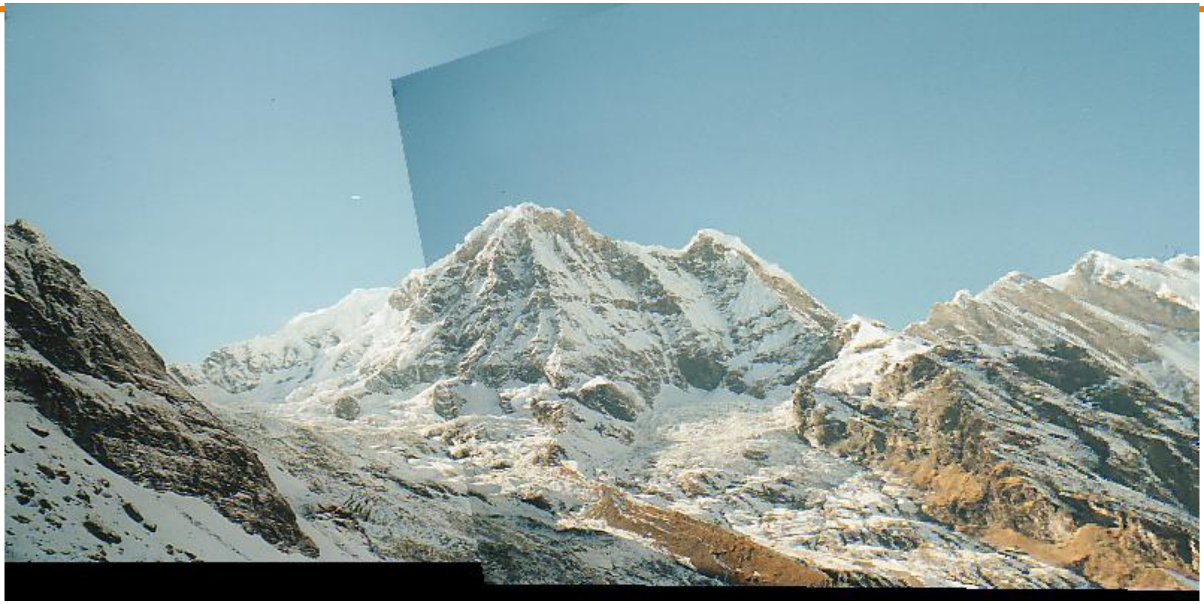
RANSAC loop:

1. Select four matches (at random)
2. Compute homography H aligning those matches
3. Find *inlier matches* where $d(p_i', \mathbf{H}p_i) < \varepsilon$
4. Re-compute H to align on all of its inliers (least squares)
5. Re-find *inlier matches* where $d(p_i', \mathbf{H}p_i) < \varepsilon$
6. H*=H if has H largest set of inliers seen so far

Warp image by H* and composite images

Source: L. Lazebnik

RANSAC for Image Mosaics



RANSAC loop:

1. Select four matches (at random)
2. Compute homography H aligning those matches
3. Find *inlier matches* where $d(p_i', \mathbf{H}p_i) < \varepsilon$
4. Re-compute H to align on all of its inliers (least squares)
5. Re-find *inlier matches* where $d(p_i', \mathbf{H}p_i) < \varepsilon$
6. H*=H if has H largest set of inliers seen so far

Warp image by H* and composite images

Source: L. Lazebnik

Outline

Introduction

Point set matching

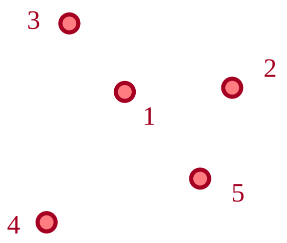
- Brute force search
- RANSAC
- ➢Geometric hashing
- Assocation graphs
- Generalized Hough transform
- Iterative closest point

Methods used for RGB-D scanning

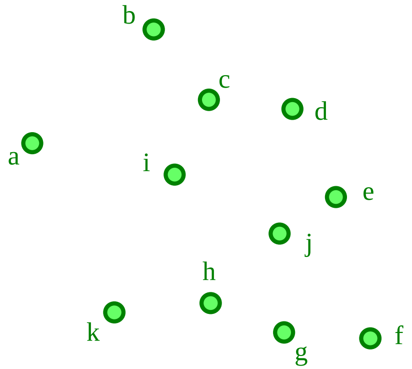
Discussion

Geometric Hashing

Discretize transformations and scoring



Point Set A

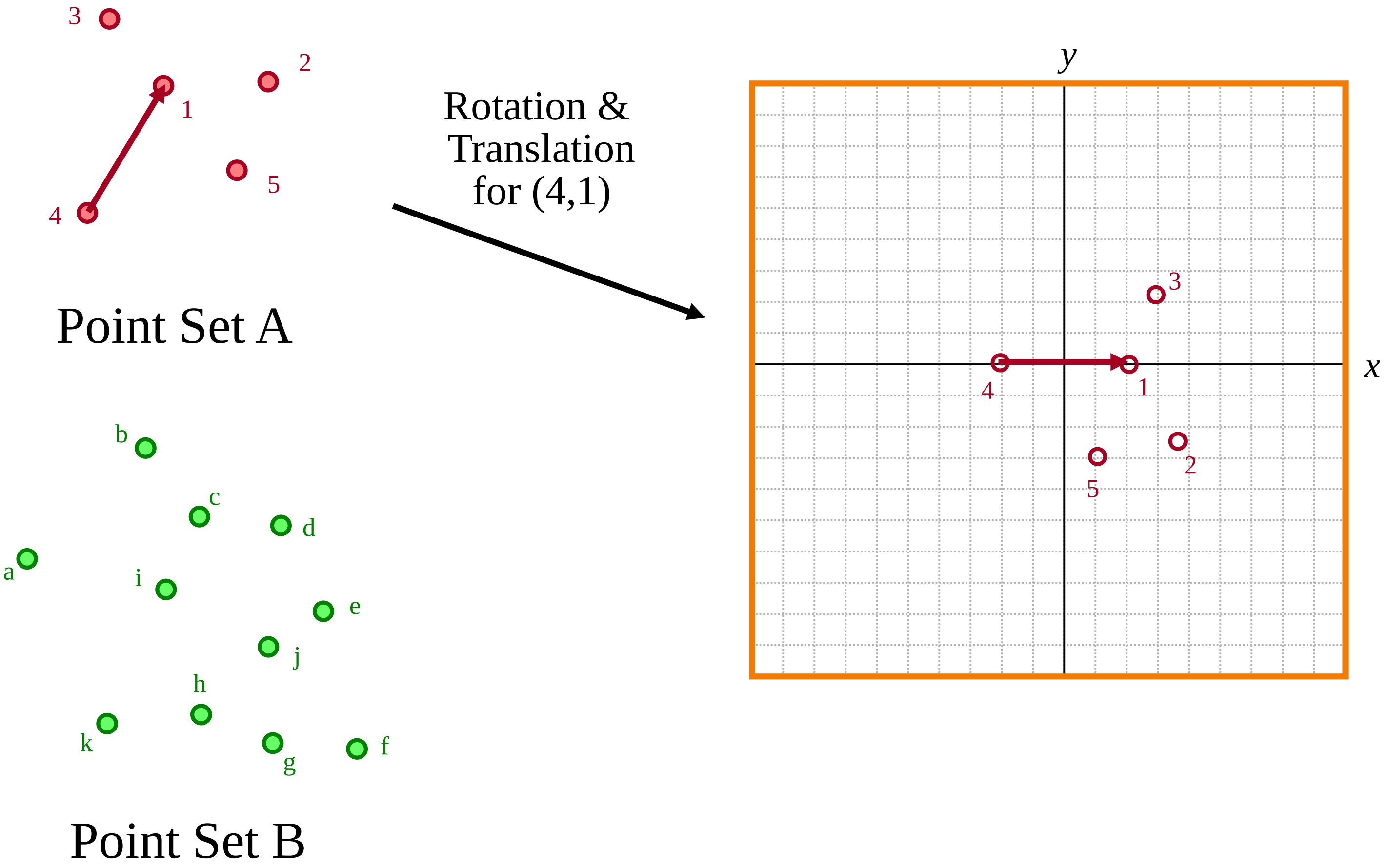


Point Set B

[Wolfson97]

Geometric Hashing

Discretize transformations and scoring



Rotation & Translation for (4,1)

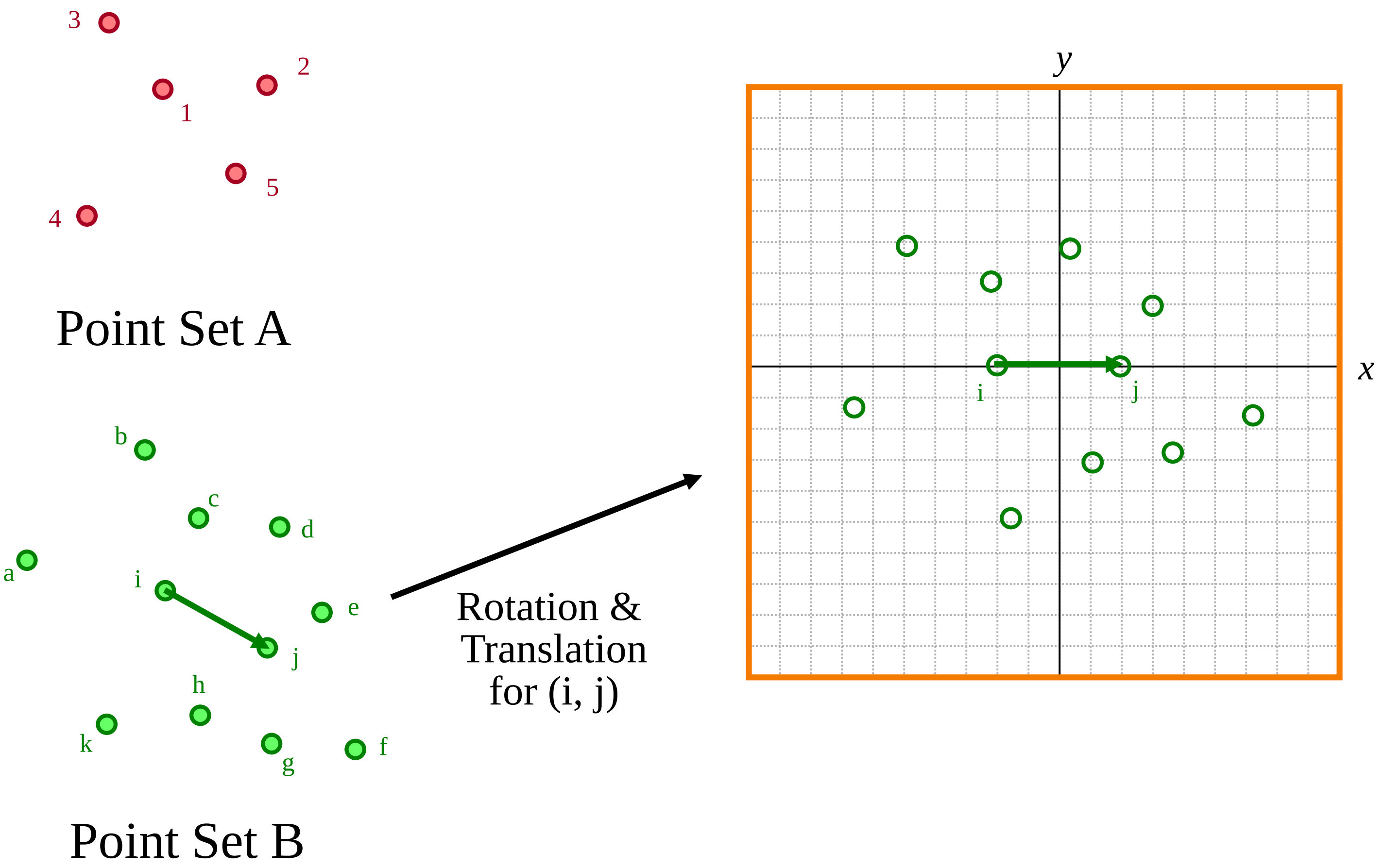
Point Set A

Point Set B

[Wolfson97]

Geometric Hashing

Discretize transformations and scoring



Point Set A

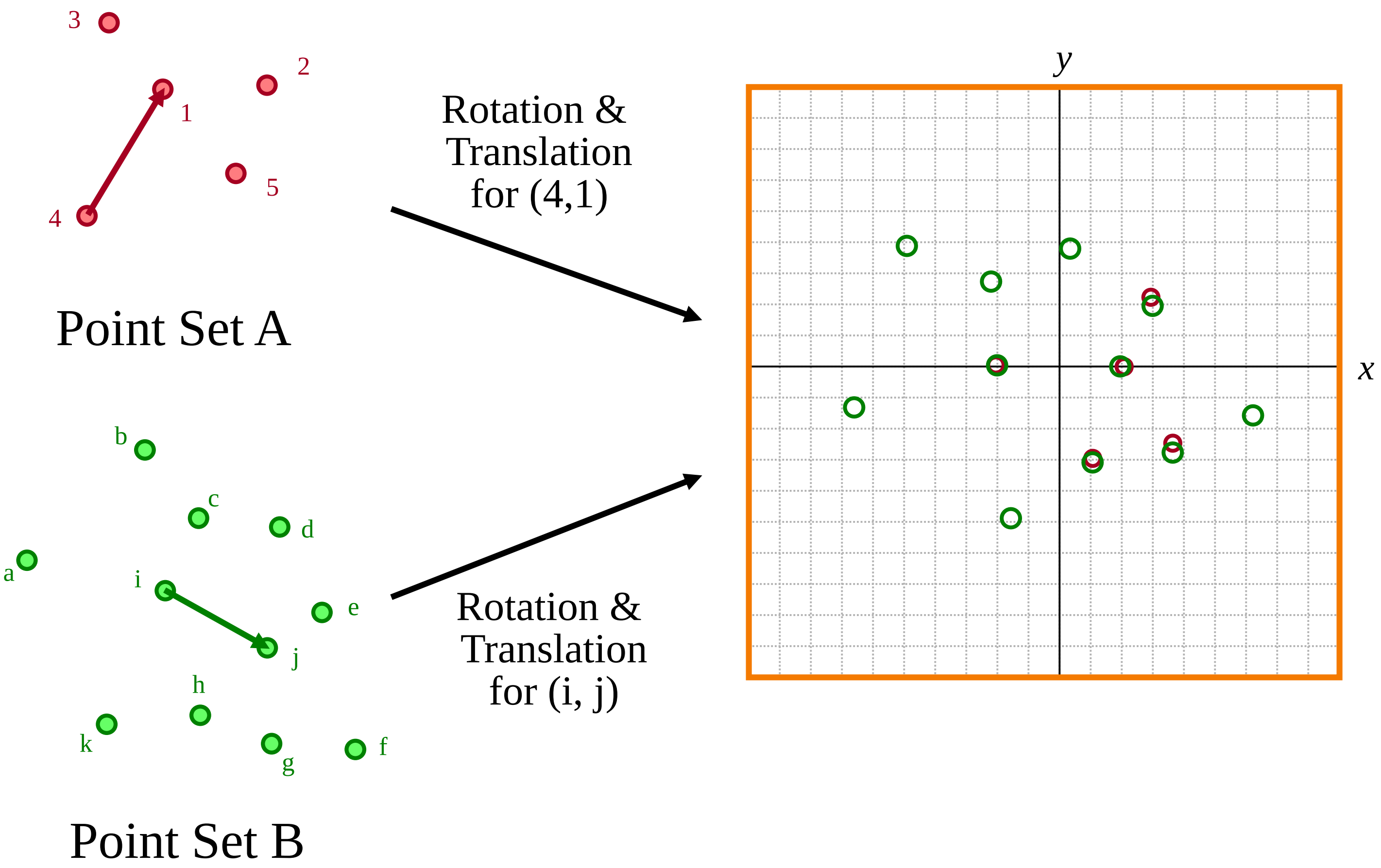
Point Set B

Rotation &
Translation
for (i, j)

[Wolfson97]

Geometric Hashing

Discretize transformations and scoring



Point Set A

Rotation & Translation for (4,1)

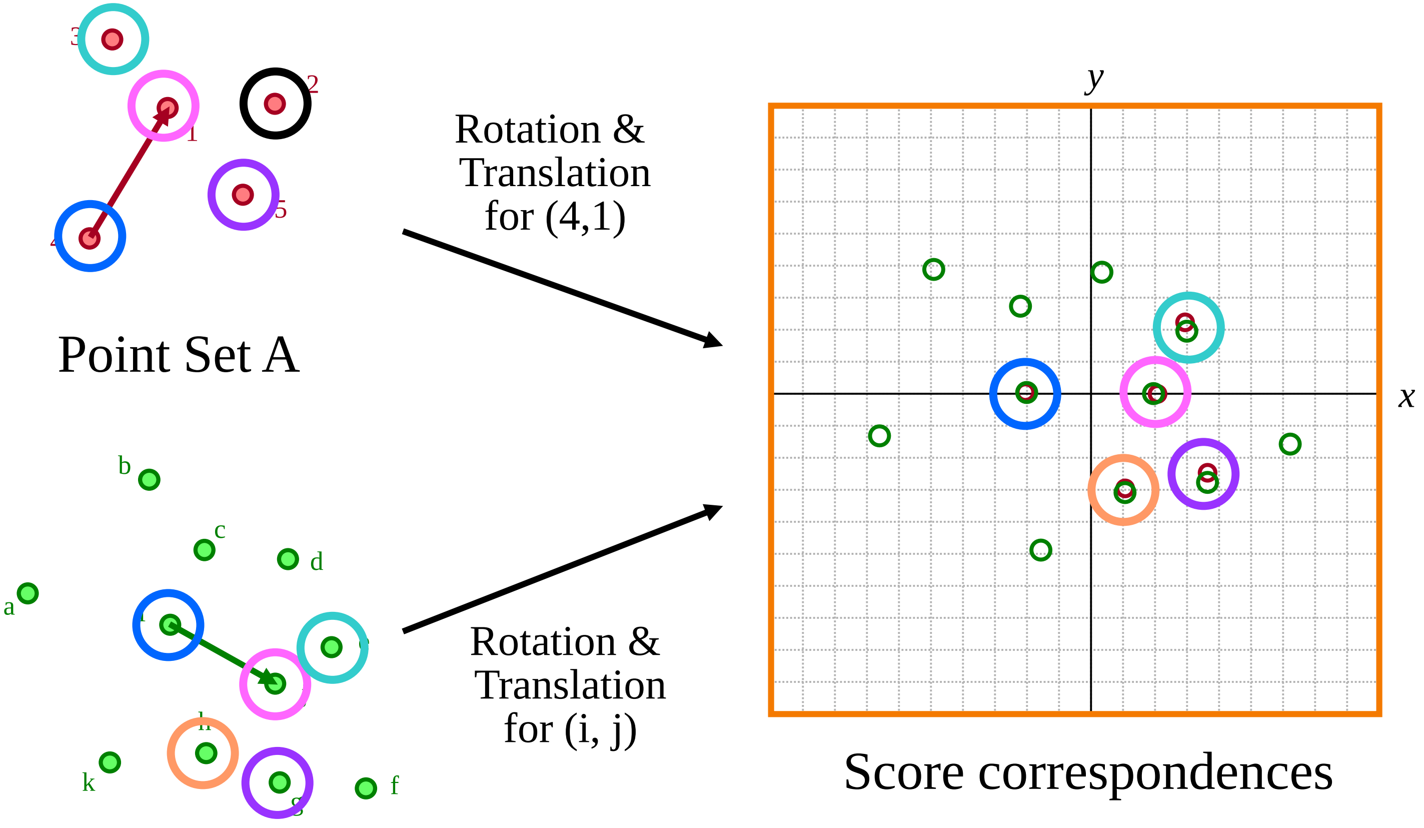
Point Set B

Rotation & Translation for (i, j)

[Wolfson97]

Geometric Hashing

Discretize transformations and scoring



Point Set A

Rotation & Translation for (4,1)

Rotation & Translation for (i, j)

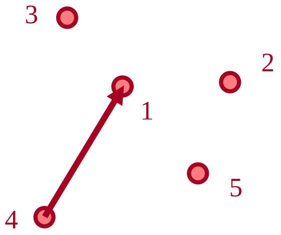
Score correspondences

Point Set B

[Wolfson97]

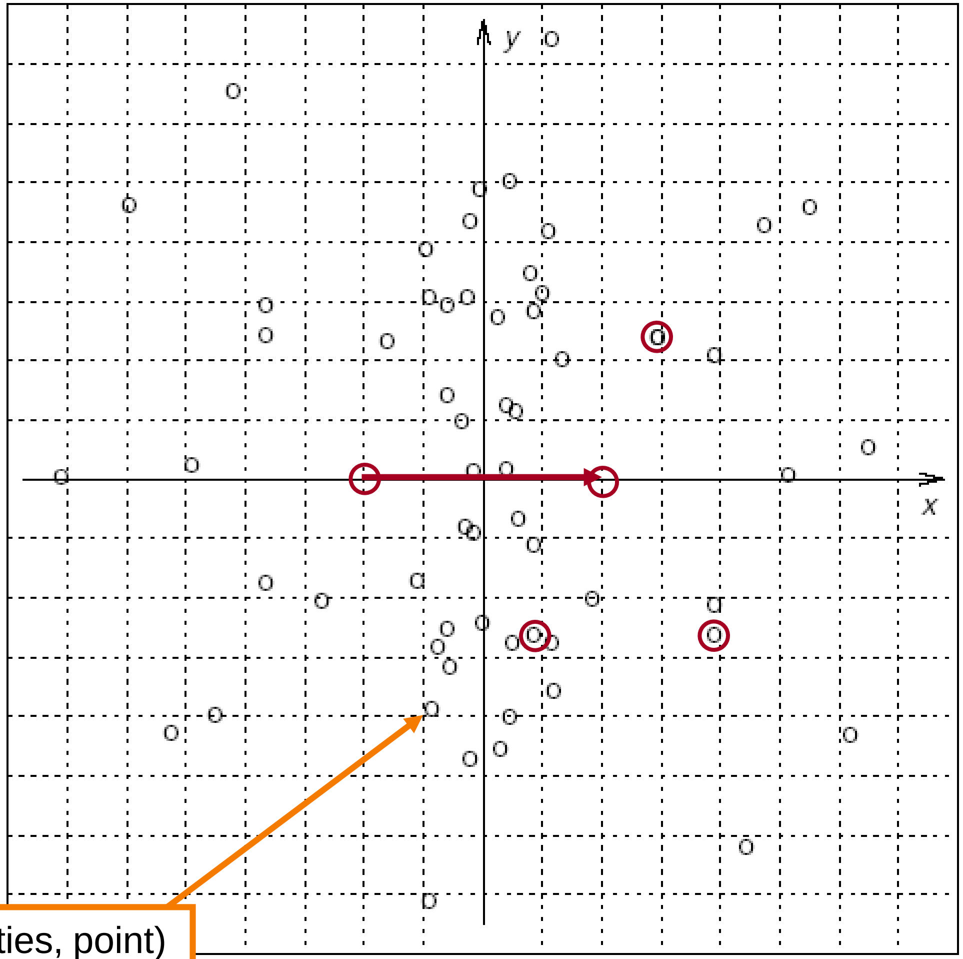
Geometric Hashing

Preprocessing



Point Set
in Database

Rotation &
translation
for all pairs
of points in
all point sets



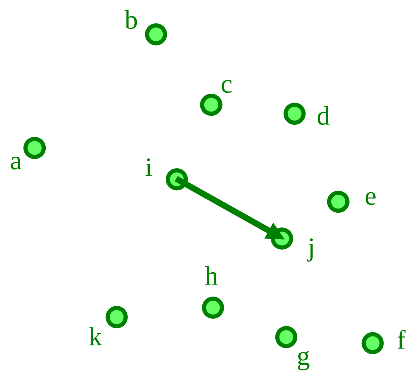
Store (point set, ref. frame, properties, point)
for every transformed point in hash table

Hash Table

[Wolfson97]

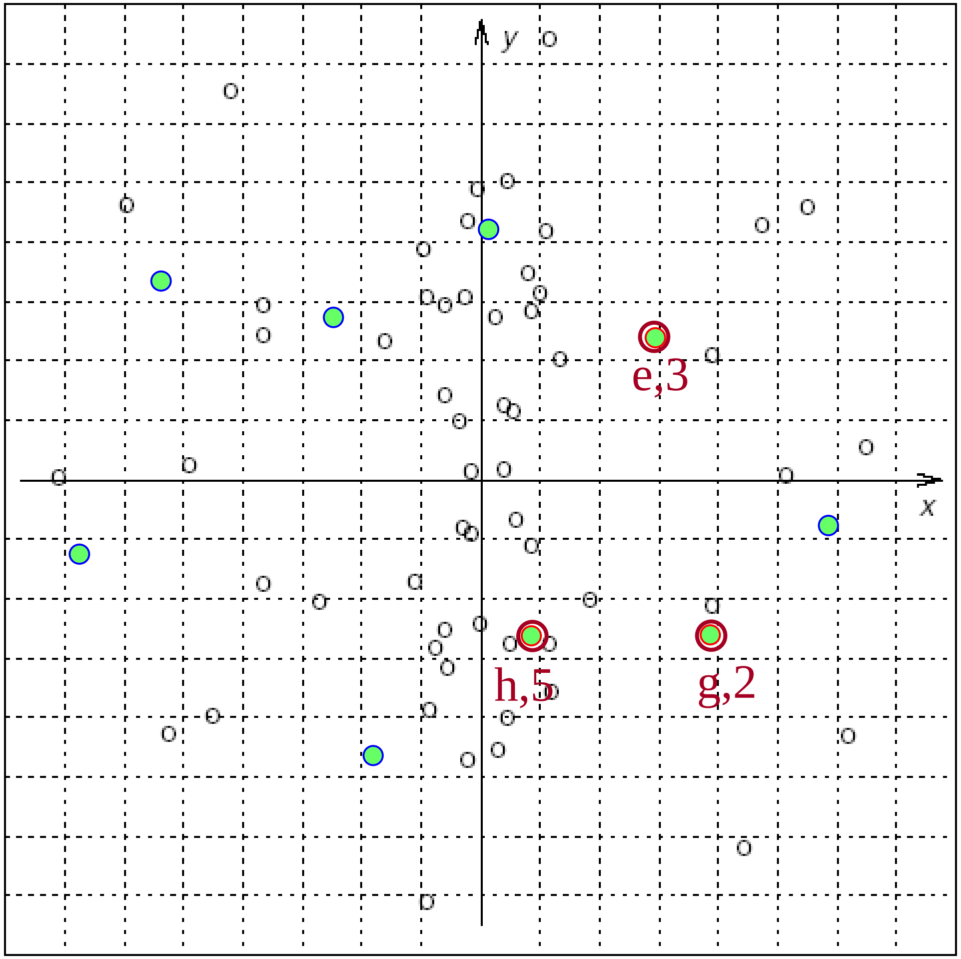
Geometric Hashing

Query processing



Point Set Query

Rotation & translation for one pair of points



[Wolfson97]

Geometric Hashing

Preprocessing complexity

- $O(n^4)$ for n points per binding site
  - $O(n^3)$ possible triples * O(n) transformations per triple

Query complexity

- O(m) * binsize for m points in query binding site
  - 1 triple * O(m) transformations per triple * binsize hash processing per transformation

[Wolfson97]

Outline

Introduction

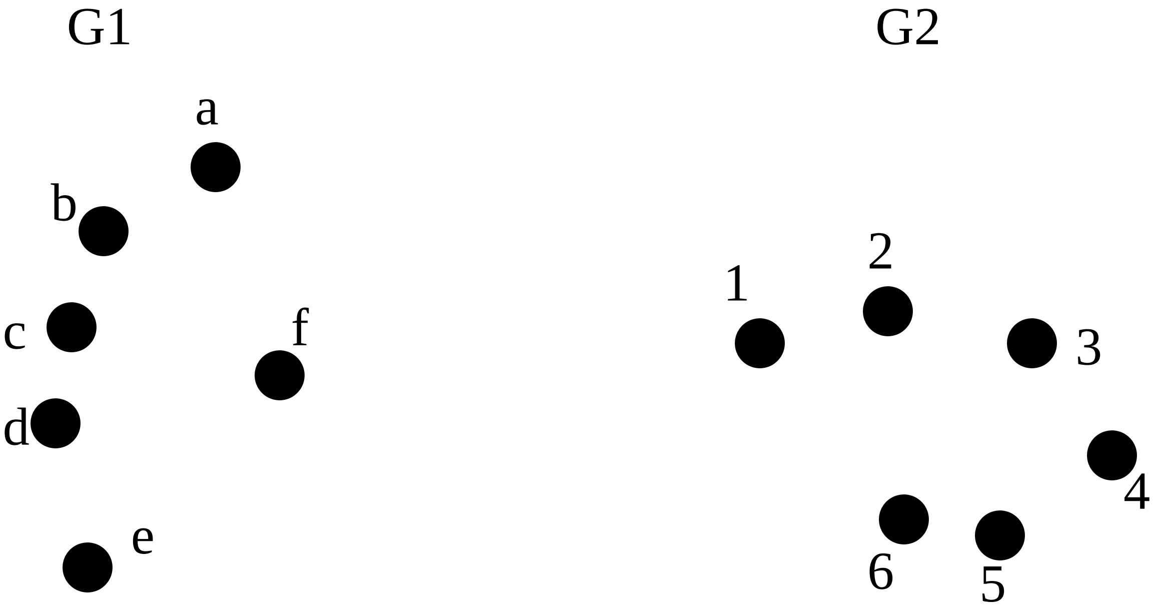
Point set matching

- Brute force search
- RANSAC
- Geometric hashing
- ➢ Association graphs
- Generalized Hough transform
- Iterative closest point

Methods used for RGB-D scanning

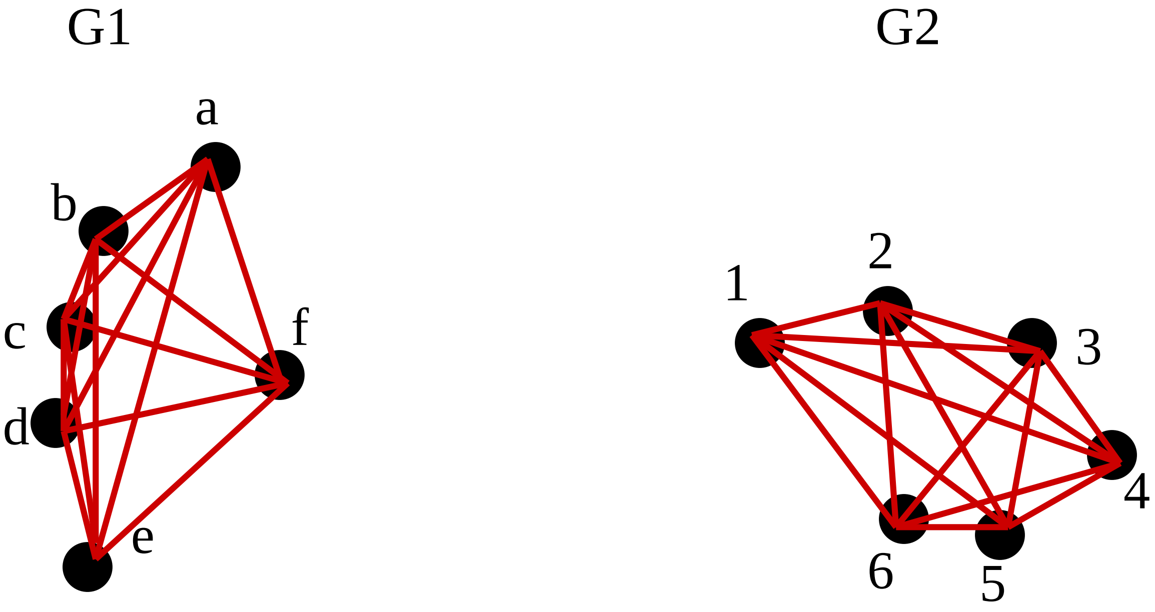
Discussion

Association Graphs



[Schmitt02, Brown82]

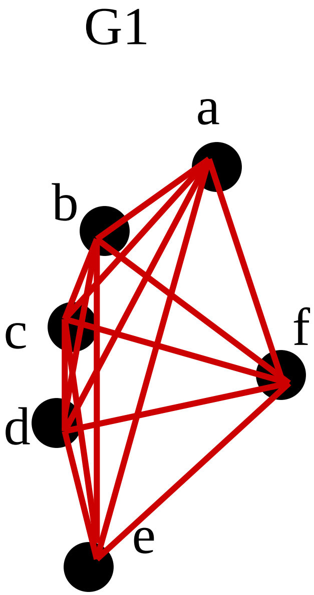
Association Graphs



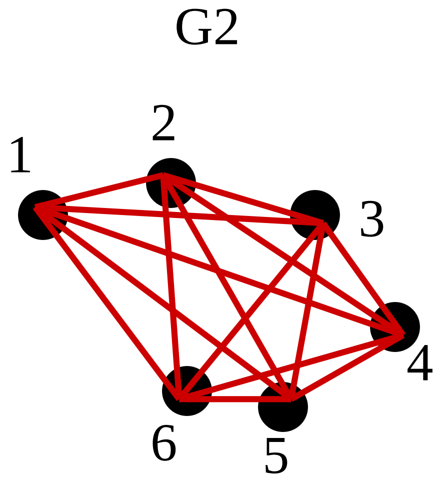
Represent both points sets as complete graphs (G1 and G2).
(edges connect all pairs of vertices within each point set)

[Schmitt02, Brown82]

Association Graphs



Association Graph

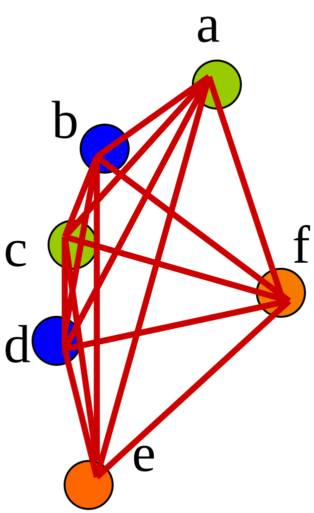


Create vertices in the association graph for all compatible pairs of vertices in the original graphs. This can lead to a large number of vertices.

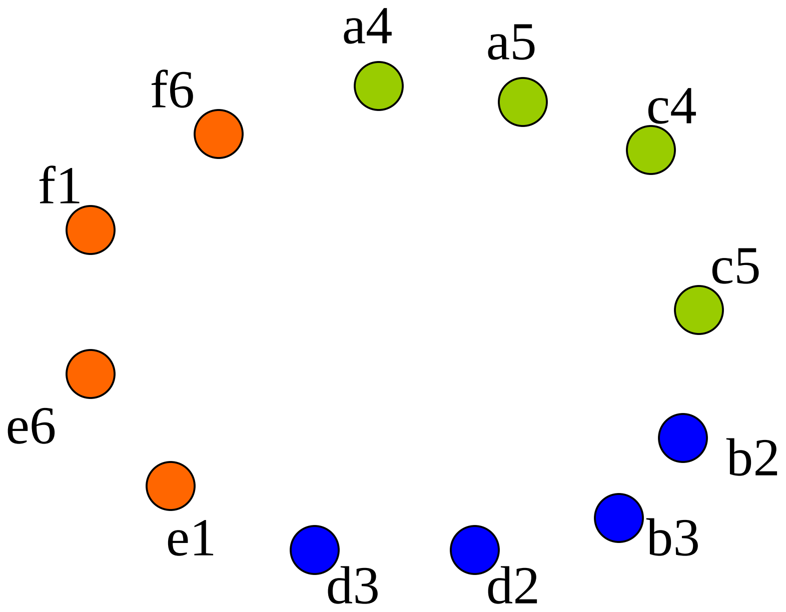
[Schmitt02, Brown82]

Association Graphs

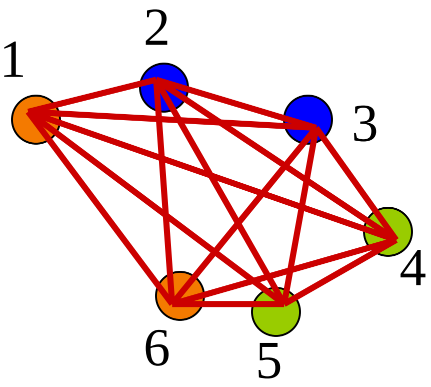
G1



Association Graph



G2

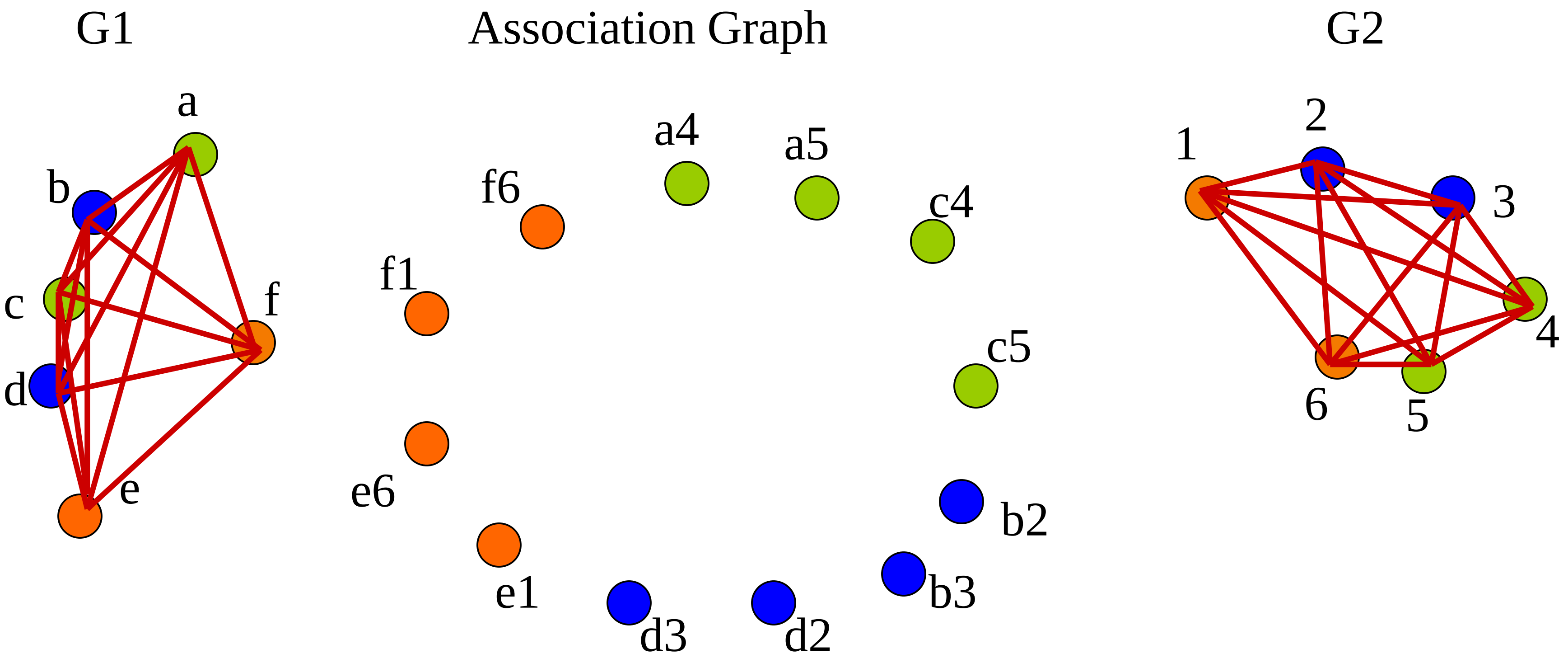


- •Depth
- •Propensity
- •Conservation
- •Charge
- •Hydrophobicity
- •Secondary structure type
- •Destabilization

Create vertices in the association graph for all compatible pairs of vertices in the original graphs. Compatibility could refer to chemical properties.

[Schmitt02, Brown82]

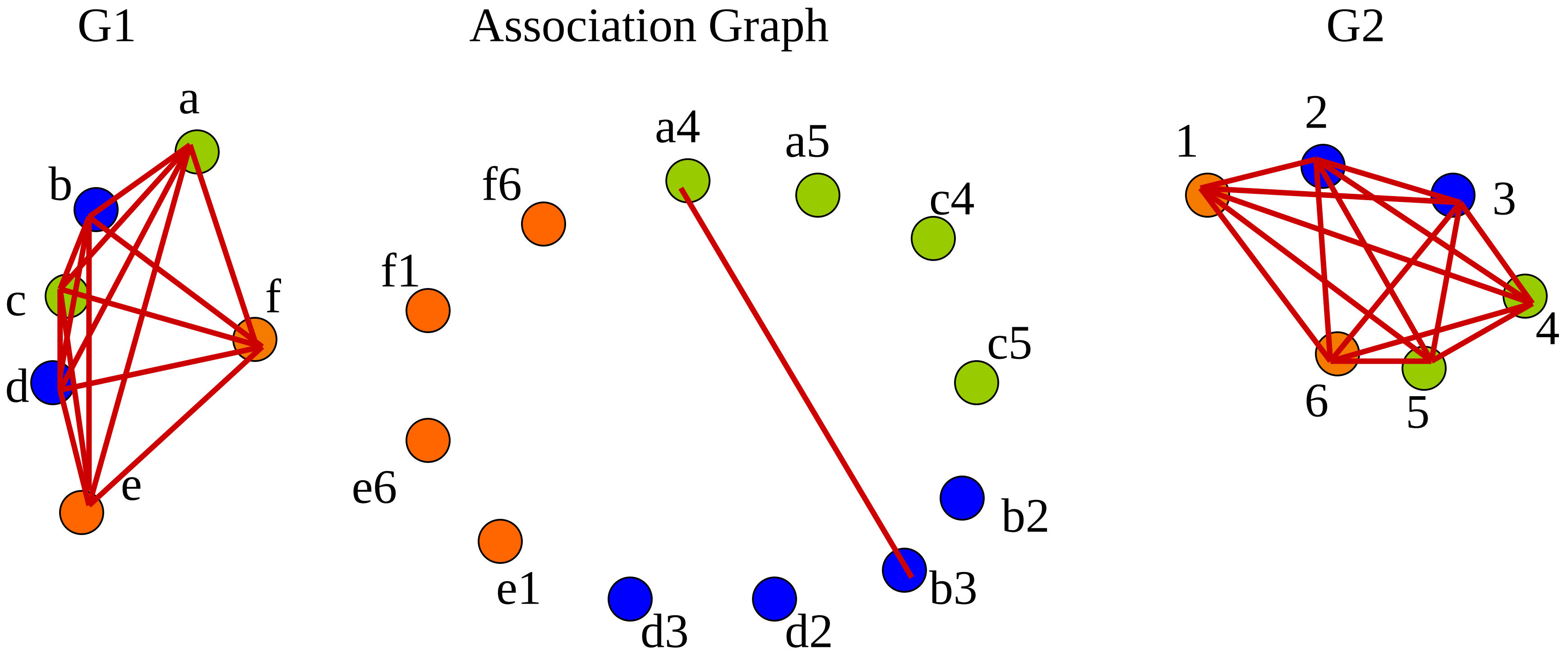
Association Graphs



Create edges between (uv) and (wx) if the edges between (u) and (w) as well as between (v) and (x) match.

[Schmitt02, Brown82]

Association Graphs

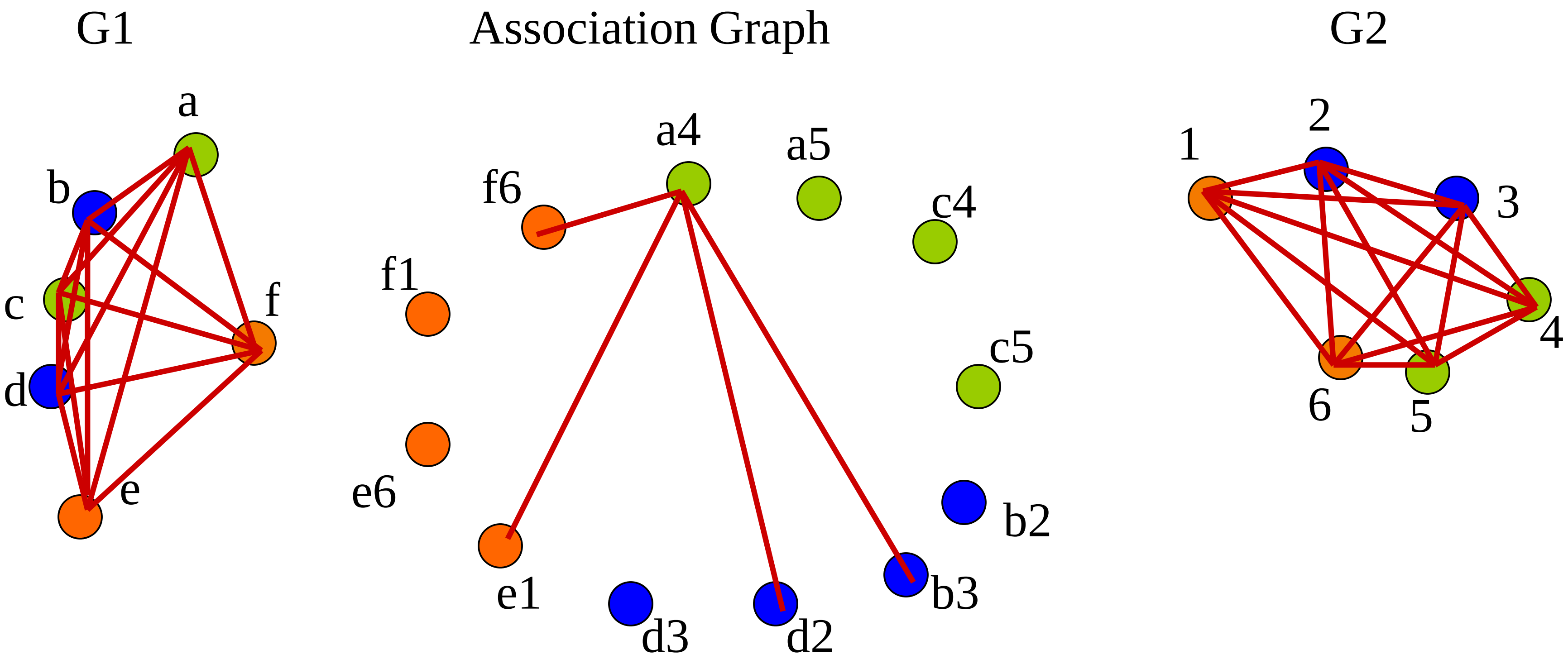


Create edges between (uv) and (wx) if the edges between (u) and (w) as well as between (v) and (x) match.

For this example, edge length is the only consideration

[Schmitt02, Brown82]

Association Graphs

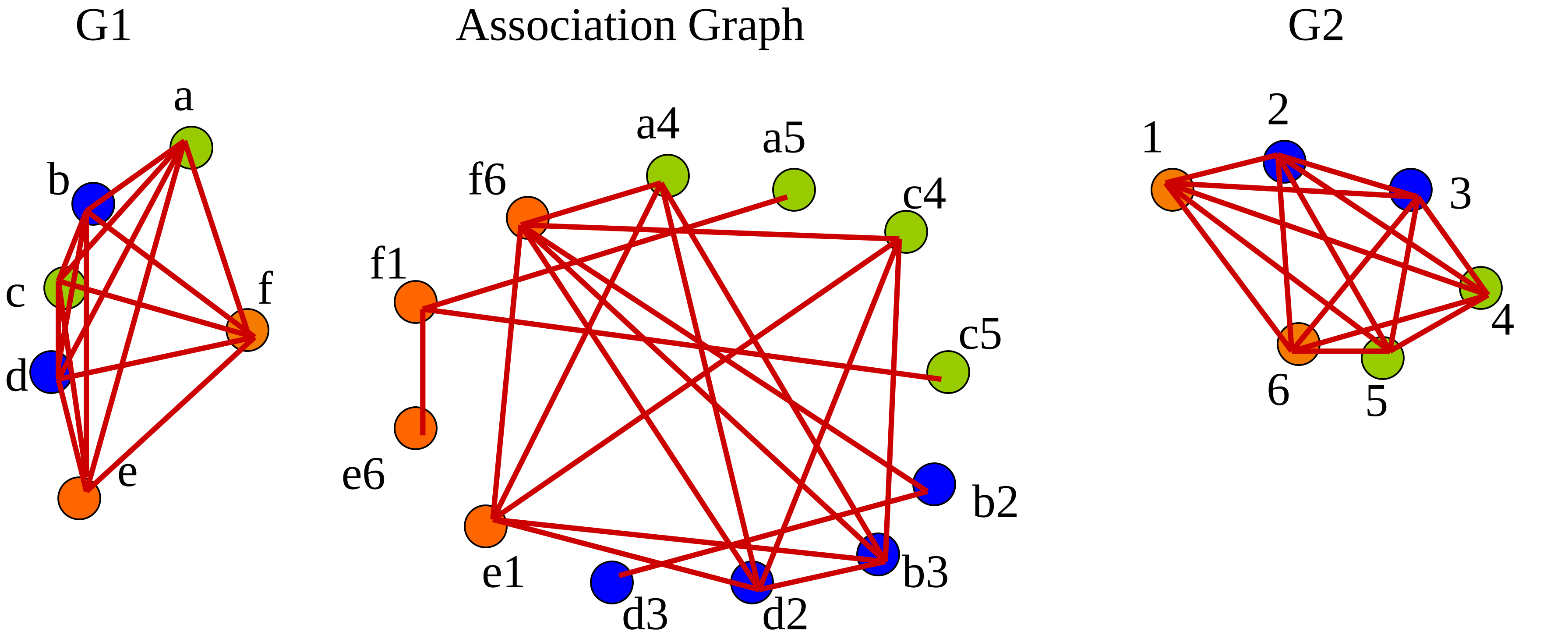


Create edges between (uv) and (wx) if the edges between (u) and (w) as well as between (v) and (x) match.

For this example, edge length is the only consideration

[Schmitt02, Brown82]

Association Graphs


G1

Association Graph

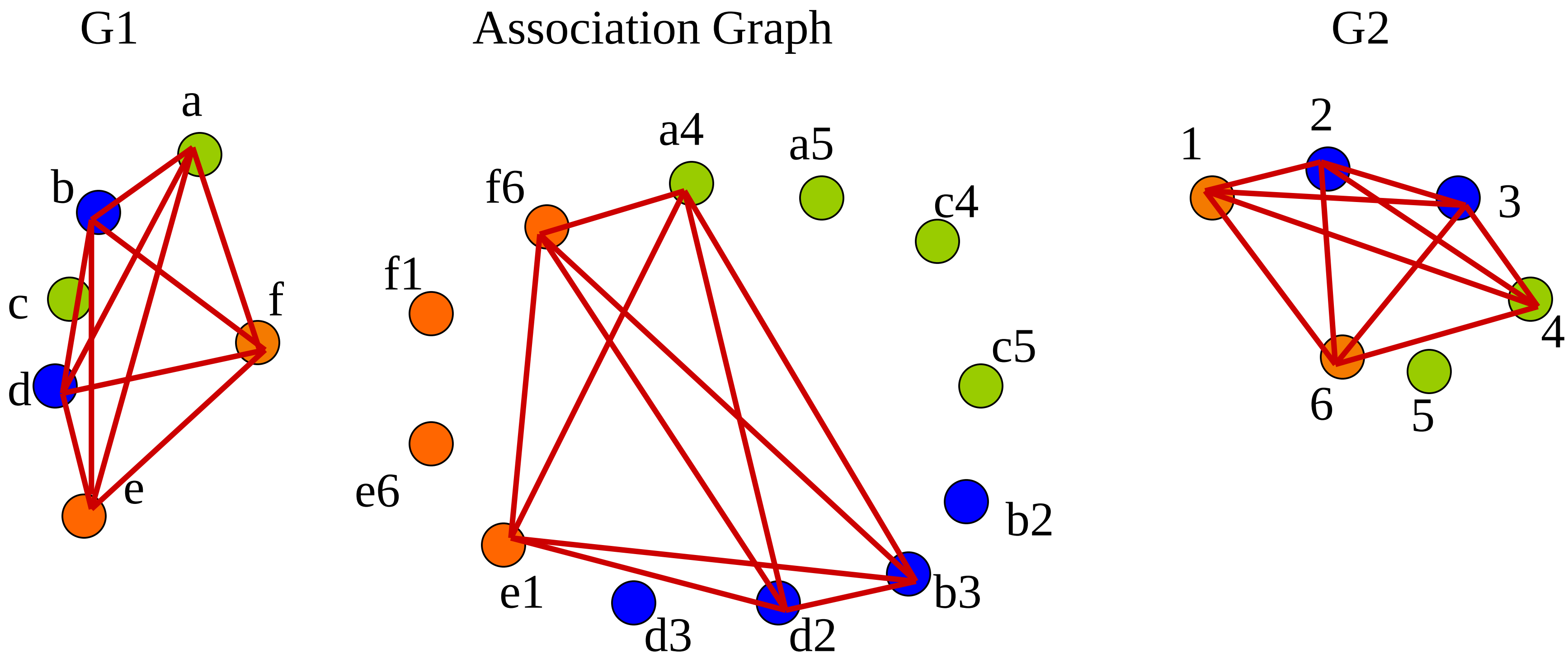
G2

Create edges between (uv) and (wx) if the edges between (u) and (w) as well as between (v) and (x) match.

For this example, edge length is the only consideration

[Schmitt02, Brown82]

Association Graphs



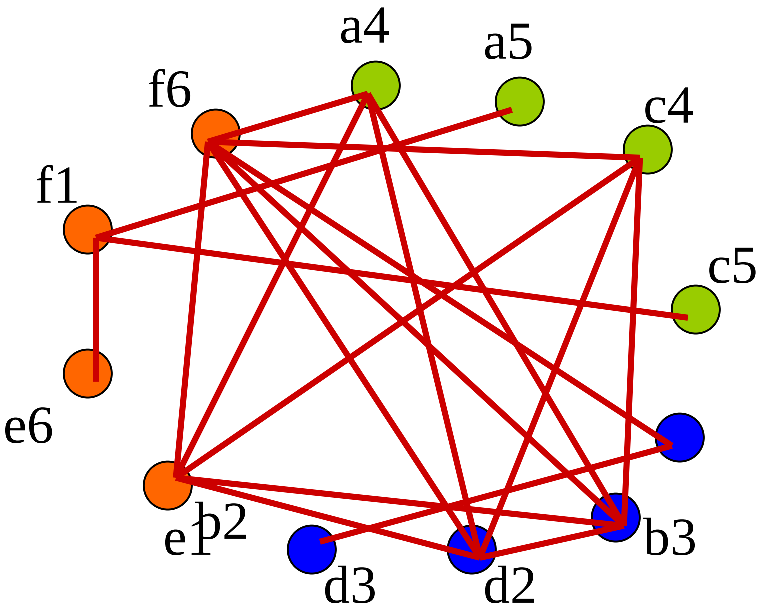
Finding correspondences: The the largest set of corresponding nodes in the same configuration is the maximal clique in the association graph

[Schmitt02, Brown82]

Association Graphs

Computational complexity:

- $O(2^n)$ for n points
- NP-complete
- Branch and bound algorithms



Association Graph

```
Find the Maximal Clique{
    return Cliques(empty, all nodes)
}

Cliques(X, Y){
    if (no node in Y-X is connected to all of X){
        return X;
    }else{
        y = node in Y connected to all of X;
        return Largest(Cliques(X union y, Y},
                              Cliques{X, Y-y});
    }
}
```

[Schmitt02, Brown82]

Outline

Introduction

Point set matching

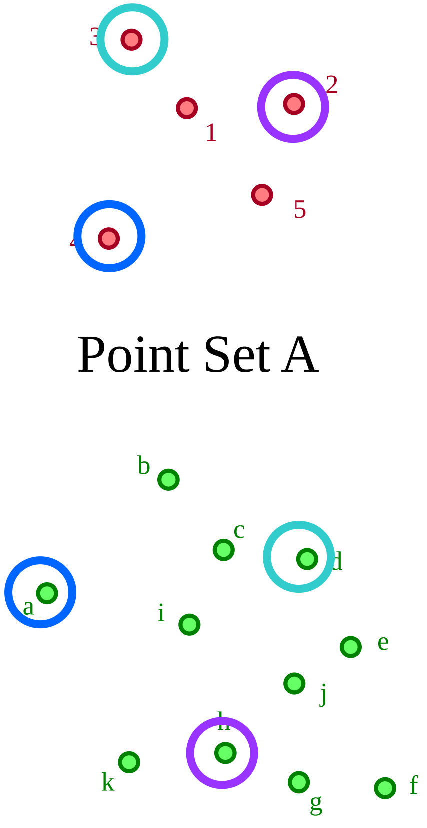
- Brute force search
- RANSAC
- Geometric hashing
- Association graphs
- ➢ Generalized Hough transform
- Iterative closest point

Methods used for RGB-D scanning

Discussion

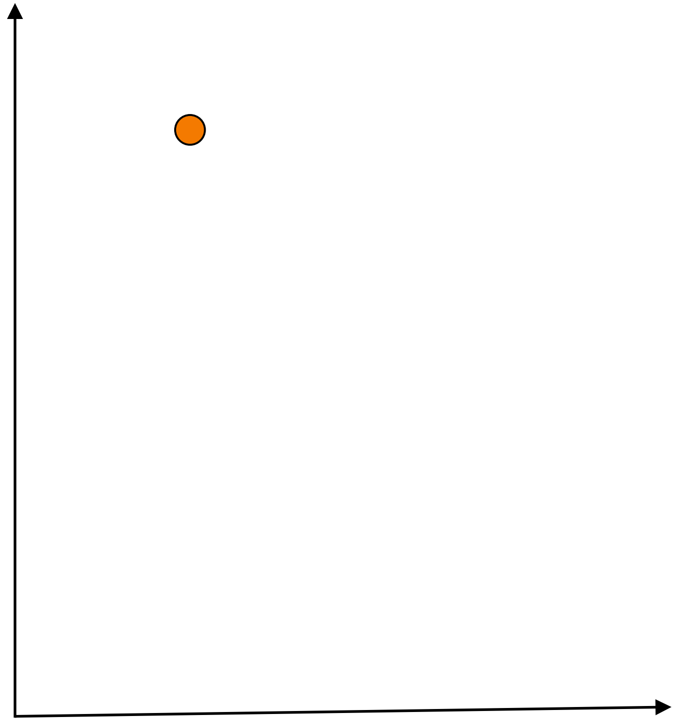
Generalized Hough Transform

Vote for transformations



Point Set A

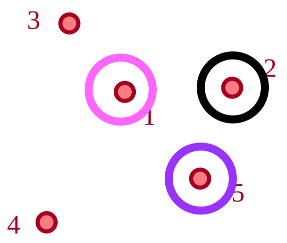
Point Set B



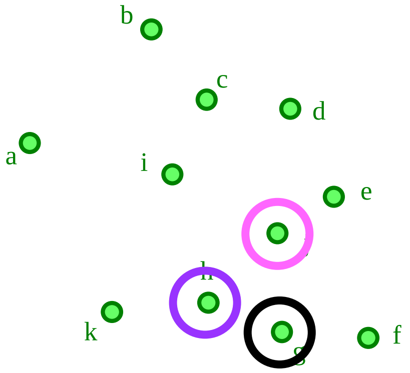
Hough Transformation Space

Generalized Hough Transform

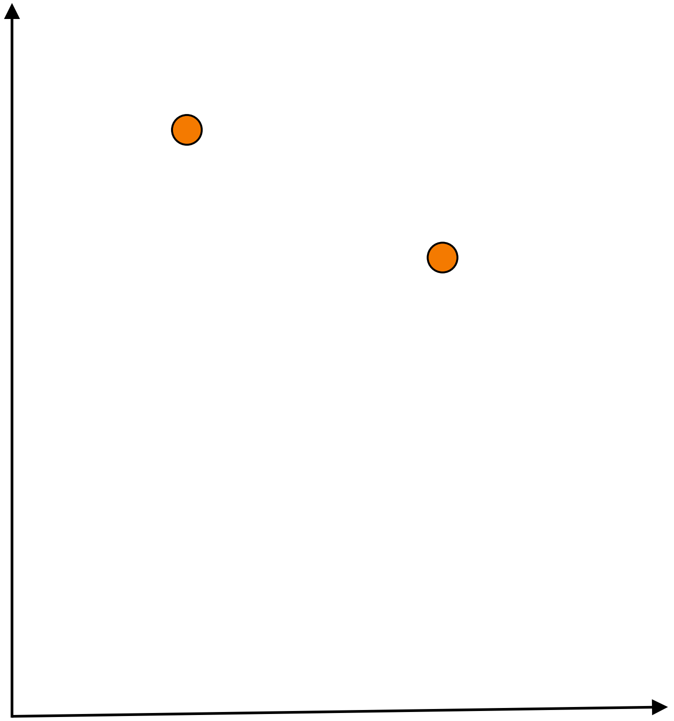
Vote for transformations



Point Set A



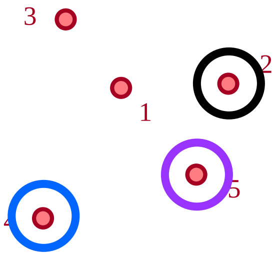
Point Set B



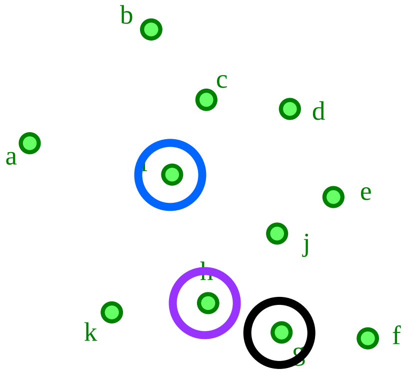
Hough Transformation Space

Generalized Hough Transform

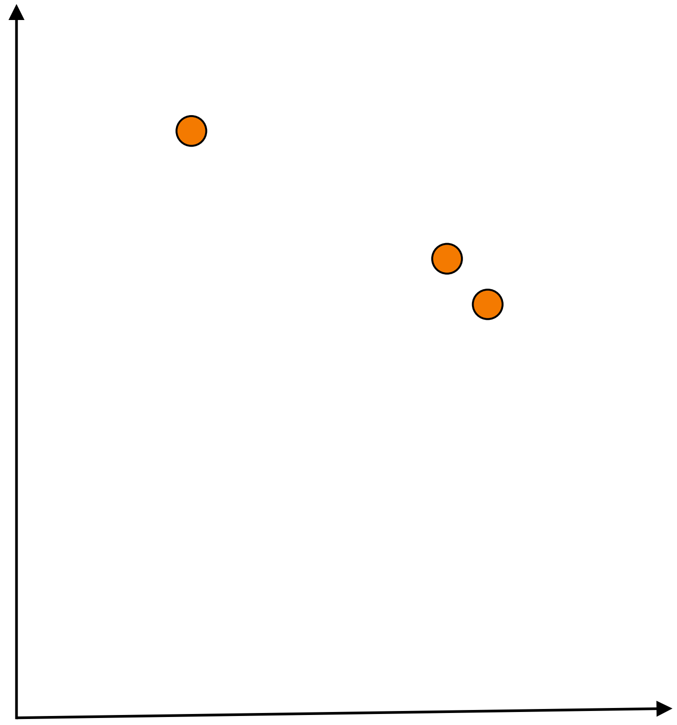
Vote for transformations



Point Set A



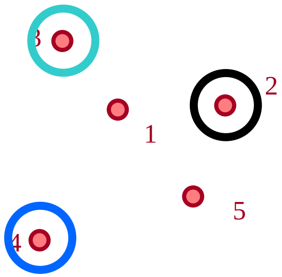
Point Set B



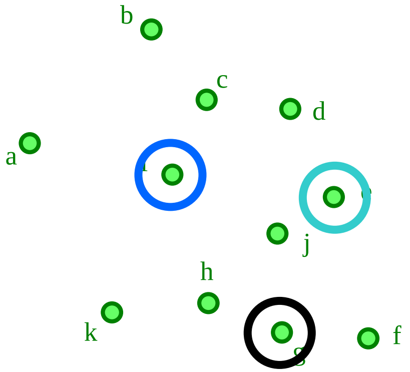
Hough Transformation Space

Generalized Hough Transform

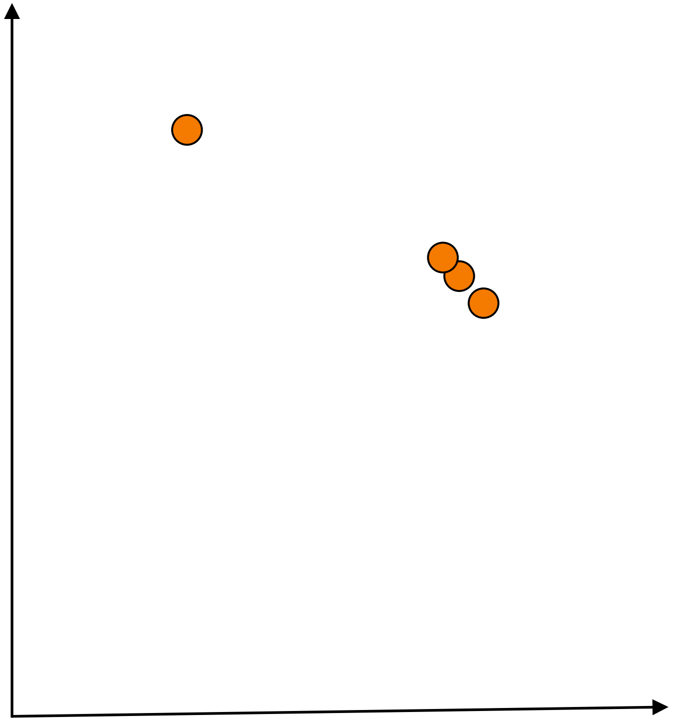
Vote for transformations



Point Set A



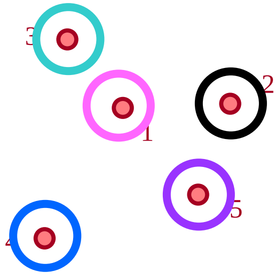
Point Set B



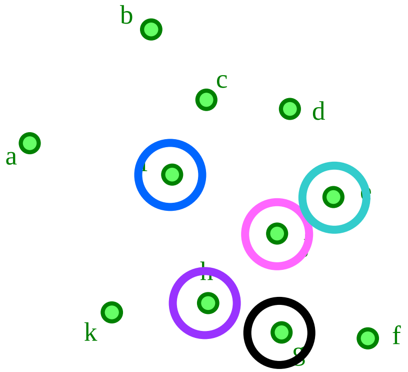
Hough Transformation Space

Generalized Hough Transform

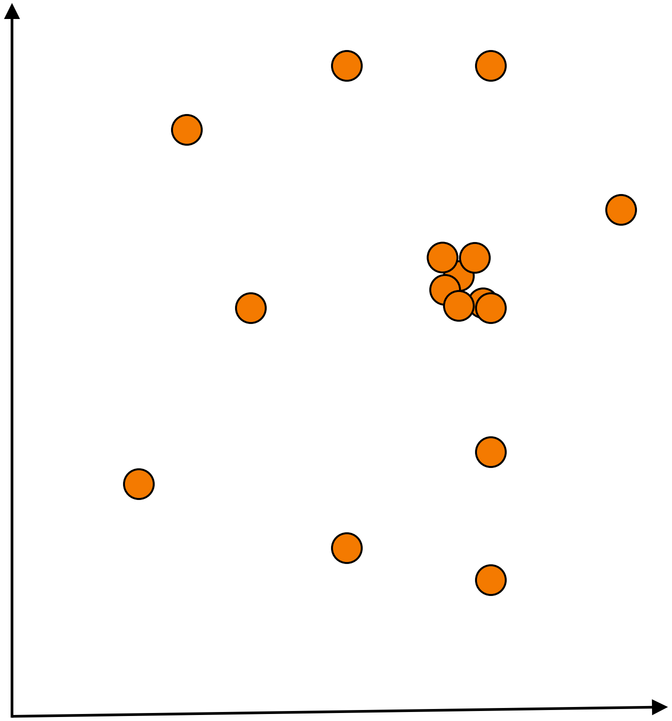
Vote for transformations



Point Set A



Point Set B



Hough Transformation Space

Generalized Hough Transform

Simple to implement

- Can use grid to represent transformation space

Expensive for high-dimensional transformations

- Storage and number of samples is exponential in dimensionality of transformation space
    - Translation (3D)
    - Rotation (3D)
    - Translation & rotation (6D)
    - Translation & rotation & scale (7D)

Outline

Introduction

Point set matching

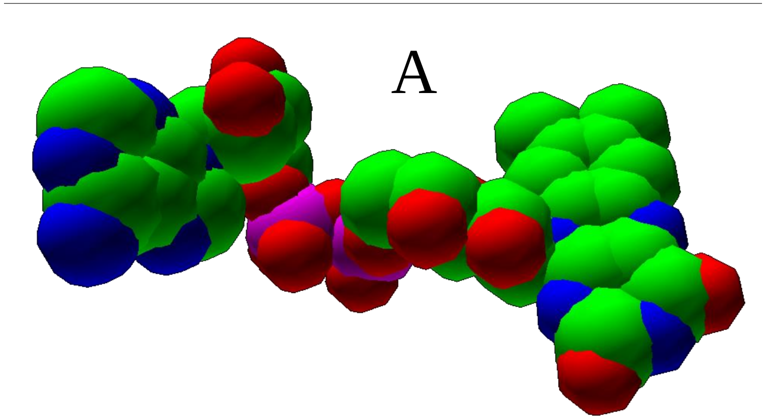
- Brute force
- RANSAC
- Geometric hashing
- Assocation graphs
- Generalized Hough transform
- ➢Iterative closest points

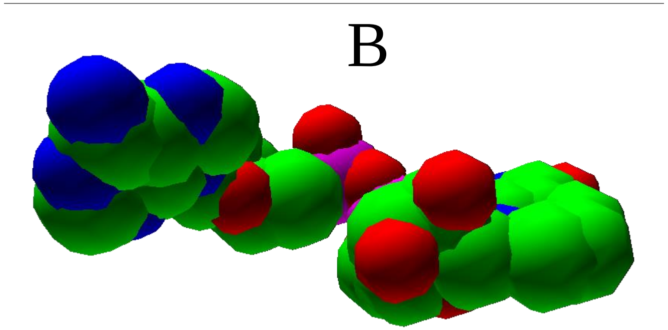
Methods used for RGB-D scanning

Discussion

Iterative Closest Points

Given two point sets

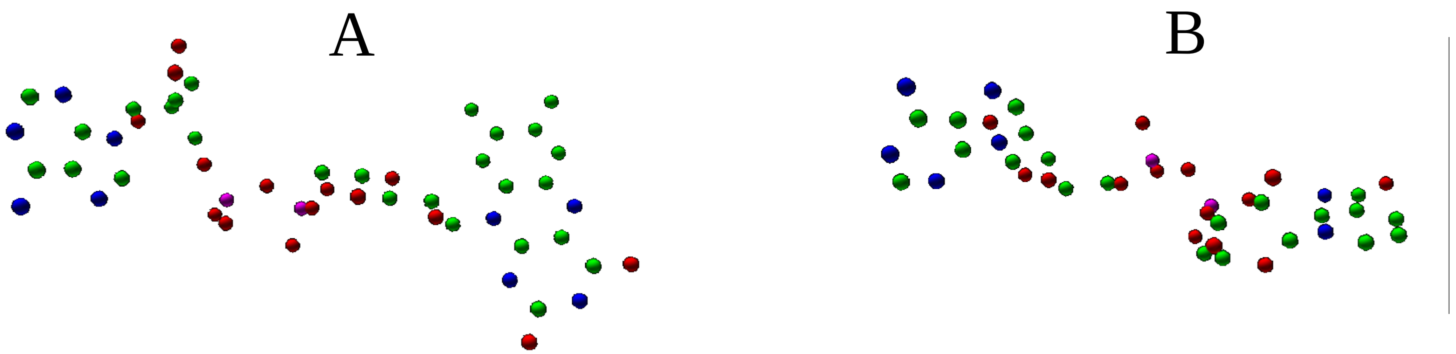




[Besl92]

Iterative Closest Points

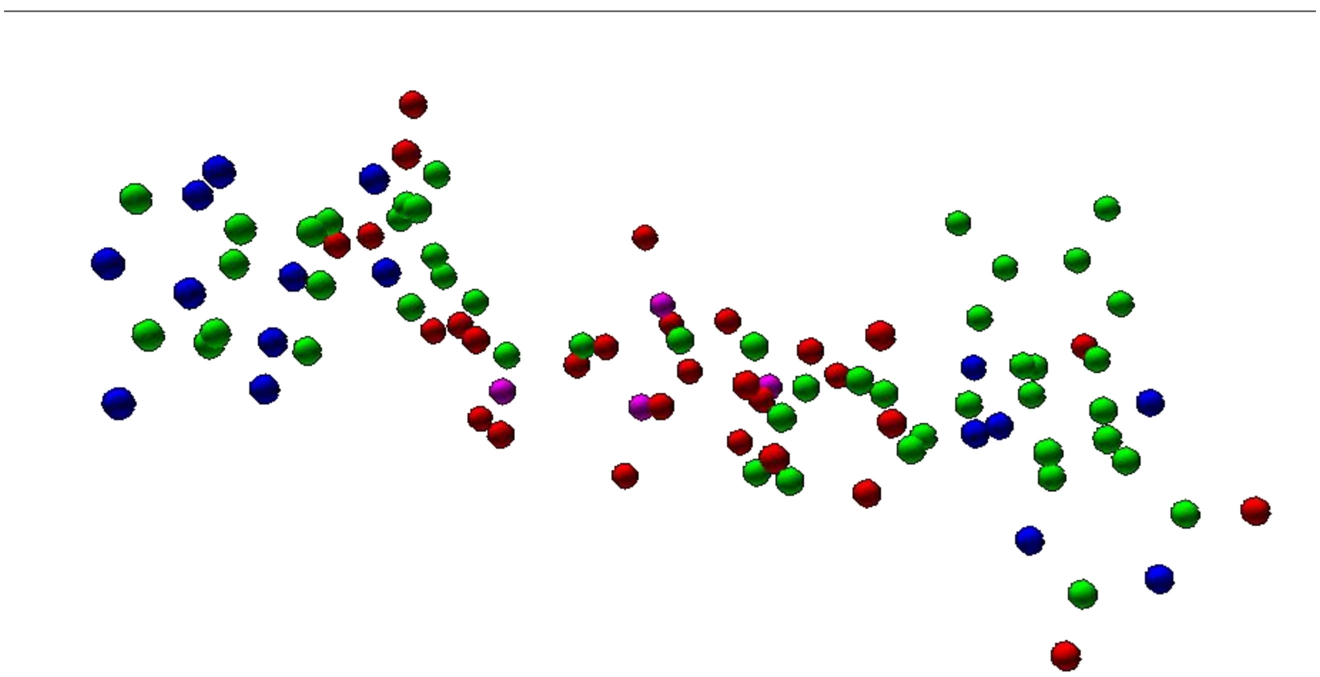
Given two point sets



[Besl92]

Iterative Closest Points

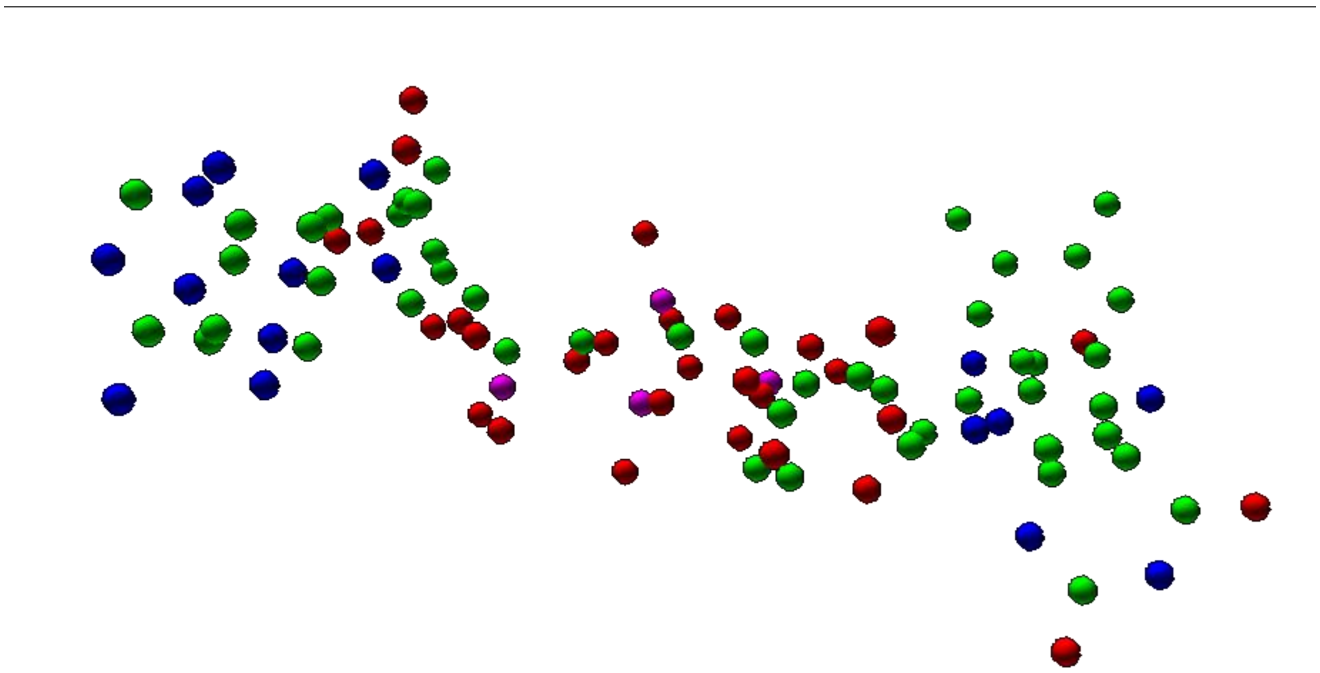
Given two point sets and an initial guess for the transformation that aligns them



[Besl92]

Iterative Closest Points

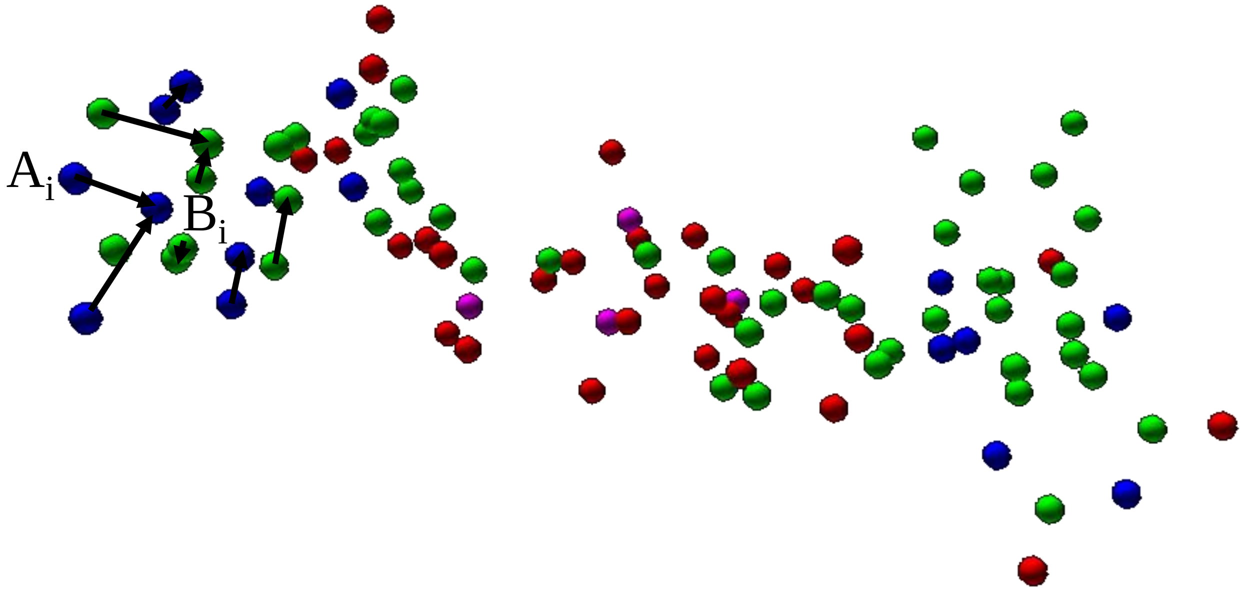
Assume closest points correspond



[Besl92]

Iterative Closest Points

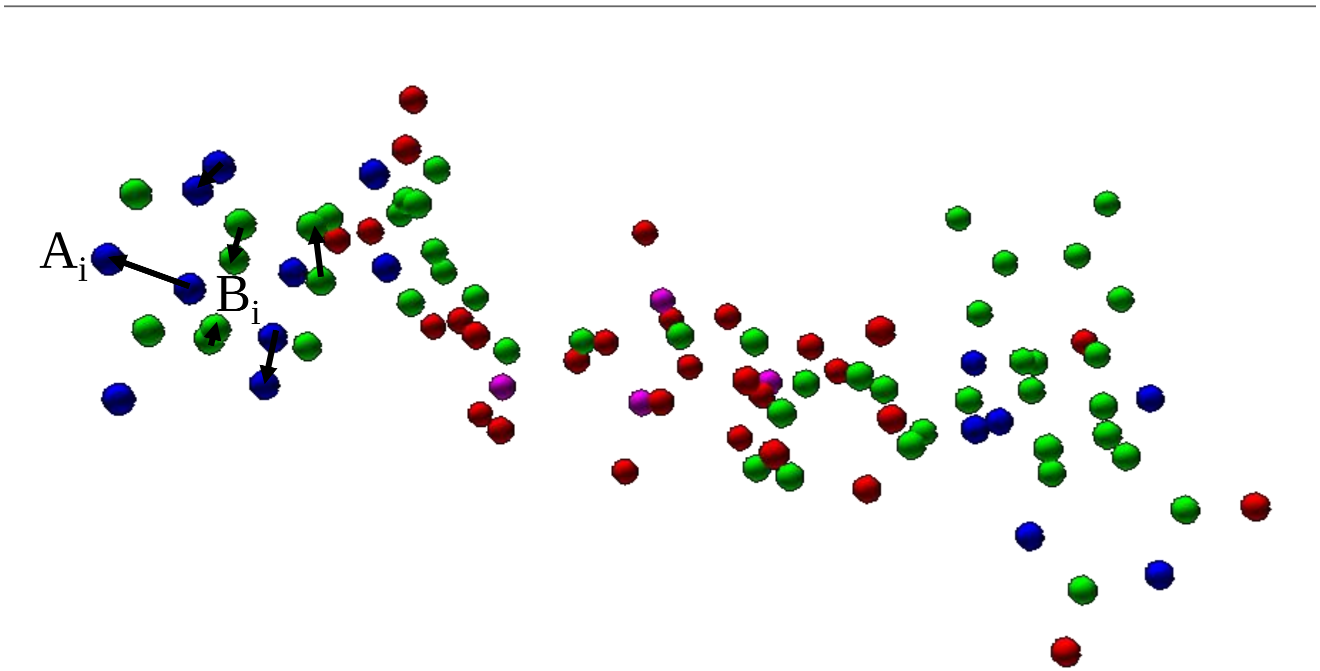
Assume closest points correspond: A→B



[Besl92]

Iterative Closest Points

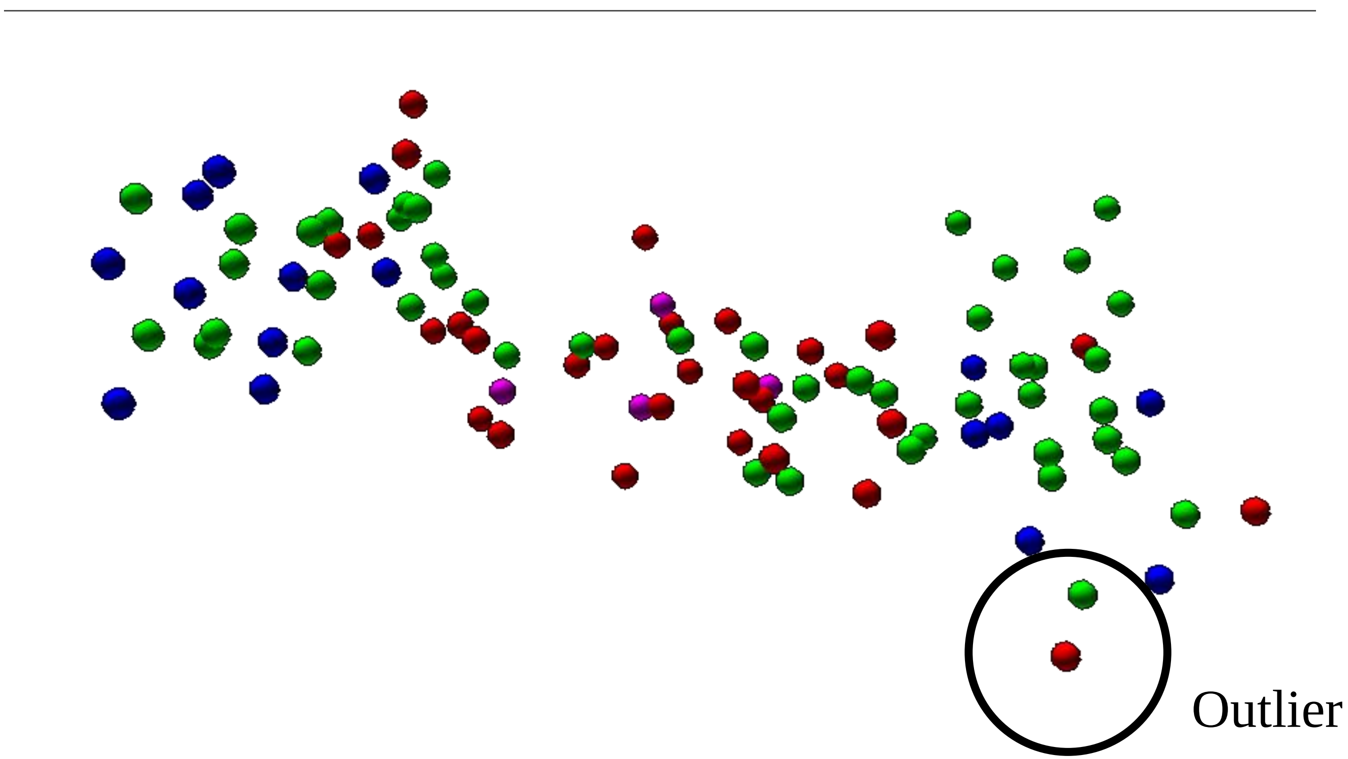
Assume closest points correspond: A→B and B→A



[Besl92]

Iterative Closest Points

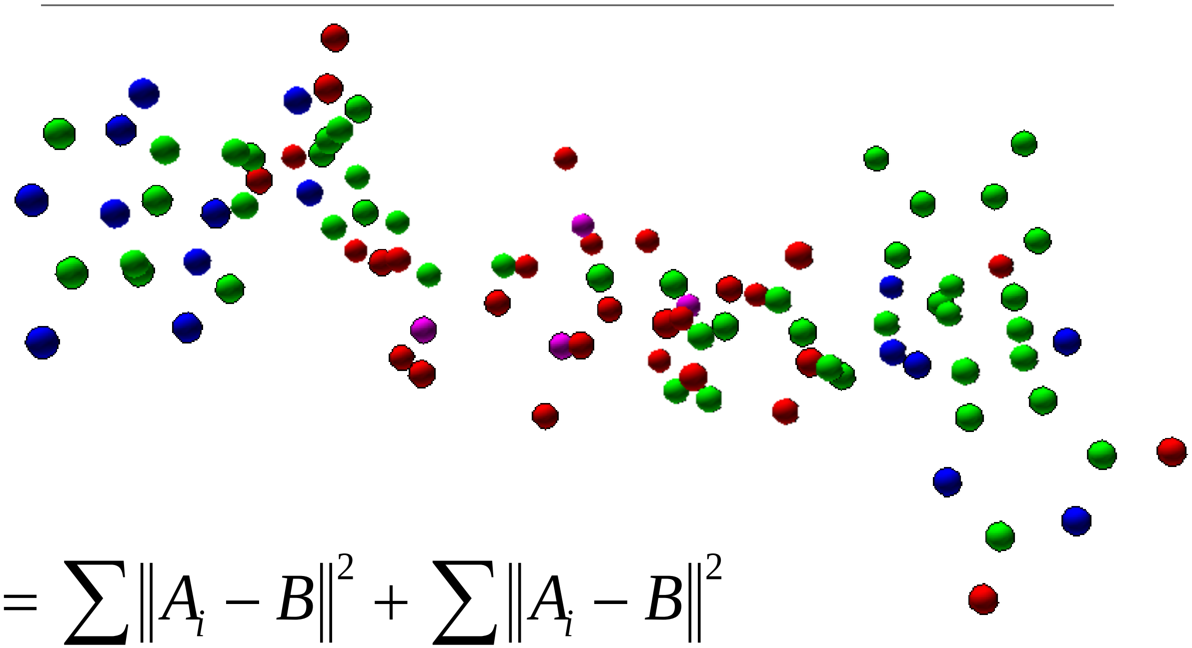
Rejecting outliers



[Besl92]

Iterative Closest Points

Find the transformation that optimally aligns proposed correspondences (superposition)

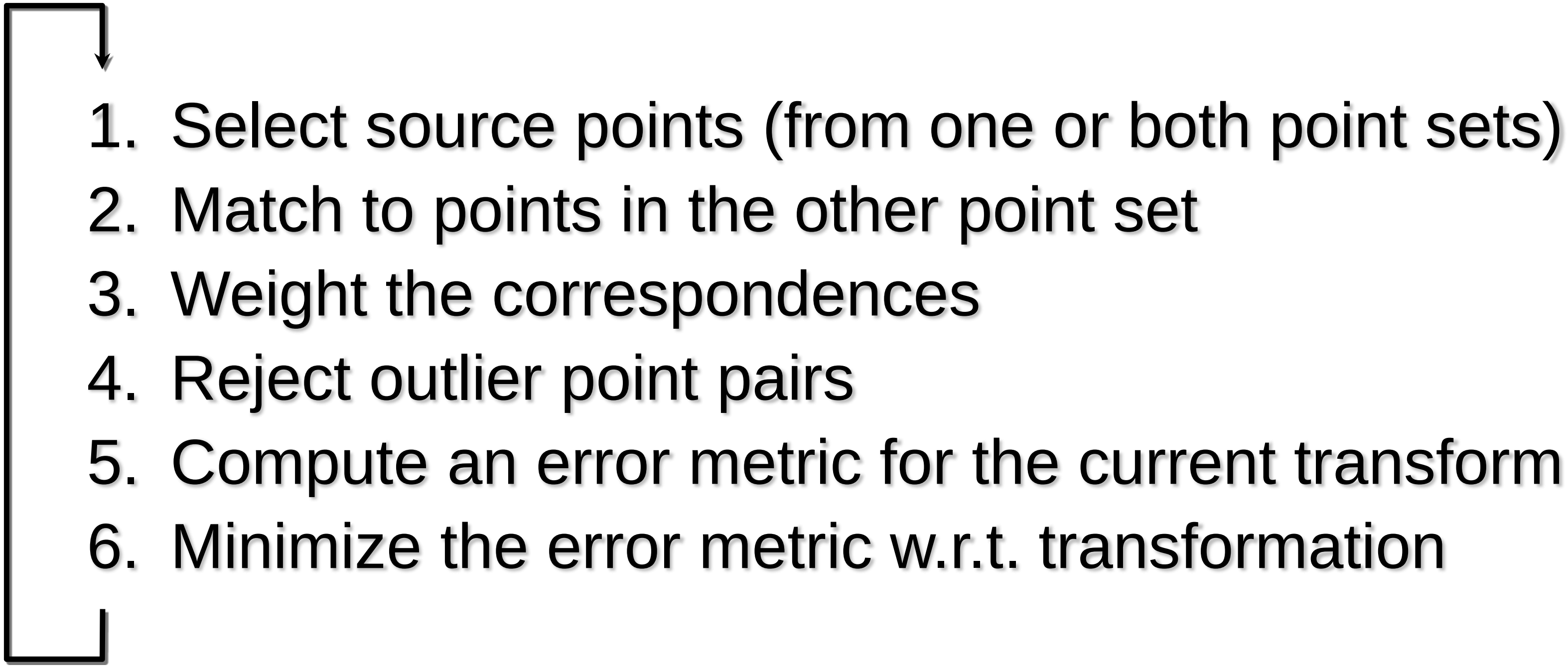


$$d(A,B) = \sum_{A_i \in A} \left\| A_i - B \right\|^2 + \sum_{B_i \in B} \left\| A_i - B \right\|^2$$

[Besl92]

Iterative Closest Points

Iterate until convergence



1. Select source points (from one or both point sets)
2. Match to points in the other point set
3. Weight the correspondences
4. Reject outlier point pairs
5. Compute an error metric for the current transform
6. Minimize the error metric w.r.t. transformation

Computational complexity

- O(k * nlogn) for n points per binding site and k iterations
  - k iterations * O(n) points * O(logn) to find closest point

Slide courtesy of Szymon Rusinkiewicz

Summary

Brute force

- Accurate, slow

RANSAC

- Approximate

Geometric hashing

- Fast query, after slow preprocessing
- Distance threshold implicit in hash bucket sizes

Association graphs

- Expensive for large point sets
- Distance threshold for “associations”

Generalized Hough transform

- Requires lots of space/samples for high dimensional transformations

Iterative closest points

- Fast, in practice
- Requires good initial guess