ACM SIGSPATIAL GIS ’09
Seattle, November 4-6, 2009

Vehicle Localization by Matching Triangulated Point Patterns

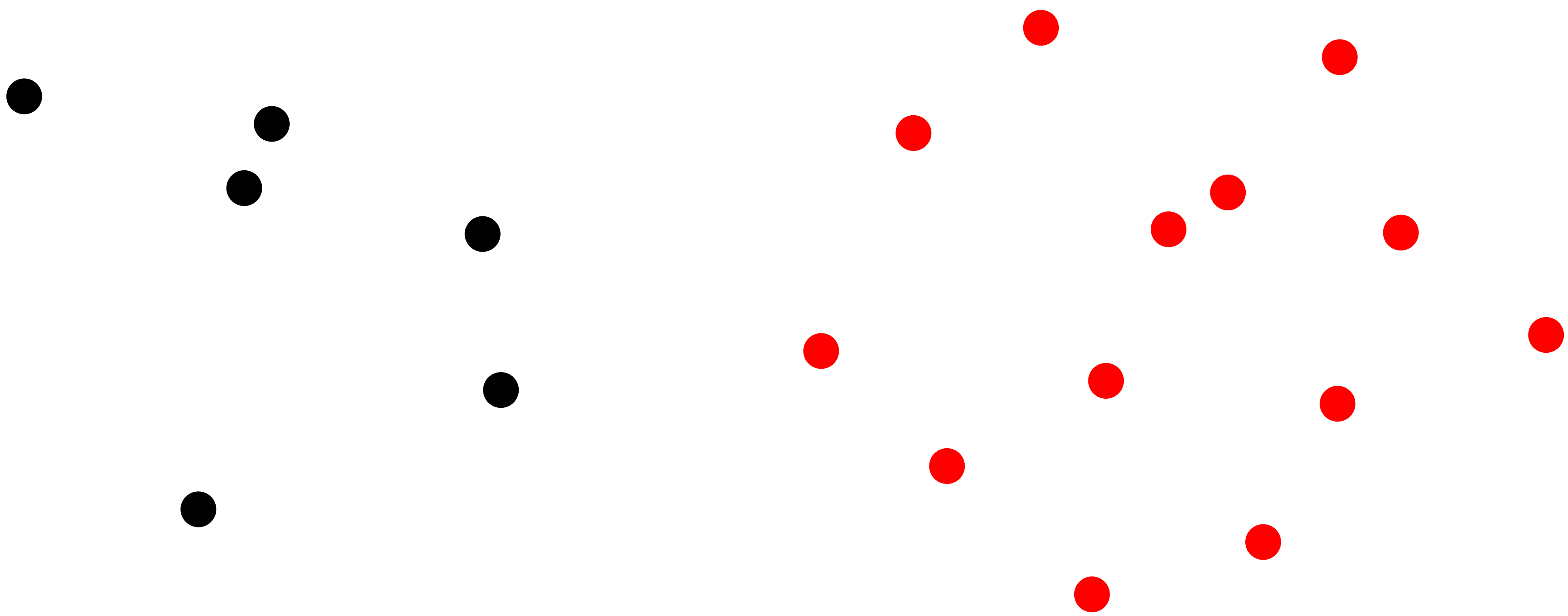
Jan-Henrik Haunert
Institut für Informatik
Universität Würzburg

Claus Brenner
Institut für Kartographie
und Geoinformatik
Universität Hannover

Introduction

Point pattern matching

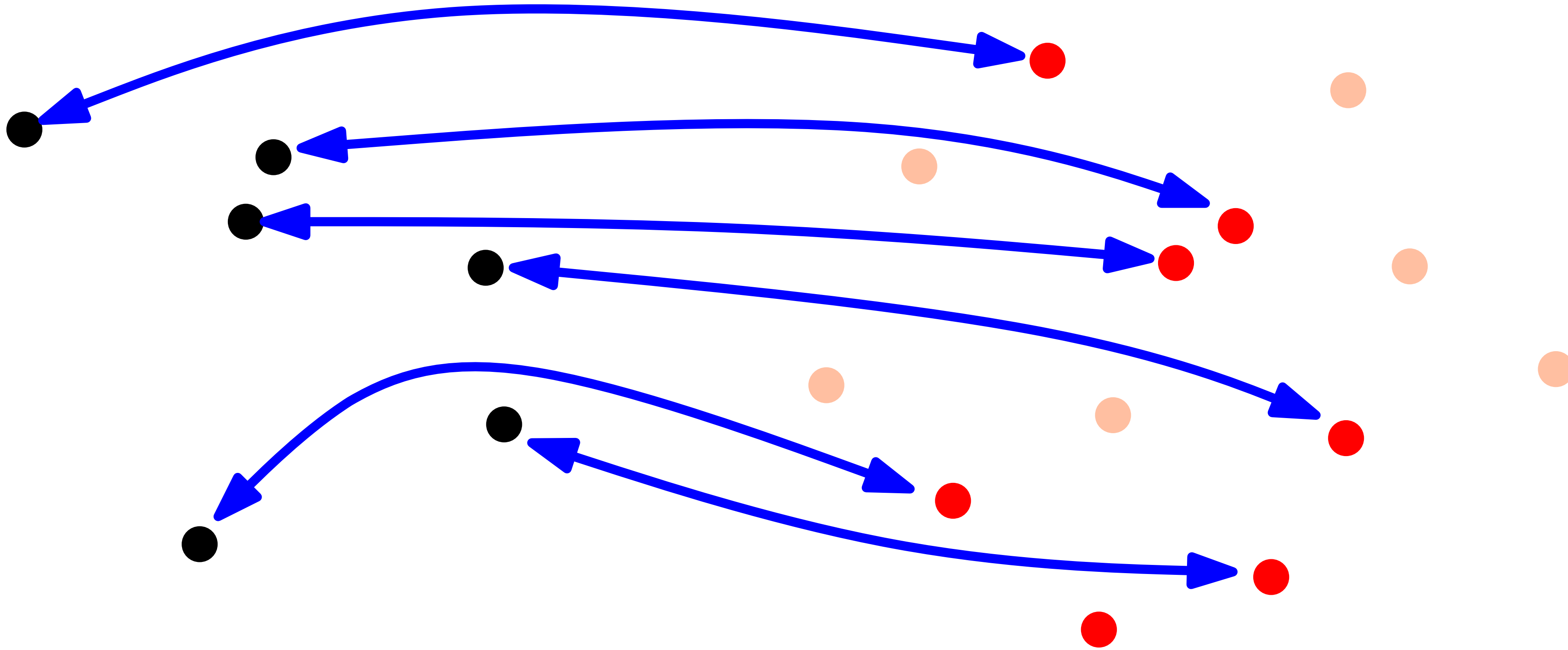
- given two point sets
- find corresponding points based on geometric configuration



Introduction

Point pattern matching

- given two point sets
- find corresponding points based on geometric configuration



Introduction

Applications of point pattern matching

- fingerprint verification

Introduction

Applications of point pattern matching

- orientation of star cameras

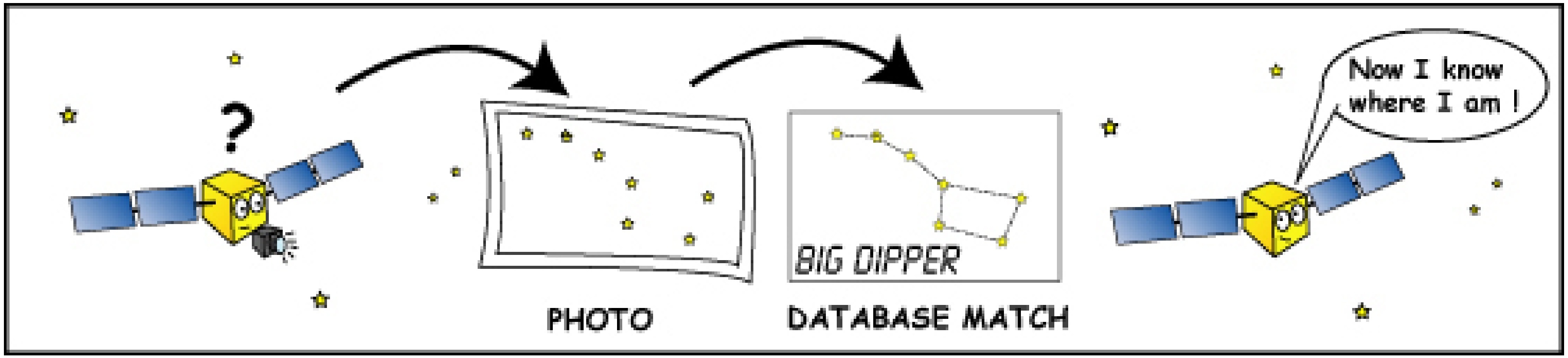


image source: NASA

Introduction

Applications of point pattern matching

- **here:** vehicle positioning

?

Introduction

Applications of point pattern matching

- **here:** vehicle positioning
  - GPS is not always/everywhere available
  - positioning a vehicle with only one system (GPS) is risky if it drives autonomously

?

Introduction

Applications of point pattern matching

- **here:** vehicle positioning

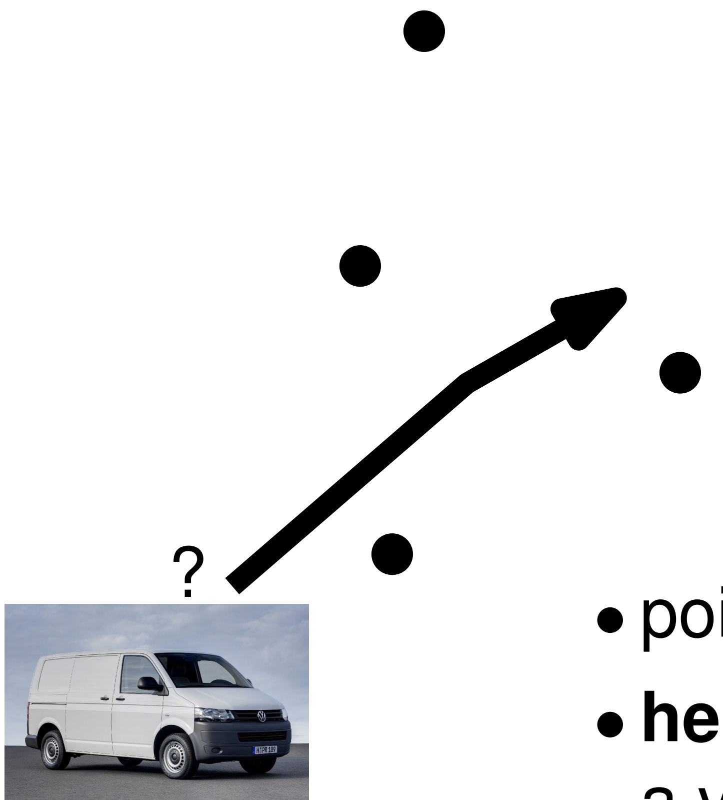
?

- points may represent any kind of landmarks
- **here:** poles (e.g., of traffic signs) observed with a vehicle-mounted laser scanner

Introduction

Applications of point pattern matching

- **here:** vehicle positioning

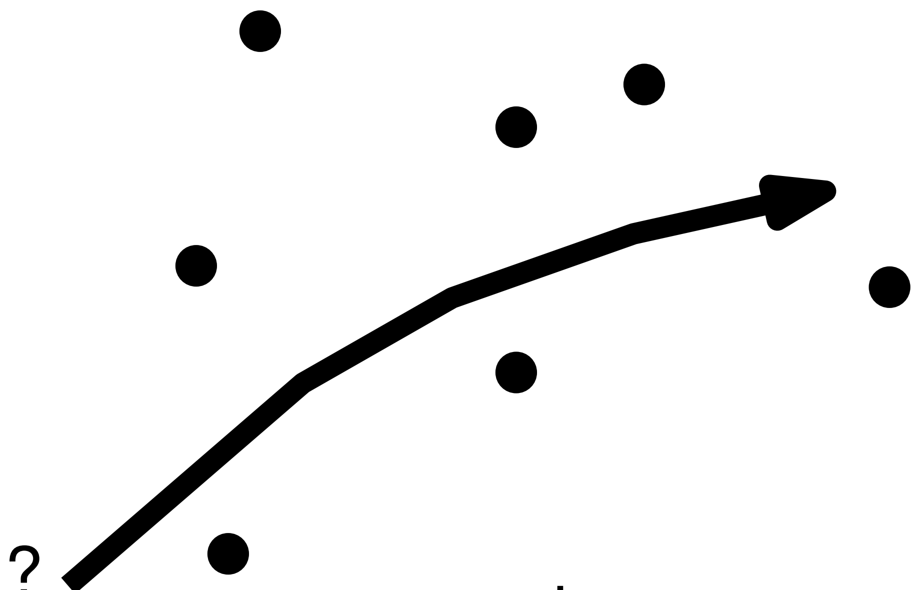


- points may represent any kind of landmarks
- **here:** poles (e.g., of traffic signs) observed with a vehicle-mounted laser scanner

Introduction

Applications of point pattern matching

- **here:** vehicle positioning

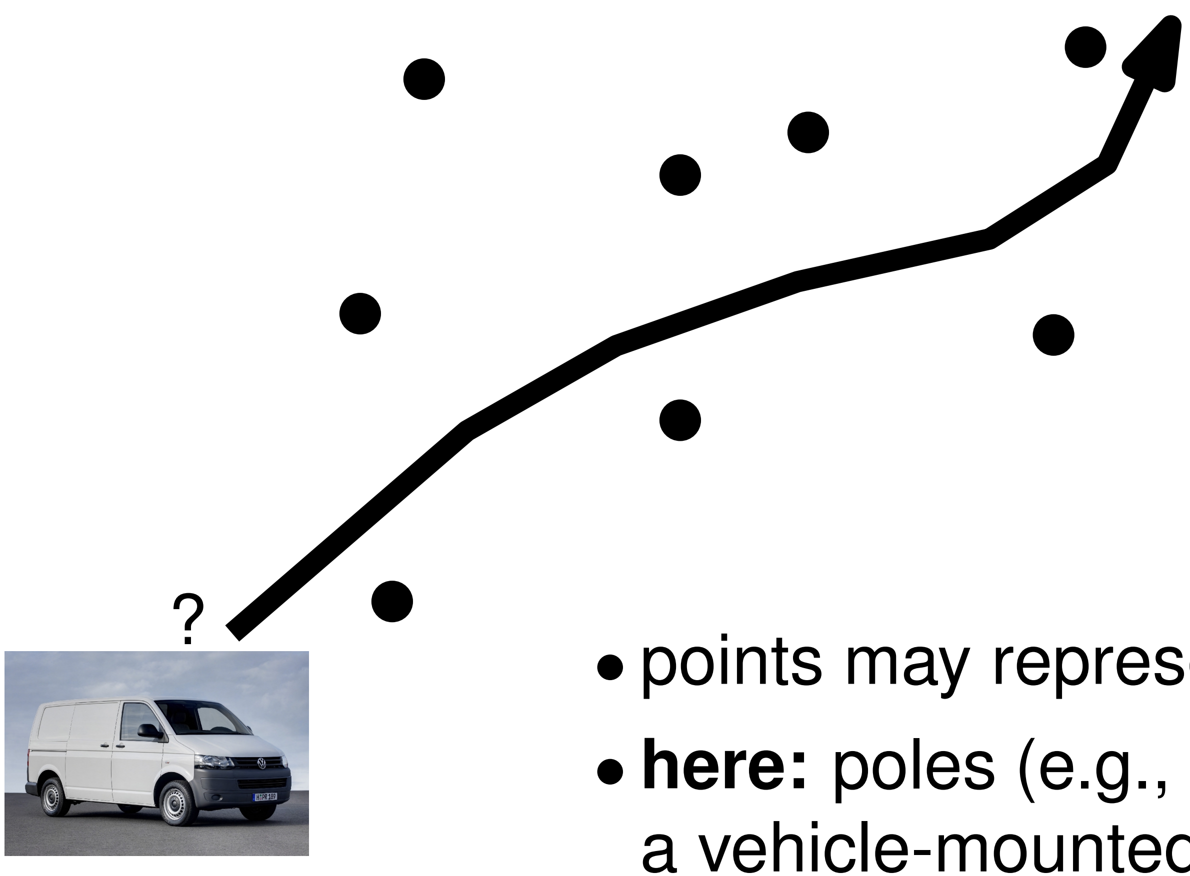


- points may represent any kind of landmarks
- **here:** poles (e.g., of traffic signs) observed with a vehicle-mounted laser scanner

Introduction

Applications of point pattern matching

- **here:** vehicle positioning

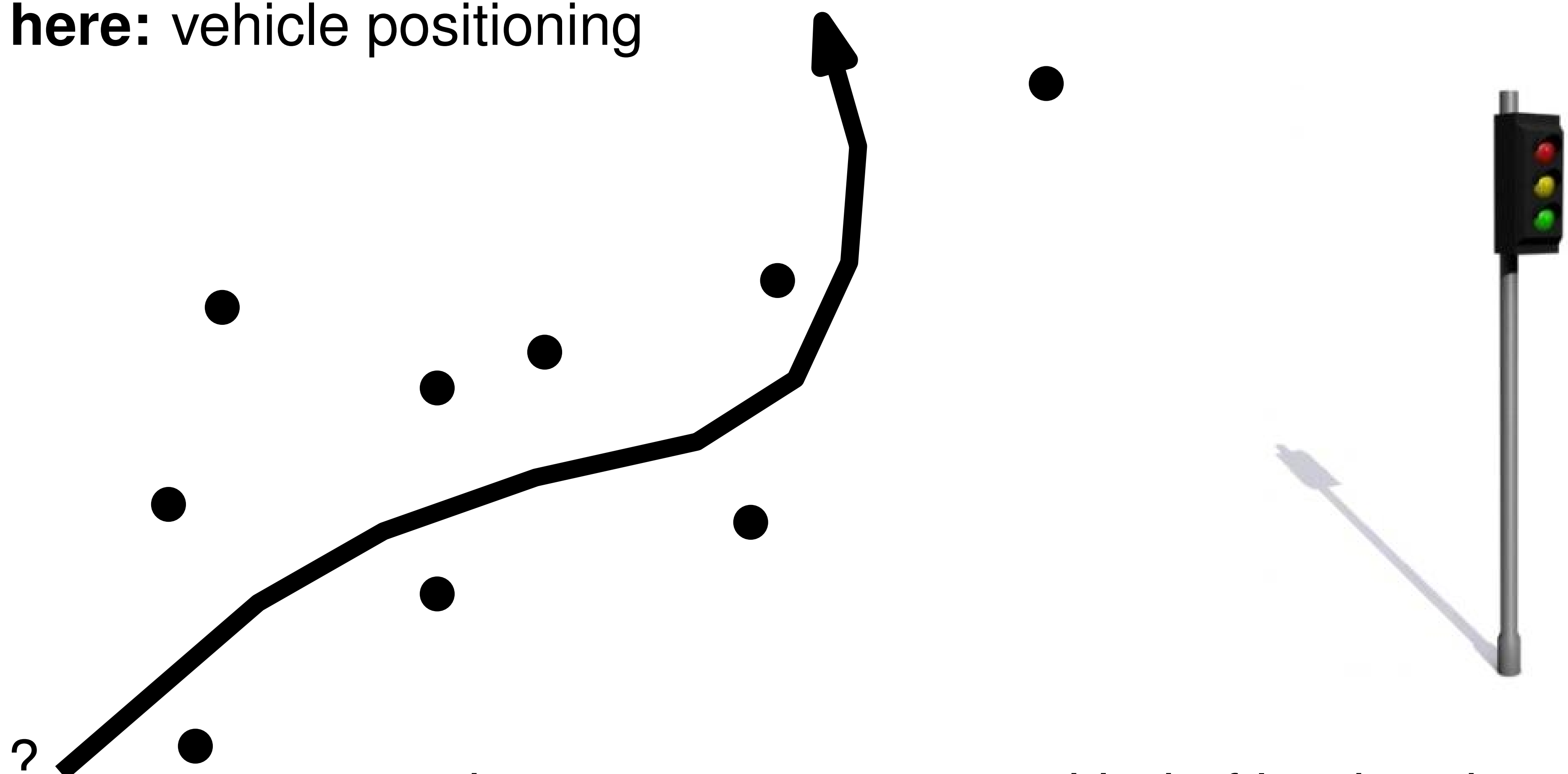


- points may represent any kind of landmarks
- **here:** poles (e.g., of traffic signs) observed with a vehicle-mounted laser scanner

Introduction

Applications of point pattern matching

- **here:** vehicle positioning

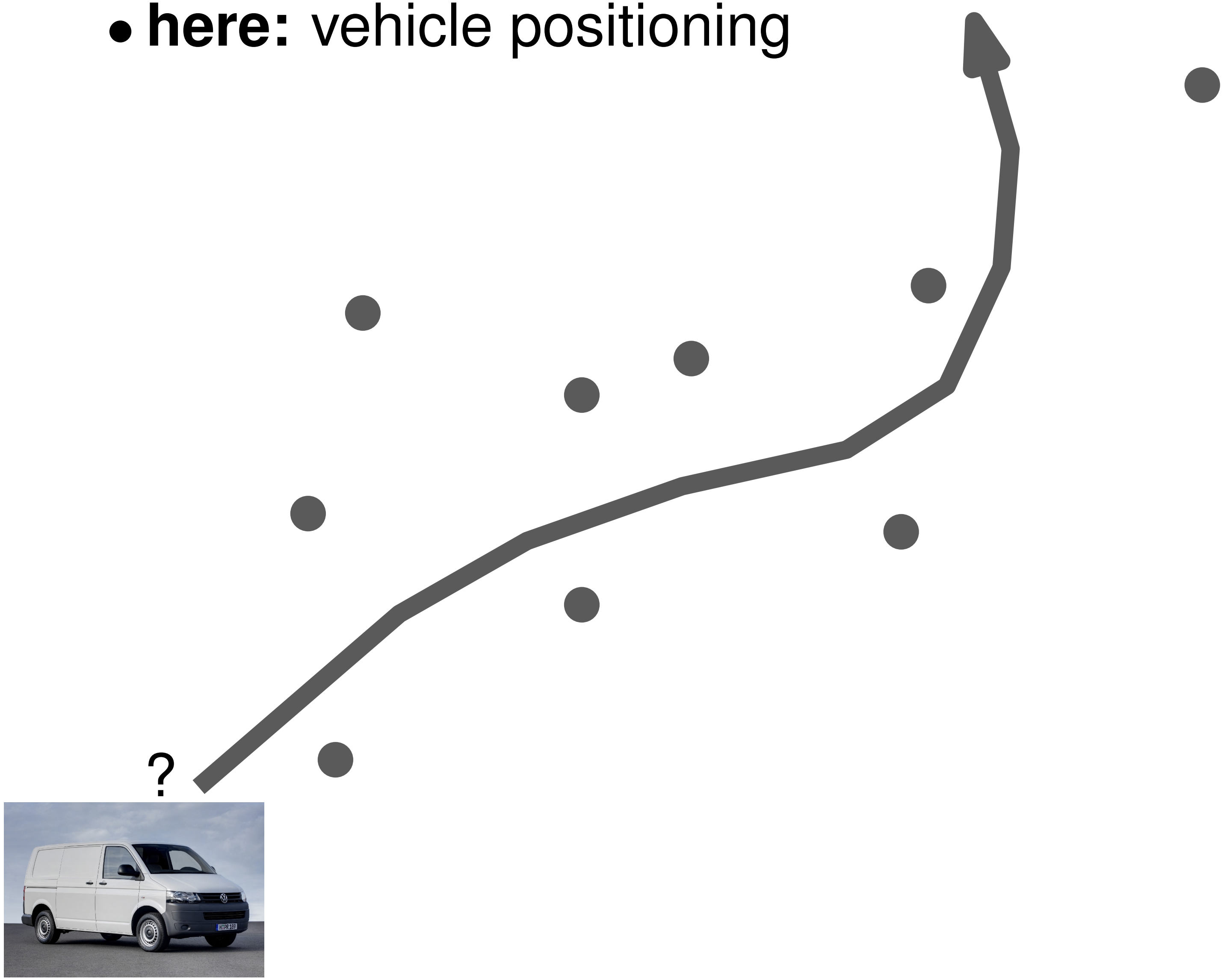


- points may represent any kind of landmarks
- **here:** poles (e.g., of traffic signs) observed with a vehicle-mounted laser scanner

Introduction

Applications of point pattern matching

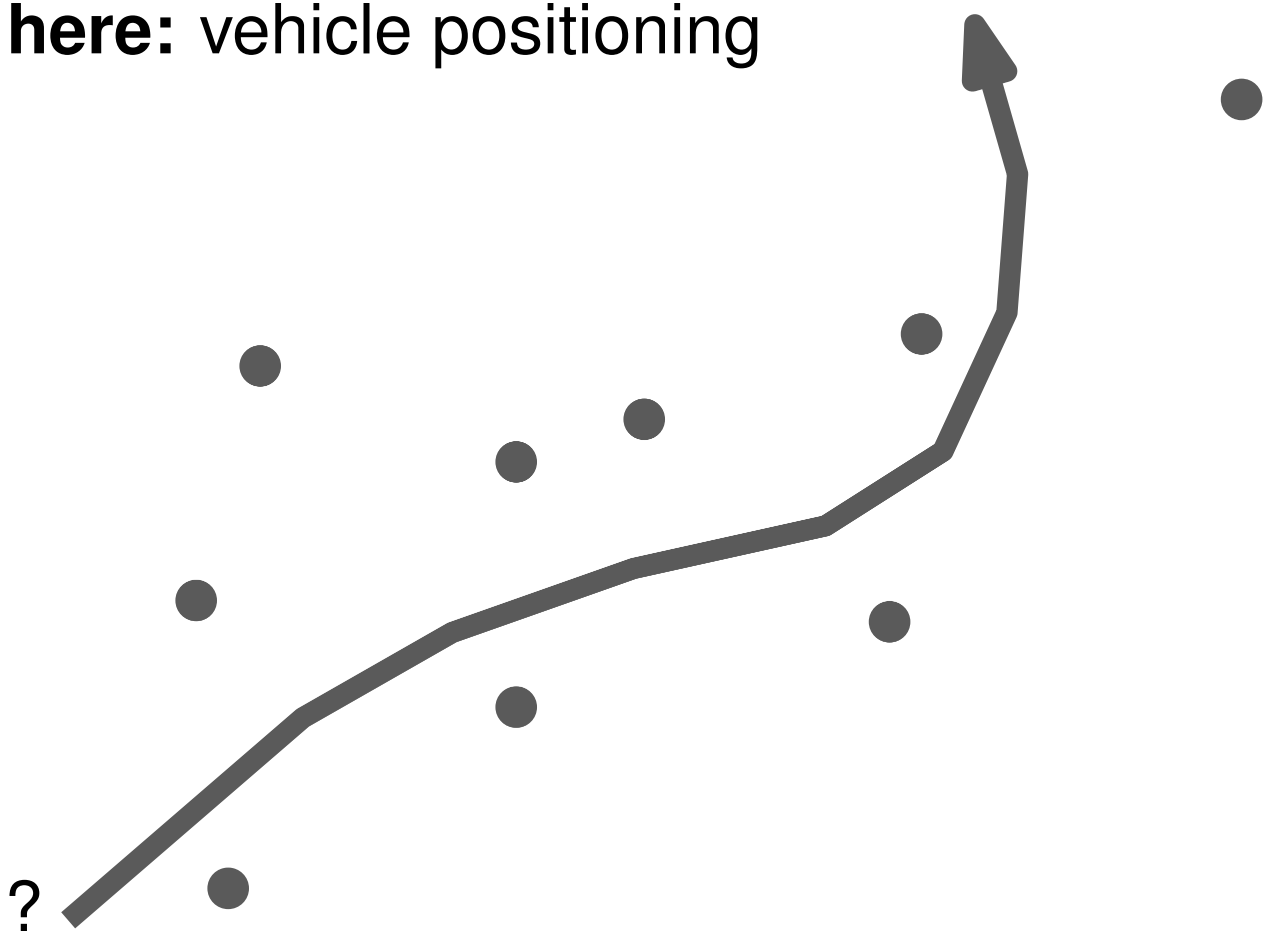
- **here:** vehicle positioning



Introduction

Applications of point pattern matching

- **here:** vehicle positioning

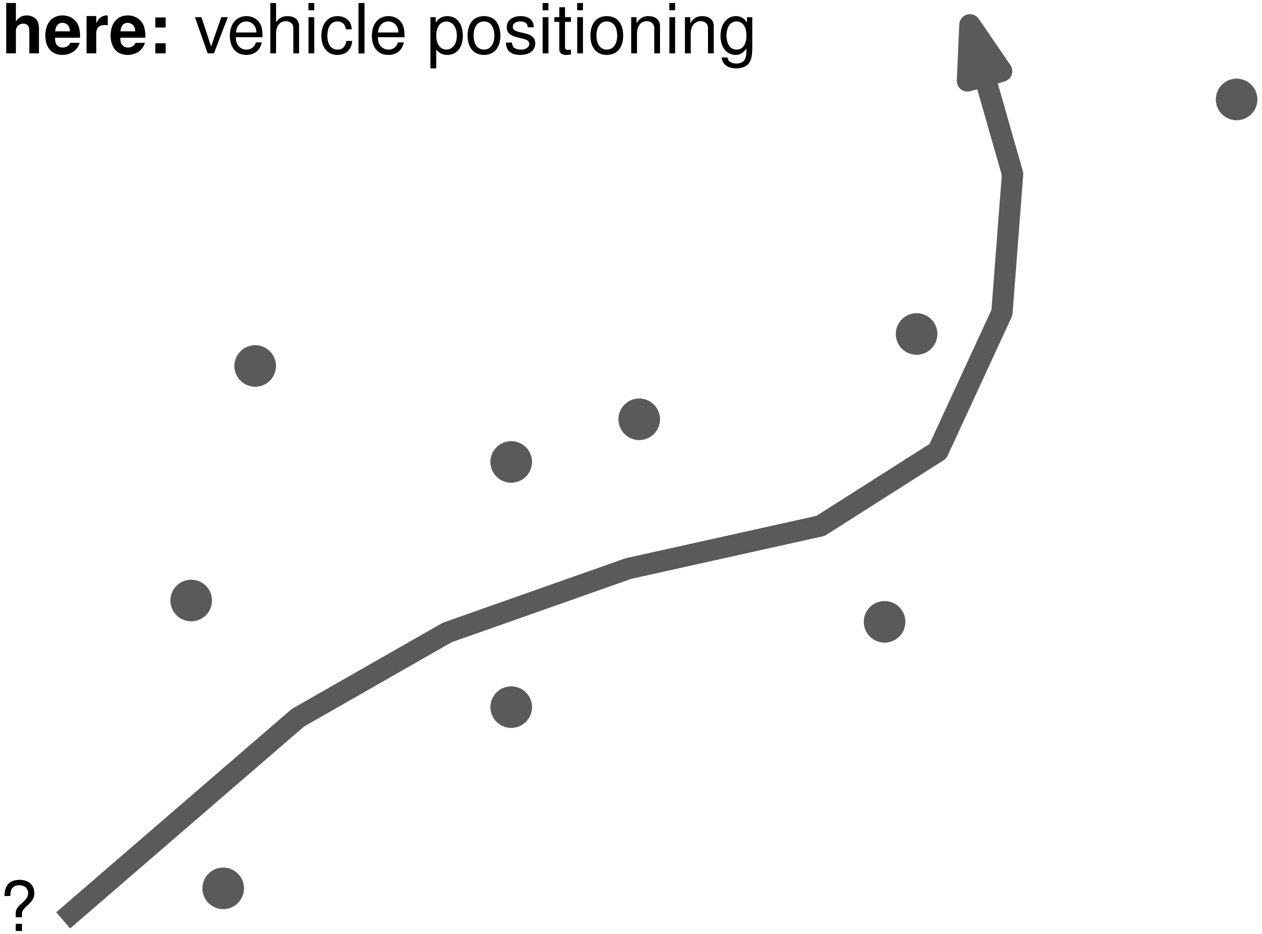


- coordinates may be erroneous
- global rigid transformation does not exist

Introduction

Applications of point pattern matching

- **here:** vehicle positioning

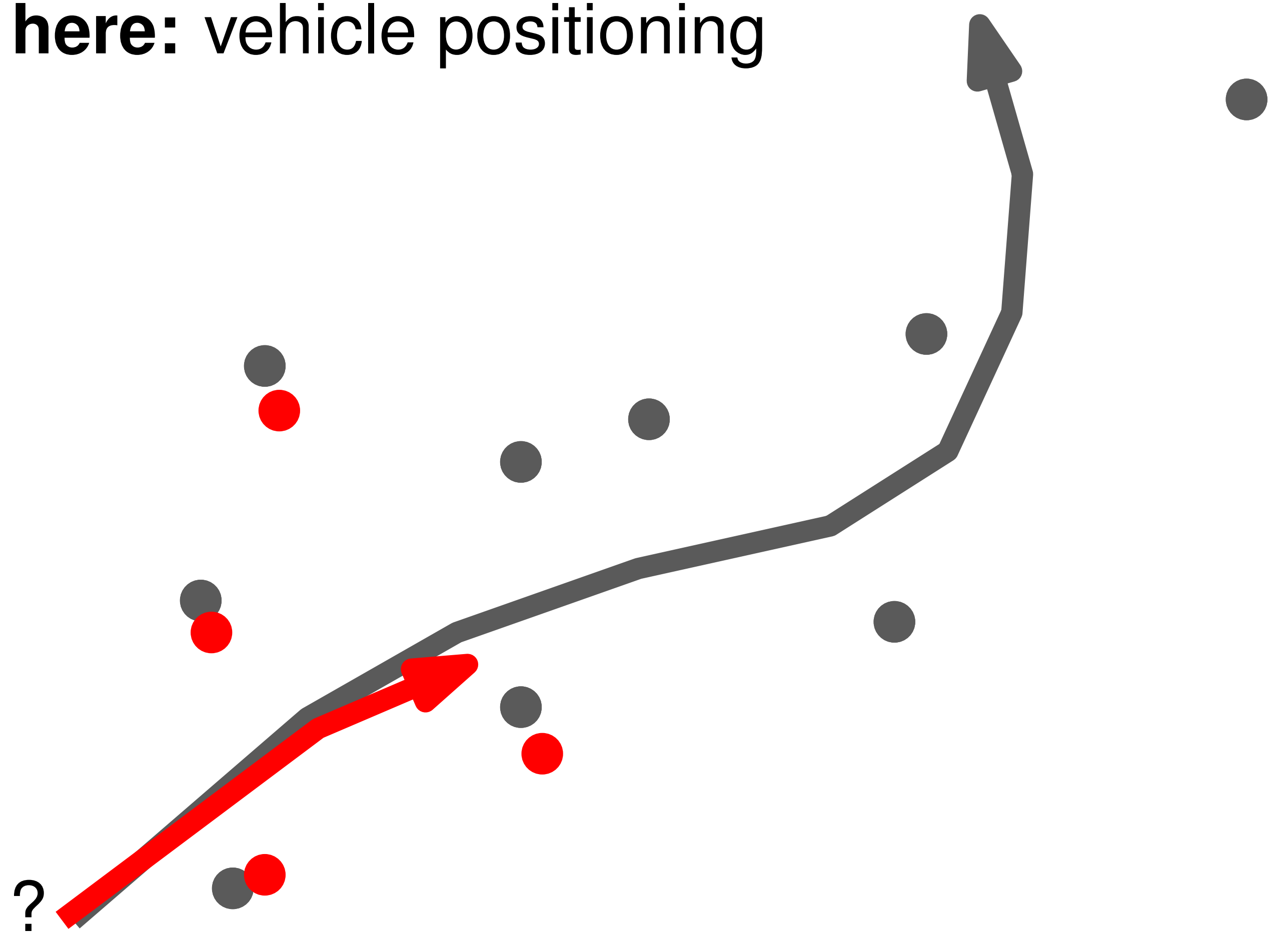


- coordinates may be erroneous
- global rigid transformation does not exist

Introduction

Applications of point pattern matching

- **here:** vehicle positioning

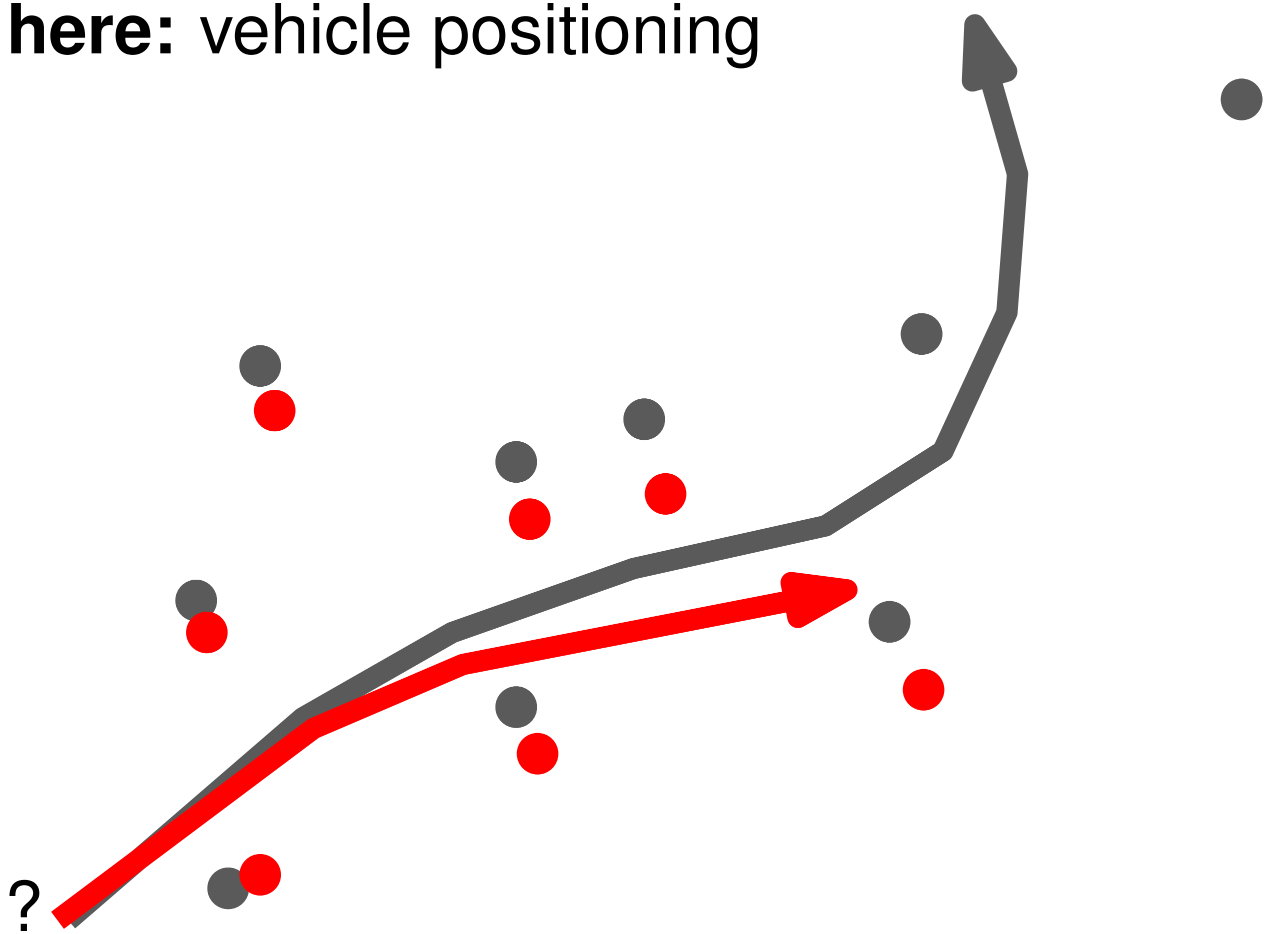


- coordinates may be erroneous
- global rigid transformation does not exist

Introduction

Applications of point pattern matching

- **here:** vehicle positioning

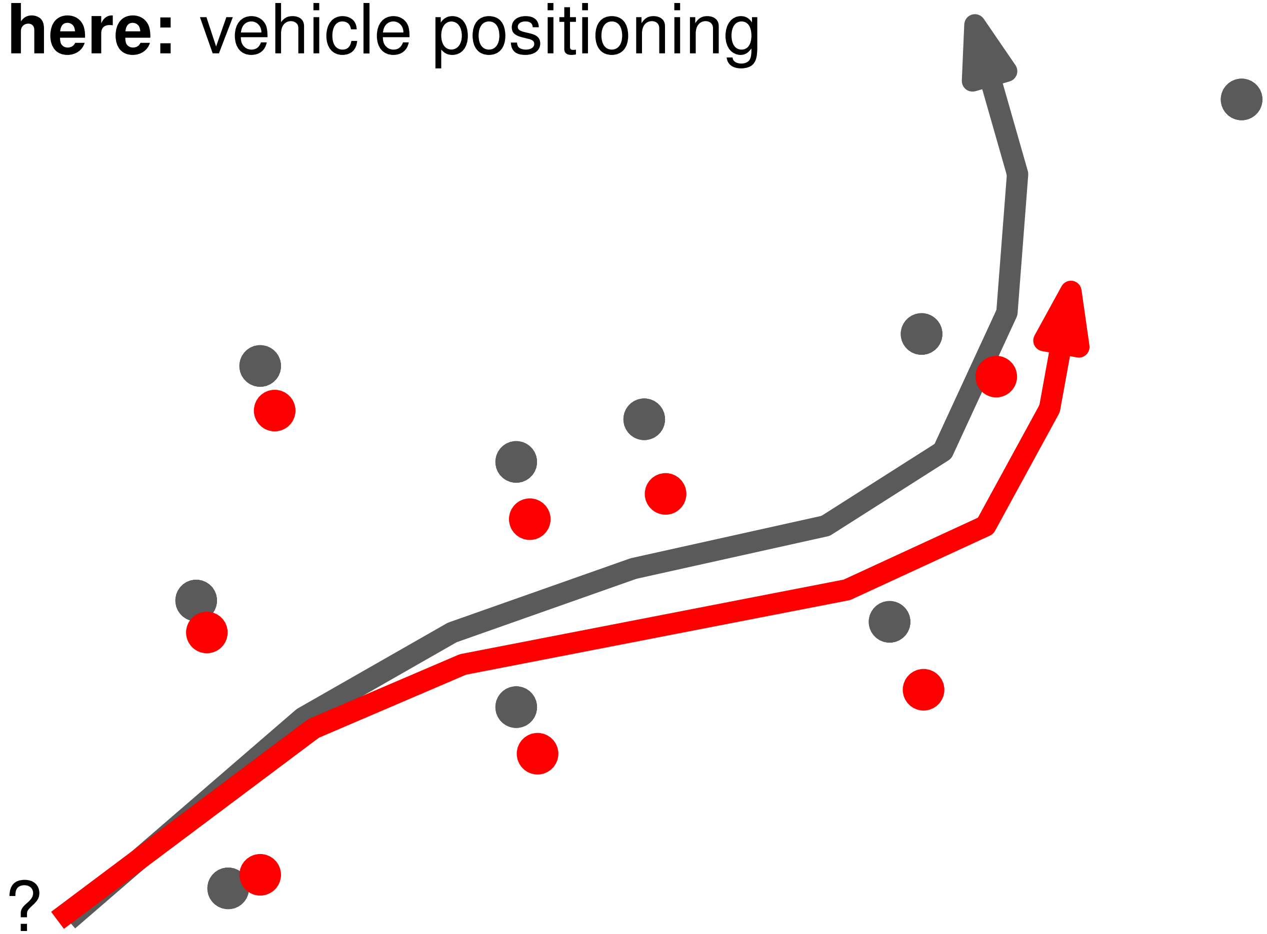


- coordinates may be erroneous
- global rigid transformation does not exist

Introduction

Applications of point pattern matching

- **here:** vehicle positioning

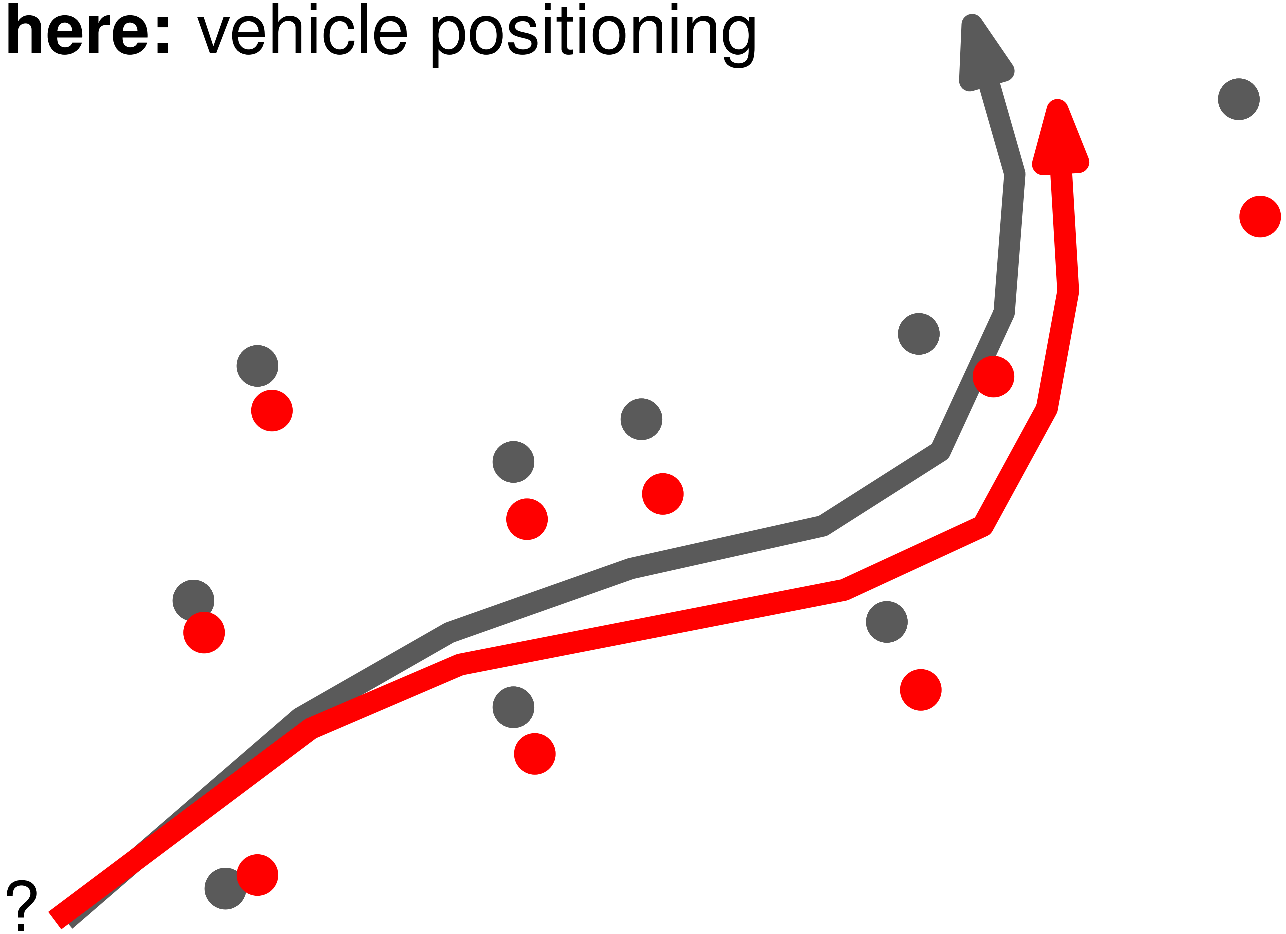


- coordinates may be erroneous
- global rigid transformation does not exist

Introduction

Applications of point pattern matching

- **here:** vehicle positioning

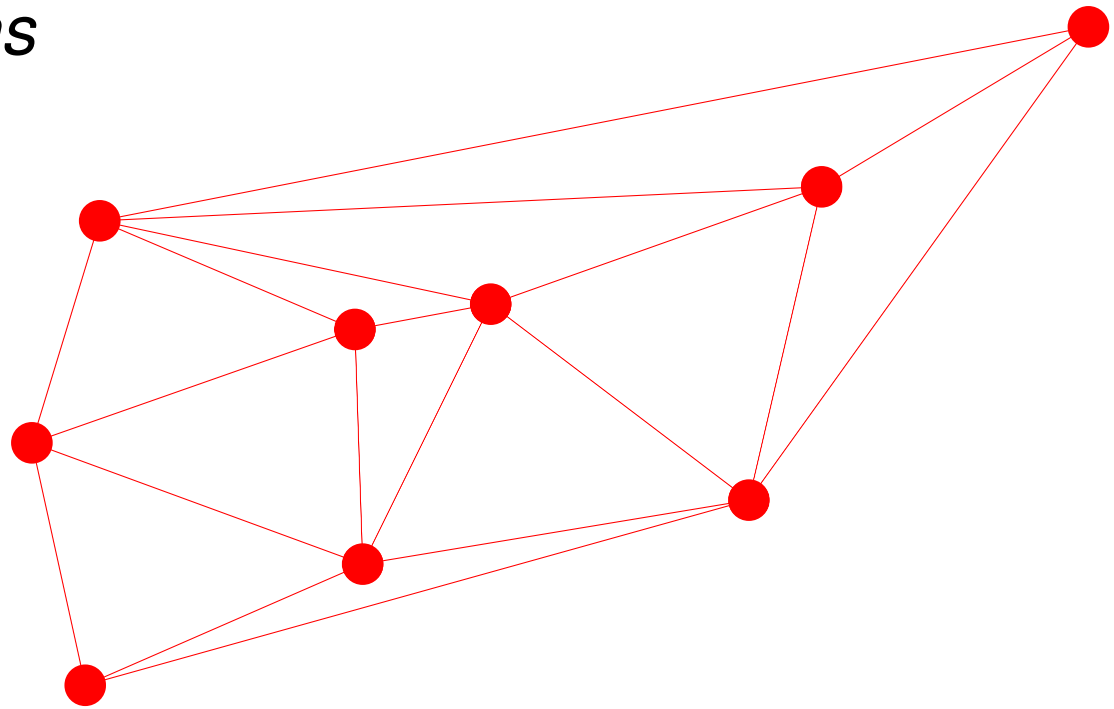


- coordinates may be erroneous
- global rigid transformation does not exist

Introduction

Our approach:

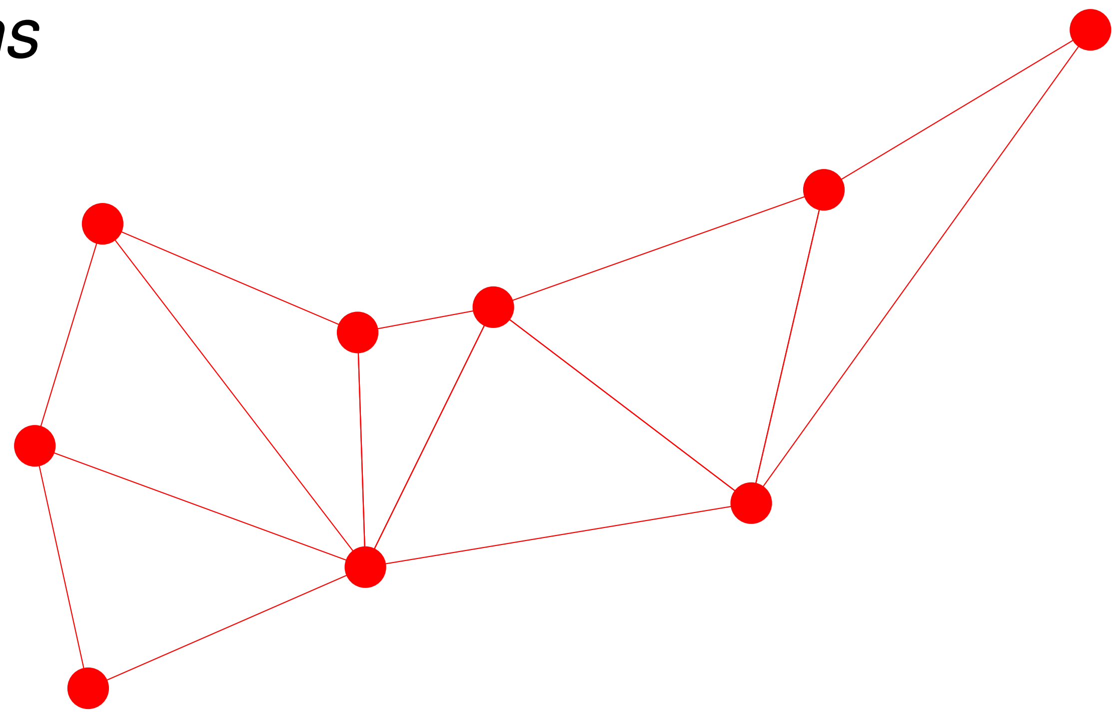
- triangulate observed points
- graph matching: match triangles with triangles in a reference database based on *geometric similarity* and *neighbourhood relations*



Introduction

Our approach:

- triangulate observed points
- graph matching: match triangles with triangles in a reference database based on *geometric similarity* and *neighbourhood relations*

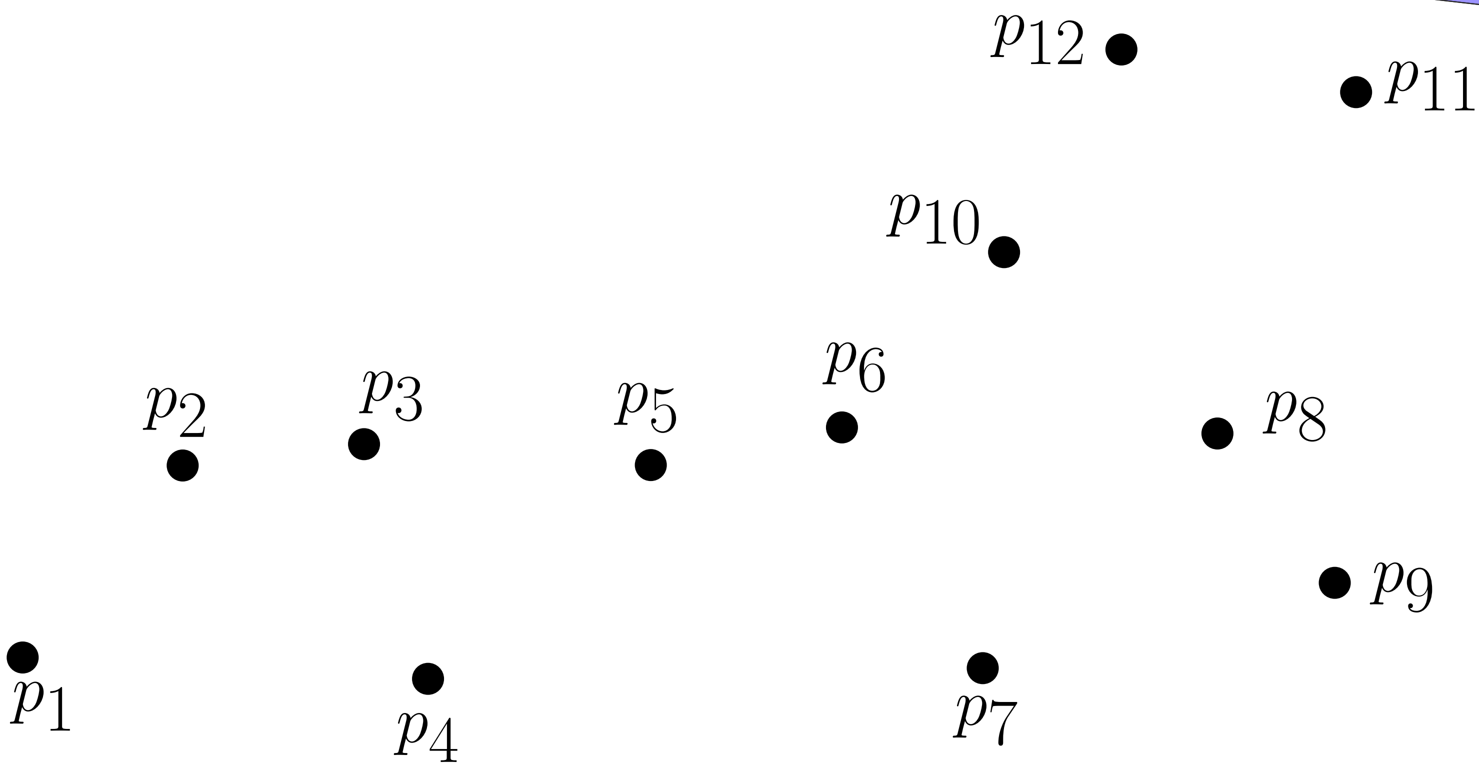


- idea to avoid NP-hard graph matching problem (e.g., subgraph isomorphism problem): only use a triangle strip

Outline

- Triangulation Algorithm
- Matching Problem
- Matching Algorithm
- Experimental Results
- Conclusion/Outlook

Triangulation Algorithm



Input:

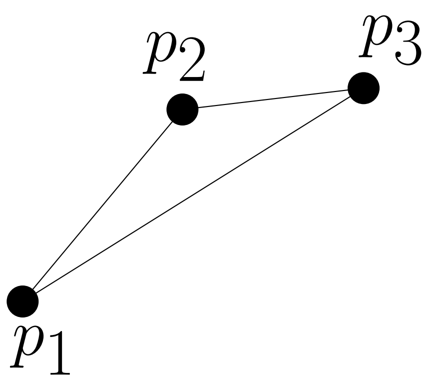
- point sequence $(p_1, p_2, \ldots, p_m)$

Output:

- triangle sequence $(t_1, t_2, \ldots, t_{m-2})$

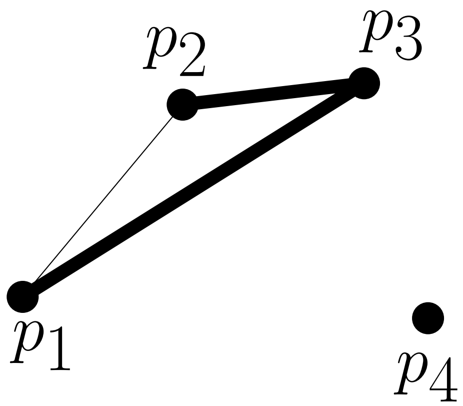
Triangulation Algorithm

- Define first triangle as (p_1, p_2, p_3)
- For $i = 4$ to m append triangle strip by a triangle including p_i and one of the two edges that were added last.



Triangulation Algorithm

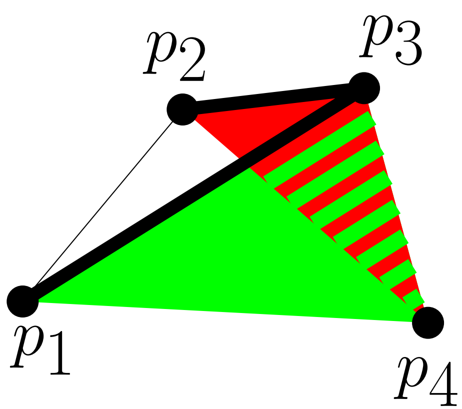
- Define first triangle as (p_1, p_2, p_3)
- For $i = 4$ to m append triangle strip by a triangle including p_i and one of the two edges that were added last.



Triangulation Algorithm

– Define first triangle as (p_1, p_2, p_3)

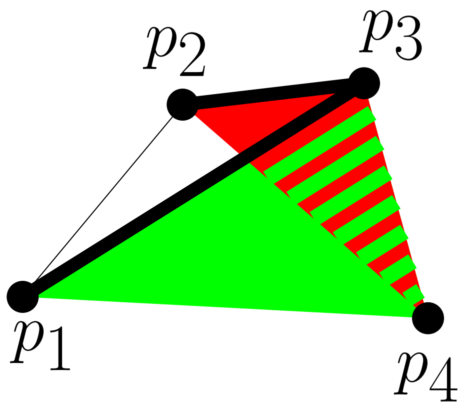
– For $i = 4$ to m append triangle strip by a triangle including p_i and one of the two edges that were added last.



Triangulation Algorithm

– Define first triangle as (p_1, p_2, p_3)

– For $i = 4$ to m append triangle strip by a triangle including p_i and one of the two edges that were added last.

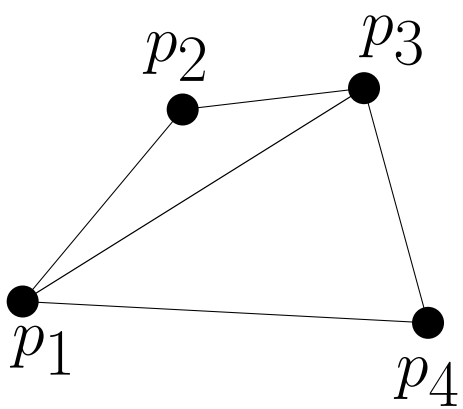


If exactly one candidate triangle overlaps the last triangle then select the other candidate triangle

Triangulation Algorithm

– Define first triangle as (p_1, p_2, p_3)

– For $i = 4$ to m append triangle strip by a triangle including p_i and one of the two edges that were added last.

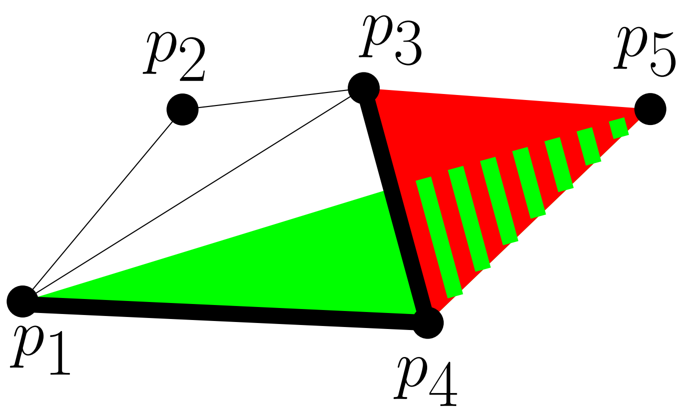


If exactly one candidate triangle overlaps the last triangle then select the other candidate triangle

Triangulation Algorithm

– Define first triangle as (p_1, p_2, p_3)

– For $i = 4$ to m append triangle strip by a triangle including p_i and one of the two edges that were added last.

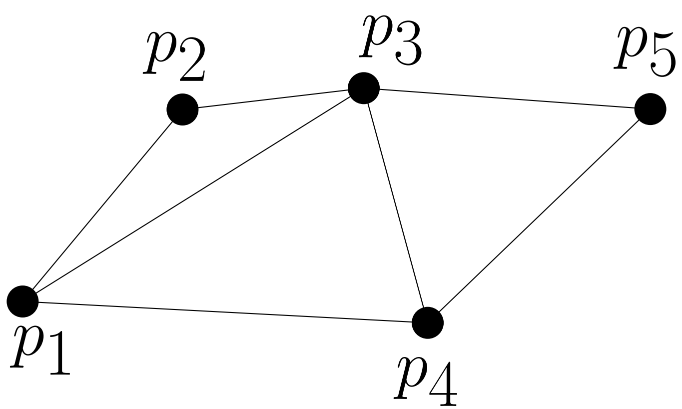


If exactly one candidate triangle overlaps the last triangle then select the other candidate triangle

Triangulation Algorithm

– Define first triangle as (p_1, p_2, p_3)

– For $i = 4$ to m append triangle strip by a triangle including p_i and one of the two edges that were added last.

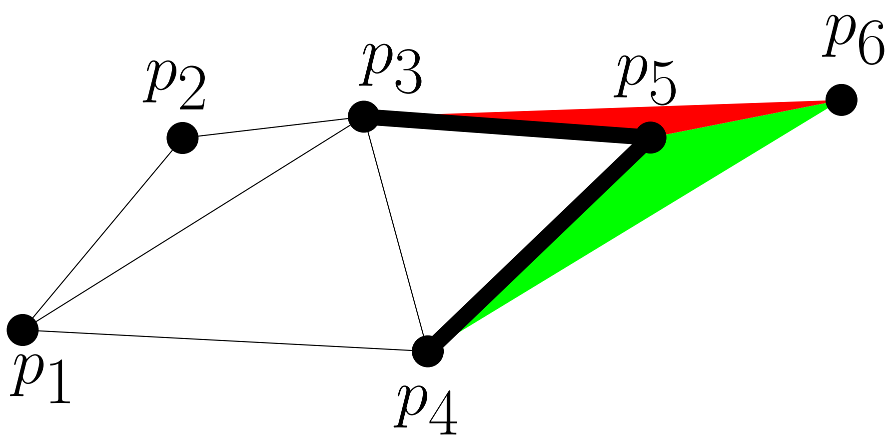


If exactly one candidate triangle overlaps the last triangle then select the other candidate triangle

Triangulation Algorithm

– Define first triangle as (p_1, p_2, p_3)

– For $i = 4$ to m append triangle strip by a triangle including p_i and one of the two edges that were added last.



If exactly one candidate triangle overlaps the last triangle then

select the other candidate triangle

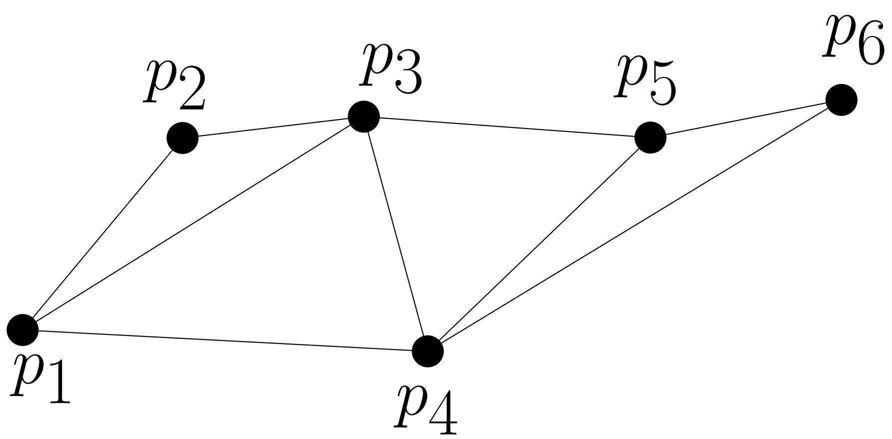
else

maximize the minimum angle

Triangulation Algorithm

– Define first triangle as (p_1, p_2, p_3)

– For $i = 4$ to m append triangle strip by a triangle including p_i and one of the two edges that were added last.



If exactly one candidate triangle overlaps the last triangle then

select the other candidate triangle

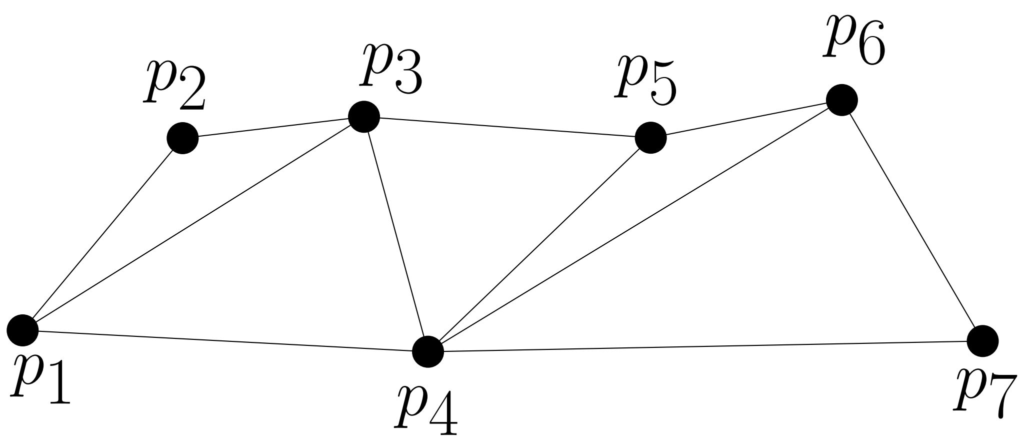
else

maximize the minimum angle

Triangulation Algorithm

– Define first triangle as (p_1, p_2, p_3)

– For $i = 4$ to m append triangle strip by a triangle including p_i and one of the two edges that were added last.



If exactly one candidate triangle overlaps the last triangle then

select the other candidate triangle

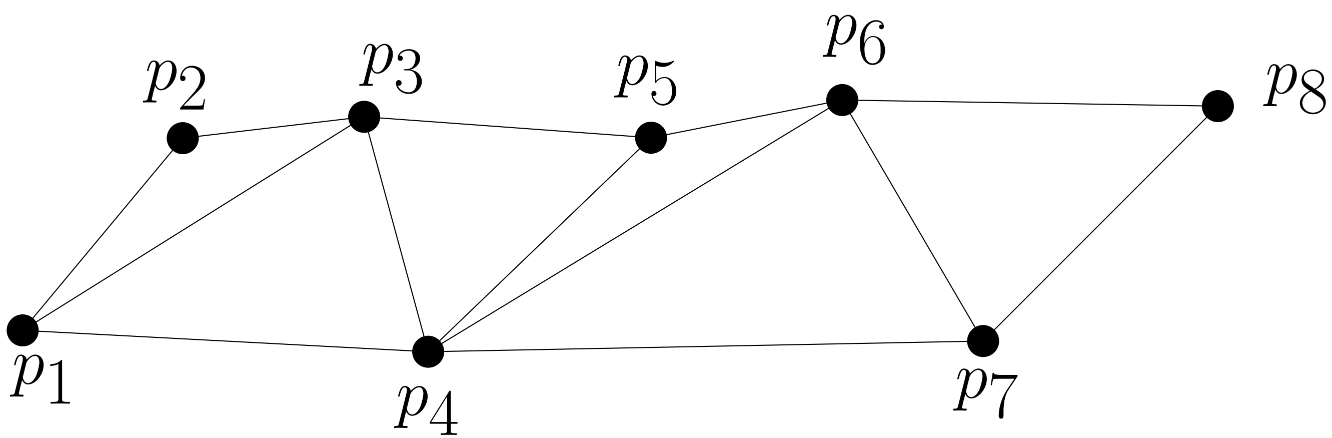
else

maximize the minimum angle

Triangulation Algorithm

– Define first triangle as (p_1, p_2, p_3)

– For $i = 4$ to m append triangle strip by a triangle including p_i and one of the two edges that were added last.



If exactly one candidate triangle overlaps the last triangle then

select the other candidate triangle

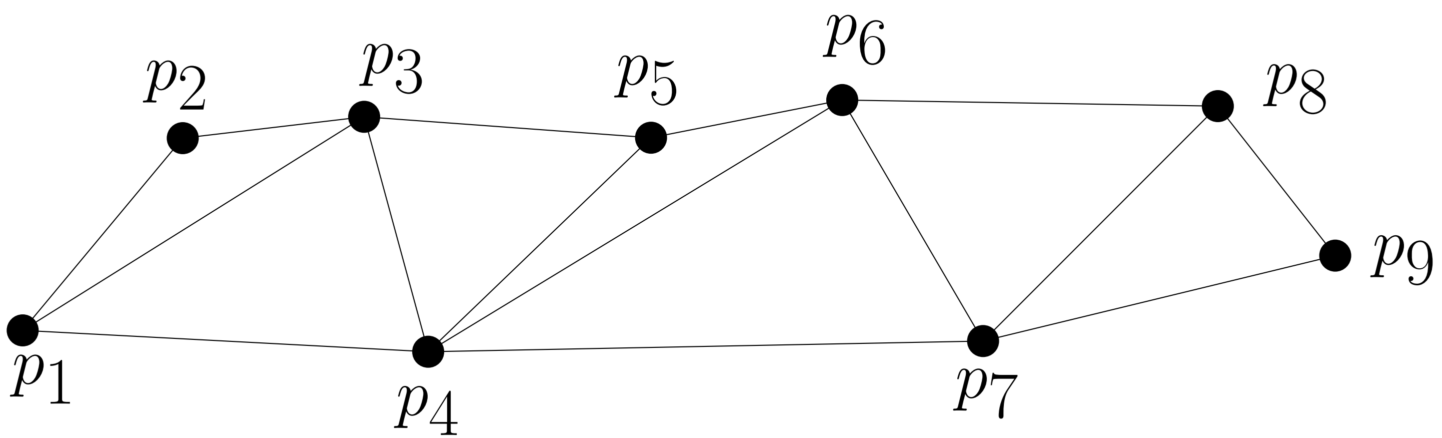
else

maximize the minimum angle

Triangulation Algorithm

– Define first triangle as (p_1, p_2, p_3)

– For $i = 4$ to m append triangle strip by a triangle including p_i and one of the two edges that were added last.



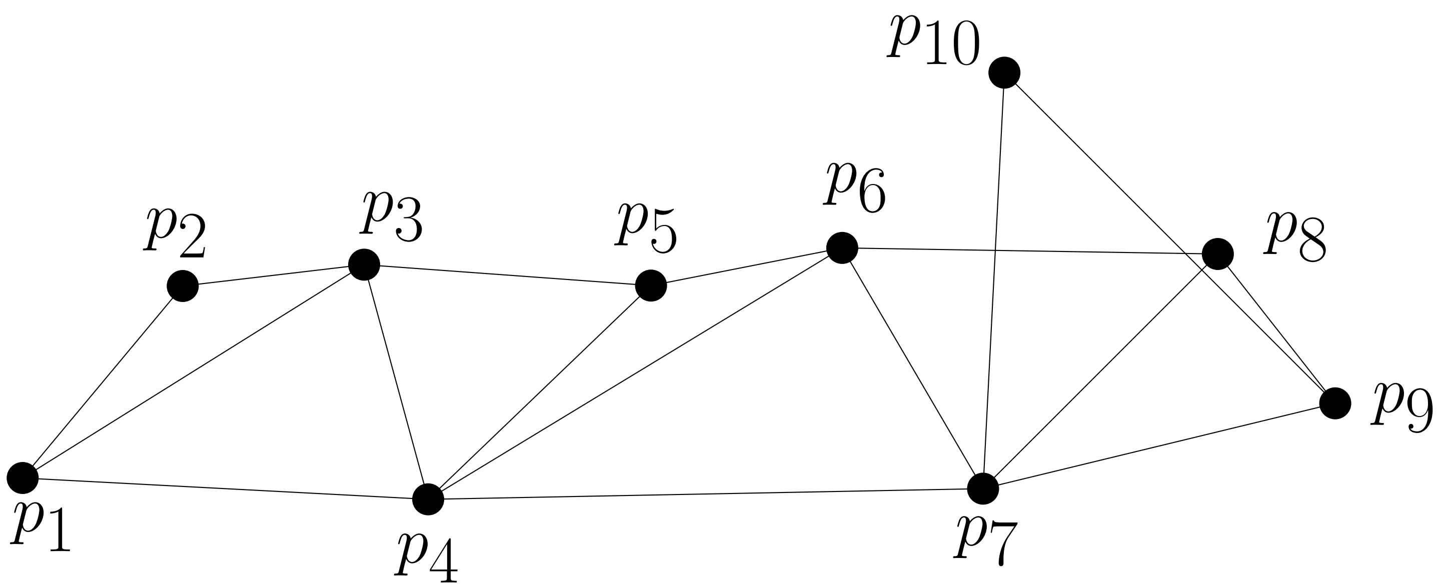
If exactly one candidate triangle overlaps the last triangle then

select the other candidate triangle

else

maximize the minimum angle

Triangulation Algorithm



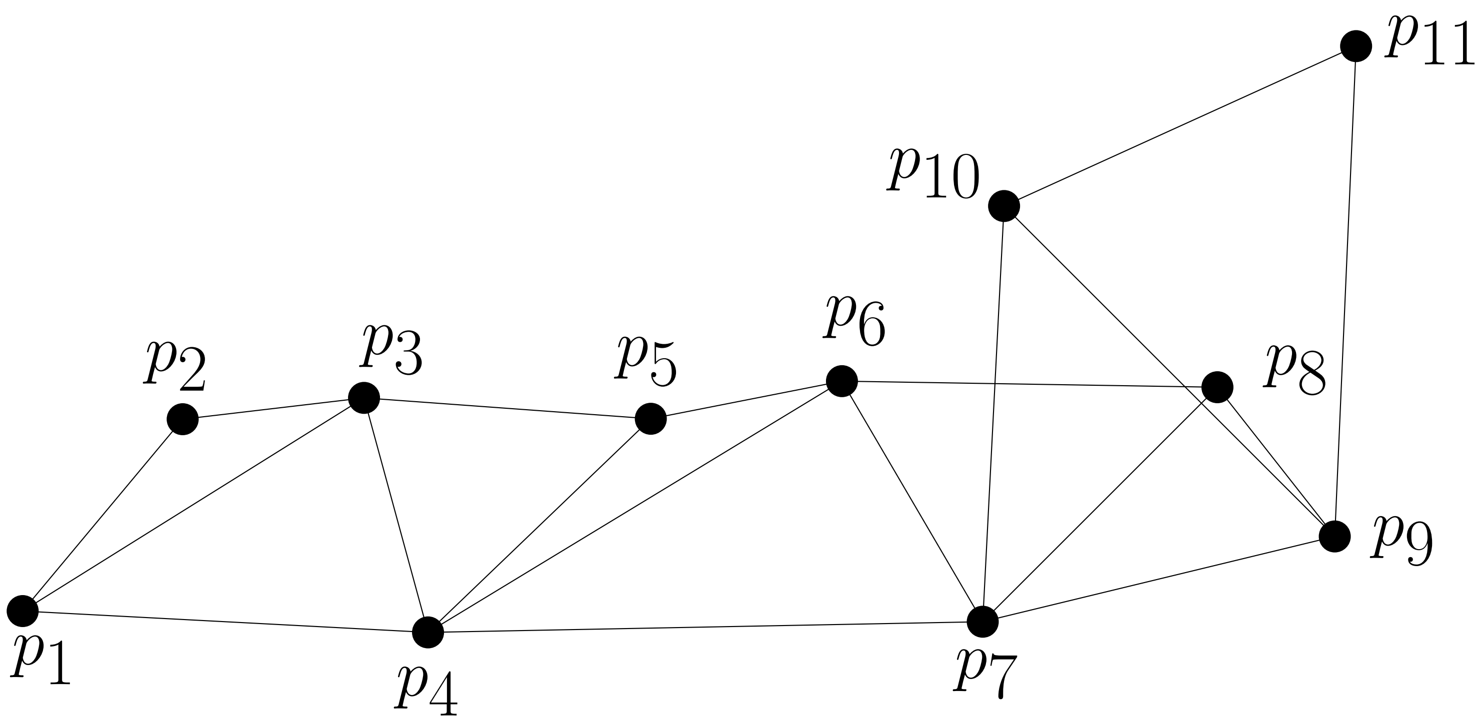
If exactly one candidate triangle overlaps the last triangle then

select the other candidate triangle

else

maximize the minimum angle

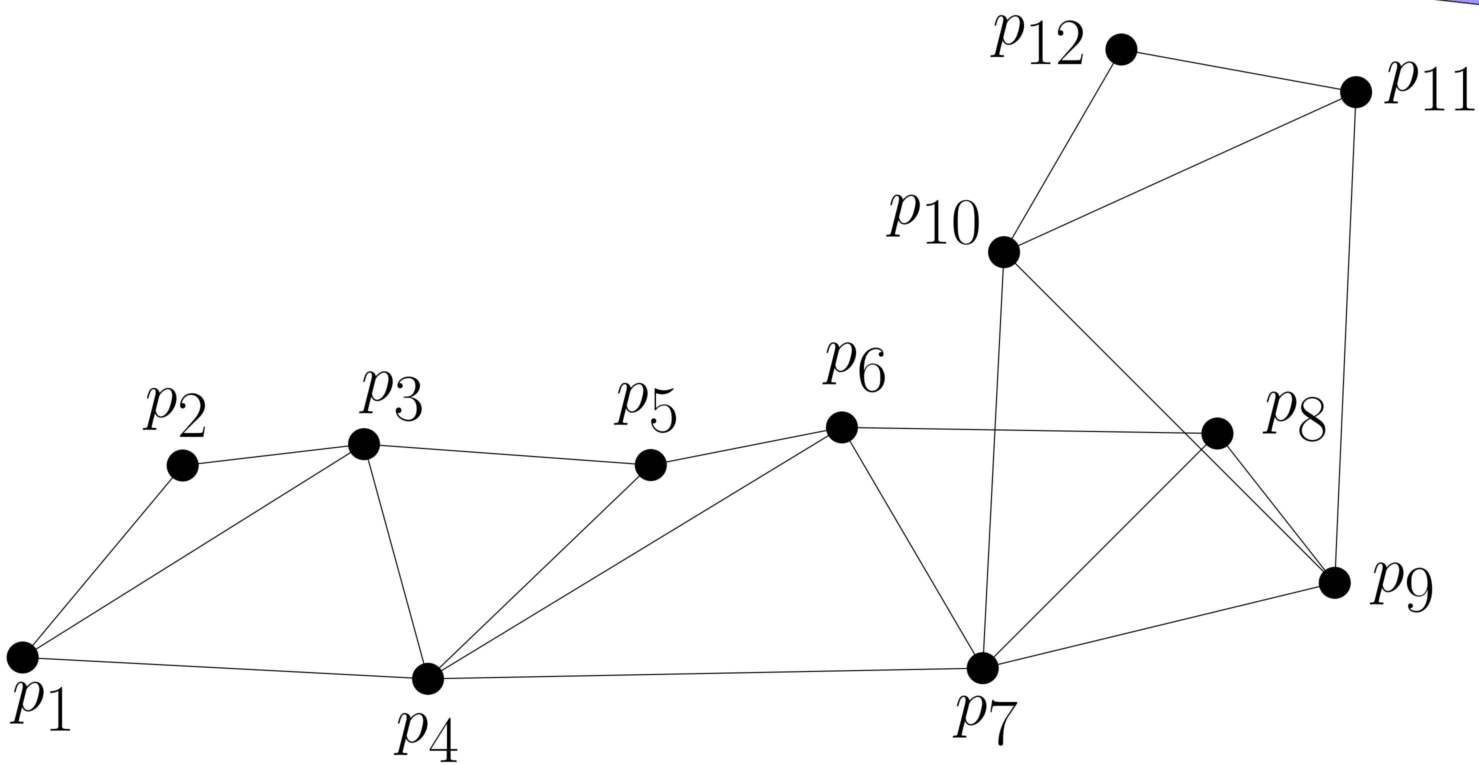
Triangulation Algorithm



If exactly one candidate triangle overlaps the last triangle then
  select the other candidate triangle

else
  maximize the minimum angle

Triangulation Algorithm



If exactly one candidate triangle overlaps the last triangle then

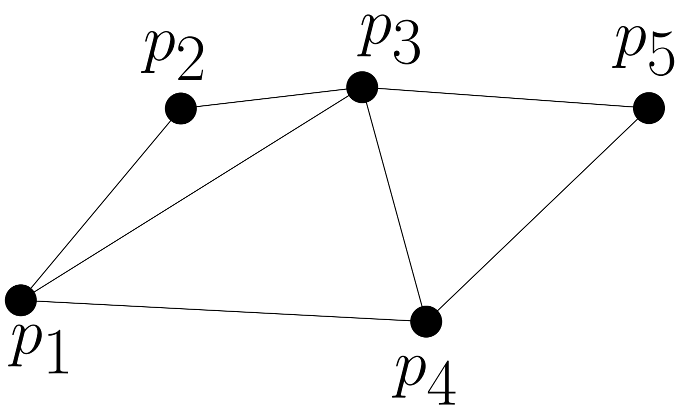
select the other candidate triangle

else

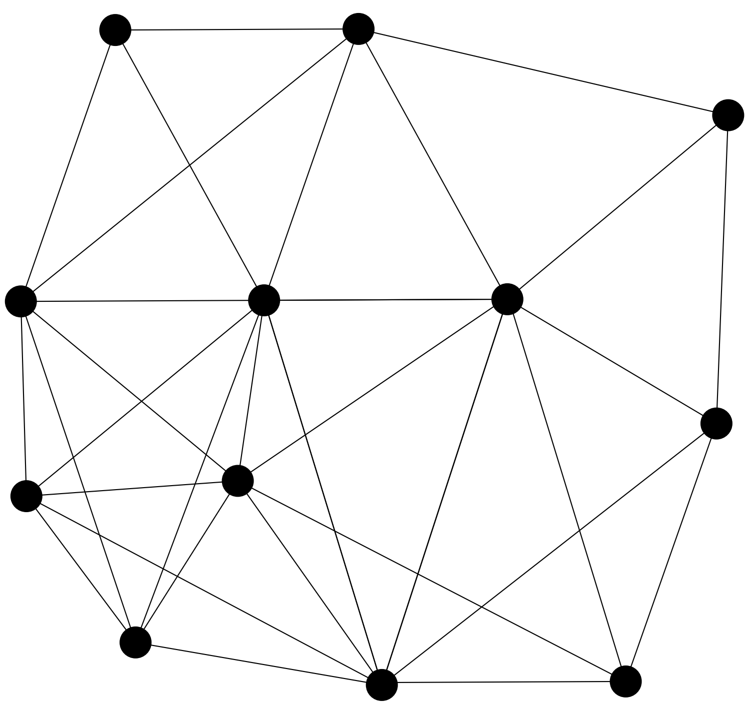
maximize the minimum angle

Matching Problem

observed triangles T

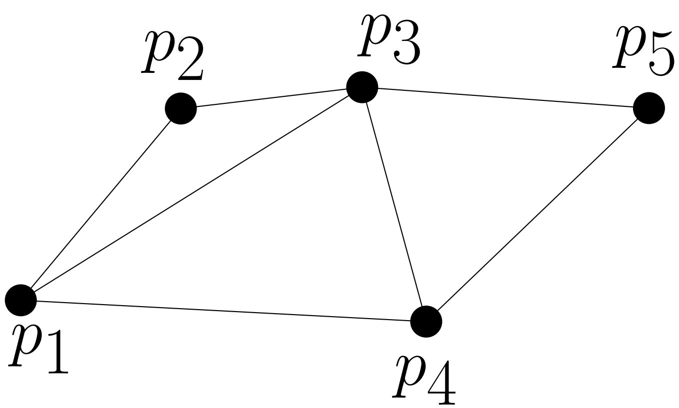


reference triangles T'

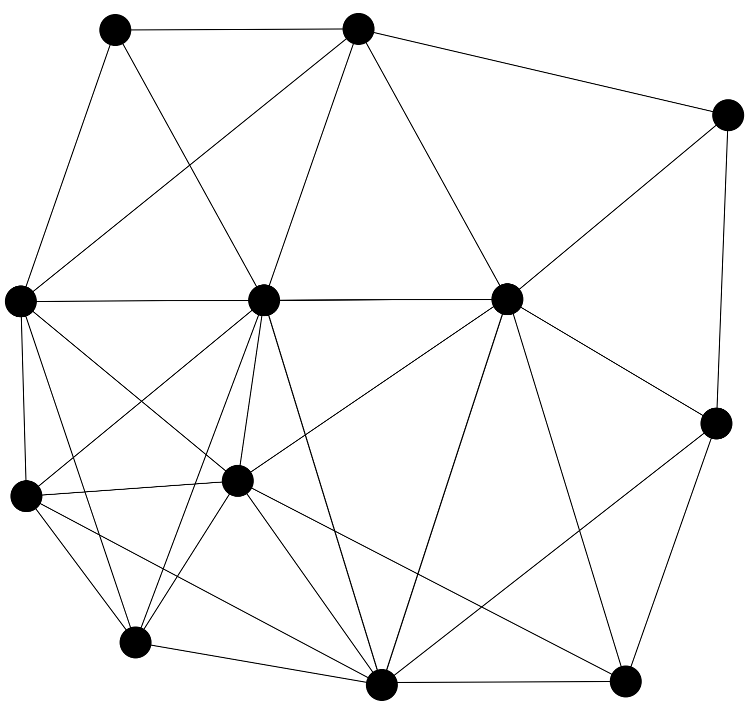


Matching Problem

observed triangles T

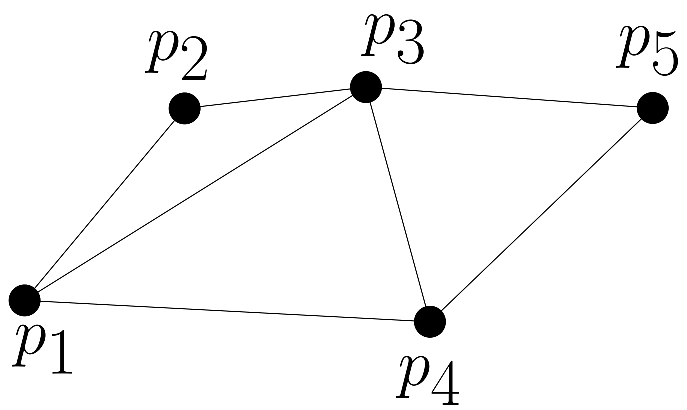


reference triangles T' $=$ all possible triangles of three reference points

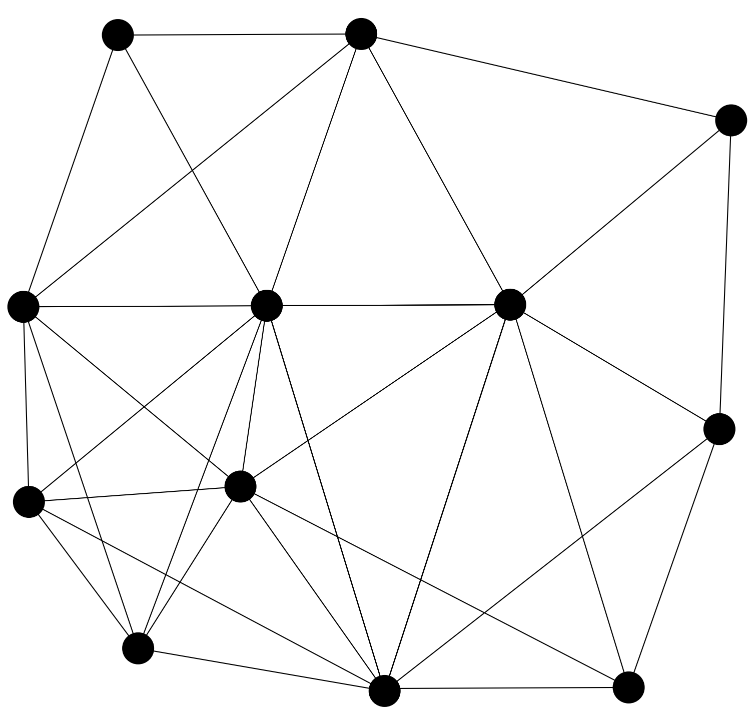


Matching Problem

observed triangles T



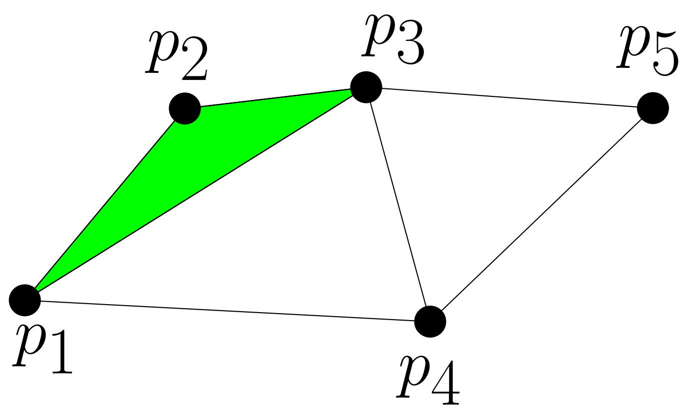
reference triangles T'



Find a set of triangle matches $\theta \in T \times T'$.

Matching Problem

observed triangles T



Find a set of triangle matches $\theta \in T \times T'$.

Constraint 1:

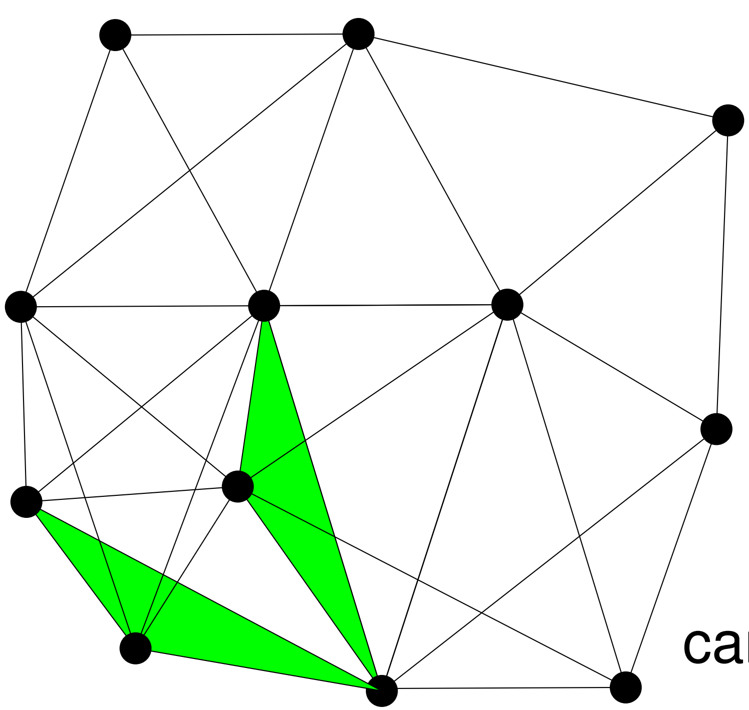
For each match $(t, t') \in \theta$ the triangles t and t' must be sufficiently similar.

$$\left| \text{longest side of } t - \text{ longest side of } t' \right| \leq \varepsilon$$

$$\left| \text{2nd longest side of } t - \text{ 2nd longest side of } t' \right| \leq \varepsilon$$

$$\left| \text{3rd longest side of } t - \text{ 3rd longest side of } t' \right| \leq \varepsilon$$

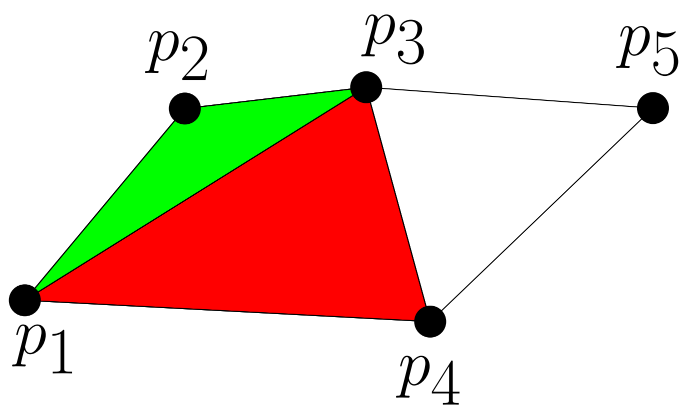
reference triangles T'



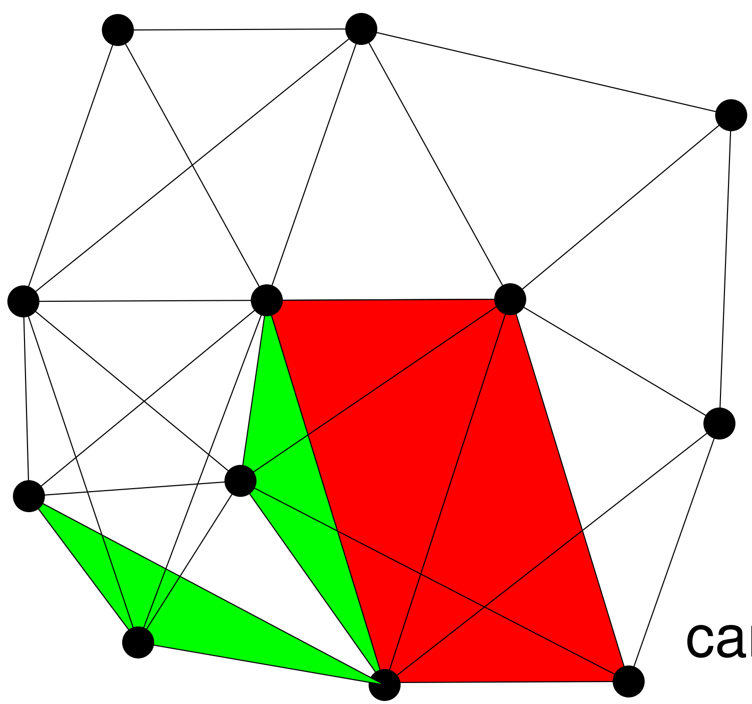
candidate matches

Matching Problem

observed triangles T



reference triangles T'



candidate matches

Find a set of triangle matches $\theta \in T \times T'$.

Constraint 1:
For each match $(t, t') \in \theta$ the triangles t and t' must be sufficiently similar.

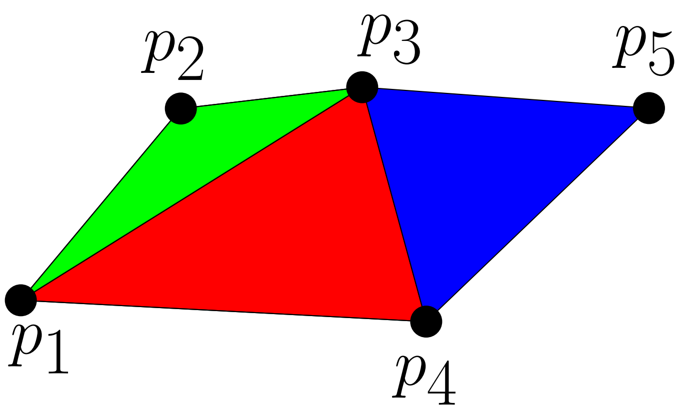
$$\left|\text{longest side of } t - \text{ longest side of } t'\right| \leq \varepsilon$$

$$\left|\text{2nd longest side of } t - \text{ 2nd longest side of } t'\right| \leq \varepsilon$$

$$\left|\text{3rd longest side of } t - \text{ 3rd longest side of } t'\right| \leq \varepsilon$$

Matching Problem

observed triangles T



Find a set of triangle matches $\theta \in T \times T'$.

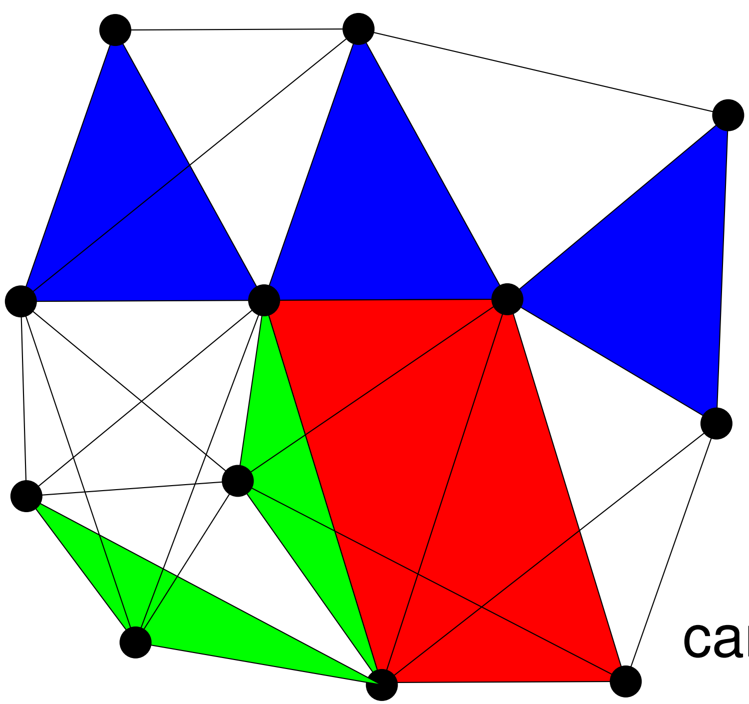
Constraint 1:
For each match $(t, t') \in \theta$ the triangles t and t' must be sufficiently similar.

$$\left| \text{longest side of } t - \text{ longest side of } t' \right| \leq \varepsilon$$

$$\left| \text{2nd longest side of } t - \text{ 2nd longest side of } t' \right| \leq \varepsilon$$

$$\left| \text{3rd longest side of } t - \text{ 3rd longest side of } t' \right| \leq \varepsilon$$

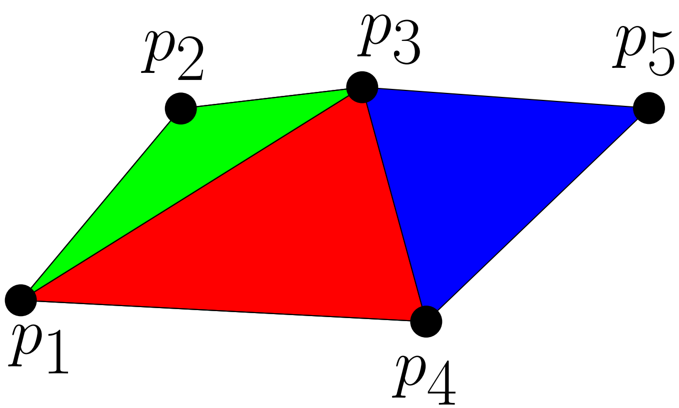
reference triangles T'



candidate matches

Matching Problem

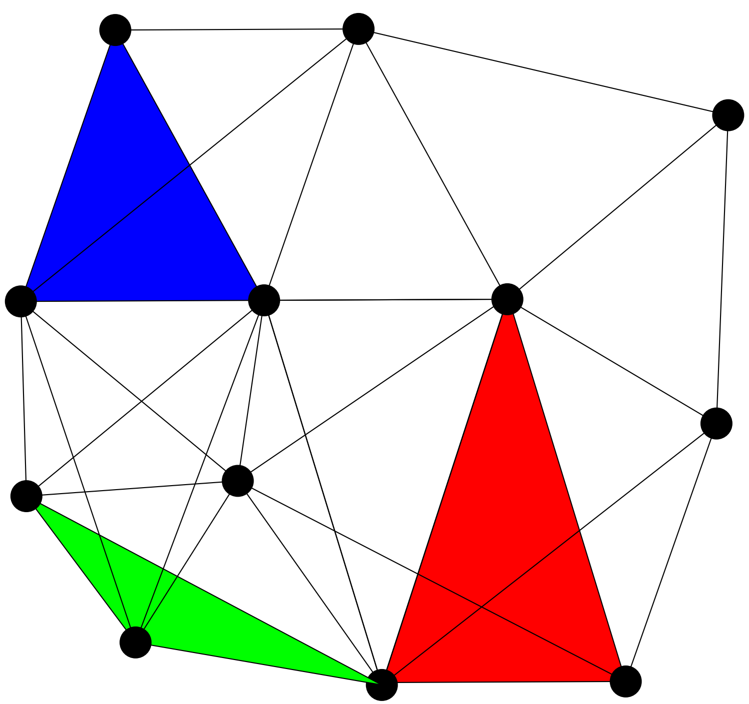
observed triangles T



Find a set of triangle matches $\theta \in T \times T'$.

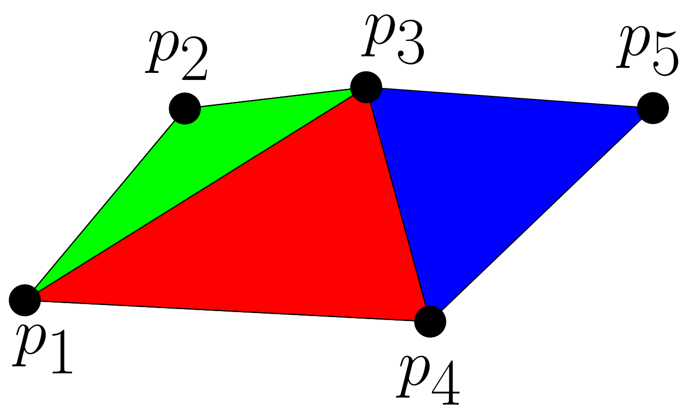
Constraint 2:
A triangle $t \in T$ must not be matched to more than one reference triangle.

reference triangles T'



Matching Problem

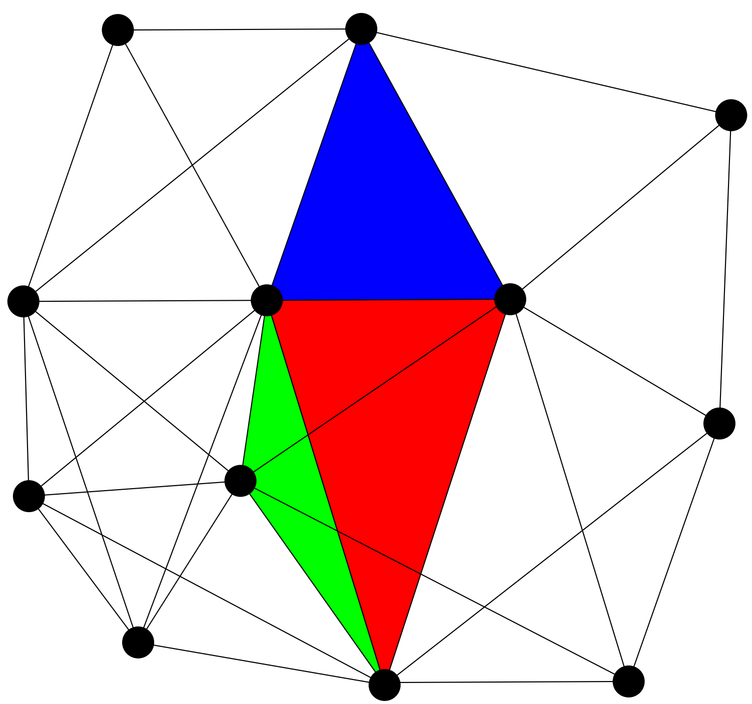
observed triangles T



Find a set of triangle matches $\theta \in T \times T'$.

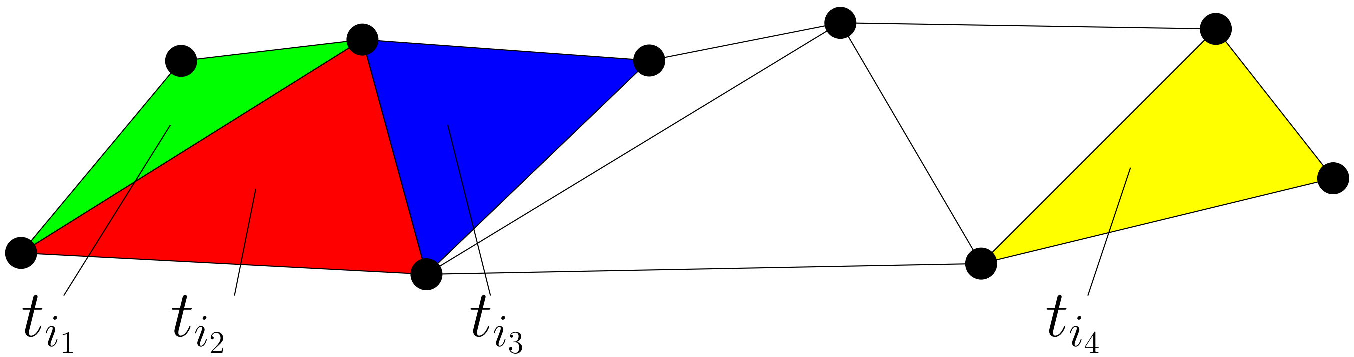
Constraint 3:
For each two matches $(a, a') \in \theta$ and $(b, b') \in \theta$ the triangles a' and b' must share an edge if a and b share an edge.

reference triangles T'



Matching Problem

observed triangles T



triangles in T that are matched

Some triangles in T cannot be matched, therefore:

- maximize $|\theta|$ (= number of matches)
- among solutions maximizing $|\theta|$ maximize quality of matches
- additional constraints to ensure that solutions for different components "fit together"

Matching Problem

Constraint 4:

t_{i_j} and $t_{i_{j+1}}$ must not be matched to the same reference triangle.

Constraint 5:

If t_{i_j} and $t_{i_{j+1}}$ do not share an edge then
the matched reference triangles must not share an edge.

Constraint 6:

If t_{i_j} and $t_{i_{j+1}}$ do not share an edge then
the distances between t_{i_j} and $t_{i_{j+1}}$ must be sufficiently similar to
the distances between the matched reference triangles.

Matching Algorithm

Offline:

- build an index (a three-dimensional kd-tree) that references each triangle in T' by its side lengths

Online:

- triangulate observed point set $\to T$
- set up directed acyclic graph G_{match} based on T and T'
- search path of maximum weight in $G_{\mathrm{match}} \to \theta$

Matching Algorithm

Offline:

- build an index (a three-dimensional kd-tree) that references each triangle in T' by its side lengths

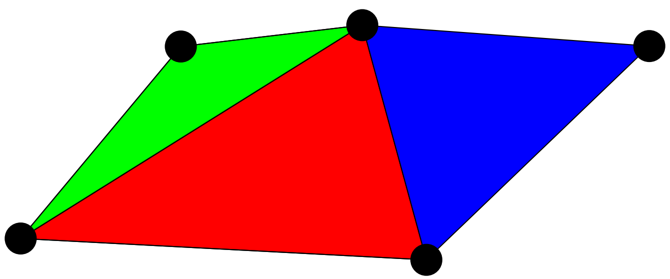
Online:

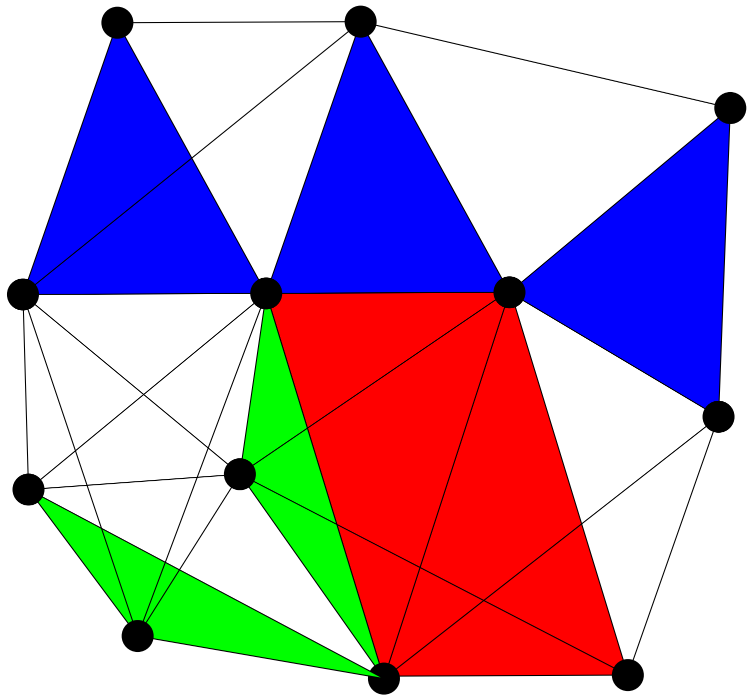
- triangulate observed point set $\rightarrow T$
- set up directed acyclic graph G_{match} based on T and T'
- search path of maximum weight in $G_{\text{match}} \rightarrow \theta$

Matching Algorithm

Set up directed acyclic graph $G_{\mathrm{match}}(V_{\mathrm{match}}, A_{\mathrm{match}})$:

- V_{match} contains a node for each candidate match
- V_{match} can be found by applying range queries to kd-tree (one query for each triangle in T)

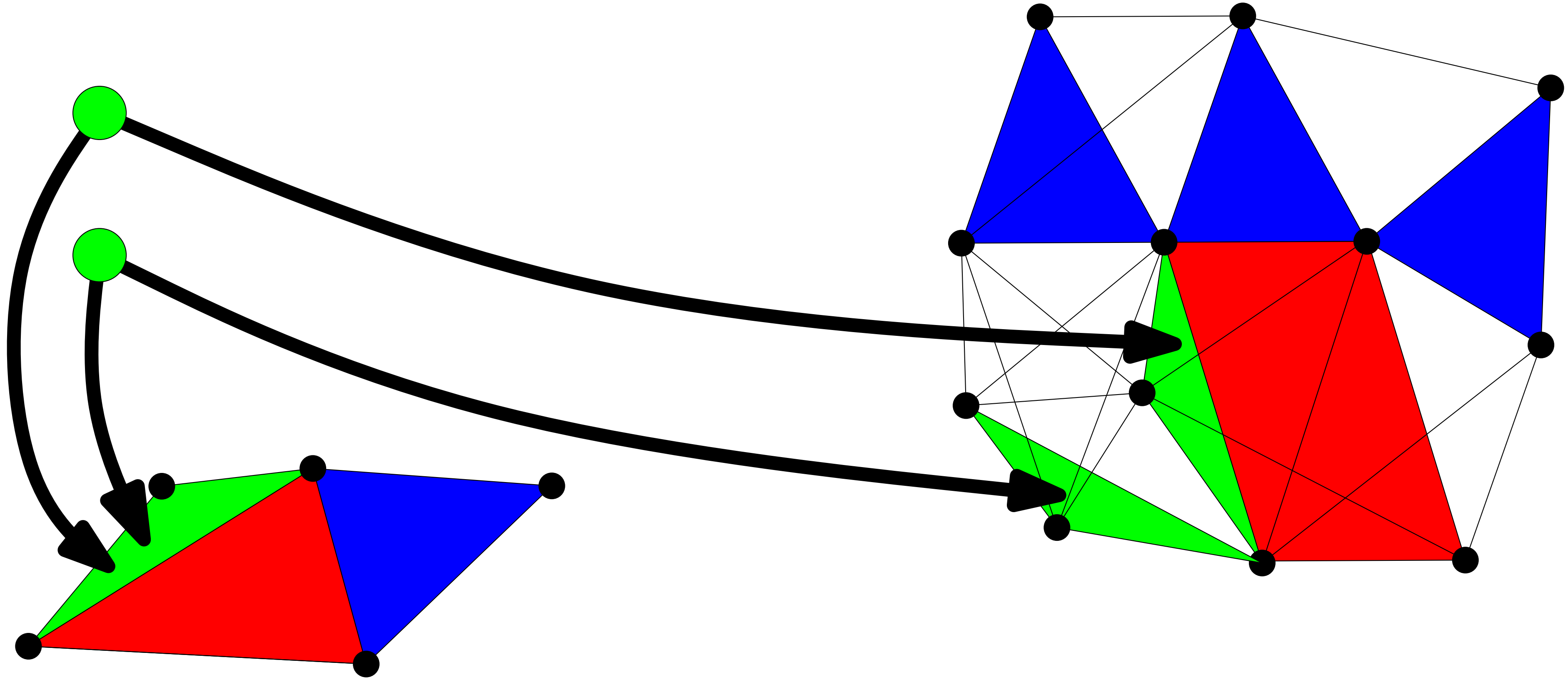




Matching Algorithm

Set up directed acyclic graph $G_{\mathrm{match}}(V_{\mathrm{match}}, A_{\mathrm{match}})$:

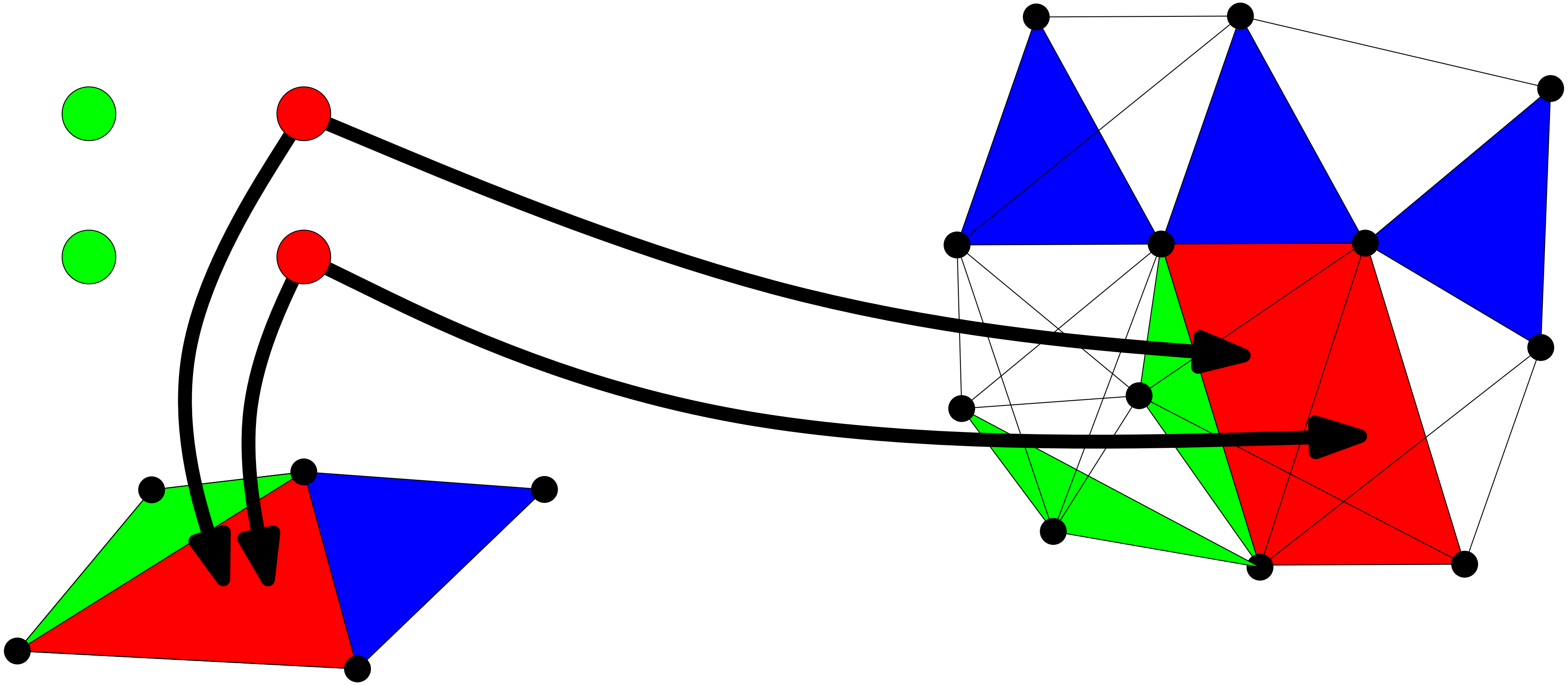
- V_{match} contains a node for each candidate match
- V_{match} can be found by applying range queries to kd-tree (one query for each triangle in T)



Matching Algorithm

Set up directed acyclic graph $G_{\mathrm{match}}(V_{\mathrm{match}}, A_{\mathrm{match}})$:

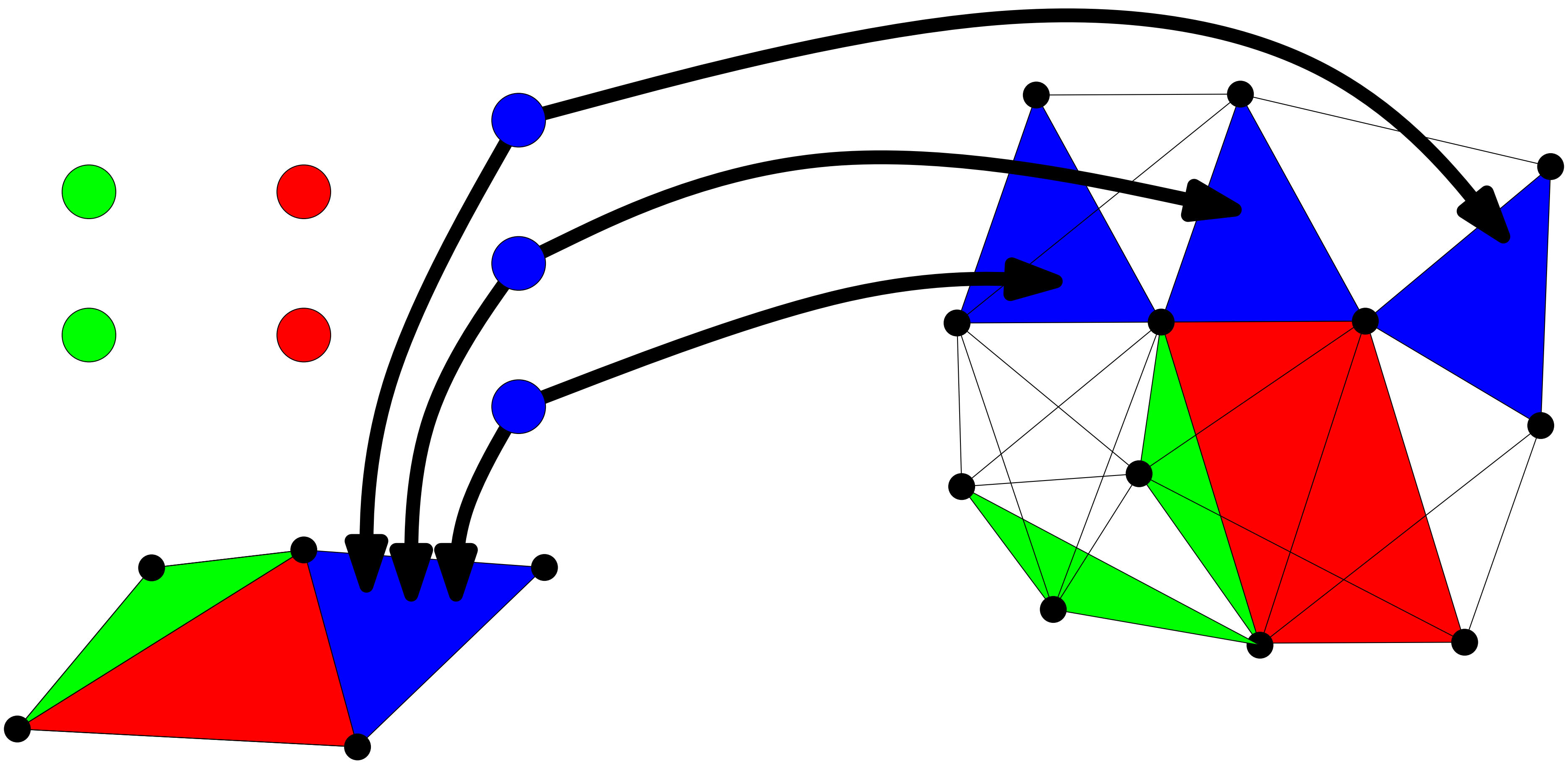
- V_{match} contains a node for each candidate match
- V_{match} can be found by applying range queries to kd-tree (one query for each triangle in T)



Matching Algorithm

Set up directed acyclic graph $G_{\mathrm{match}}(V_{\mathrm{match}}, A_{\mathrm{match}})$:

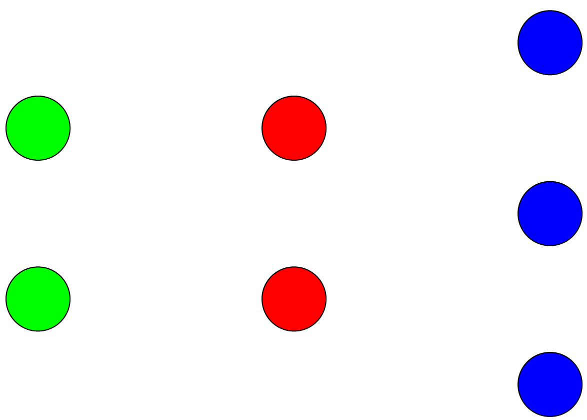
- V_{match} contains a node for each candidate match
- V_{match} can be found by applying range queries to kd-tree (one query for each triangle in T)



Matching Algorithm

Set up directed acyclic graph $G_{\mathrm{match}}(V_{\mathrm{match}}, A_{\mathrm{match}})$:

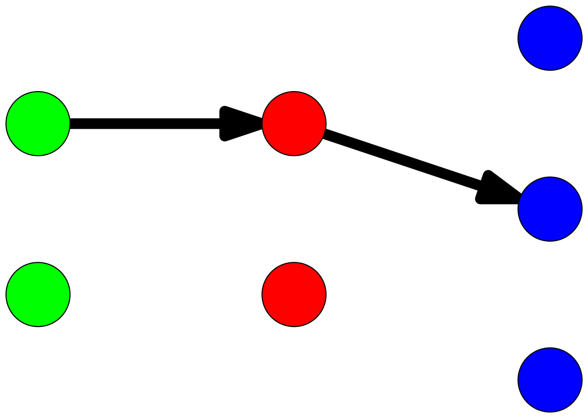
- V_{match} contains a node for each candidate match
- V_{match} can be found by applying range queries to kd-tree (one query for each triangle in T)



Matching Algorithm

Set up directed acyclic graph $G_{\mathrm{match}}(V_{\mathrm{match}}, A_{\mathrm{match}})$:

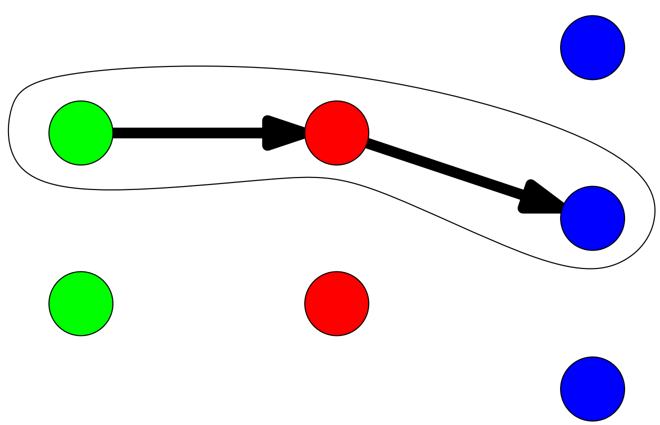
- V_{match} contains a node for each candidate match
- V_{match} can be found by applying range queries to kd-tree (one query for each triangle in T)
- A_{match} contains an arc for each pair of candidate matches that satisfies constraints 1–6



Matching Algorithm

Set up directed acyclic graph $G_{\mathrm{match}}(V_{\mathrm{match}}, A_{\mathrm{match}})$:

- V_{match} contains a node for each candidate match
- V_{match} can be found by applying range queries to kd-tree (one query for each triangle in T)
- A_{match} contains an arc for each pair of candidate matches that satisfies constraints 1–6



Search path of maximum weight in G_{match}:

- solution by dynamic programming in $\mathcal{O}(|V_{\mathrm{match}}| + |A_{\mathrm{match}}|)$ time

Experimental Results

Streetmapper system:

- 4 laser scanners
- GPS
- odometer
- IMU
- used to create reference point set

Experimental Results

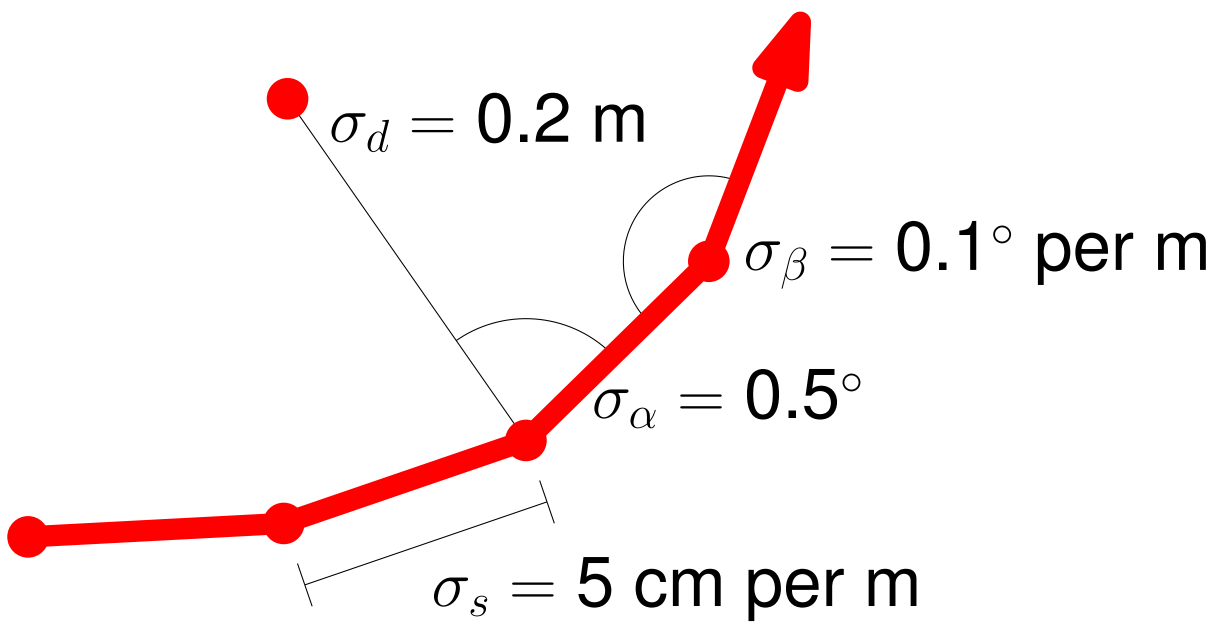
Reference dataset:

- 22 km track in Hannover, Germany
- 2658 reference points
- 643247 reference triangles

Experimental Results

Test samples matched with reference set:

- 88 sub-tracks of the whole track
- noise added



Experimental Results

Experiments with different error tolerances:

ε	0.25 m	0.50 m	0.75 m	1.00 m

All experiments run on Windows PC with 64 bits, 8 GB RAM, 2.93 GHz CPU

Experimental Results

Experiments with different error tolerances:

ε	0.25 m	0.50 m	0.75 m	1.00 m
unmatched triangles	91.5%	55.6%	12.7%	3.5%
correctly matched triangles	7.8%	44.1%	86.9%	96.0%
incorrectly matched triangles	0.7%	0.3%	0.4%	0.5%

All experiments run on Windows PC with 64 bits, 8 GB RAM, 2.93 GHz CPU

Experimental Results

Experiments with different error tolerances:

ε	0.25 m	0.50 m	0.75 m	1.00 m
unmatched triangles	91.5%	55.6%	12.7%	3.5%
correctly matched triangles	7.8%	44.1%	86.9%	96.0%
incorrectly matched triangles	0.7%	0.3%	0.4%	0.5%
avg. # cand. matches / triangle	1.5	9.2	26.7	58.1

All experiments run on Windows PC with 64 bits, 8 GB RAM, 2.93 GHz CPU

Experimental Results

Experiments with different error tolerances:

ε	0.25 m	0.50 m	0.75 m	1.00 m
unmatched triangles	91.5%	55.6%	12.7%	3.5%
correctly matched triangles	7.8%	44.1%	86.9%	96.0%
incorrectly matched triangles	0.7%	0.3%	0.4%	0.5%
avg. # cand. matches / triangle	1.5	9.2	26.7	58.1
avg. solution time	0.04s	0.41s	2.75s	11.88s

All experiments run on Windows PC with 64 bits, 8 GB RAM, 2.93 GHz CPU

Experimental Results

Experiments with different error tolerances:

ε	0.25 m	0.50 m	0.75 m	1.00 m
unmatched triangles	91.5%	55.6%	12.7%	3.5%
correctly matched triangles	7.8%	44.1%	86.9%	96.0%
incorrectly matched triangles	0.7%	0.3%	0.4%	0.5%
avg. # cand. matches / triangle	1.5	9.2	26.7	58.1
avg. solution time	0.04s	0.41s	2.75s	11.88s
instances where majority of matches is correct	77.1%	96.4%	97.6%	96.4%

All experiments run on Windows PC with 64 bits, 8 GB RAM, 2.93 GHz CPU

Experimental Results

Experiments with different error tolerances:

ε	0.25 m	0.50 m	0.75 m	1.00 m
unmatched triangles	91.5%	55.6%	12.7%	3.5%
correctly matched triangles	7.8%	44.1%	86.9%	96.0%
incorrectly matched triangles	0.7%	0.3%	0.4%	0.5%
avg. # cand. matches / triangle	1.5	9.2	26.7	58.1
avg. solution time	0.04s	0.41s	2.75s	11.88s
instances where majority of matches is correct	77.1%	96.4%	97.6%	96.4%

very high success rate in reasonable time

All experiments run on Windows PC with 64 bits, 8 GB RAM, 2.93 GHz CPU

Conclusion

- new deterministic and efficient method for point pattern matching
- robust against different errors, e.g., trajectory deformation
- geometric configurations of observed landmarks are unique, i.e., they allow us to unambiguously determine our location

Outlook

- tests with low-cost sensors
- consider more objects than poles, i.e., other point features, planes, road markings

Outlook