Geometry SLAM from lidar and camera

Karel Zimmermann
http://cmp.felk.cvut.cz/~zimmerk/

Vision for Robotics and Autonomous Systems
https://cyber.felk.cvut.cz/vras/

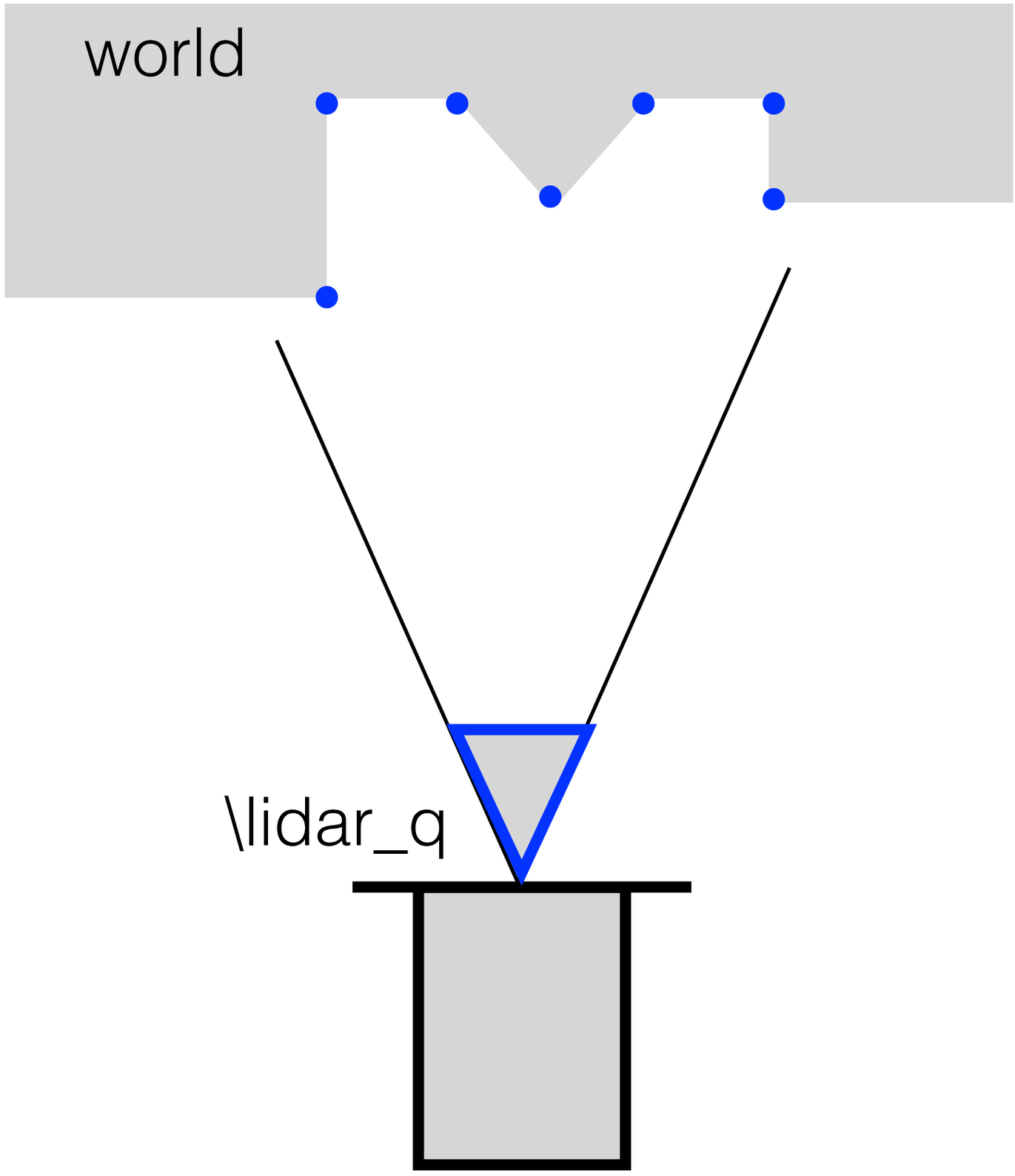
Center for Machine Perception
https://cmp.felk.cvut.cz

Department for Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague

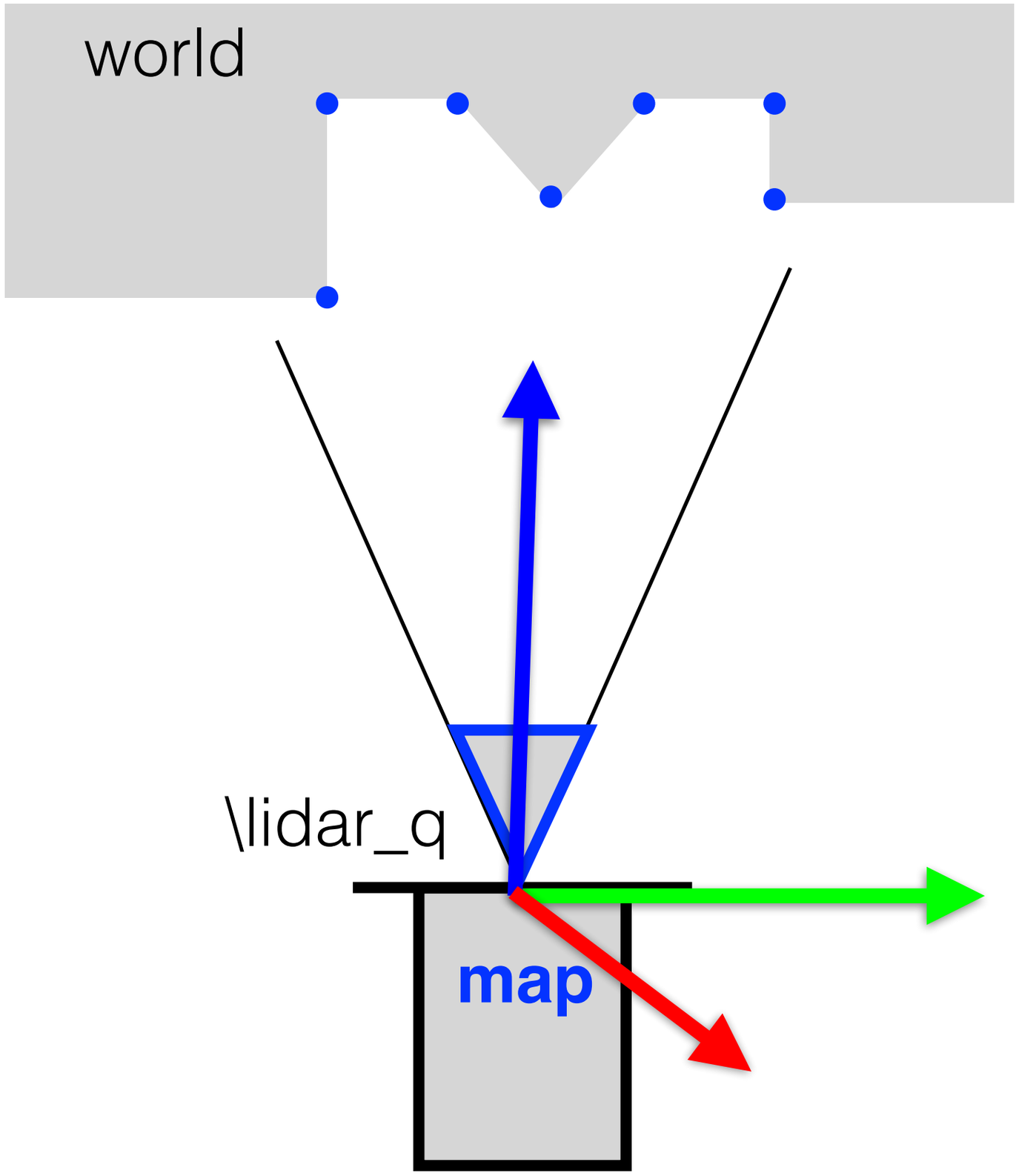
Motivation

- Motion of the robot cause Euclidean transformation of its c.f.
- Motion estimated from interoceptive sensors is inaccurate (odometry wheel slipping, GPS-denied environment)
- Errors are often compensated from exteroceptive sensors (camera and lidar)

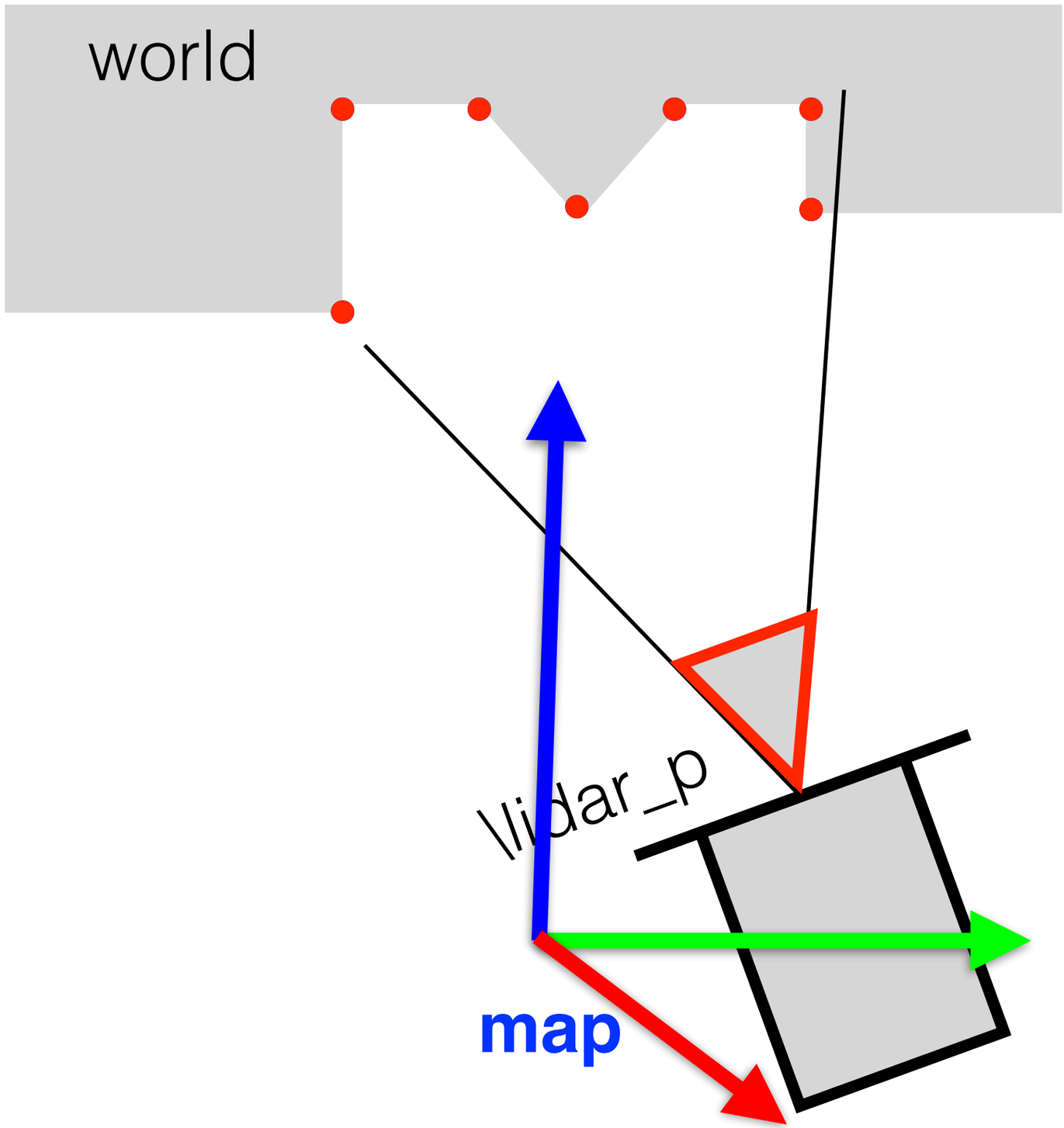
Motivation



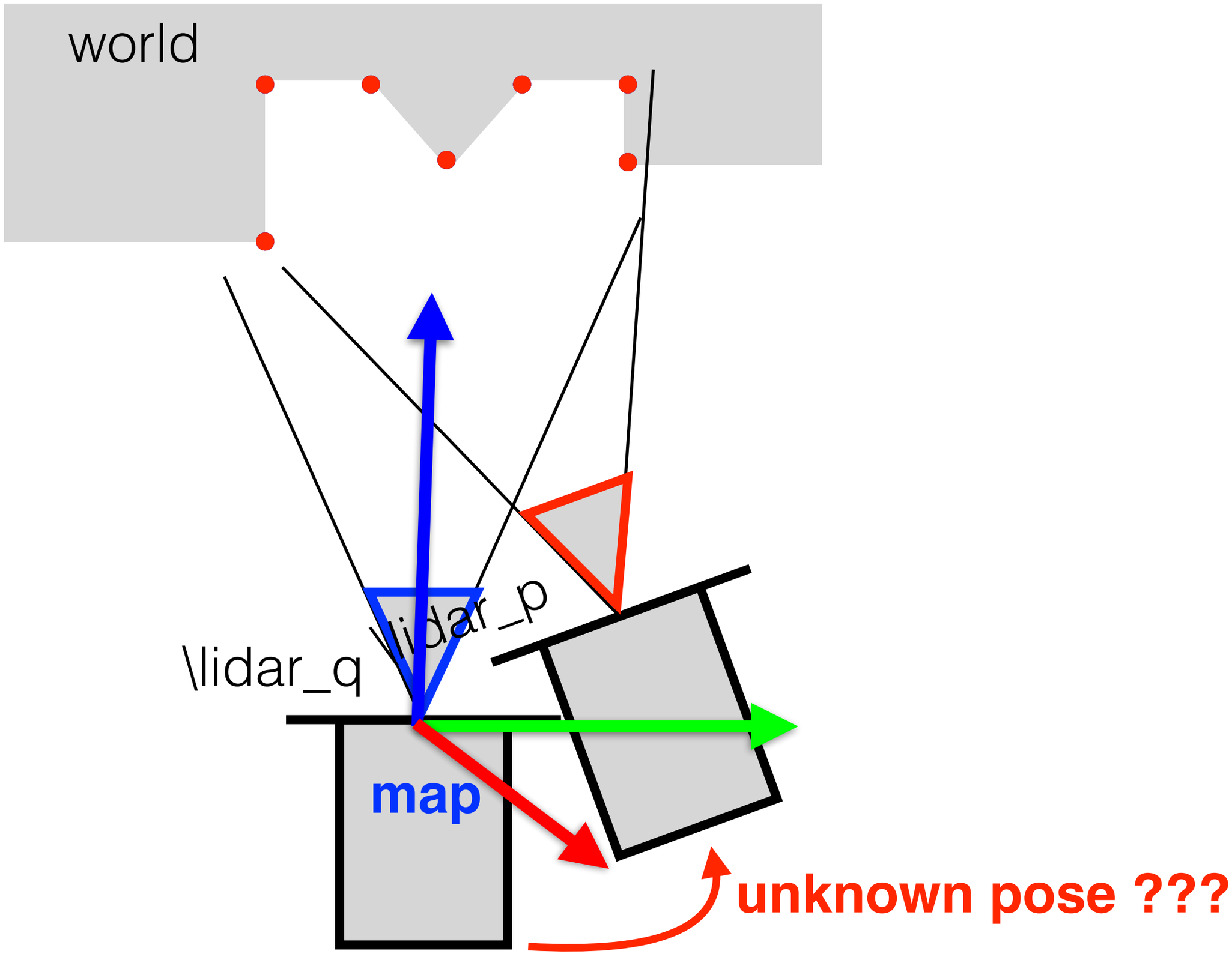
Motivation



Motivation

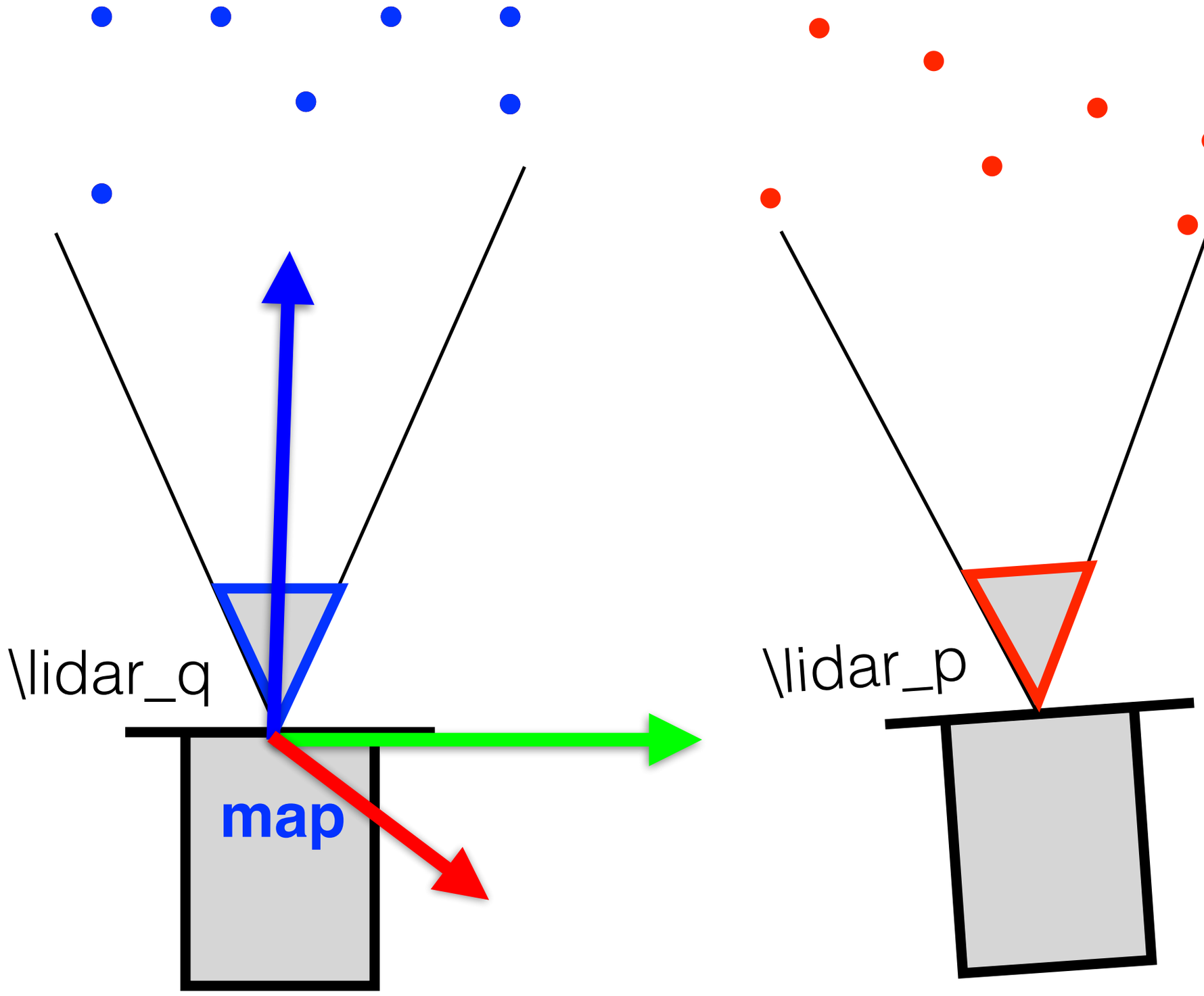


Motivation



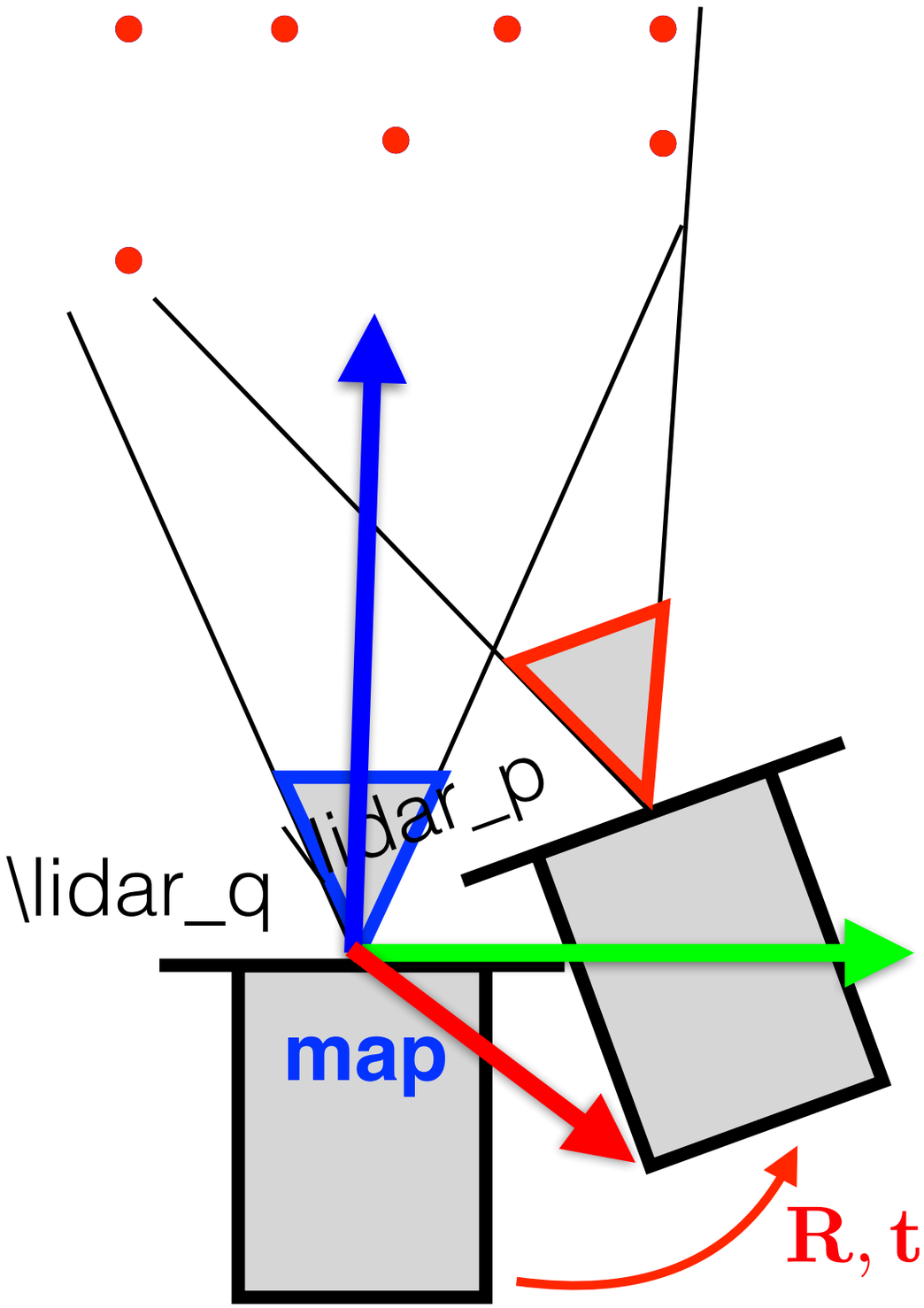
Motivation

- given these two pointclouds, what is the motion of the robot?



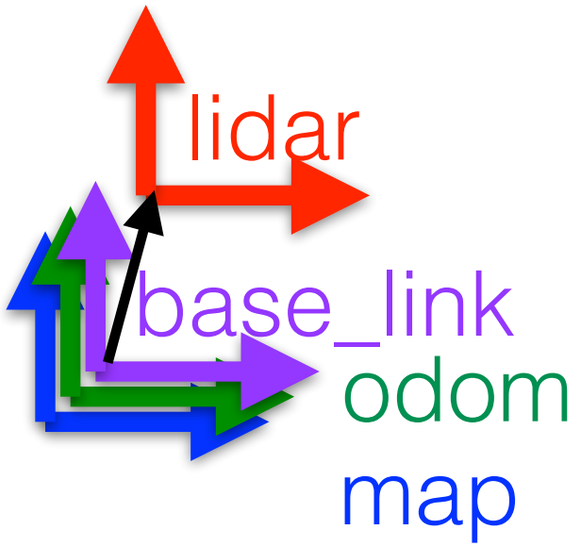
Motivation

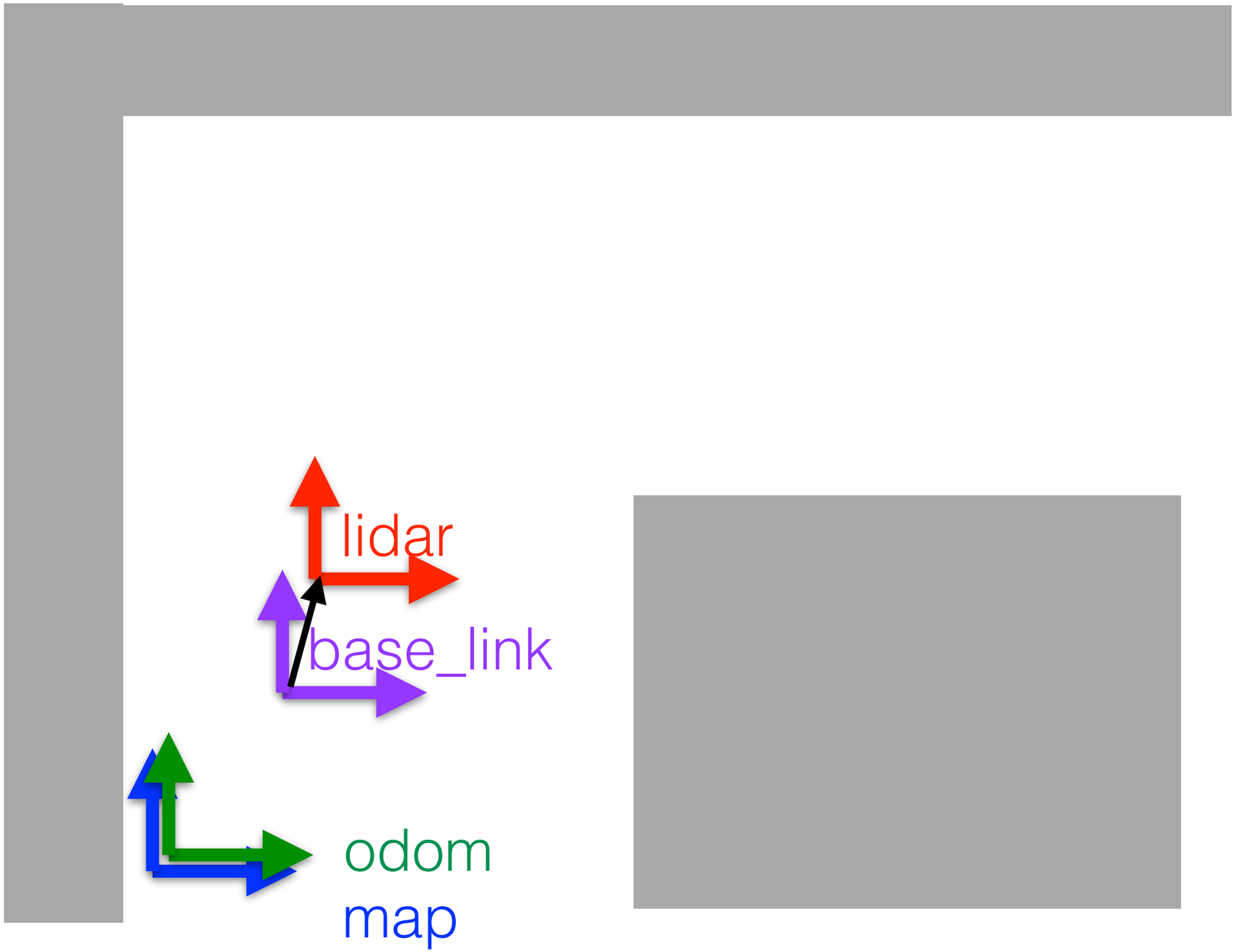
- the motion, which aligns correspondent points from PCLs

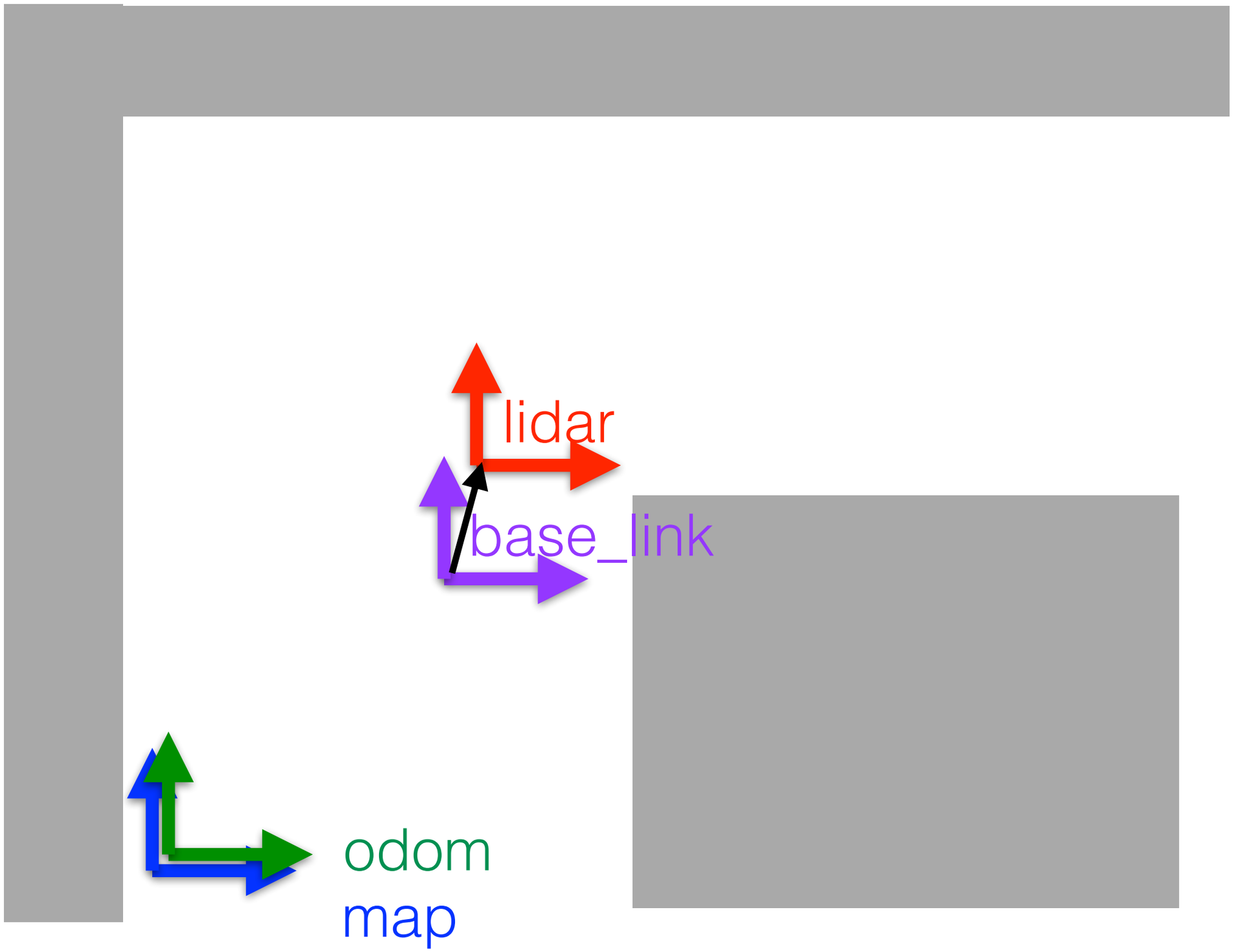


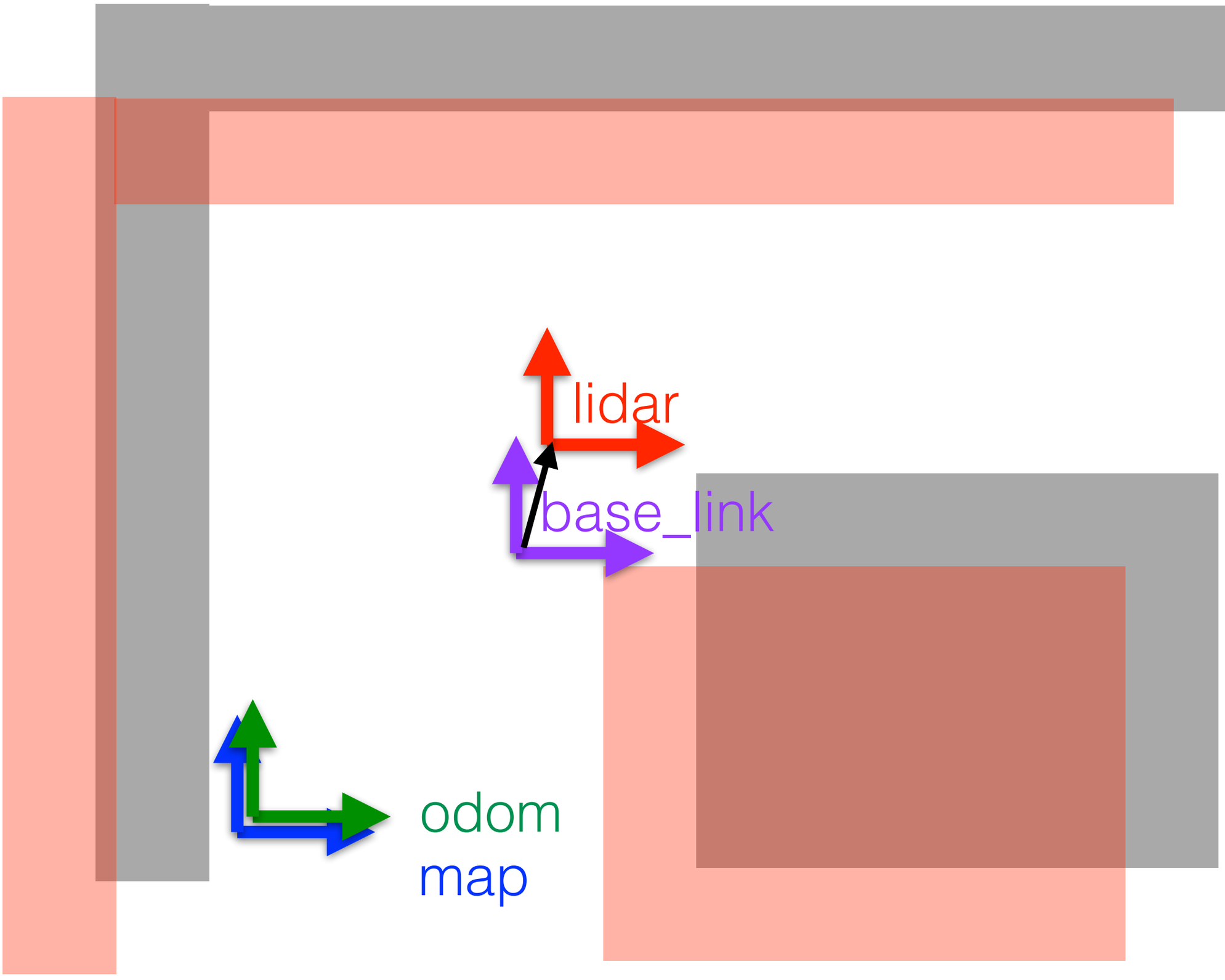
Motivation

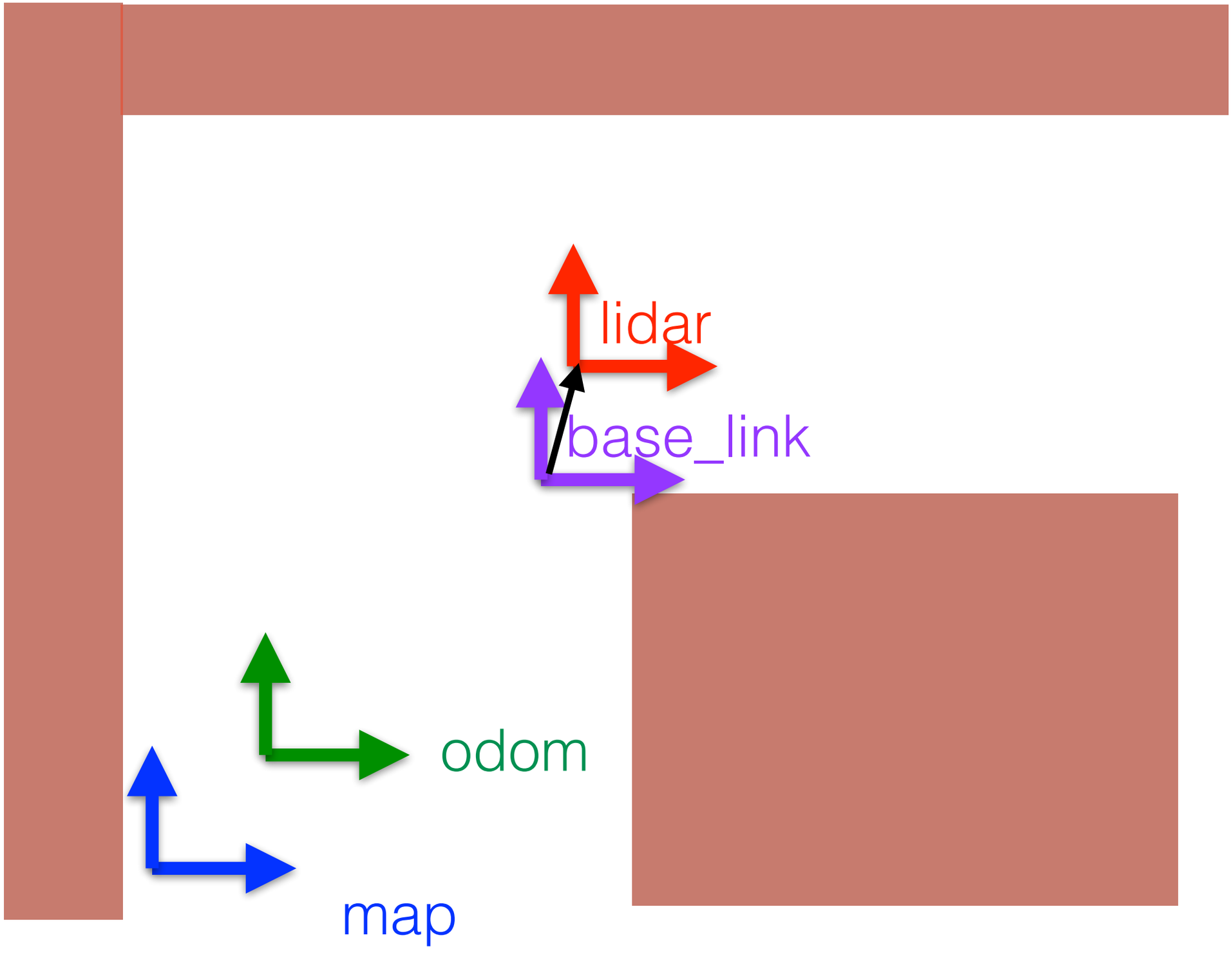
- the task is same as calibration of two lidars, but 3D-3D correspondences cannot be provided manually.
- autonomous search for 3D-3D correspondences needed.

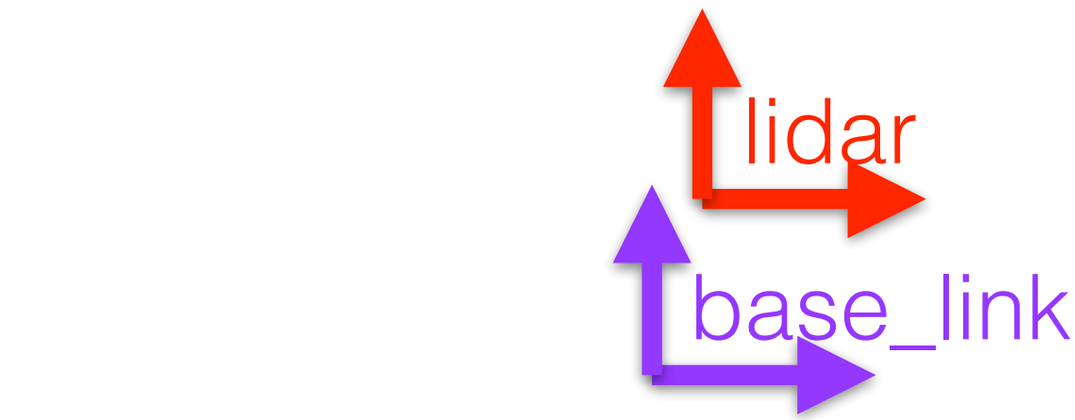

lidar
base_link
odom
map

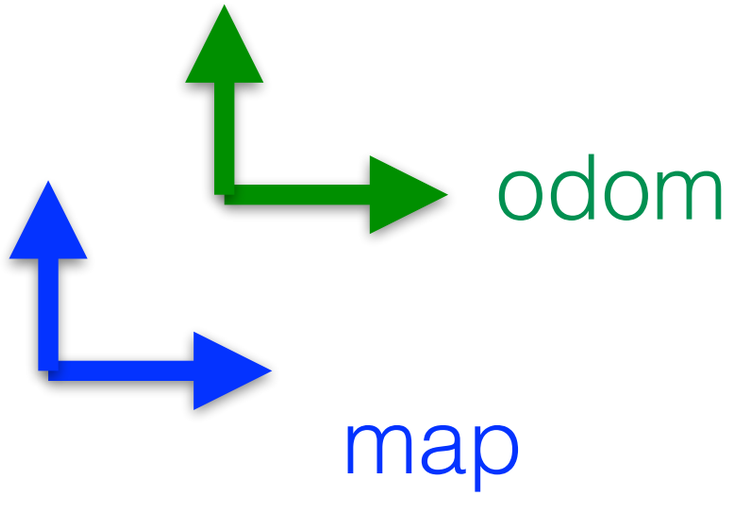

lidar
base_link
odom
map


lidar
base_link
odom
map


lidar
base_link
odom
map

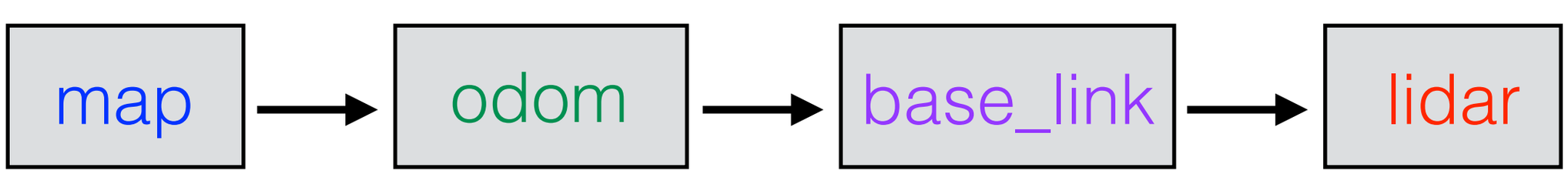

lidar
base_link
odom
map


lidar
base_link


odom
map

SLAM coordinate frames in ROS

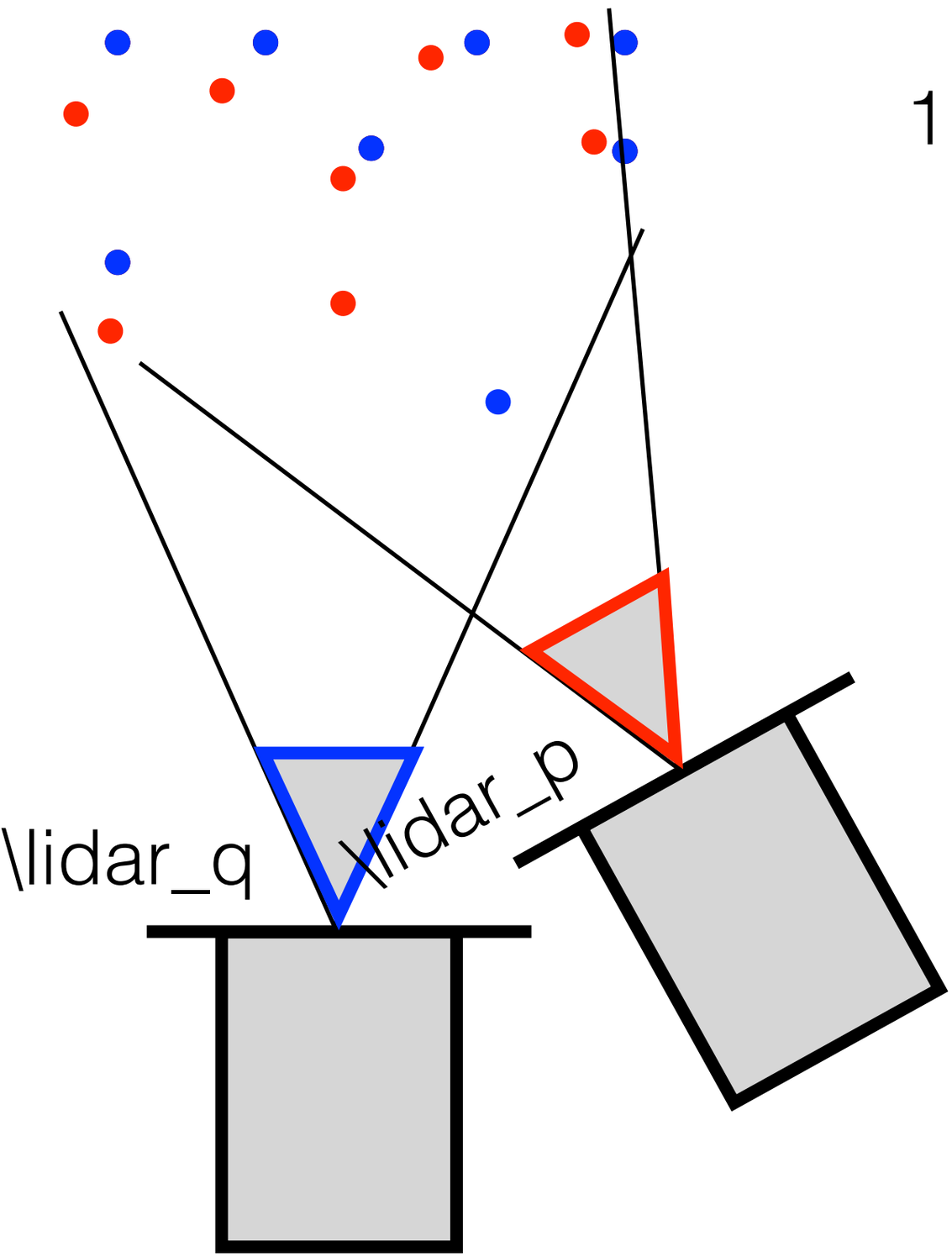
- This lecture is about estimating map to odom transformation
- We assume that the lidar-measured pointcloud is aligned with the map



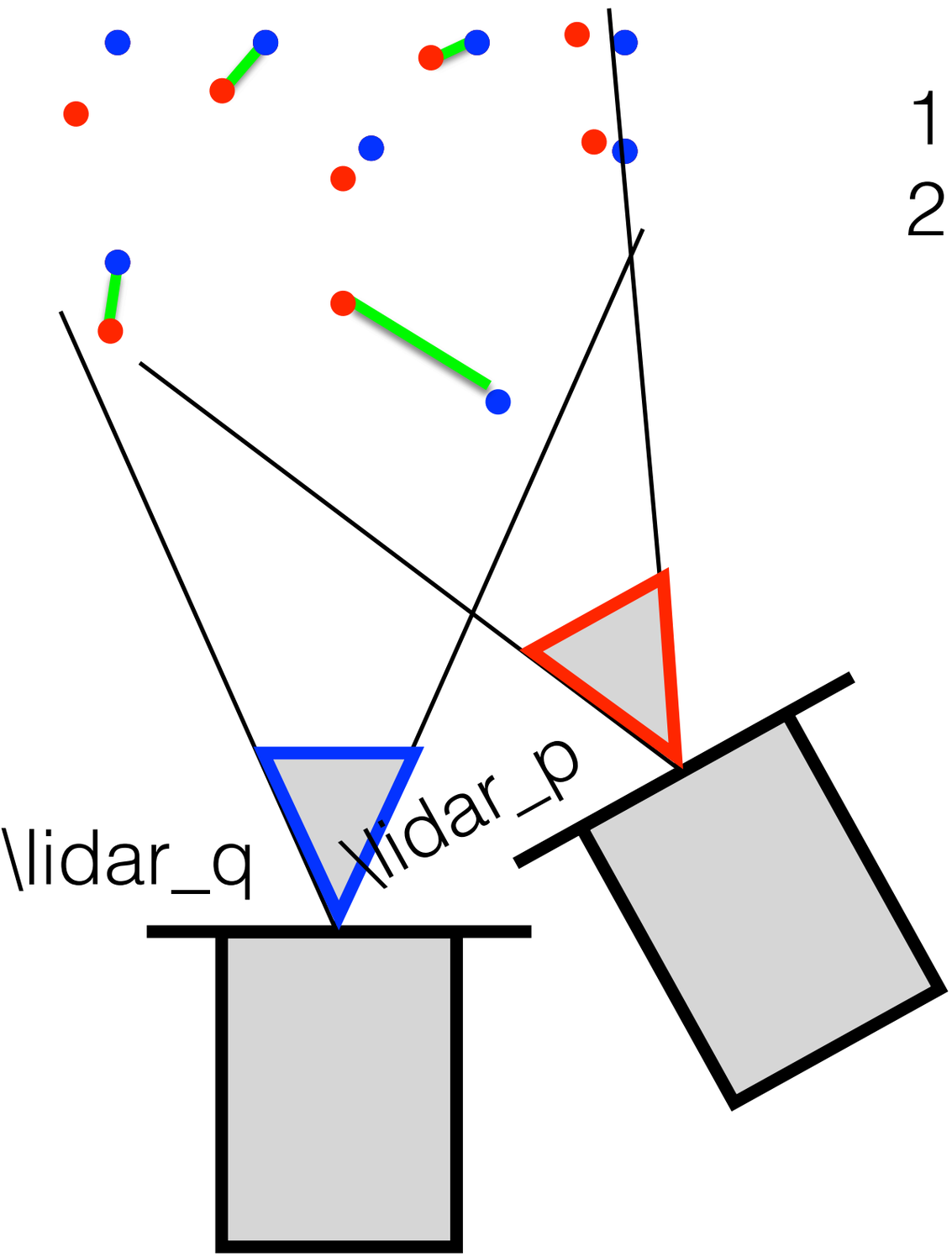
Iterative Closest Point (ICP) [Besl and McKay 92]

Input:
- two pointclouds P and Q
- scan P best aligned with map Q

1. Initialize $\mathbf{R} = \mathbb{I},\ \mathbf{t} = \mathbf{0}$



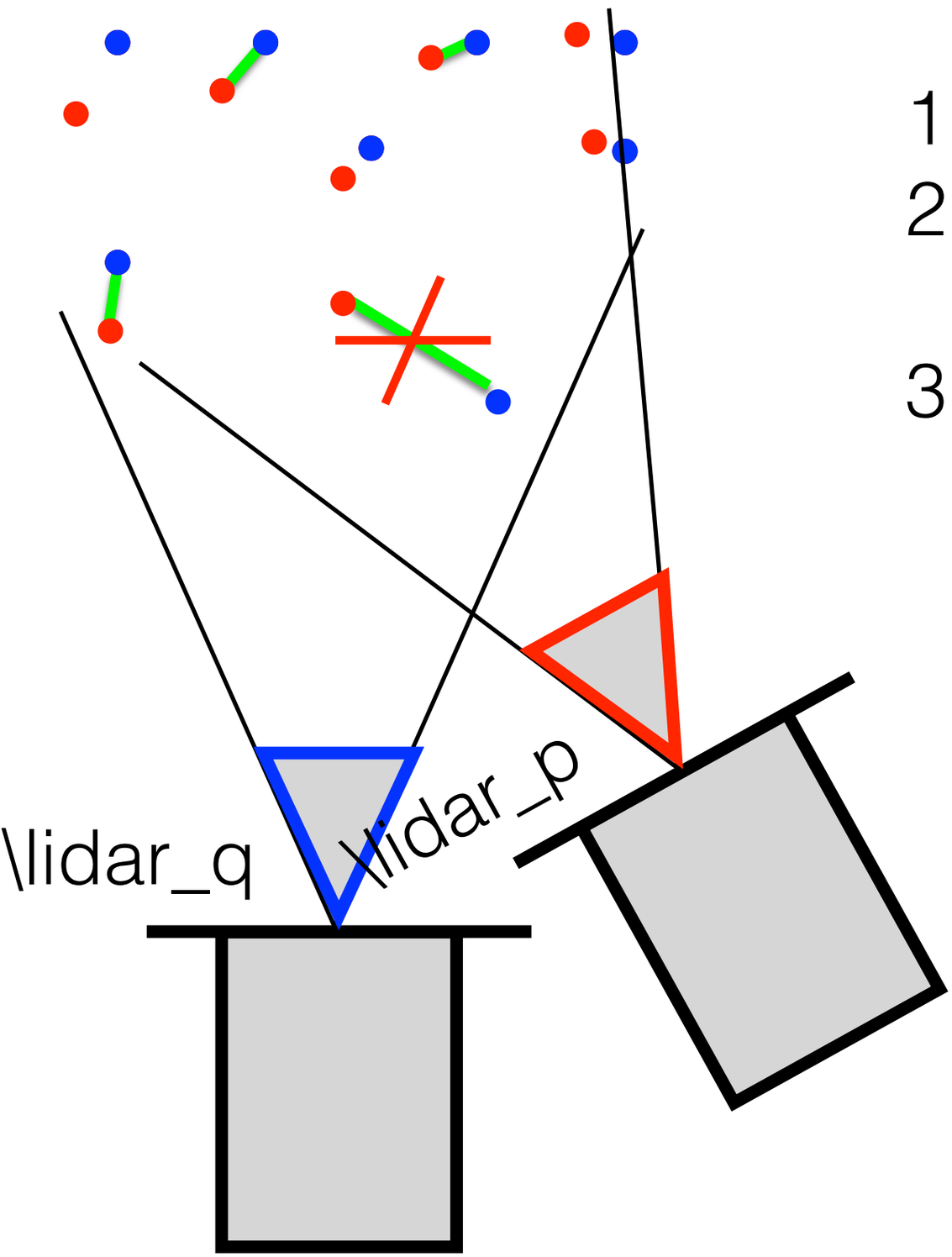
Iterative Closest Point (ICP) [Besl and McKay 92]



Input:
- two pointclouds P and Q
- scan P best aligned with map Q

1. Initialize $\mathbf{R} = \mathbb{I},\ \mathbf{t} = \mathbf{0}$
2. For each point $\mathbf{p}_i$ find closest point $\mathbf{q}_i$ from Q

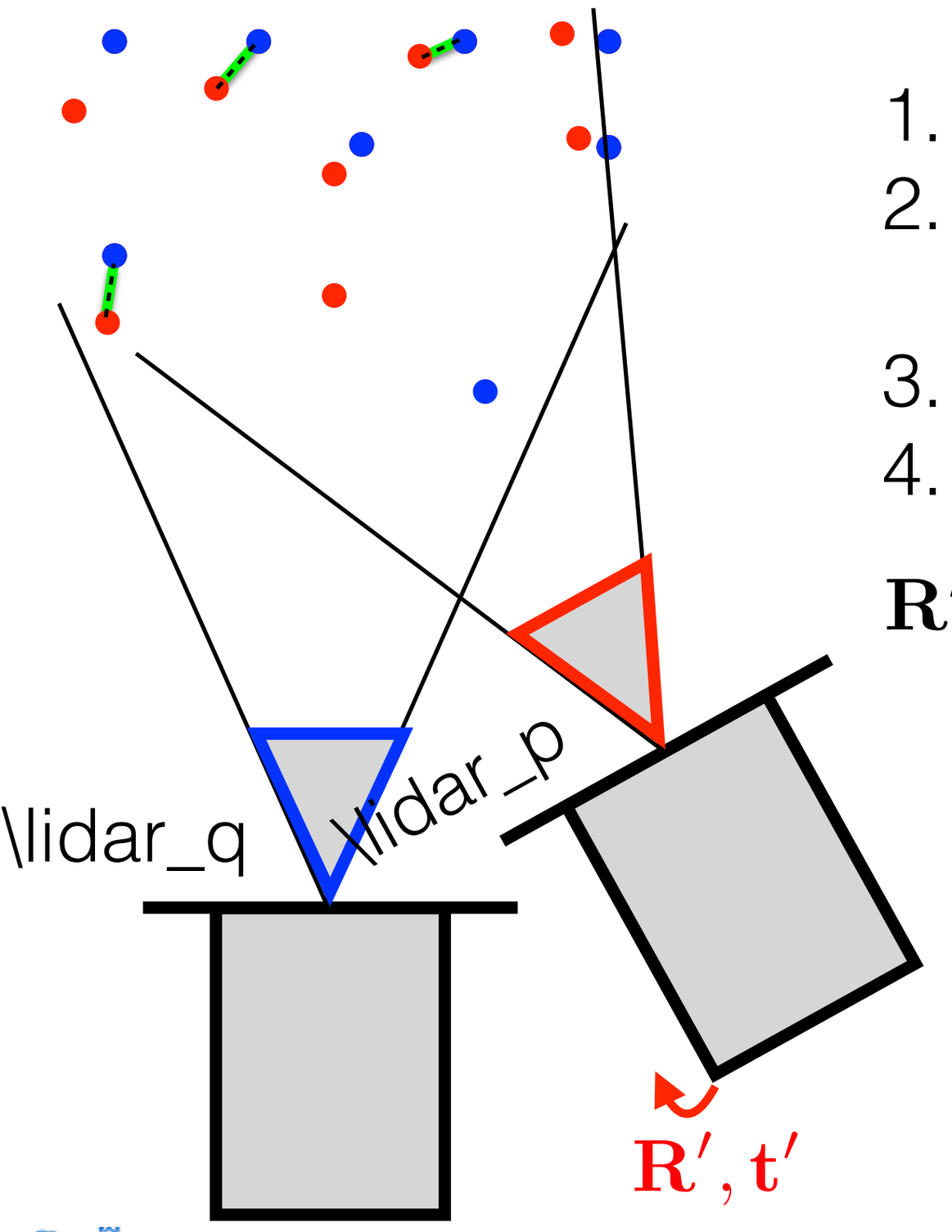
Iterative Closest Point (ICP) [Besl and McKay 92]



Input:
- two pointclouds P and Q
- scan P best aligned with map Q

1. Initialize $\mathbf{R} = \mathbb{I},\ \mathbf{t} = \mathbf{0}$
2. For each point $\mathbf{p}_i$ find closest point $\mathbf{q}_i$ from Q
3. Reject outliers by median thresh.

Iterative Closest Point (ICP) [Besl and McKay 92]

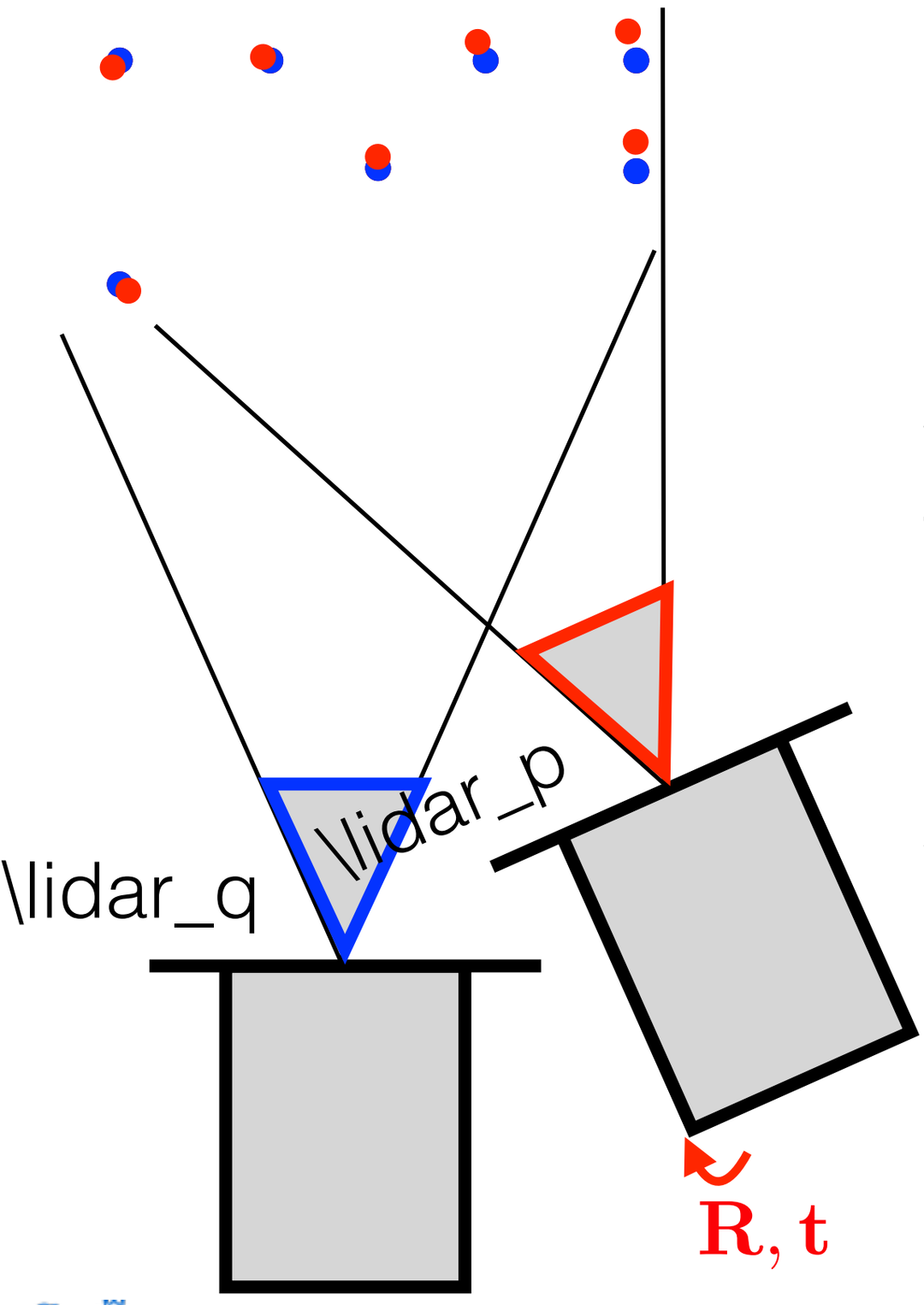


Input:
- two pointclouds P and Q
- scan P best aligned with map Q

1. Initialize $\mathbf{R} = \mathbb{I},\ \mathbf{t} = \mathbf{0}$
2. For each point $\mathbf{p}_i$ find closest point $\mathbf{q}_i$ from Q
3. Reject outliers by median thresh.
4. Solve [Arun-TPAMI-87]:

$$\mathbf{R}', \mathbf{t}' = \underset{\mathbf{R}' \in SO(3), \mathbf{t}' \in \mathcal{R}^3}{\arg\min} \sum \|\mathbf{R}'\mathbf{p}_i + \mathbf{t}' - \mathbf{q}_i\|_2^2$$

Iterative Closest Point (ICP) [Besl and McKay 92]



Input:
- two pointclouds P and Q
- scan P best aligned with map Q

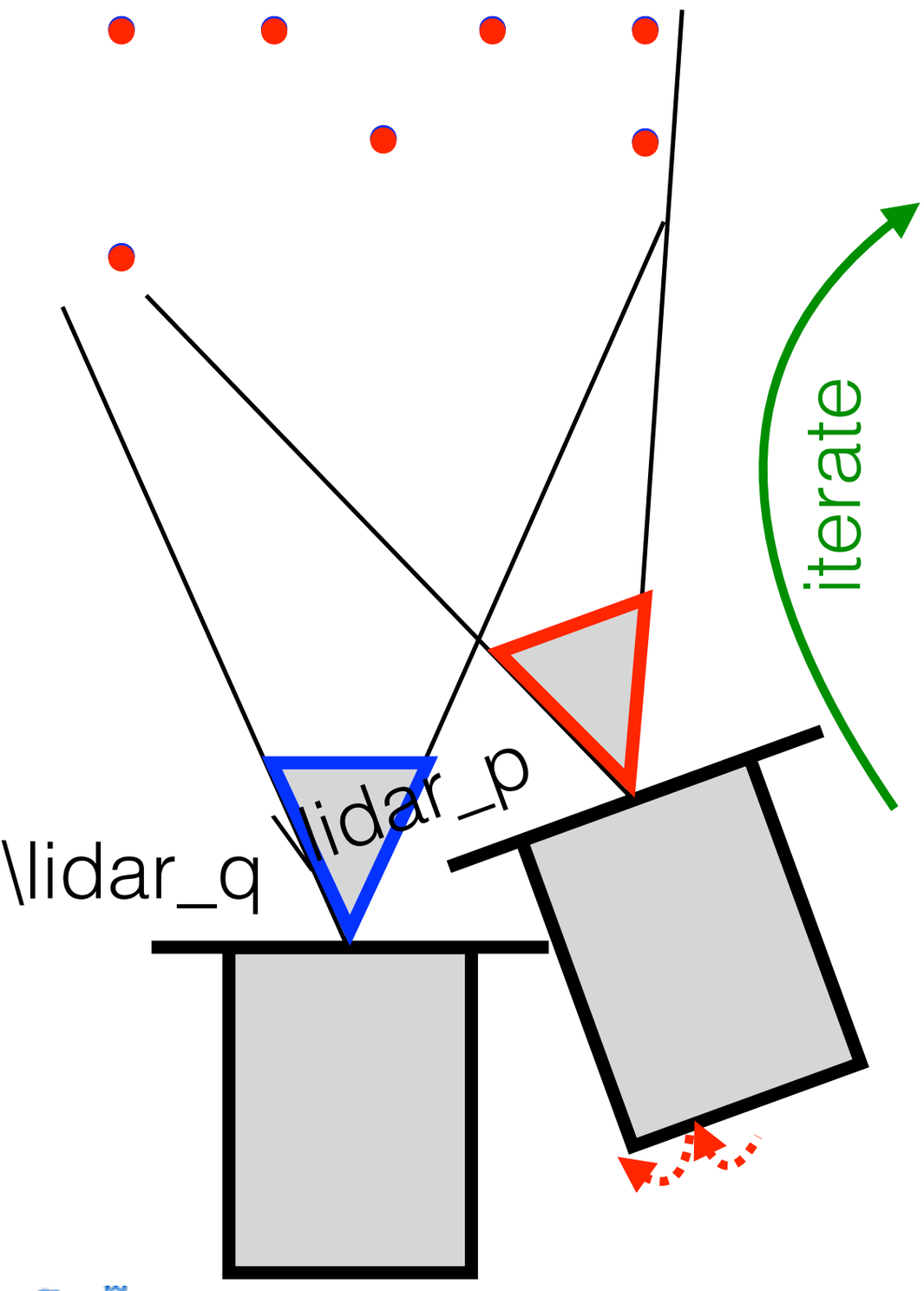
1. Initialize $\mathbf{R} = \mathbb{I},\ \mathbf{t} = \mathbf{0}$
2. For each point $\mathbf{p}_i$ find closest point $\mathbf{q}_i$ from Q
3. Reject outliers by median thresh.
4. Solve [Arun-TPAMI-87]:

$$\mathbf{R}', \mathbf{t}' = \arg\min_{\mathbf{R}' \in SO(3), \mathbf{t}' \in \mathcal{R}^3} \sum \|\mathbf{R}'\mathbf{p}_i + \mathbf{t}' - \mathbf{q}_i\|_2^2$$

5. Update P + robot position:

$$\begin{aligned} \mathbf{p}_i &:= \mathbf{R}'\mathbf{p}_i + \mathbf{t}' \quad \forall_i \\ \mathbf{R} &:= \mathbf{R}'\mathbf{R} \\ \mathbf{t} &:= \mathbf{R}'\mathbf{t} + \mathbf{t}' \end{aligned}$$

Iterative Closest Point (ICP) [Besl and McKay 92]



Input:
- two pointclouds P and Q
- scan P best aligned with map Q

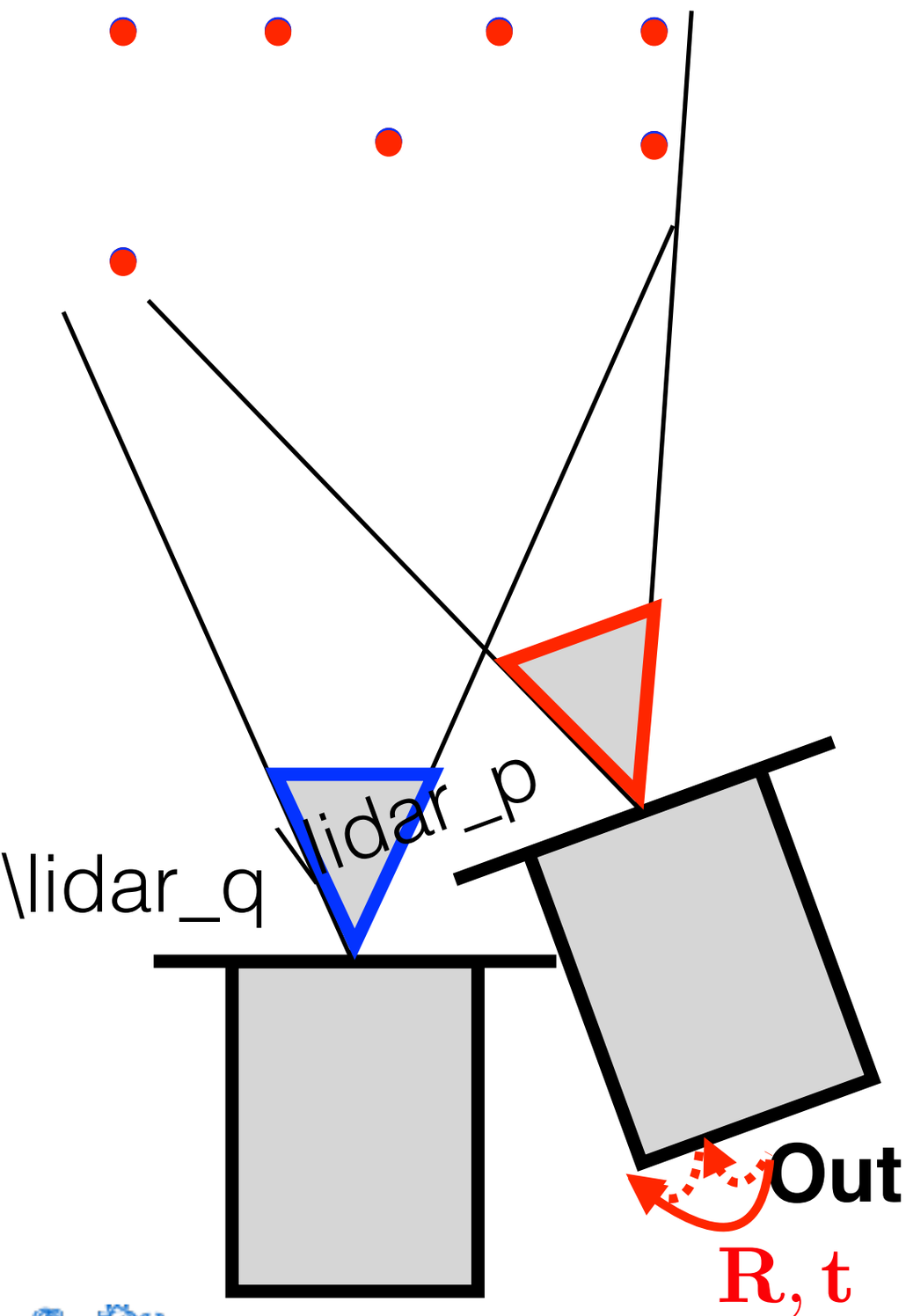
1. Initialize $\mathbf{R} = \mathbb{I},\ \mathbf{t} = \mathbf{0}$
2. For each point $\mathbf{p}_i$ find closest point $\mathbf{q}_i$ from Q
3. Reject outliers by median thresh.
4. Solve [Arun-TPAMI-87]:

$$\mathbf{R}', \mathbf{t}' = \arg\min_{\mathbf{R}' \in SO(3), \mathbf{t}' \in \mathcal{R}^3} \sum \|\mathbf{R}'\mathbf{p}_i + \mathbf{t}' - \mathbf{q}_i\|_2^2$$

5. Update P + robot position:

$$\begin{aligned} \mathbf{p}_i &:= \mathbf{R}'\mathbf{p}_i + \mathbf{t}' \quad \forall_i \\ \mathbf{R} &:= \mathbf{R}'\mathbf{R} \\ \mathbf{t} &:= \mathbf{R}'\mathbf{t} + \mathbf{t}' \end{aligned}$$

Iterative Closest Point (ICP) [Besl and McKay 92]



Input:
- two pointclouds P and Q
- scan P best aligned with map Q

1. Initialize $\mathbf{R} = \mathbb{I},\ \mathbf{t} = \mathbf{0}$
2. For each point $\mathbf{p}_i$ find closest point $\mathbf{q}_i$ from Q
3. Reject outliers by median thresh.
4. Solve [Arun-TPAMI-87]:

$$\mathbf{R}', \mathbf{t}' = \arg\min_{\mathbf{R}' \in SO(3), \mathbf{t}' \in \mathcal{R}^3} \sum \|\mathbf{R}'\mathbf{p}_i + \mathbf{t}' - \mathbf{q}_i\|_2^2$$

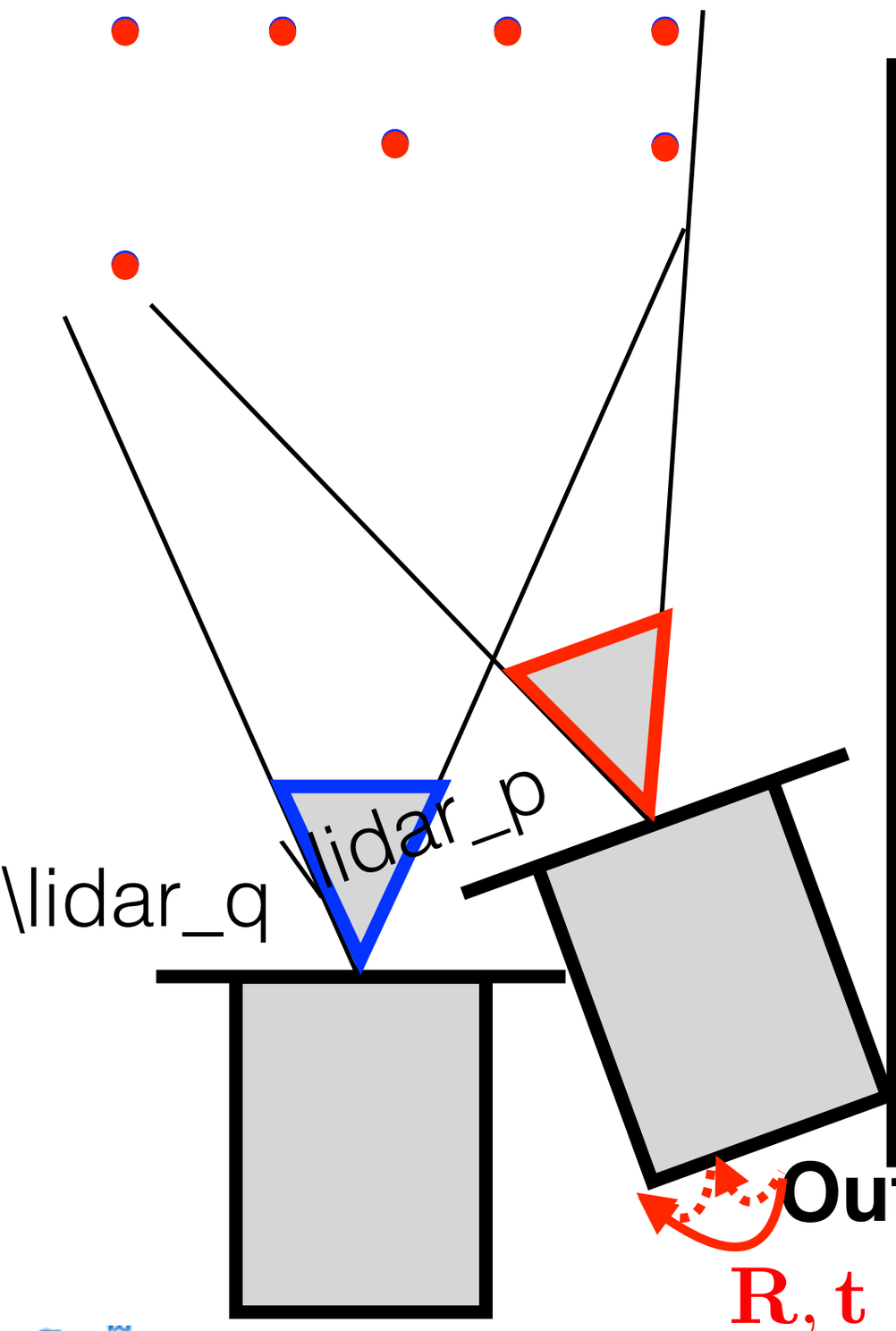
5. Update P + robot position:

$$\begin{aligned} \mathbf{p}_i &:= \mathbf{R}'\mathbf{p}_i + \mathbf{t}' \quad \forall_i \\ \mathbf{R} &:= \mathbf{R}'\mathbf{R} \\ \mathbf{t} &:= \mathbf{R}'\mathbf{t} + \mathbf{t}' \end{aligned}$$

Output: $\mathbf{R}, \mathbf{t}$ - correction of transformation induced by alignment of P wrt Q

Iterative Closest Point (ICP) [Besl and McKay 92]

Input:
- two pointclouds P and Q
- scan P best aligned with map Q



1. Initialize $\mathbf{R} = \mathbb{I},\ \mathbf{t} = \mathbf{0}$
2. For each point $\mathbf{p}_i$ find closest point $\mathbf{q}_i$ from Q
3. Reject outliers by median thresh.
4. Solve [Arun-TPAMI-87]:

$$\mathbf{R}', \mathbf{t}' = \underset{\mathbf{R}' \in SO(3), \mathbf{t}' \in \mathcal{R}^3}{\arg\min} \sum \|\mathbf{R}'\mathbf{p}_i + \mathbf{t}' - \mathbf{q}_i\|_2^2$$

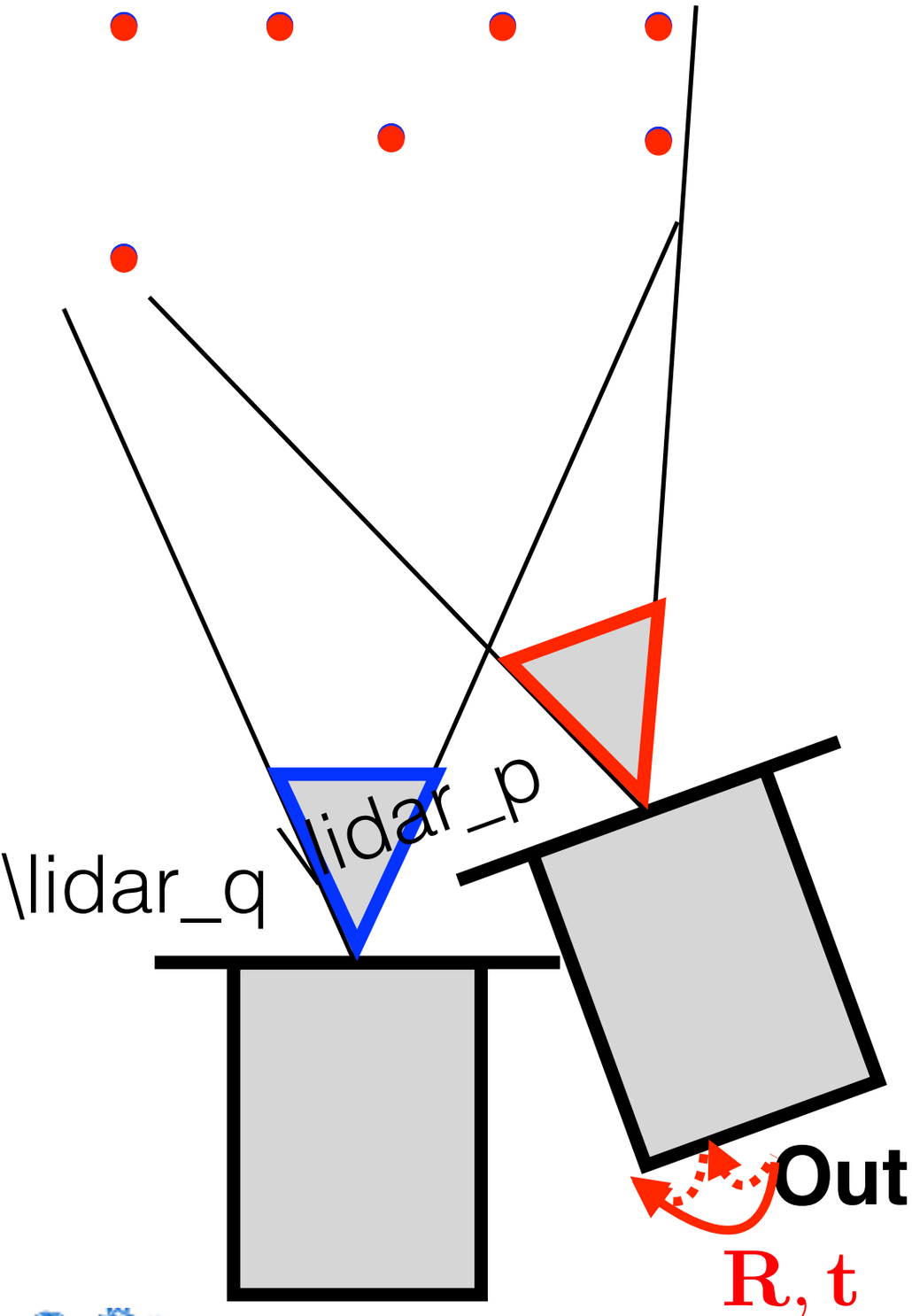
5. Update P + robot position:

$$\begin{aligned} \mathbf{p}_i &:= \mathbf{R}'\mathbf{p}_i + \mathbf{t}' \quad \forall_i \\ \mathbf{R} &:= \mathbf{R}'\mathbf{R} \\ \mathbf{t} &:= \mathbf{R}'\mathbf{t} + \mathbf{t}' \end{aligned}$$

Output: $\mathbf{R}, \mathbf{t}$ - correction of transformation induced by alignment of P wrt Q

Iterative Closest Point (ICP) [Besl and McKay 92]

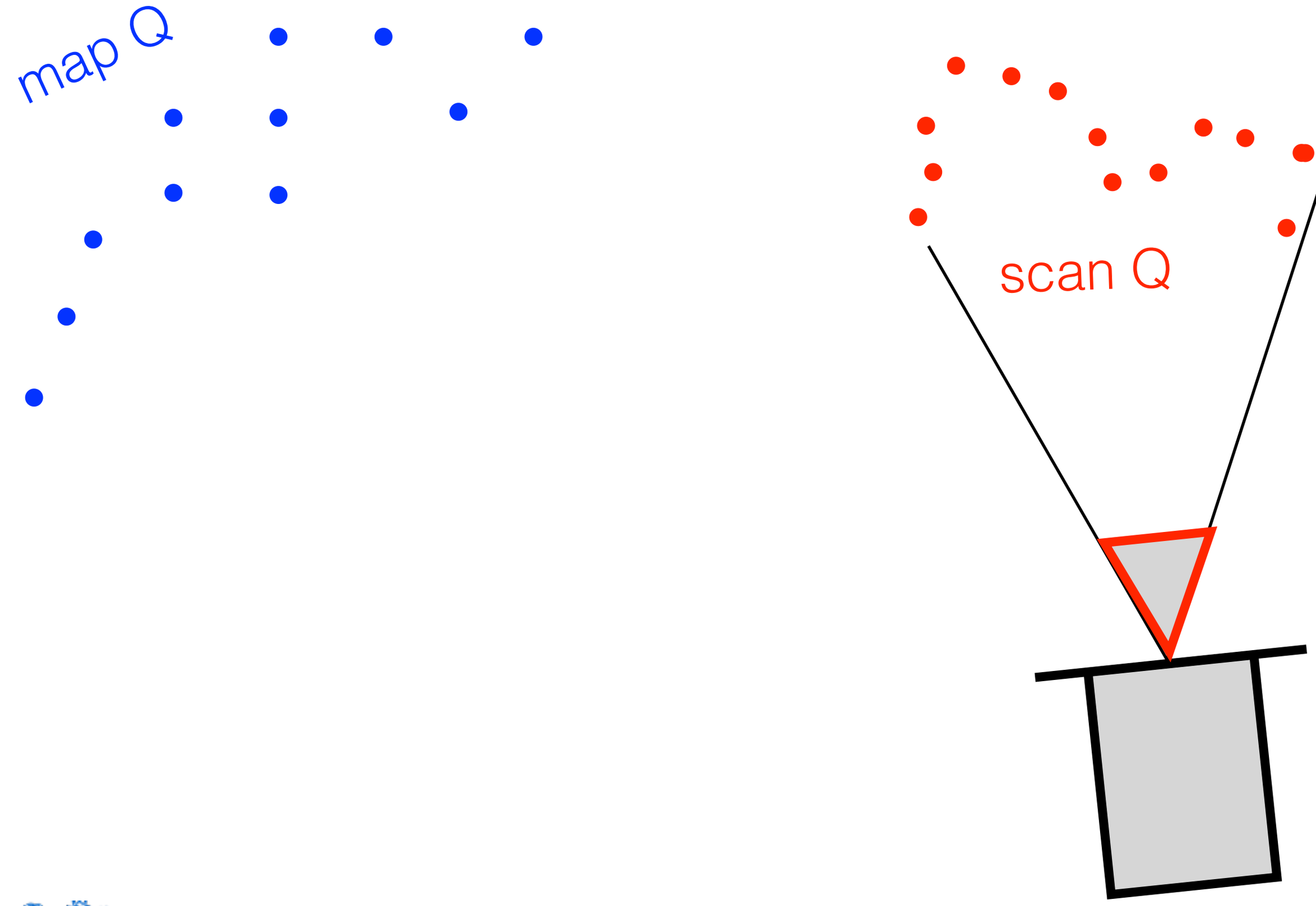
Input:
- two pointclouds P and Q
- scan P best aligned with map Q




```
R,t = ICP(P,Q)
```

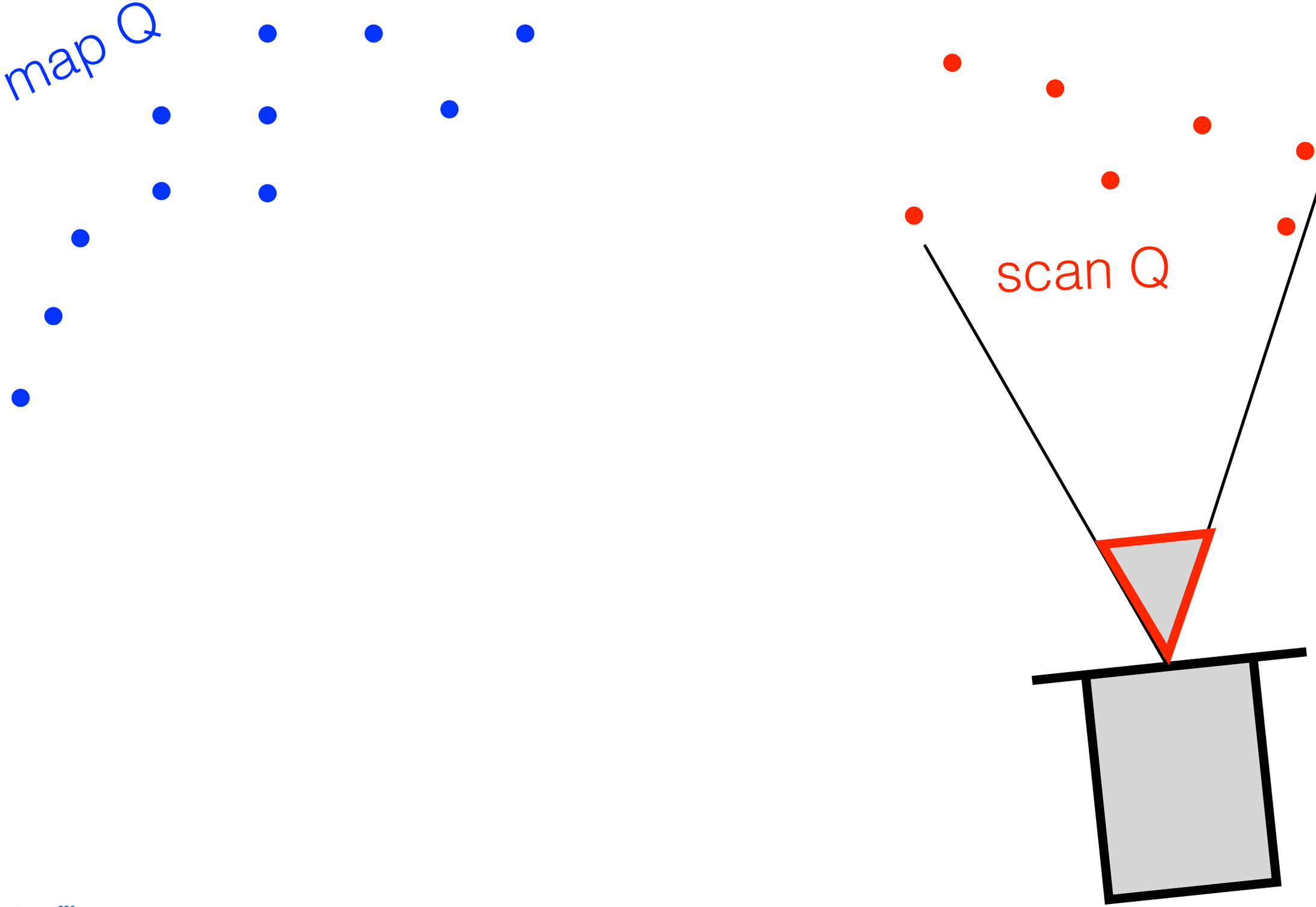
Output: $\mathbf{R}, \mathbf{t}$ - correction of transformation induced by alignment of P wrt Q

ICP SLAM

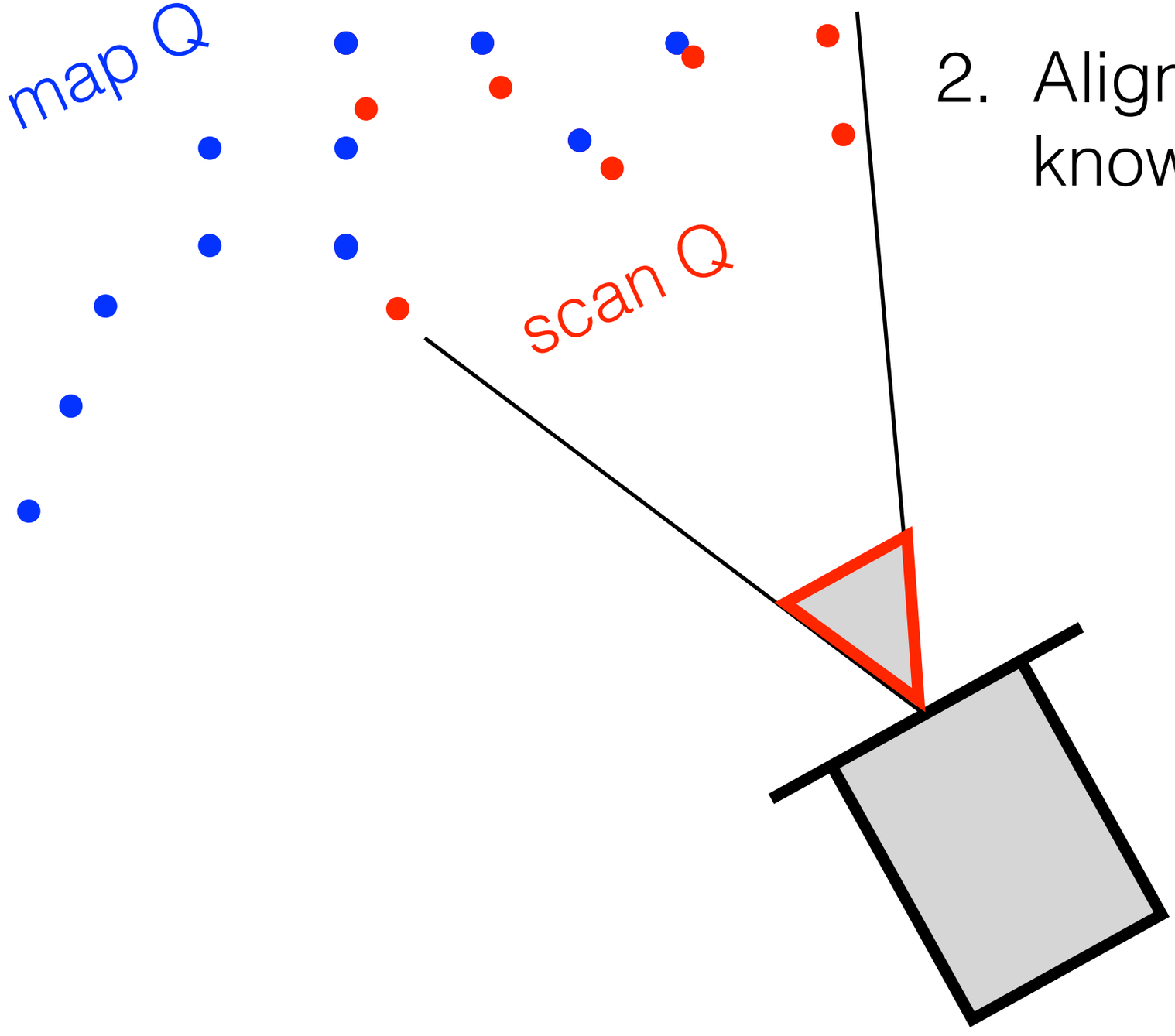


ICP SLAM

1. Filter P by uniform sampling

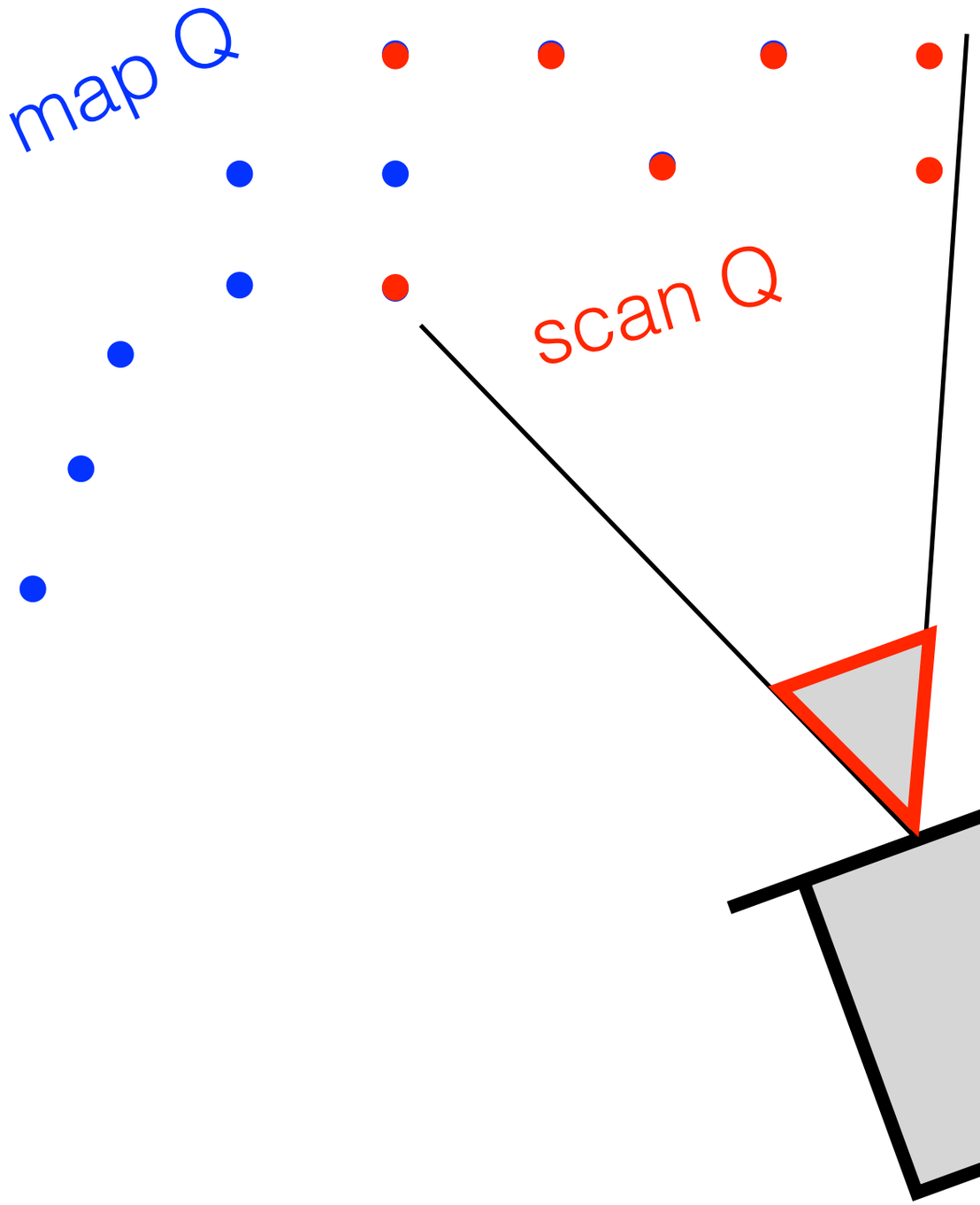


ICP SLAM



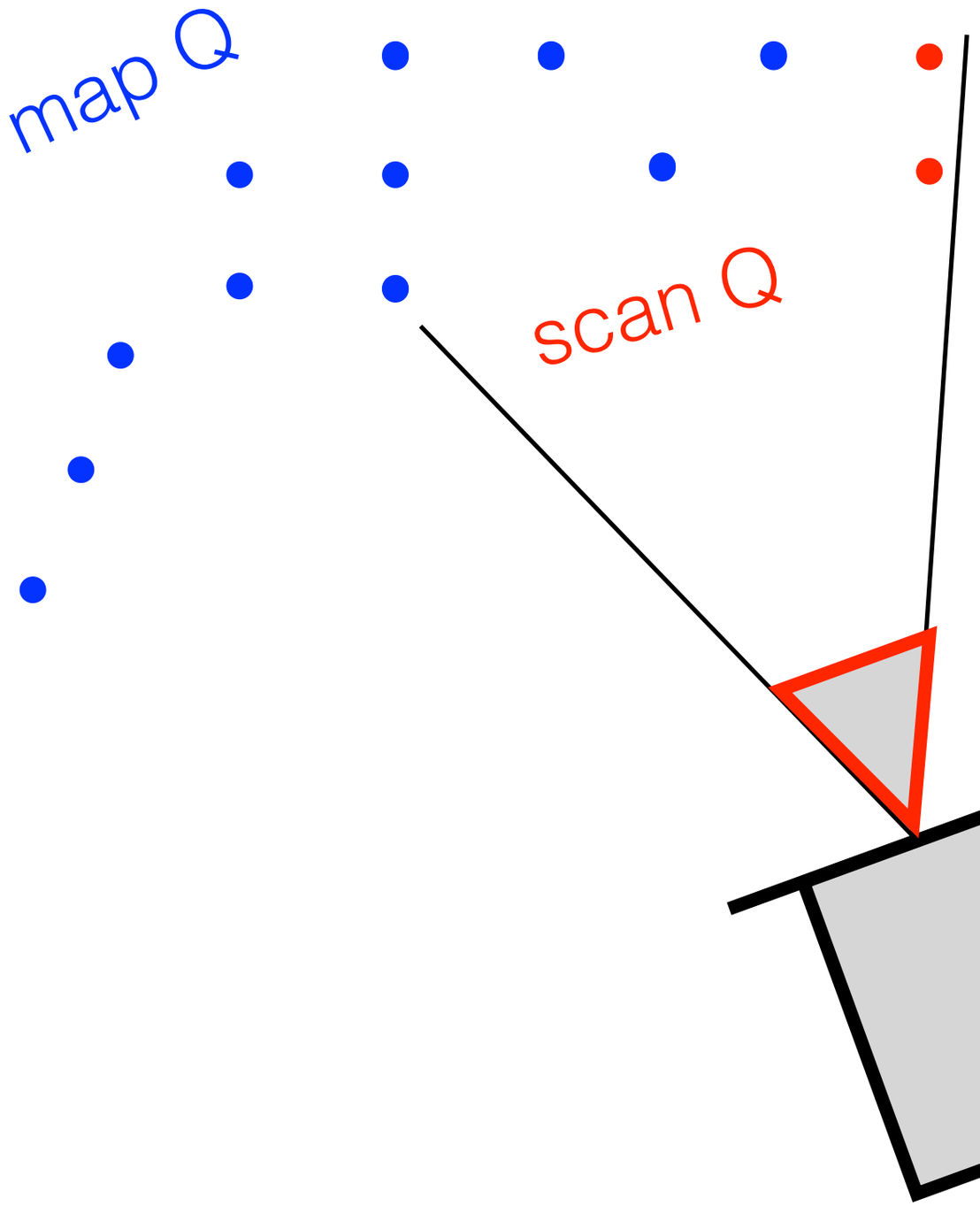
1. Filter P by uniform sampling
2. Align P with Q by the last known transformation

ICP SLAM



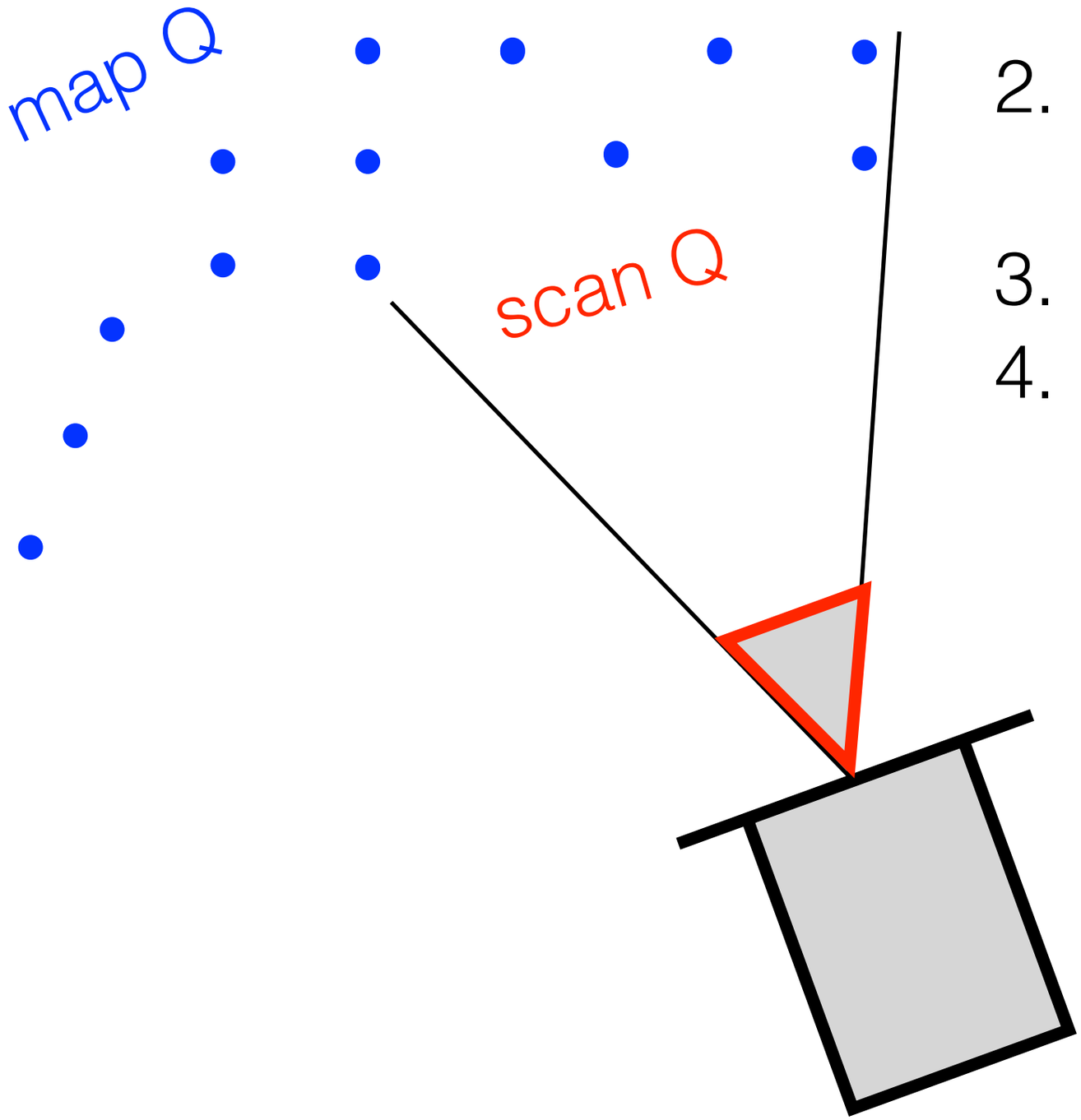
1. Filter P by uniform sampling
2. Align P with Q by the last known transformation
3. R,t = ICP(P,Q)

ICP SLAM



1. Filter P by uniform sampling
2. Align P with Q by the last known transformation
3. R,t = ICP(P,Q)
4. Update map

ICP SLAM

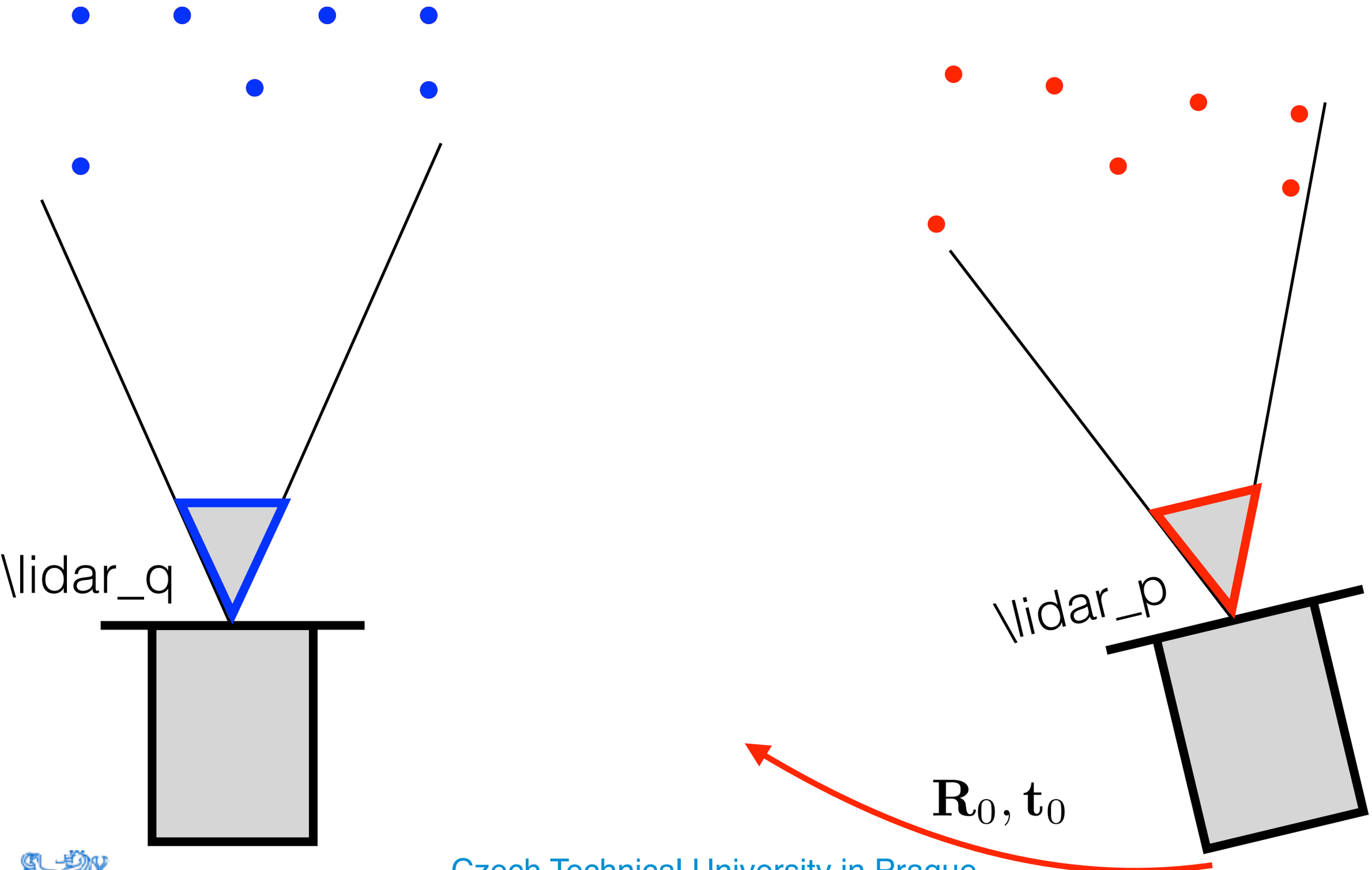


1. Filter P by uniform sampling
2. Align P with Q by the last known transformation
3. R,t = ICP(P,Q)
4. Update map

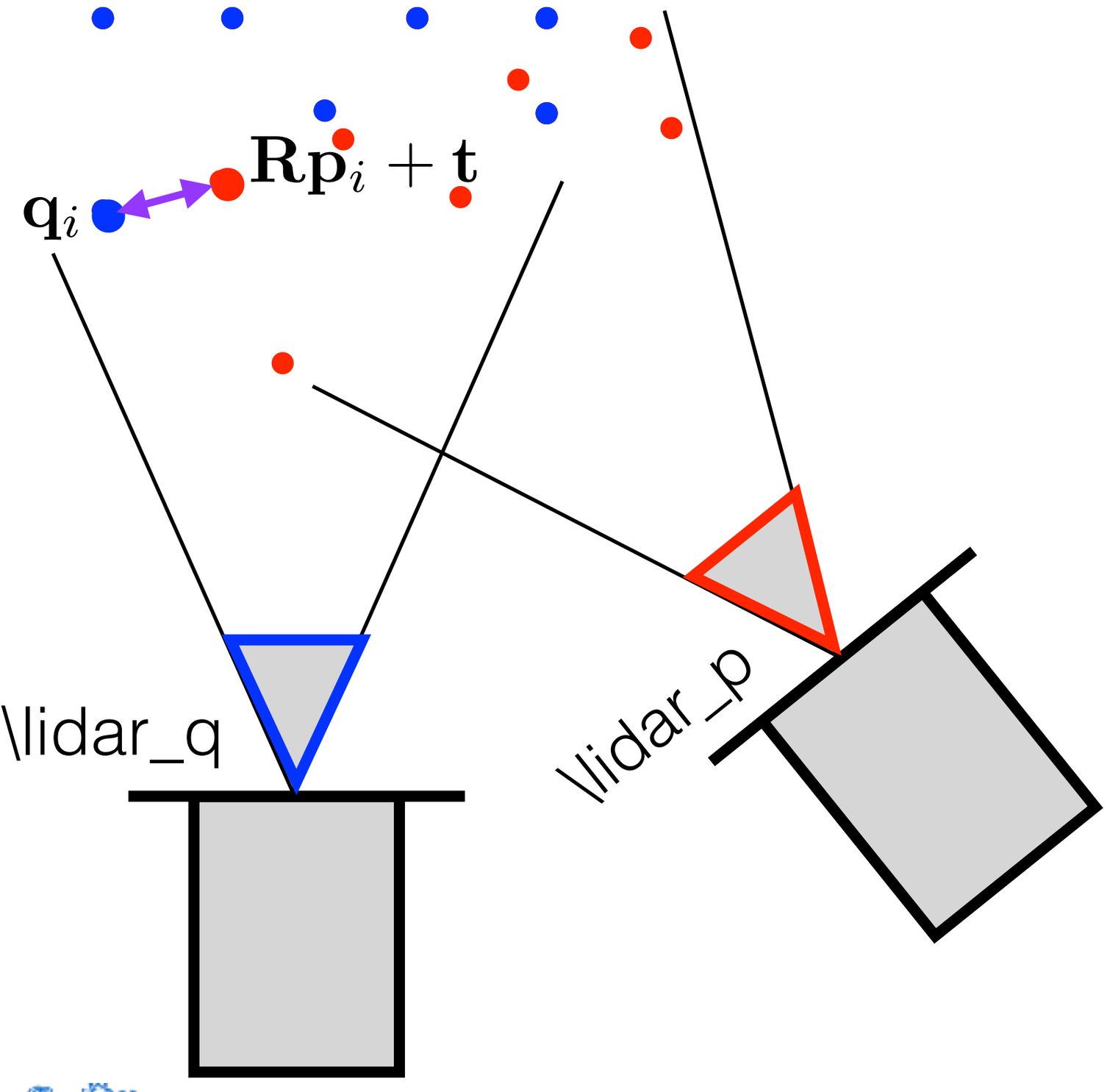
ICP SLAM - known issues

- Does not work in degenerate environments (e.g. forward motion in long hallways, or rotation in circular room)
- Converges to a local minima (unknown region of convergence => sensitive to good initialization)
- Sensitive to outliers due to L2 norm => RANSAC
- Large map requires huge amount of memory => better representation (e.g. surfel map)
- Some improvements discussed.

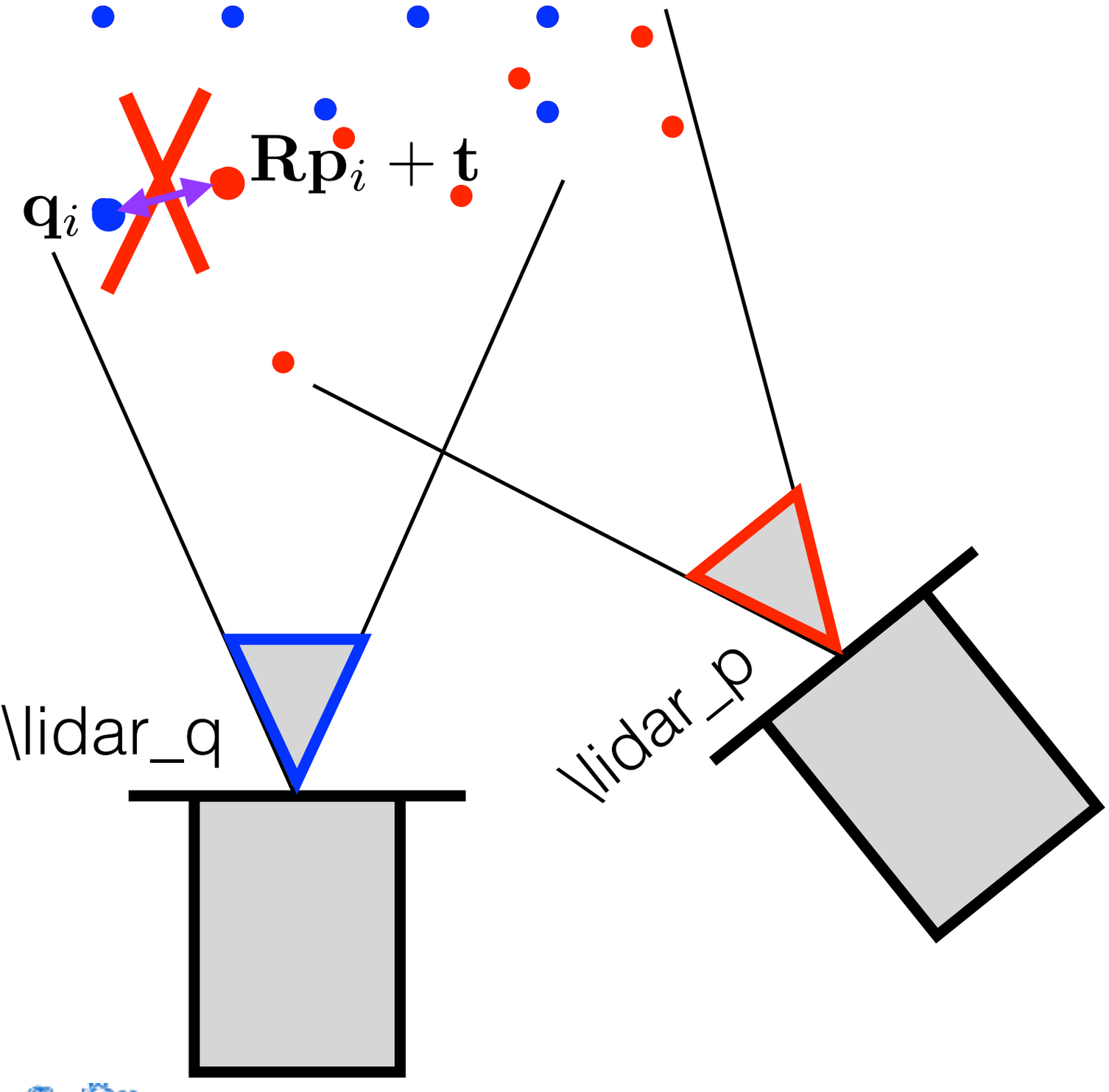
Iterative Closest Point (ICP) with point-to-plane distance



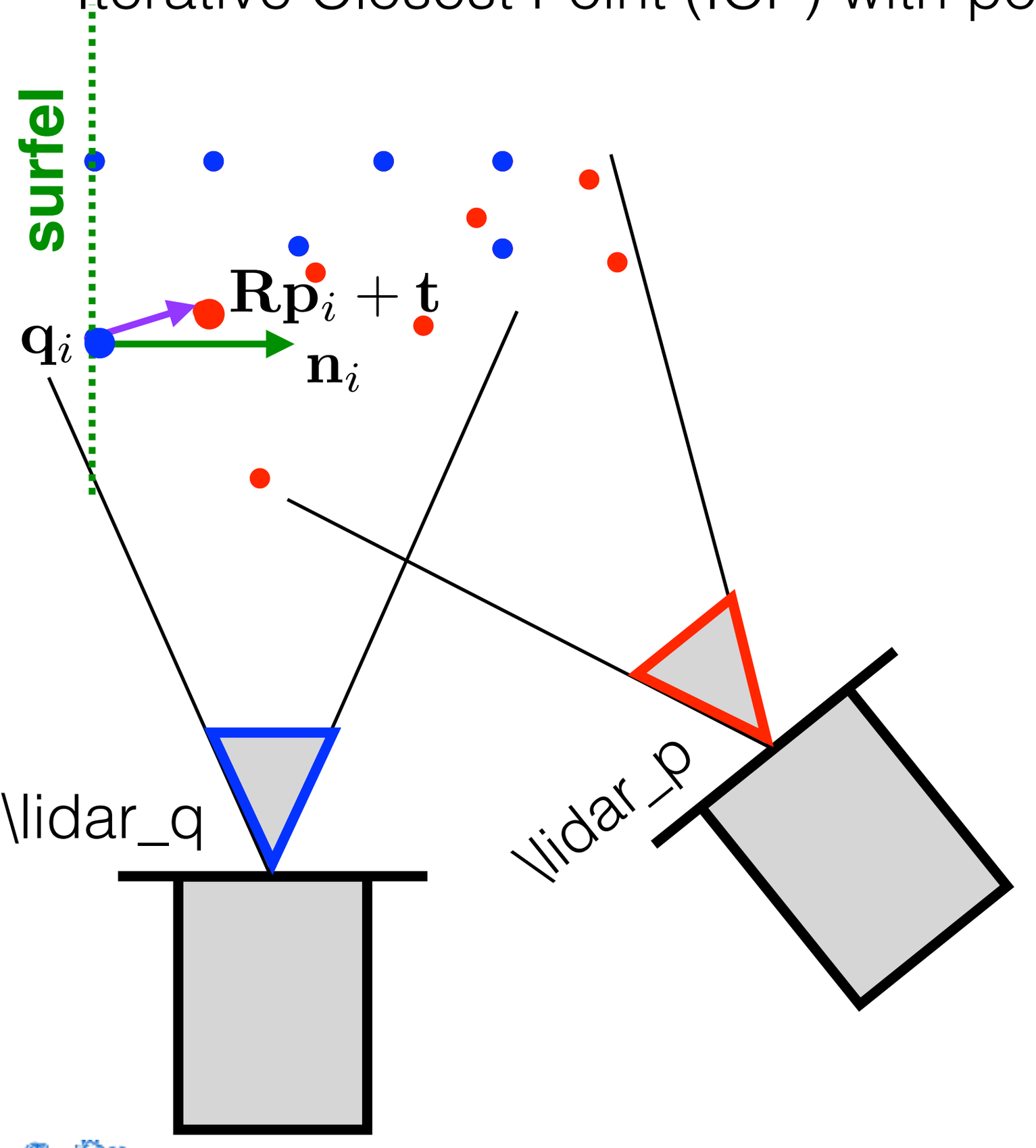
Iterative Closest Point (ICP) with point-to-plane distance



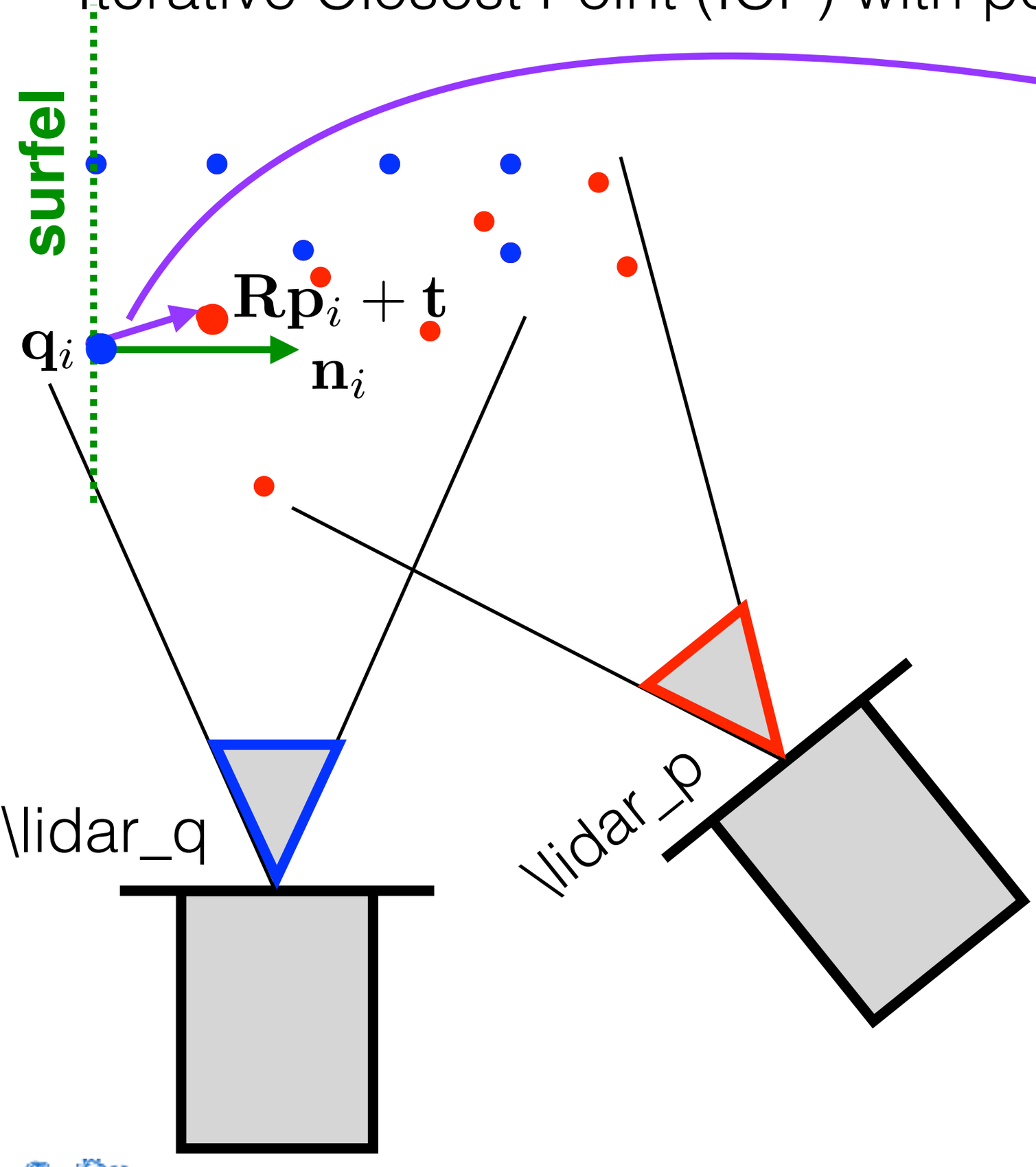
Iterative Closest Point (ICP) with point-to-plane distance



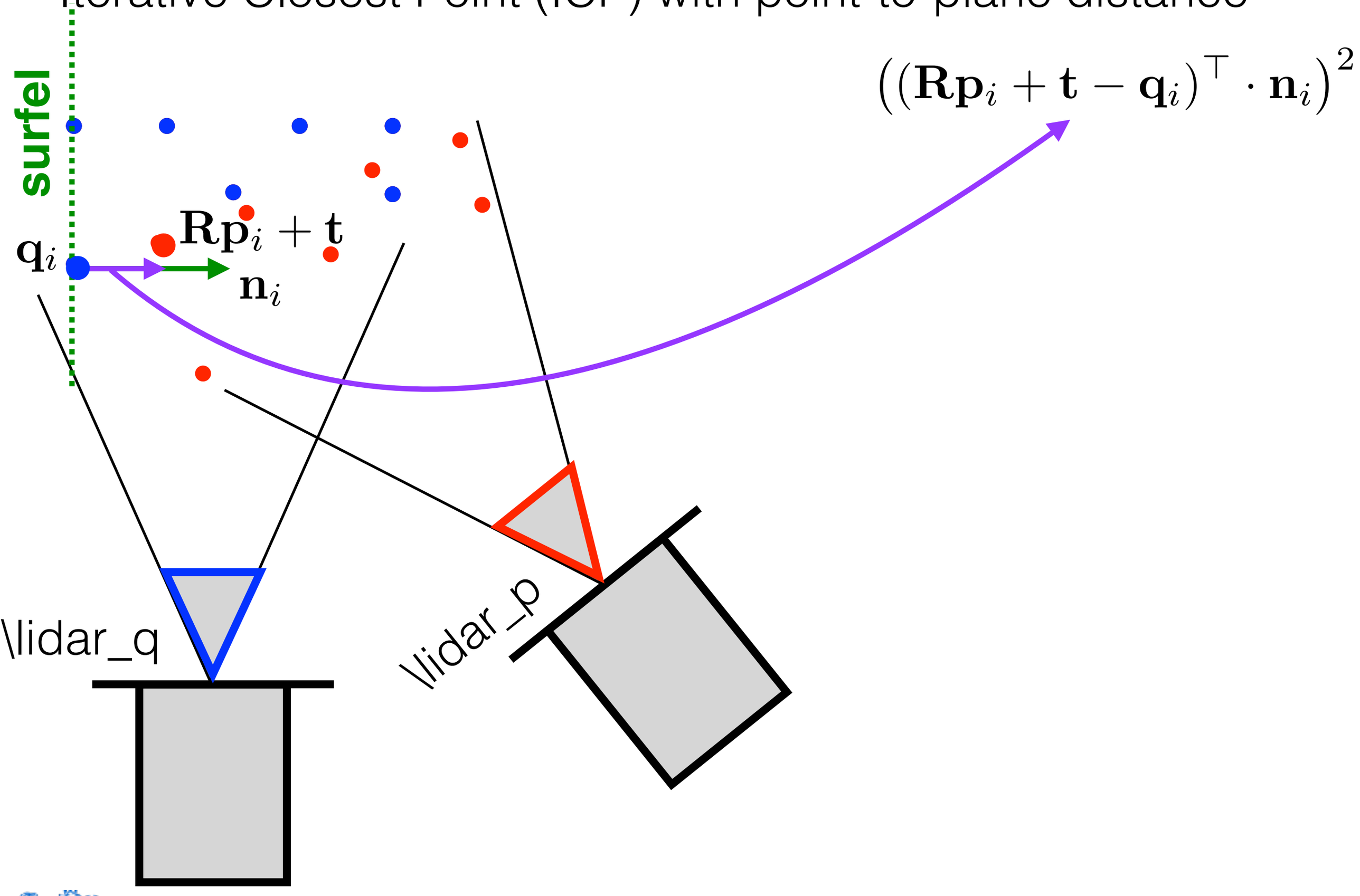
Iterative Closest Point (ICP) with point-to-plane distance



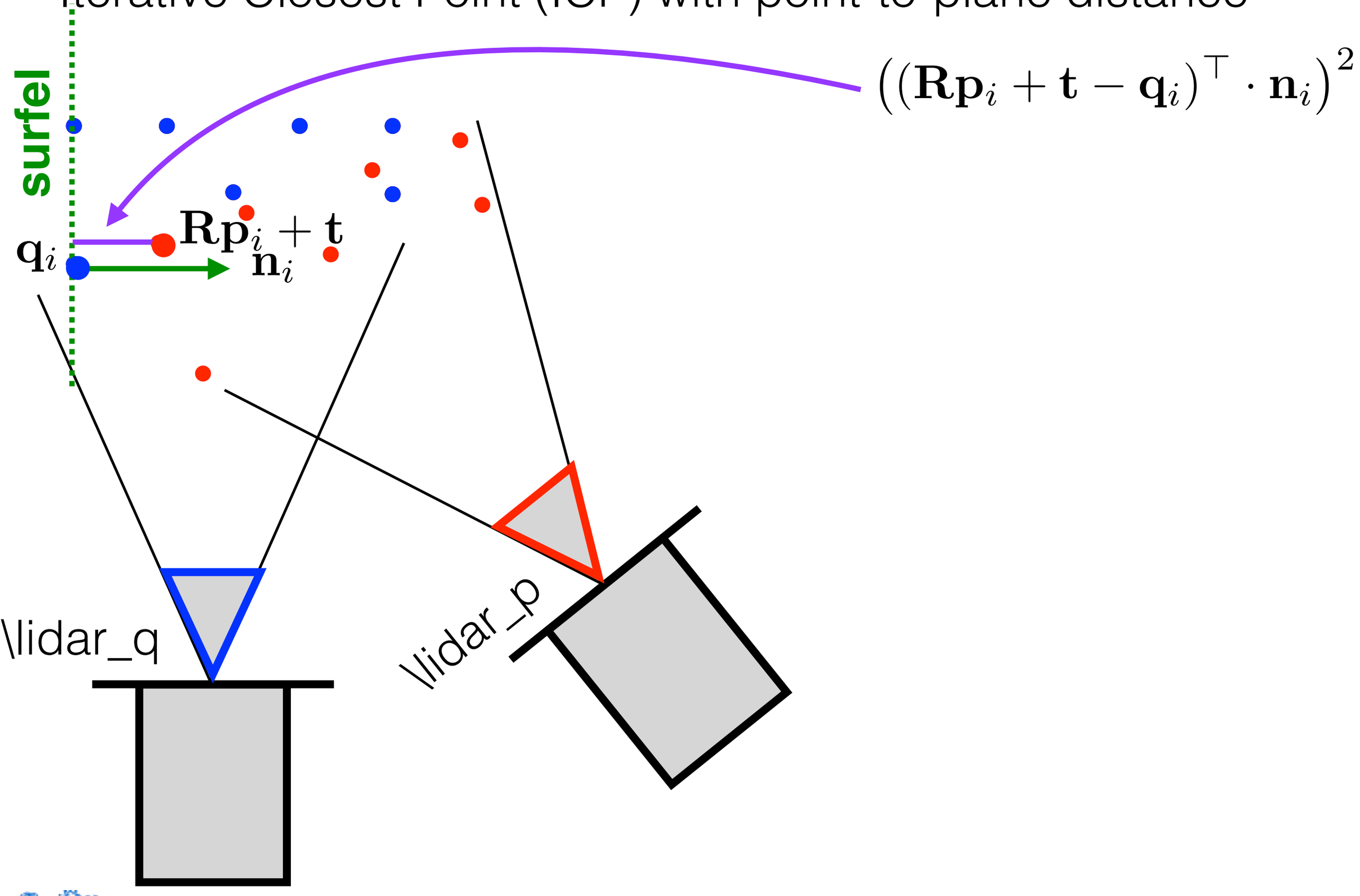
Iterative Closest Point (ICP) with point-to-plane distance



Iterative Closest Point (ICP) with point-to-plane distance

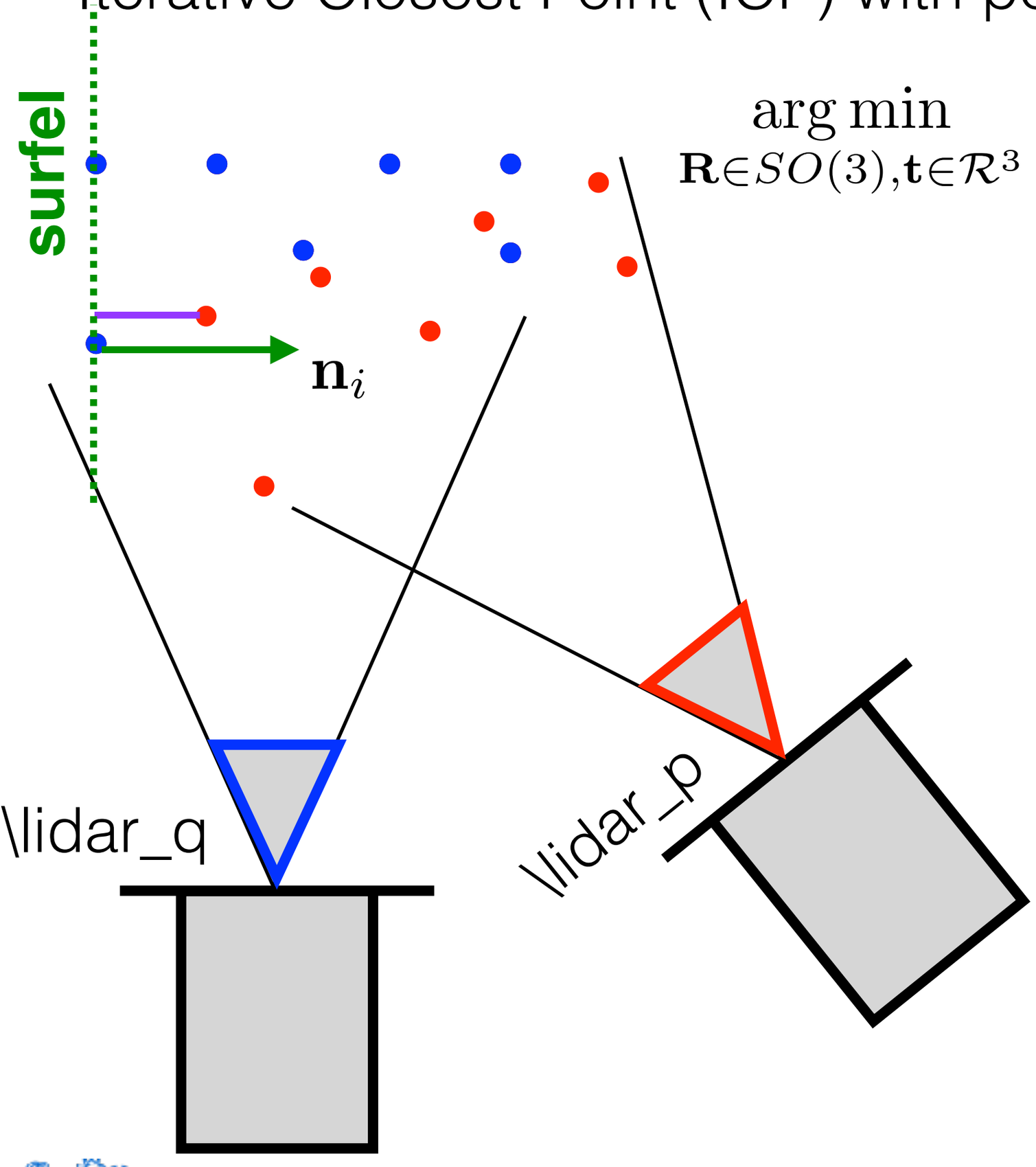


Iterative Closest Point (ICP) with point-to-plane distance

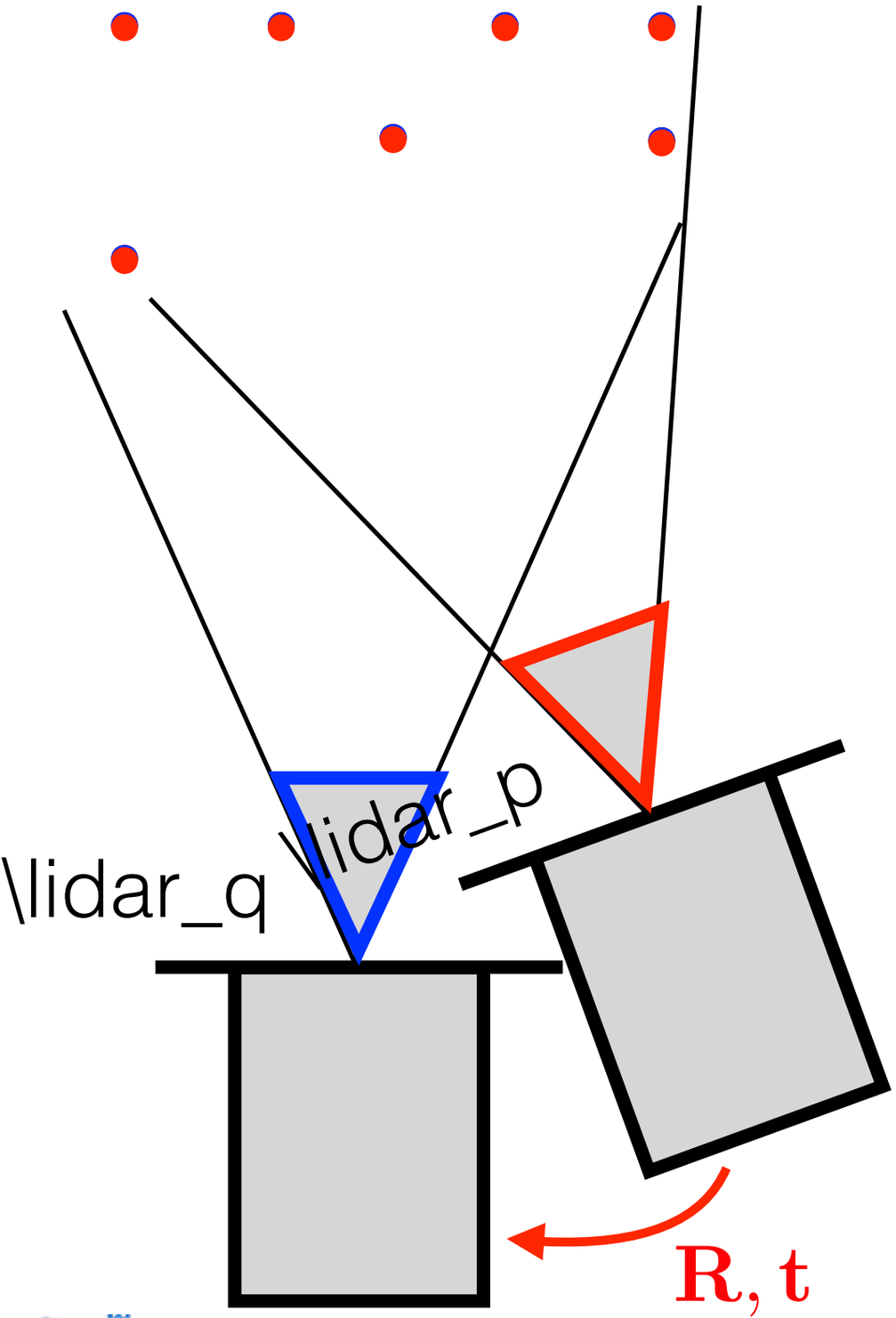


Iterative Closest Point (ICP) with point-to-plane distance

$$\underset{\mathbf{R}\in SO(3),\mathbf{t}\in\mathcal{R}^3}{\arg\min} \sum_i \left((\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i)^\top \cdot \mathbf{n}_i \right)^2$$



Iterative Closest Point (ICP) with point-to-plane distance



3D pointcloud

- pros:
  - accurate (no discretization)
  - allows fast search of nearest neighbours via KD-trees
- cons:
  - memory limitations
    => updating only new space,
    => spatial filtering
  - does not distinguish unknown and unoccupied space
    => not suitable for autonomous exploration

2D/3D voxel grid

- pros:
  - allows to distinguish unknown and unoccupied voxels
    => suitable for planning in partially unknown environment
    => suitable for filtering of noisy points from pointclouds.
- cons:
  - naive implementation is memory exhaustive
    => octree-like implementation required
  - spatial discretization cause inaccuracy
    => not suitable for SLAM directly

Surfel map

- pros:
  - allows for point-to-plane measure
  - provides normals (important for scene understanding)
- cons:
  - requires surfel updating procedure
  -

2.5D height map

- pros:
  - memory efficient
    (represent 3D world in 2D representation)
  - does not discretize vertical axis
- cons:
  - cannot represent tunnels

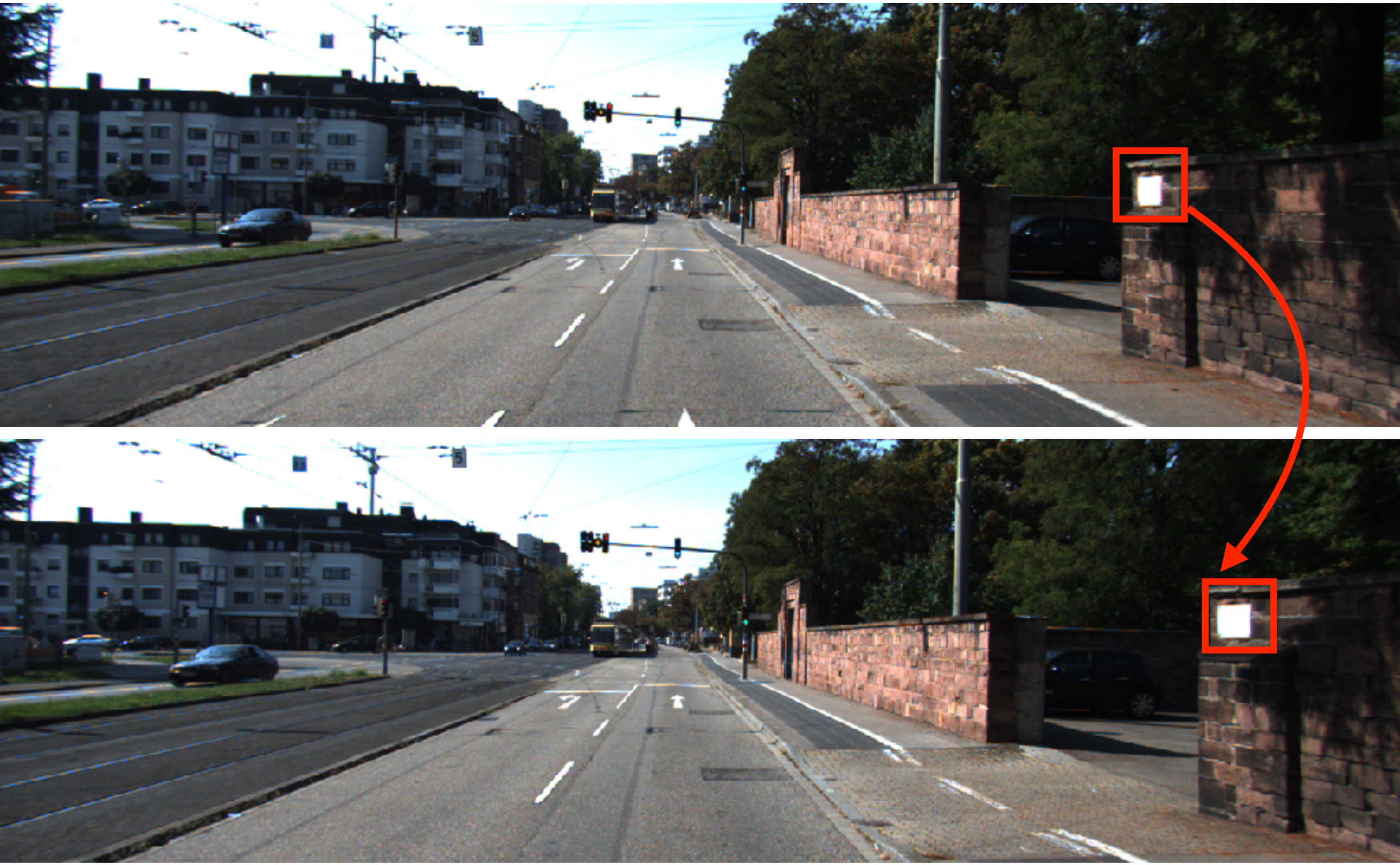
Better 3D-3D correspodences

- Compatibility check of closest points:
  - color compatibility [Godin et al. 94]
  - surface normals should be similar [Pulli 99]
- Directly search for the correspondences by comparing visual patches in camera images.

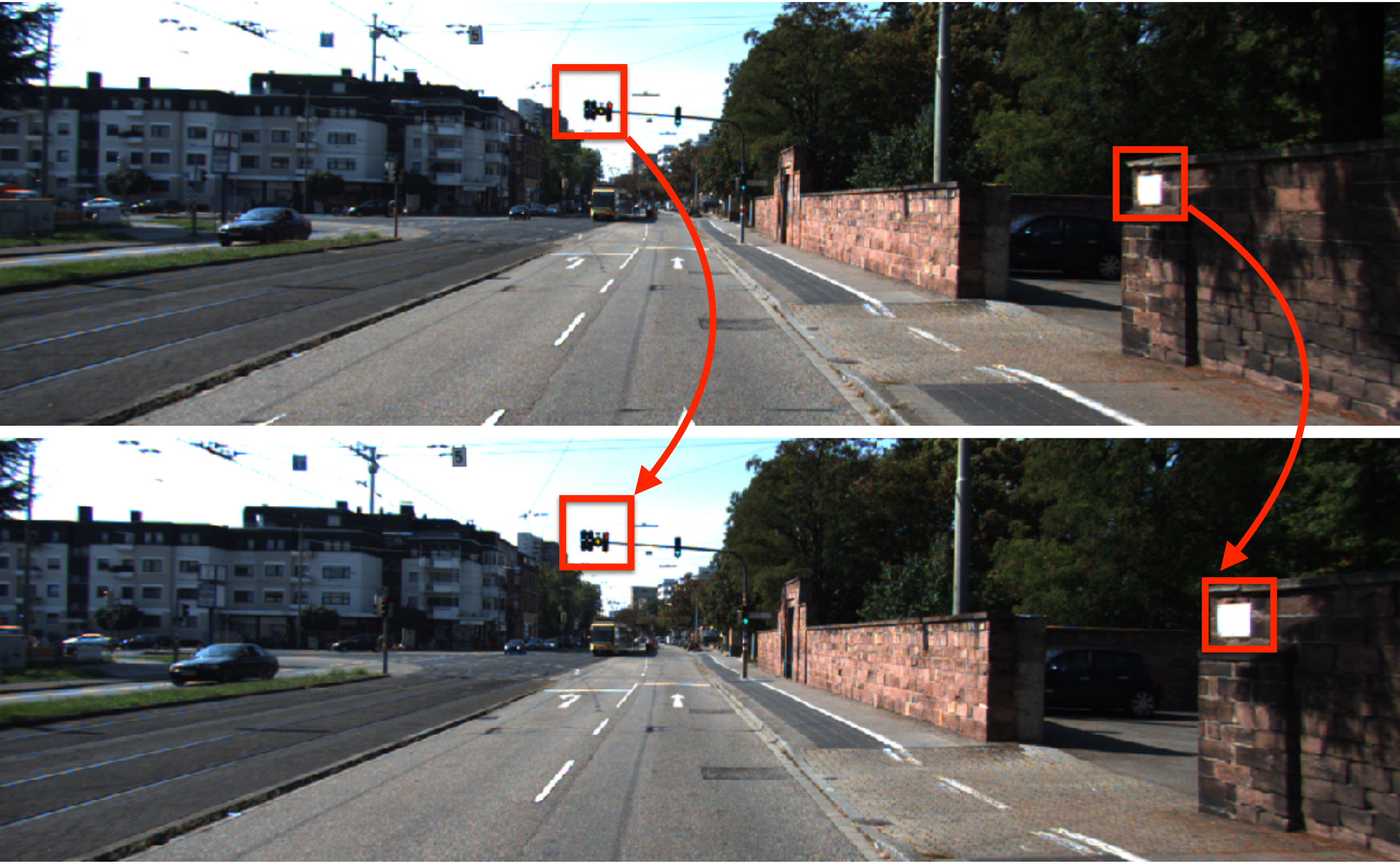
Search for correspondences in camera images

Search for correspondences in camera images

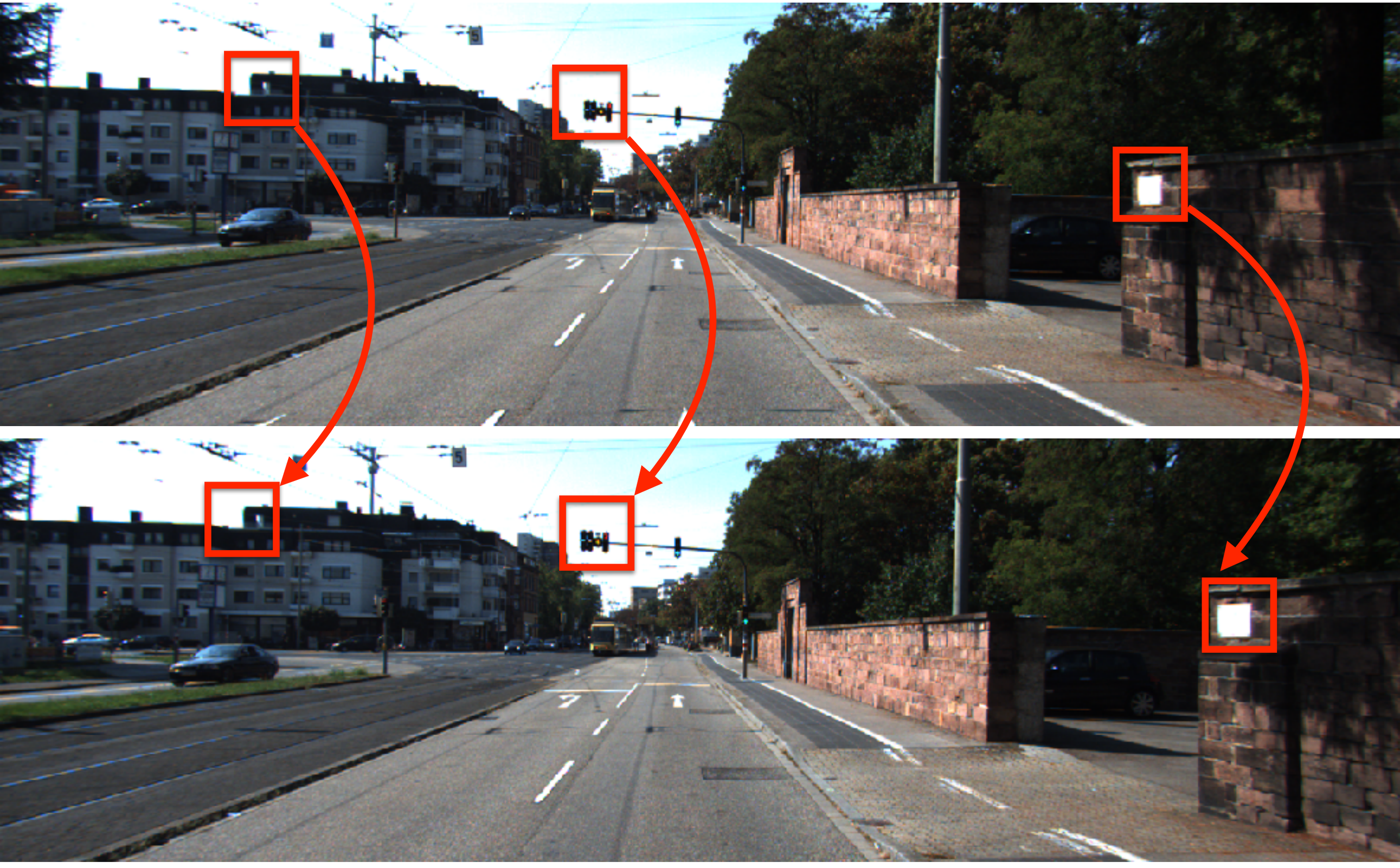
Search for correspondences in camera images



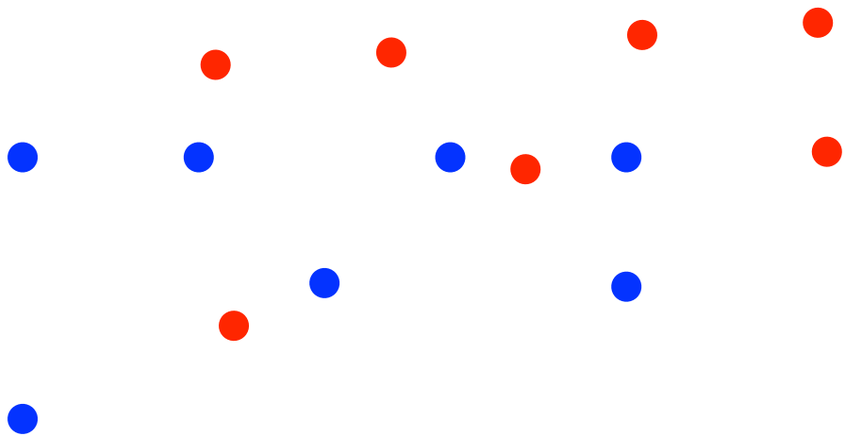
Search for correspondences in camera images

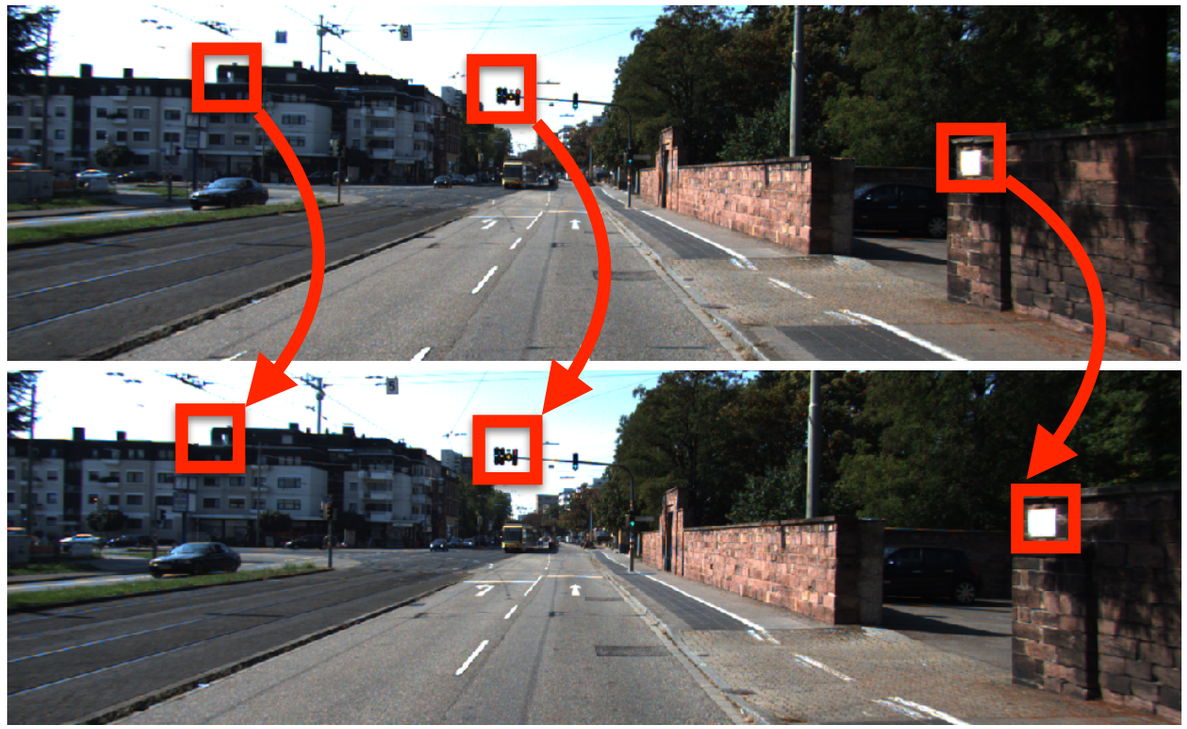


Search for correspondences in camera images



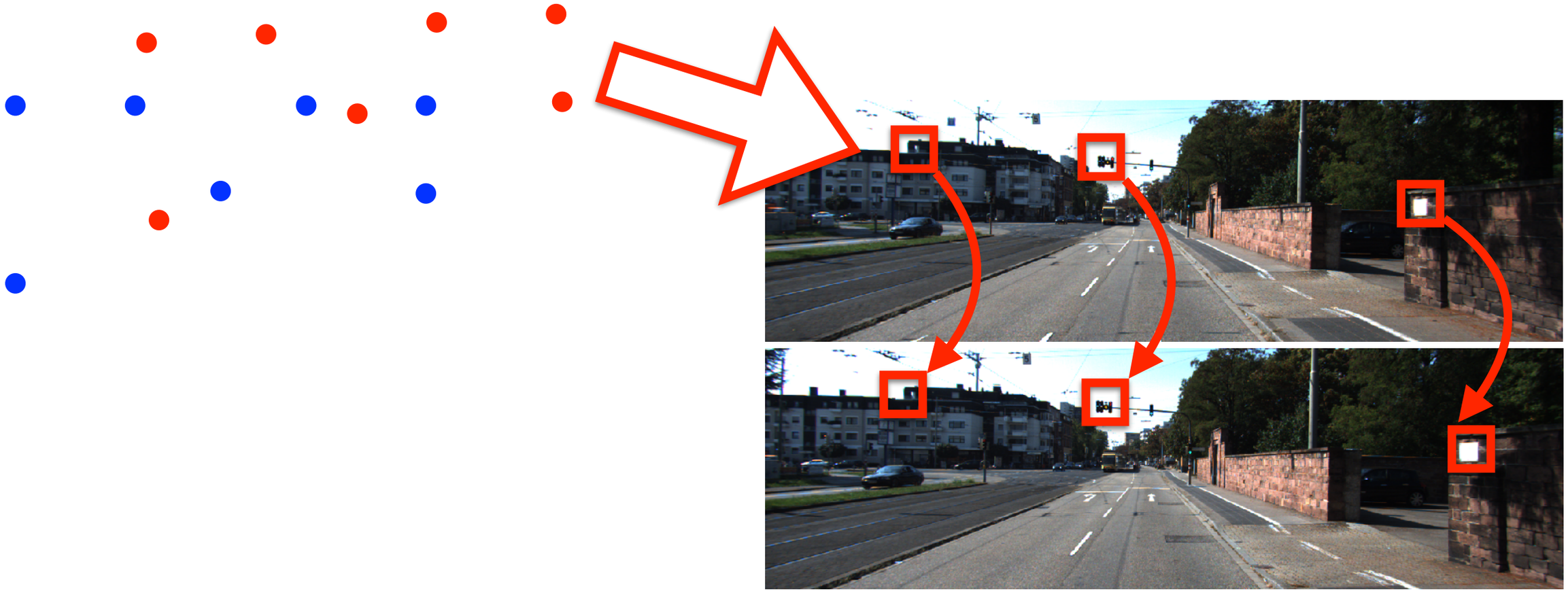
Search for correspondences in camera images





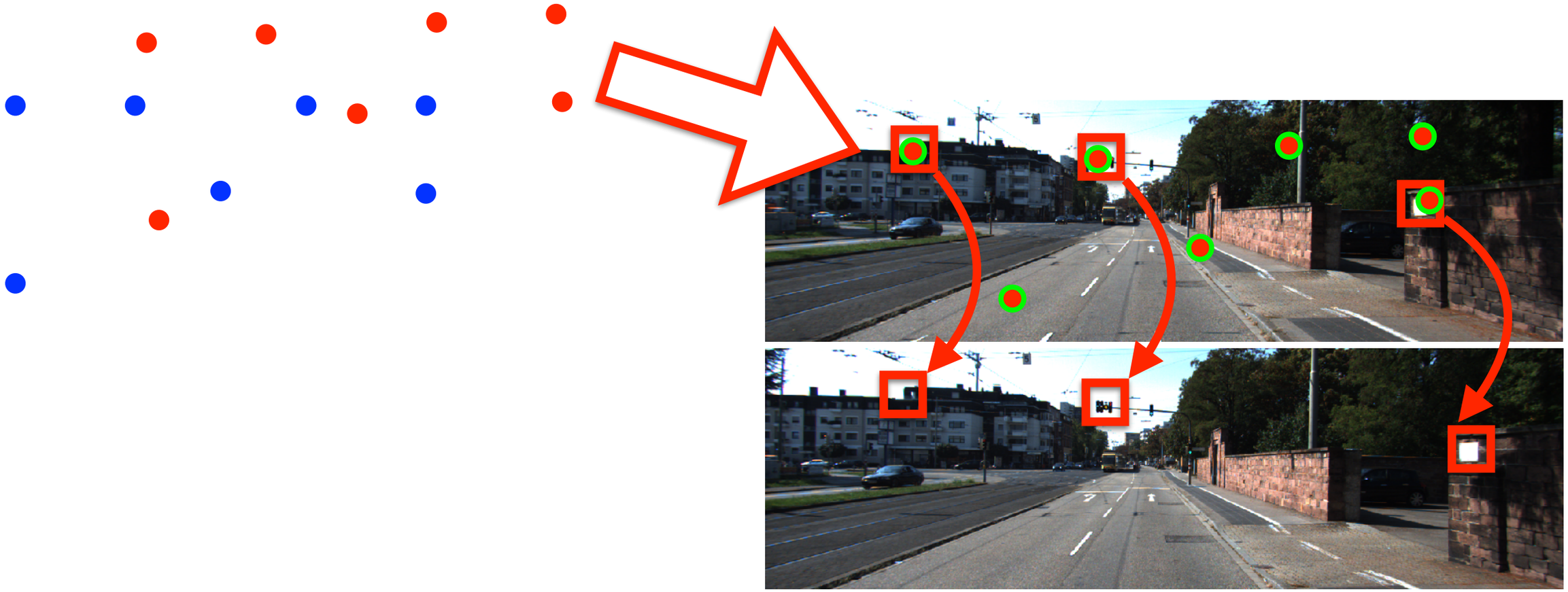
Lidar calibrated wrt camera

Search for correspondences in camera images



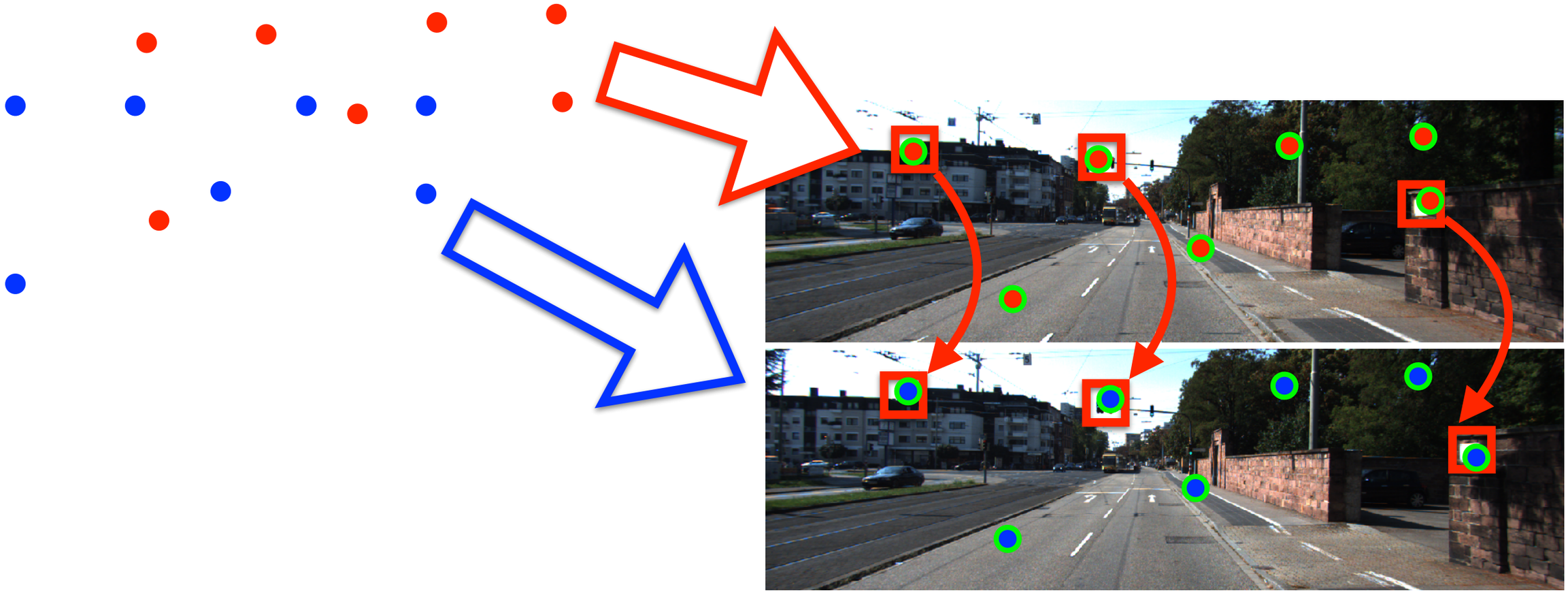
Project 3D pointcloud on image plane

Search for correspondences in camera images



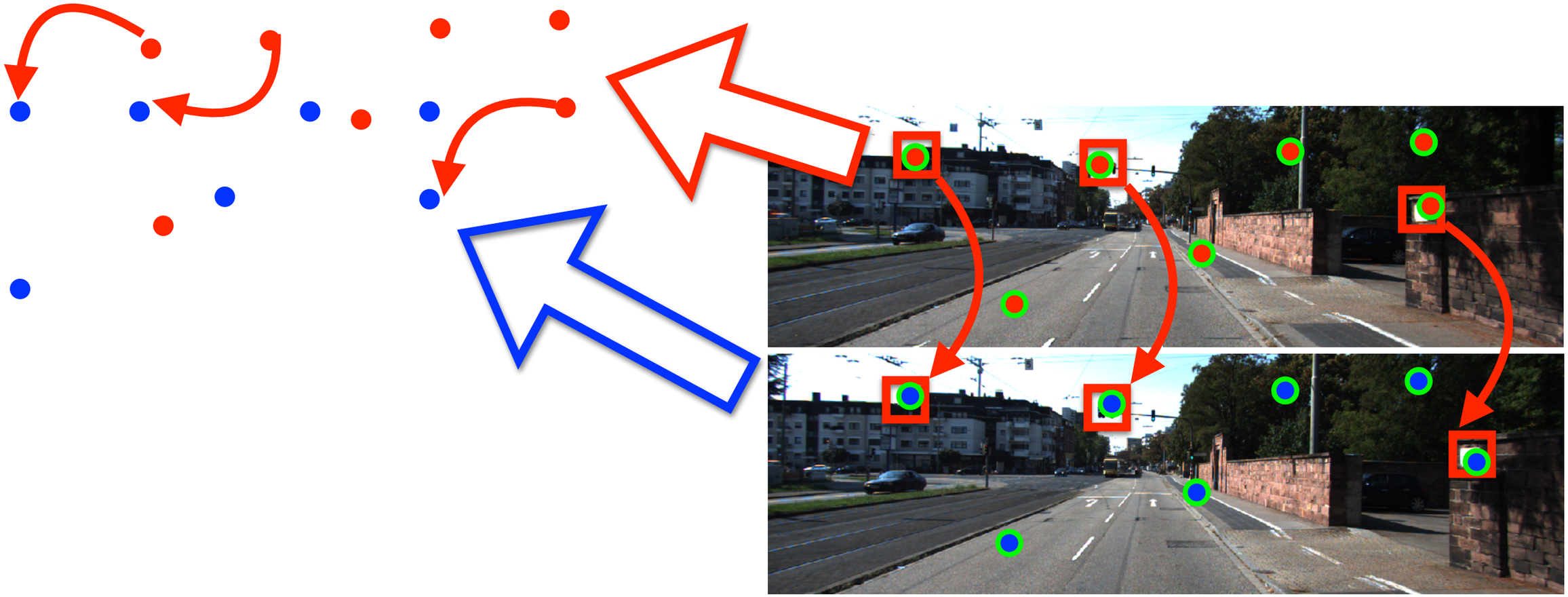
Project 3D pointcloud on image plane

Search for correspondences in camera images



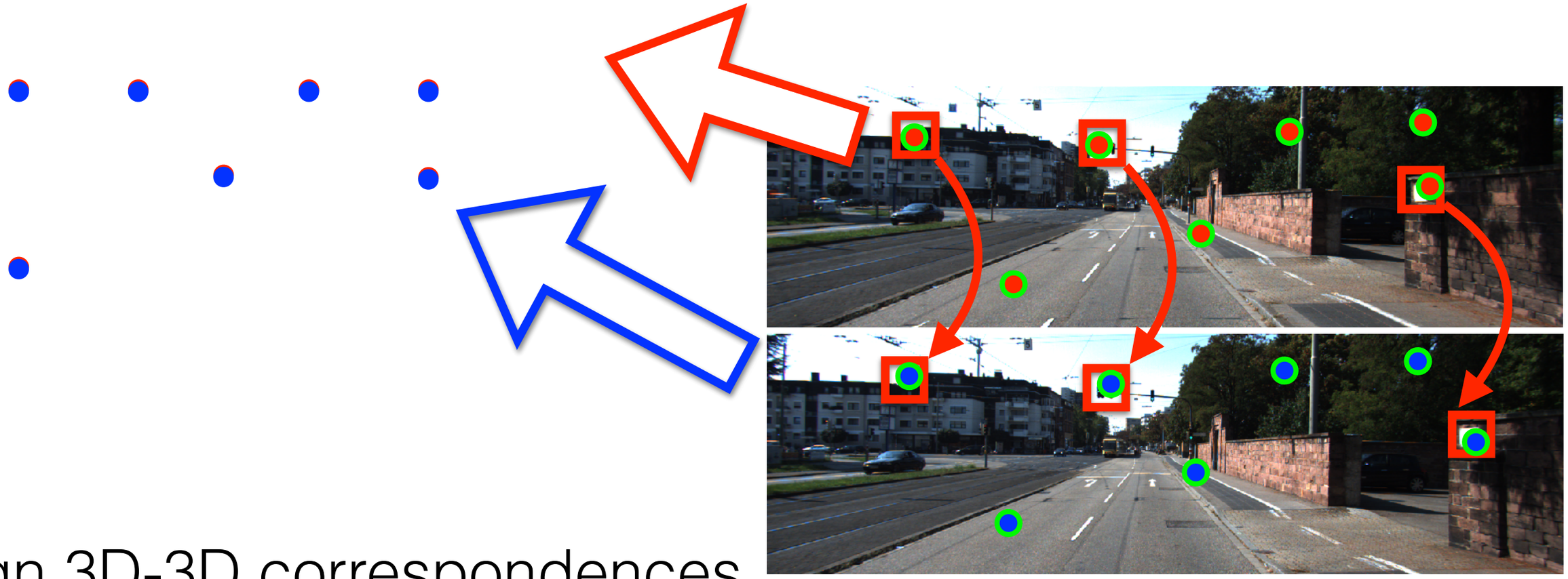
Project 3D pointcloud on image plane

Search for correspondences in camera images



For each 2D-2D correspondence, find closest lidar point
=> 3D-3D correspondences

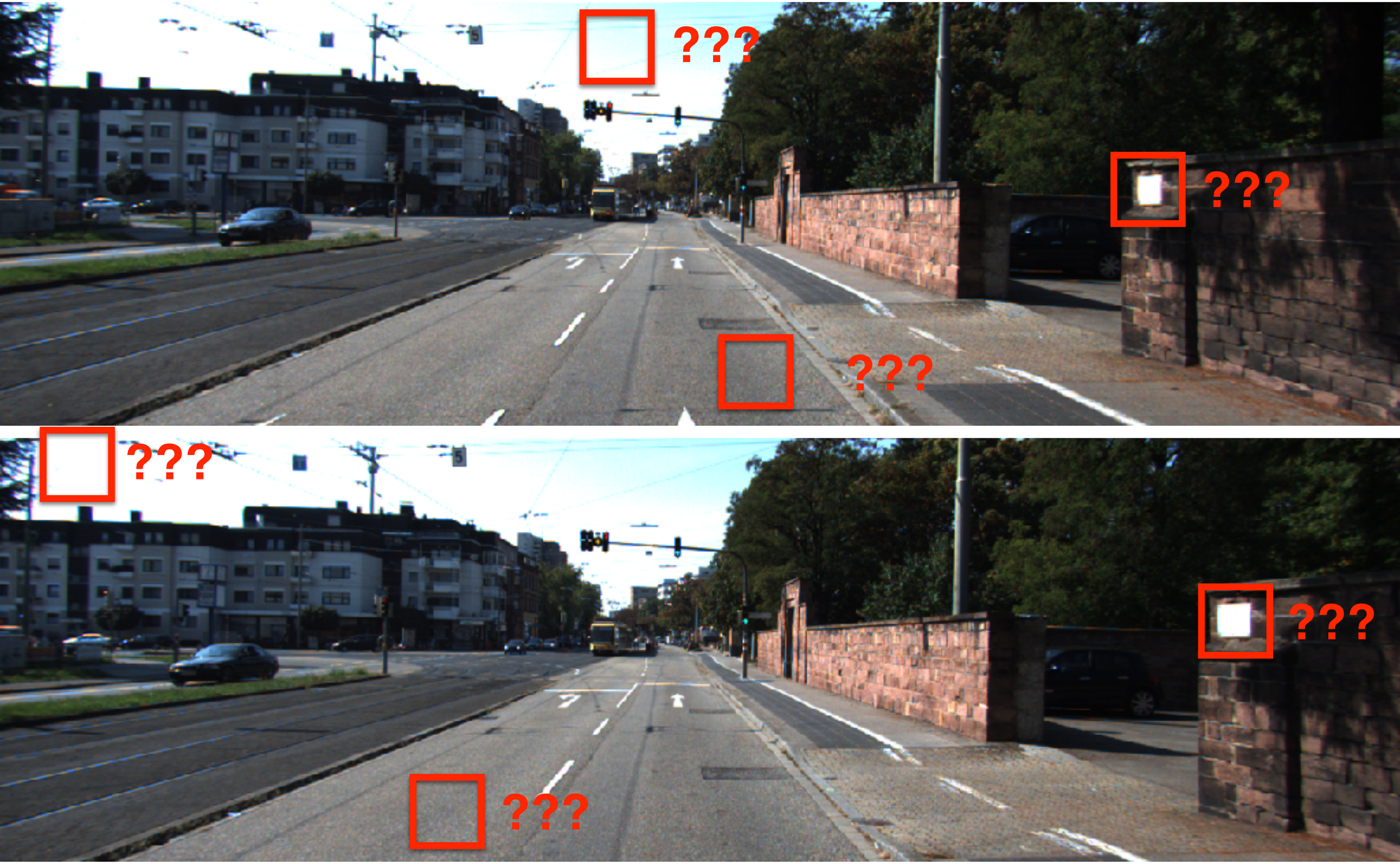
Search for correspondences in camera images



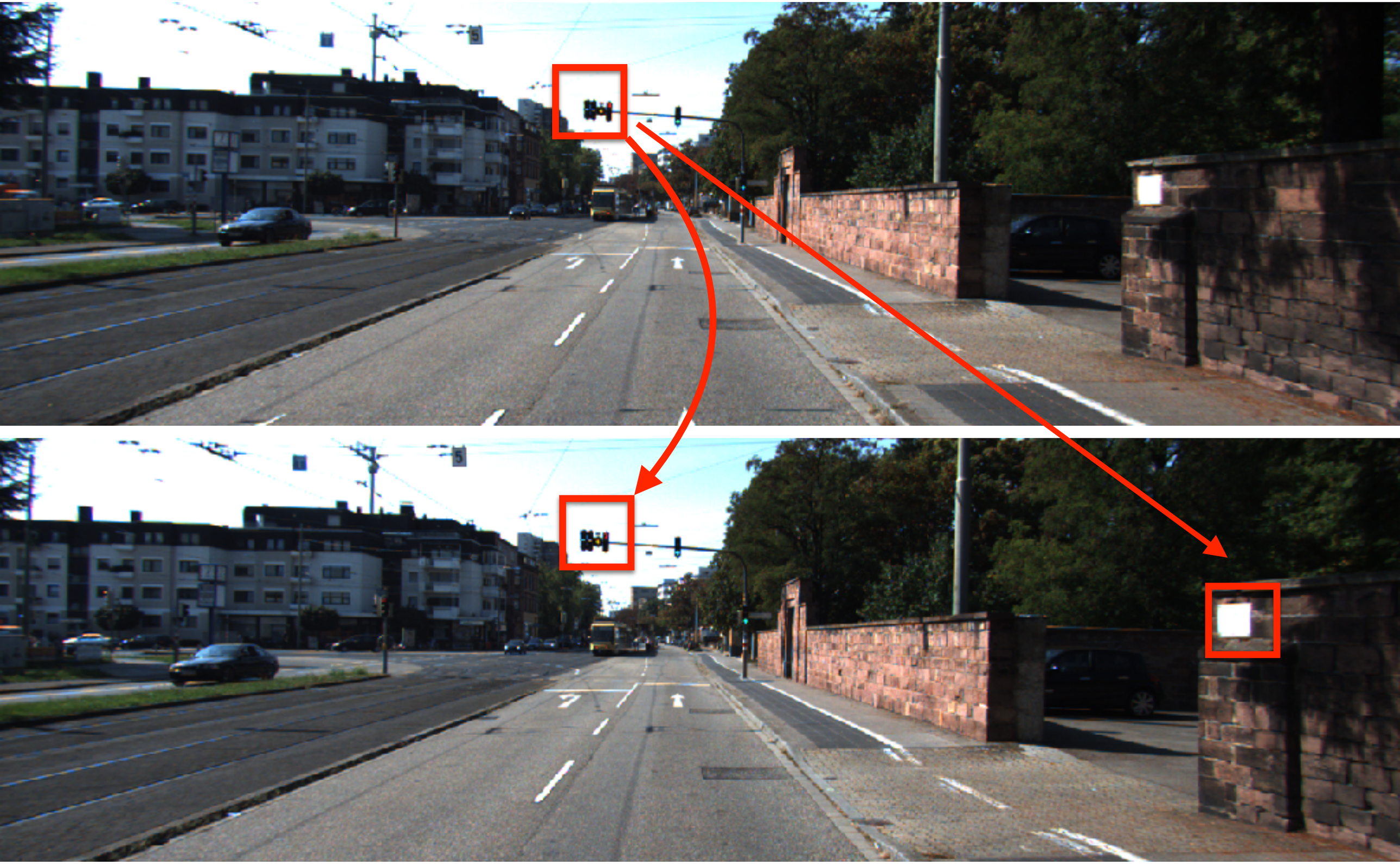
Align 3D-3D correspondences

How to find these 2D-2D correspondences in images?

Problem1: Which points are suitable candidates?



Problem2: How to measure similarity?



Problem2: How to measure similarity?

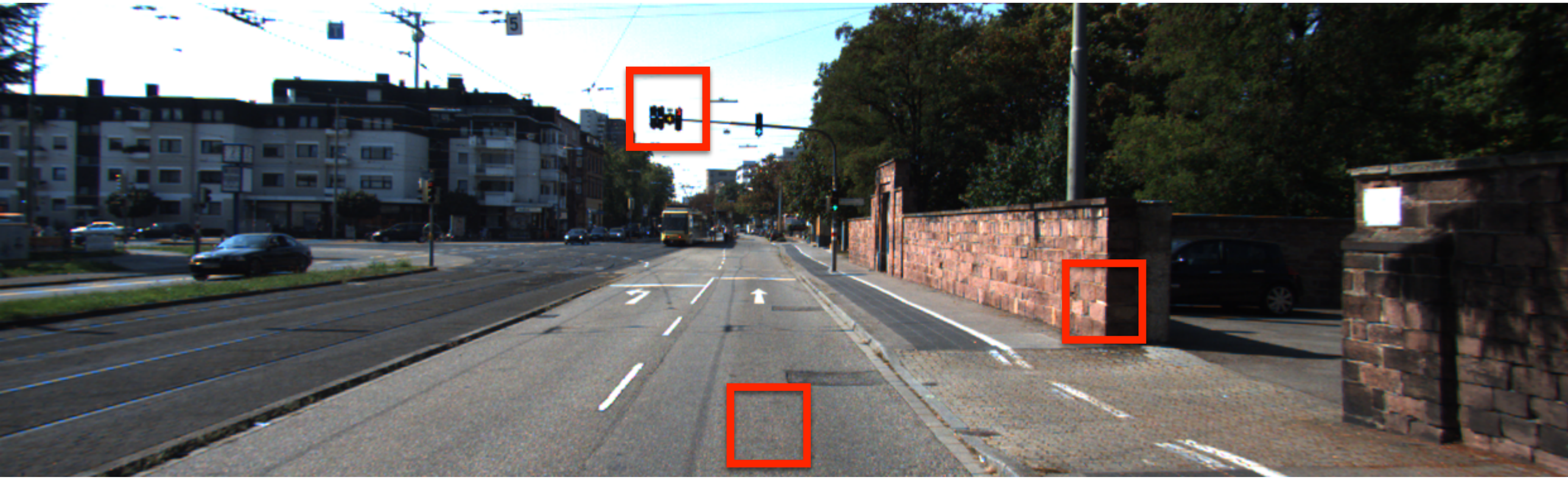
Problem2: How to measure similarity?

Outline: Search for correspondences in camera images

Problem1: Which points are suitable candidates?

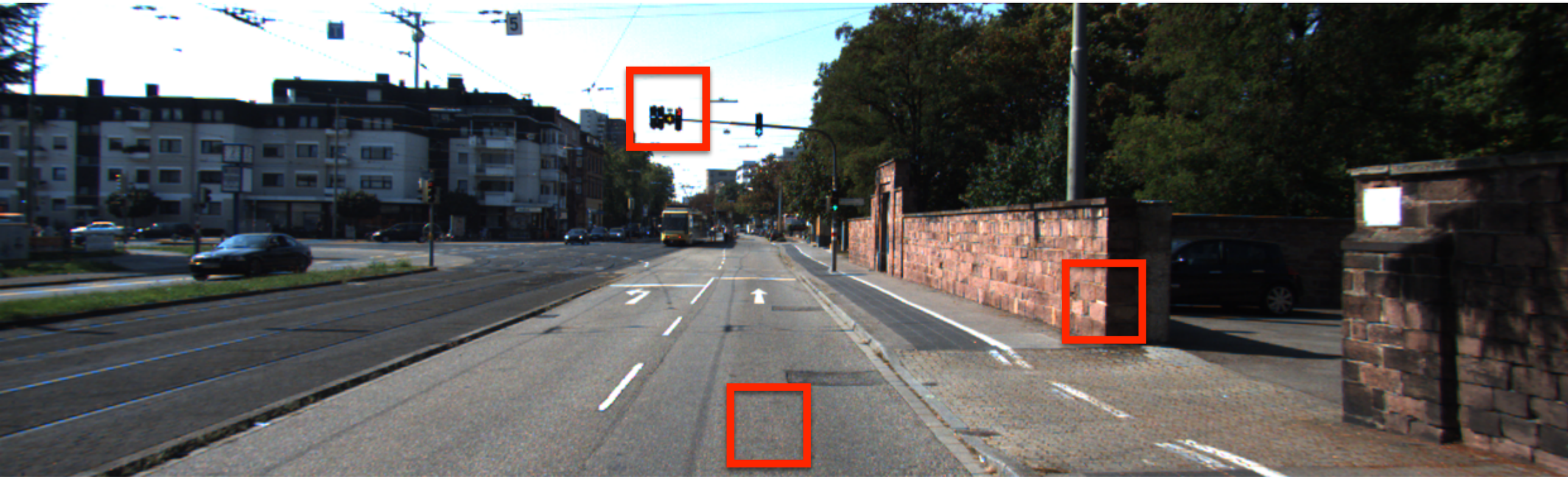
Problem2: How to measure similarity?

Problem1: Which points are suitable candidates?



Problem1: Which points are suitable candidates?

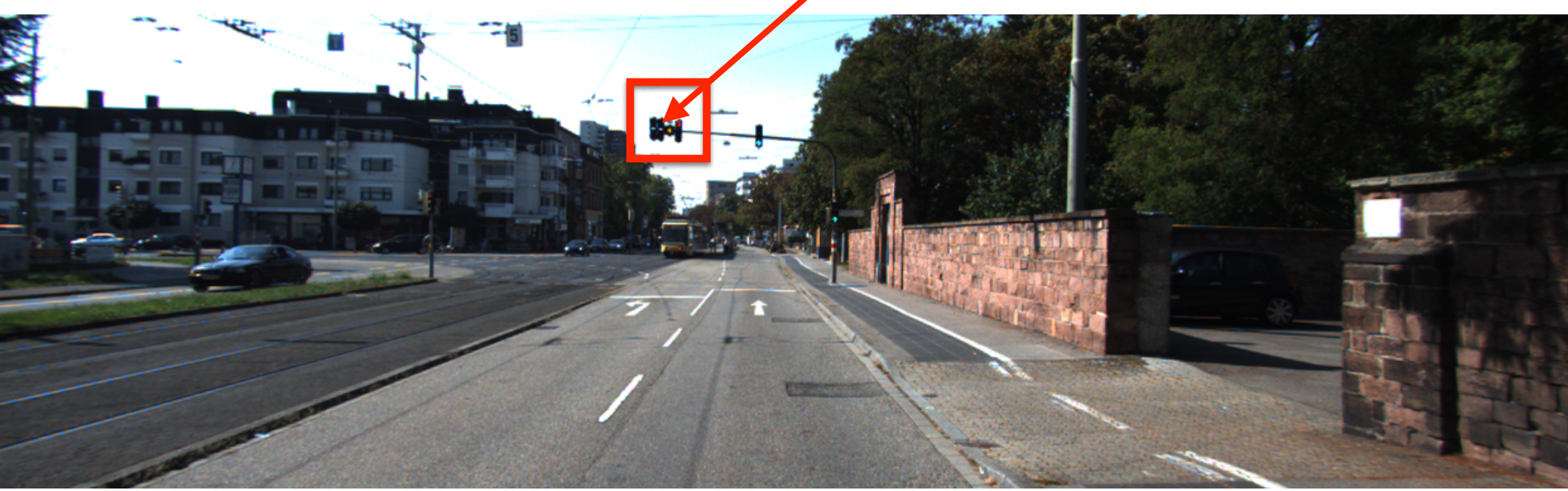
Points with patches dissimilar to their neighbourhood



Problem1: Which points are suitable candidates?

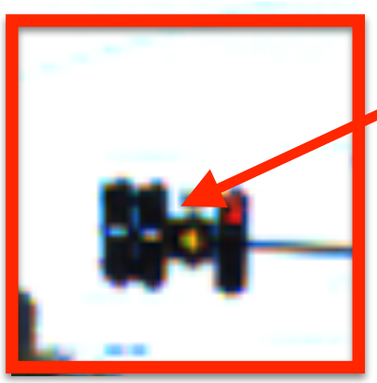
Points with patches dissimilar to their neighbourhood

Let us study suitability of point $\mathbf{u} \in \mathcal{R}^2$



Problem1: Which points are suitable candidates?

Points with patches dissimilar to their neighbourhood

$I(\mathbf{x}) =$  $\mathbf{u} \in \mathcal{R}^2$

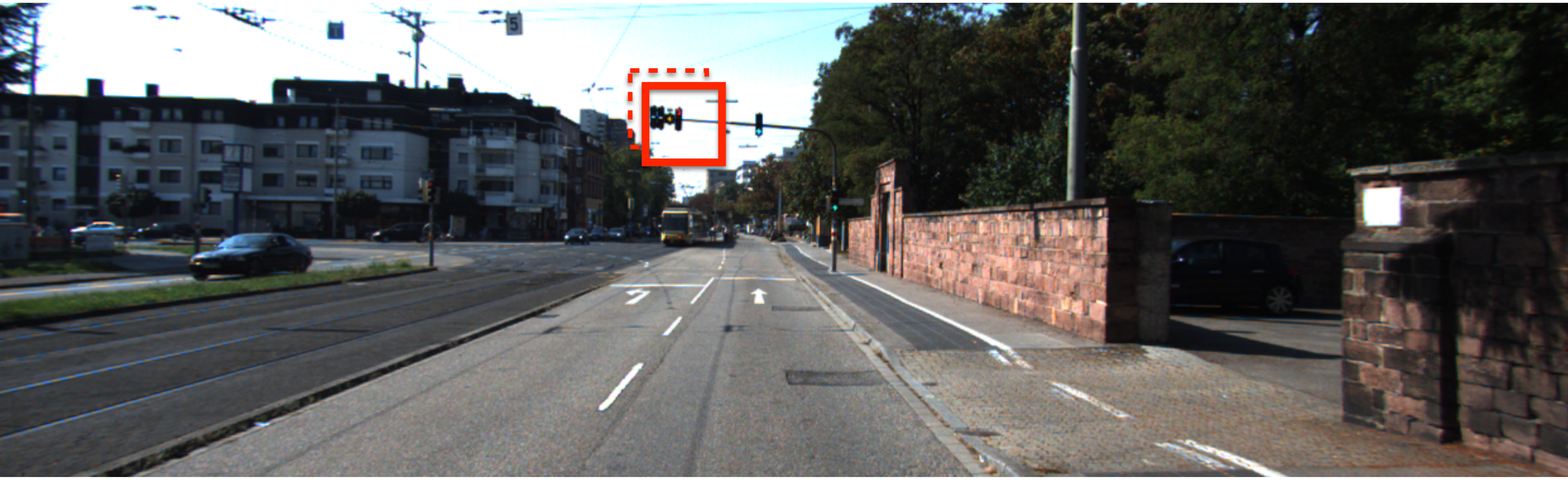
…image patch in the neighbood of $\mathbf{u}$

$\mathbf{x} \in \mathcal{W}(\mathbf{u})$ …set of pixels in the neighbourhood of $\mathbf{u}$

Problem1: Which points are suitable candidates?

Points with patches dissimilar to their neighbourhood

$$I(\mathbf{x}) = \quad \mathbf{x} \in \mathcal{W}(\mathbf{u})$$



Problem1: Which points are suitable candidates?

Points with patches dissimilar to their neighbourhood

$$I(\mathbf{x}) = \quad \mathbf{x} \in \mathcal{W}(\mathbf{u})$$

$$I(\mathbf{x}+\mathbf{t}) = \quad \mathbf{x} \in \mathcal{W}(\mathbf{u})$$

image patch in neighbourhood

Problem1: Which points are suitable candidates?

Points with patches dissimilar to their neighbourhood

$$I(\mathbf{x}) = \underset{\mathbf{x} \in \mathcal{W}(\mathbf{u})}{\text{ }} \qquad I(\mathbf{x}+\mathbf{t}) = \underset{\mathbf{x} \in \mathcal{W}(\mathbf{u})}{\text{ }}$$

Point $\mathbf{u}$ whose dissimilarity with neigbouring patches grows fast in all directions is the suitable point.

$$E(\mathbf{t}) = \sum_{\mathbf{x} \in \mathcal{W}(\mathbf{u})} \left(I(\mathbf{x}+\mathbf{t}) - I(\mathbf{x})\right)^2 \approx \mathbf{t}^\top \underbrace{\sum_{\mathbf{x} \in \mathcal{W}(\mathbf{u})} \frac{\partial I(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial I(\mathbf{x})}{\partial \mathbf{x}}^\top}_{\mathbf{M}} \mathbf{t}$$

$$I(\mathbf{x}+\mathbf{t}) \approx I(\mathbf{x}) + \frac{\partial I(\mathbf{x})}{\partial \mathbf{x}}^\top \mathbf{t}$$

Problem1: Which points are suitable candidates?

Points with patches dissimilar to their neighbourhood

$$I(\mathbf{x}) = \underset{\mathbf{x} \in \mathcal{W}(\mathbf{u})}{\text{ }} \qquad I(\mathbf{x}+\mathbf{t}) = \underset{\mathbf{x} \in \mathcal{W}(\mathbf{u})}{\text{ }}$$

Point $\mathbf{u}$ whose dissimilarity with neigbouring patches grows fast in all directions is the suitable point.

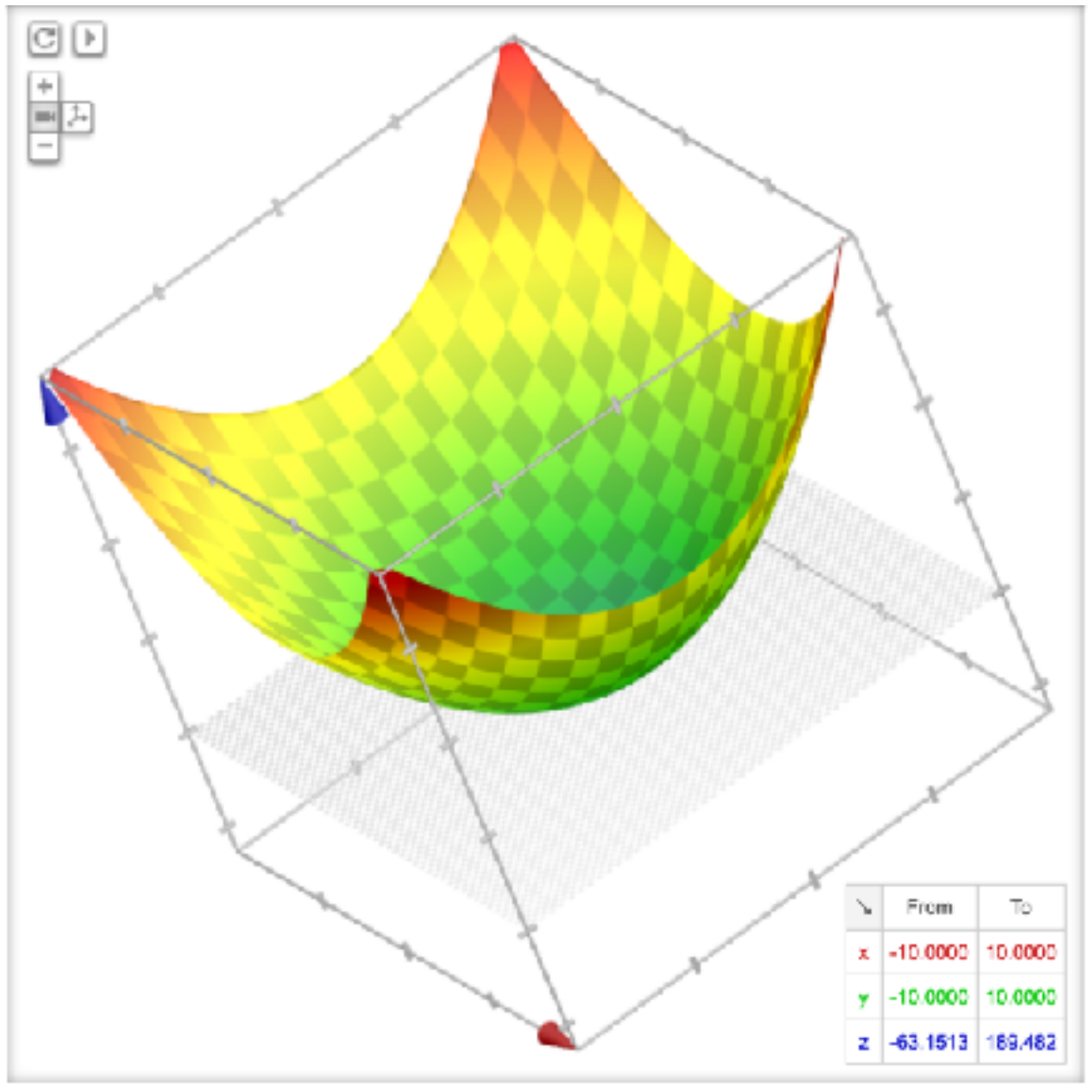
$$E(\mathbf{t}) = \sum_{\mathbf{x} \in \mathcal{W}(\mathbf{u})} \left(I(\mathbf{x}+\mathbf{t}) - I(\mathbf{x})\right)^2 \approx \mathbf{t}^\top \underbrace{\sum_{\mathbf{x} \in \mathcal{W}(\mathbf{u})} \frac{\partial I(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial I(\mathbf{x})}{\partial \mathbf{x}}^\top}_{\mathbf{M}} \mathbf{t}$$

$E(\mathbf{t}) \approx \mathbf{t}^\top \mathbf{M}\, \mathbf{t}$ is positive semidefinite quadric

Problem1: Which points are suitable candidates?

Points with patches dissimilar to their neighbourhood

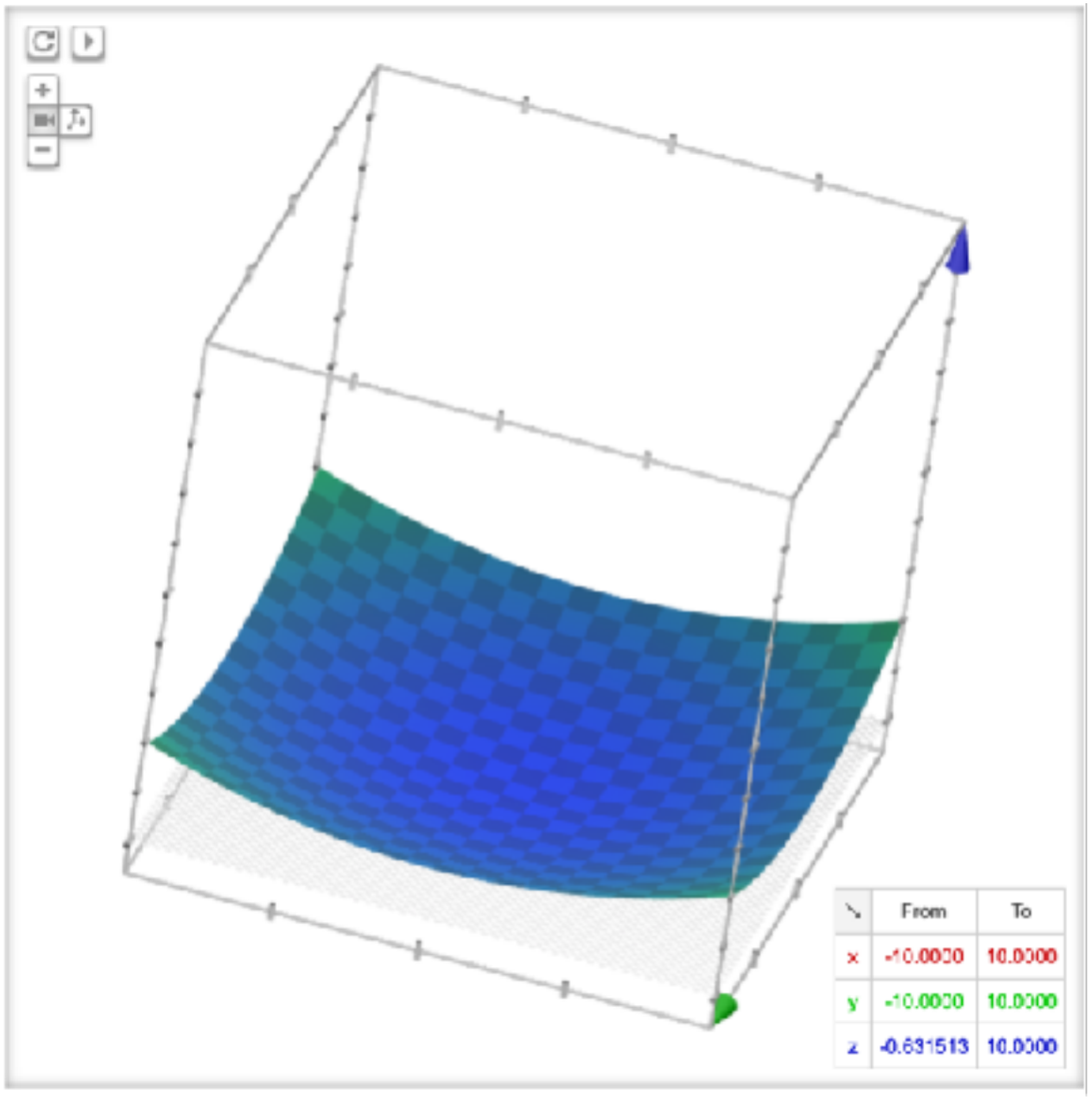
$E(\mathbf{t}) \approx \mathbf{t}^\top \mathbf{M}\, \mathbf{t}$ is positive semidefinite quadric



Problem1: Which points are suitable candidates?

Points with patches dissimilar to their neighbourhood

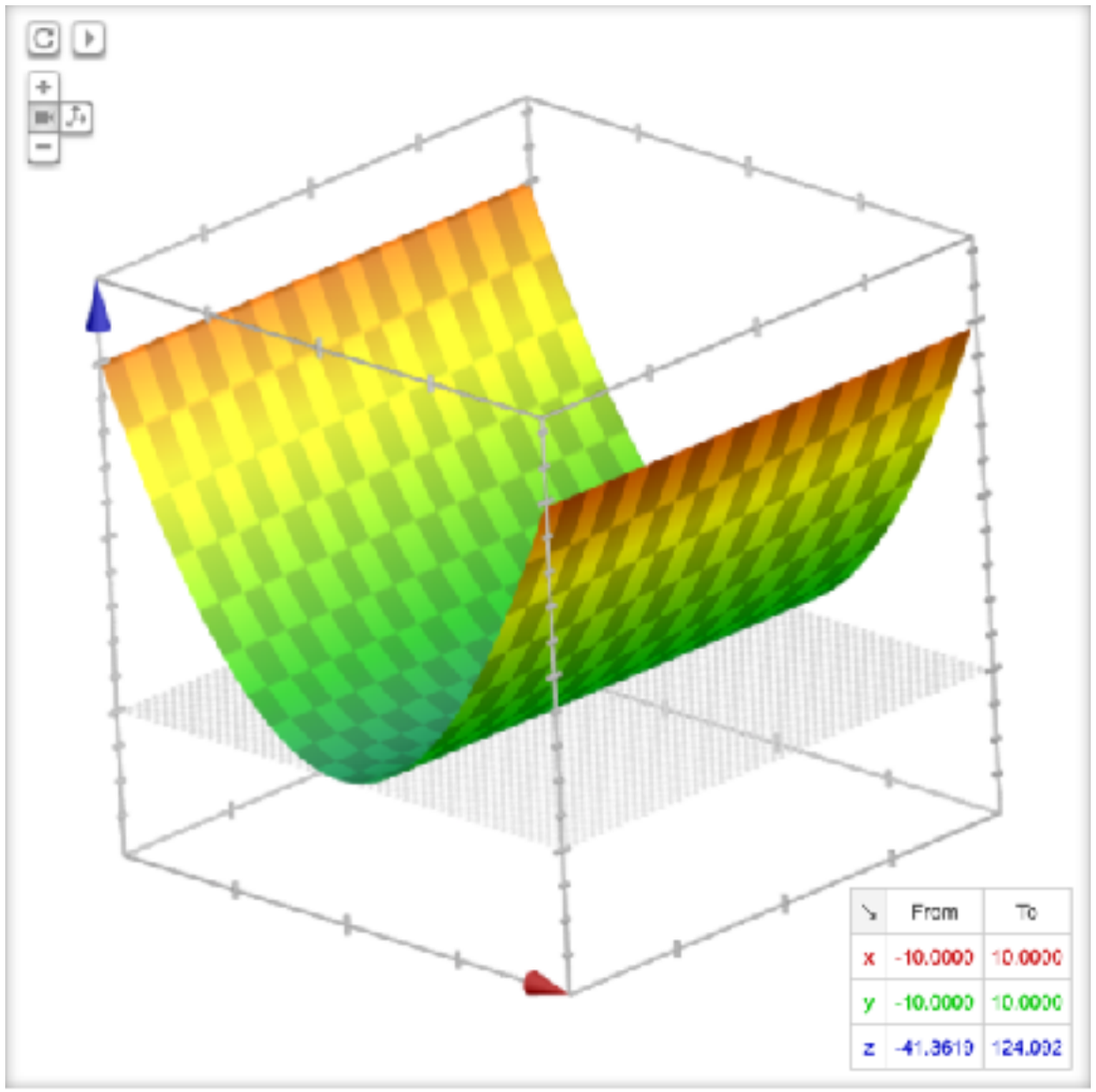
$E(\mathbf{t}) \approx \mathbf{t}^\top \mathbf{M}\, \mathbf{t}$ is positive semidefinite quadric



Problem1: Which points are suitable candidates?

Points with patches dissimilar to their neighbourhood

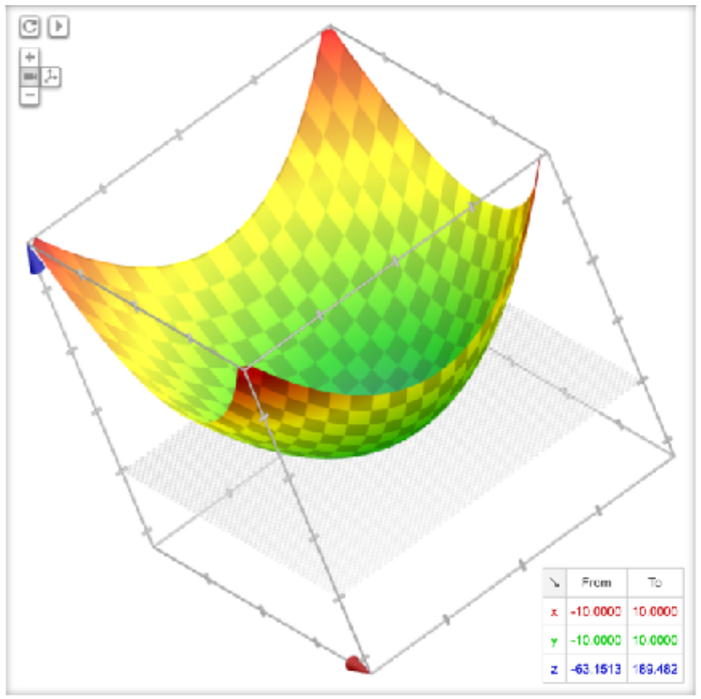
$E(\mathbf{t}) \approx \mathbf{t}^\top \mathbf{M}\, \mathbf{t}$ is positive semidefinite quadric



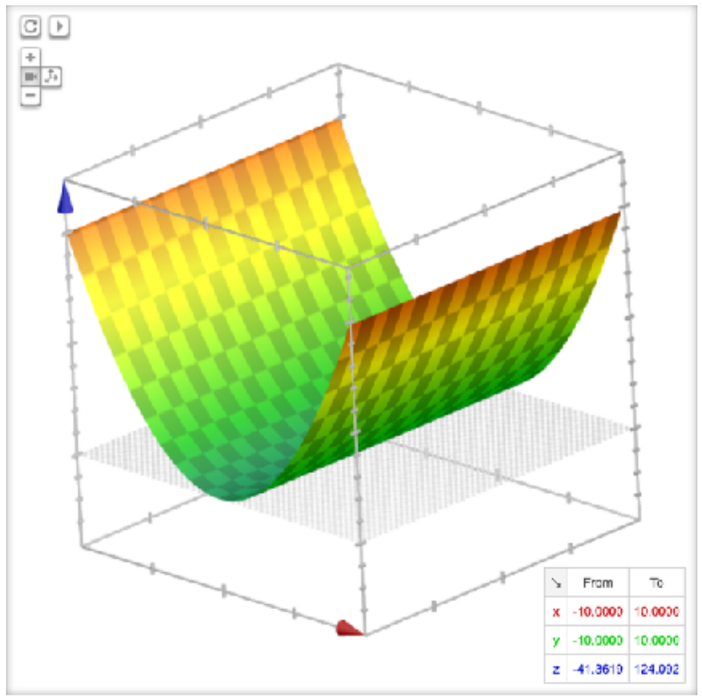
Problem1: Which points are suitable candidates?

Points with patches dissimilar to their neighbourhood

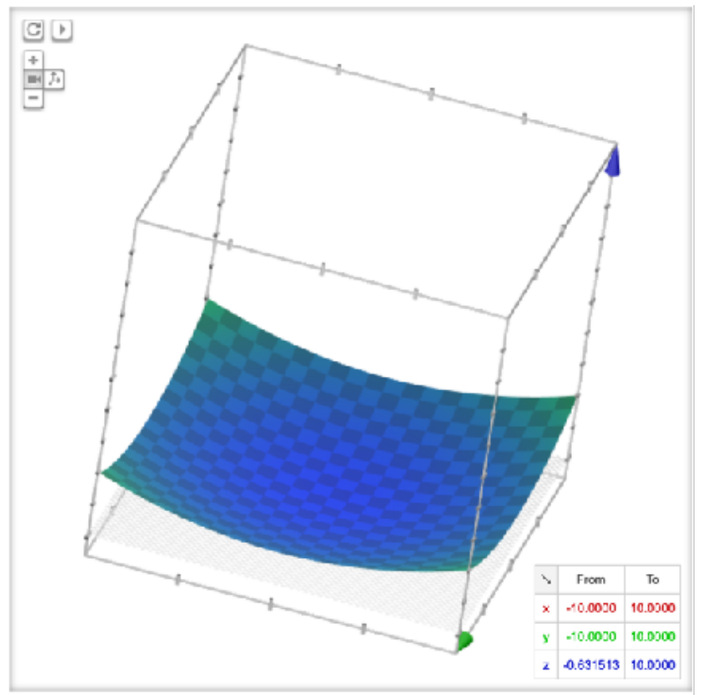
$E(\mathbf{t}) \approx \mathbf{t}^\top \mathbf{M}\, \mathbf{t}$ is positive semidefinite quadric



$\lambda_1 >> 0, \lambda_2 >> 0$



$\lambda_1 >> 0, \lambda_2 \approx 0$



$\lambda_1 \approx 0, \lambda_2 \approx 0$

Shape of quadric determined by eigenvalues λ_1, λ_2 of $\mathbf{M}$

Problem1: Which points are suitable candidates?

Harris corner detector (simplified):

1. Compute gradient image
2. Efficiently estimate (e.g. convolution) matrix for every point

$$\mathbf{M}(\mathbf{u}) = \sum_{\mathbf{x} \in \mathcal{W}(\mathbf{u})} \frac{\partial I(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial I(\mathbf{x})}{\partial \mathbf{x}}^{\top}$$

3. Compute (or approximate) eigenvalues $\lambda_1(\mathbf{u}), \lambda_2(\mathbf{u})$
4. Denote local maximas with both eigs bigger than threshold

Problem1: Which points are suitable candidates?

Summary

- Corner points are suitable candidates.
- There is a faster but more hacked version of corner detector called FAST [Rosten TPAMI 2010].
- OpenCV implementation

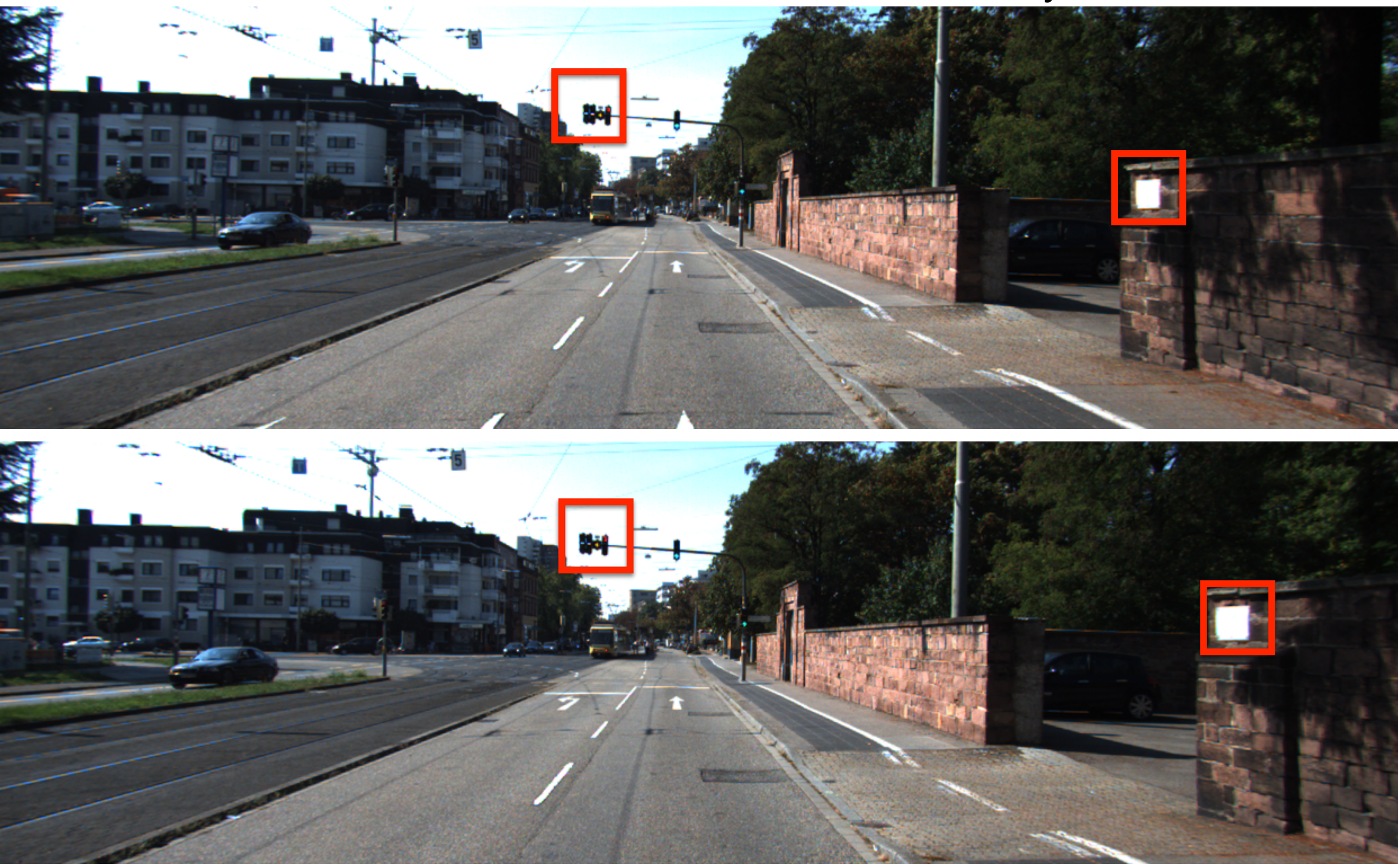
```
# initialise
fast = cv2.FastFeatureDetector()
# detect corner points
kp = fast.detect(img,None)
# draw detected corner points
img2 = cv2.drawKeypoints(img, kp, color=(255,0,0))
```

Outline: Search for correspondences in camera images

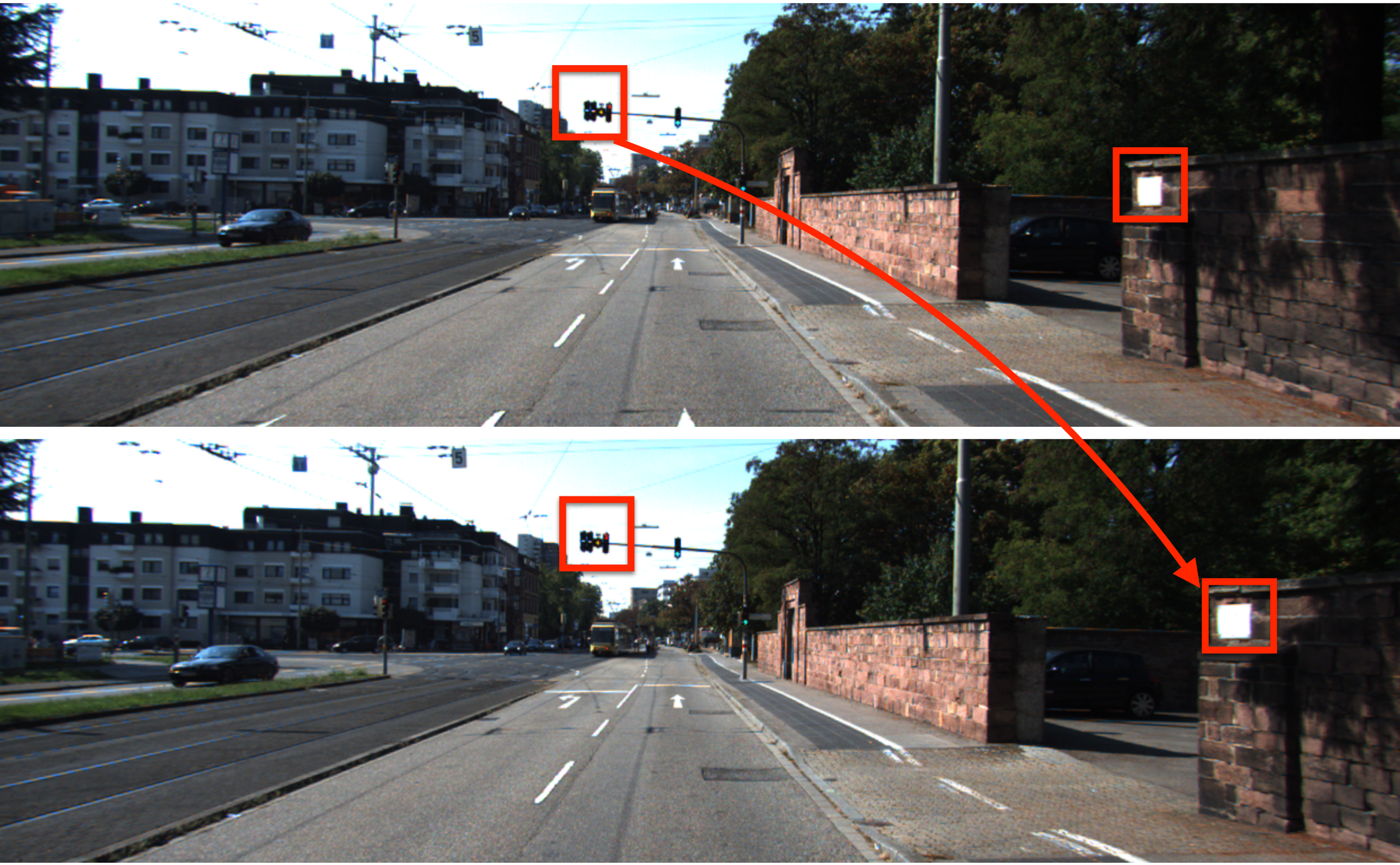
Problem1: Which points are suitable candidates?

Problem2: How to measure similarity?

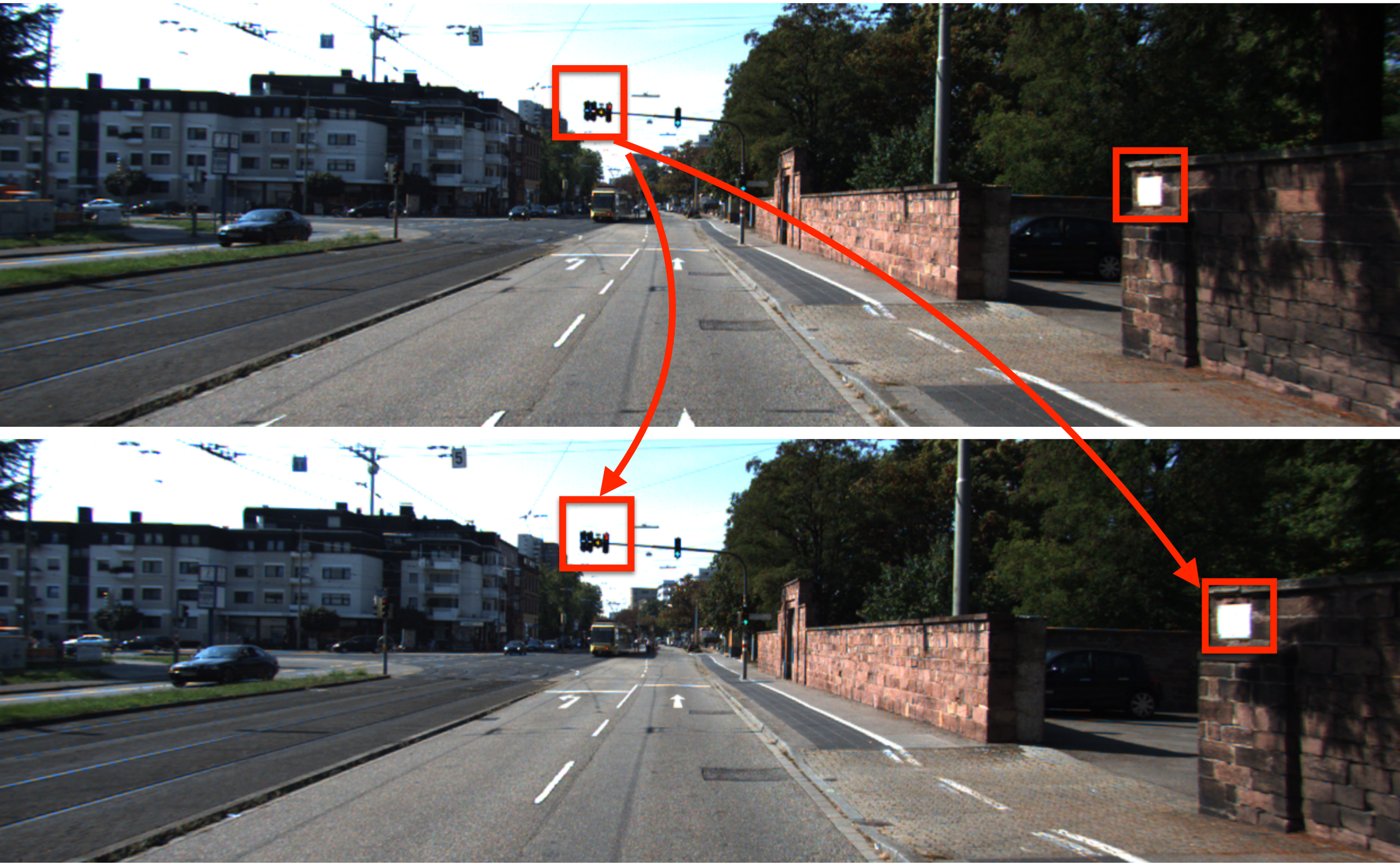
Problem2: How to measure similarity?



Problem2: How to measure similarity?



Problem2: How to measure similarity?



Problem2: How to measure similarity?

$I(\mathbf{x}) =$

$J(\mathbf{x}) =$

pixel values: $\sum_{\mathbf{x} \in \mathcal{W}} (I(\mathbf{x}) - J(\mathbf{x}))^2$

extremely illumination and pose sensitive !!!

=>replace pixel values by less sensitive descriptor

Problem2: How to measure similarity?
BRIEF descriptor [Calonder ECCV 2010]

$$I(\mathbf{x}) =$$

Problem2: How to measure similarity?
BRIEF descriptor [Calonder ECCV 2010]

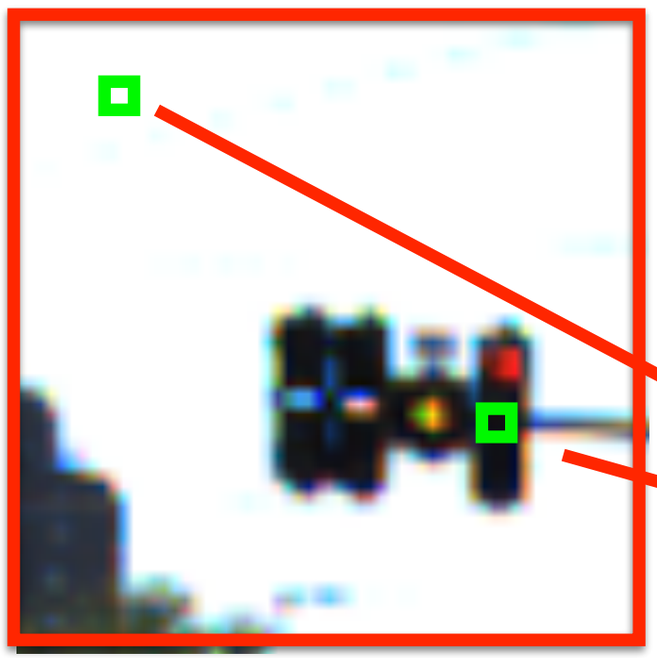
$$I(\mathbf{x}) =$$

Define test on image patch:

$$\tau(I, \mathbf{x}, \mathbf{y}) = \begin{cases} 1, & \text{if } I(\mathbf{x}) < I(\mathbf{y}) \\ 0, & \text{otherwise} \end{cases}$$

Problem2: How to measure similarity?
BRIEF descriptor [Calonder ECCV 2010]

$$I(\mathbf{x}) =$$



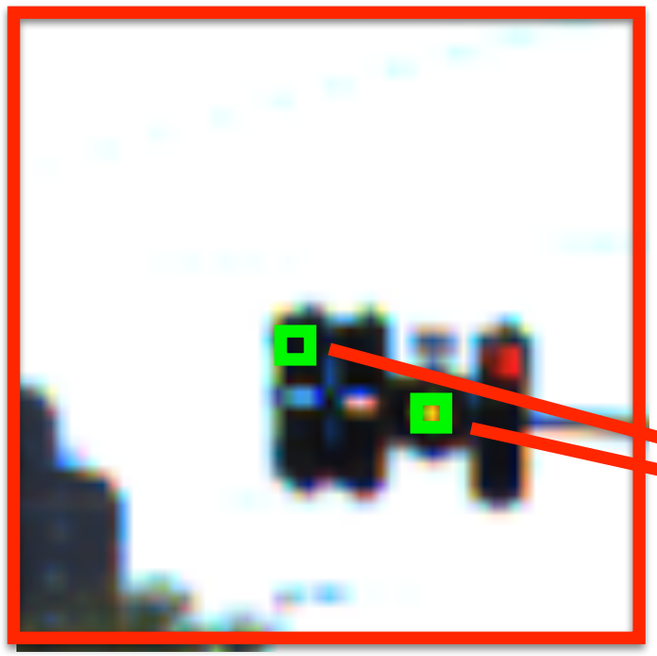
Define test on image patch:

$$\tau(I, \mathbf{x}, \mathbf{y}) = \begin{cases} 1, & \text{if } I(\mathbf{x}) < I(\mathbf{y}) \\ 0, & \text{otherwise} \end{cases}$$

1

Problem2: How to measure similarity?
BRIEF descriptor [Calonder ECCV 2010]

$$I(\mathbf{x}) =$$



Define test on image patch:

$$\tau(I, \mathbf{x}, \mathbf{y}) = \begin{cases} 1, & \text{if } I(\mathbf{x}) < I(\mathbf{y}) \\ 0, & \text{otherwise} \end{cases}$$

1 0

Problem2: How to measure similarity?
BRIEF descriptor [Calonder ECCV 2010]

$$I(\mathbf{x}) =$$

Define test on image patch:

$$\tau(I, \mathbf{x}, \mathbf{y}) = \begin{cases} 1, & \text{if } I(\mathbf{x}) < I(\mathbf{y}) \\ 0, & \text{otherwise} \end{cases}$$

N tests form an N-dimensional binary vector (e.g. 1001011)

Problem2: How to measure similarity?
BRIEF descriptor [Calonder ECCV 2010]

$$I(\mathbf{x}) =$$

Define test on image patch:

$$\tau(I, \mathbf{x}, \mathbf{y}) = \begin{cases} 1, & \text{if } I(\mathbf{x}) < I(\mathbf{y}) \\ 0, & \text{otherwise} \end{cases}$$

N tests form an N-dimensional binary vector (e.g. 1001011)

BRIEF descriptor is decimal representation of this binary code

$$f(I) = \sum_{i=1}^{N} 2^{i-1} \tau(I, \mathbf{x}_i, \mathbf{y}_i)$$

Problem2: How to measure similarity?
Simplified matching with BRIEF descriptor

Input: Randomly defined N binary tests + image patches I_k, J_k

1. Smooth each image patch

Problem2: How to measure similarity?
Simplified matching with BRIEF descriptor

Input: Randomly defined N binary tests + image patches I_k, J_k

1. Smooth each image patch
2. For each image patch compute BRIEF descriptor:

$$f(I_k) = \sum_{i=1}^{N} 2^{i-1} \tau(I_k, \mathbf{x}_i, \mathbf{y}_i) \quad f(J_k) = \sum_{i=1}^{N} 2^{i-1} \tau(J_k, \mathbf{x}_i, \mathbf{y}_i)$$

Problem2: How to measure similarity?
Simplified matching with BRIEF descriptor

Input: Randomly defined N binary tests + image patches I_k, J_k

1. Smooth each image patch
2. For each image patch compute BRIEF descriptor:

$$f(I_k) = \sum_{i=1}^{N} 2^{i-1} \tau(I_k, \mathbf{x}_i, \mathbf{y}_i) \quad f(J_k) = \sum_{i=1}^{N} 2^{i-1} \tau(J_k, \mathbf{x}_i, \mathbf{y}_i)$$

3. For each image patch I from the first image find the most similar patch in the second image.

$$\arg\min_k (f(I) - f(J_k))^2$$

Problem2: How to measure similarity?
Simplified matching with BRIEF descriptor

Input: Randomly defined N binary tests + image patches I_k, J_k

1. Smooth each image patch
2. For each image patch compute BRIEF descriptor:

$$f(I_k) = \sum_{i=1}^{N} 2^{i-1} \tau(I_k, \mathbf{x}_i, \mathbf{y}_i) \quad f(J_k) = \sum_{i=1}^{N} 2^{i-1} \tau(J_k, \mathbf{x}_i, \mathbf{y}_i)$$

3. For each image patch I from the first image find the most similar patch in the second image.

$$\arg\min_k (f(I) - f(J_k))^2$$

4. Optionally reject dissimilar patches (outliers) and those whose the second similar patch is not sufficiently dissimilar.

Output: 2D-2D correspondences

Summary: Search for correspondences in camera images

1. Find corner points (FASTs, harris corners)
2. Compute descriptor (BRIEF)
3. Estimate 2D-2D correspondences

Summary: Search for correspondences in camera images

1. Find corner points (FASTs, harris corners)
2. Compute descriptor (BRIEF)
3. Estimate 2D-2D correspondences

- Commonly used version is ORB-detector [Rublee ICCV 2011]
- It is FAST+BRIEF+few hacks. OpenCV implementation:

```
# initialise
orb = cv2.ORB_create()
# find points and compute descriptors
kp1, des1 = orb.detectAndCompute(img1,None)
kp2, des2 = orb.detectAndCompute(img2,None)
# match descriptors
bf =cv2.BFMatcher(cv2.NORM_HAMMING,crossCheck=True)
matches = bf.match(des1,des2)
```

http://www.willowgarage.com/sites/default/files/orb_final.pdf

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_orb/py_orb.html

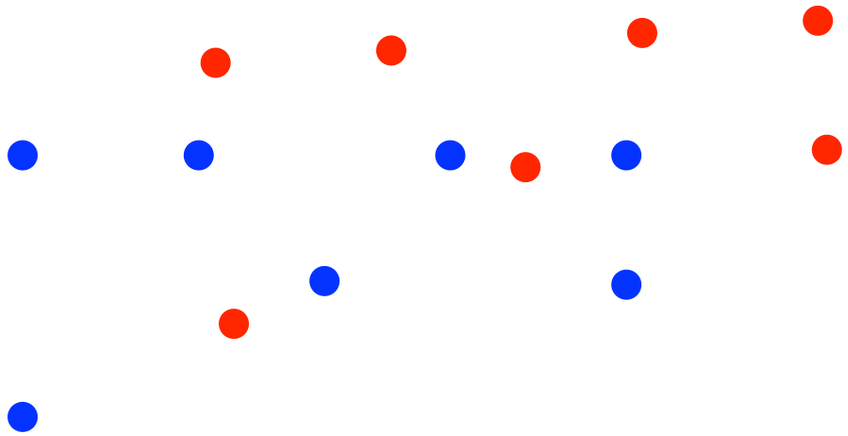
Simplified mapping pipeline overview

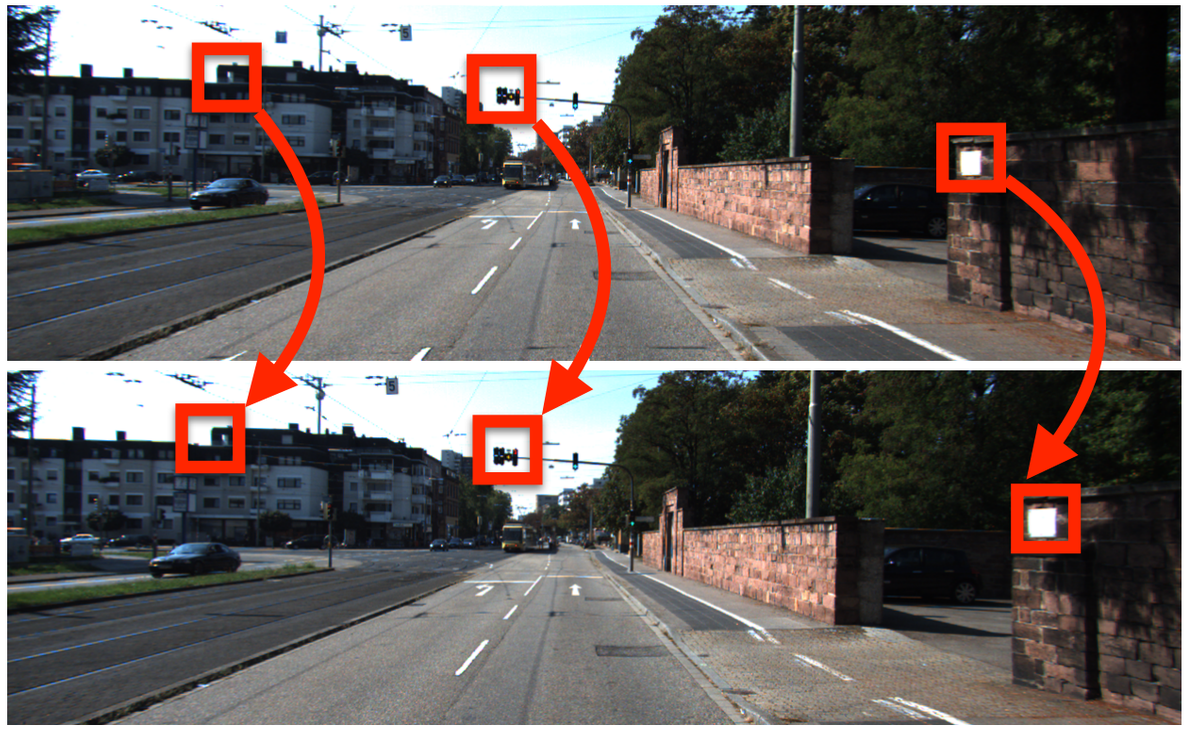
Input: Two consecutive images and two 3D pointclouds

1. Estimate 2D-2D correspondences (e.g. ORB-detector)
2. Project pointclouds and estimate 3D-3D correspondences
3. Align pointclouds and estimate $\mathbf{R}, \mathbf{t}$ (motion of the robot)
4. Merge aligned point clouds to create a map.
5. Use the map as initial pointcloud and repeat from 1

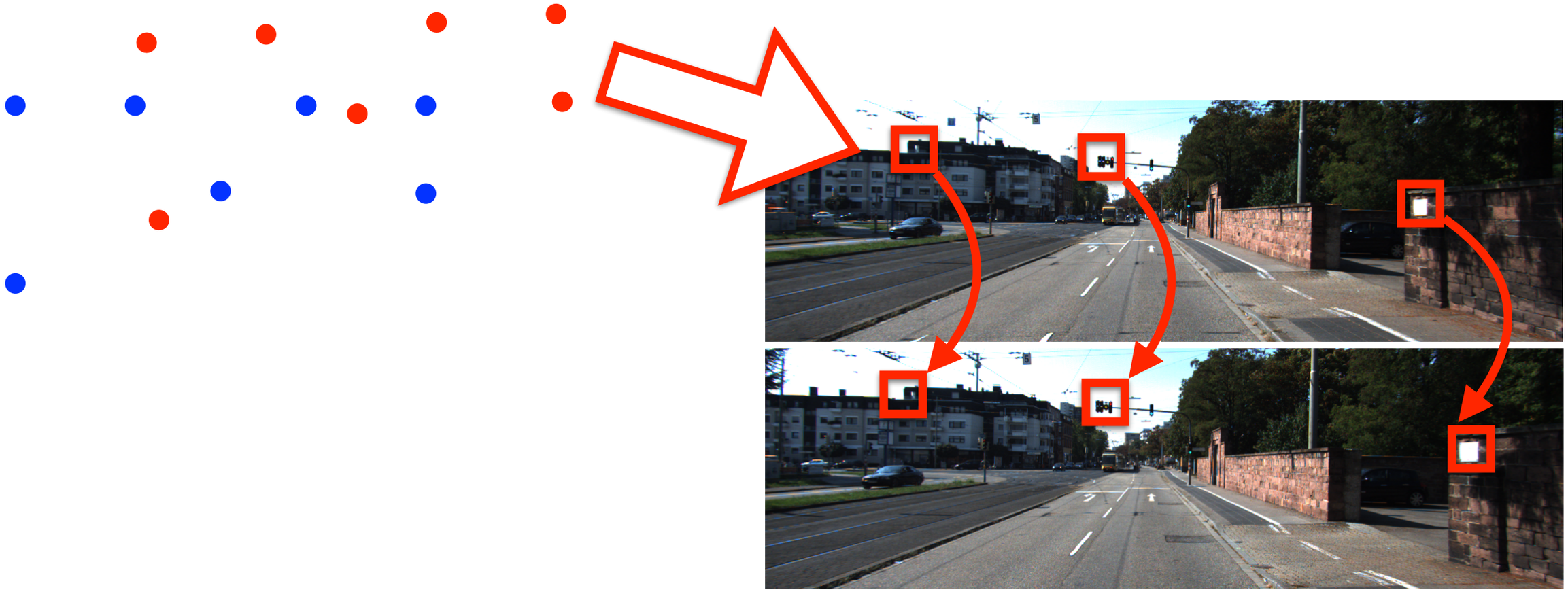
Output: motion of the robot and map

Estimate 2D-2D correspondences

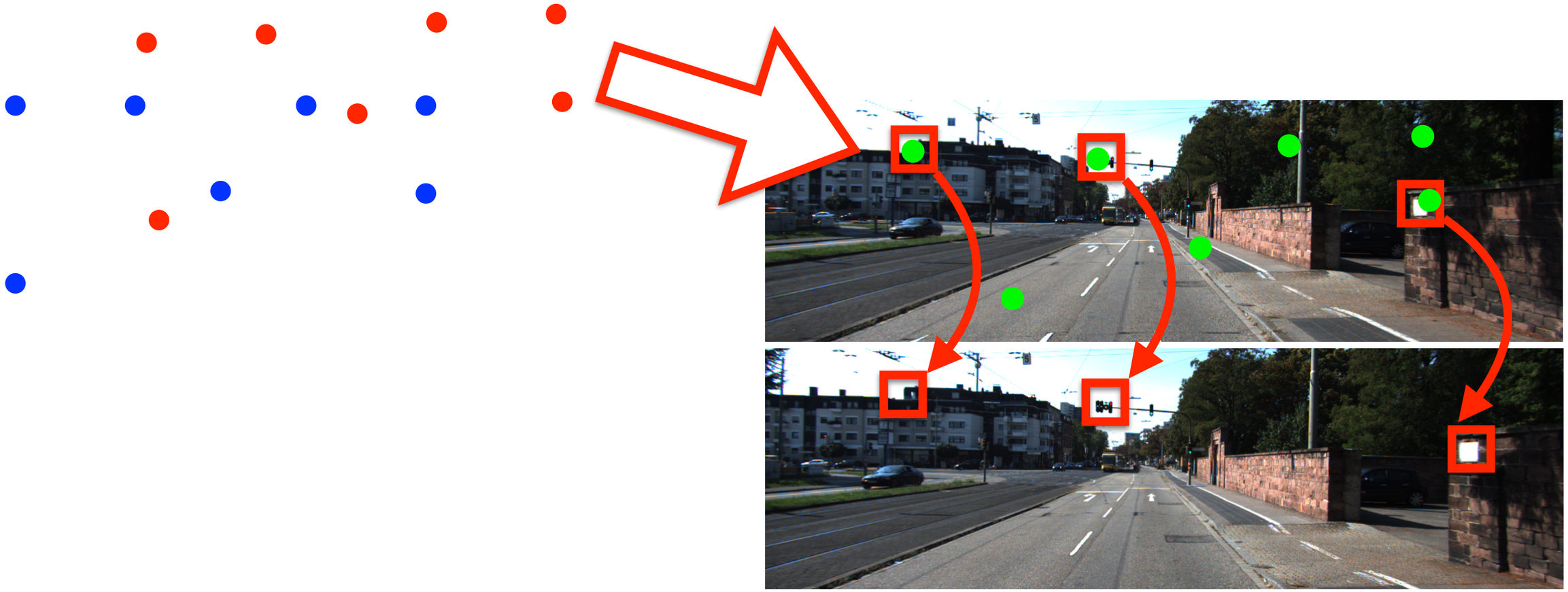




Project 3D pointclouds

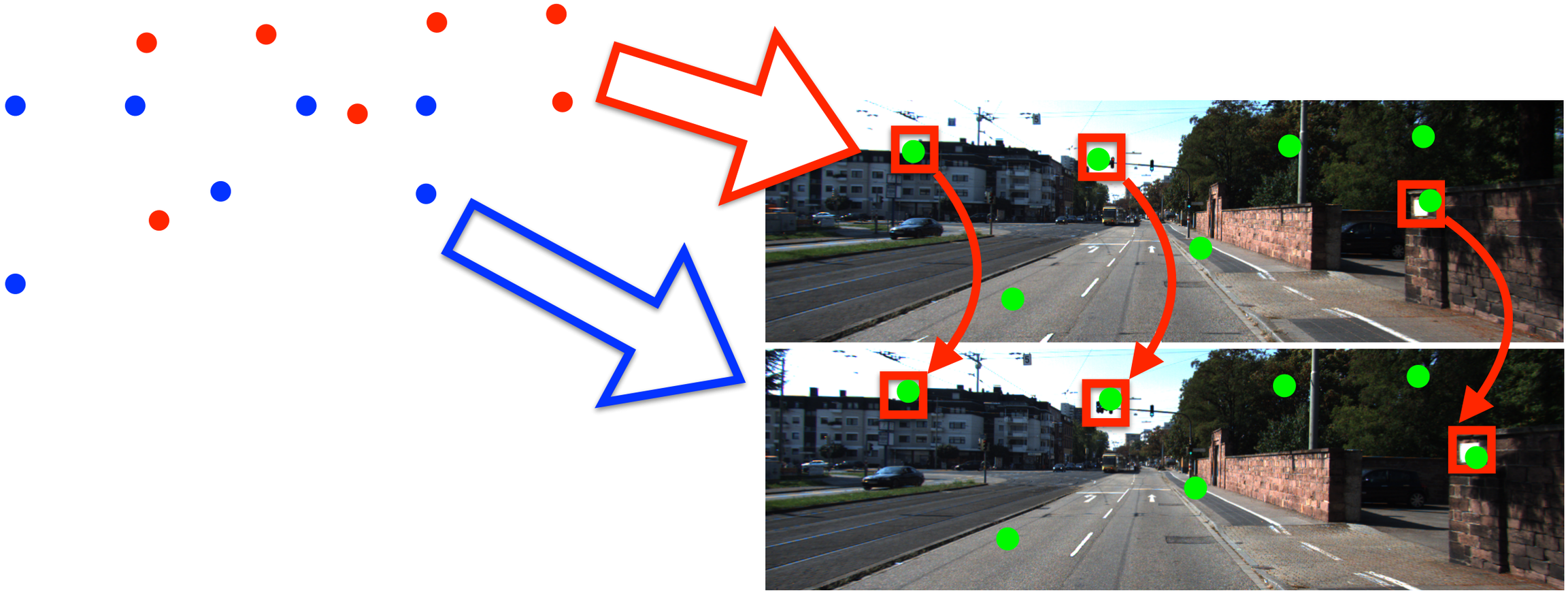


Project 3D pointclouds

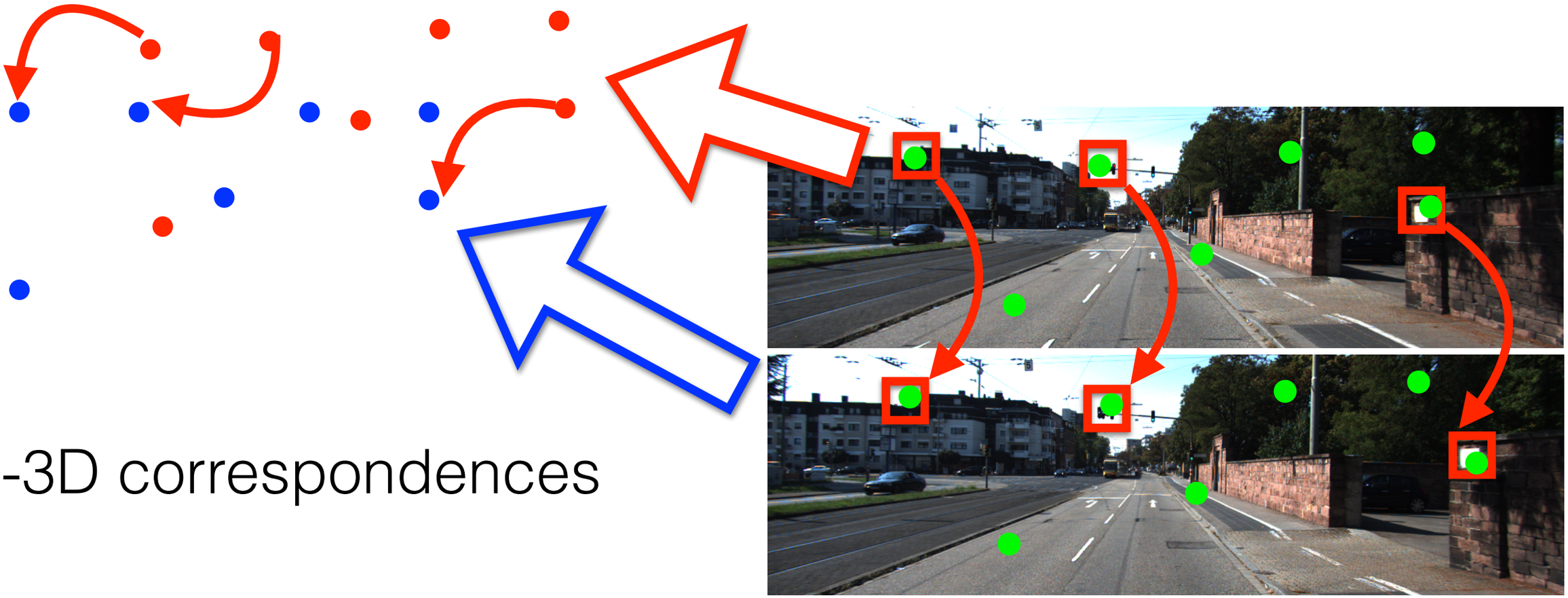


Project 3D pointcloud on image plane

Project 3D pointclouds



For each 2D-2D correspondence, find closest lidar point

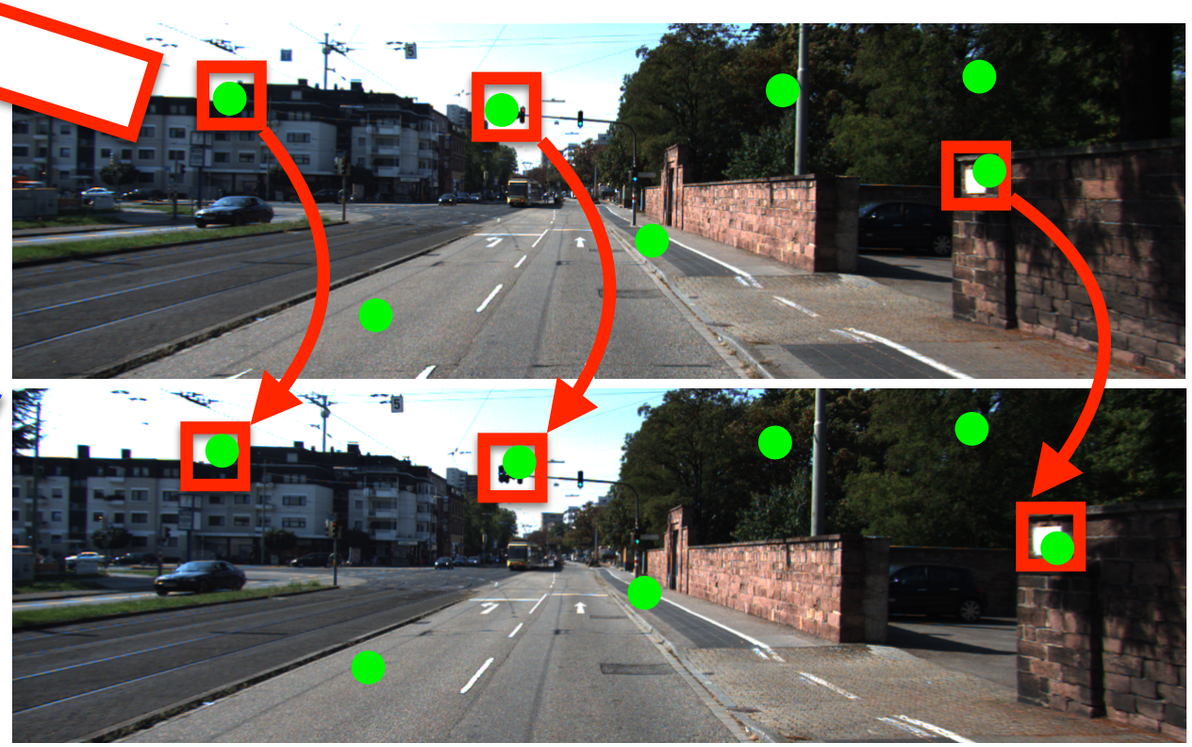


3D-3D correspondences

2D-2D correspondences

Align 3D-3D correspondences

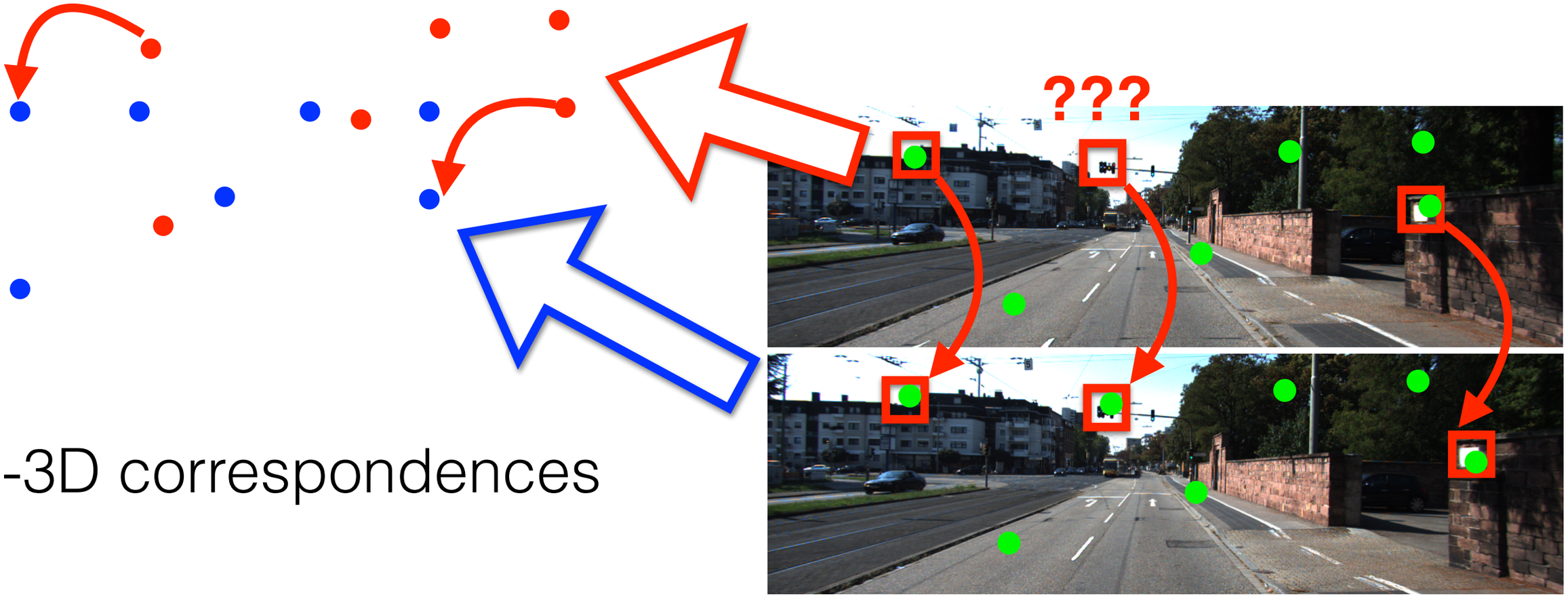
3D-3D correspondences



2D-2D correspondences

Do you see any catch?

What if 3D point for a 2D-2D correspondence is not available?

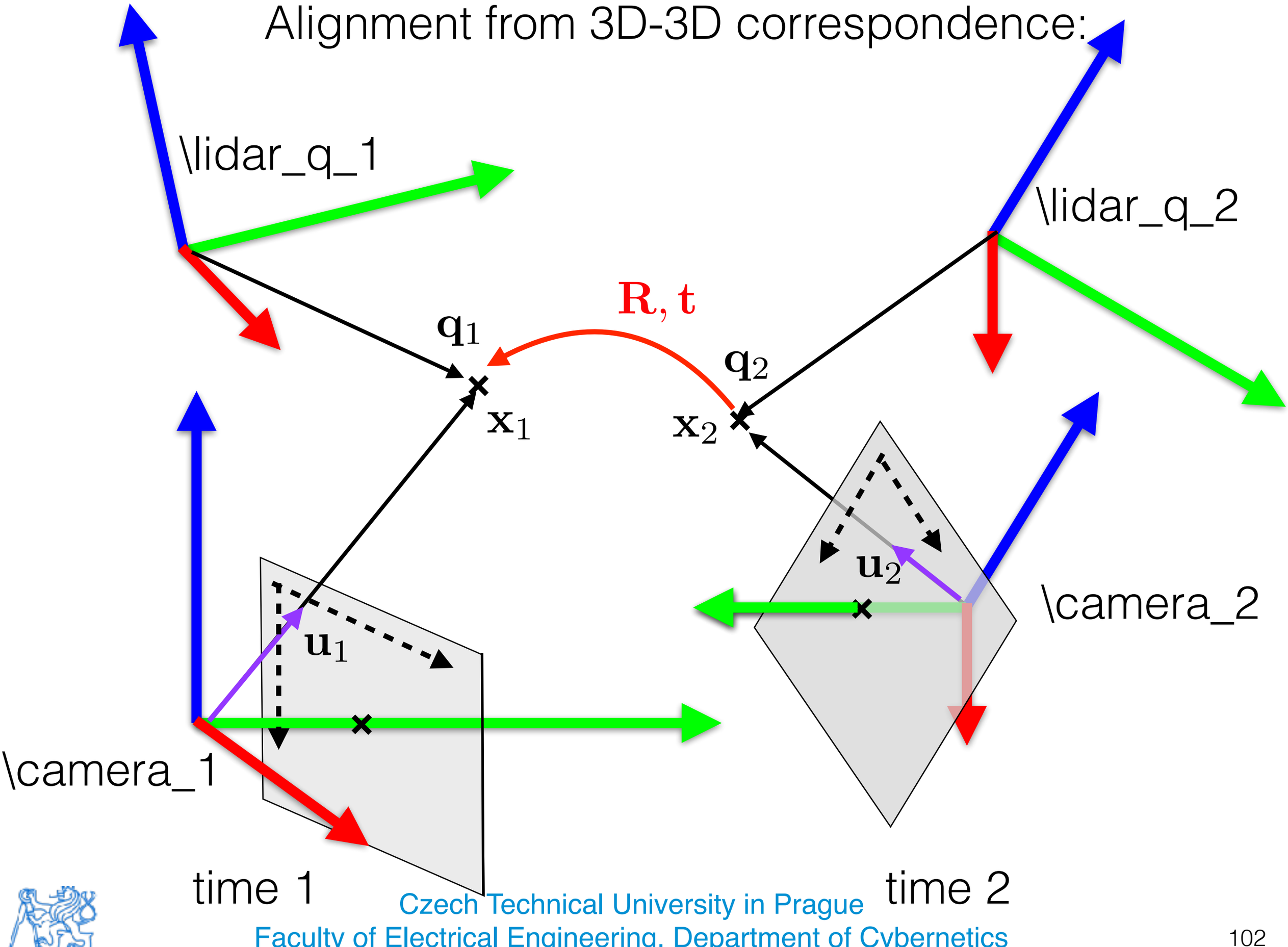


3D-3D correspondences

2D-2D correspondences

Why 3D measurement could be missing?

Alignment from 3D-3D correspondence:



Alignment from 3D-3D correspondence:

- Search for transformation which maps 3D-3D correspondence on the same point

$$\mathbf{x}_1 = \mathbf{R}\mathbf{x}_2 + \mathbf{t}$$

- Each 3D-3D correspondence => 3 equations:

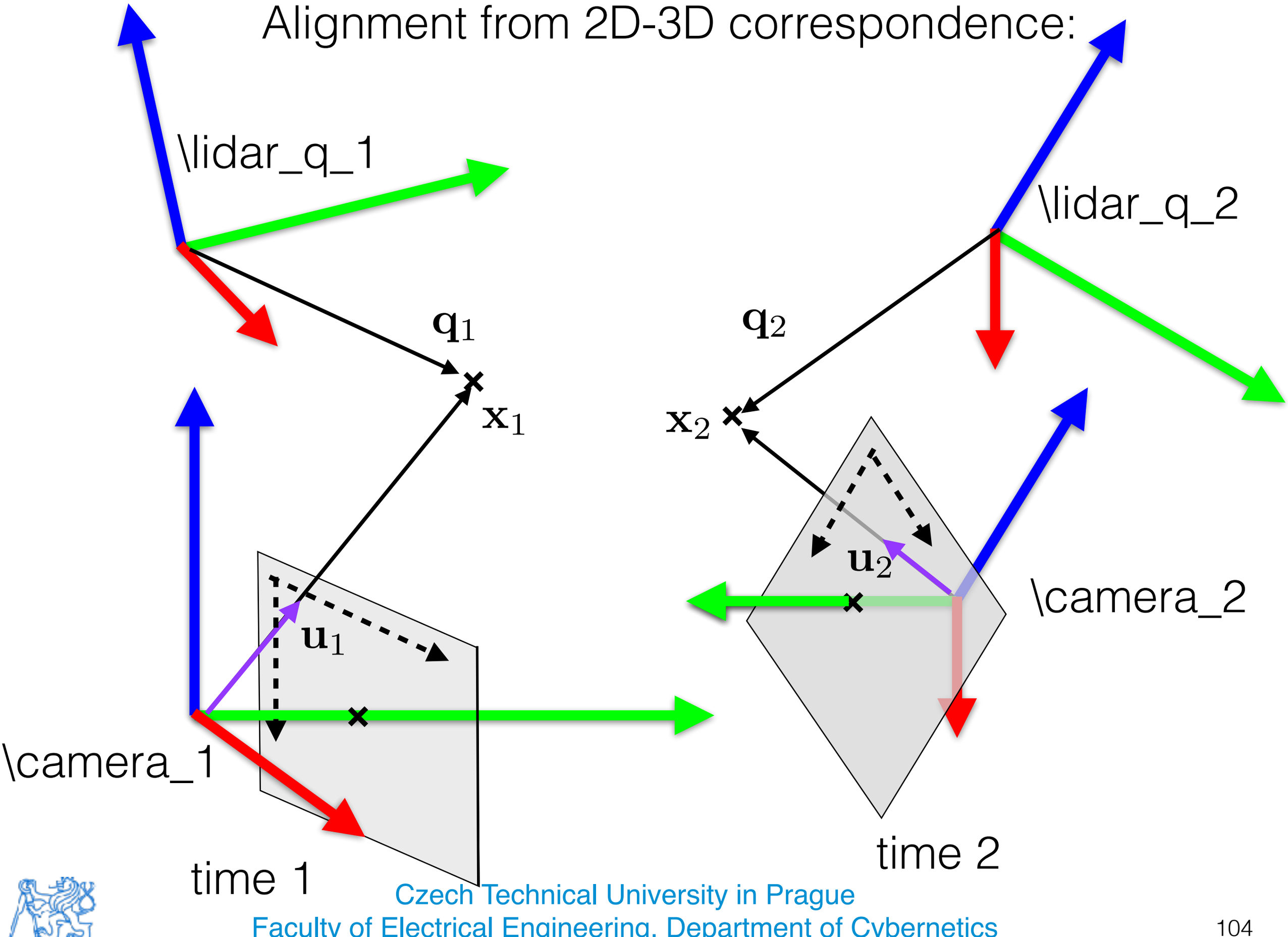
$$f_{33}(\mathbf{R}, \mathbf{t}) = \mathbf{x}_1 - (\mathbf{R}\mathbf{x}_2 + \mathbf{t}) = 0$$

- ML estimate under Gaussian noise is

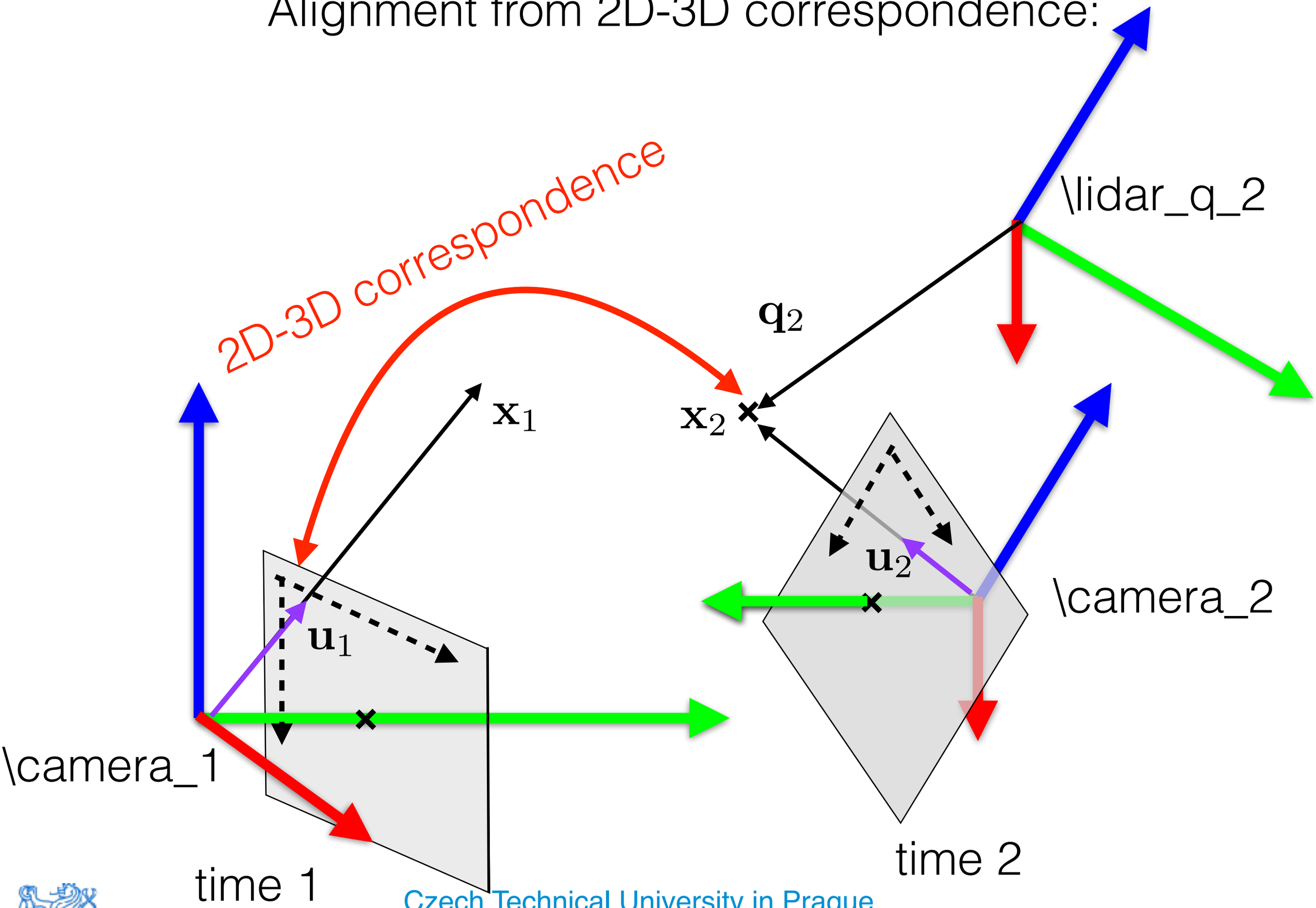
$$\arg\min_{\mathbf{R},\mathbf{t}} \| f_{33}(\mathbf{R}, \mathbf{t}) \|_2^2$$

- One 3D-3D correspondece = > 3 equations

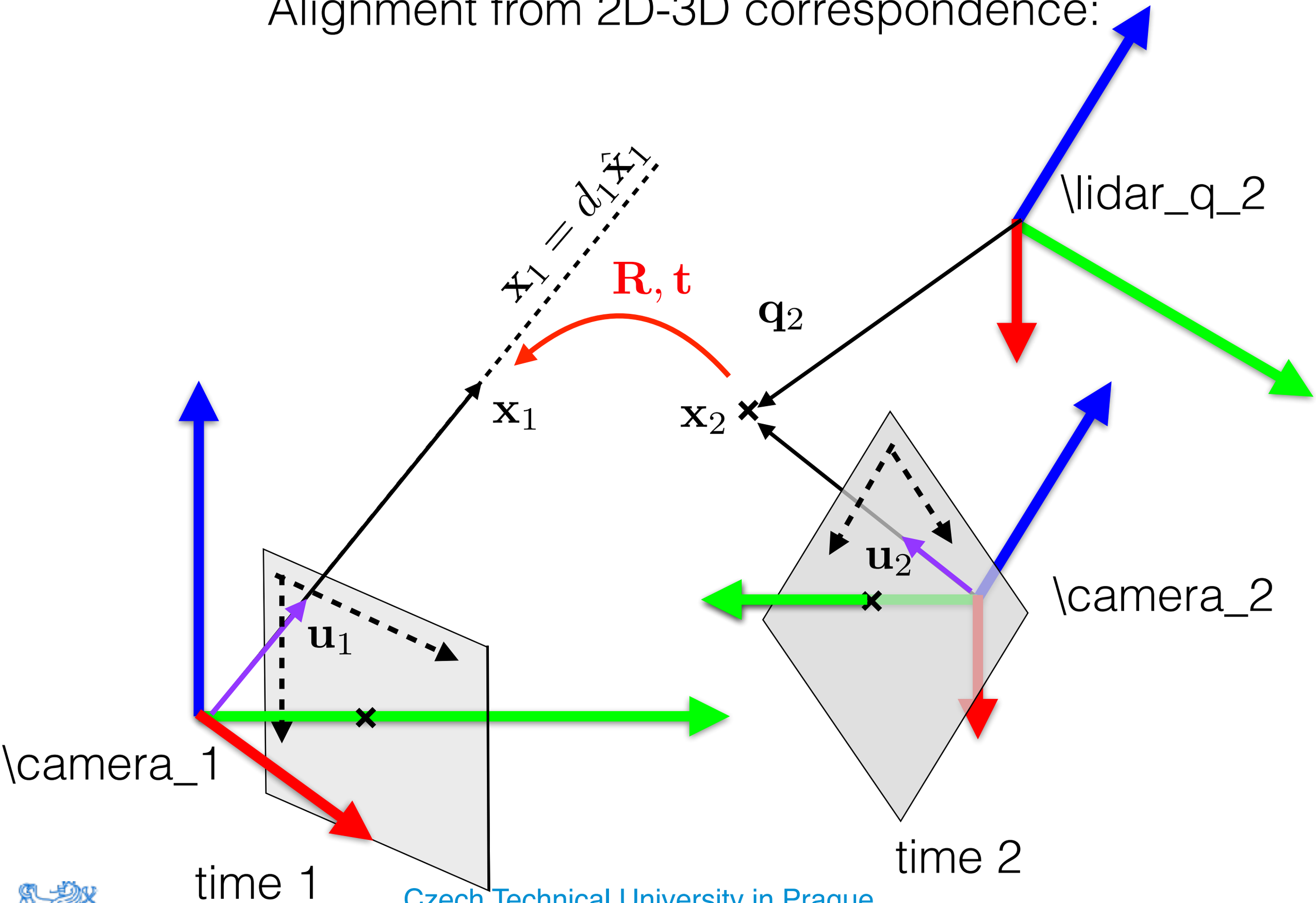
Alignment from 2D-3D correspondence:



Alignment from 2D-3D correspondence:



Alignment from 2D-3D correspondence:



Alignment from 2D-3D correspondence:

Camera point in time 1: $\mathbf{u}_1 \in \mathcal{R}^2$
Lidar point in time 2: $\mathbf{q}_2 \in \mathcal{R}^3$

1.Transform lidar pointcloud to the camera c.f. (known depth)

$$\mathbf{x}_2 = [\mathbf{R}_c \mid \mathbf{t}_c]\,\overline{\mathbf{q}}_2$$

2.Corresponding point in time 1 lies on the ray (unknown depth)

$$\hat{\mathbf{x}}_1 = \mathbf{K}^{-1}\mathbf{u}_1 / \|\mathbf{K}^{-1}\mathbf{u}_1\|$$

3.Search for transformation with unknown depth in camera

$$d_1\hat{\mathbf{x}}_1 = \mathbf{R}\mathbf{x}_2 + \mathbf{t}$$

Alignment from 2D-3D correspondence:

Camera point in time 1: $\mathbf{u}_1 \in \mathcal{R}^2$
Lidar point in time 2: $\mathbf{q}_2 \in \mathcal{R}^3$

1.Transform lidar pointcloud to the camera c.f. (known depth)

$$\mathbf{x}_2 = [\mathbf{R}_c \mid \mathbf{t}_c]\,\overline{\mathbf{q}}_2$$

2.Corresponding point in time 1 lies on the ray (unknown depth

$$\hat{\mathbf{x}}_1 = \mathbf{K}^{-1}\mathbf{u}_1 / \|\mathbf{K}^{-1}\mathbf{u}_1\|$$

3.Search for transformation with unknown depth in camera

$$d_1\hat{\mathbf{x}}_1 = \mathbf{R}\mathbf{x}_2 + \mathbf{t}$$

which aligns 3D point on the ray (for some depth)

Alignment from 2D-3D correspondence:

3.Search for transformation with unknown depth in camera

$$d_1 \hat{\mathbf{x}}_1 = \mathbf{R}\mathbf{x}_2 + \mathbf{t}$$

$$\text{(a)} \quad d_1 \hat{x}_1 = \mathbf{R}_1^\top \mathbf{x}_2 + t_1$$

$$\text{(b)} \quad d_1 \hat{y}_1 = \mathbf{R}_2^\top \mathbf{x}_2 + t_2$$

$$\text{(c)} \quad d_1 \hat{z}_1 = \mathbf{R}_3^\top \mathbf{x}_2 + t_3$$

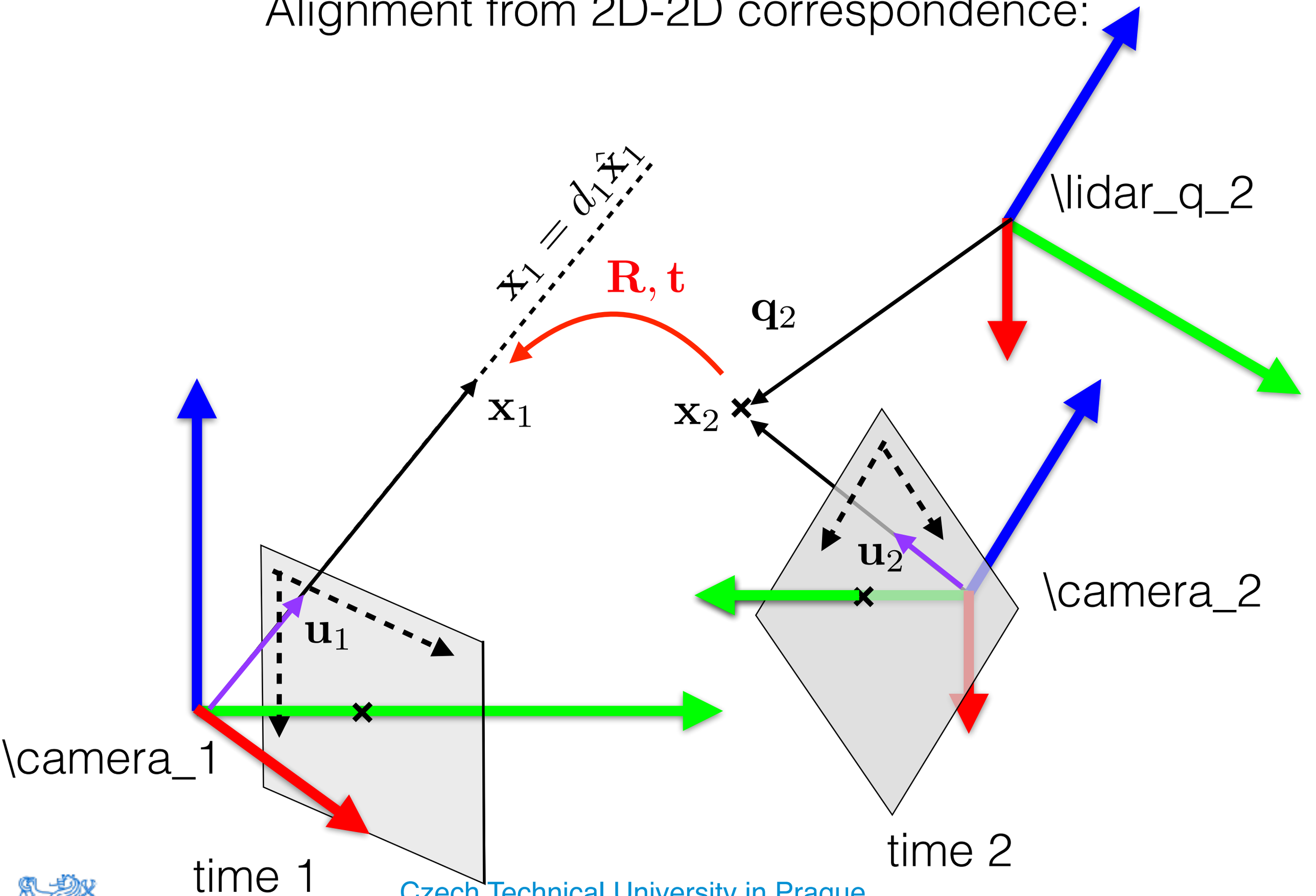
$$\text{(a)/(c)} \quad \hat{x}_1(\mathbf{R}_3^\top z_2 + t_3) - \hat{z}_1(\mathbf{R}_1^\top \mathbf{x}_2 + t_1) = 0$$

$$\text{(b)/(c)} \quad \hat{y}_1(\mathbf{R}_3^\top z_2 + t_3) - \hat{z}_1(\mathbf{R}_2^\top \mathbf{x}_2 + t_2) = 0$$

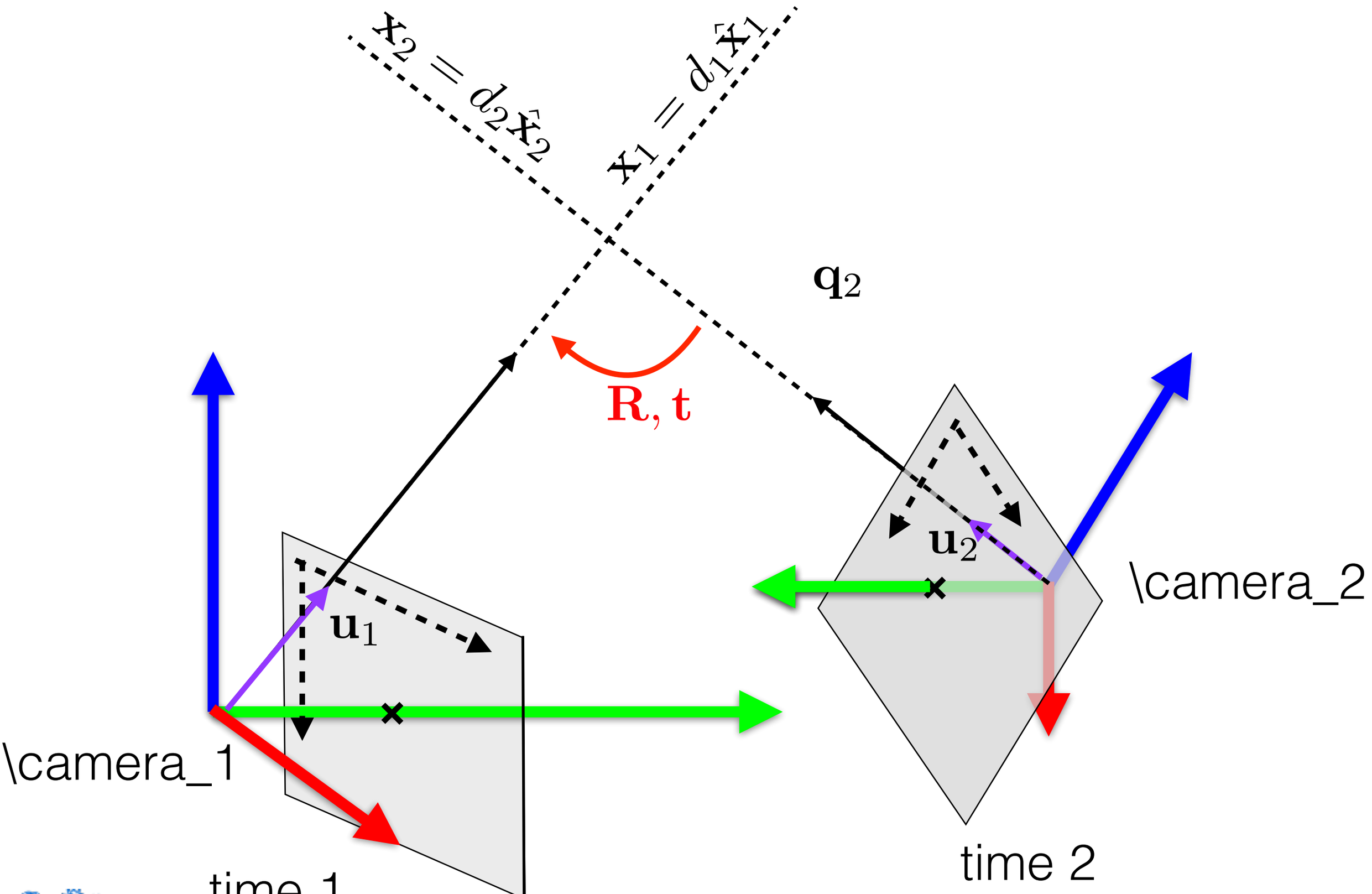
$$\arg\min_{\mathbf{R},\mathbf{t}} \|f_{23}(\mathbf{R}, \mathbf{t})\|_2^2$$

- One 2D-3D correspondece = > 2 equations

Alignment from 2D-2D correspondence:



Alignment from 2D-2D correspondence:



Alignment from 2D-2D correspondence:

Search for transformation with unknown depth in cameras

$$d_1\hat{\mathbf{x}}_1 = \mathbf{R}d_2\hat{\mathbf{x}}_2 + \mathbf{t}$$ ← unknowns

Alignment from 2D-2D correspondence:

Search for transformation with unknown depth in cameras

$$d_1 \hat{\mathbf{x}}_1 = \mathbf{R} d_2 \hat{\mathbf{x}}_2 + \mathbf{t}$$

---

get rid of depths

(a) $d_1 \hat{x}_1 = \mathbf{R}_1^\top d_2 \hat{\mathbf{x}}_2 + t_1$

(b) $d_1 \hat{y}_1 = \mathbf{R}_2^\top d_2 \hat{\mathbf{x}}_2 + t_2$

(c) $d_1 \hat{z}_1 = \mathbf{R}_3^\top d_2 \hat{\mathbf{x}}_2 + t_3$

three eq. !!!

$$\begin{bmatrix} -\hat{y}_1 t_3 + \hat{z}_1 t_2 \\ \hat{x}_1 t_3 + \hat{z}_1 t_1 \\ -\hat{x}_1 t_2 + \hat{y}_1 t_1 \end{bmatrix} \mathbf{R}\,\mathbf{x}_2 = 0$$

one eq. !!!

$$\arg\min_{\mathbf{R},\mathbf{t}} \|f_{22}(\mathbf{R},\mathbf{t})\|_2^2$$

- One 2D-2D correspondence = > 1 equation

Alignment from mixed correspondences:

- Typically you have mixed correspondences, due to noise on reflective and semitransparent surfaces and different frame-rate of camera and lidar
- Camera-lidar SLAM reduces to the following optimisation task:

$$\underset{\mathbf{R}\in\mathcal{SO}(3),\mathbf{t}\in\mathcal{R}^3}{\arg\min} \left\| \begin{bmatrix} \sum_i f_{33}^i(\mathbf{R},\mathbf{t}) \\ \sum_i f_{23}^i(\mathbf{R},\mathbf{t}) \\ \sum_i f_{23}^i(\mathbf{R},\mathbf{t}) \\ \sum_i f_{22}^i(\mathbf{R},\mathbf{t}) \end{bmatrix} \right\|_2^2$$

- Optimisation typically tackled by Levenberg-Marquardt https://en.wikipedia.org/wiki/Levenberg–Marquardt_algorithm
- Map is updated by aligning and concatenating the last measurement with the existing map. Missing depths can be reconstructed by backward substitution of $\mathbf{R}, \mathbf{t}$.

Summary: geometry SLAM

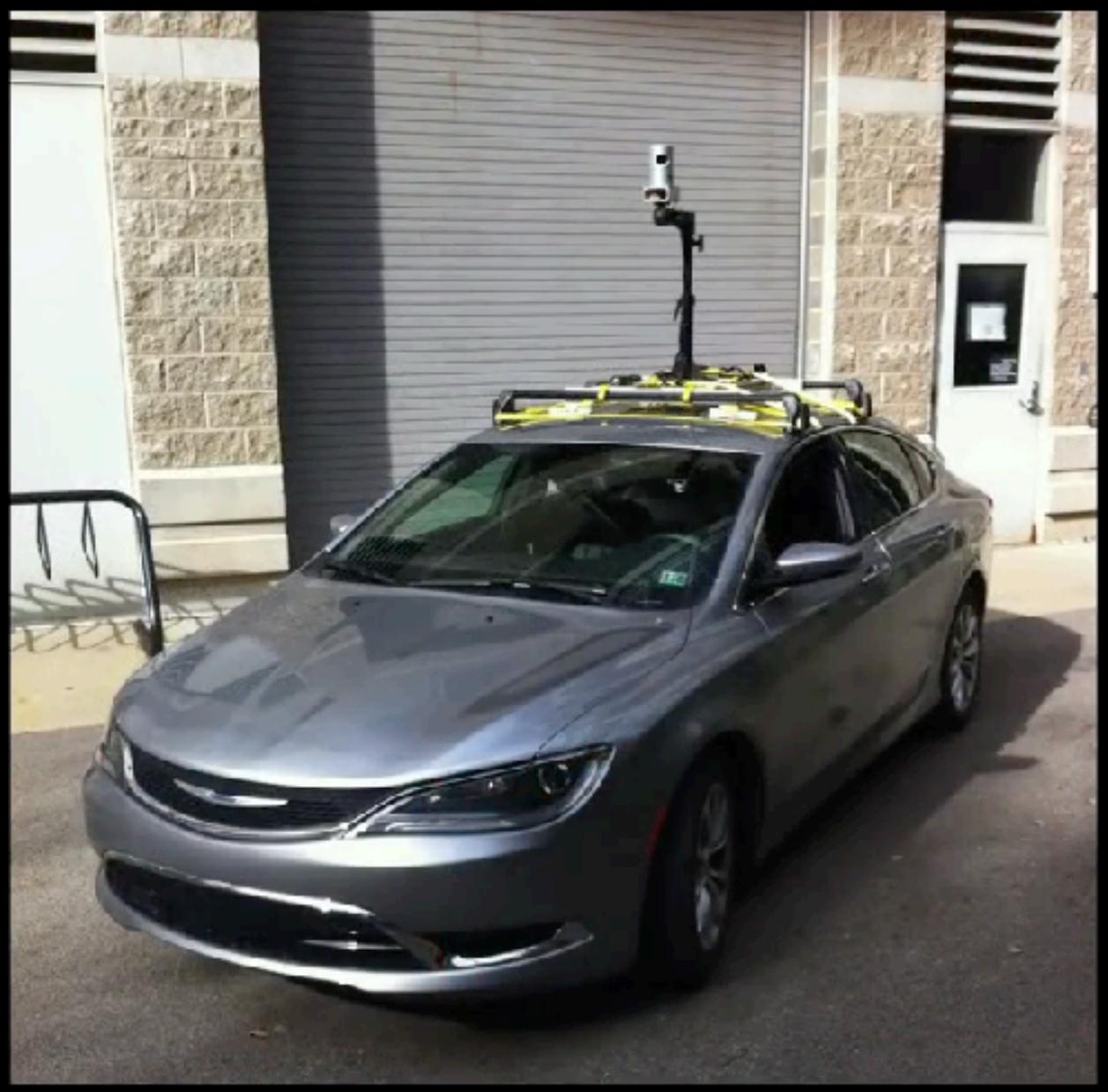
- Typical application self-driving cars (Kitti dataset)
  http://www.cvlibs.net/datasets/kitti/eval_odometry.php

- Winners of Kitti visual odometry challenge:
  [Zhang ICRA 2015]
  https://frc.ri.cmu.edu/~zhangji/publications/ICRA_2015.pdf

Summary: geometry SLAM

- Winners of Kitti visual odometry challenge:
  [Zhang ICRA 2015]

  https://frc.ri.cmu.edu/~zhangji/publications/ICRA_2015.pdf



Special case 0: Known map

- Previously known map (with visual features)
  - 2D-3D correspondences for camera localization
  - 3D-3D correspondences for camer+lidar localization

Special case I: Alignment from 2D-2D correspondences only

- When only camera is available due to costs limitations or lidar failure (rain, snow), the pose can be estimated from 2D-2D correspondences (up to the absolute scale)

$$\underset{\mathbf{R}\in\mathcal{SO}(3),\mathbf{t}\in\mathcal{R}^3}{\arg\min} \left\| \sum_i f_{22}^i(\mathbf{R},\mathbf{t}) \right\|_2^2$$

- Several explanations:
  A. only directions towards the real 3D point are known => scale of the world/translations is arbitrary
  B. if “something” is zero, then “alpha”x“something” is also zero (see following slides)

Special case I: Alignment from 2D-2D correspondences only

- When only camera is available due to costs limitations or lidar failure (rain, snow), the pose can be estimated from 2D-2D correspondences (up to the absolute scale)

$$\underset{\mathbf{R}\in\mathcal{SO}(3),\mathbf{t}\in\mathcal{R}^3}{\arg\min} \left\| \sum_i f_{22}^i(\mathbf{R},\mathbf{t}) \right\|_2^2$$

- Several explanations:
  A. only directions towards the real 3D point are known => scale of the world/translations is arbitrary
  B. if “something” is zero, then “alpha”x“something” is also zero (see following slides)

Special case I: Alignment from 2D-2D correspondences only

Search for transformation with unknown depth in cameras

$$d_1 \hat{\mathbf{x}}_1 = \mathbf{R} d_2 \hat{\mathbf{x}}_2 + \mathbf{t}$$

(a) $d_1 \hat{x}_1 = \mathbf{R}_1^\top d_2 \hat{\mathbf{x}}_2 + t_1$

(b) $d_1 \hat{y}_1 = \mathbf{R}_2^\top d_2 \hat{\mathbf{x}}_2 + t_2$ three eq. !!!

(c) $d_1 \hat{z}_1 = \mathbf{R}_3^\top d_2 \hat{\mathbf{x}}_2 + t_3$

$$\begin{bmatrix} -\hat{y}_1 t_3 + \hat{z}_1 t_2 \\ \hat{x}_1 t_3 + \hat{z}_1 t_1 \\ -\hat{x}_1 t_2 + \hat{y}_1 t_1 \end{bmatrix} \mathbf{R}\,\hat{\mathbf{x}}_2 = 0$$ one eq. !!!

$$\arg\min_{\mathbf{R},\mathbf{t}} \|f_{22}(\mathbf{R},\mathbf{t})\|_2^2$$

- One 2D-2D correspondence = > 1 equation

Special case I: Alignment from 2D-2D correspondences only

Search for transformation with unknown depth in cameras

$$d_1\hat{\mathbf{x}}_1 = \mathbf{R}d_2\hat{\mathbf{x}}_2 + \mathbf{t}$$

$$\begin{aligned}
&\text{(a)}\ d_1\hat{x}_1 = \mathbf{R}_1^\top d_2\hat{\mathbf{x}}_2 + t_1 \\
&\text{(b)}\ d_1\hat{y}_1 = \mathbf{R}_2^\top d_2\hat{\mathbf{x}}_2 + t_2 \qquad \text{three eq. !!!} \\
&\text{(c)}\ d_1\hat{z}_1 = \mathbf{R}_3^\top d_2\hat{\mathbf{x}}_2 + t_3
\end{aligned}$$

$$\alpha \begin{bmatrix} -\hat{y}_1 t_3 + \hat{z}_1 t_2 \\ \hat{x}_1 t_3 + \hat{z}_1 t_1 \\ -\hat{x}_1 t_2 + \hat{y}_1 t_1 \end{bmatrix} \mathbf{R}\,\hat{\mathbf{x}}_2 = 0 \qquad \text{one eq. !!!}$$

$$\arg\min_{\mathbf{R},\mathbf{t}} \|f_{22}(\mathbf{R},\mathbf{t})\|_2^2$$

- One 2D-2D correspondence = > 1 equation

Special case I: Alignment from 2D-2D correspondences only

Search for transformation with unknown depth in cameras

$$d_1 \hat{\mathbf{x}}_1 = \mathbf{R} d_2 \hat{\mathbf{x}}_2 + \mathbf{t}$$

$$\text{(a)}\ d_1 \hat{x}_1 = \mathbf{R}_1^\top d_2 \hat{x}_2 + t_1$$

$$\text{(b)}\ d_1 \hat{y}_1 = \mathbf{R}_2^\top d_2 \hat{y}_2 + t_2$$

three eq. !!!

$$\text{(c)}\ d_1 \hat{z}_1 = \mathbf{R}_3^\top d_2 \hat{z}_2 + t_3$$

$$\begin{bmatrix} -\hat{y}_1(\alpha \cdot t_3) + \hat{z}_1(\alpha \cdot t_2) \\ \hat{x}_1(\alpha \cdot t_3) + \hat{z}_1(\alpha \cdot t_1) \\ -\hat{x}_1(\alpha \cdot t_2) + \hat{y}_1(\alpha \cdot t_1) \end{bmatrix} \mathbf{R}\,\mathbf{x}_2 = 0$$

one eq. !!!

$$\arg\min_{\mathbf{R},\mathbf{t}} \|f_{22}(\mathbf{R}, \mathbf{t})\|_2^2$$

- One 2D-2D correspondence = > 1 equation

Special case I: Alignment from 2D-2D correspondences only

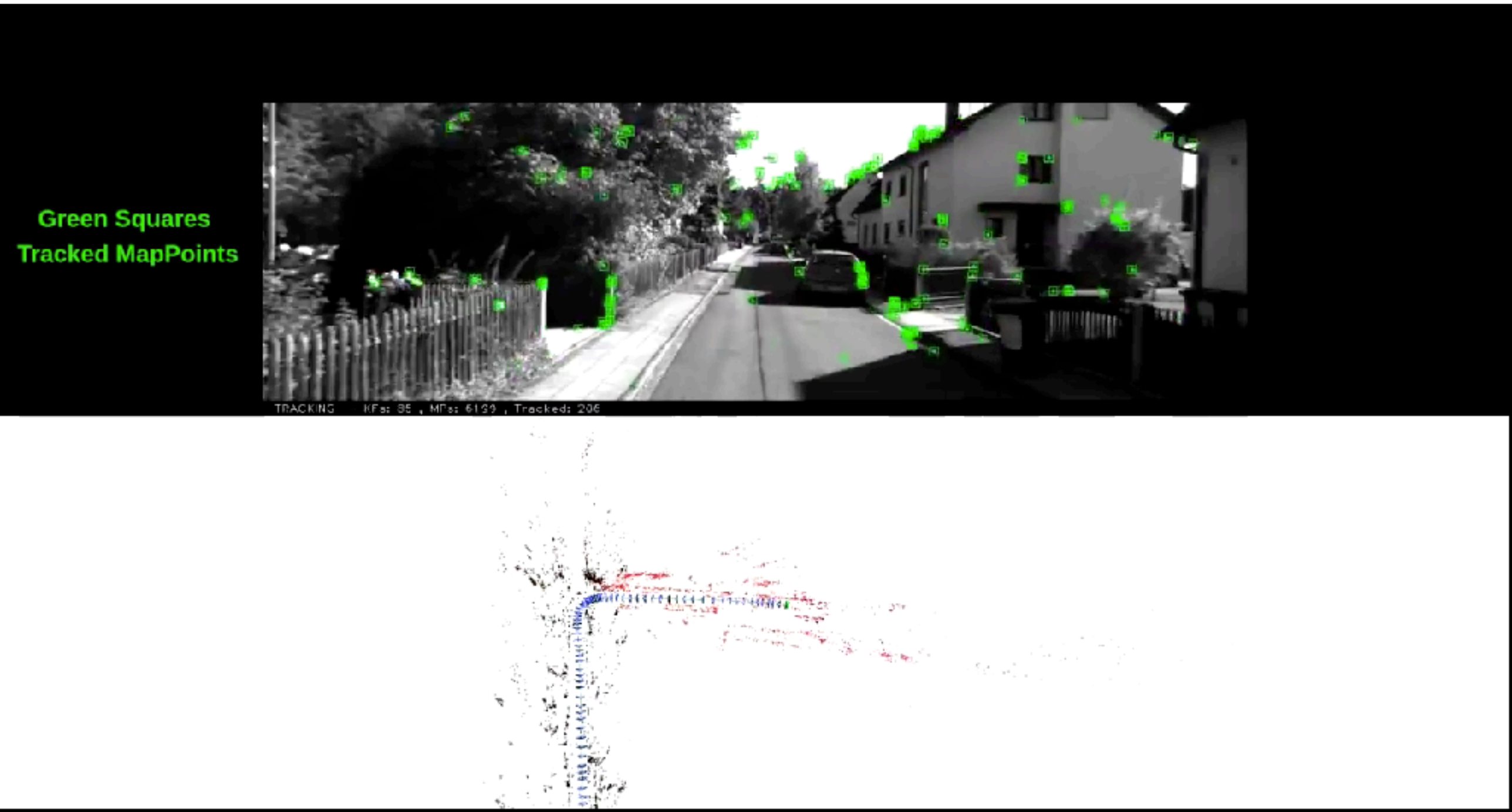
- When only camera is available due to costs limitations or lidar failure (rain, snow), the pose can be estimated from 2D-2D correspondences (up to the absolute scale)

$$\underset{\mathbf{R}\in\mathcal{SO}(3),\mathbf{t}\in\mathcal{R}^3}{\arg\min} \left\| \sum_i f_{22}^i(\mathbf{R},\mathbf{t}) \right\|_2^2$$

- Several explanations:
  A. only directions towards the real 3D point are known => scale of the world/translations is arbitrary
  B. if “something” is zero, then “alpha”x“something” is also zero (see following slides)
- Absolute scale estimated from prior knowledge (e.g. size of a known object).

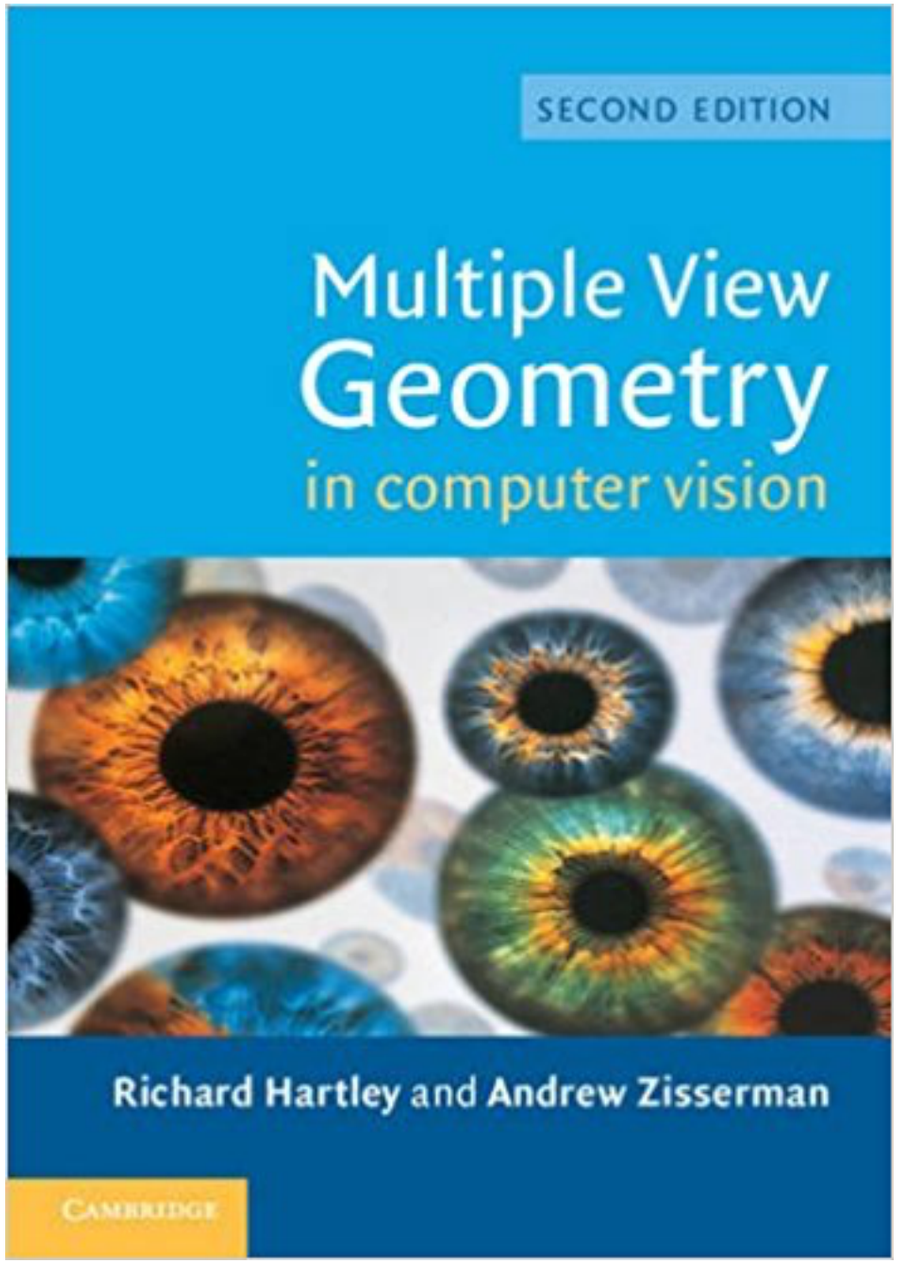
Summary: geometry SLAM

- ORB-SLAM2 RGBD implementation [Raul TRO 2015]: http://webdiis.unizar.es/~raulmur/orbslam/



Summary: geometry SLAM

- Further reading on camera geometry



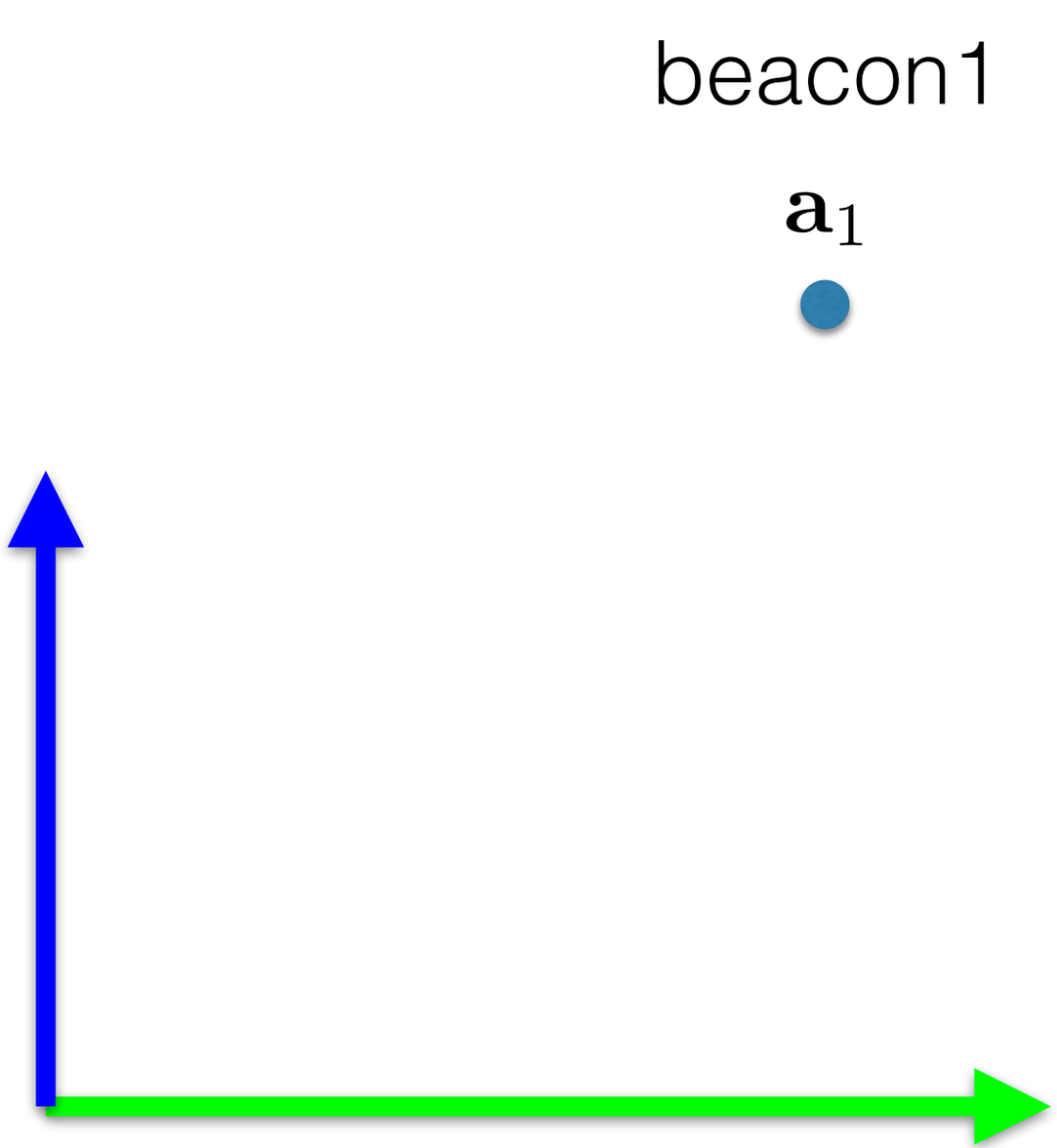
https://www.robots.ox.ac.uk/~vgg/hzbook/

Special case II: Alignment from distance measurements

- SONARDYNE sensors

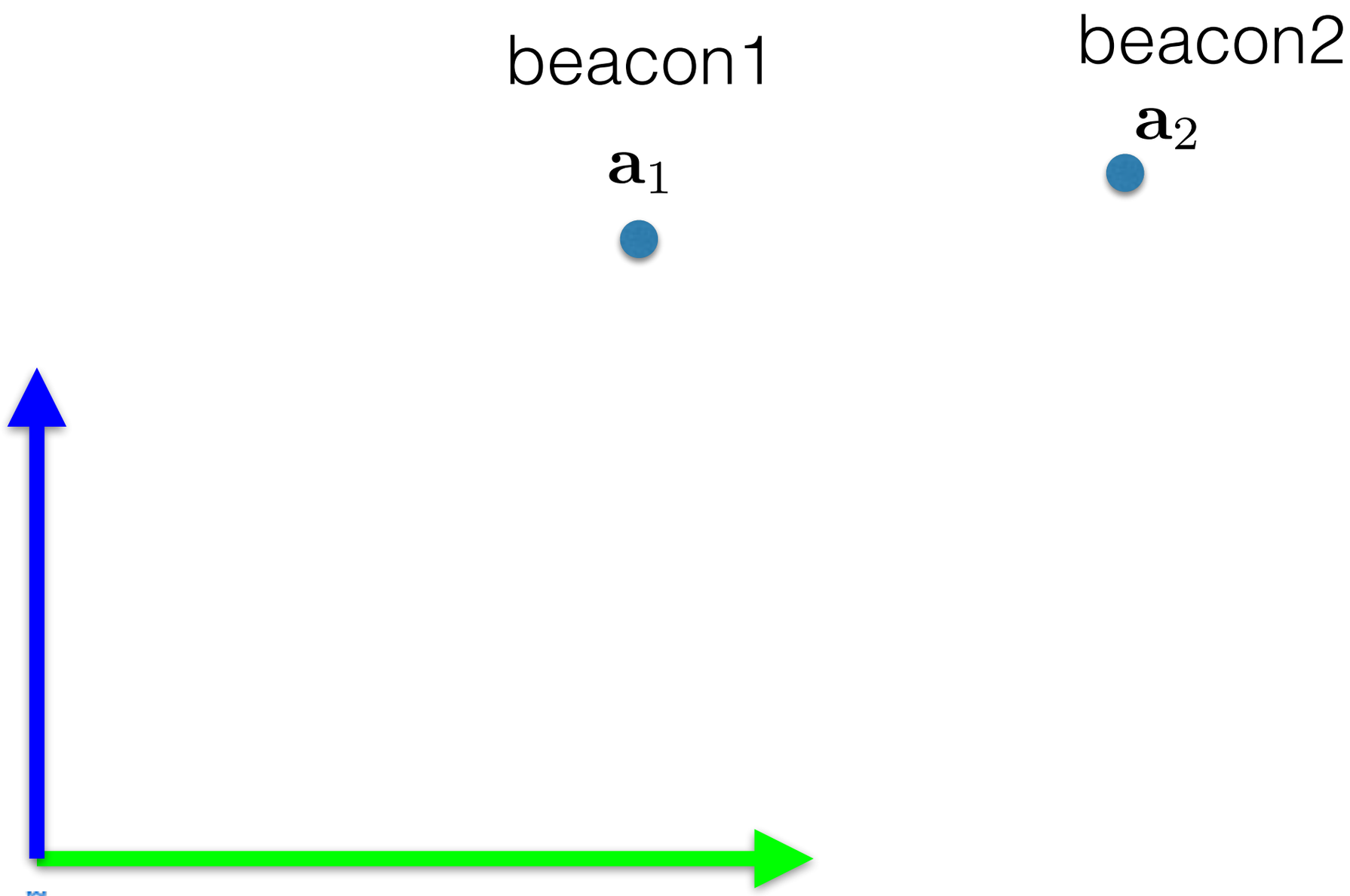
Special case II: Alignment from distance measurements

- SONARDYNE sensors

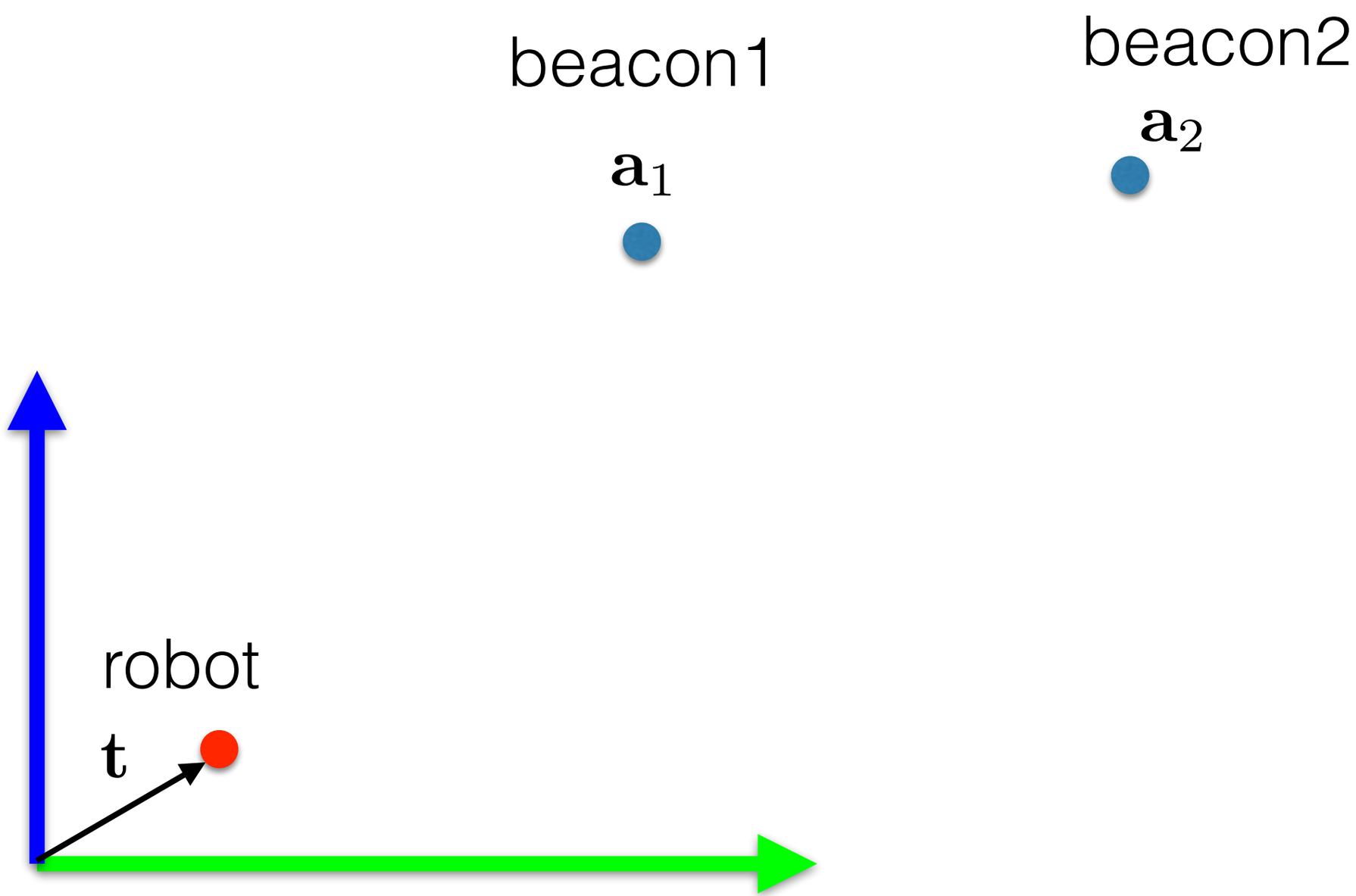


Special case II: Alignment from distance measurements

- SONARDYNE sensors

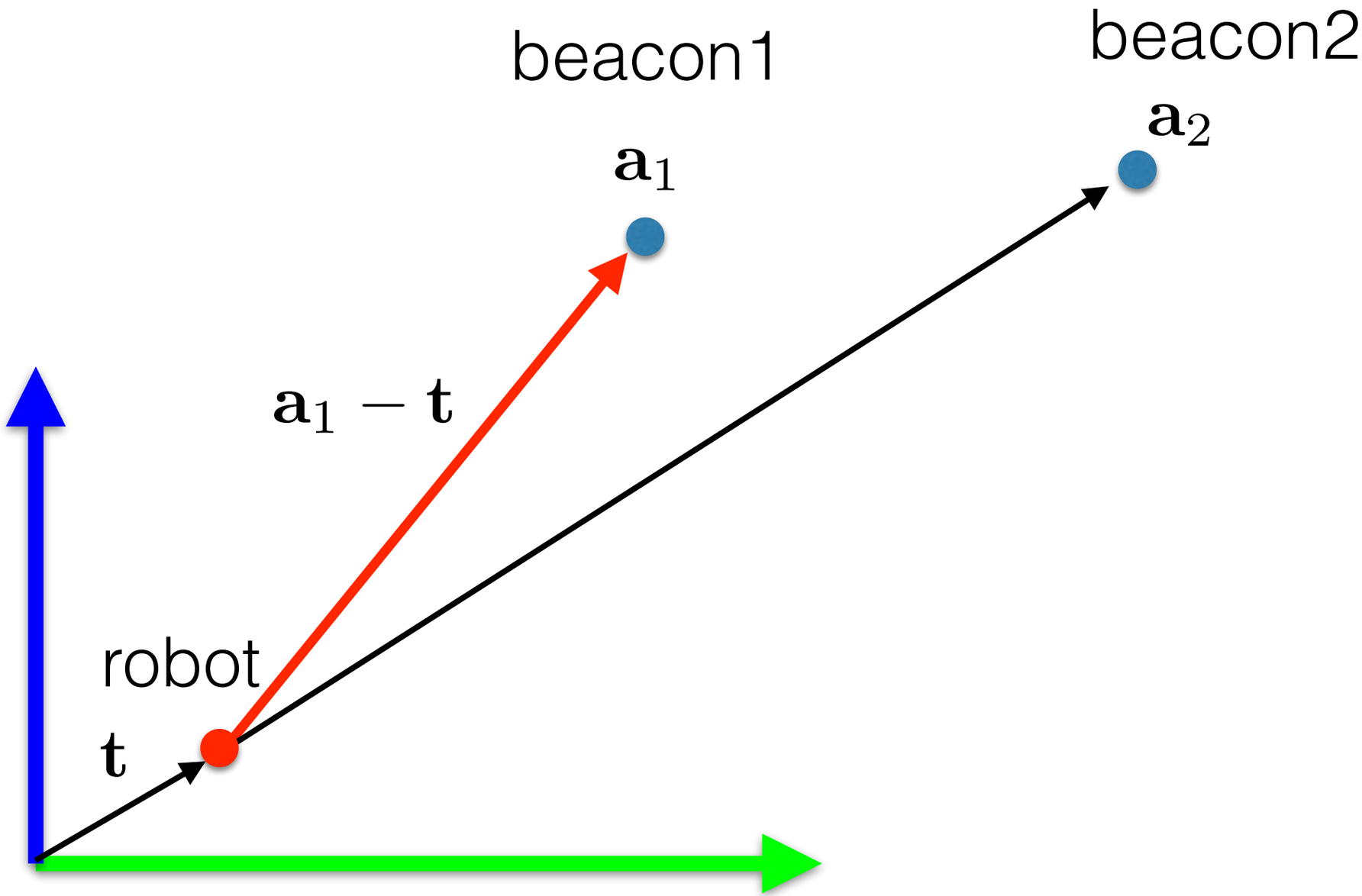


Special case II: Alignment from distance measurements



Special case II: Alignment from distance measurements

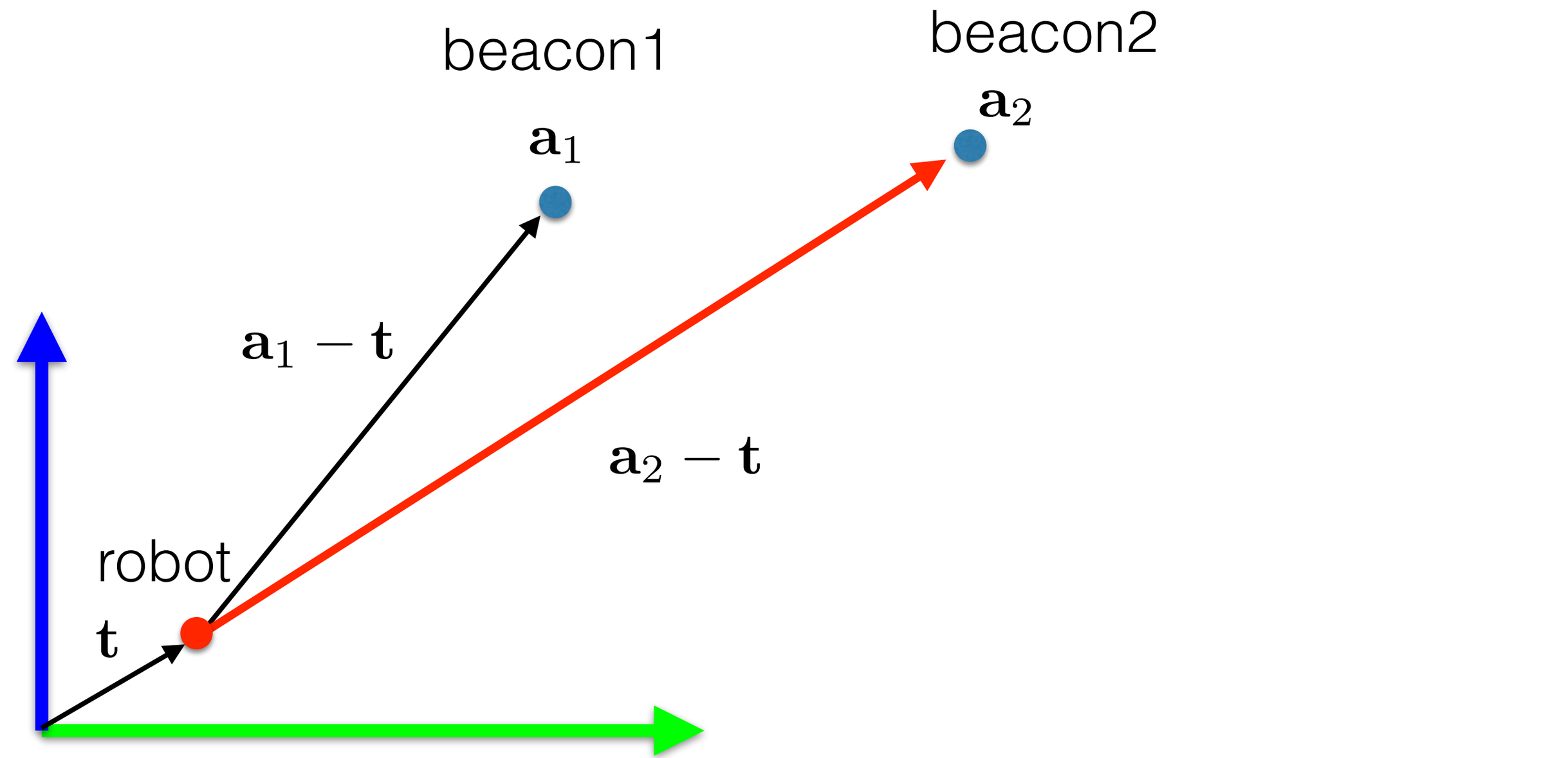
$$\|\mathbf{a}_1 - \mathbf{t}\|_2^2 = d_1^2$$



Special case II: Alignment from distance measurements

$$\|\mathbf{a}_1 - \mathbf{t}\|_2^2 = d_1^2$$
$$\|\mathbf{a}_2 - \mathbf{t}\|_2^2 = d_2^2$$

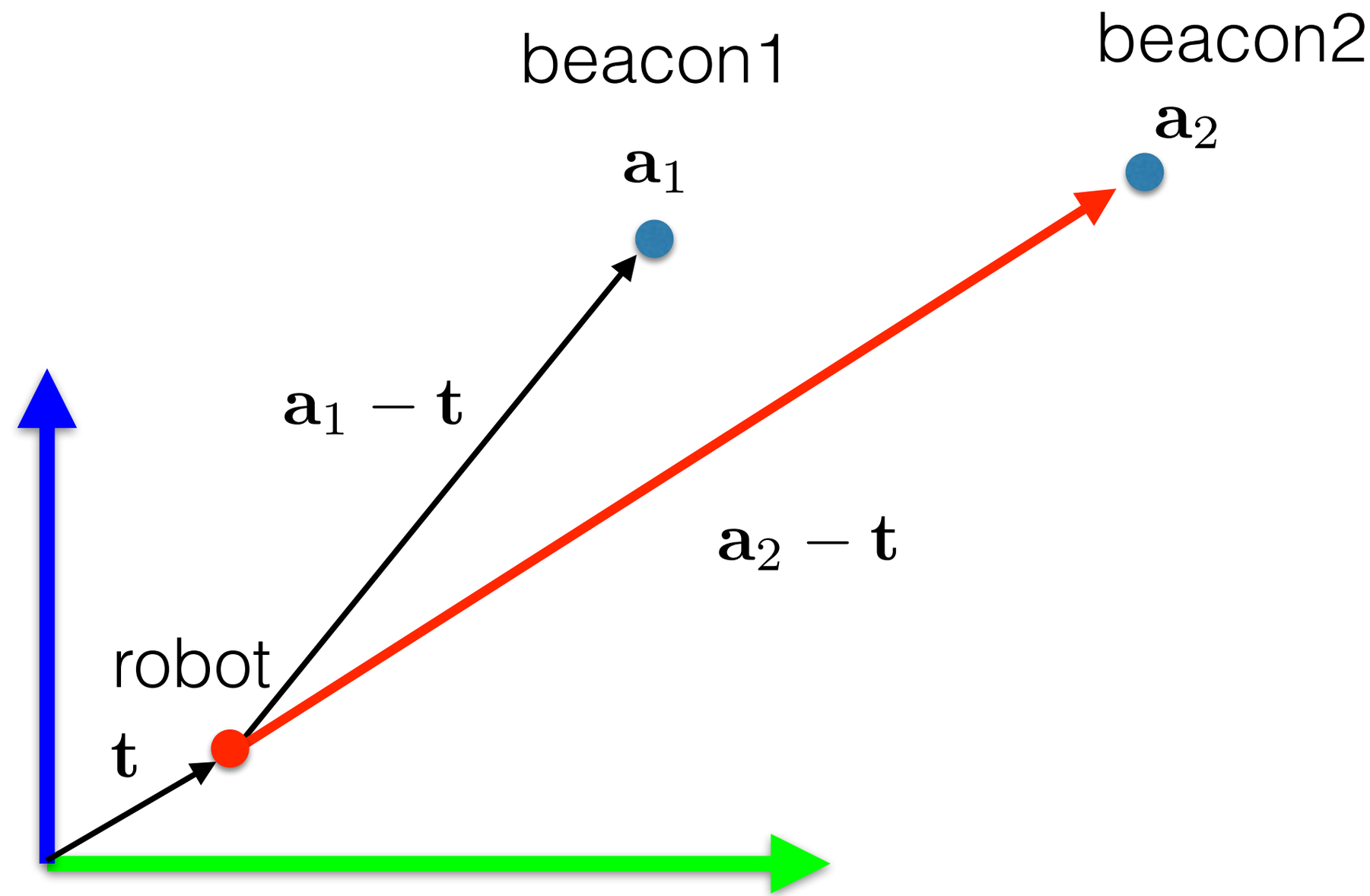
known measurements



Special case II: Alignment from distance measurements

$$\|\mathbf{a}_1 - \mathbf{t}\|_2^2 = d_1^2$$
$$\|\mathbf{a}_2 - \mathbf{t}\|_2^2 = d_2^2$$

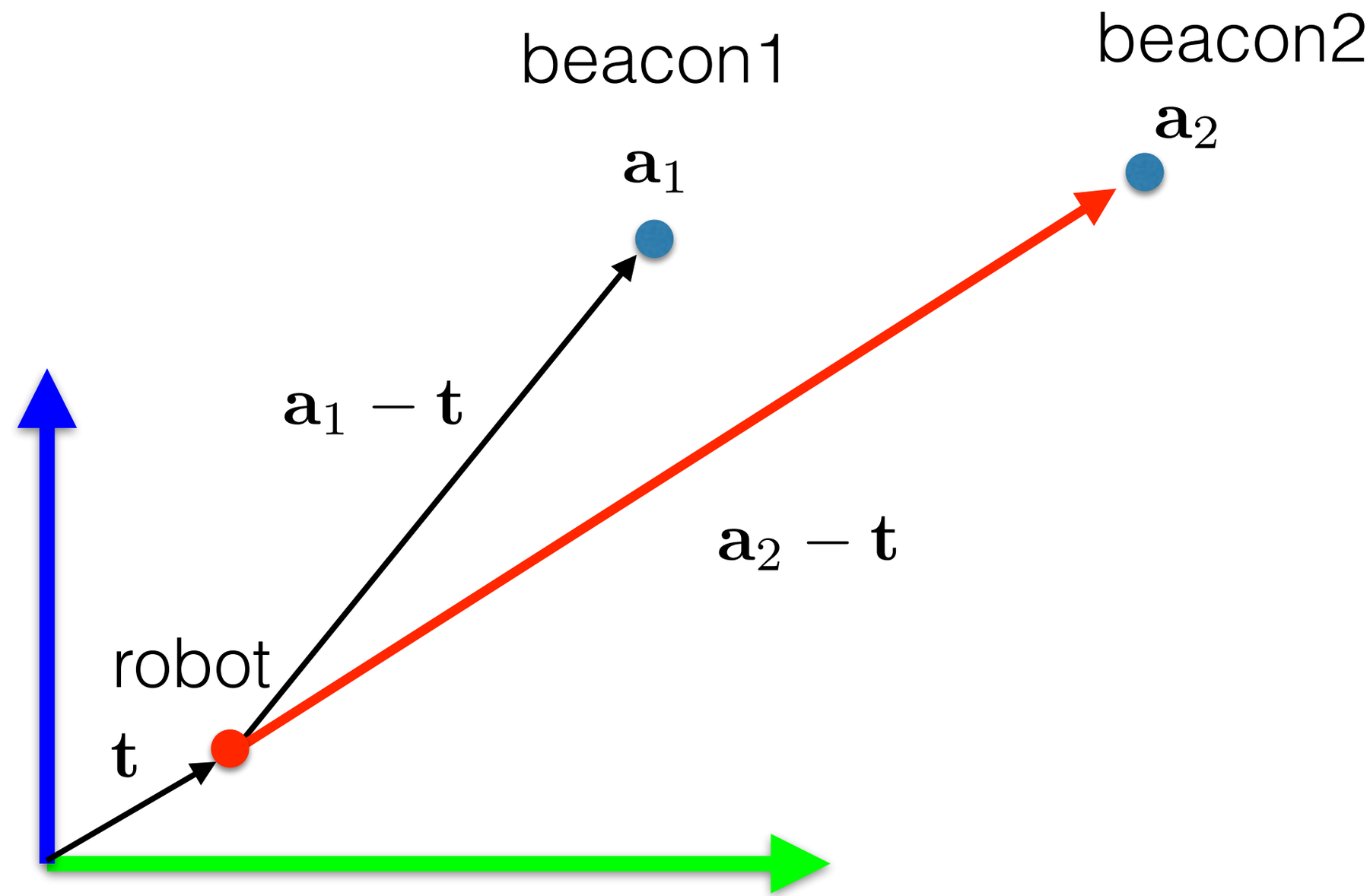
unknowns



Special case II: Alignment from distance measurements

$$\|\mathbf{a}_1 - \mathbf{t}\|_2^2 = d_1^2$$
$$\|\mathbf{a}_2 - \mathbf{t}\|_2^2 = d_2^2$$

unknowns



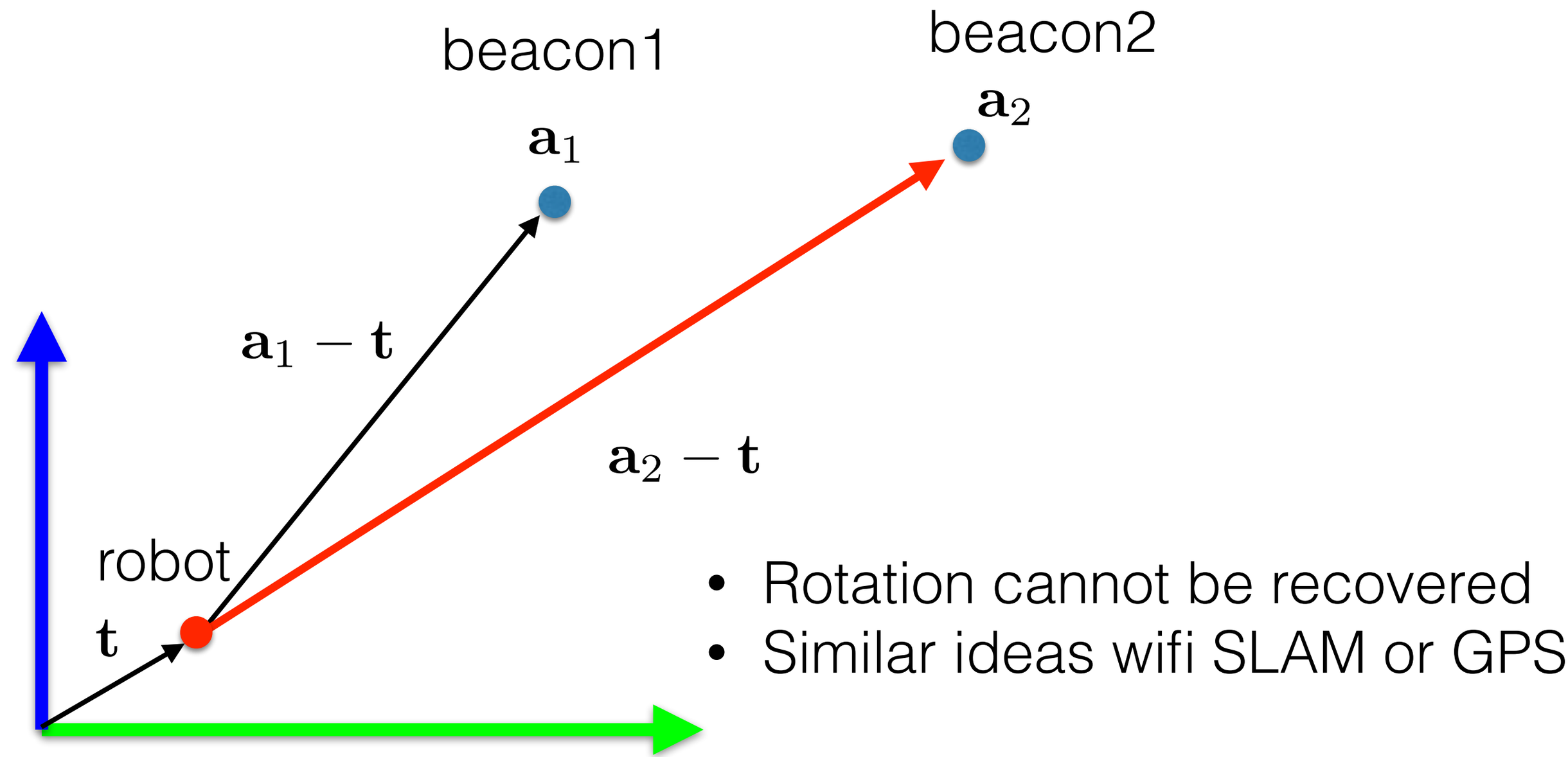
Special case II: Alignment from distance measurements

$$\|\mathbf{a}_1 - \mathbf{t}\|_2^2 = d_1^2$$
$$\|\mathbf{a}_2 - \mathbf{t}\|_2^2 = d_2^2$$

unknowns

=>

$$\arg \min_{\mathbf{a}_i, \mathbf{t} \in \mathcal{R}^3} \sum_i \| \underbrace{\|\mathbf{a}_i - \mathbf{t}\|_2^2 - d_i^2}_{f_{00}^i(\mathbf{t})} \|_2^2$$



- Rotation cannot be recovered
- Similar ideas wifi SLAM or GPS

Special case III: Alignment with poor initialization

Next lecture

$$\underset{\mathbf{R}\in\mathcal{SO}(3),\mathbf{t}\in\mathcal{R}^3}{\arg\min} \left\| \begin{array}{c} \sum_i f_{33}^i(\mathbf{R},\mathbf{t}) \\ \sum_i f_{23}^i(\mathbf{R},\mathbf{t}) \\ \sum_i f_{23}^i(\mathbf{R},\mathbf{t}) \\ \sum_i f_{22}^i(\mathbf{R},\mathbf{t}) \end{array} \right\|_2^2$$

When is the L2 norm not reasonable?

other losses: L1, Welsch loss function, box-penalty

other optimizations: RANSAC