ELEC 3210
Introduction to Mobile Robotics
Lecture 15

(Machine Learning and Infomation Processing for Robotics)

Huan YIN

Research Assistant Professor, Dept. of ECE

eehyin@ust.hk



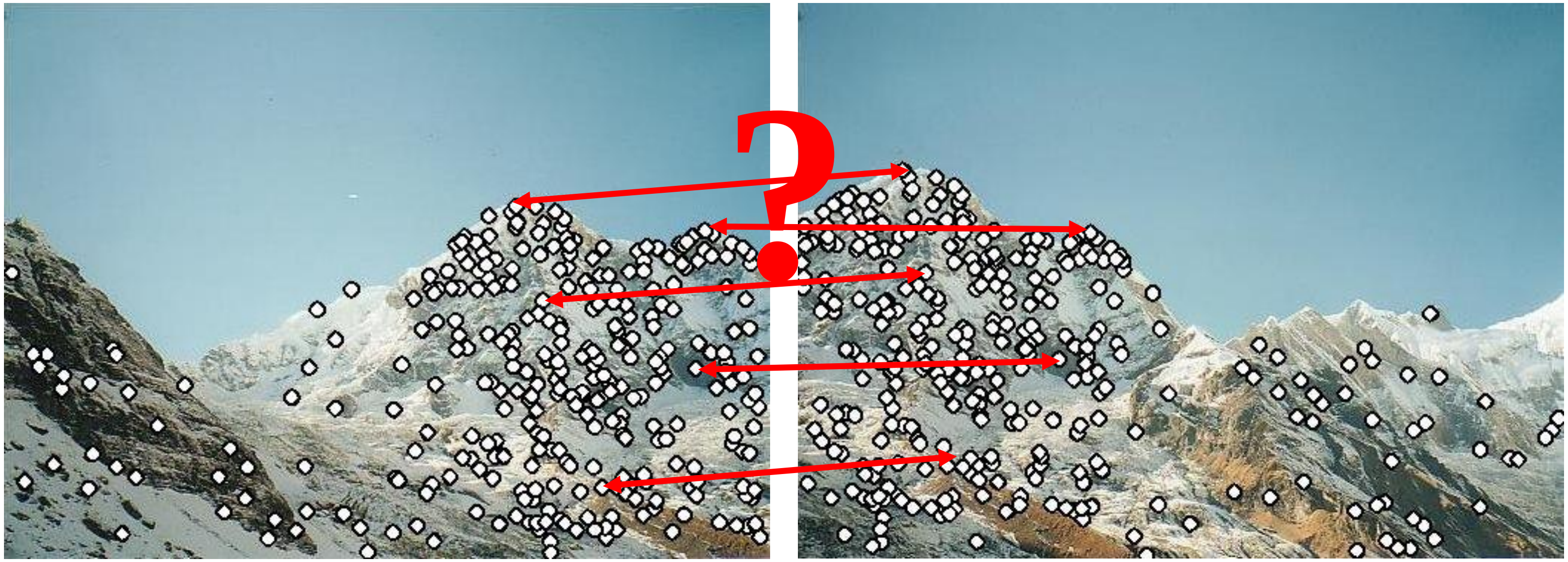

Recap L14

- Harris Corner
  - Eigen value and eigen vectors

Courtesy:

Feature Descriptors

- We know how to detect good points
- Next question: How to match them?



Courtesy: Steve Seitz and Richard Szeliski

Feature Descriptor and Matching

Invariance

- Suppose we are comparing two images I_1 and I_2
  - I_2 may be a transformed version of I_1
  - What kinds of transformations are we likely to encounter in practice?
- We'd like to find the same features regardless of the transformation
  - This is called transformational ***invariance***
  - Most feature methods are designed to be invariant to
    - Translation, 2D rotation, scale
  - They can usually also handle
    - Limited 3D rotations (SIFT works up to about 60 degrees)
    - Limited illumination/contrast changes

Courtesy: Steve Seitz and Richard Szeliski

How to achieve invariance

Need both of the following:

1. **Make sure your detector is invariant**
   - Harris is invariant to translation and rotation
   - Scale is trickier
     - Many sophisticated methods find "the best scale" to represent each feature (e.g., SIFT), Autoscale in Lecture 14
2. **Design an invariant feature *descriptor***
   - A descriptor captures the information in a region around the detected feature point
   - The simplest descriptor: a square window of pixels
     - What's this invariant to?
   - Let's look at some better approaches…

Courtesy: Steve Seitz and Richard Szeliski

Scale Invariant Feature Transform

- Basic idea:
  - Take 16x16 square window around detected feature
  - Compute edge orientation for each pixel
  - Throw out weak edges (threshold gradient magnitude)
  - Create histogram of surviving edge orientations

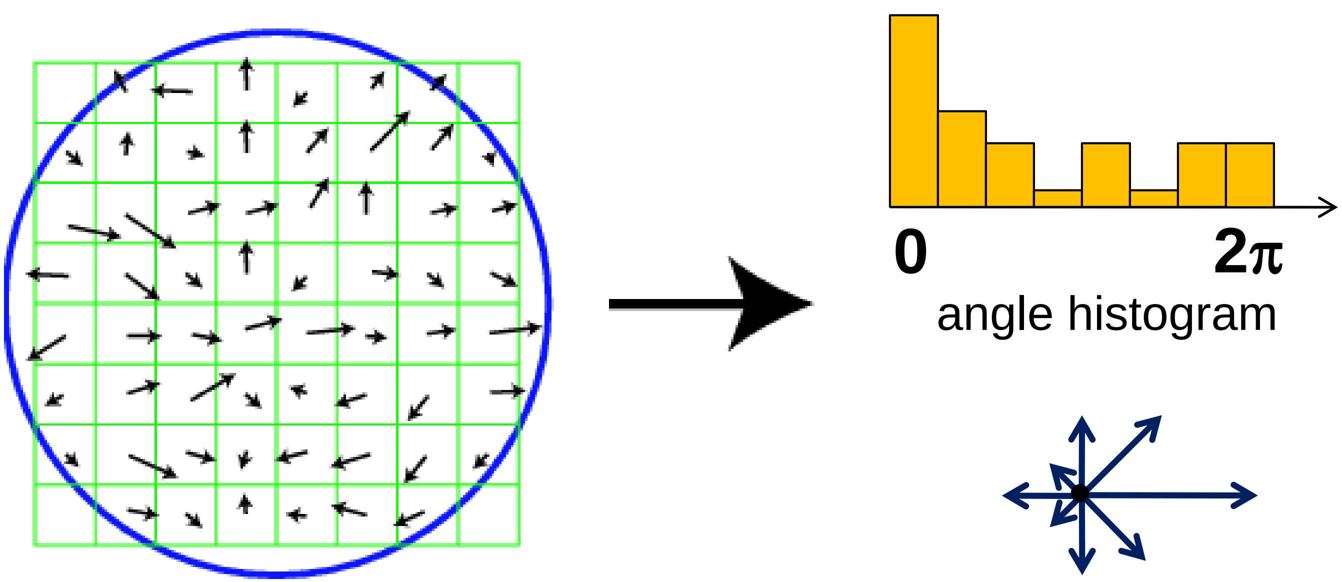
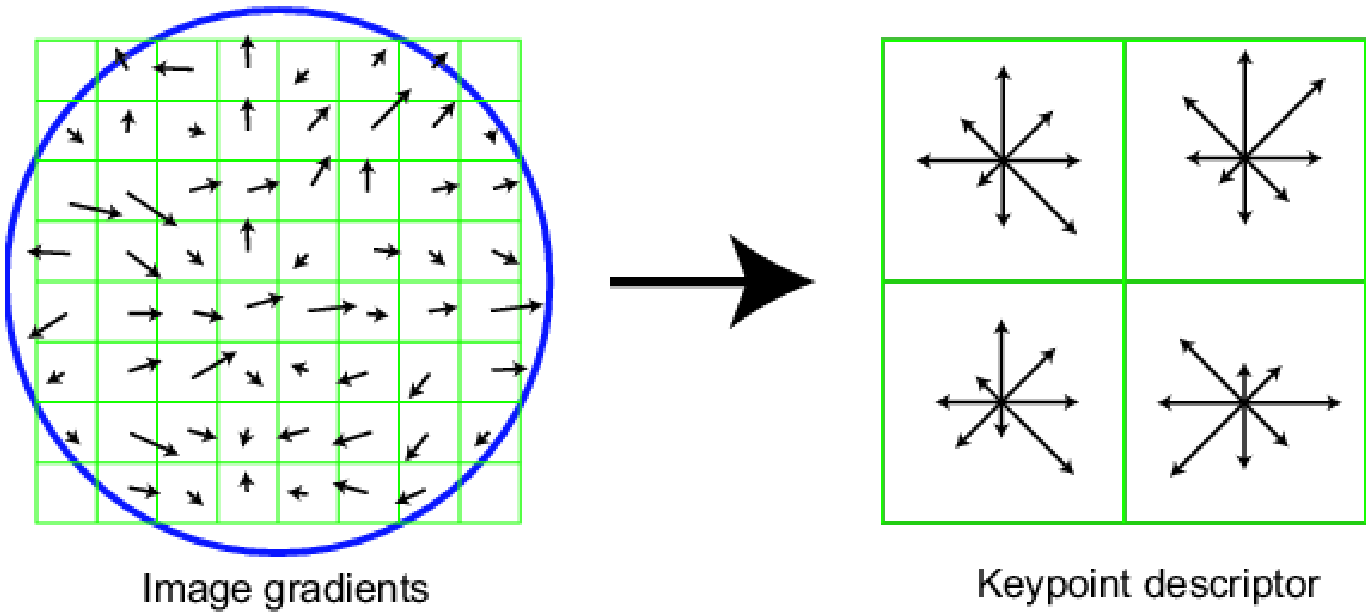


Image gradients

Courtesy: Steve Seitz and Richard Szeliski

SIFT

- Full version
  - Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
  - Compute an orientation histogram for each cell
  - 16 cells * 8 orientations = 128 dimensional descriptor

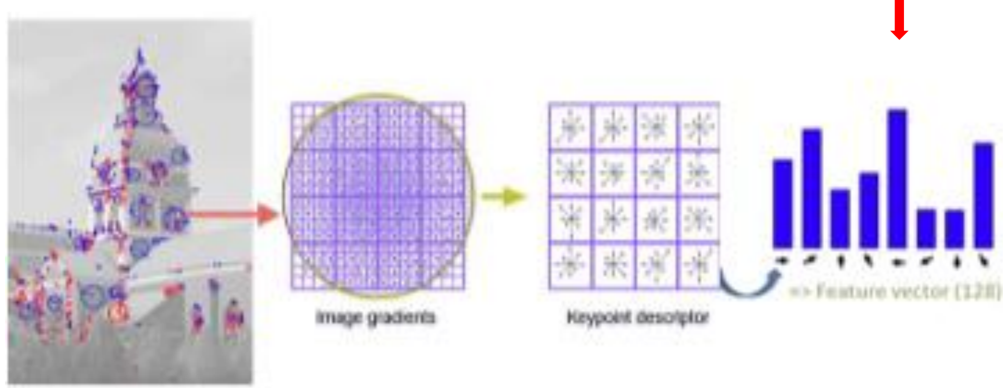


Courtesy: Steve Seitz and Richard Szeliski

SIFT Properties

- Translation invariance
- Illumination invariance

- Rotation invariance
  - rotate all pixel gradients to 90 deg using to the “main direction”
  - main direction = the highest bin in histogram
  - invariant up to about 60 degree out of plane rotation



Problem

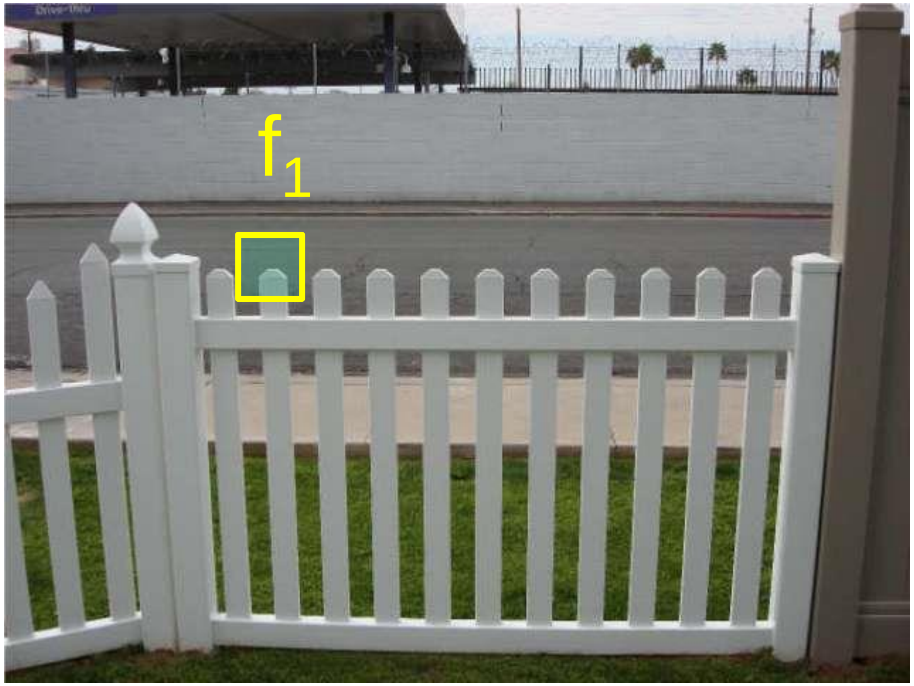
Given a feature in I_1, how to find the best match in I_2?

1. Define distance function that compares two descriptors
2. Test all the features in I_2, find the one with min distance

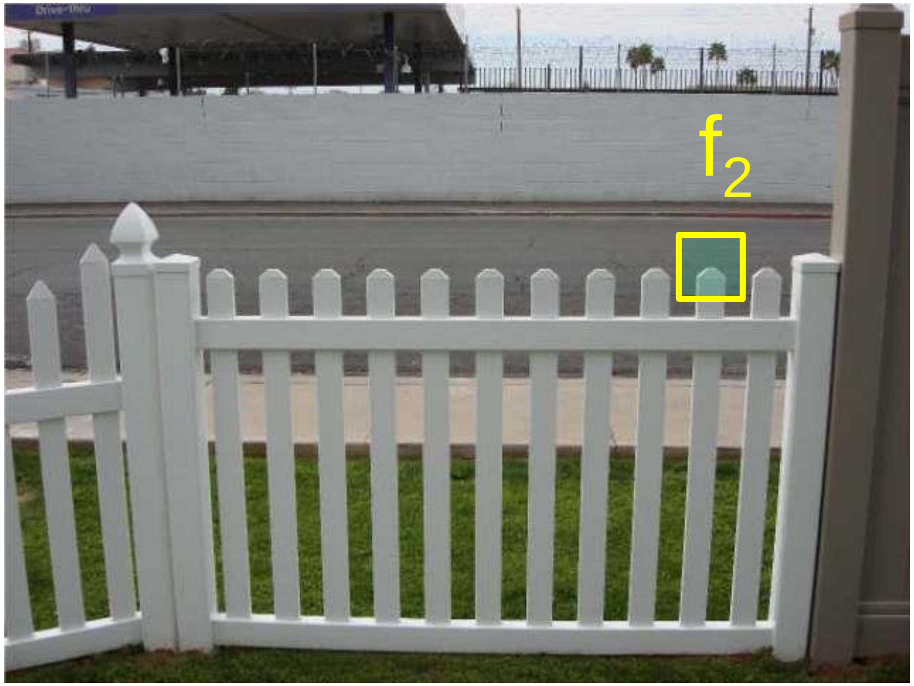
Courtesy: Steve Seitz and Richard Szeliski

Feature Distance

- How to define the difference between two features f_1, f_2?
  - Simple approach is SSD(f_1, f_2)
    - sum of square differences between entries of the two descriptors
    - can give good scores to very ambiguous (bad) matches



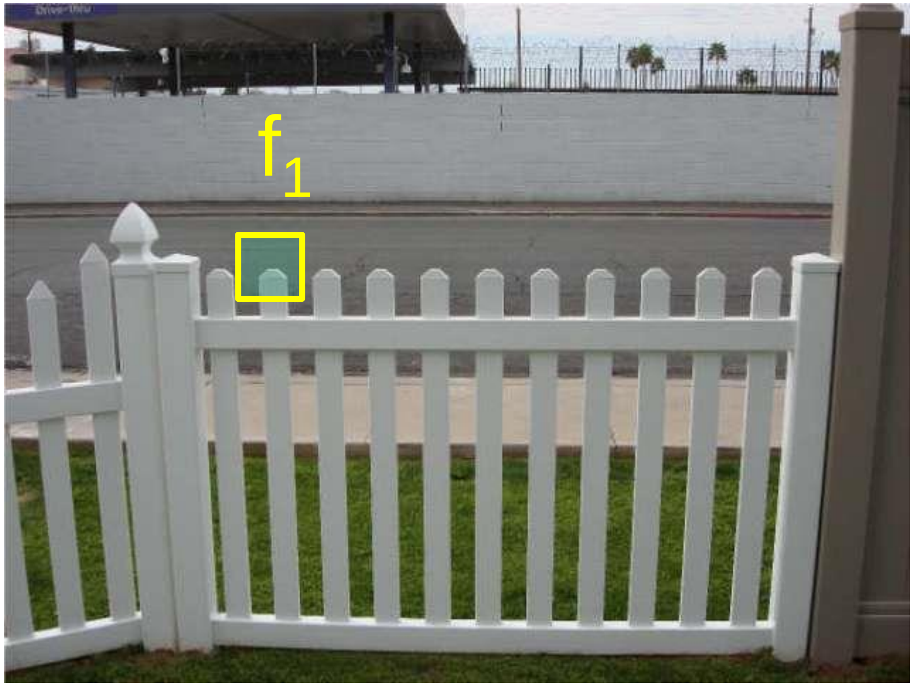
I_1



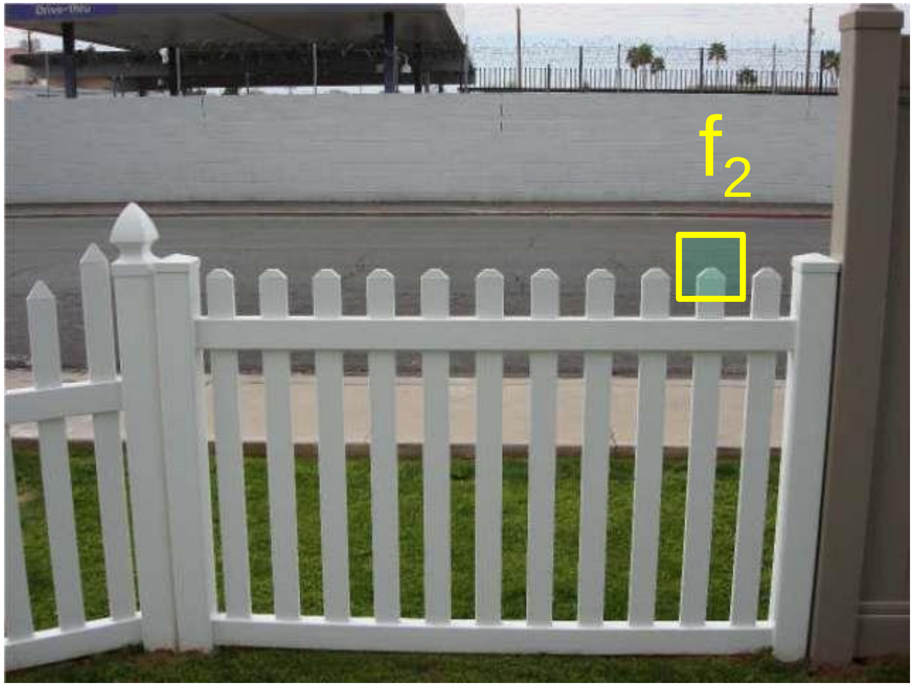
I_2

Feature Distance

- How to define the difference between two features f_1, f_2?
  - Better approach: ratio distance = $SSD(f_1, f_2) / SSD(f_1, f_2')$
    - f_2 is best SSD match to f_1 in I_2
    - f_2' is 2^{nd} best SSD match to f_1 in I_2
    - gives small values for ambiguous matches

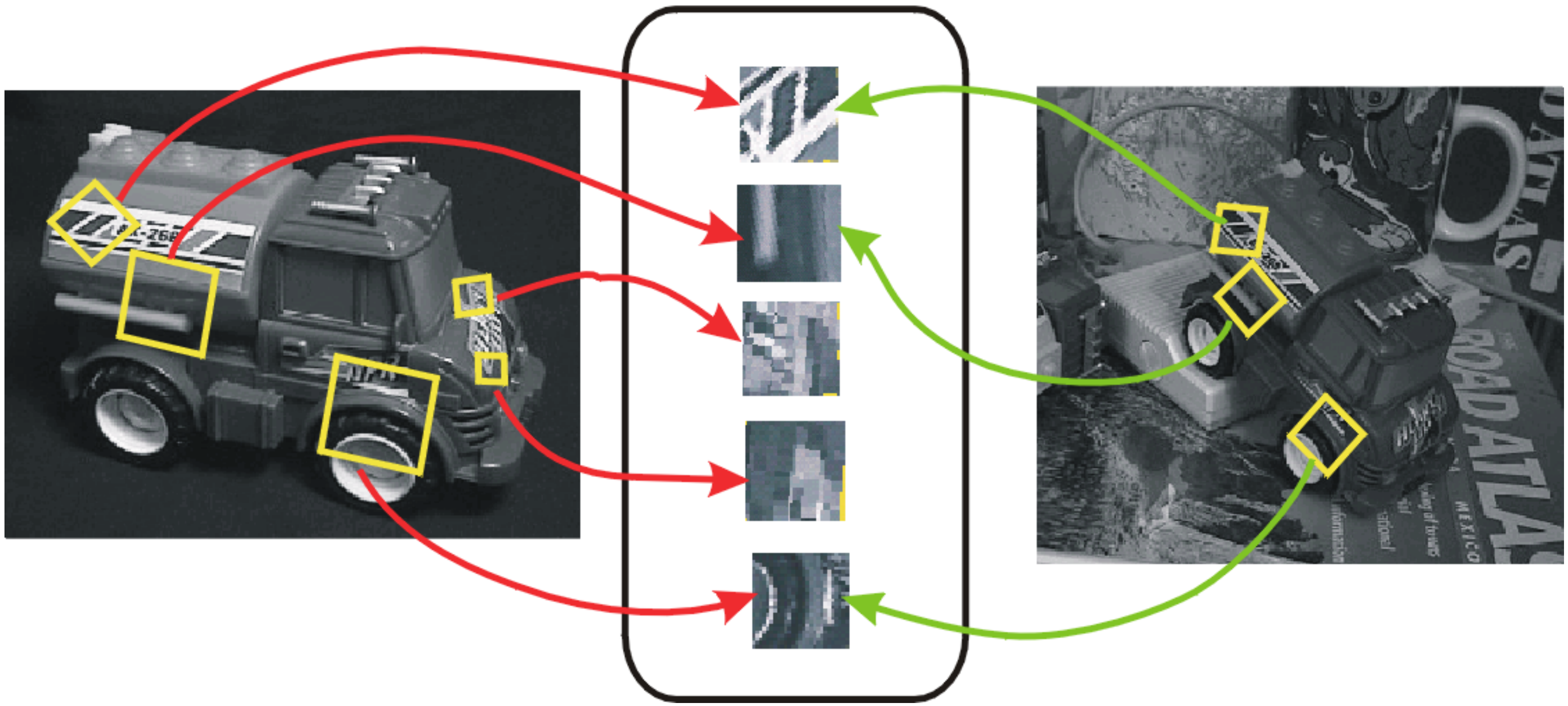


I_1



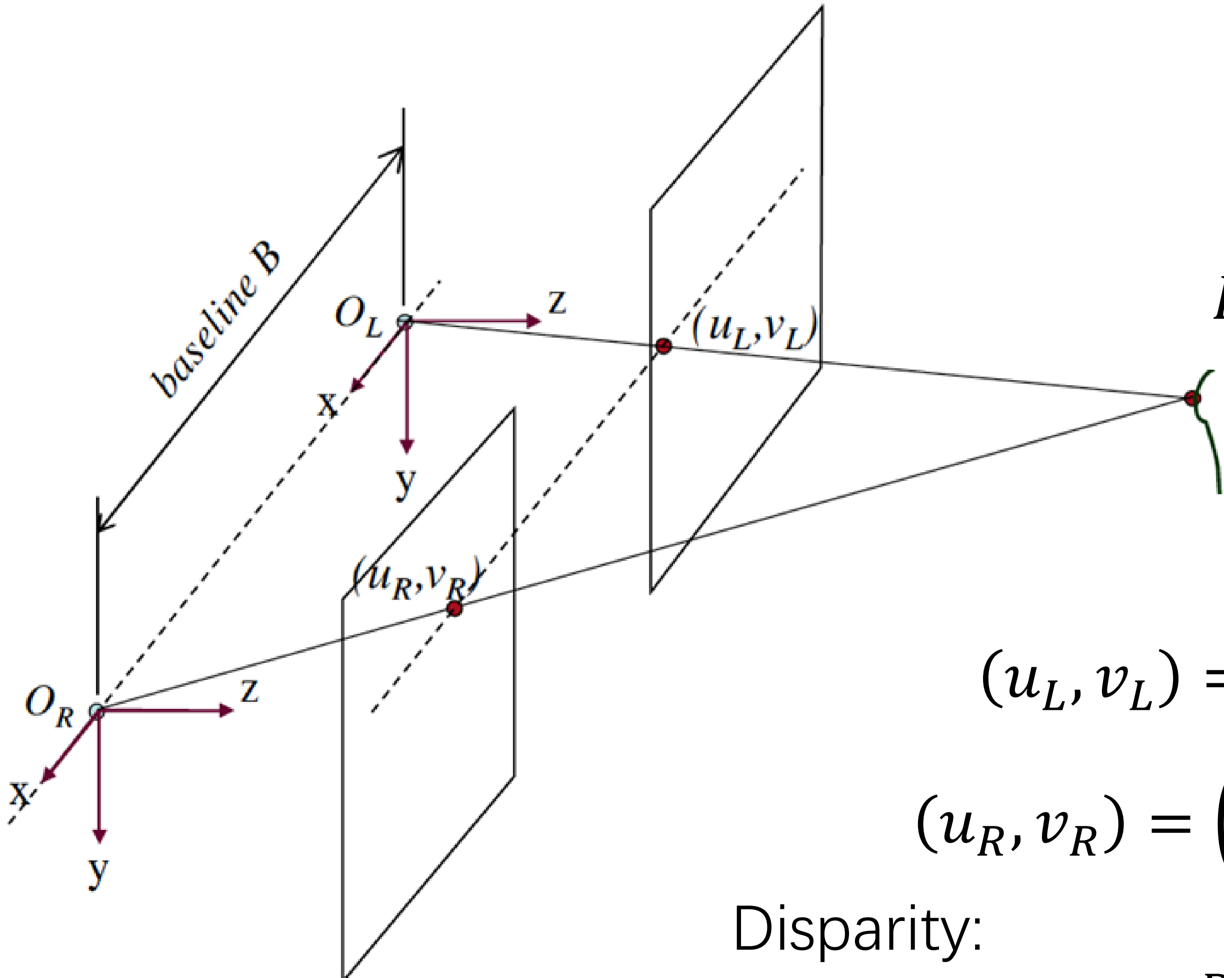
I_2

Feature Matching by SIFT



3D-2D Pose Estimation
(Pose from Projective Transform)
(The PnP Problem)

Recap L4 - Depth from Stereo Vision



$$P_L = (X, Y, Z)$$

$$(u_L, v_L) = \left(f\frac{X}{Z}, f\frac{Y}{Z}\right)$$

$$(u_R, v_R) = \left(f\frac{X-B}{Z}, f\frac{Y}{Z}\right)$$

Disparity:

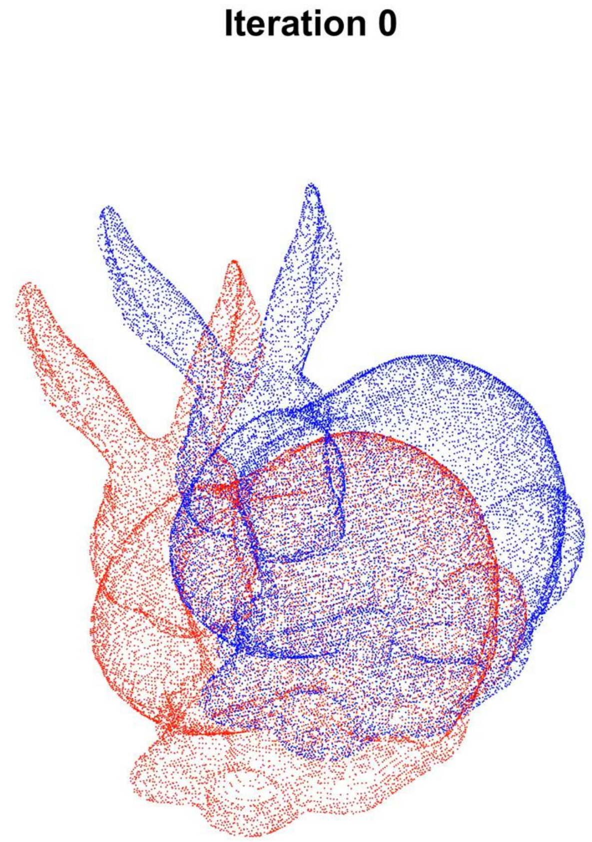
$$d = u_L - u_R = f\frac{B}{Z}$$

$$\boxed{Z = f\frac{B}{d}}$$

Courtesy: Shaojie Shen, Kostas Daniilidis

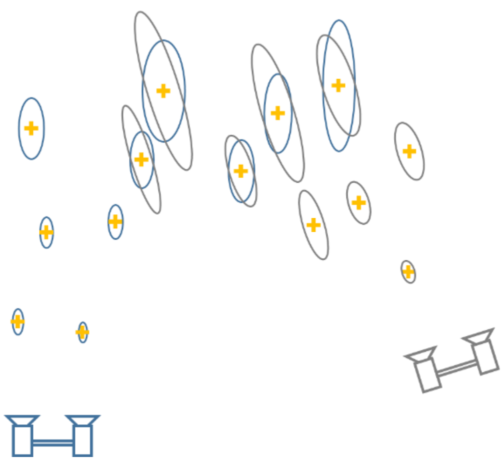
Recap L5 ICP

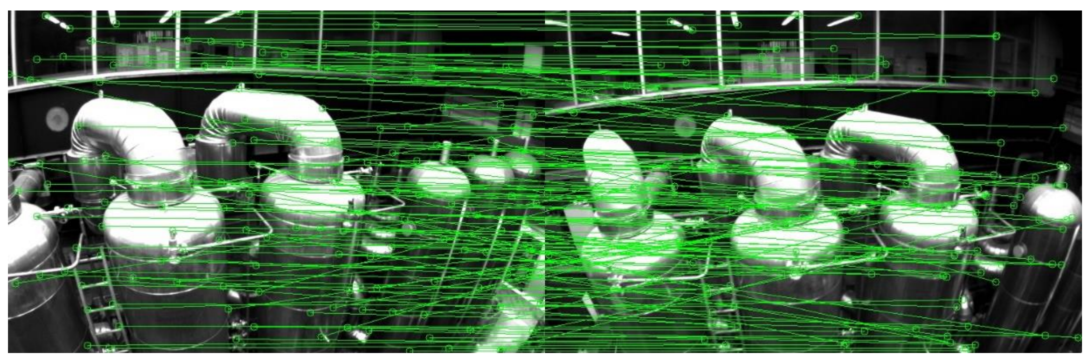
- Pose estimation via Iterative Closest Point
- 3D-3D pose estimation



3D-3D for Visual Pose Estimation

- How to obtain 3D-3D data association?
  - Calibrated stereo image pairs as input
  - Spatial matching – feature matching between stereo image pairs, for computation of 3D points
  - Temporal matching – feature matching between images captures at different times, for motion estimation
  - Need to address outlier removal (RANSAC)– to be discussed soon
  - Usually poor performance due to increased ranging error at longer distance with stereo vision – use 3D-2D pose estimation instead

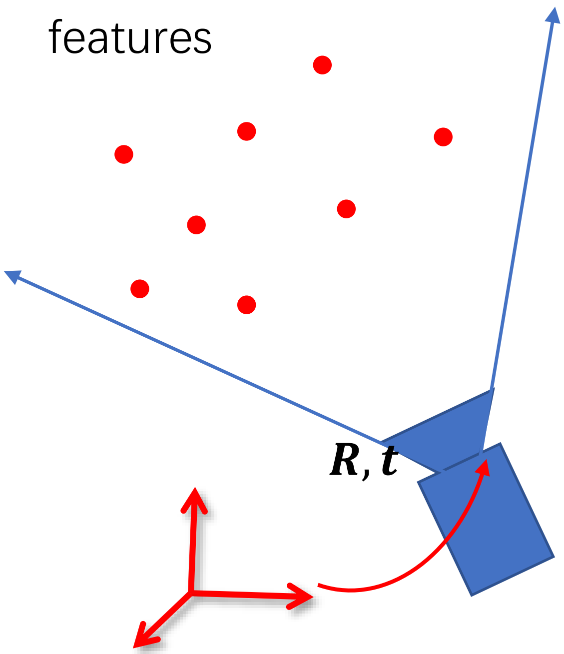




Courtesy: Shaojie Shen

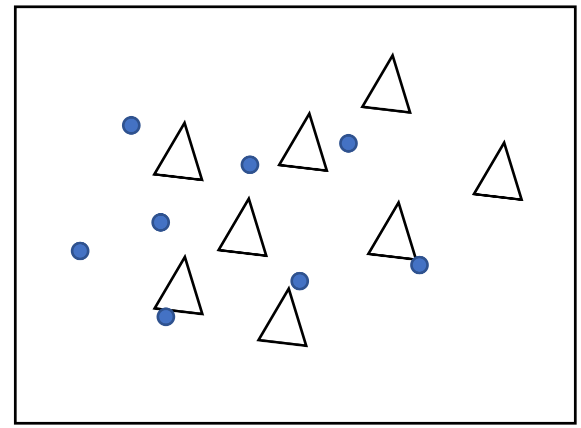
How about 2D-3D?

Known 3D features



3D-2D

Image



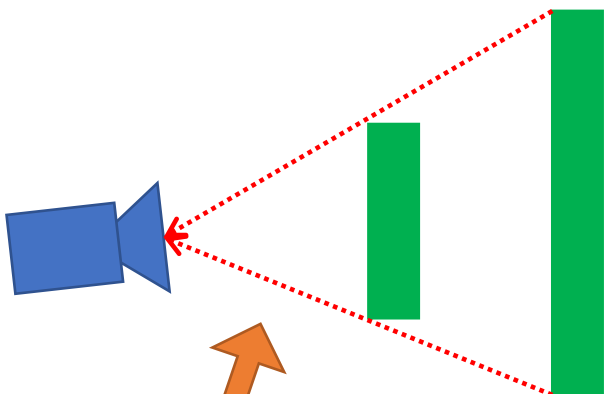
3D features in frame 1
(or known 3D features in world frame)

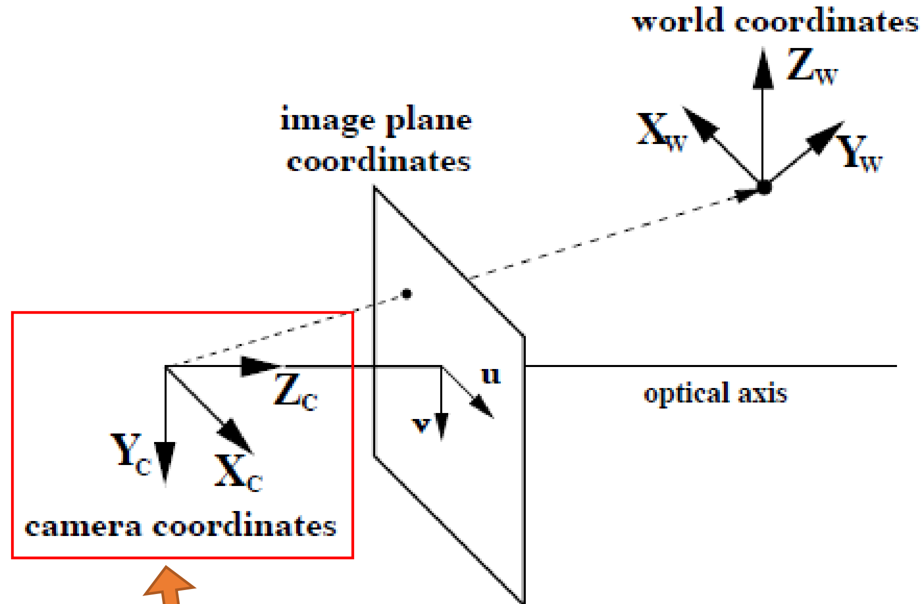
2D feature observations in frame 2
(or feature observations in body frame)

2D feature reprojections of given a estimated pose of frame 2 in frame 1
(or estimated transformation of body frame in world frame)

Courtesy: Shaojie Shen

Recap L4 Pin-hole Camera Model





$$\underbrace{\boxed{\lambda}}_{\text{Unknown Depth}} \underbrace{\boxed{\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}}}_{\text{Pixel Values}} = \underbrace{\boxed{\begin{pmatrix} f & 0 & u_o \\ 0 & f & v_o \\ 0 & 0 & 1 \end{pmatrix}}}_{\text{Intrinsic Calibration Matrix}} \underbrace{\boxed{\begin{pmatrix} R & t \end{pmatrix}}}_{\text{Camera Pose}} \underbrace{\boxed{\begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}}}_{\text{World Point}} = P \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

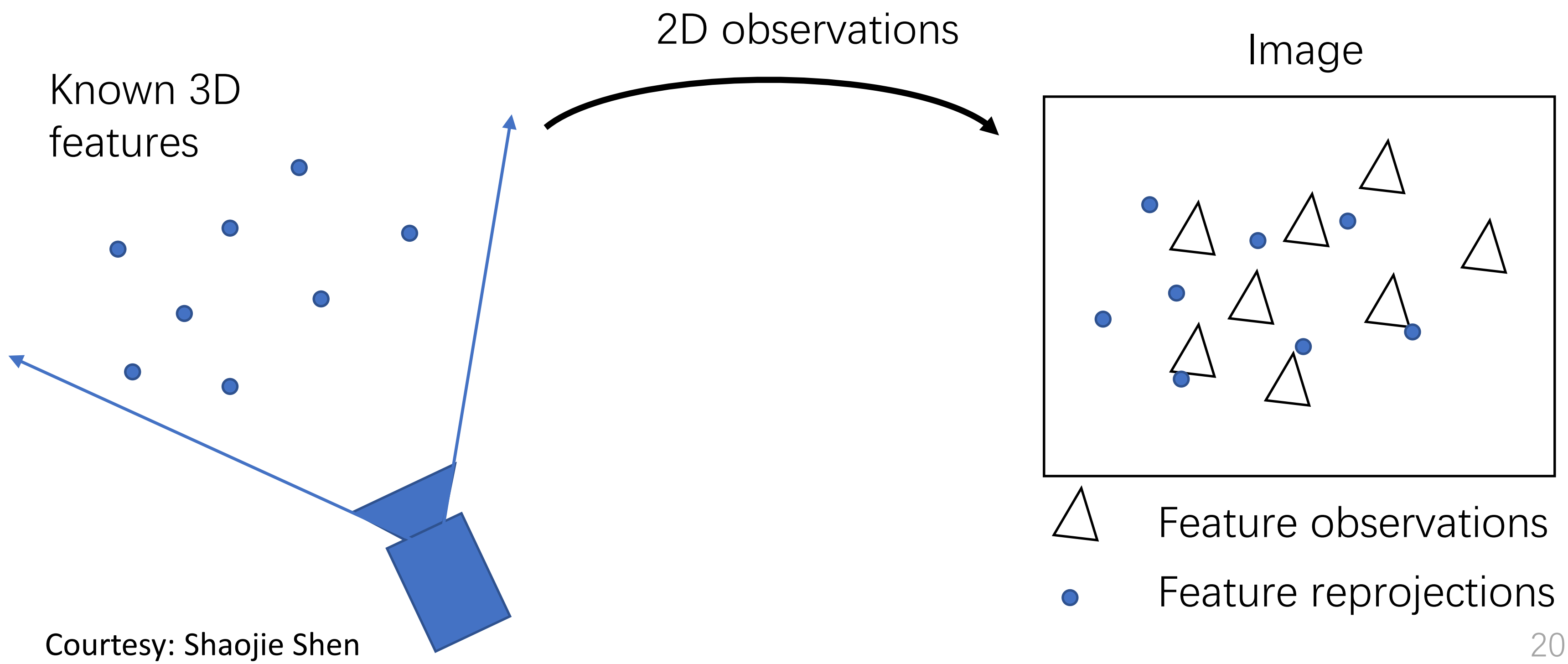
Courtesy: Shaojie Shen

Nonlinear 3D-2D Pose Estimation

$\boldsymbol{\theta}$: $Euler\ Angles \in R^3$ $\quad$ $\boldsymbol{t}$: $Translation \in R^3$ $\quad$ $\pi(\cdot)$: $projection\ function$

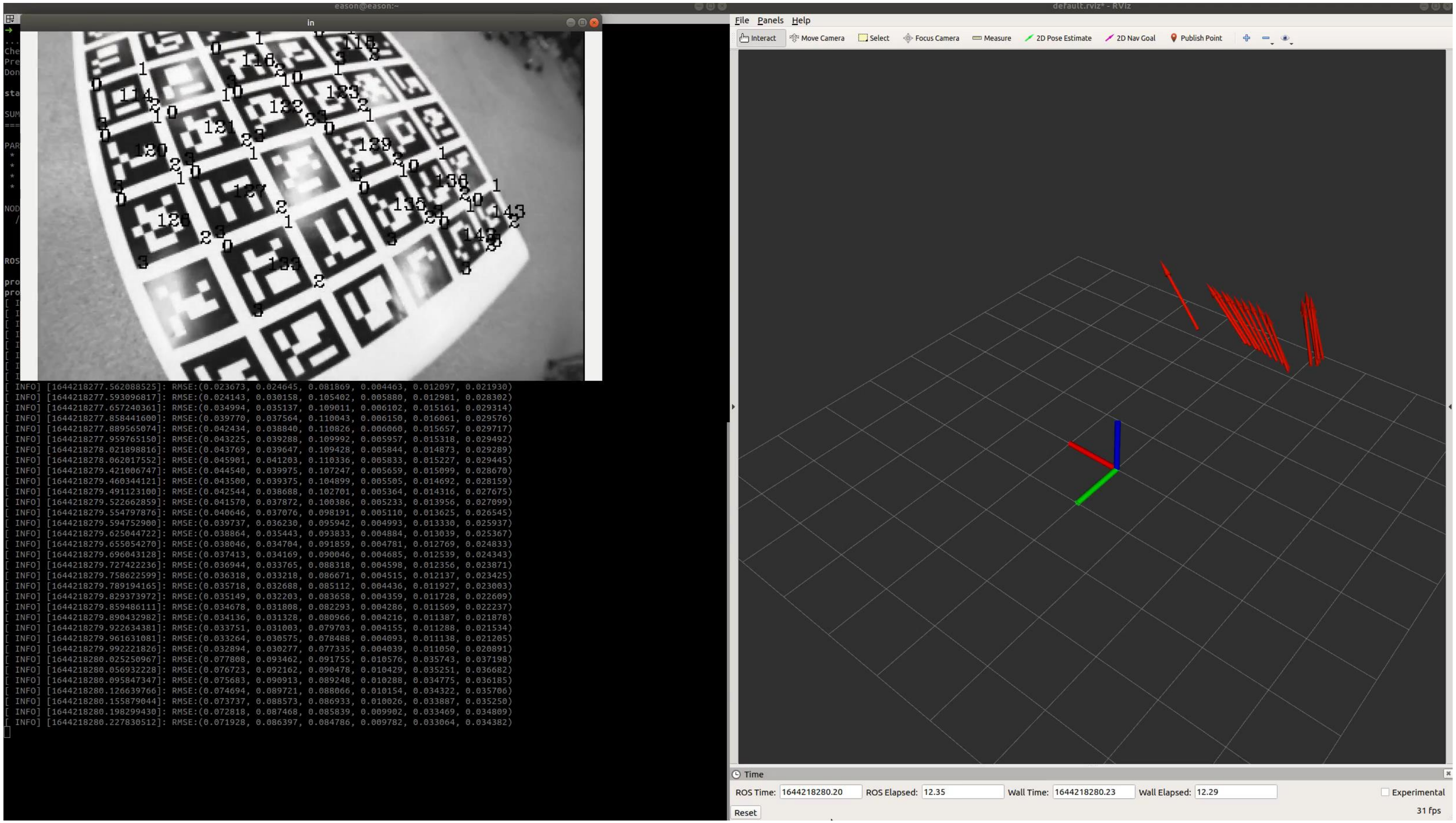
- Minimize the reprojection error w.r.t. camera pose
  - Can be solved via Gauss-Newton method

$$\min_{\boldsymbol{\theta},\boldsymbol{t}} \sum_i \left\| \begin{bmatrix} u_i \\ v_i \end{bmatrix} - \pi\left(\boldsymbol{K} \cdot \left(\boldsymbol{R}(\boldsymbol{\theta}) \cdot \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \boldsymbol{t} \right) \right) \right\|^2$$



Courtesy: Shaojie Shen

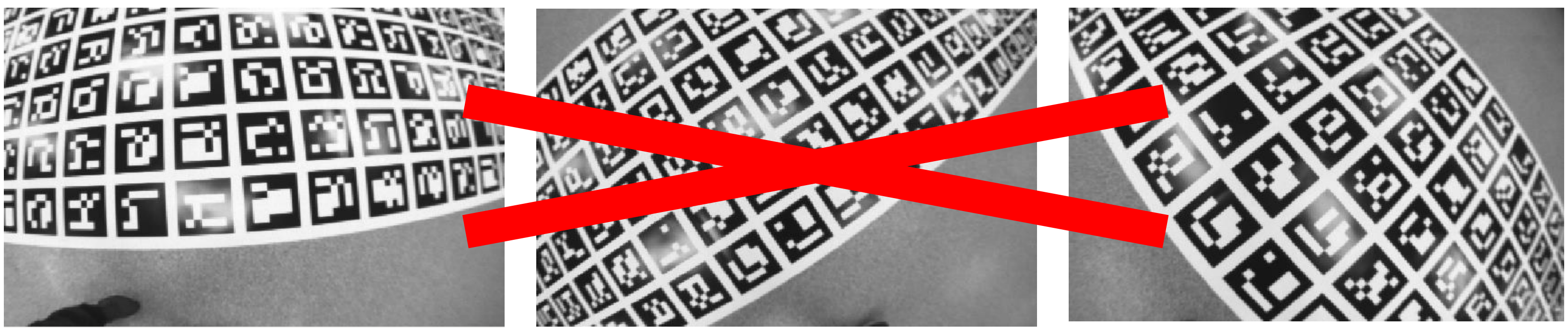
3D-2D Visual Localization with Markers



Courtesy: Shaojie Shen

What if no markers?

- In unstructured environments
  - For data association



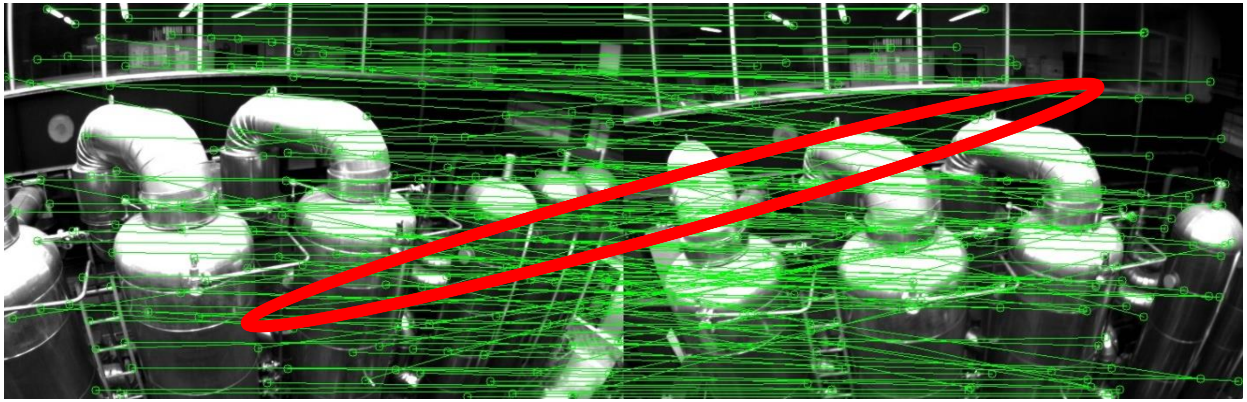
???

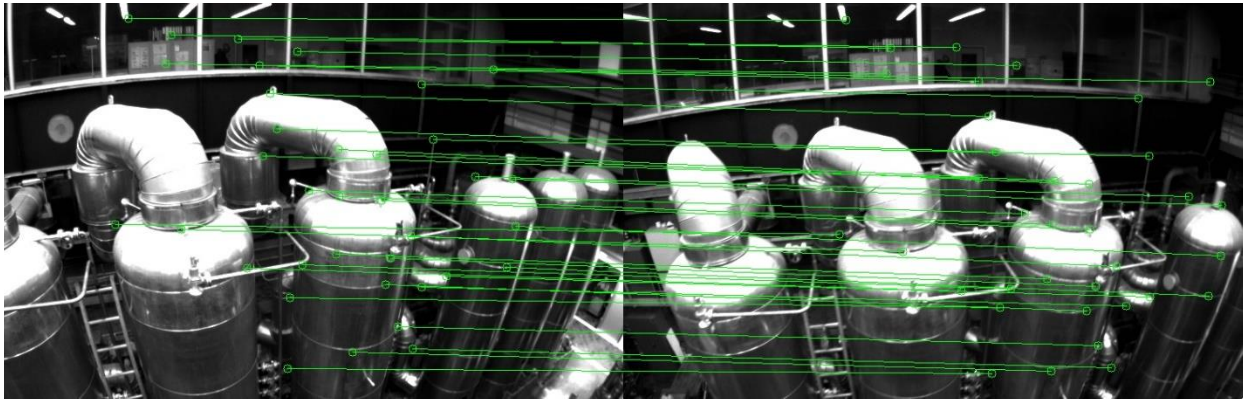
$$\min_{\boldsymbol{\theta},\boldsymbol{t}} \sum_i \left\| \begin{bmatrix} u_i \\ v_i \end{bmatrix} - \pi\left(\boldsymbol{K} \cdot \left(\boldsymbol{R}(\boldsymbol{\theta}) \cdot \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \boldsymbol{t} \right) \right) \right\|^2$$

Courtesy: Shaojie Shen

No Markers

- What if you do not have markers?
  - Outlier rejection





Courtesy: Shaojie Shen

RANdom SAmple Consensus (RANSAC)

RANSAC

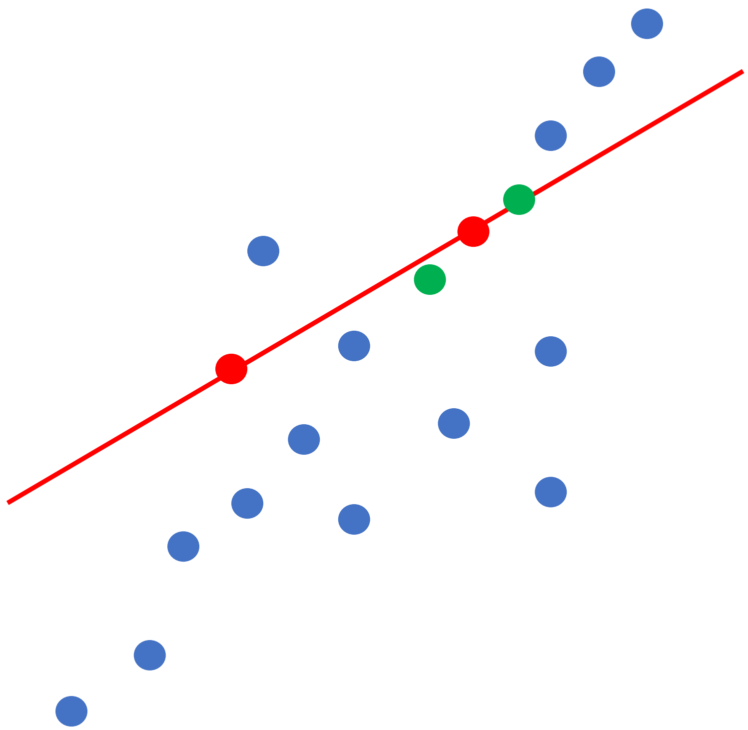
- Model fitting with outlier rejection
  - The 6-DOF pose you are trying to estimation is a model
- Algorithm:
  - Loop:
    - Randomly select a small amount of (or minimum) data points to find a model
    - See the error between the model and all other data points
    - Find the data points with error smaller than a threshold as inliers
    - If the current model has more inliers than all previous ones, record all inliers
    - Repeat
  - Use all inliers to find the best estimate of the model

Courtesy:

RANSAC

- RANSAC for 2D line fitting
  - Minimum number of points to define a 2D line: 2
  - Error metric: point to line distance
  - How many iterations are required?

- Iteration 1: 4 Inliers

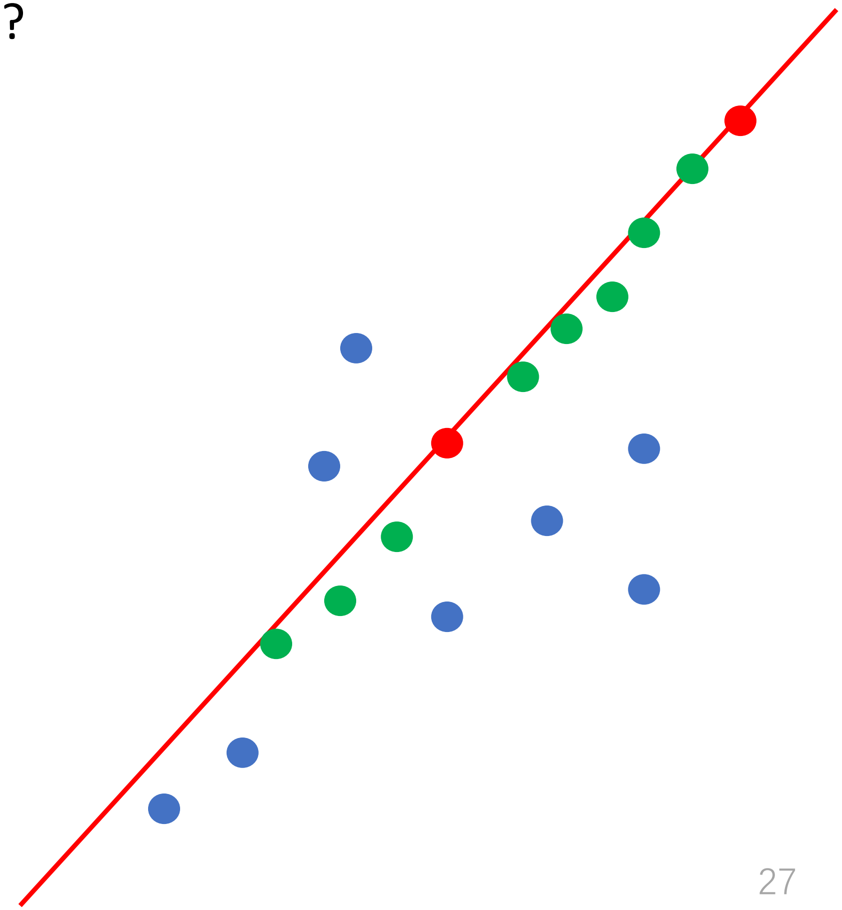


Courtesy: Shaojie Shen

RANSAC

- RANSAC for 2D line fitting
  - Minimum number of points to define a 2D line: 2
  - Error metric: point to line distance
  - How many iterations are required?

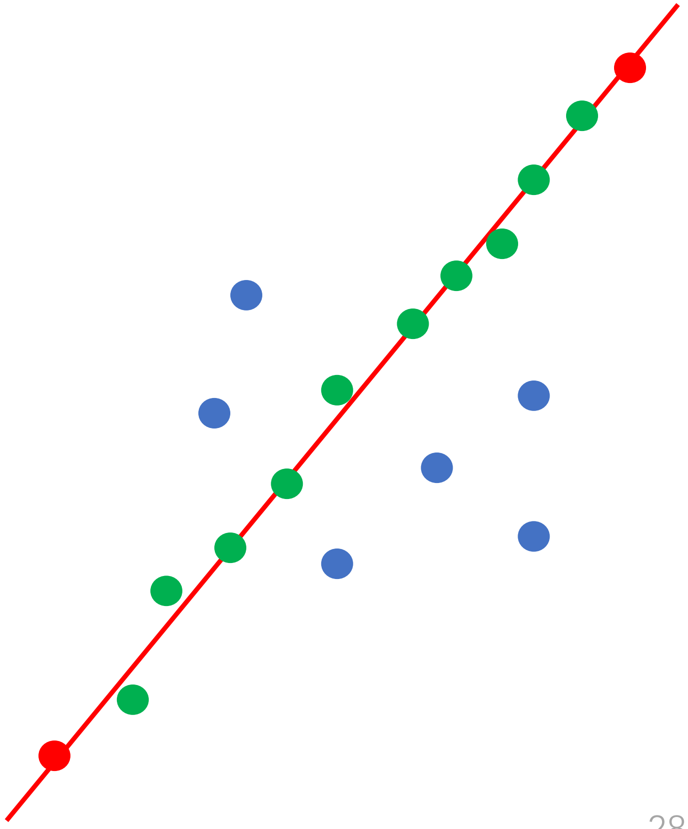
- Iteration 2: 10 Inliers



Courtesy: Shaojie Shen

- RANSAC for 2D line fitting
  - Minimum number of points to define a 2D line: 2
  - Error metric: point to line distance
  - How many iterations are required?

- Iteration 3: 12 Inliers

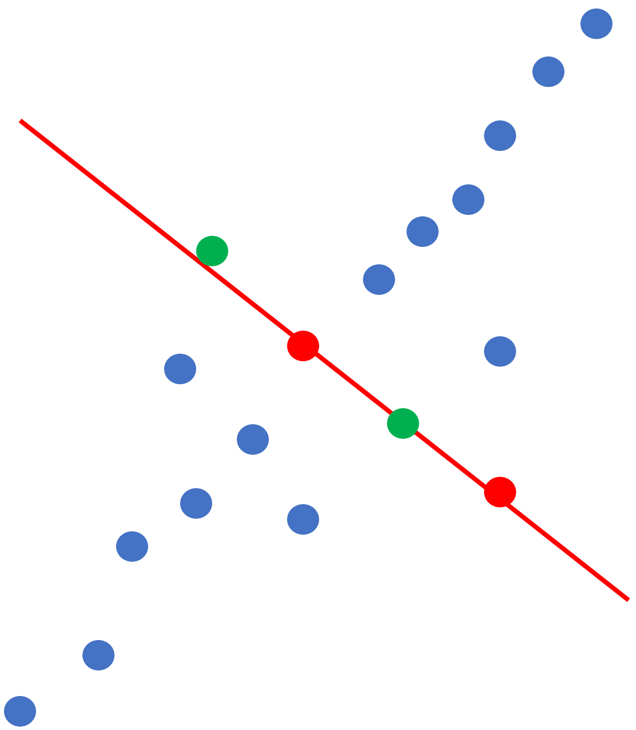


Courtesy: Shaojie Shen

Failure

- RANSAC for 2D line fitting
  - Minimum number of points to define a 2D line: 2
  - Error metric: point to line distance
  - How many iterations are required?

- Iteration 4: 4 Inliers



Courtesy: Shaojie Shen

Pose Estimation with RANSAC

- How many feature correspondences are required to create a model?
  - 3D-3D: 3
  - 3D-2D: 3
  - It is OK to use more points to find the model
  - But few number of points is better (Why?)
- How to define the error metric?
  - 3D-3D: point distance
  - 3D-2D: reprojection error
- How many iterations are required?

Courtesy: Shaojie Shen

RANSAC

- How many iterations are required - the probability
  - Probability of outlier: X (X < 1)
  - M number of data points to create a model
  - N iterations
- RANSAC failure: all random sample contains at least 1 outlier
  - Failure probability = $(1-(1-X)^M)^N$
  - 2D line fitting example
    - 30% outliers
    - 2 data points to create a model
    - Failure probability for 5 iterations: 3.45%
    - Failure probability for 10 iterations: 0.12%

Courtesy: Shaojie Shen

RANSAC

- How many iterations are required - the probability
  - Probability of outlier: X (X < 1)
  - M number of data points to create a model
  - N iterations
- RANSAC failure: all random sample contains at least 1 outlier
  - Failure probability = $(1-(1-X)^M)^N$
  - 3D-3D pose estimation
    - 30% outliers
    - 3 data points to create a model
    - Failure probability for 5 iterations: 12.24%
    - Failure probability for 10 iterations: 1.49%
    - Failure probability for 20 iterations: 0.02%

Courtesy: Shaojie Shen

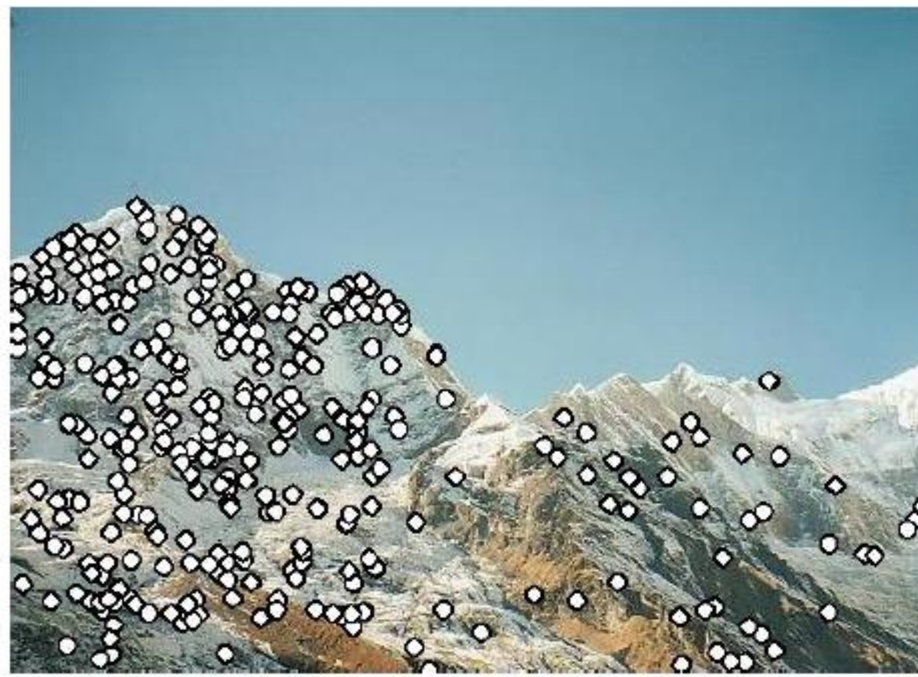
RANSAC

- How many iterations are required - the probability
  - Probability of outlier: X ($X < 1$)
  - M number of data points to create a model
  - N iterations
- RANSAC failure: all random sample contains at least 1 outlier
  - Failure probability = $(1-(1-X)^M)^N$
  - 3D-3D pose estimation
    - 30% outliers
    - 20 data points to create a model (bad example)
    - Failure probability for 5 iterations: 99.6%
    - Failure probability for 10 iterations: 99.2%
    - Failure probability for 20 iterations: 98.4%
    - Failure probability for 1000 iterations: 45%
    - Failure probability for 10000 iterations: 0.03%

Courtesy: Shaojie Shen

RANSAC for Alignment

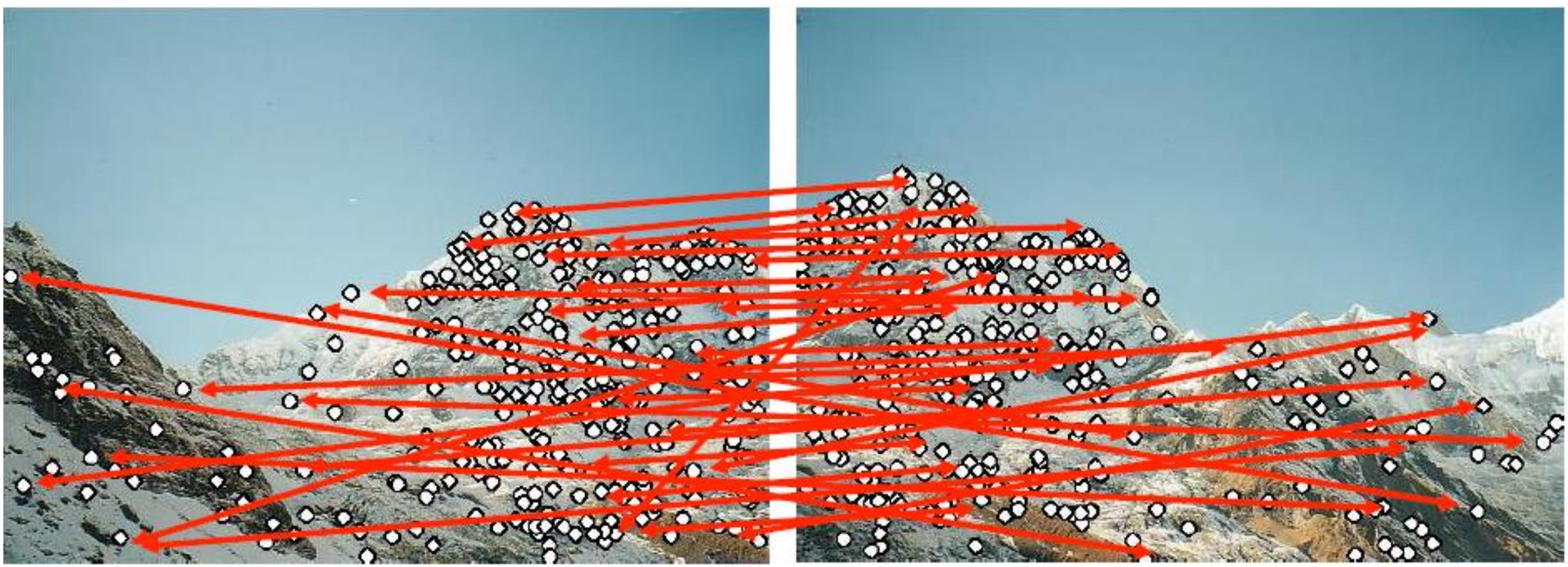
- Extract features



Courtesy: Cyrill Stachniss

RANSAC for Alignment

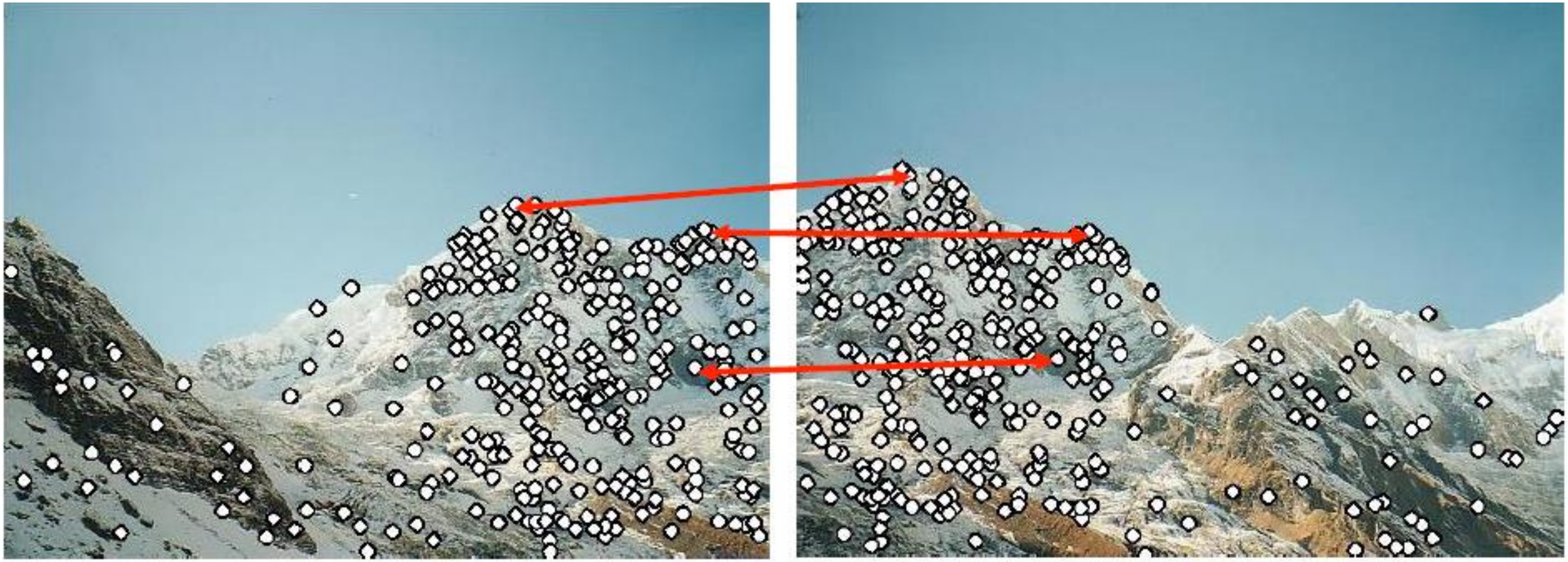
- Extract features
- Compute putative matches



Courtesy: Cyrill Stachniss

RANSAC for Alignment

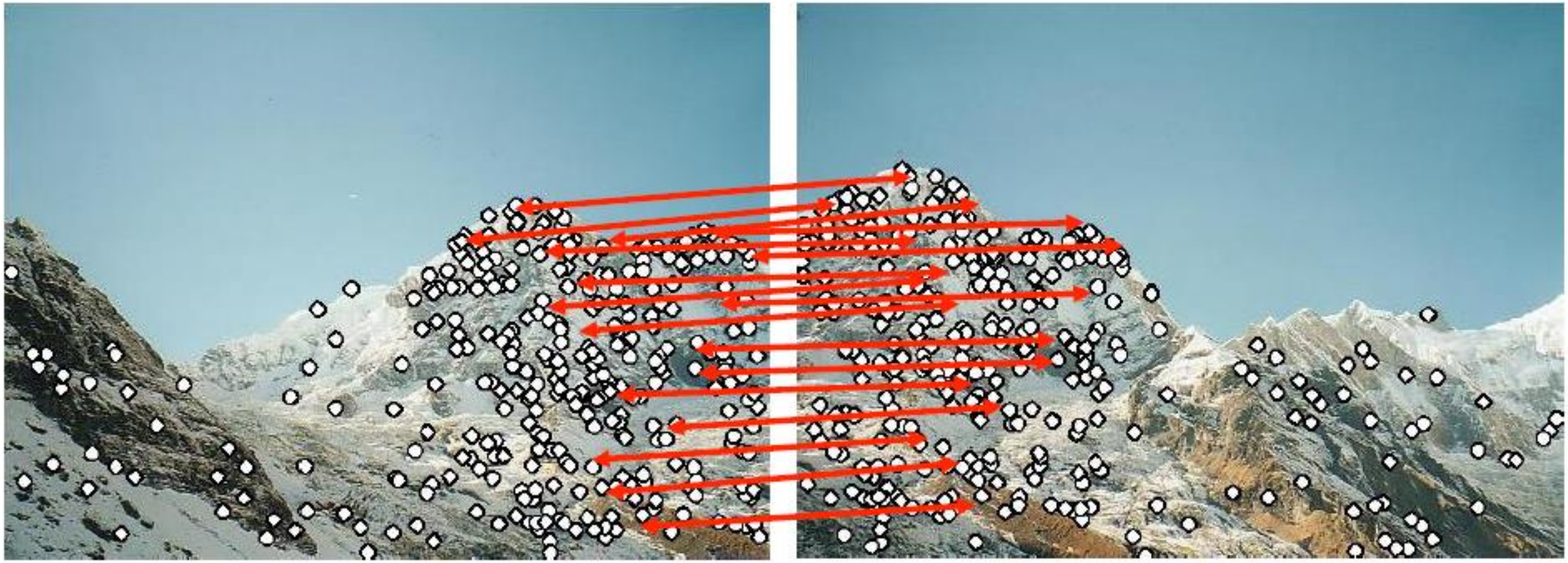
- Extract features
- Compute putative matches
- Loop:
  - Hypothesize transformation T
  - Verify transformation (search for other matches consistent with T)



Courtesy: Cyrill Stachniss

RANSAC for Alignment

- Extract features
- Compute putative matches
- Loop:
  - Hypothesize transformation T
  - Verify transformation (search for other matches consistent with T)

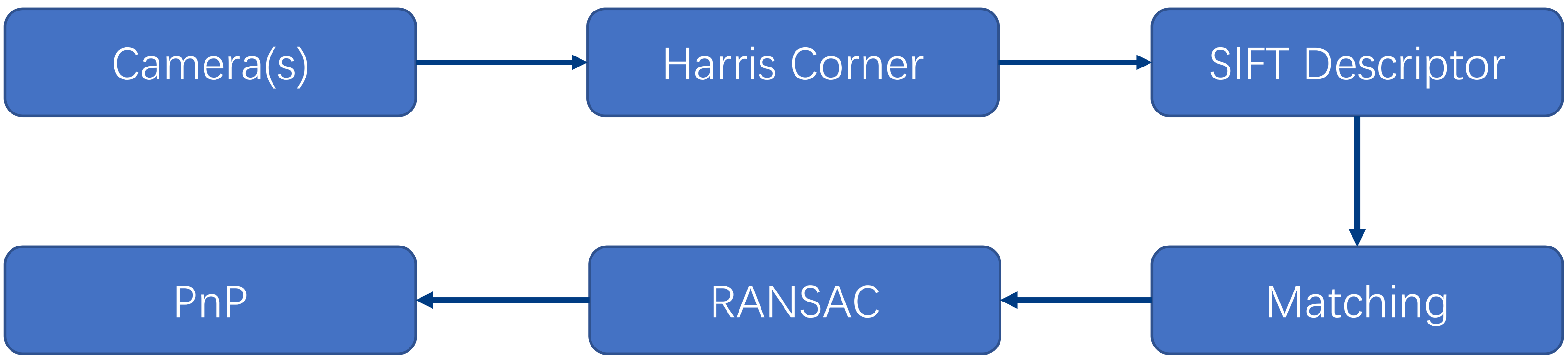


Courtesy: Cyrill Stachniss

Visual Odometry

Pipeline

- A mini pipeline, but can work

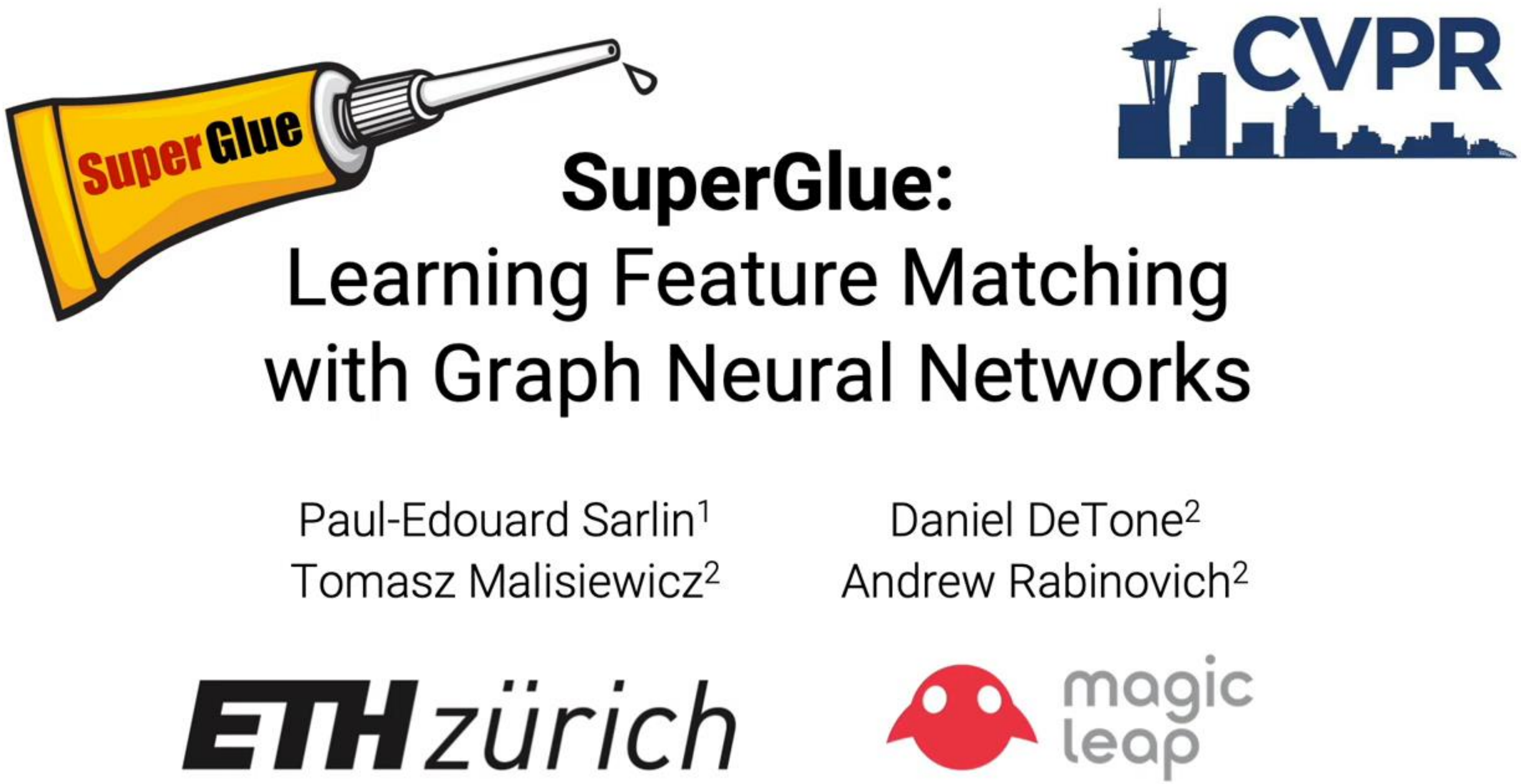


Visual Odometry

Courtesy: YouTube

Deep Learning-based

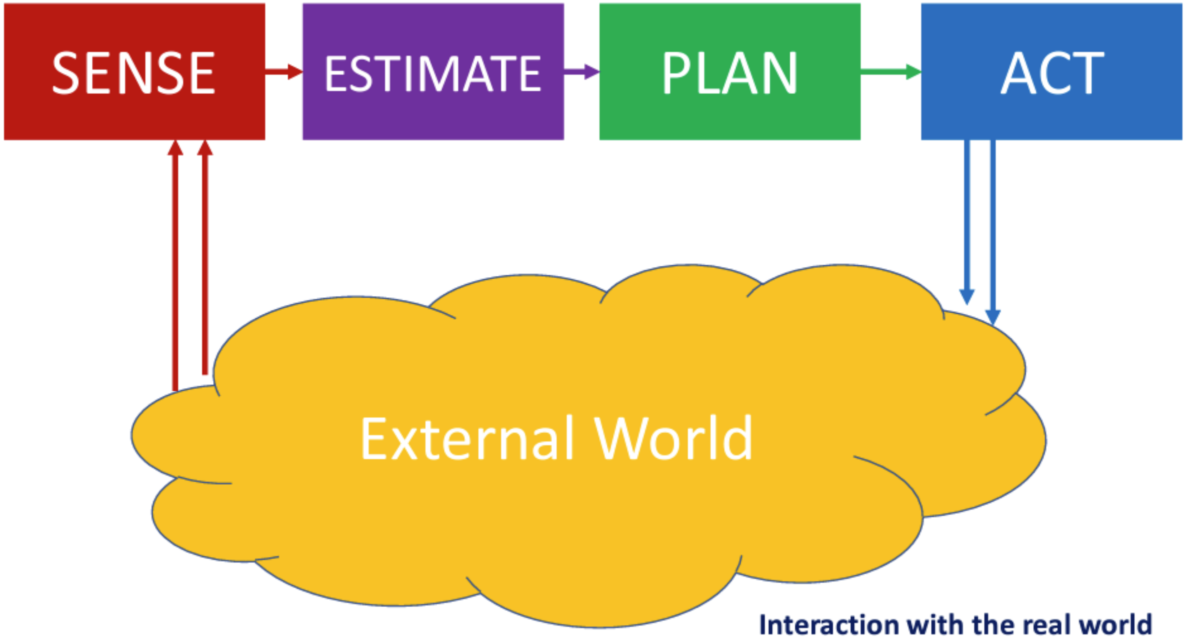
- Sarlin PE, DeTone D, Malisiewicz T, Rabinovich A. Superglue: Learning feature matching with graph neural networks. CVPR 2020 (pp. 4938-4947).



Courtesy: Super Glue

Next Lecture

- Sensing + Estimation ☺
- Planning



Interaction with the real world introduces uncertainty!