Introduction to Image Matching

Ersin Çine

Image matching

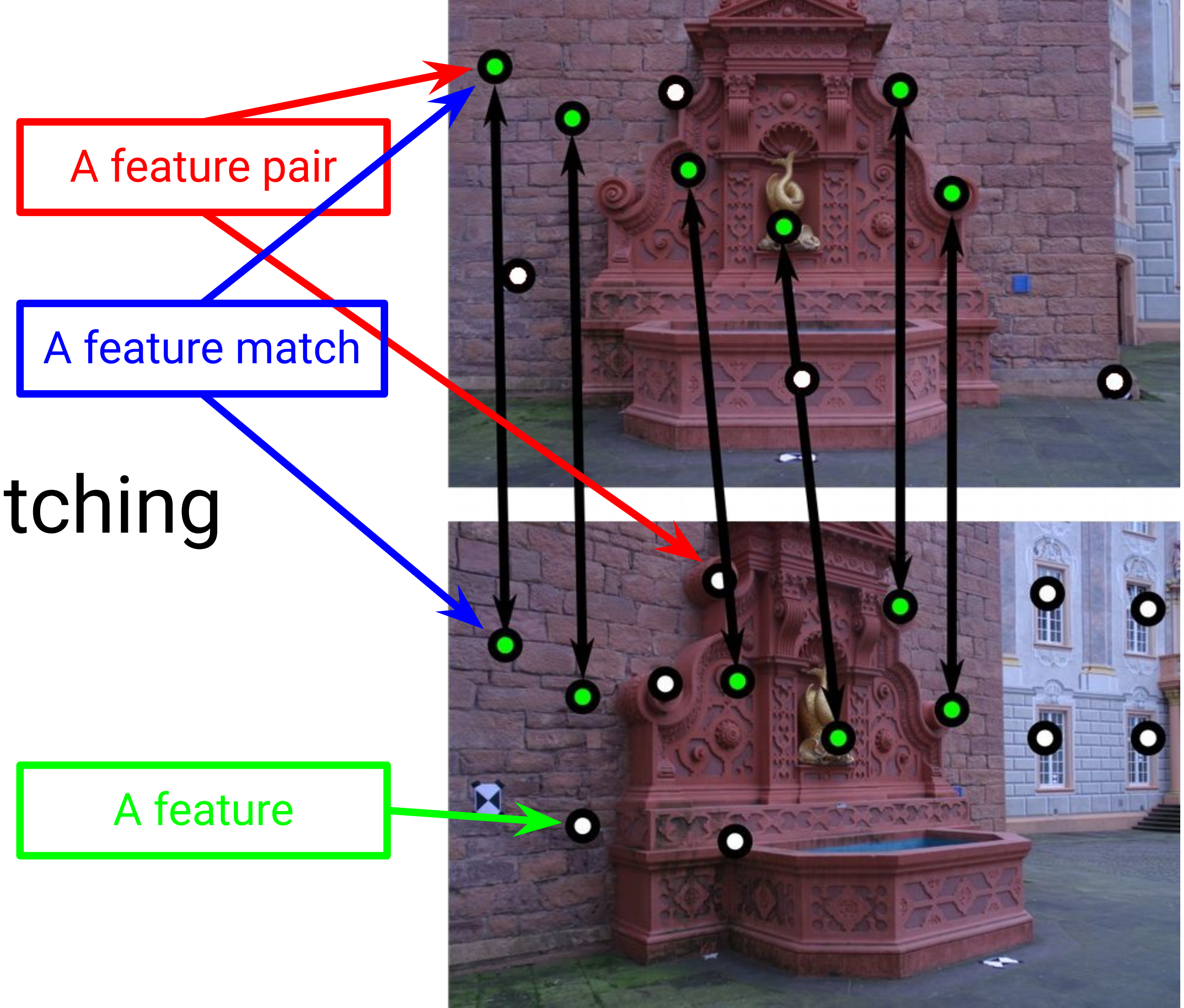


Image matching paradigms

- Sparse matching
- Dense matching

Image matching paradigms

- Sparse matching
  - Descriptor matching with mismatch removal
    1. **Promising match set construction** (using similarities of descriptors)
    2. **Mismatch removal** (local and/or global geometric constraints)

- Dense matching

Image matching paradigms

- Sparse matching
  - Descriptor matching with mismatch removal
    1. **Promising match set construction** (using similarities of descriptors)
    2. **Mismatch removal** (local and/or global geometric constraints)
       - **Resampling-based methods** (e.g., epipolar geometry) ✪
       - **Non-parametric model-based methods** (e.g., motion coherence)
       - **Relaxed methods** (e.g., complex deformations)
- Dense matching

The most popular pipeline **(Part 1/2)**

The most popular pipeline **(Part 1/2)**

(1)
detect
local features

(handcrafted or
learned features)

The most popular pipeline **(Part 1/2)**

(1)
detect
local features

(handcrafted or learned features)

(2)
describe
local features

(handcrafted or learned descriptors)

40	0	...	13

11	95	...	2

⋮

The most popular pipeline **(Part 2/2)**

Match local features

(3) **Select** promising pairs

e.g., nearest neighbors in descriptor space

(4)

The most popular pipeline **(Part 2/2)**

Match local features

(3) **Select** promising pairs

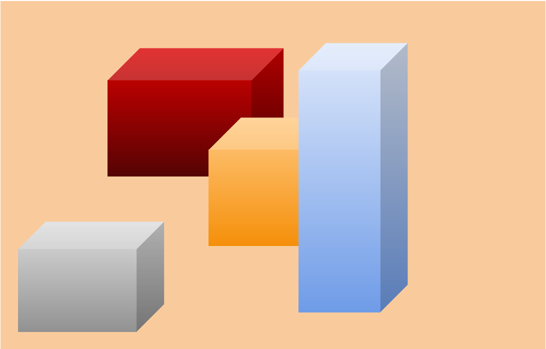
e.g., nearest neighbors in descriptor space

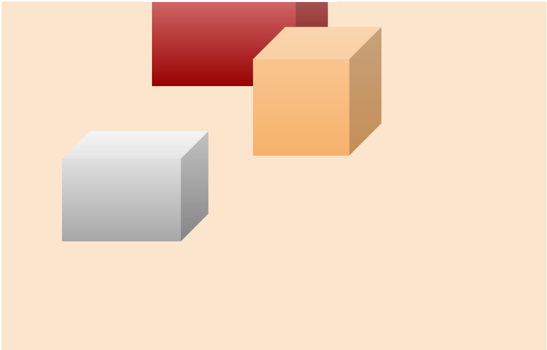
(4) Remove outliers & estimate geometric parameters

RANdom **SA**mple **C**onsensus (RANSAC) with DLT, etc.

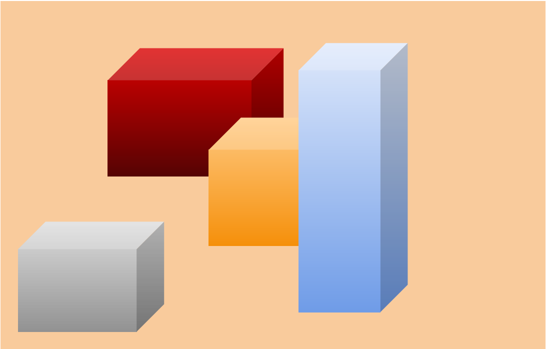
What? (Maximum consensus) Find a large subset of promising pairs in which there is a consensus on model parameters.
How? (RANSAC) Draw a random minimal subset of promising pairs, compute a geometric model, and count the promising pairs that are compatible with this model. Do this many times.

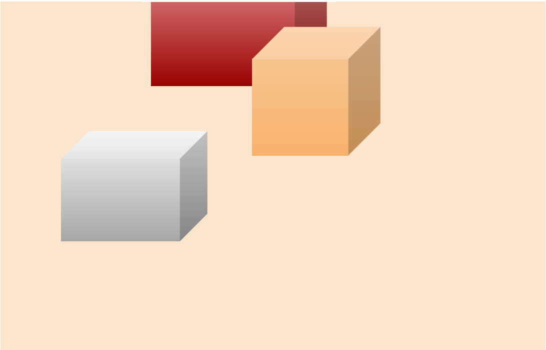
Two images of the same scene





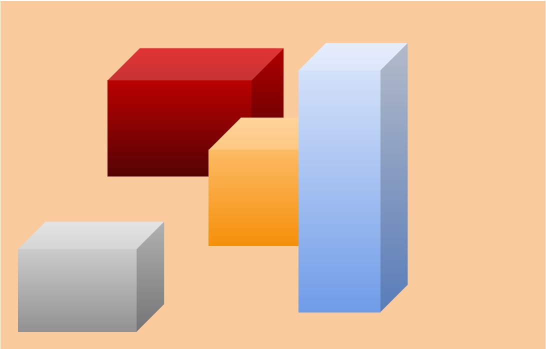
Two images of the same scene

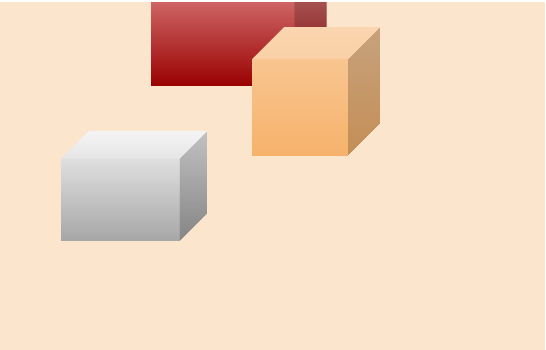




- Observe objects in the scene: **Several boxes**

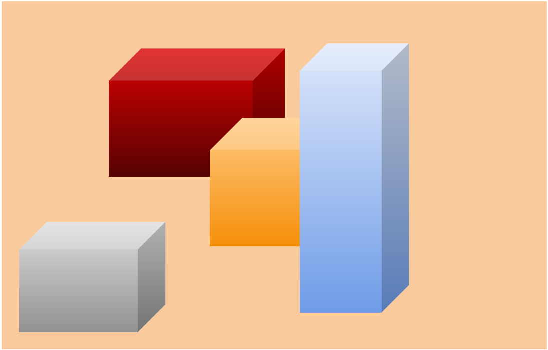
Two images of the same scene

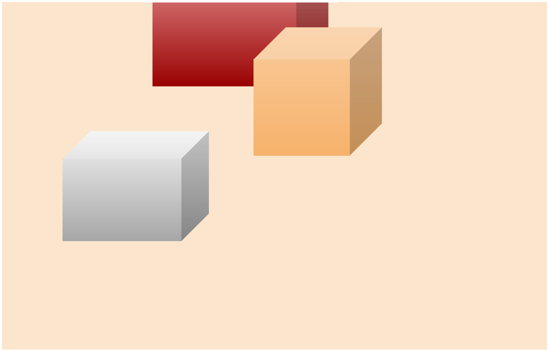




- Observe objects in the scene: **Several boxes**
- Notice dynamic environment: **Blue box gone**
-
-

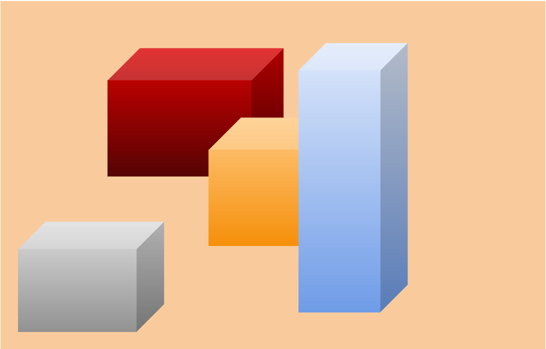
Two images of the same scene

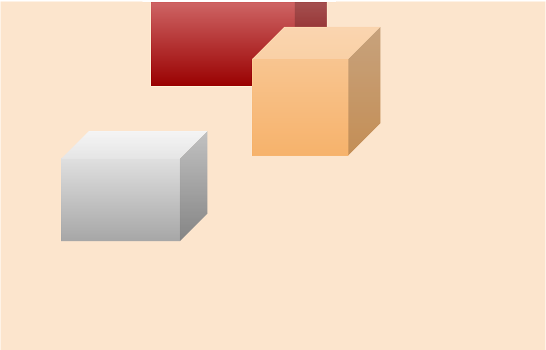




- Observe objects in the scene: **Several boxes**
- Notice dynamic environment: **Blue box gone**
- Note photometric transformation: **All pixels brighter**
-

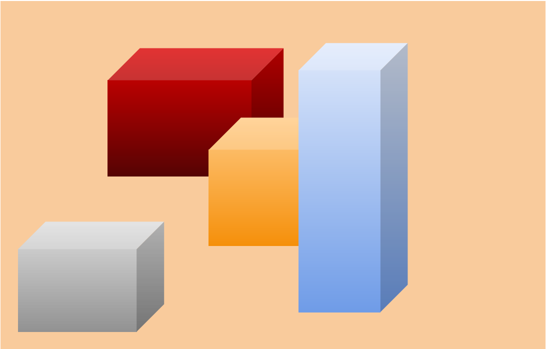
Two images of the same scene

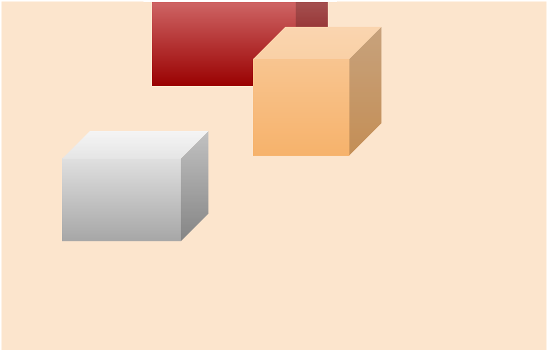




- Observe objects in the scene: **Several boxes**
- Notice dynamic environment: **Blue box gone**
- Note photometric transformation: **All pixels brighter**
- Estimate geometric transformation: $\mathbf{(t_x, t_y)}$

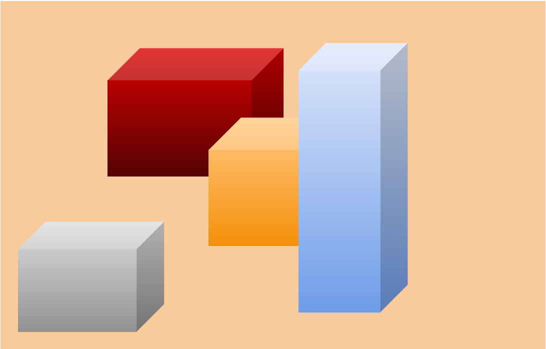
Two images of the same scene

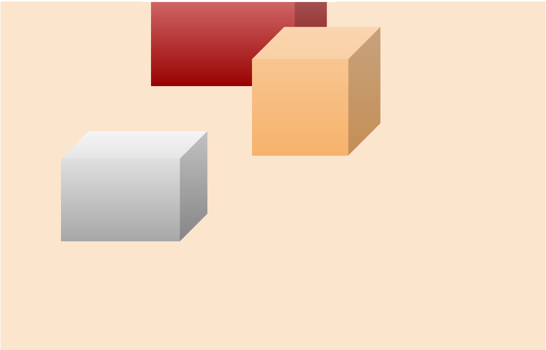




- Observe objects in the scene: **Several boxes**
  Normally: objects with various shapes and textures
- Notice dynamic environment: **Blue box gone**
- Note photometric transformation: **All pixels brighter**
- Estimate geometric transformation: **(t_x, t_y)**

Two images of the same scene

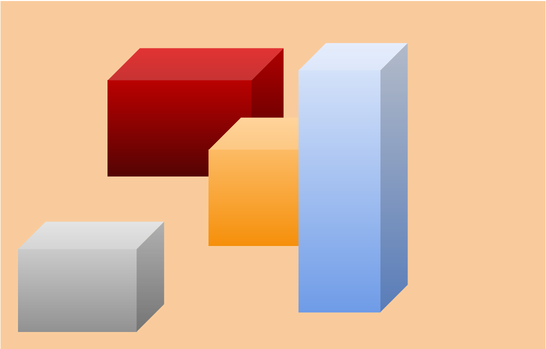


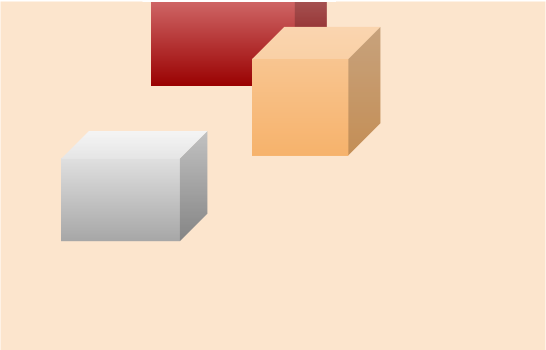


- Observe objects in the scene: **Several boxes**
  Normally: objects with various shapes and textures
- Notice dynamic environment: **Blue box gone**
  Normally: both images can contain occlusions
- Note photometric transformation: **All pixels brighter**

- Estimate geometric transformation: $(\mathbf{t_x}, \mathbf{t_y})$

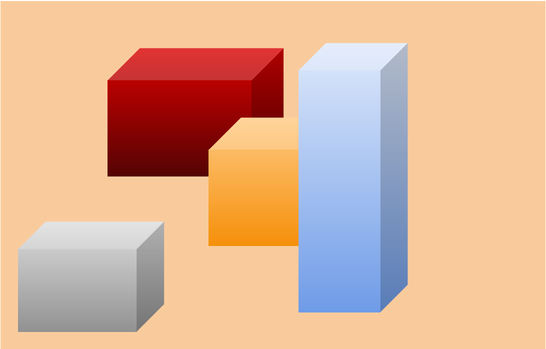
Two images of the same scene

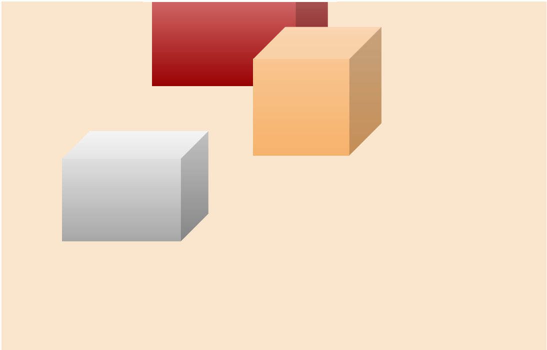




- Observe objects in the scene: **Several boxes**
  Normally: objects with various shapes and textures
- Notice dynamic environment: **Blue box gone**
  Normally: both images can contain occlusions
- Note photometric transformation: **All pixels brighter**
  Normally: more complicated transformations
- Estimate geometric transformation: $\mathbf{(t_x, t_y)}$

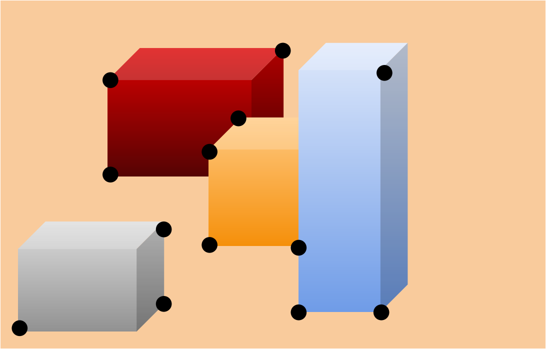
Two images of the same scene

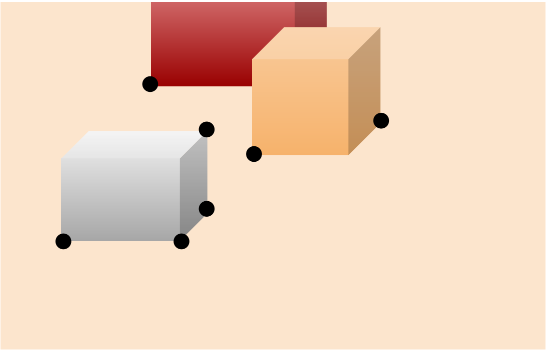




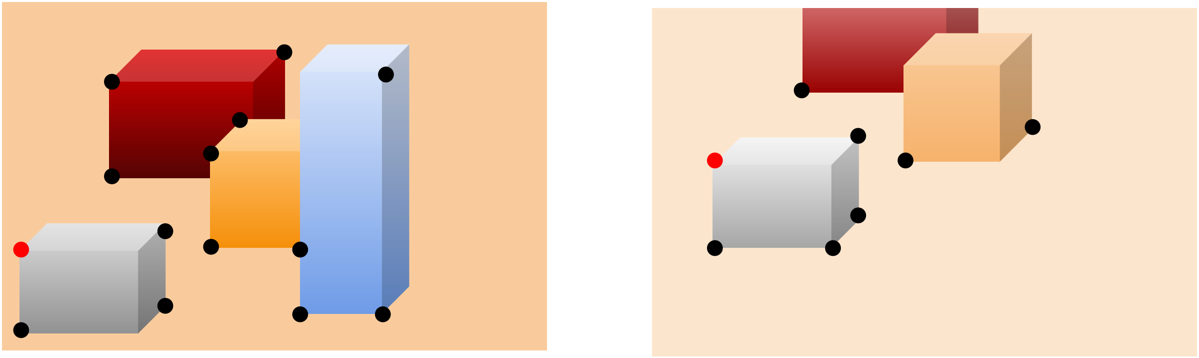
- Observe objects in the scene: **Several boxes**
  Normally: objects with various shapes and textures
- Notice dynamic environment: **Blue box gone**
  Normally: both images can contain occlusions
- Note photometric transformation: **All pixels brighter**
  Normally: more complicated transformations
- Estimate geometric transformation: $(\mathbf{t_x}, \mathbf{t_y})$
  Normally: more complicated transformations and smaller overlap

(1) Detect local features



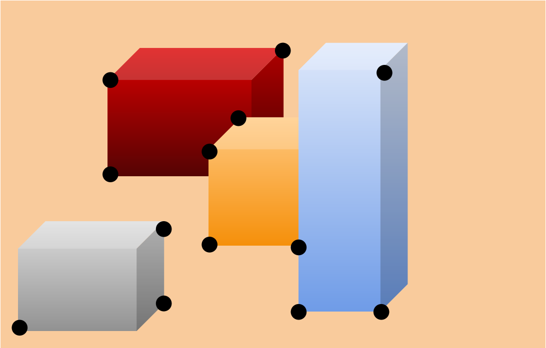


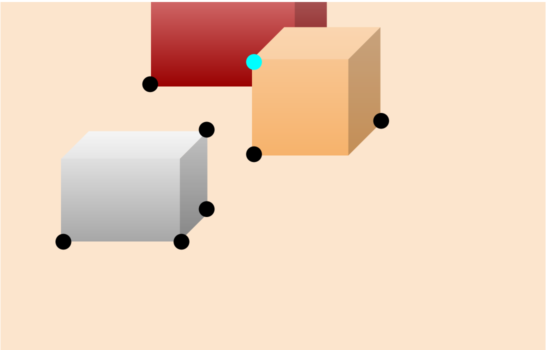
(1) Detect local features



- We failed to detect…
-
-

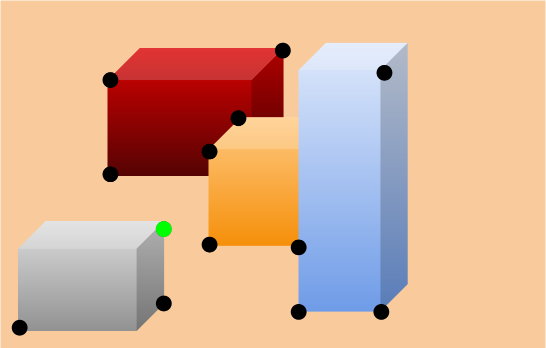
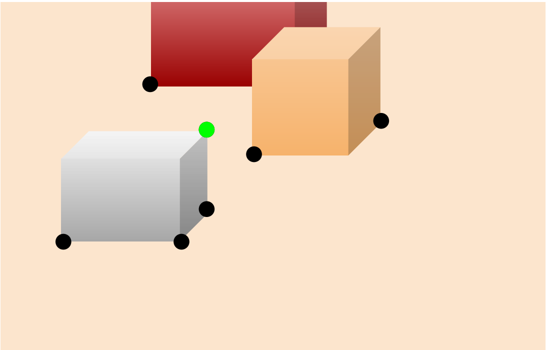
(1) Detect local features





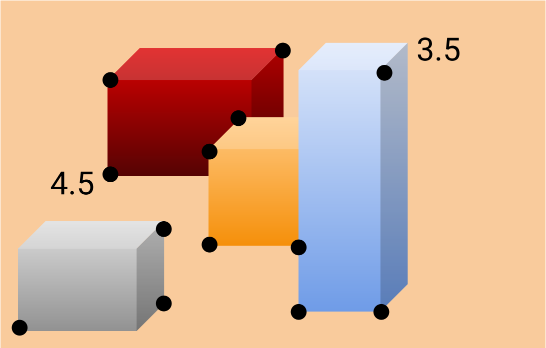
- We failed to detect…
- We failed to repeat…
-

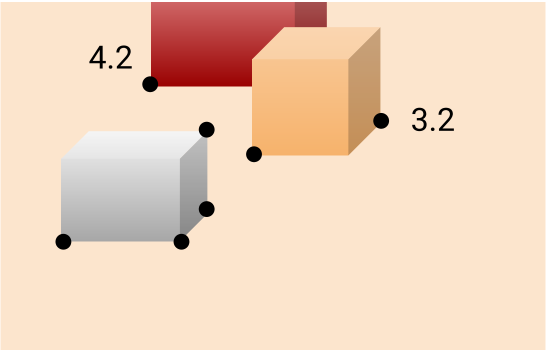
(1) Detect local features

- We failed to detect…
- We failed to repeat…
- We failed to localize…

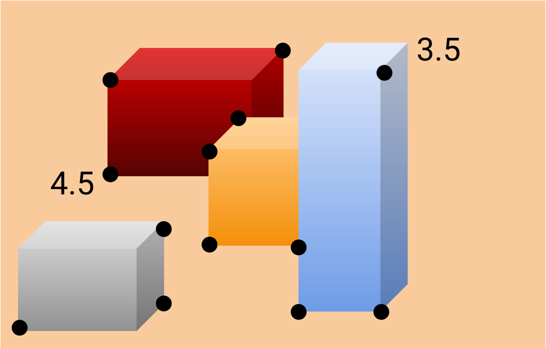
(2) Describe local features

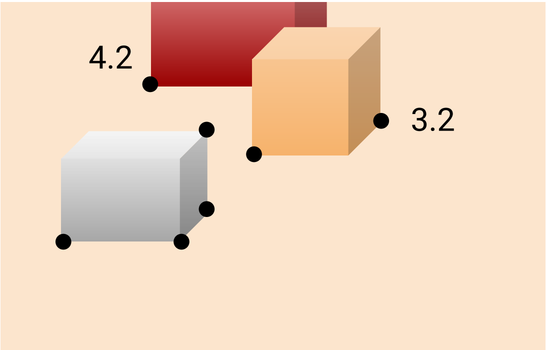




- We described each feature with a single number.
  Normally: a vector
-

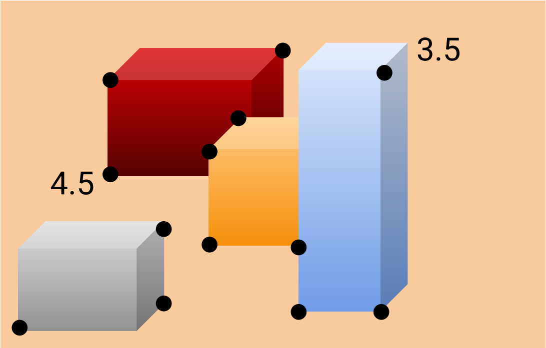
(2) Describe local features

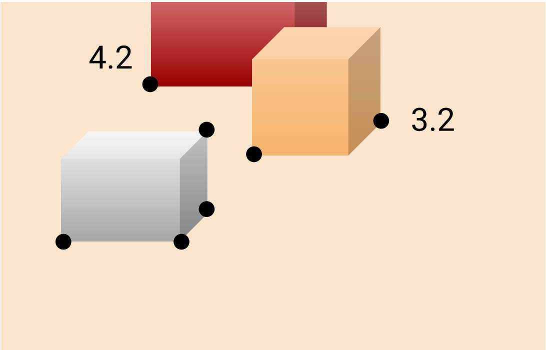




- We described each feature with a single number.
  Normally: a vector
- Descriptors must be invariant to some properties:
  - geometric: rotation, scale, etc.
  - photometric: brightness, exposure, etc.

(3) Select promising pairs

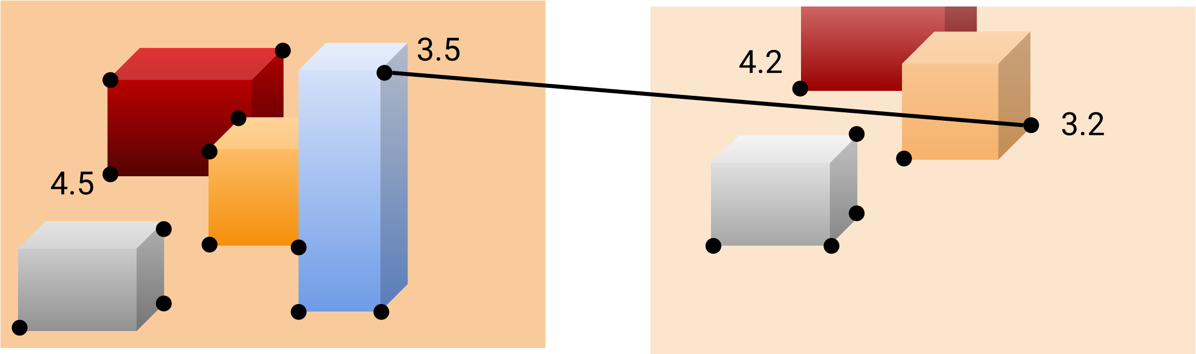




1.

2.

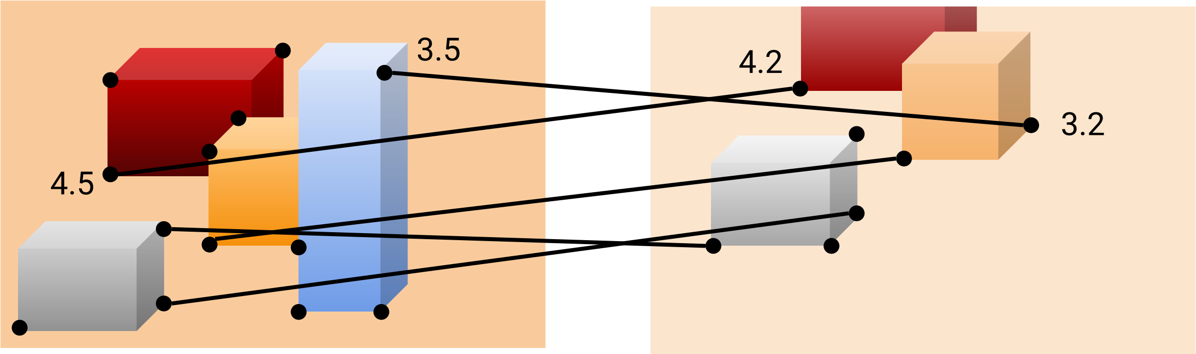
(3) Select promising pairs



1. Find the nearest neighbors.
   Normally: approximate nearest neighbors (using k-d trees) or other methods

2.

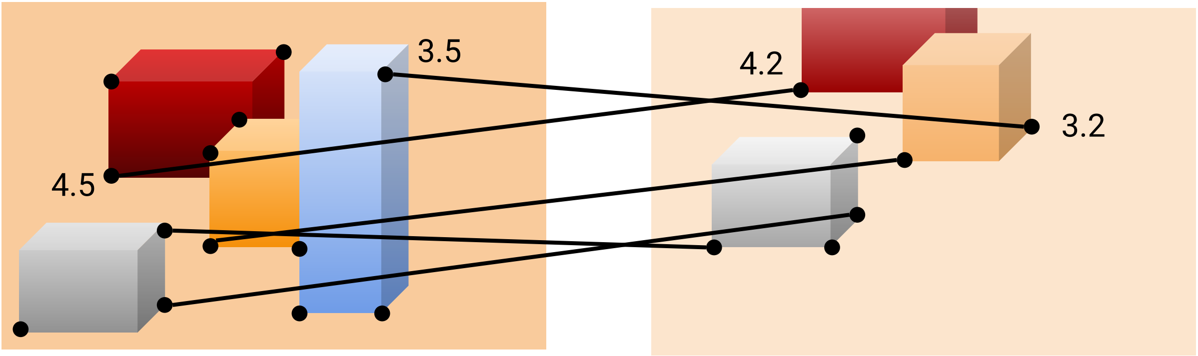
(3) Select promising pairs



1. Find the nearest neighbors.
   Normally: approximate nearest neighbors (using k-d trees) or other methods

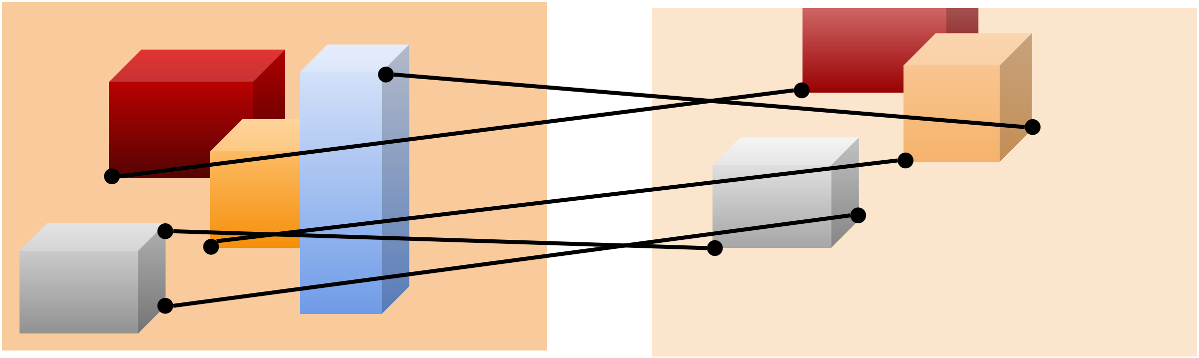
2.

(3) Select promising pairs



1. Find the nearest neighbors.
   Normally: approximate nearest neighbors (using k-d trees) or other methods
2. Maybe eliminate some of them.
   (e.g., ratio test for eliminating duplicate features)

(4) Remove outliers & estimate geometric parameters

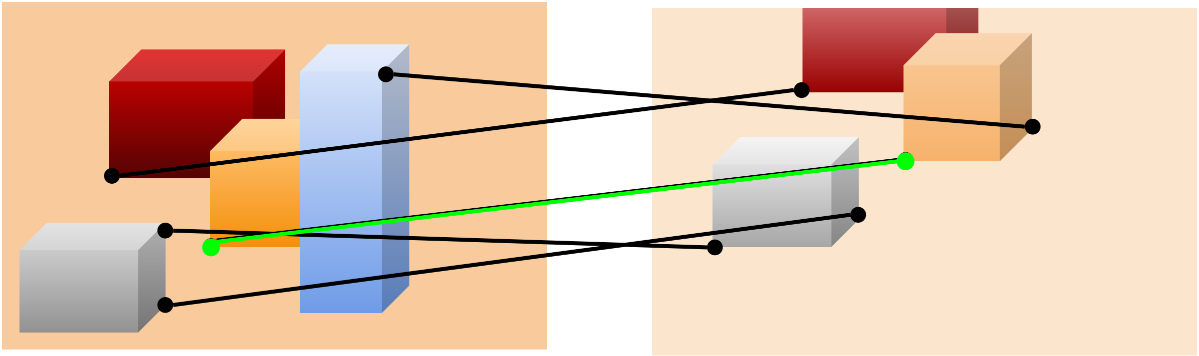


1.

2.

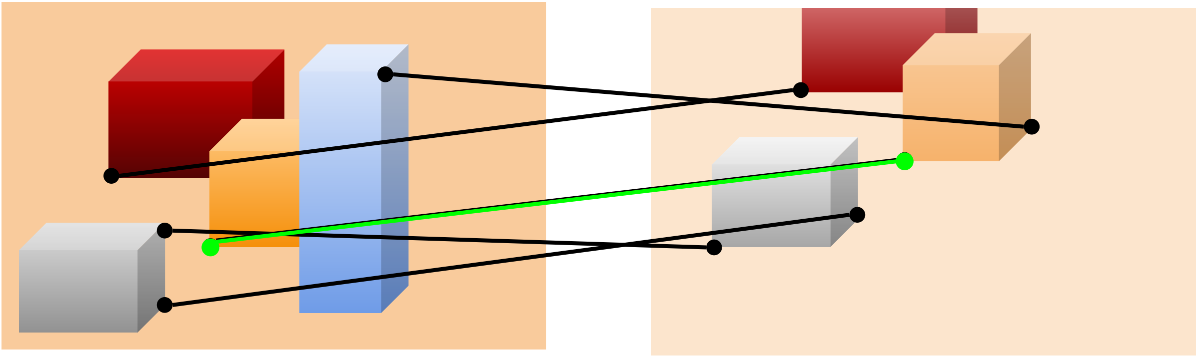
3.

(4) Remove outliers & estimate geometric parameters



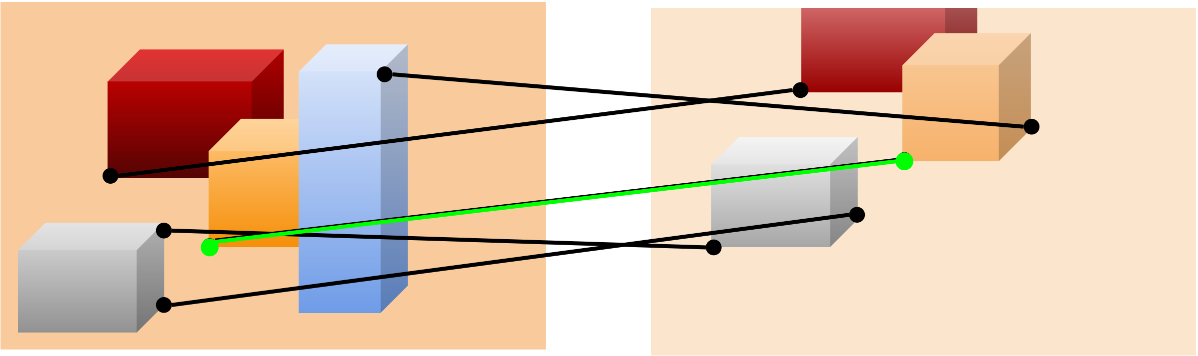
1. Draw a random minimal subset (1 pair is enough for translation)
2.
3.

(4) Remove outliers & estimate geometric parameters



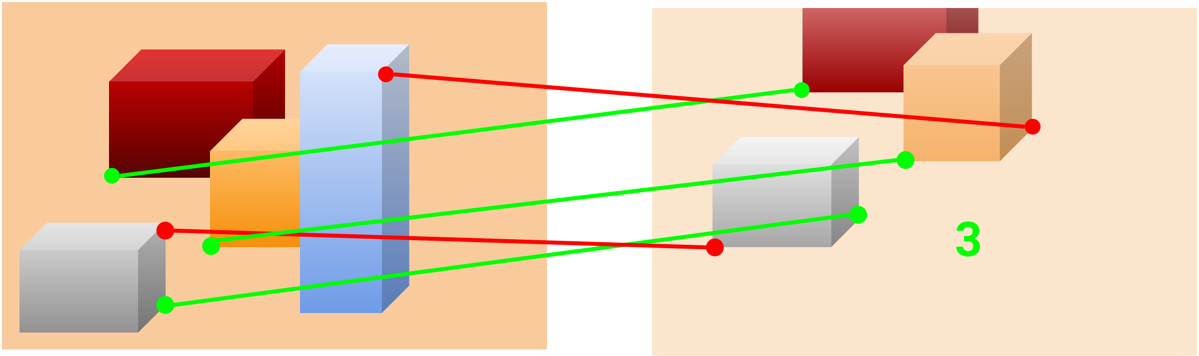
1. Draw a random minimal subset (1 pair is enough for translation)
2. Compute geometric model ($t_x = x_1 - x_0$ and $t_y = y_1 - y_0$)
3.

(4) Remove outliers & estimate geometric parameters



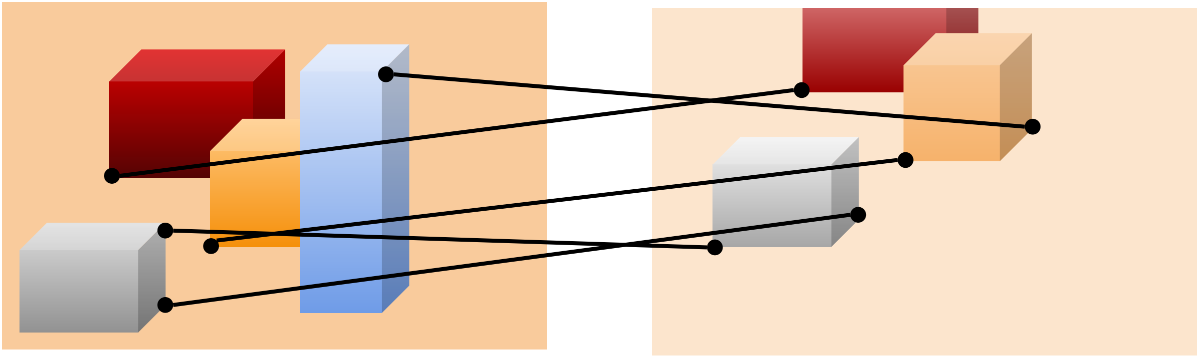
1. Draw a random minimal subset (1 pair is enough for translation)
2. Compute geometric model ($t_x = x_1 - x_0$ and $t_y = y_1 - y_0$)
3. Determine and count inliers (reprojection error < threshold)

(4) Remove outliers & estimate geometric parameters



1. Draw a random minimal subset (1 pair is enough for translation)
2. Compute geometric model ($t_x = x_1 - x_0$ and $t_y = y_1 - y_0$)
3. Determine and count inliers (reprojection error < threshold)

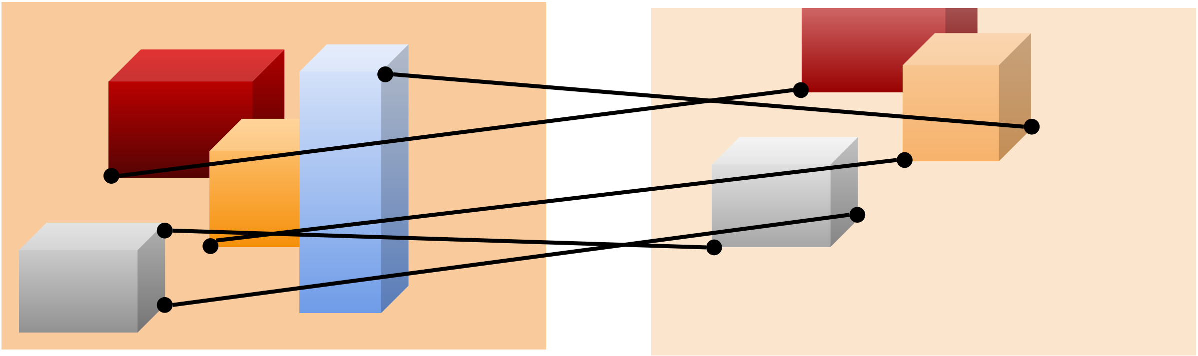
(4) Remove outliers & estimate geometric parameters



1. Draw a random minimal subset (1 pair is enough for translation)
2. Compute geometric model ($t_x = x_1 - x_0$ and $t_y = y_1 - y_0$)
3. Determine and count inliers (reprojection error < threshold)

Do this many times.
Pick the model with the highest support.

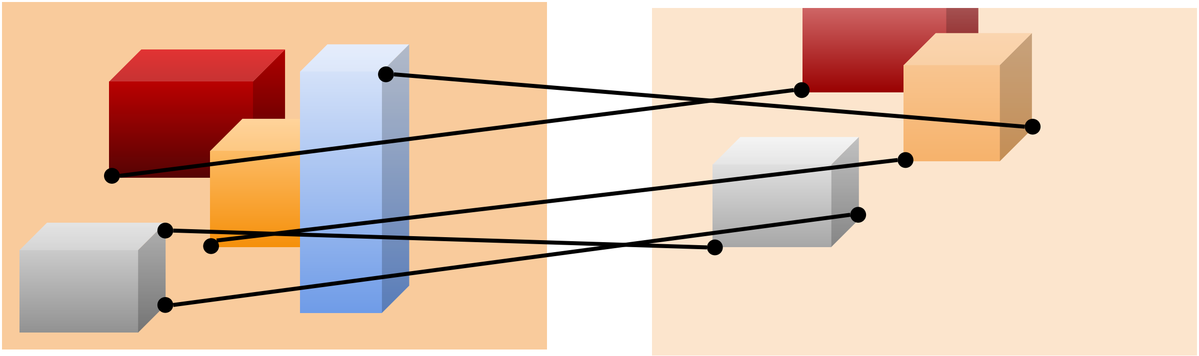
(4) Remove outliers & estimate geometric parameters



1. Draw a random minimal subset (1 pair is enough for translation)
2. Compute geometric model ($t_x = x_1 - x_0$ and $t_y = y_1 - y_0$)
3. Determine and count inliers (reprojection error < threshold)

Do this many times.
Pick the model with the highest support.
Estimate parameters again minimizing sum of squared errors on all inliers.

(4) Remove outliers & estimate geometric parameters



There are many heuristic improvements:

- Better sampling
- Better quality functions
- Local optimization
- Pre-emptive verification
- ...

Selected Heuristics

Different Sampling Heuristics
How to reduce the expected number of iterations to find an all-inlier sample?
RANSAC (1981) Drawing random samples uniformly
NAPSAC (2002) Sampling from the close data points
PROSAC (2005) Sampling from the most promising subset and growing the set progressively
EVSAC (2013) Assigning confidence values for data points using extreme value analysis
P-NAPSAC (2020) Combining the ideas of PROSAC and NAPSAC.

Selected Heuristics

Different Sampling Heuristics

How to reduce the expected number of iterations to find an all-inlier sample?
RANSAC (1981) Drawing random samples uniformly
NAPSAC (2002) Sampling from the close data points
PROSAC (2005) Sampling from the most promising subset and growing the set progressively
EVSAC (2013) Assigning confidence values for data points using extreme value analysis
P-NAPSAC (2020) Combining the ideas of PROSAC and NAPSAC.

Different Quality Heuristics

How good is a model?
RANSAC (1981) Number of inliers
LMedS (1984) Median of squared errors
MLESAC (2000) Maximum likelihood estimation
MSAC (2002) Truncated squared error
AC-RANSAC (2012) Estimating error threshold
MAGSAC++ (2020) Marginalization over a range of error thresholds

Selected Heuristics

Different Sampling Heuristics

How to reduce the expected number of iterations to find an all-inlier sample?
RANSAC (1981) Drawing random samples uniformly
NAPSAC (2002) Sampling from the close data points
PROSAC (2005) Sampling from the most promising subset and growing the set progressively
EVSAC (2013) Assigning confidence values for data points using extreme value analysis
P-NAPSAC (2020) Combining the ideas of PROSAC and NAPSAC.

Different Quality Heuristics

How good is a model?
RANSAC (1981) Number of inliers
LMedS (1984) Median of squared errors
MLESAC (2000) Maximum likelihood estimation
MSAC (2002) Truncated squared error
AC-RANSAC (2012) Estimating error threshold
MAGSAC++ (2020) Marginalization over a range of error thresholds

Different Optimization Heuristics

How to improve a model?
LO-RANSAC (2003) Inner RANSAC
RANSAAC (2016) Averaging good models
GC-RANSAC (2018) Graph-cut optimization