Mapping and Modeling with RGB-D Cameras

University of Washington

Dieter Fox

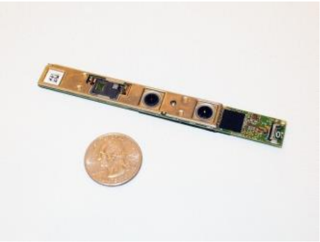
Outline

- Motivation
- RGB-D Mapping:
  1. Visual Odometry (frame-to-frame alignment)
  2. Loop Closure (revisiting places)
  3. Map representation (Surfels)

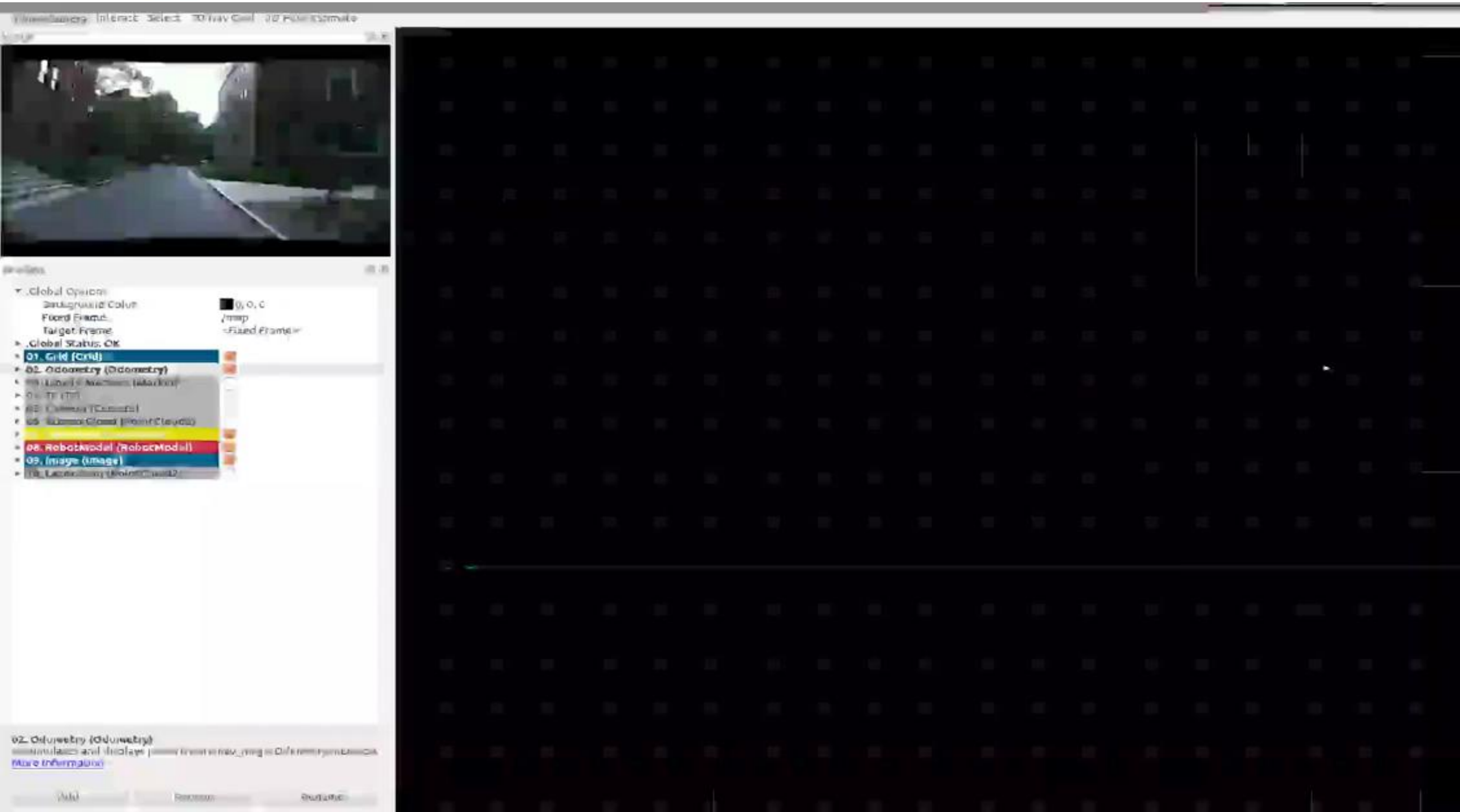
Outline

- Motivation
- RGB-D Mapping:
  1. Visual Odometry (frame-to-frame alignment)
  2. Loop Closure (revisiting places)
  3. Map representation (Surfels)

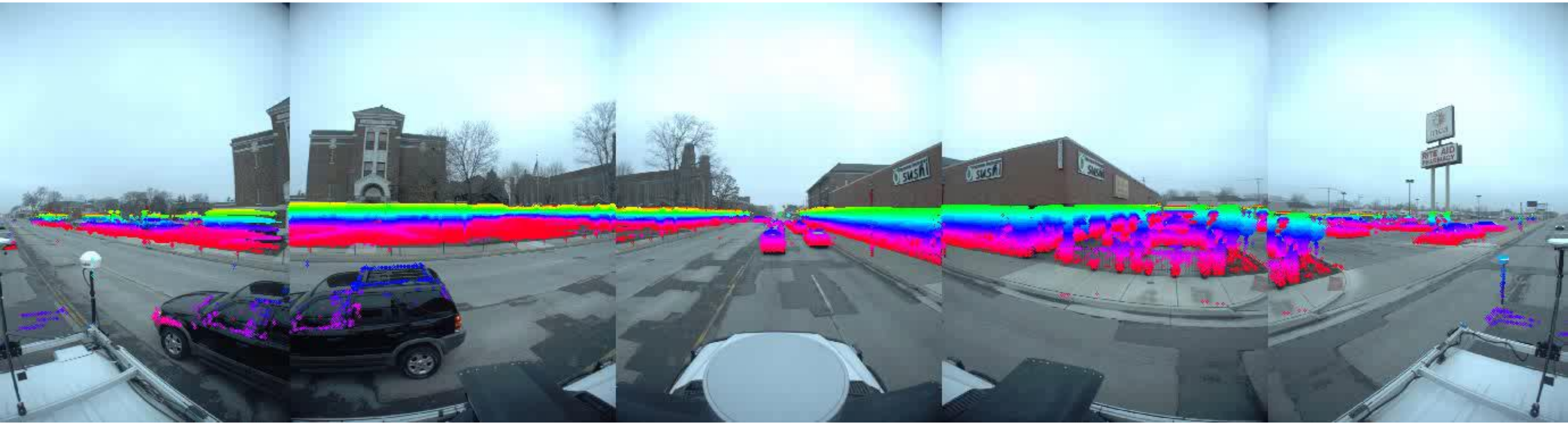
RGB-D (Kinect-style) Cameras



Multisense SL



Velodyne & LadyBug3



Motivation

- Tracking RGB-D camera motion and creating a 3D model has applications for
  - Rich interior maps
  - Robotics
    - Localization / Mapping
    - Manipulation
  - Augmented reality
  - 3D content creation

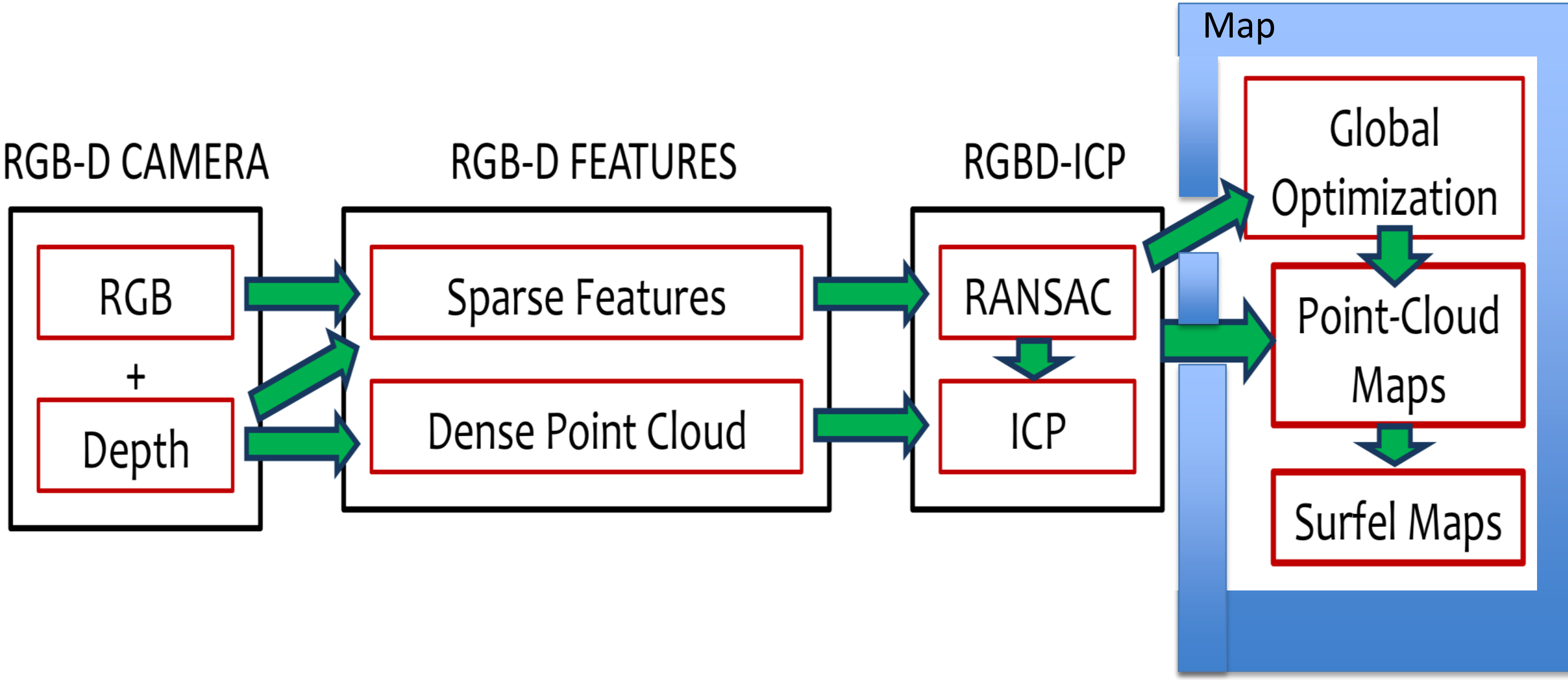
Goal

- Track the 3D motion of an RGB-D camera
- Build a useful and accurate model of the environment

Outline

- Motivation
- RGB-D Mapping:
  1. Frame-to-frame motion (visual odometry)
  2. Revisiting places (loop closure detection)
  3. Map representation (Surfels)

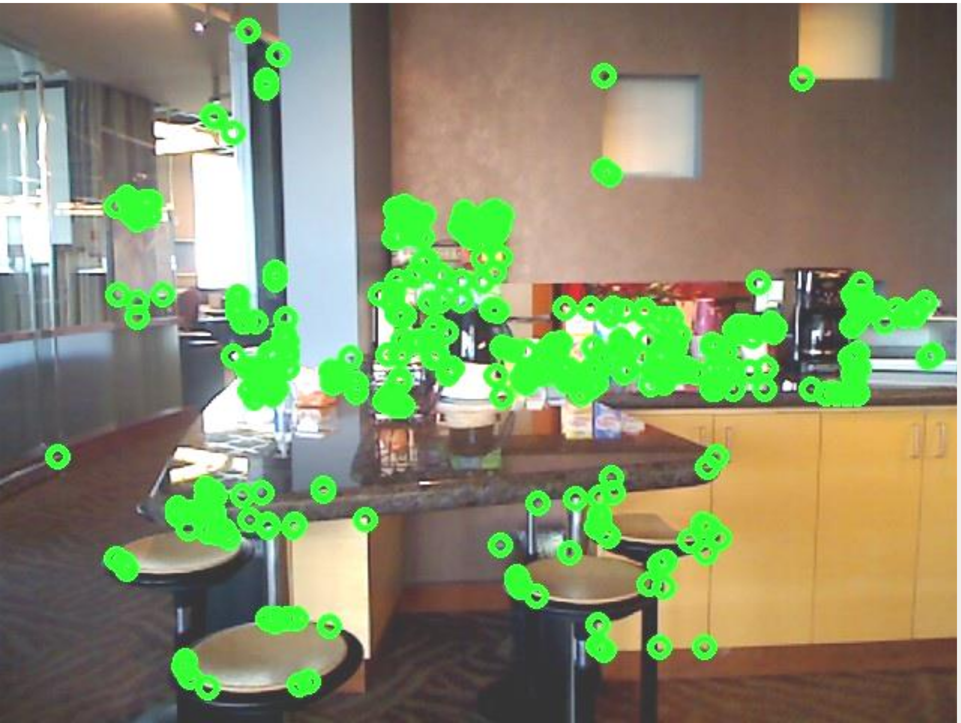
RGB-D Mapping Overview



RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments. Henry et al. ISER 2010
RGB-D Mapping: Using Kinect-style Depth Cameras for Dense 3D Modeling of Indoor Environments. Henry et al. IJRR 2012

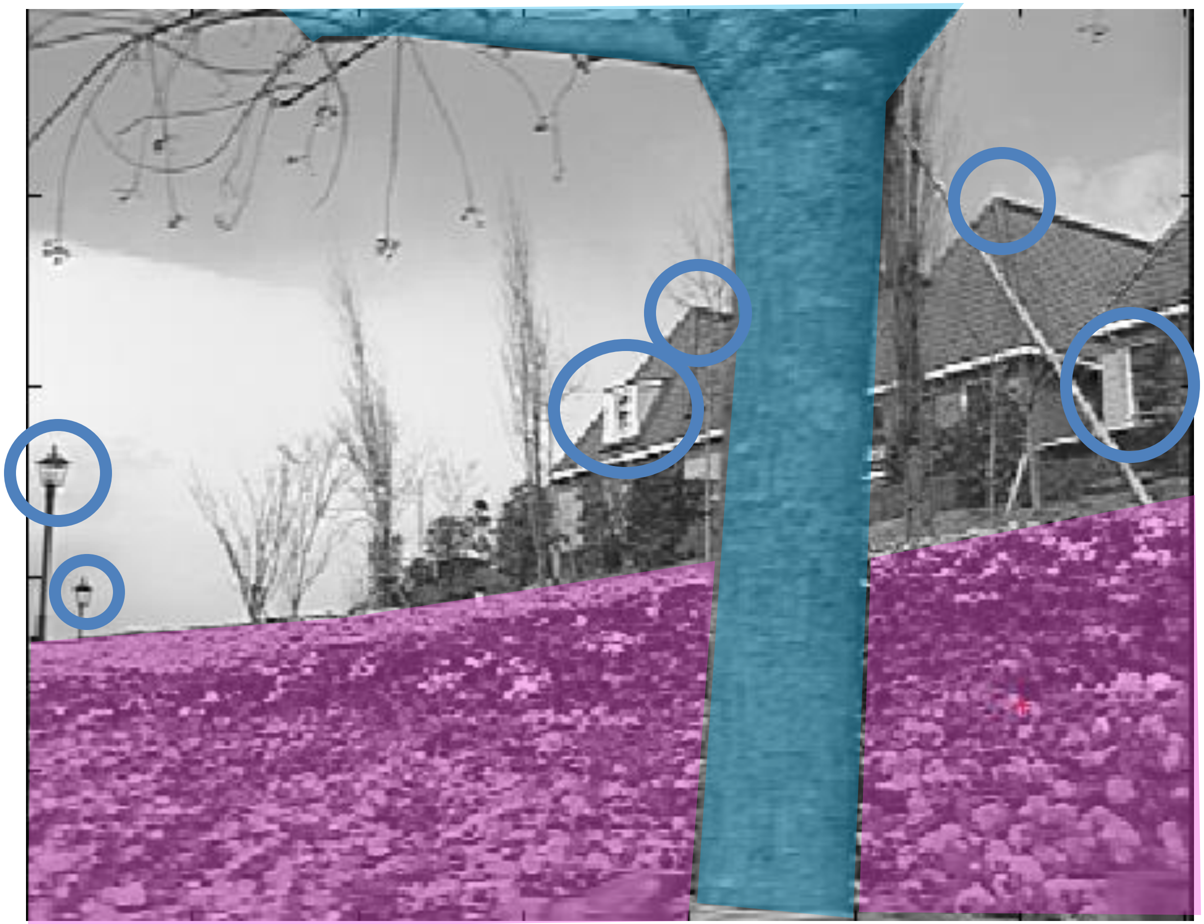
Visual Odometry

- Compute the motion between consecutive camera frames from visual feature correspondences.
- Visual features from RGB image have a 3D counterpart from depth image.



Visual Features

- Tree bark itself not really distinct
- Rocky ground not distinct
- Rooftops, windows, lamp post fairly distinct and should be easier to match across images



Say we have 2 images of this scene we'd like to align by matching local features

What would be good local features (ones easy to match)?

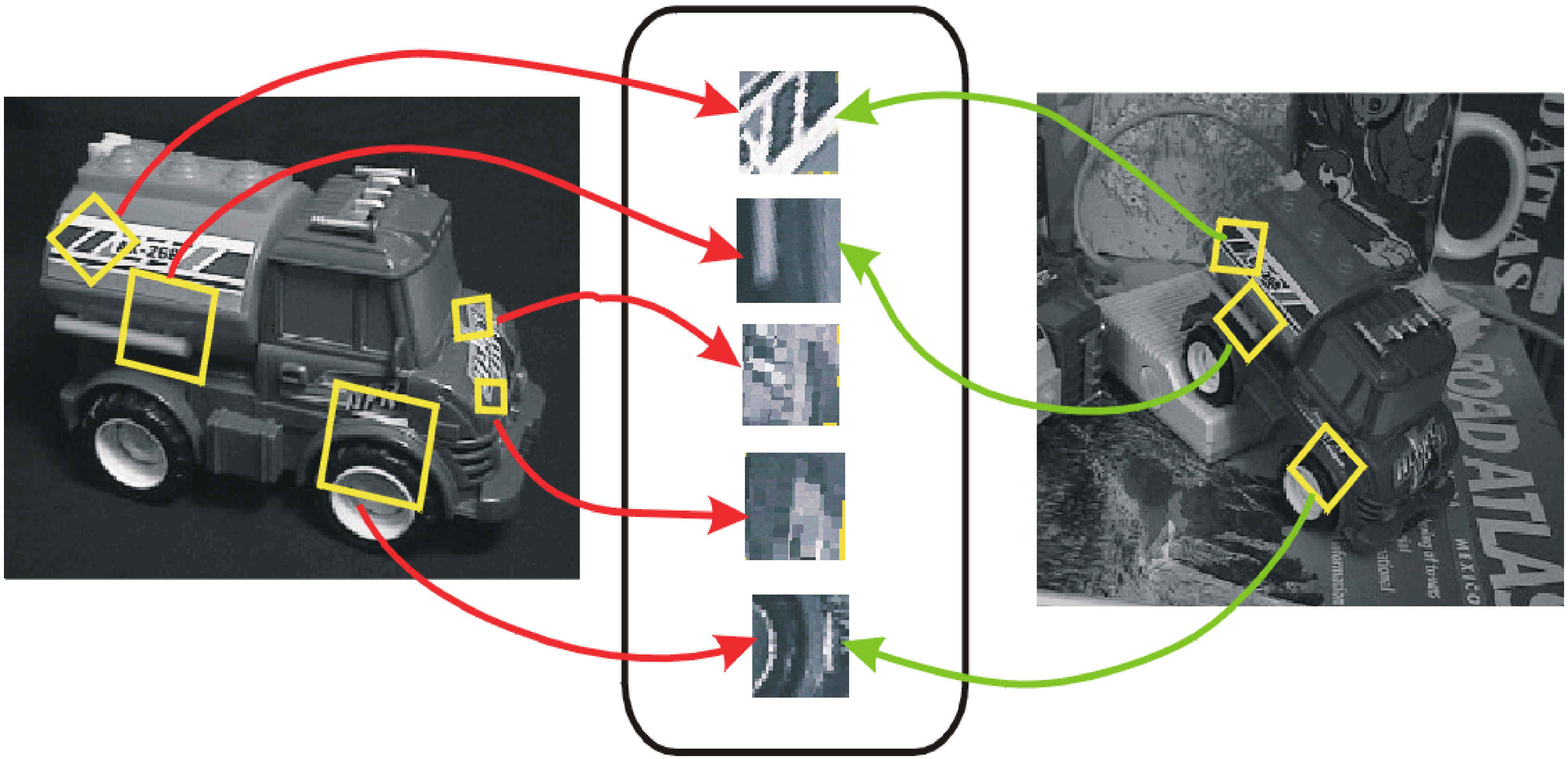
Courtesy: S. Seitz and R. Szeliski

Invariant local features

-Algorithm for finding points and representing their patches should produce similar results even when conditions vary

-Buzzword is "invariance"

- geometric invariance: translation, rotation, scale
- photometric invariance: brightness, exposure, …

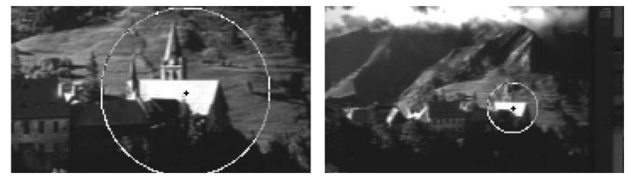


Feature Descriptors

Courtesy: S. Seitz and R. Szeliski

Robust visual features

- Goal: Detect distinctive features, maximizing repeatability
  - Scale invariance
    - Robust to changes in distance



  - Rotation invariance
    - Robust to rotations of camera

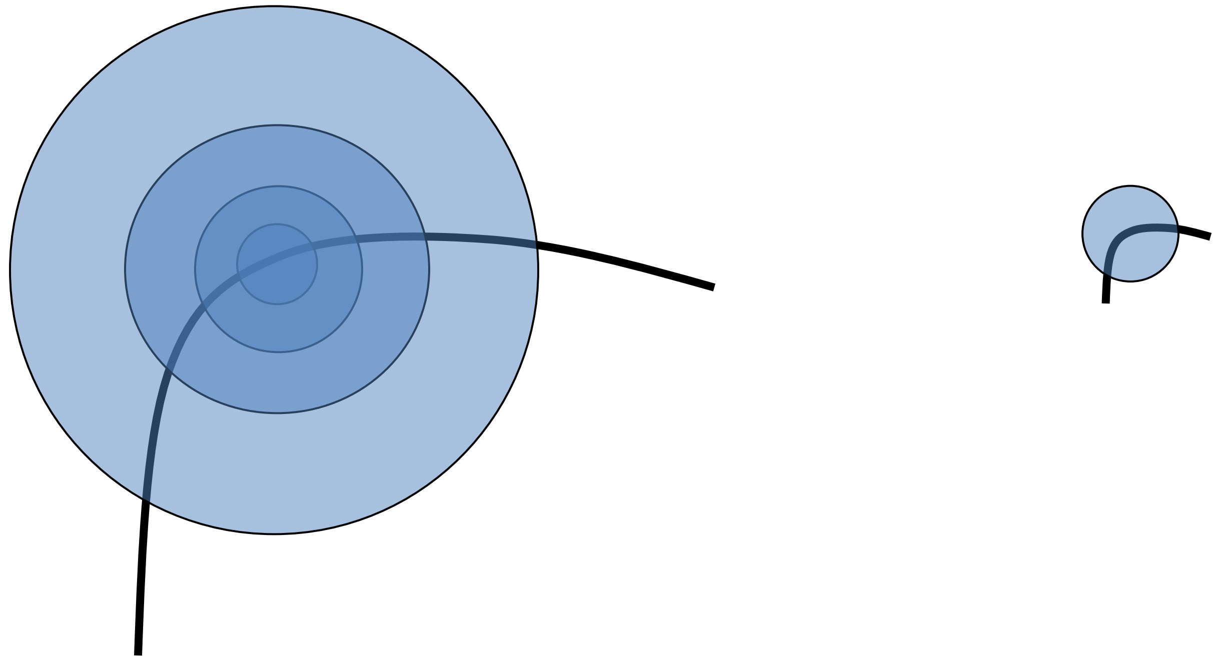
  - Affine invariance
    - Robust to tilting of camera

  - Brightness invariance
    - Robust to minor changes in illumination

  - Produce small descriptors that can be compared using simple mathematical operations
    - (SSE)
    - Euclidean distance

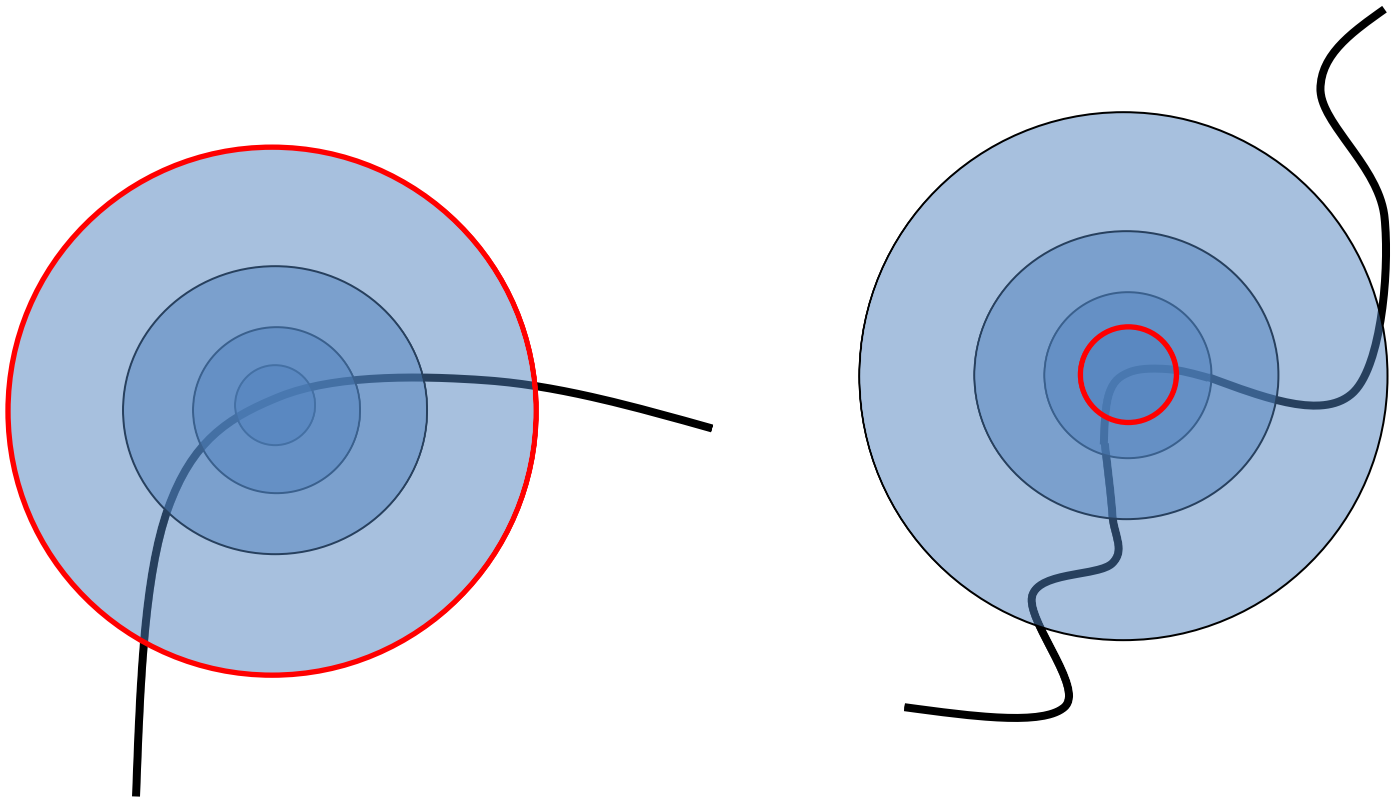
Scale Invariant Detection

- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images



Scale Invariant Detection

- The problem: how do we choose corresponding circles ***independently*** in each image?



Scale Invariant Detection

- Solution:
  – Design a function on the region (circle), which is “scale invariant” (the same for corresponding regions, even if they are at different scales)

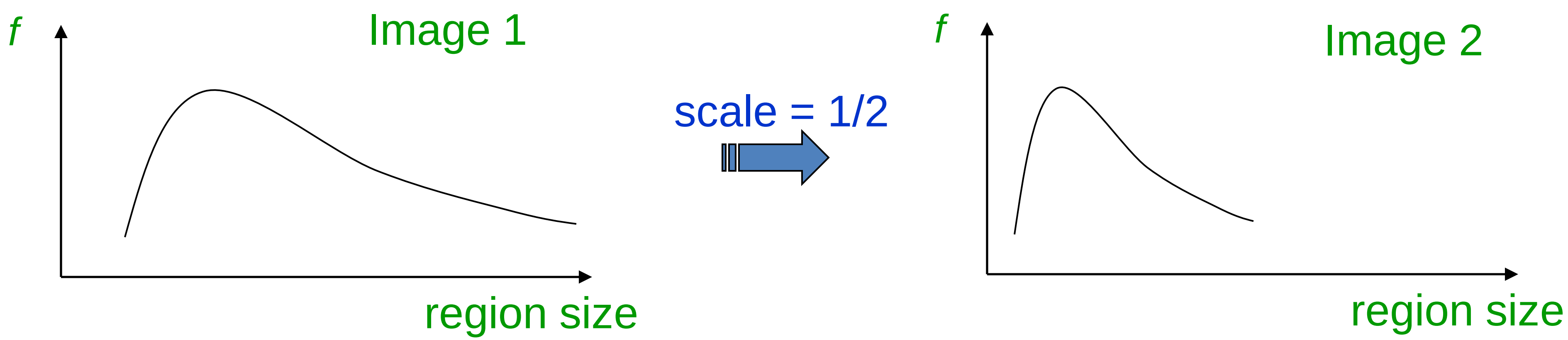
Example: average intensity. For corresponding regions (even of different sizes) it will be the same.

Scale Invariant Detection

- Solution:
  - Design a function on the region (circle), which is "scale invariant" (the same for corresponding regions, even if they are at different scales)

    Example: average intensity. For corresponding regions (even of different sizes) it will be the same.

  - For a point in one image, we can consider it as a function of region size (circle radius)



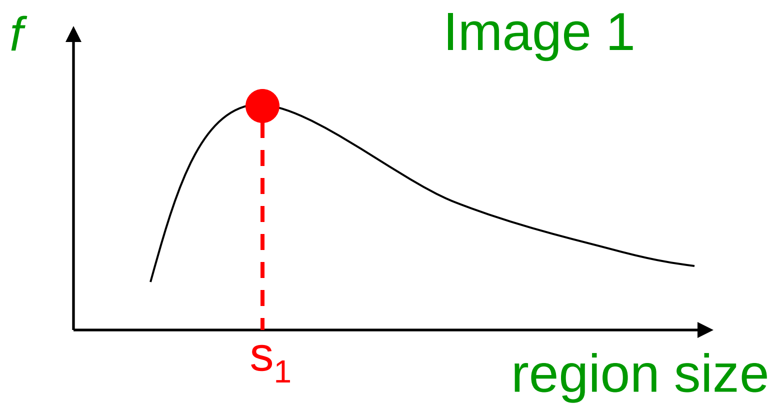
Scale Invariant Detection

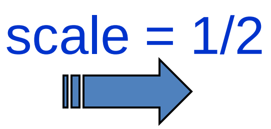
- Common approach:

  Take a local maximum of this function

  Observation: region size, for which the maximum is achieved, should be *invariant* to image scale.

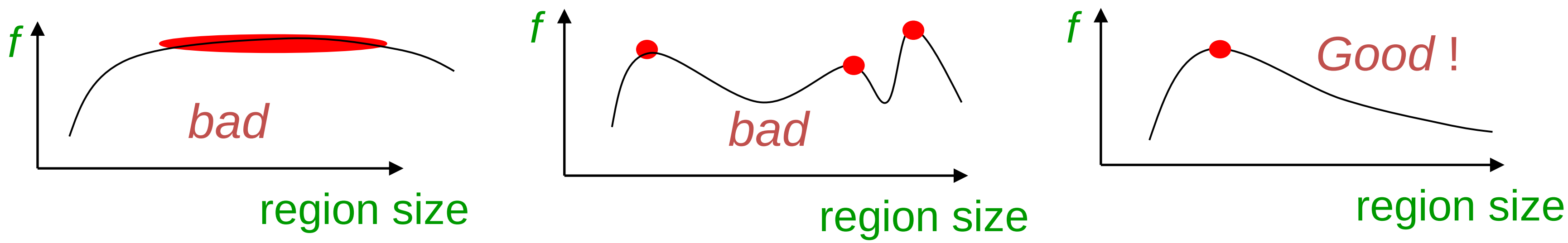
Important: this scale invariant region size is found in each image independently!





Scale Invariant Detection

- A "good" function for scale detection:
  has one stable sharp peak



- For usual images: a good function would be a one which responds to contrast (sharp local intensity change)

Scale Invariant Detection

- Functions for determining scale

$$f = \text{Kernel} * \text{Image}$$

Kernels:

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

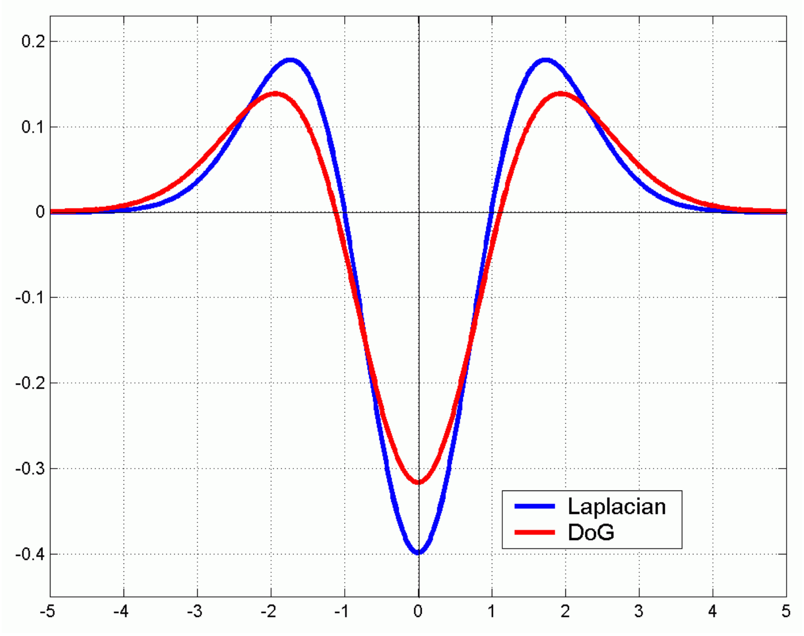
(Laplacian of Gaussians)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)

where Gaussian

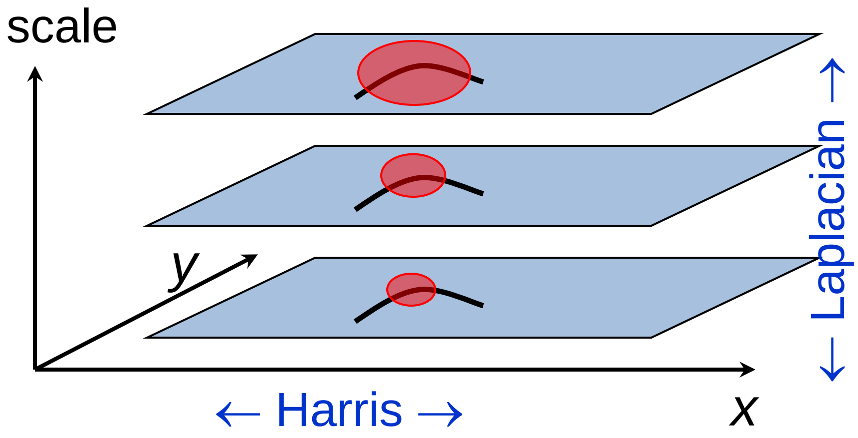
$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



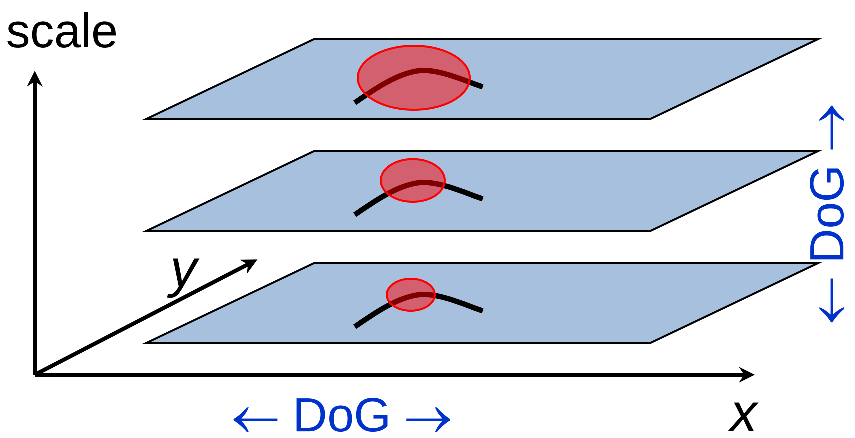
Note: both kernels are invariant to *scale* and *rotation*

Scale Invariant Detectors

- Harris-Laplacian[1]
  Find local maximum of:
  - Harris corner detector in space (image coordinates)
  - Laplacian in scale



- SIFT (Lowe)[2]
  Find local maximum of:
  - Difference of Gaussians in space and scale



[1] K.Mikolajczyk, C.Schmid. “Indexing Based on Scale Invariant Interest Points”. ICCV 2001
[2] D.Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. IJCV 2004

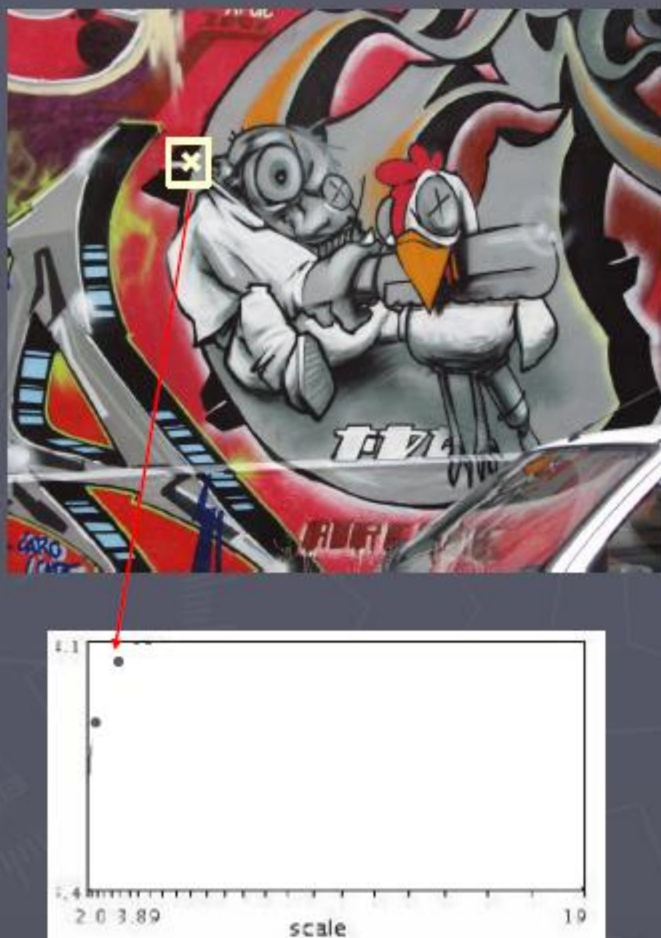
Automatic scale selection

Lindeberg et al., 1996

$f(I_{i_1 \ldots i_m}(x, \sigma))$

Slide from Tinne Tuytelaars

Automatic scale selection



$f(I_{i_1 \ldots i_m}(x,\sigma))$

Slide from Tinne Tuytelaars

Automatic scale selection

$f(I_{i_1 \ldots i_m}(x,\sigma))$

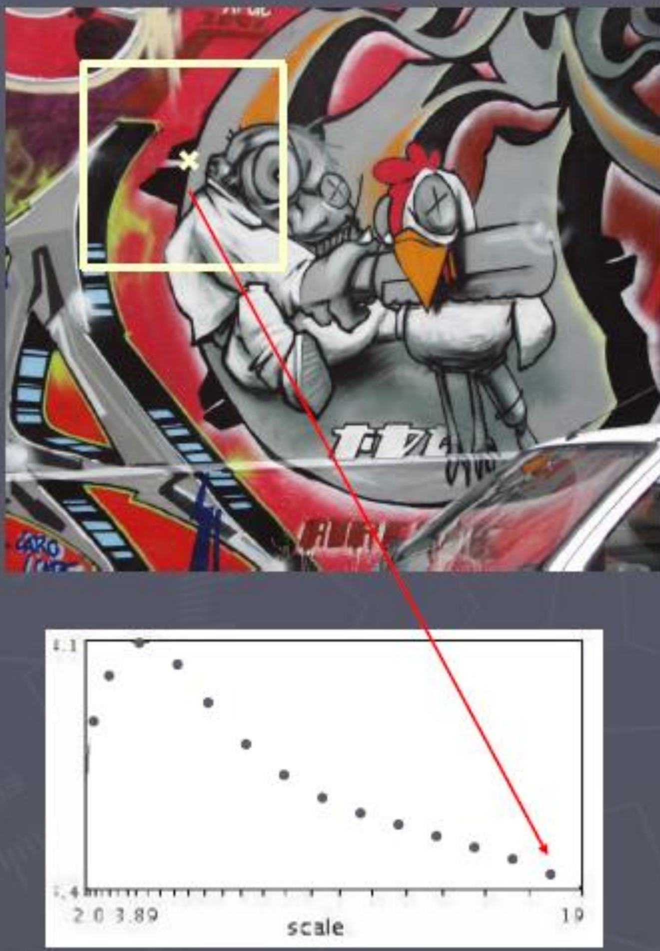
Slide from Tinne Tuytelaars

Automatic scale selection

$f(I_{i_1 \dots i_m}(x, \sigma))$

Slide from Tinne Tuytelaars

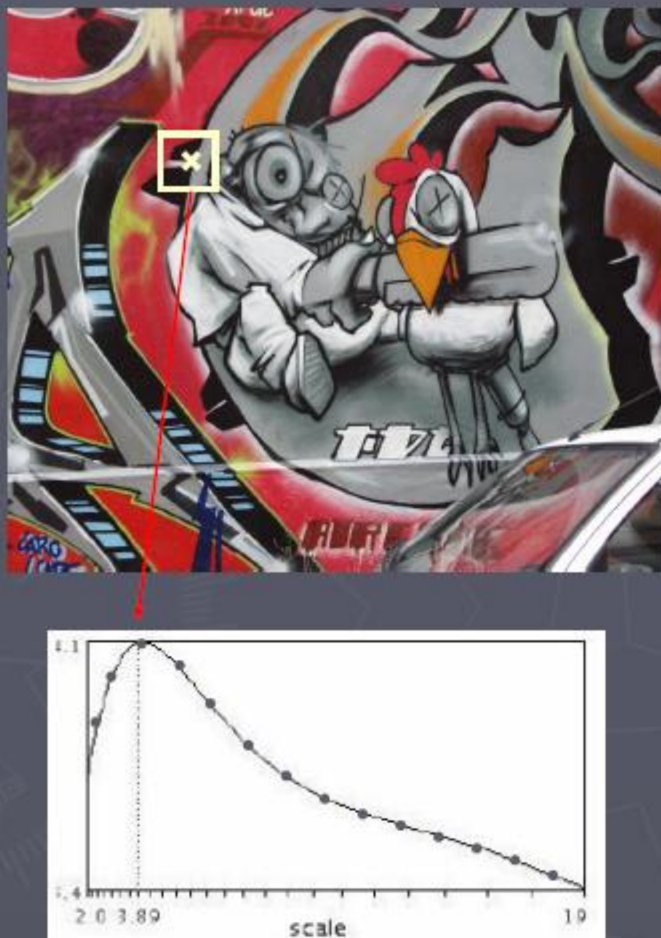
Automatic scale selection



$f(I_{i_1 \ldots i_m}(x,\sigma))$

Slide from Tinne Tuytelaars

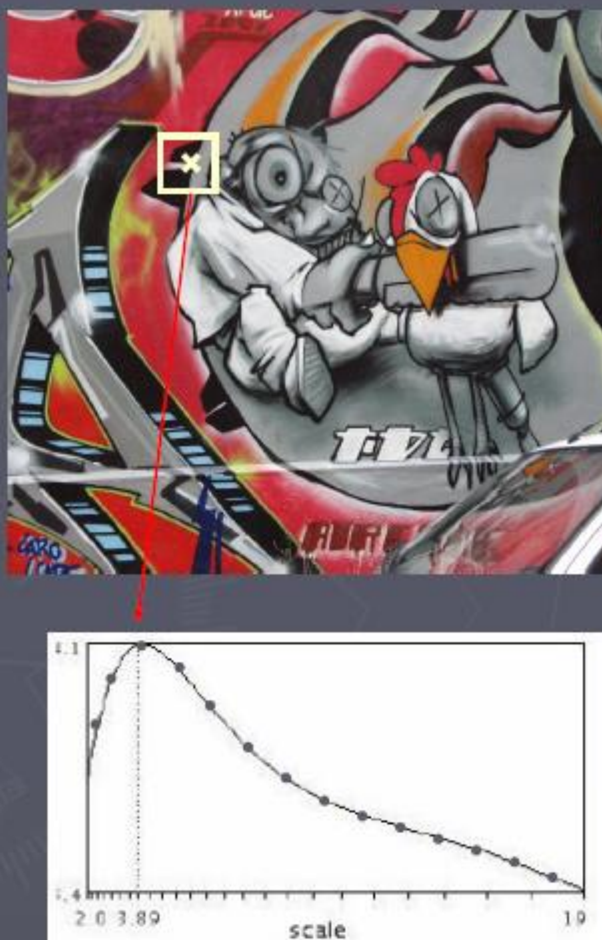
Automatic scale selection



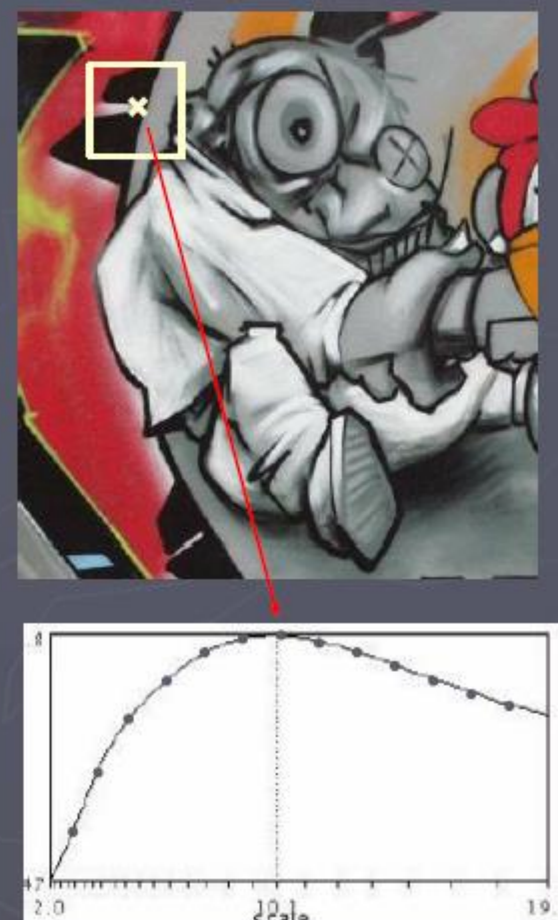
$$f(I_{i_1\ldots i_m}(x,\sigma))$$

Slide from Tinne Tuytelaars

Automatic scale selection



$f(I_{i_1 \ldots i_m}(x,\sigma))$

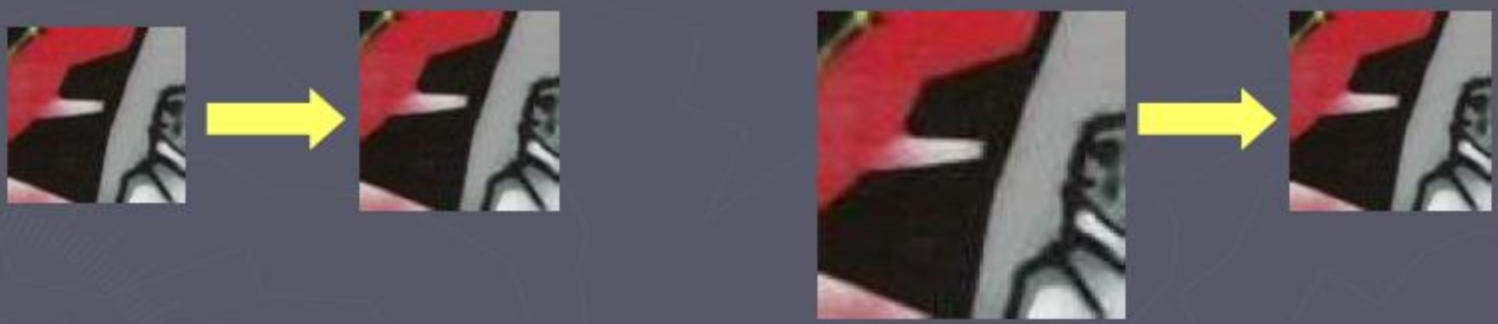


$f(I_{i_1 \ldots i_m}(x',\sigma'))$

Slide from Tinne Tuytelaars

Automatic scale selection

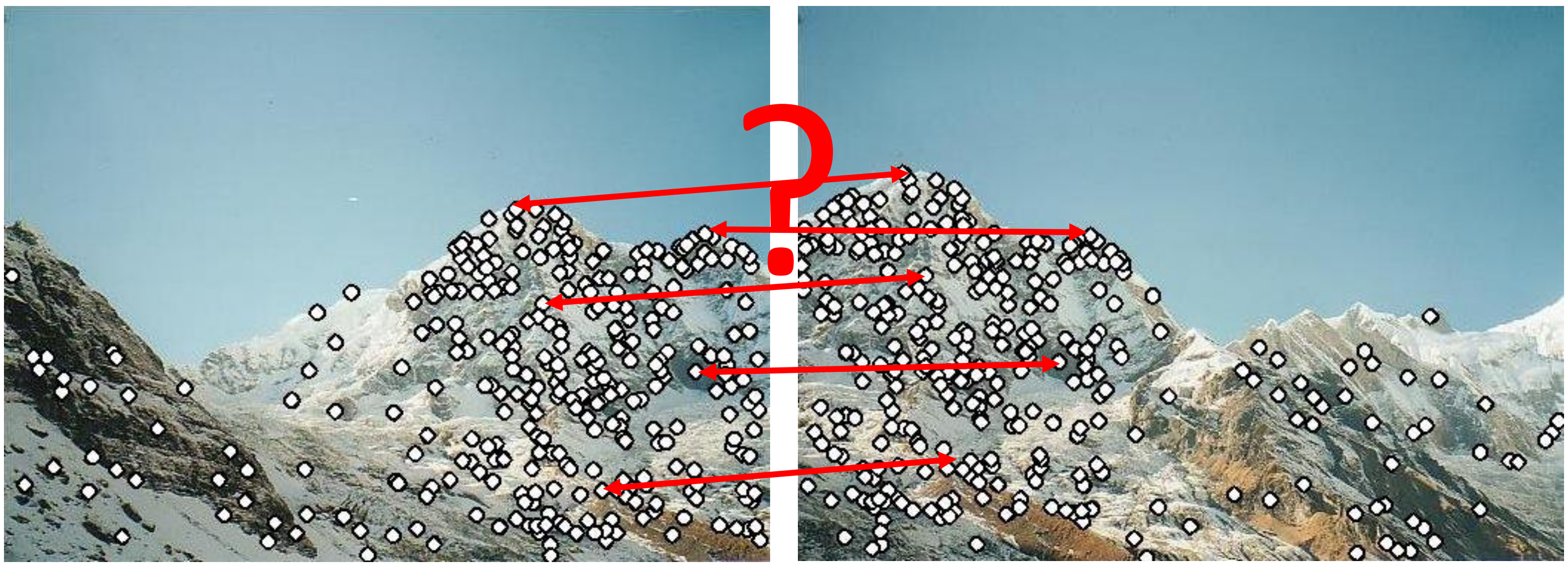
Normalize: rescale to fixed size



Slide from Tinne Tuytelaars

Feature descriptors

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

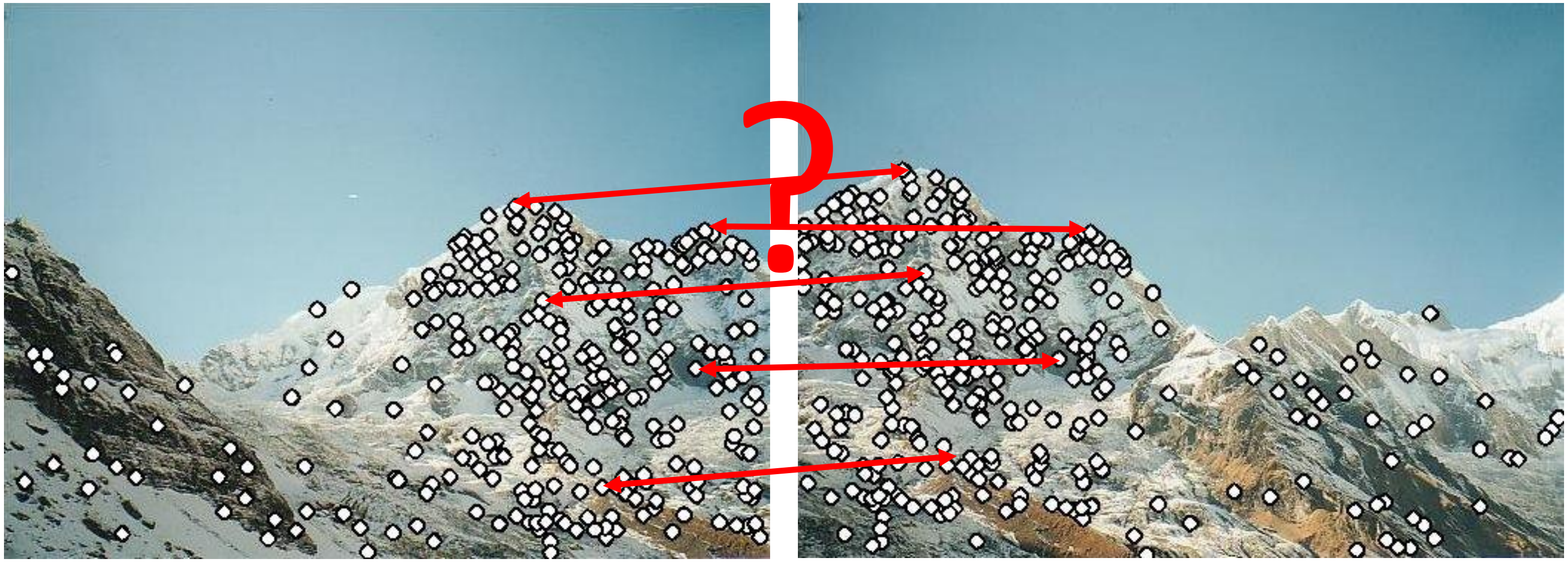


Courtesy: S. Seitz and R. Szeliski

Feature descriptors

We now know how to detect good points

Next question: **How to match them?**



Point descriptor should be:

1. Invariant
2. Distinctive

Courtesy: S. Seitz and R. Szeliski

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?

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?
    - Translation, 2D rotation, scale

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?
    - Translation, 2D rotation, scale

- Descritpors can usually also handle
  - Limited 3D rotations (**SIFT** works up to about 60 degrees)
  - Limited affine transformations (2D rotation, scale, shear)
  - Limited illumination/contrast changes

How to achieve invariance

Need both of the following:

1. Make sure your *detector* is invariant
   – SIFT is invariant to translation, rotation and scale
2. Design an invariant feature *descriptor*
   – A descriptor captures the information in a region around the detected feature point

Scale Invariant Feature Transform

- Algorithm outline:
  - Detect interest points
  - For each interest point
    - Determine dominant orientation
    - Build histograms of gradient directions
    - Output feature *descriptor*

Scale Invariant Feature Transform

Basic idea:

- Take 16x16 square window around detected feature
- Compute gradient for each pixel
- Throw out weak gradient magnitudes
- Create histogram of surviving gradient orientations

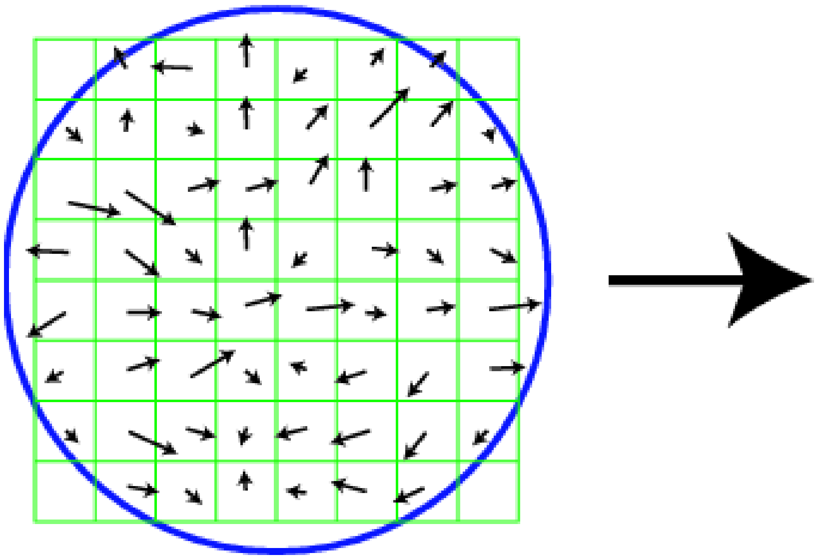
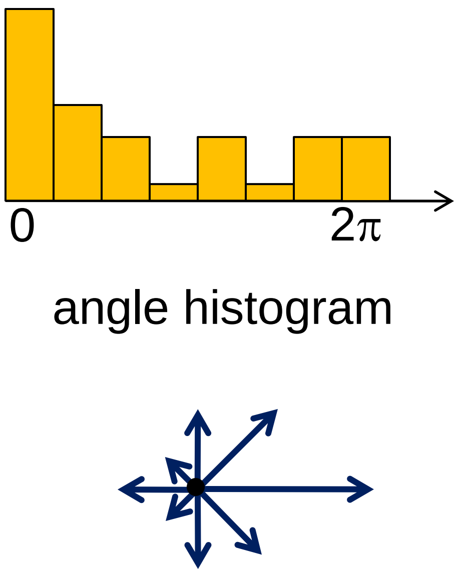


Image gradients



angle histogram

Adapted from slide by David Lowe

SIFT keypoint descriptor

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

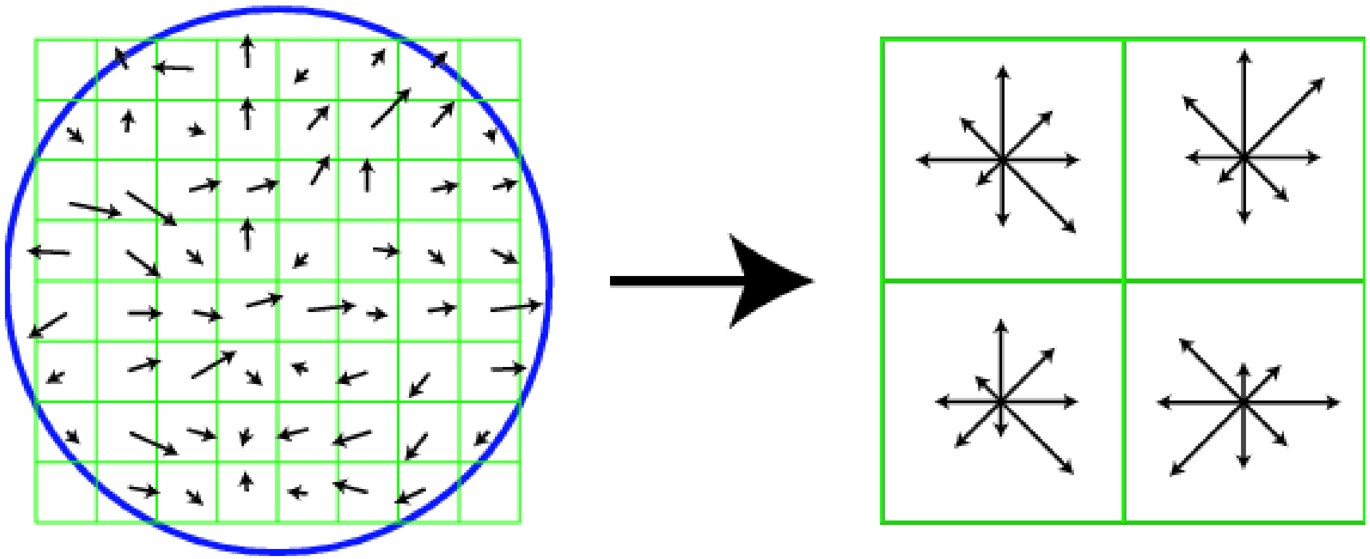


Image gradients

Keypoint descriptor

Adapted from slide by David Lowe

Properties of SIFT

Extraordinarily robust matching technique

- Can handle changes in viewpoint
  - Up to about 60 degree out of plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night (below)
- Fast and efficient—can run in real time
- Lots of code available
  - http://www.vlfeat.org
  - http://www.cs.unc.edu/~ccwu/siftgpu/

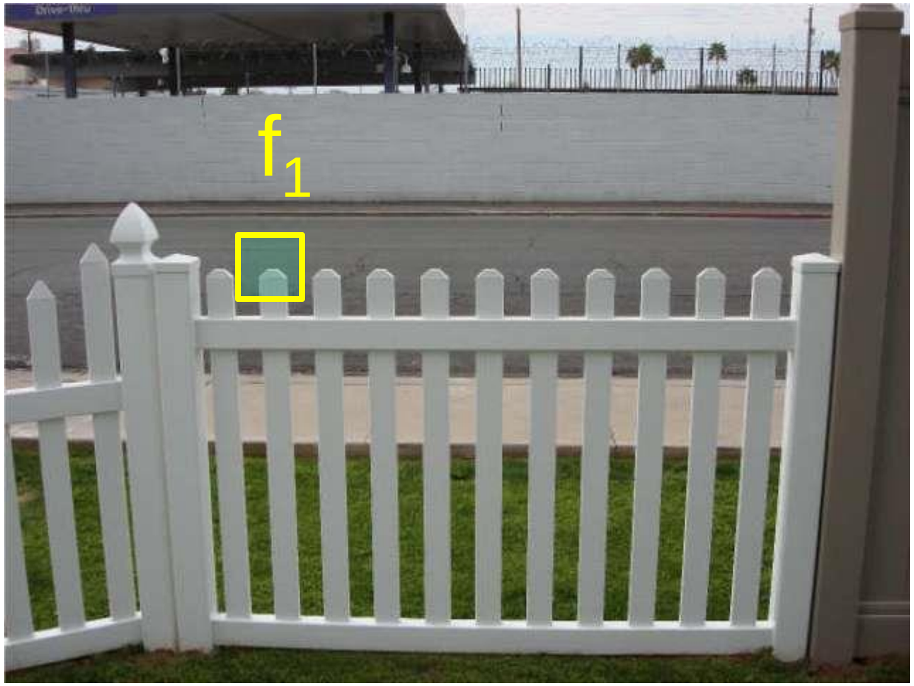
Feature matching

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

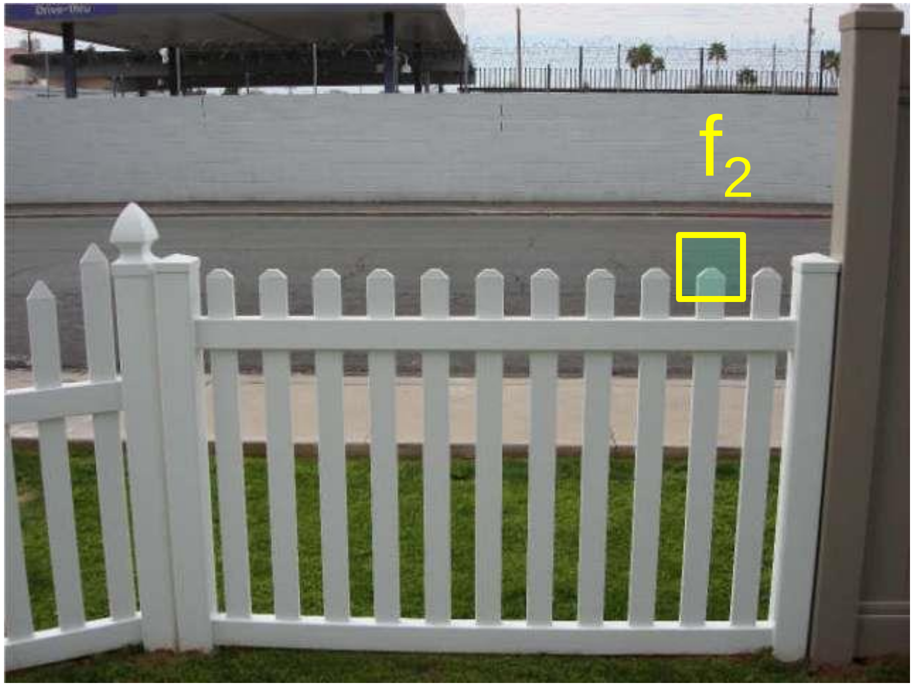
1. Define distance function that compares two descriptors
2. Test all the features in I_2, find the one with min distance

Feature distance

- How to define the difference between two features f1, f2?
  - Simple approach is SSD(f1, f2)
    - sum of square differences between entries of the two descriptors
    - can give good scores to very ambiguous (bad) matches



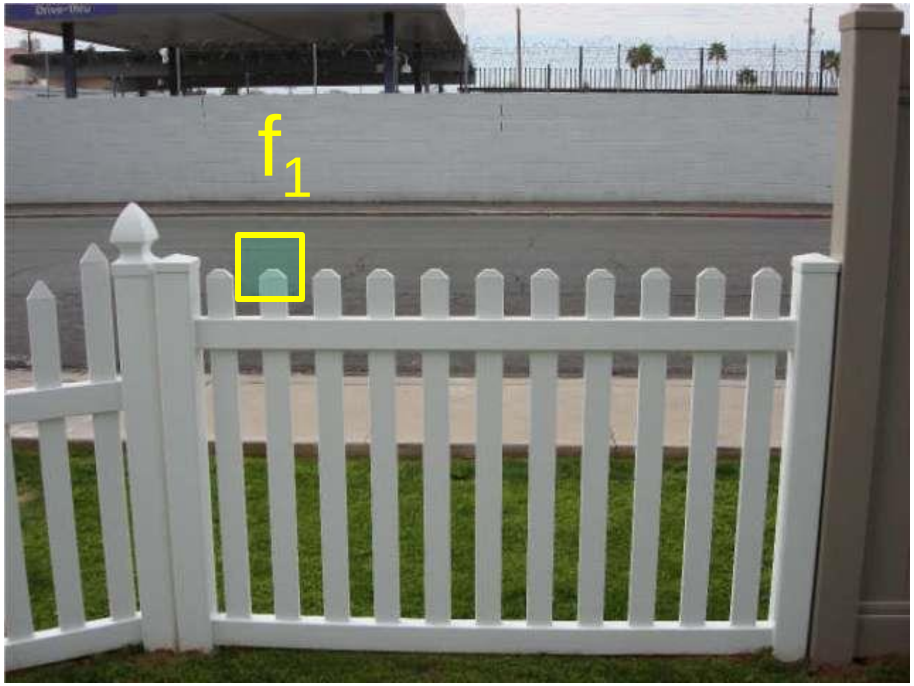
I_1



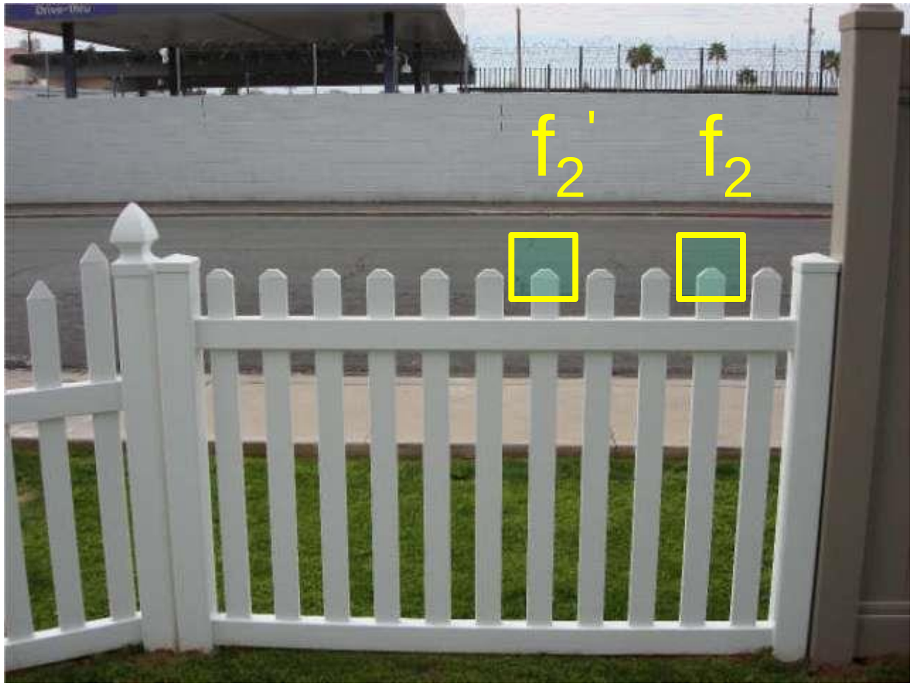
I_2

Feature distance

- How to define the difference between two features f1, f2?
  - Better approach: ratio distance = SSD(f1, f2) / SSD(f1, f2')
    - f2 is best SSD match to f1 in I2
    - f2' is 2nd best SSD match to f1 in I2
    - gives small values for ambiguous matches



I_1



I_2

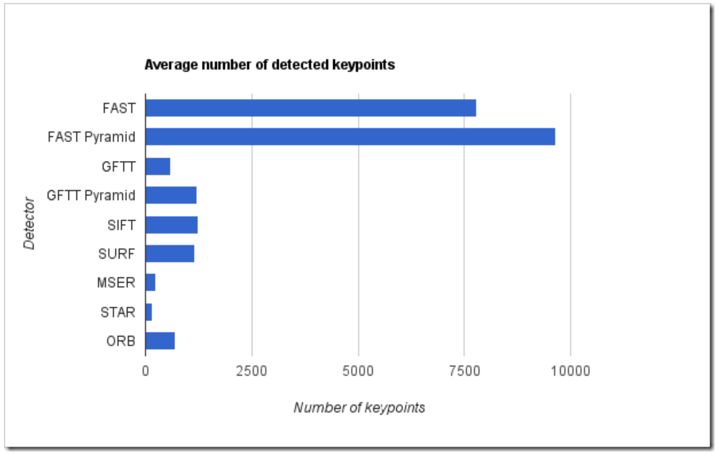
Lots of applications

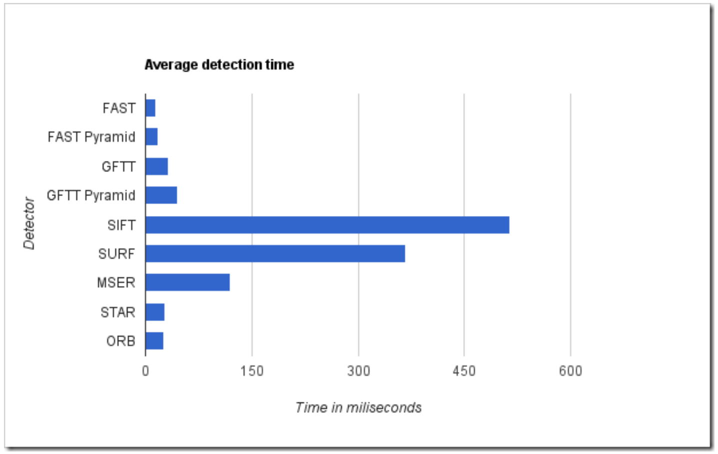
Features are used for:

- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation
- … other

More Features

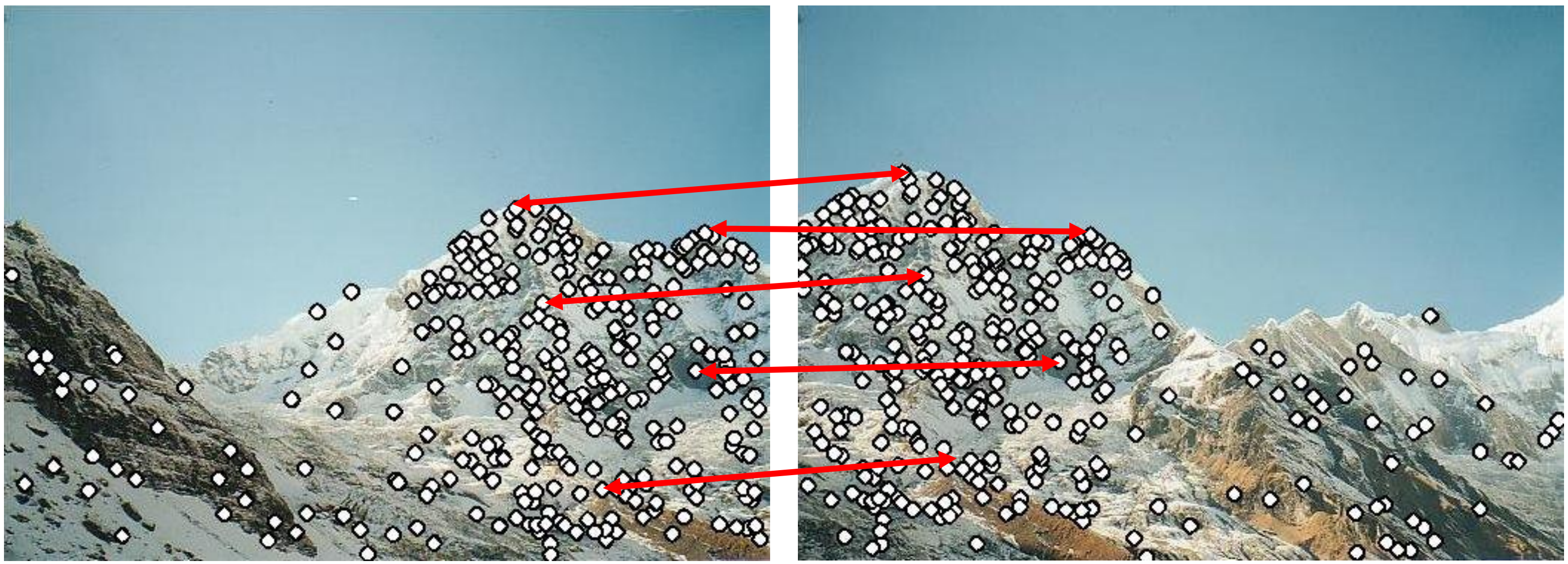
- FAST
- GFTT
- SURF
- ORB
- STAR
- MSER
- KAZE
- A-KAZE



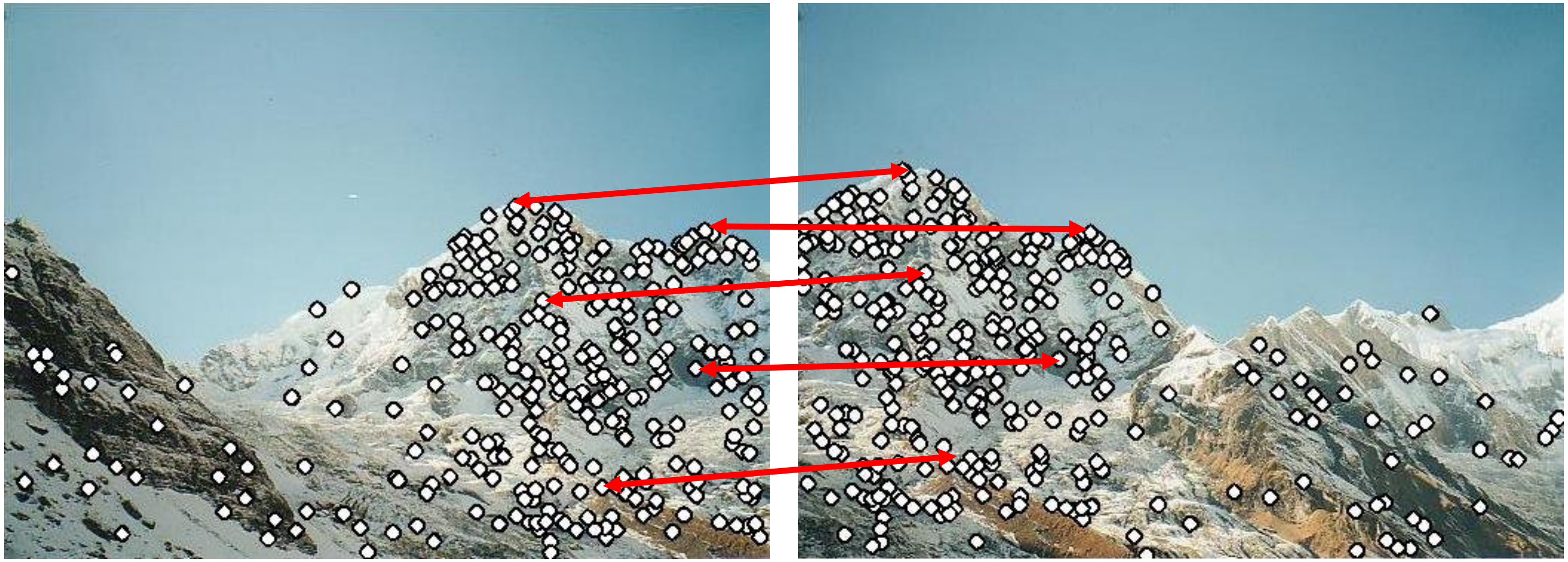


http://computer-vision-talks.com/articles/2011-07-13-comparison-of-the-opencv-feature-detection-algorithms/

Are descriptors unique?



Are descriptors unique?



No, they can be matched to wrong features, generating outliers.

Dealing with outliers

- Fit a geometric transformation to a small subset of all possible matches.
- Possible strategies:
  - RANSAC
  - Incremental alignment
  - Hough transform

Strategy: RANSAC

- RANSAC loop:

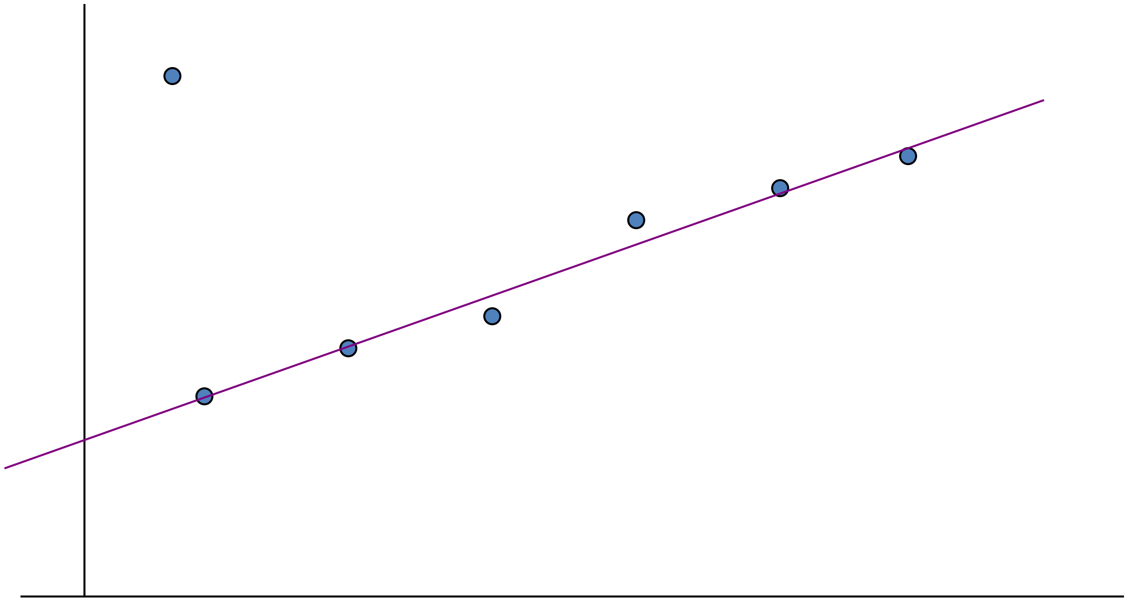
1. Randomly select a *seed group* of matches
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers

- Keep the transformation with the largest number of inliers

M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.

Simple Example

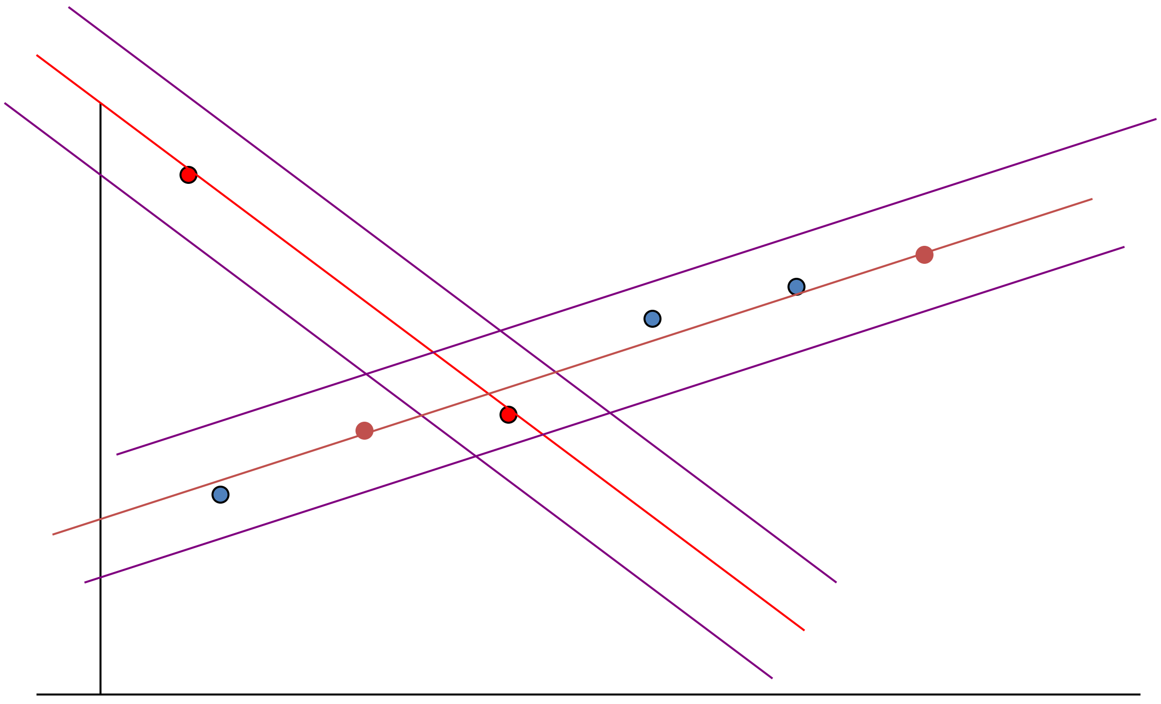
- Fitting a straight line



Main Idea

- Select 2 points at random
- Fit a line
- “Support” = number of inliers
- Line with most inliers wins

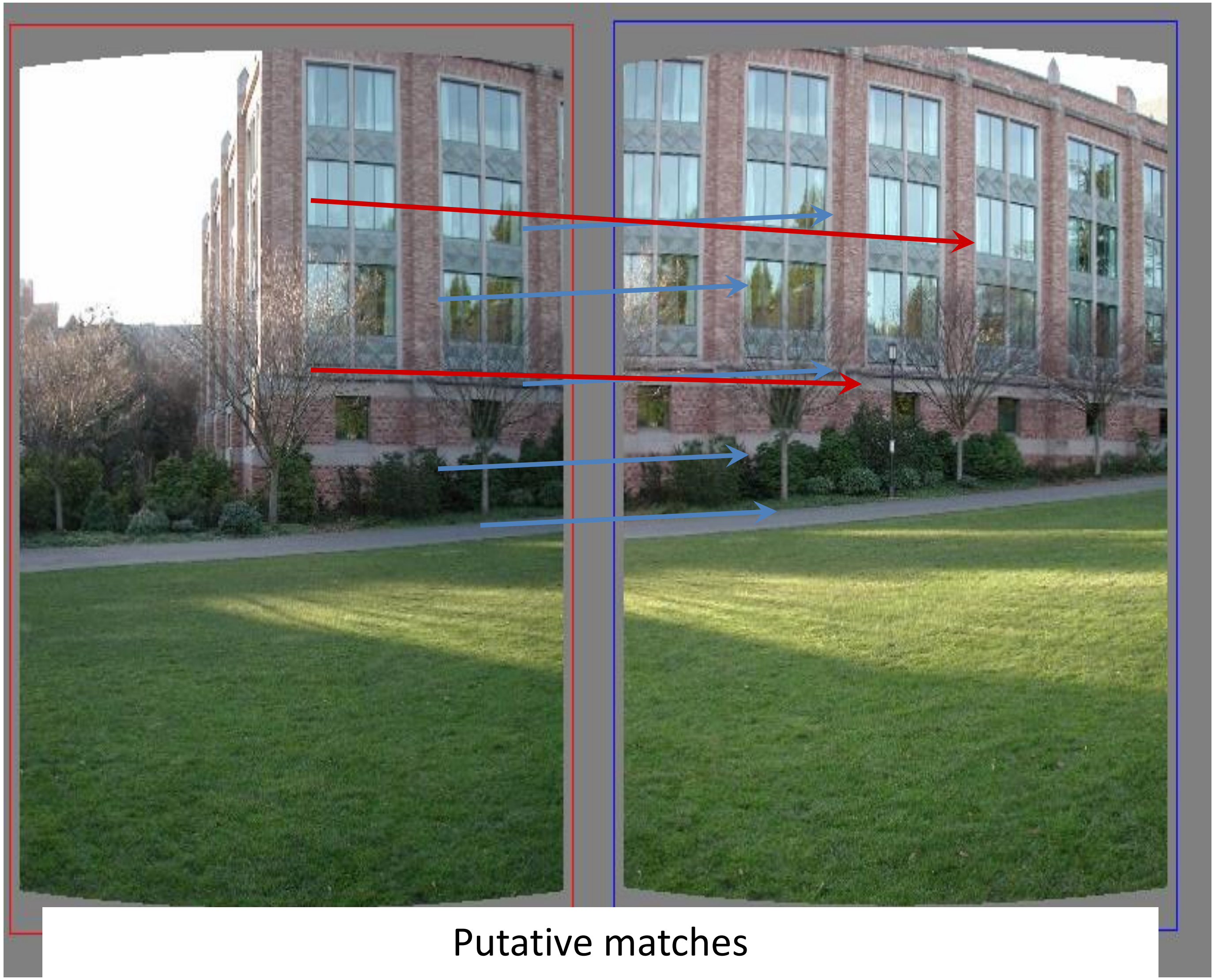
Why will this work ?



Best Line has most support

- More support -> better fit

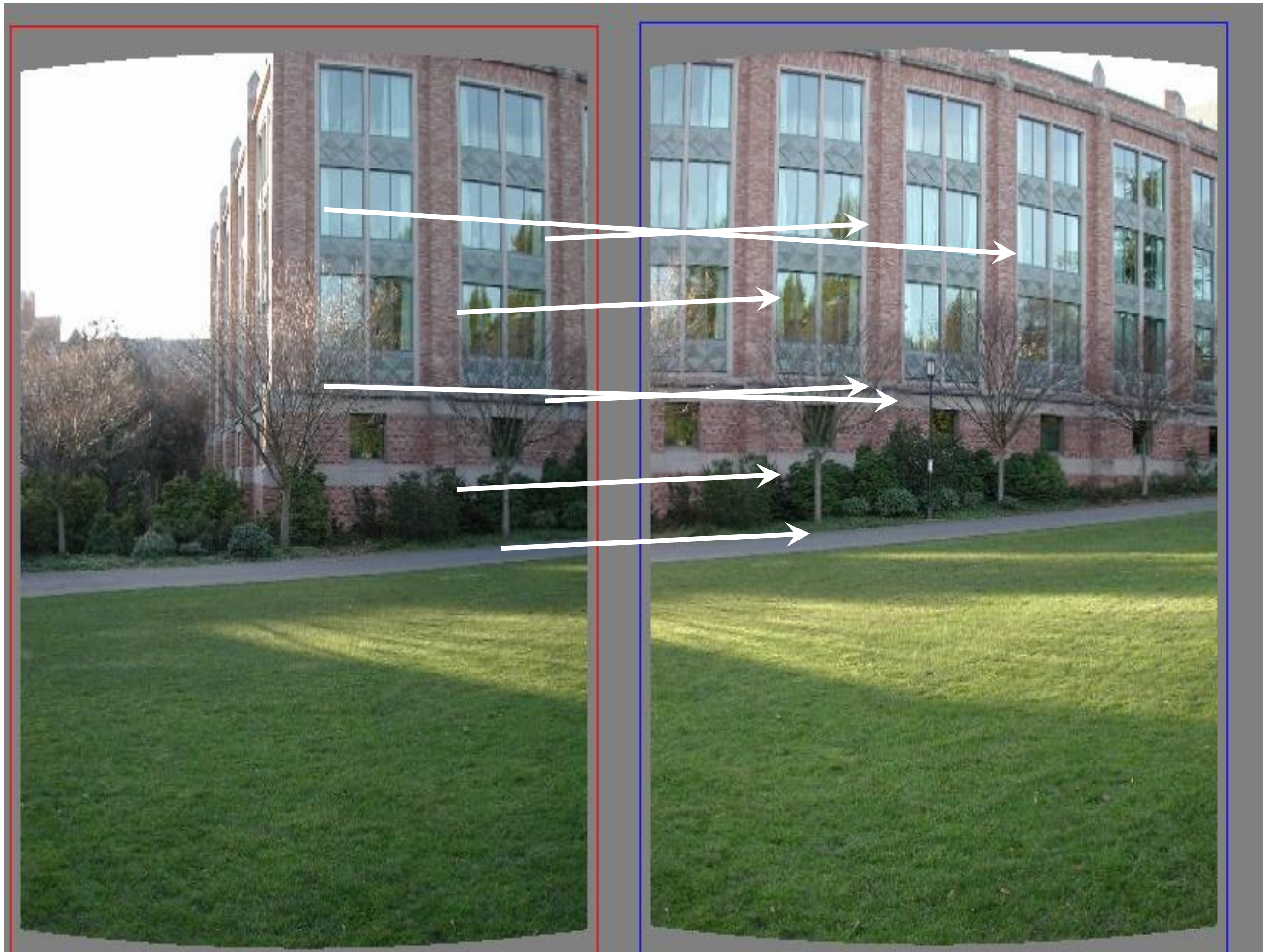
RANSAC example: Translation



Putative matches

Slide: A. Efros

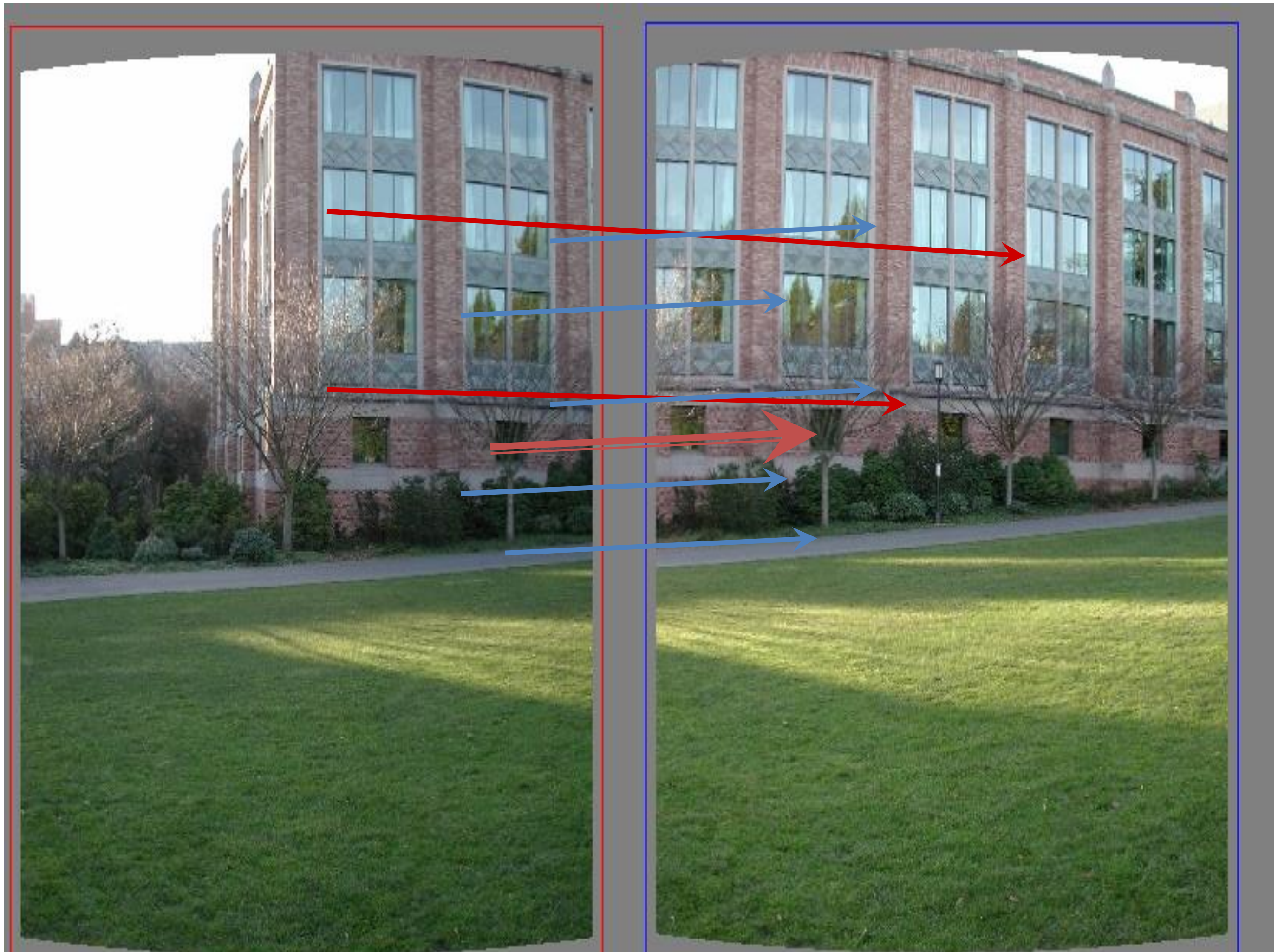
RANSAC example: Translation



Select *one* match, count *inliers*

Slide: A. Efros

RANSAC example: Translation



Find “average” translation vector

Slide: A. Efros

RANSAC: General Case

- Objective:
  - Robust fit of a model to data S
- Algorithm
  - Randomly select s points
  - Instantiate a model
  - Get consensus set S_i
  - If $|S_i|>T$, terminate and return model
  - Repeat for N trials, return model with max $|S_i|$

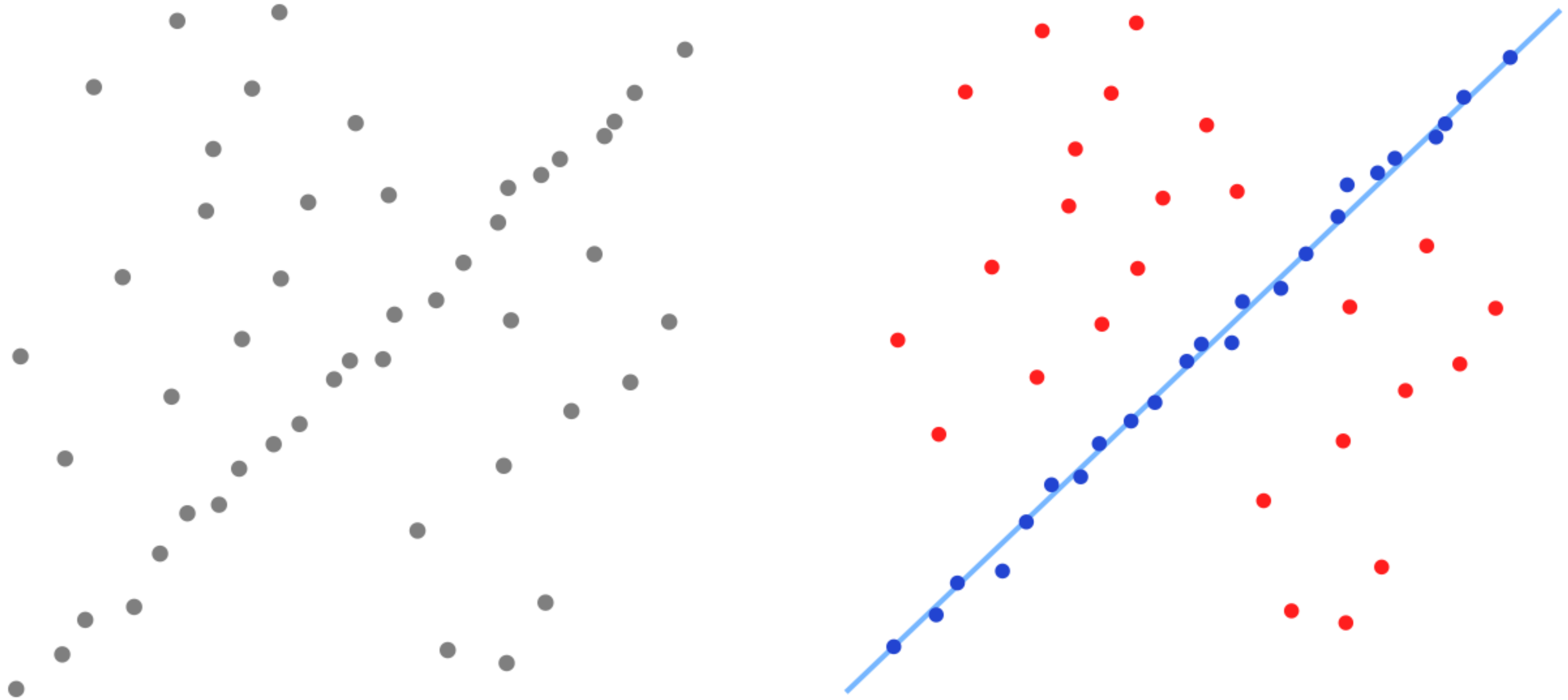
How many samples ?

- We want: at least one sample with all inliers
  - Can't guarantee: probability *p*
  - e.g., $p = 0.99$
- Let *e = % of outliers*, and *s = # of required data points to fit model*
- With probability *p*, we want at least one trial with all inliers:

  1 - P(N trials with at least one outlier) ≥ p
- Hence, the required number of trials is ?

$$N \geq \log(1-p)/\log(1-(1-e)^s)$$

RANSAC: Line Fitting



Adaptive RANSAC

- $N = \infty$, sample_count$= 0$.
- While $N >$ sample_count Repeat
  - Choose a sample and count the number of inliers.
  - Set $\epsilon = 1 -$ (number of inliers)/(total number of points)
  - Set N from ϵ and (3.18) with $p = 0.99$.
  - Increment the sample_count by 1.
- Terminate.

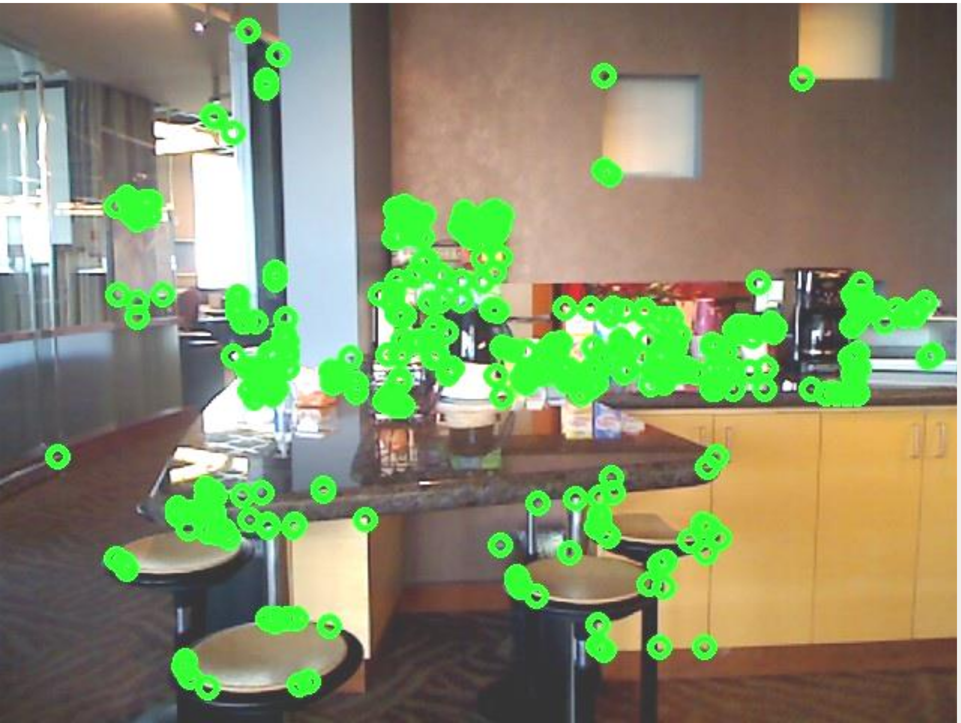
Algorithm 3.5. *Adaptive algorithm for determining the number of RANSAC samples.*

RANSAC pros and cons

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- Cons
  - Lots of parameters to tune
  - Can’t always get a good initialization of the model based on the minimum number of samples
  - Sometimes too many iterations are required
  - Can fail for extremely low inlier ratios

Visual Odometry

- Compute the motion between consecutive camera frames from visual feature correspondences.
- Visual features from RGB image have a 3D counterpart from depth image.
- Three 3D-3D correspondences constrain the motion.



Visual Odometry Failure Cases

- Low light, lack of visual texture or features

Visual Odometry Failure Cases

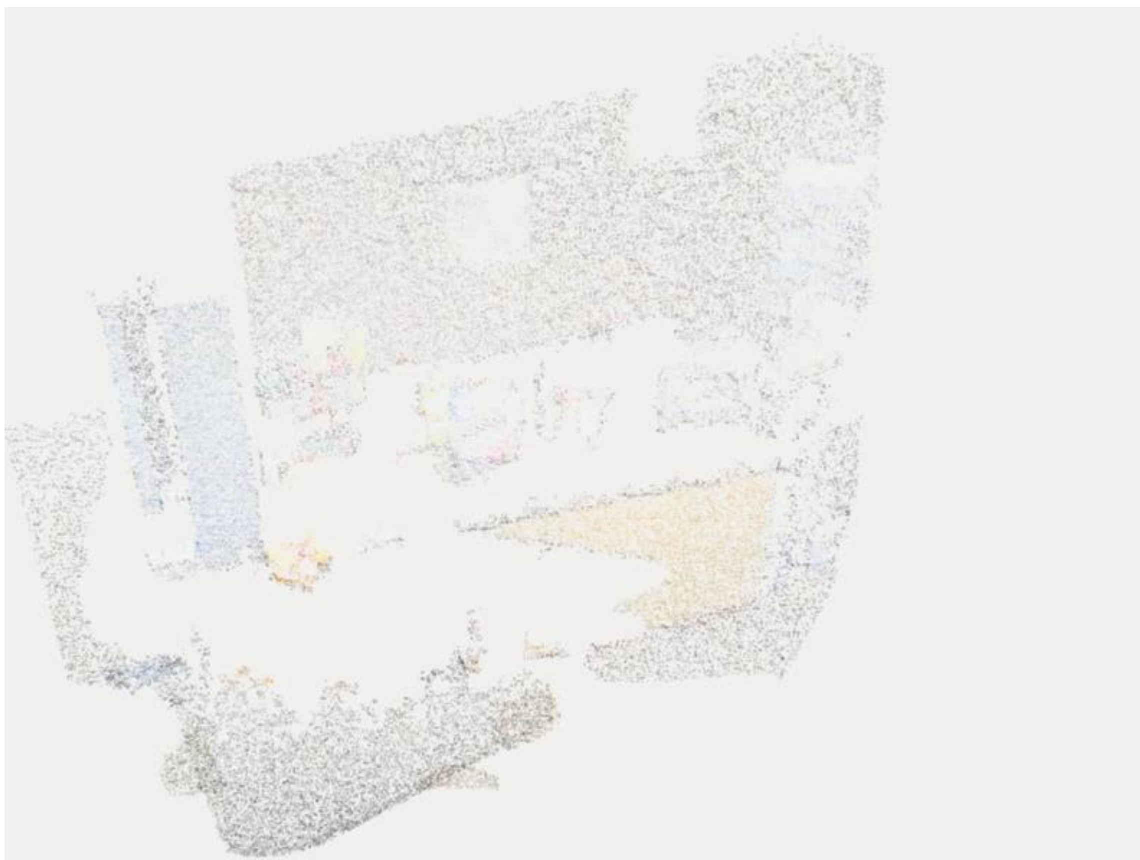
- Low light, lack of visual texture or features
- Poor distribution of features across image

Visual Odometry Failure Cases

- Low light, lack of visual texture or features
- Poor distribution of features across image
- RGB-D camera still provides shape information

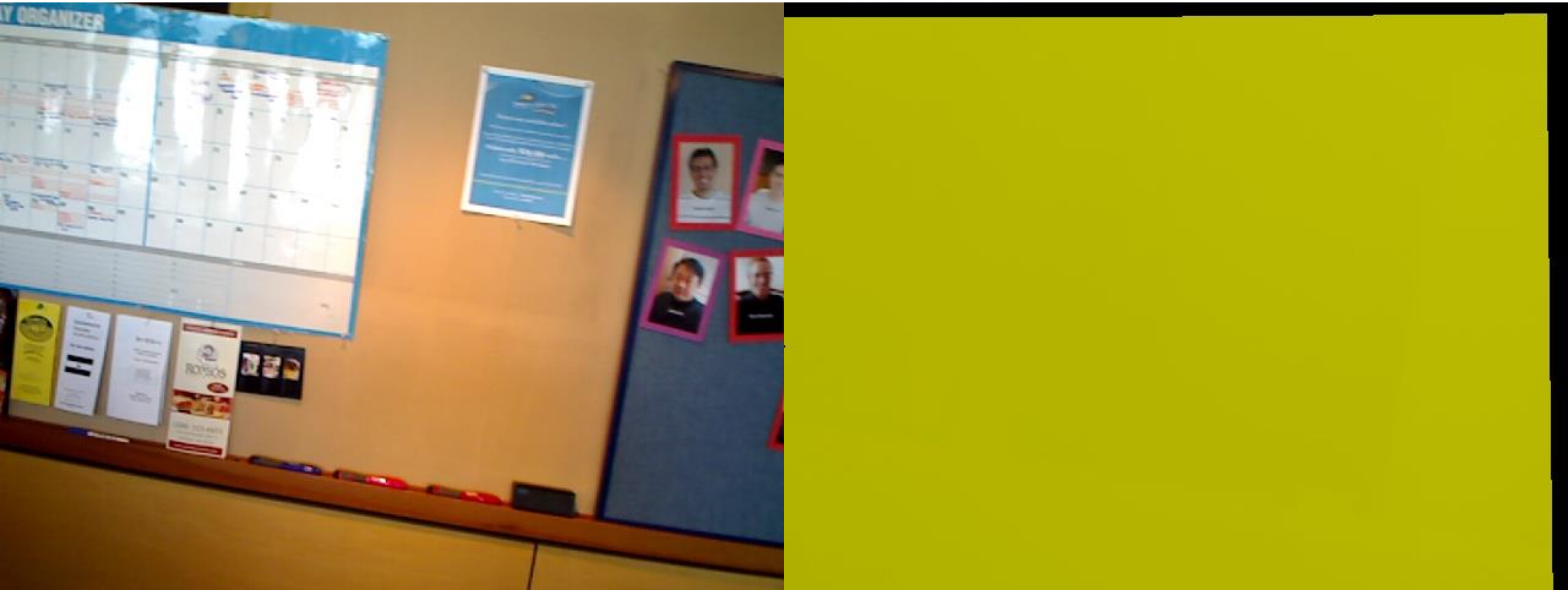
ICP (Iterative Closest Point)

- Iteratively align frames based on shape
- Needs a good initial estimate of the pose



ICP Failure Cases

- Not enough distinctive shape
- Don’t have a close enough initial “guess”
- Here the shape is basically a simple plane…



Optimal Transformation

- Jointly minimize feature reprojection and ICP:

$$\mathbf{t}^* = \operatorname*{argmin}_{\mathbf{t}} \left[\left(\frac{1}{|A_f|} \sum_{i \in A_f} \left| Proj(\mathbf{t}(f_s^i)) - Proj(f_t^i) \right|^2 \right) + \beta \left(\frac{1}{|A_d|} \sum_{j \in A_d} w_j \left| (\mathbf{t}(p_s^j) - p_t^j) \cdot n_t^j \right|^2 \right) \right]$$

RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments. Henry et al. ISER 2010
RGB-D Mapping: Using Kinect-style Depth Cameras for Dense 3D Modeling of Indoor Environments. Henry et al. IJRR 2012

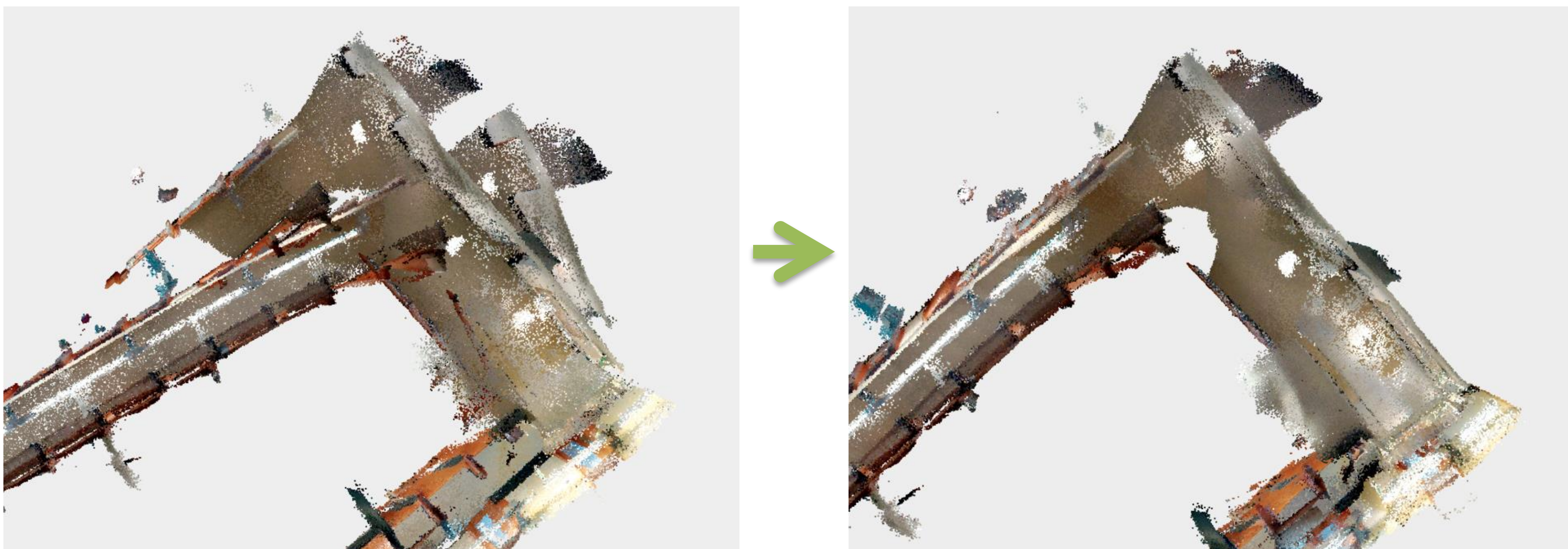
Outline

- Motivation
- RGB-D Mapping:

1. Frame-to-frame motion (visual odometry)
2. Revisiting places (loop closure detection)
3. Map representation (Surfels)

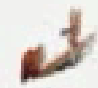
Loop Closure

- Sequential alignments accumulate error
- Revisiting a previous location results in an inconsistent map



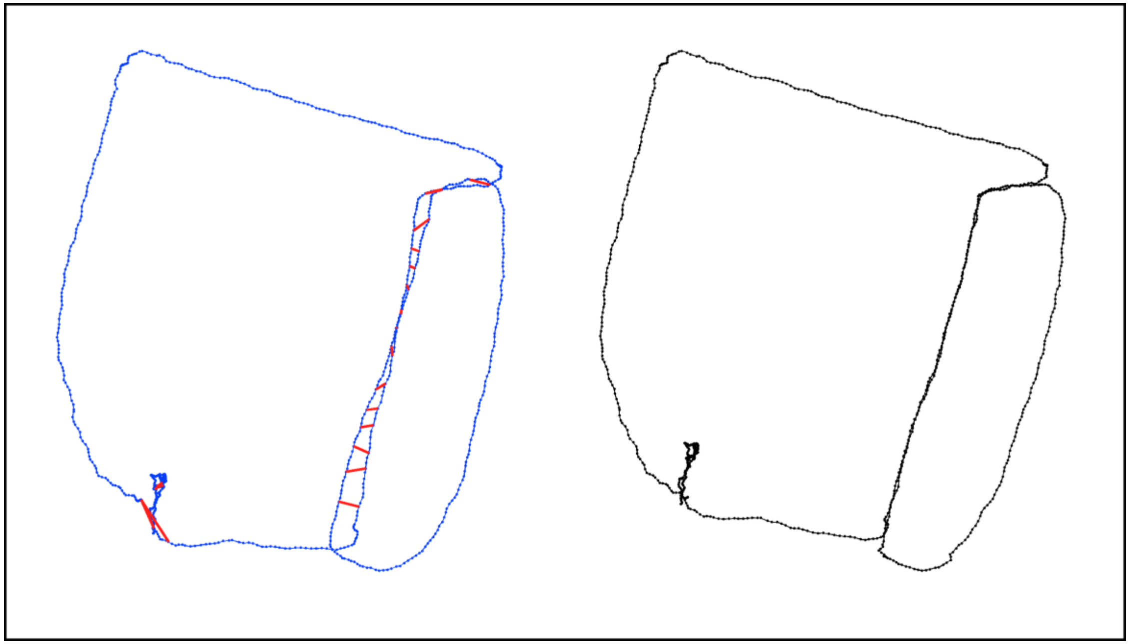
Loop Closure Detection

- Detect by running RANSAC against previous frames
- Pre-filter options (for efficiency):
  - Only a subset of frames (*keyframes*)
  - Only keyframes with similar estimated 3D pose
  - Place recognition using vocabulary tree
    - *Scalable recognition with a vocabulary tree*, David Nister and Henrik Stewenius, 2006
- Post-filter (avoid false positives)
  - Estimate maximum expected drift and reject detections changing pose too greatly



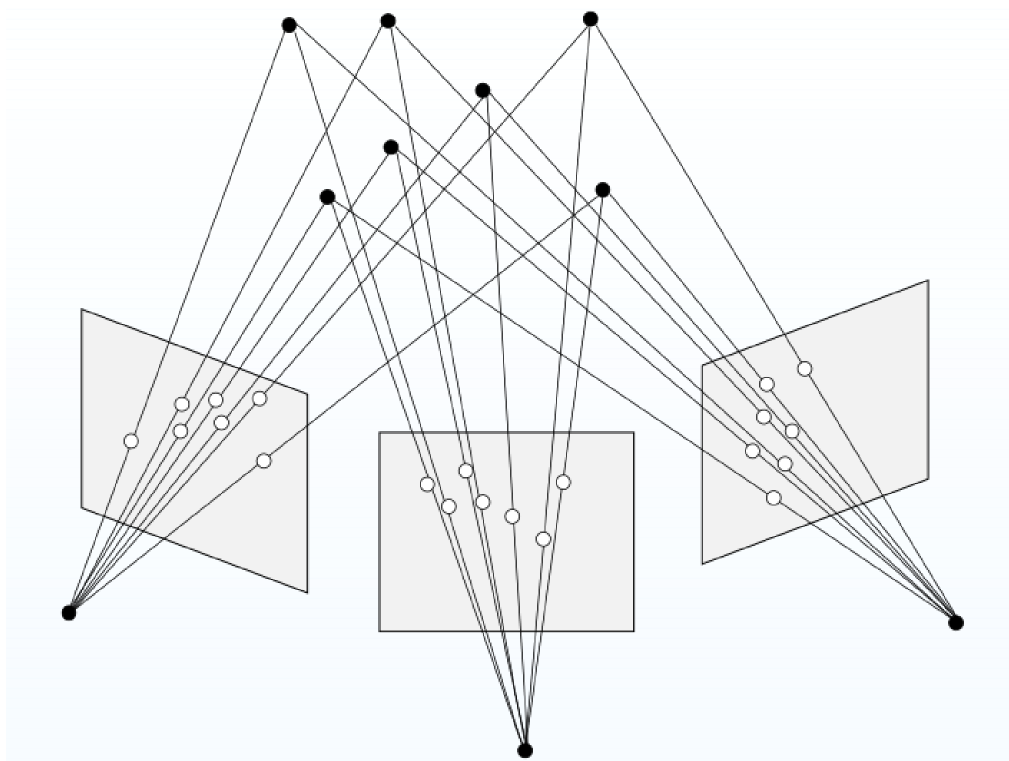
Loop Closure Correction (TORO)

- TORO [Grisetti 2007, 2009]:
  - Constraints between camera locations in *pose graph*
  - Maximum likelihood global camera poses



[Image: Manolis Lourakis]

Loop Closure Correction: Bundle Adjustment



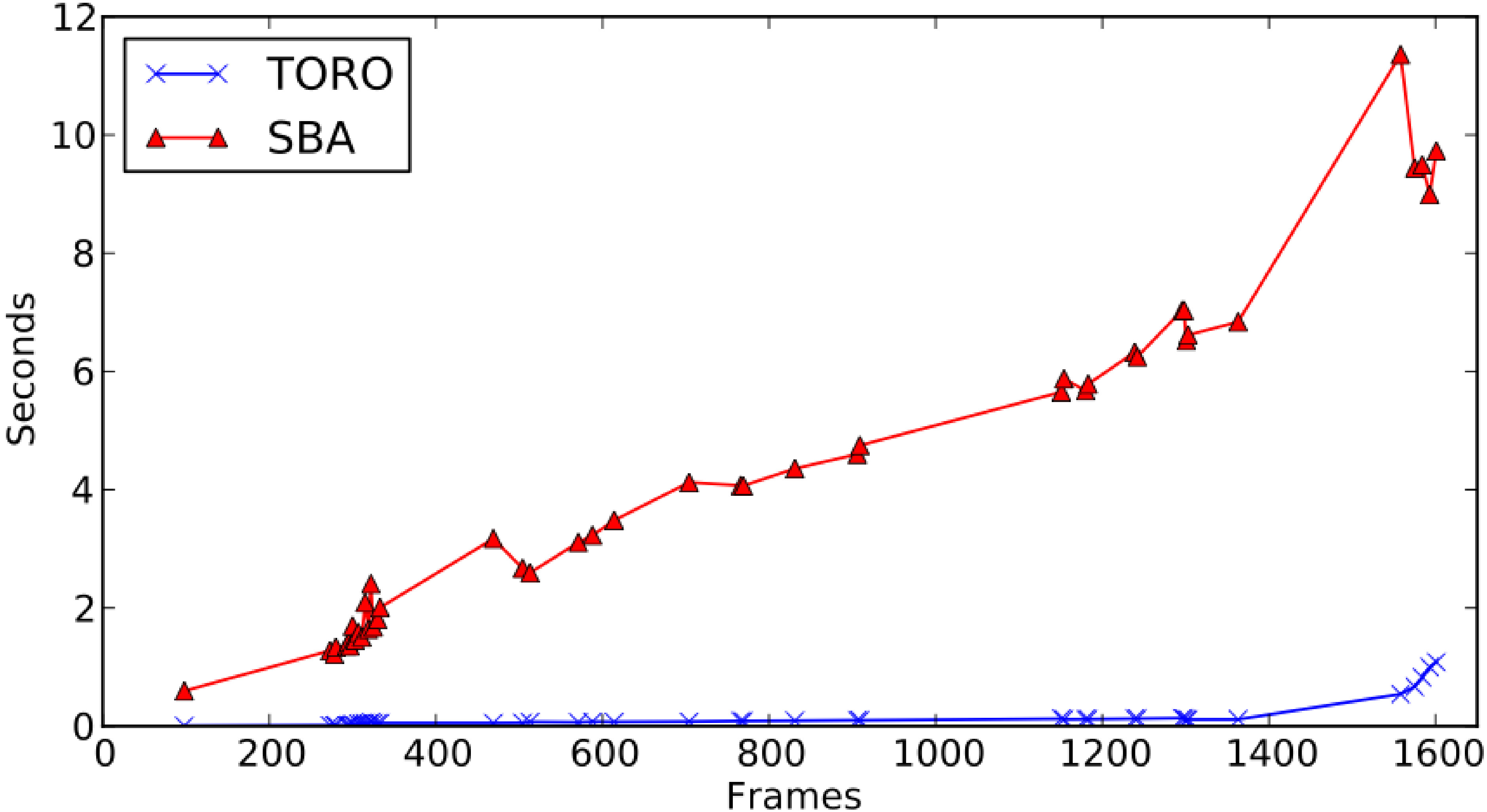
$$\sum_{c_i \in C} \sum_{p_j \in P} v_{ij} |Proj(c_i, p_j) - (\bar{u}, \bar{v}, \bar{d})|^2$$

A Second Comparison

TORO

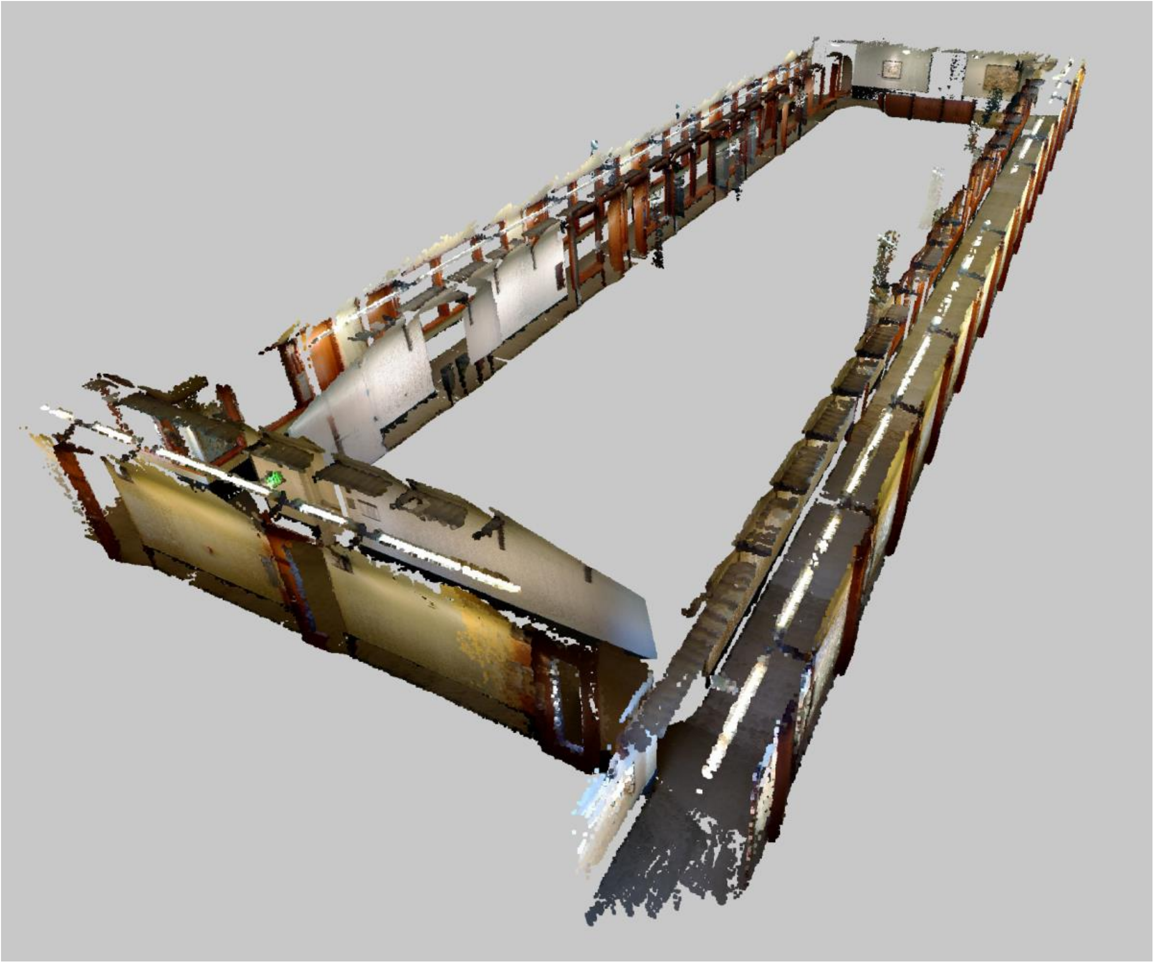
SBA

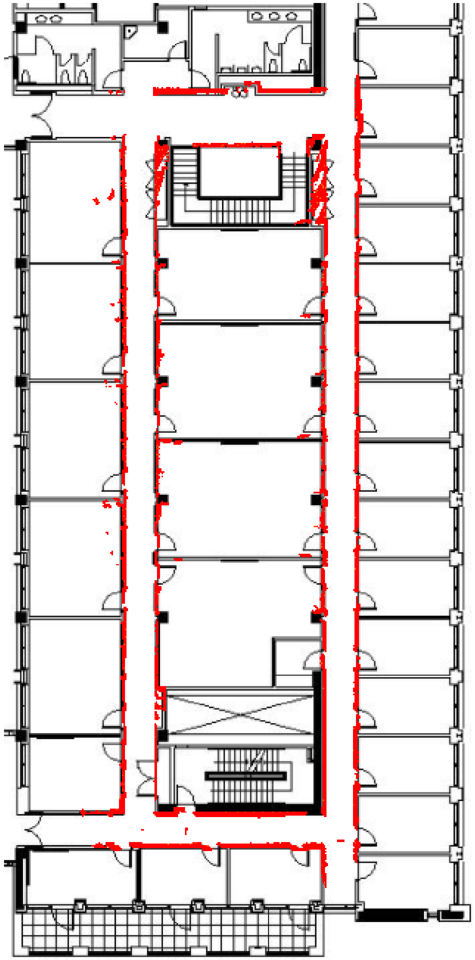
Timing



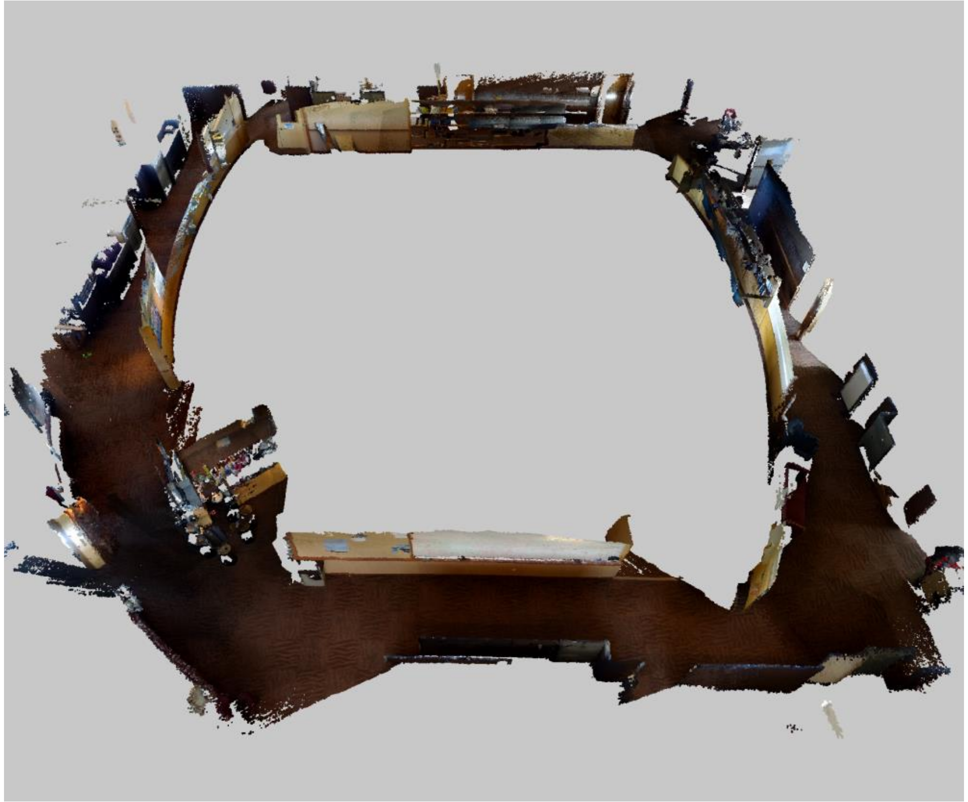
TORO
SBA
Seconds
Frames
0
2
4
6
8
10
12
0
200
400
600
800
1000
1200
1400
1600

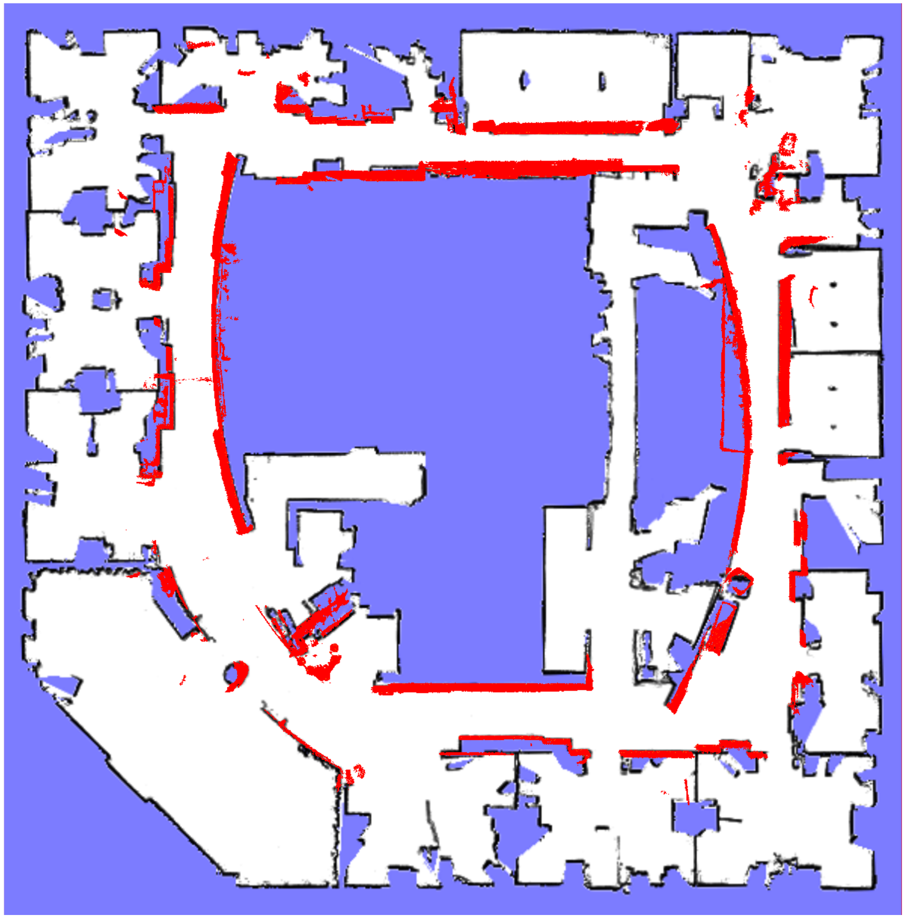
Overlay 1





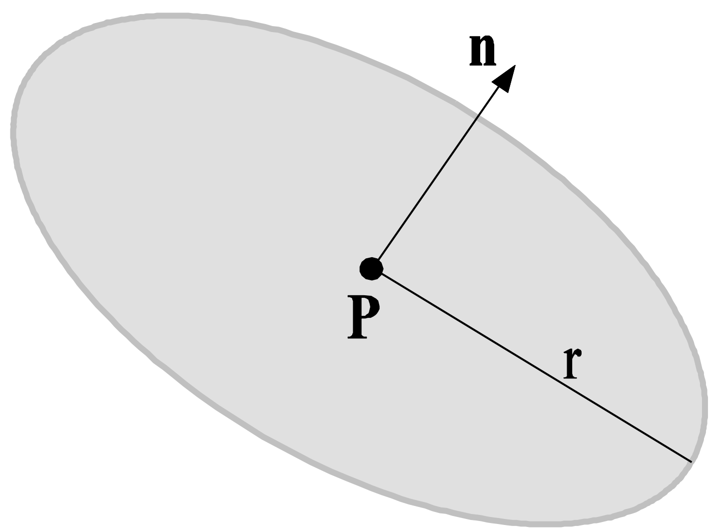
Overlay 2





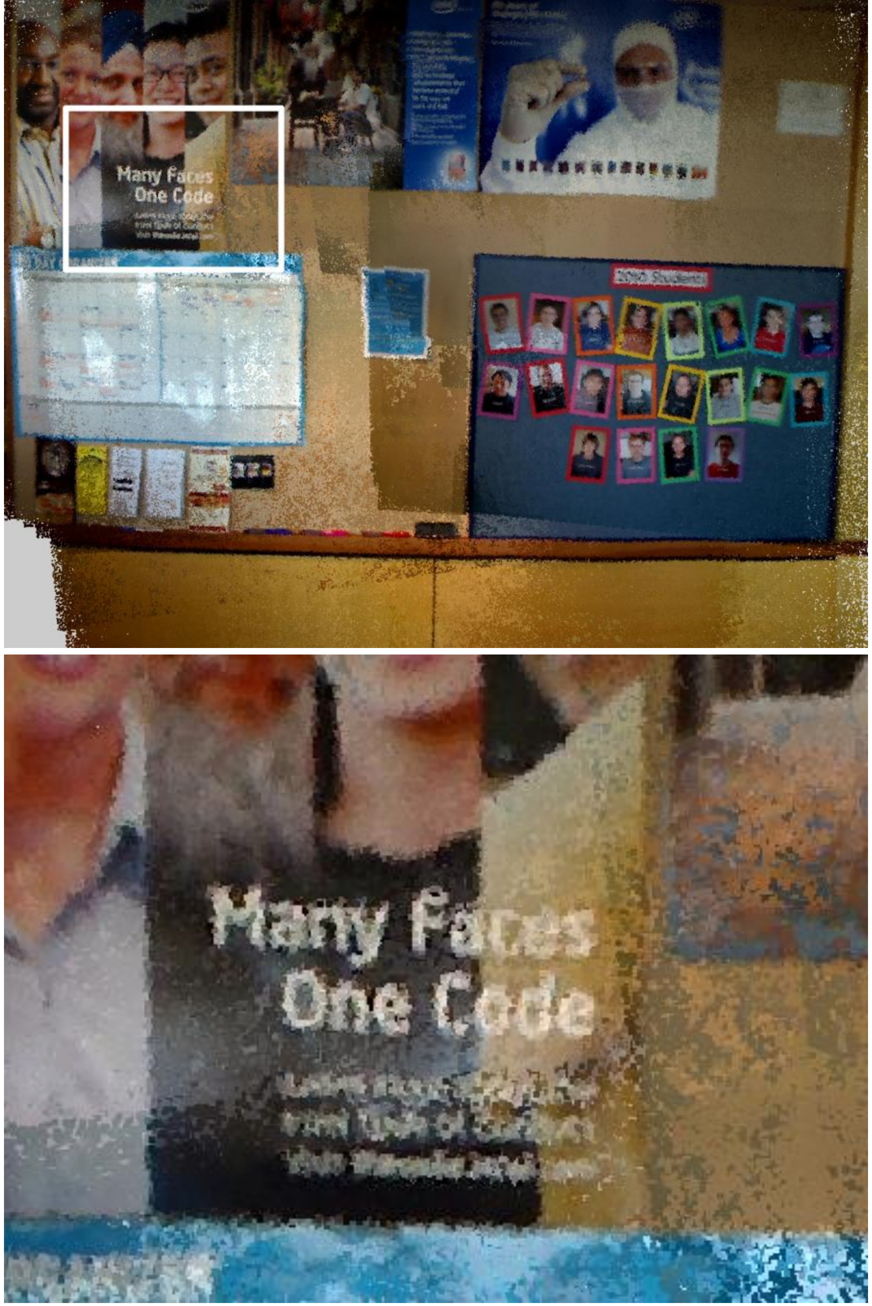
Map Representation: Surfels

- *Surface Elements* [Pfister 2000, Weise 2009, Krainin 2010]
  - Points parameterized with a normal and a radius
  - Describe circular discs in 3D (≈ellipse in image space)
  - Set of surfels can be used to approximate 3D surface

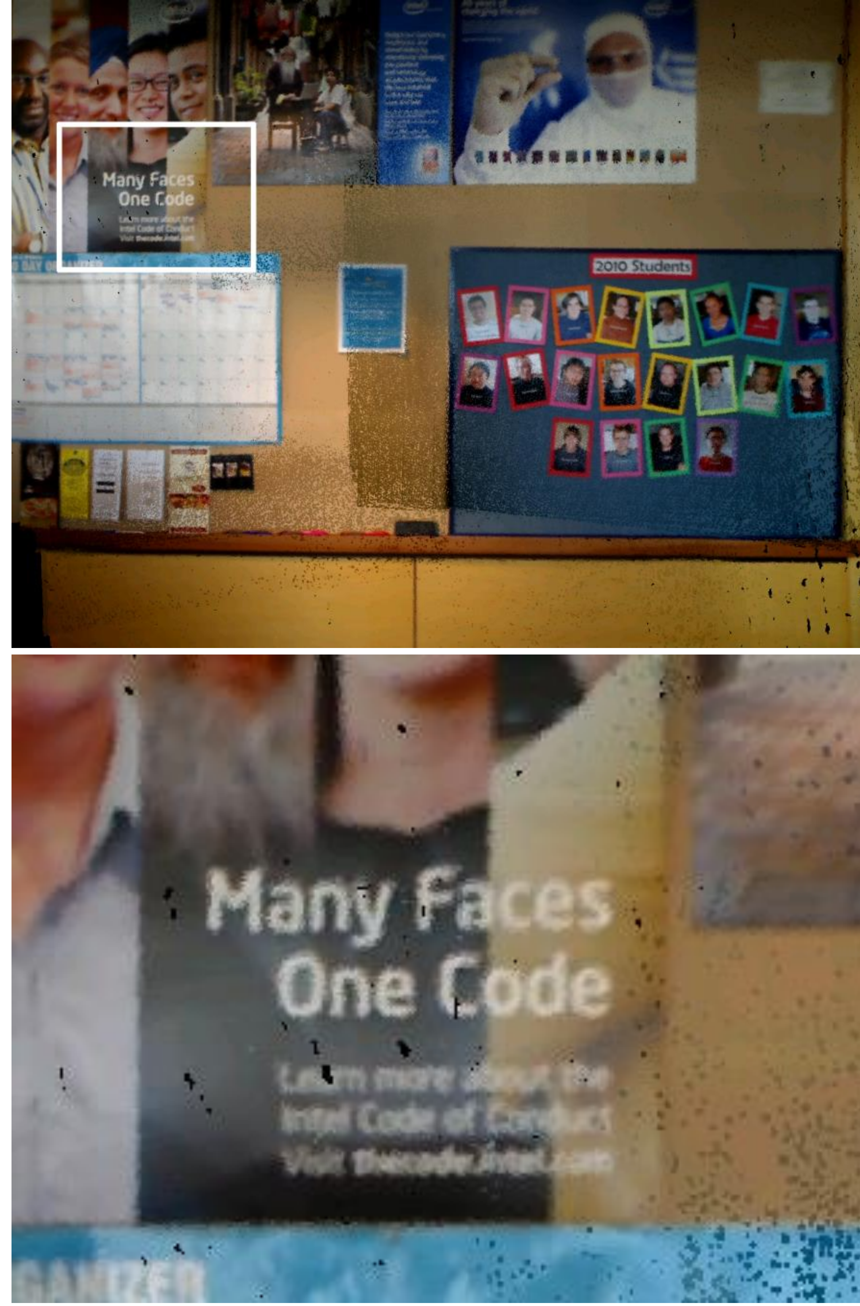


Map Representation: Surfels

- *Surface Elements* [Pfister 2000, Weise 2009, Krainin 2010]
- Circular surface patches
- Accumulate color/orientation/size information
- Incremental, independent updates
- Incorporate occlusion reasoning
- 750 million points reduced to 9 million surfels

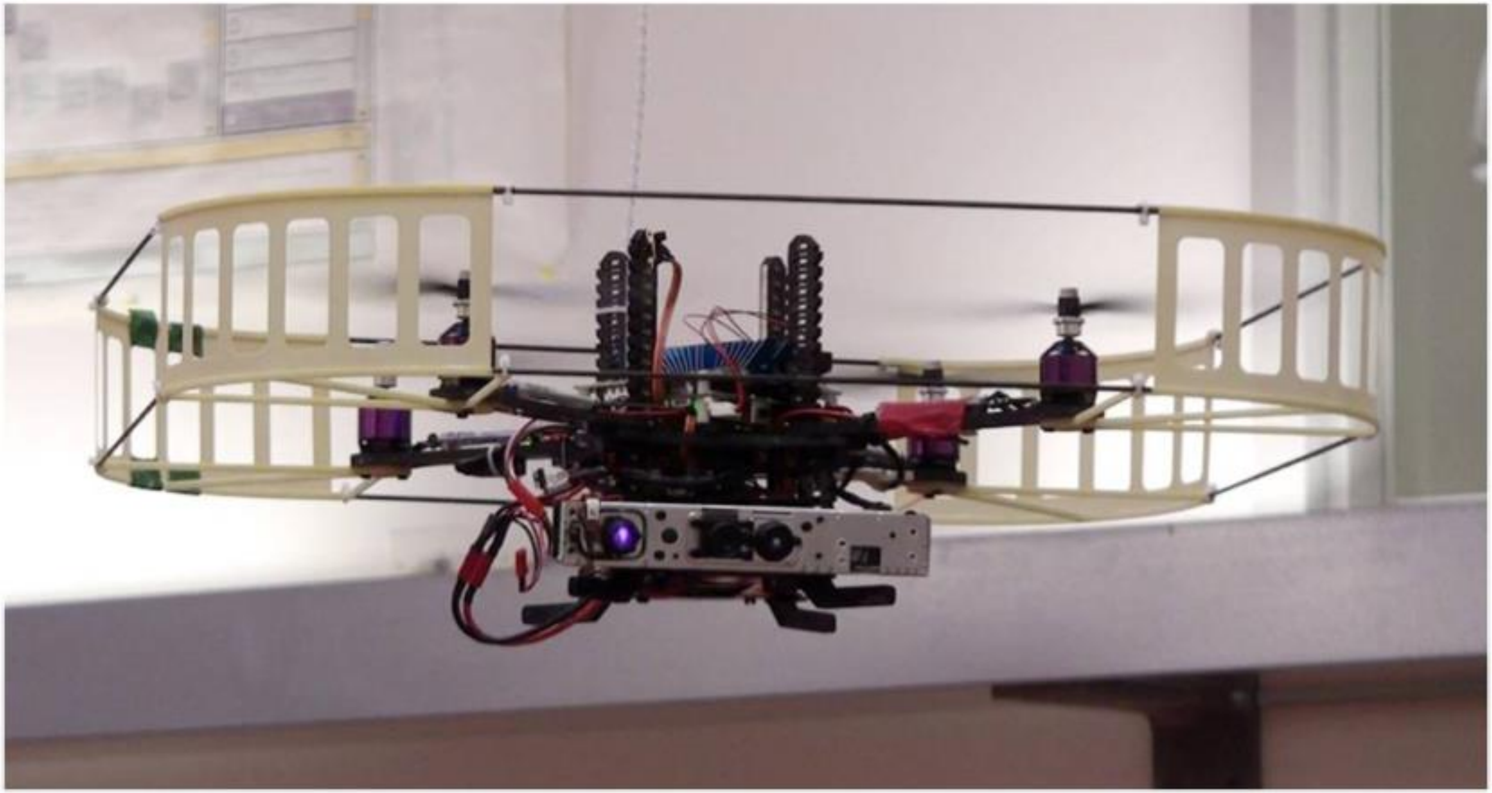


750 million points



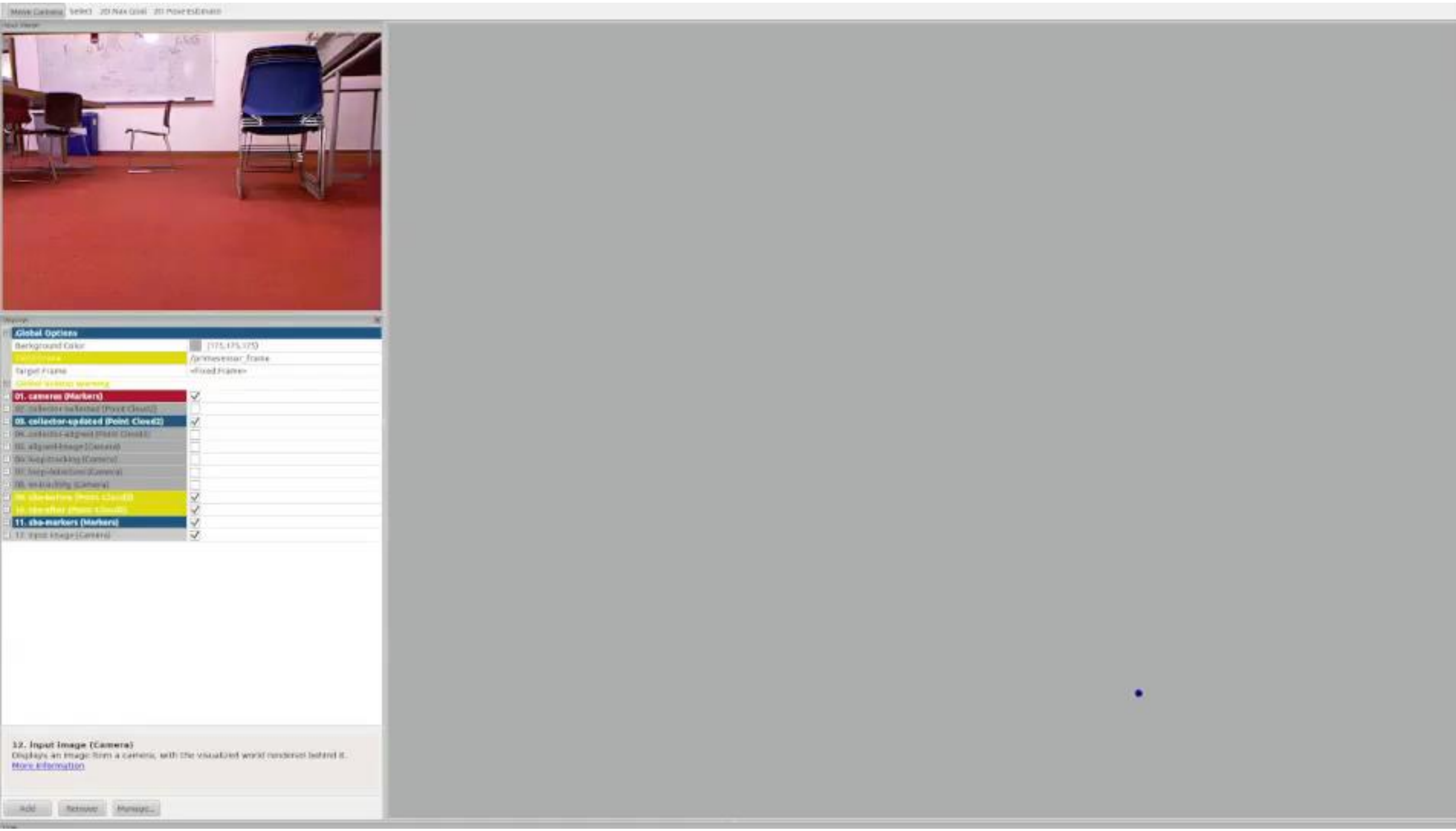
9 million surfels

Application: Quadcopter

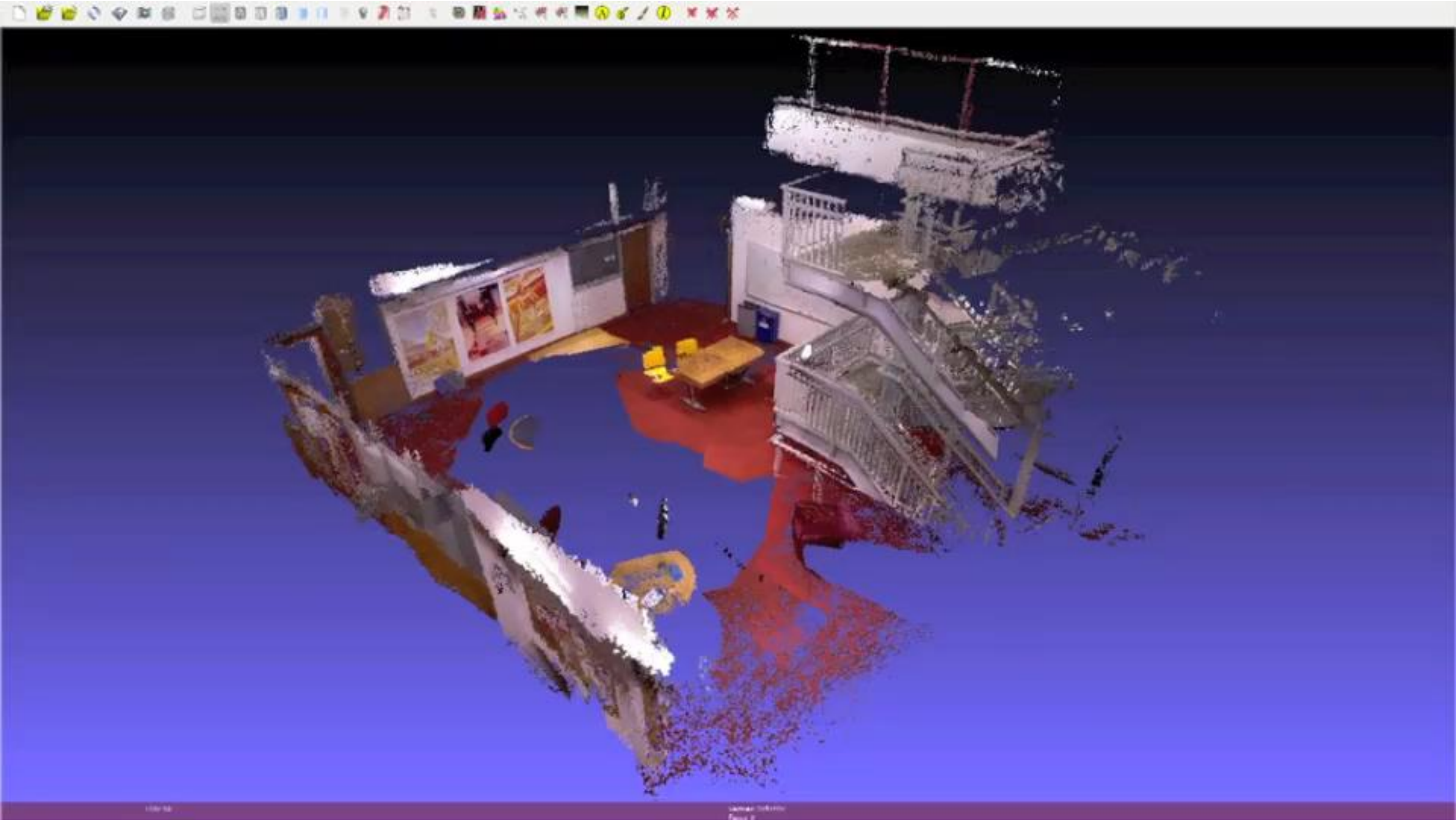


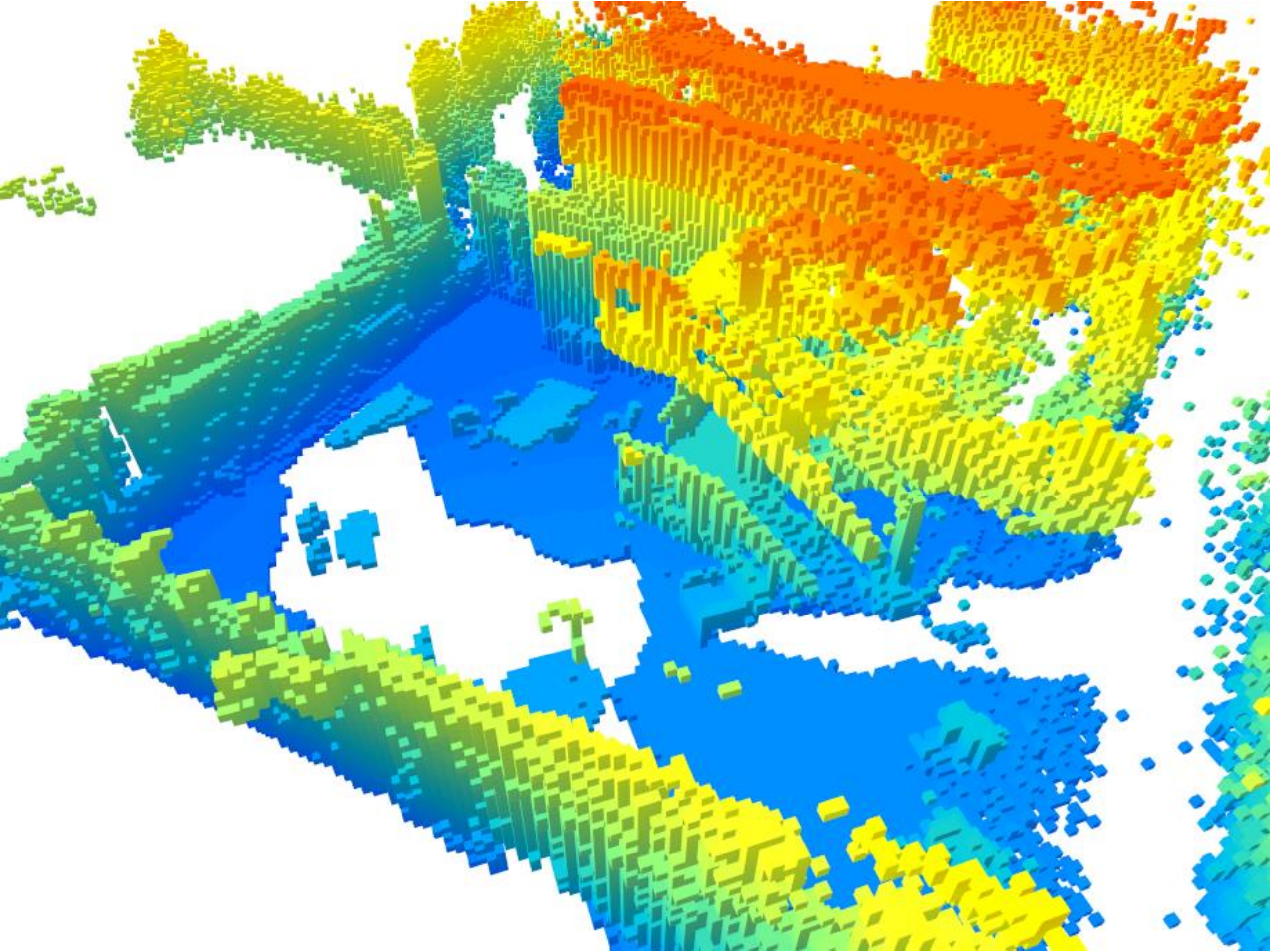
Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera. Huang, Bachrach, Henry, Krainin, Maturana, Fox, Roy. ISRR 2011

Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments. Bachrach, Prentice, He, Henry, Huang, Krainin, Maturana, Fox, Roy et al. IJRR 2012



Global Options
Background Color
Target Frame
01. cameras (Markers)
03. collector-updated (Point Cloud2)
11. sba-markers (Markers)
12. Input Image (Camera)
More information
Add
Remove
Manage...





Application: Interactive Mapping

- Allow anyone to construct a 3D map with an RGB-D camera
- Detect lack of features, guide user to correct errors
- Show map progress to assist completion
- Example applications
  - Localization
  - Measurements
  - Virtual flythrough / furniture shopping

Interactive 3D Modeling of Indoor Environments with a Consumer Depth Camera. Du, Henry, Ren, Cheng, Goldman, Seitz, Fox. UbiComp 2011

Point Cloud View
Latest Frame : 3
Health : Poor
RGB-D Camera View

Larger Maps

Mapping and Modeling with RGB-D Cameras

University of Washington

Dieter Fox