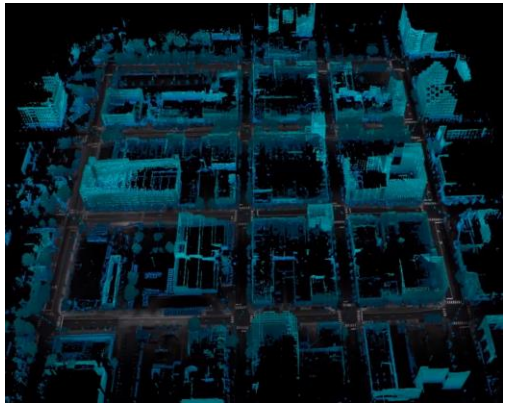
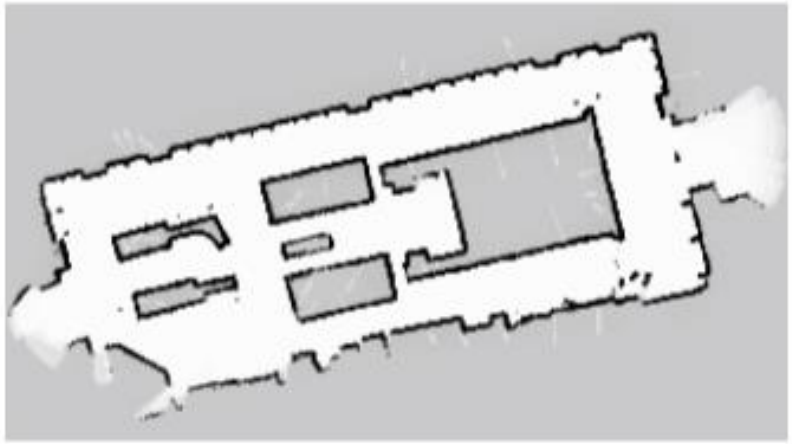
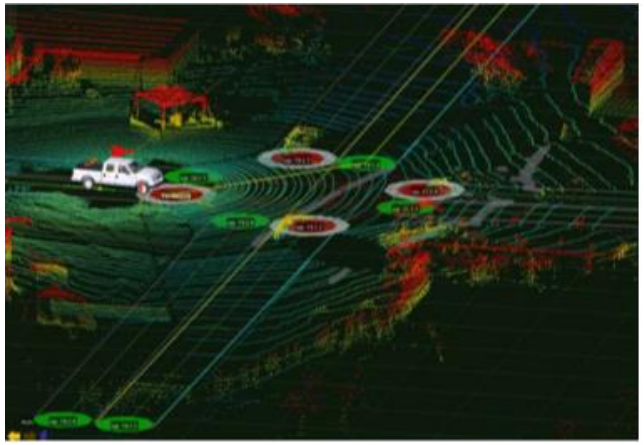
BYU Electrical & Computer Engineering





INTRODUCTION TO CAMERAS AND COMPUTER VISION

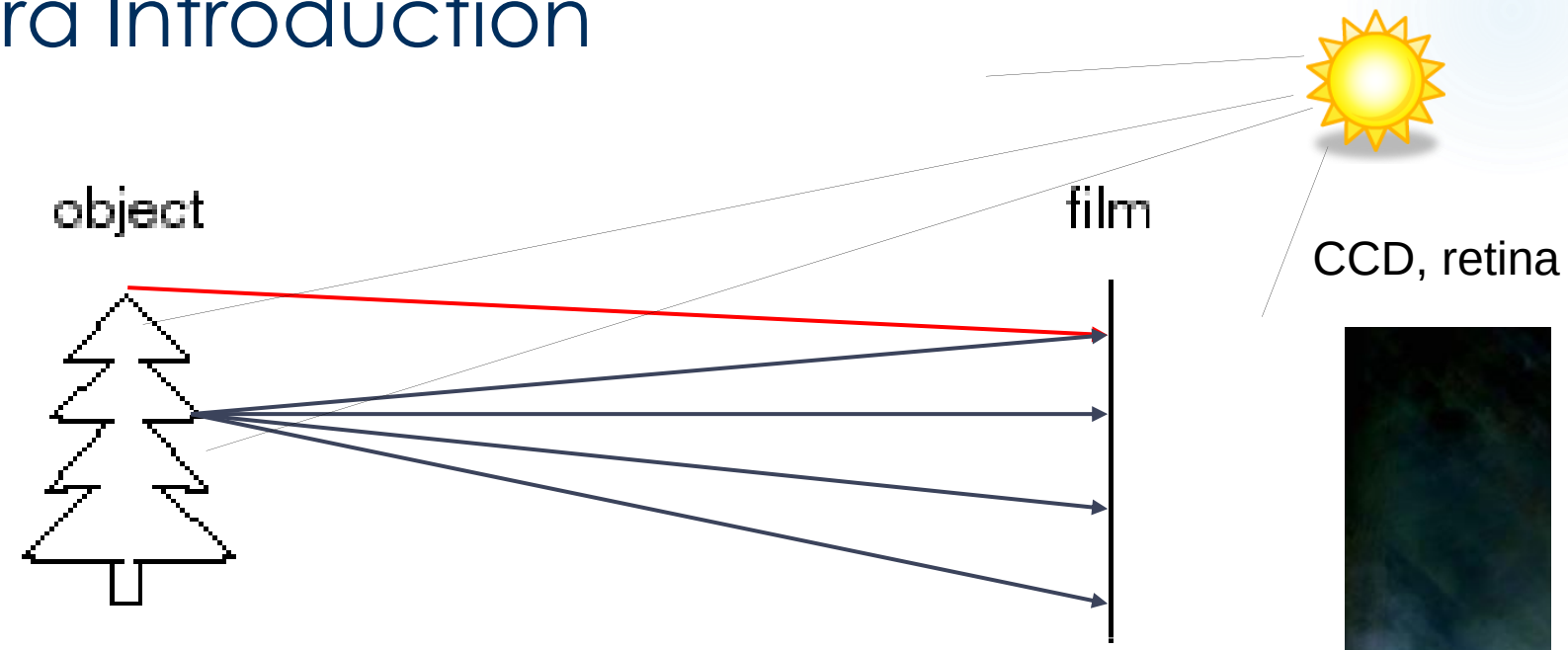
ECEN 633: Robotic Localization and Mapping

Slides courtesy of Ryan Eustice

Cameras Overview

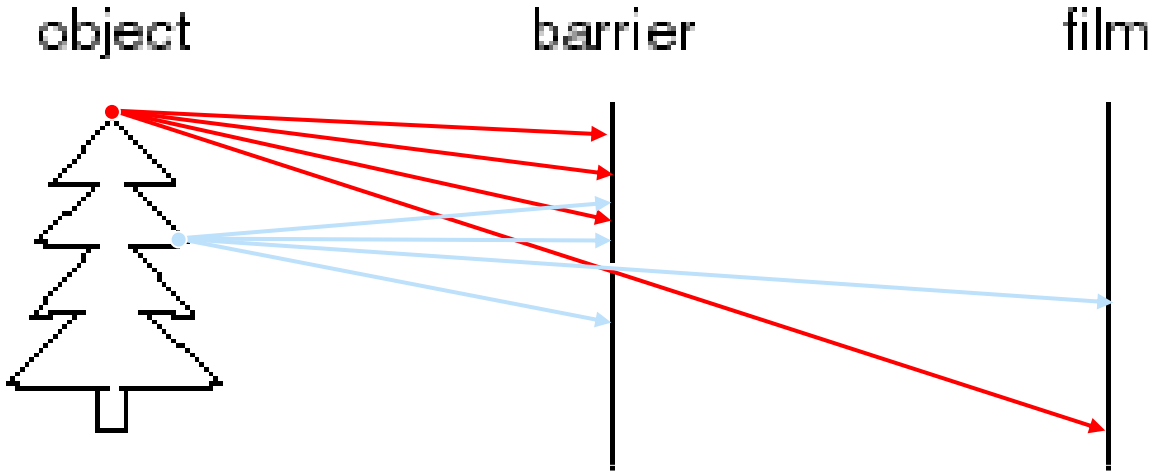
- Camera Projection Models
  - Intrinsics
  - Extrinsics
  - Lens Distortion
- Image Feature Detection
  - Harris Corners
  - SIFT Features
- Image Registration Models
  - Homography
  - Essential Matrix
  - Fundamental Matrix

Camera Introduction



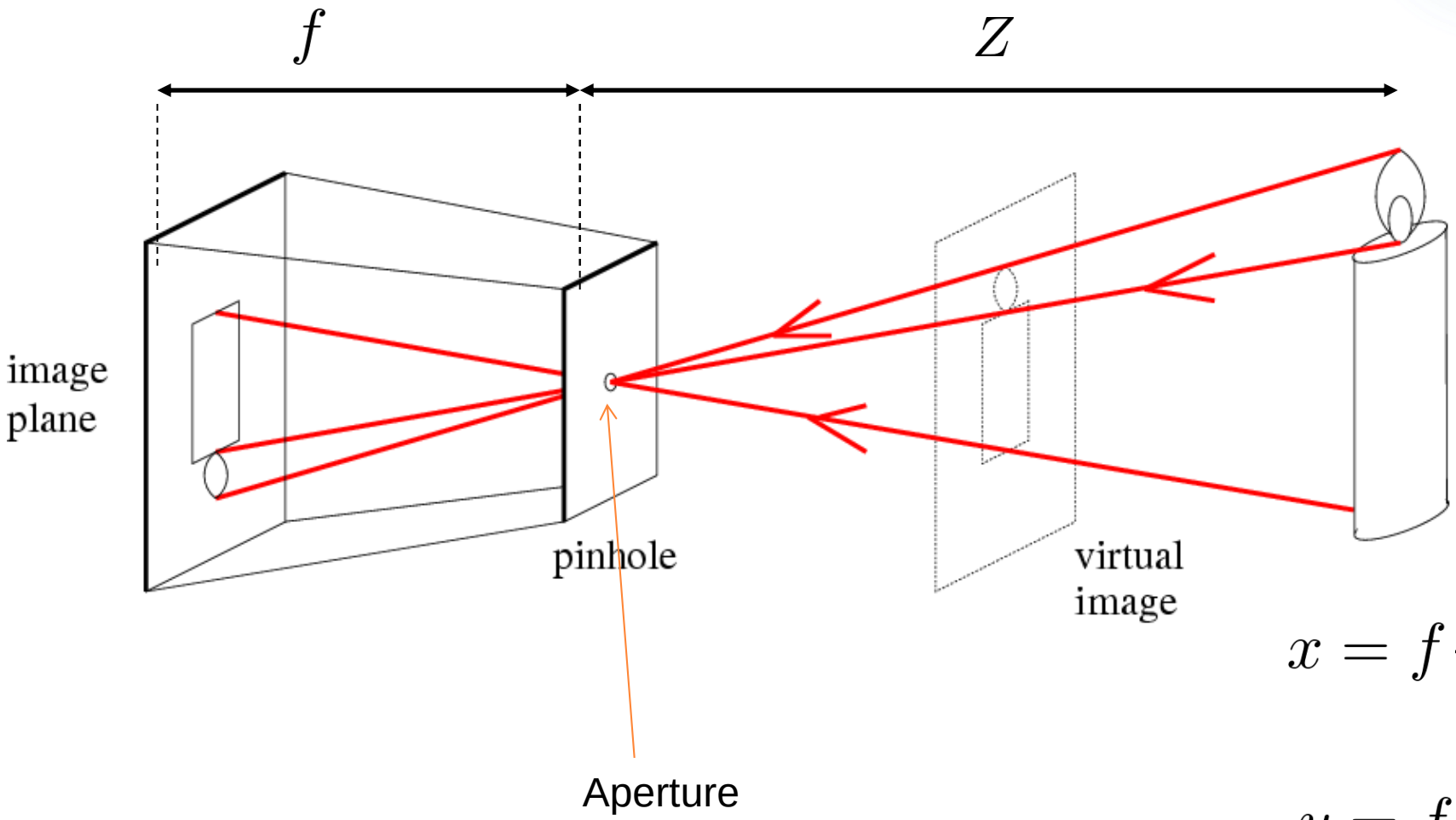
- Let's design a camera
  - Idea: put a piece of film in front of an object
  - Do we get a reasonable image?
- For now, assume:
  - Camera is at the origin
  - Looking down the z axis.

Apertures



- Add a barrier to block off most of the rays
  - Reduces blurring: obtain an image on the film
  - The opening is known as the **aperture**

Pinhole Model

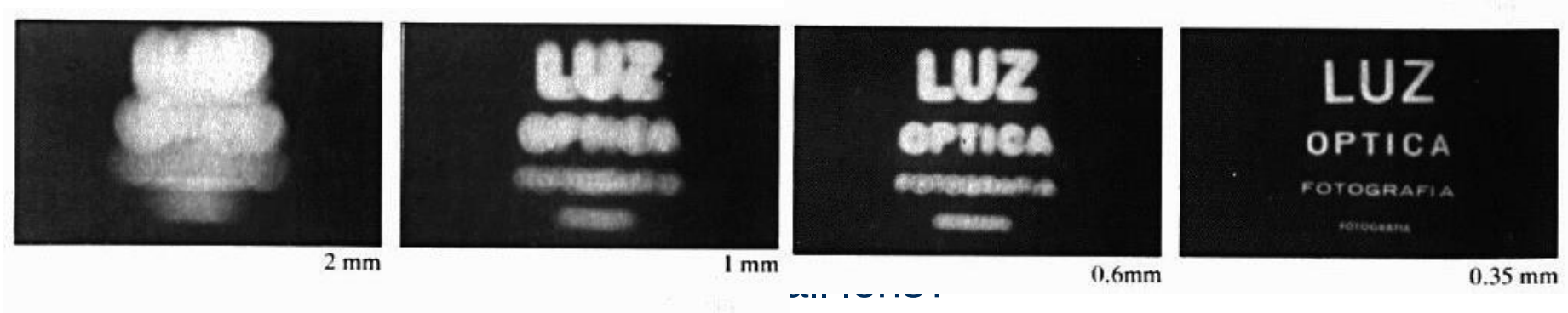


$$x = f\frac{X}{Z}$$

$$y = f\frac{Y}{Z}$$

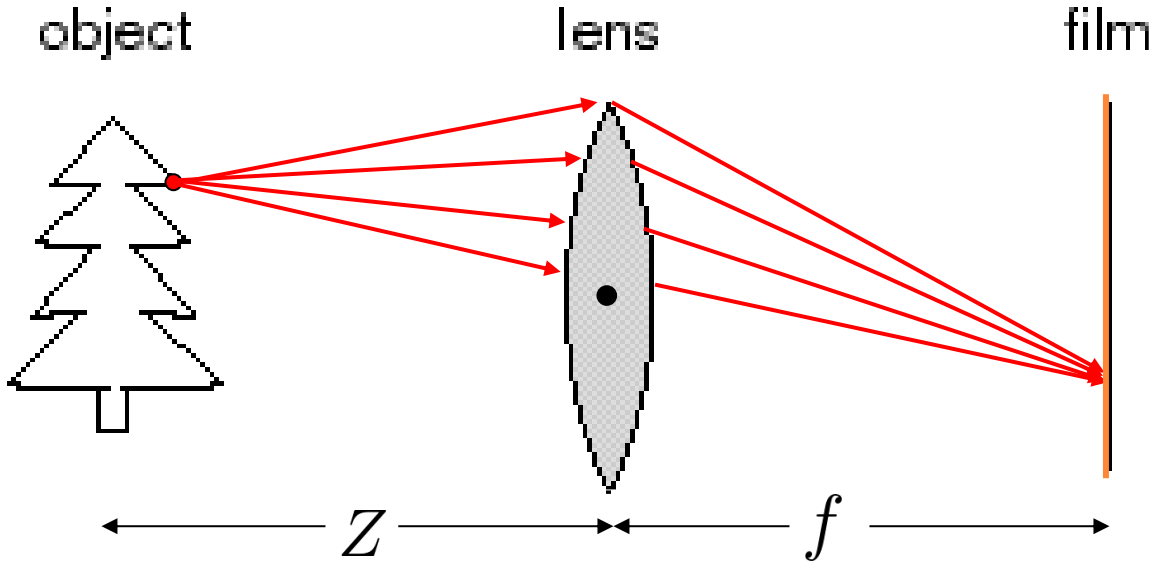
Aperture Size

- Large aperture→blurry image



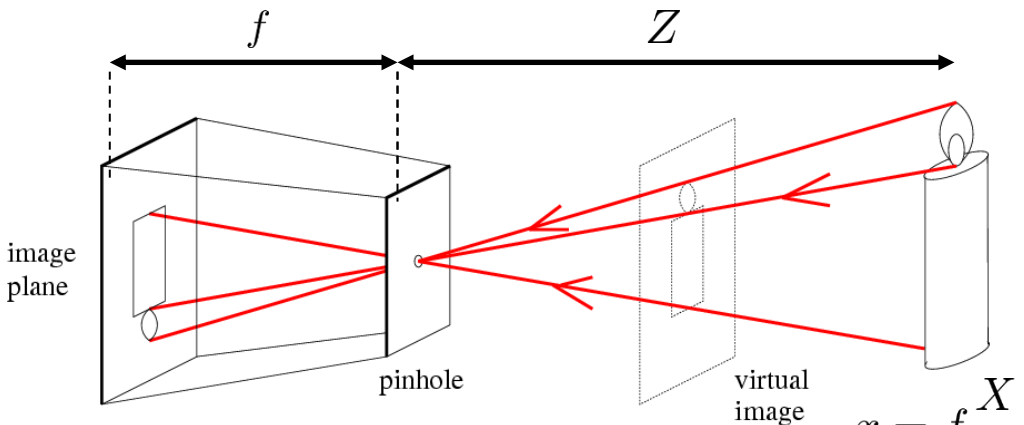
- Well, no…
  - No light gets through
  - Pesky diffraction effects.
- Answer: Lenses. Sharp images, lots of light.

Lenses



- A lens focuses light onto the film
  - Rays passing through the center are not deviated
  - Diverging rays from an object at distance Z are refocused to a point at the *focal length* f
- When in focus, we can use a pinhole perspective model.
  - But now we have the risk of being out of focus.

Intrinsic Parameters



$$x = f\frac{X}{Z}$$

$$y = f\frac{Y}{Z}$$

Homogeneous Image coordinates

$$\mathbf{x}' = \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathrm{K}} \begin{bmatrix} \mathrm{I} \mid 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathrm{P}\mathbf{X}'$$

Homogeneous World coordinates

K 3£3 *matrix of Intrinsic parameters*

Intrinsic Parameters

- Suppose that the image center was offset in the image plane by (p_x, p_y) …

$$\mathbf{x}' = \begin{bmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{bmatrix} = \underbrace{\begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathrm{K}} \begin{bmatrix} \mathrm{I} \mid 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Intrinsic Parameters

- Now map to a CCD camera where m_x and m_y represent the number of pixels per unit distance in the x and y directions, respectively (possibly not the same, i.e., not square pixels)

$$\begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

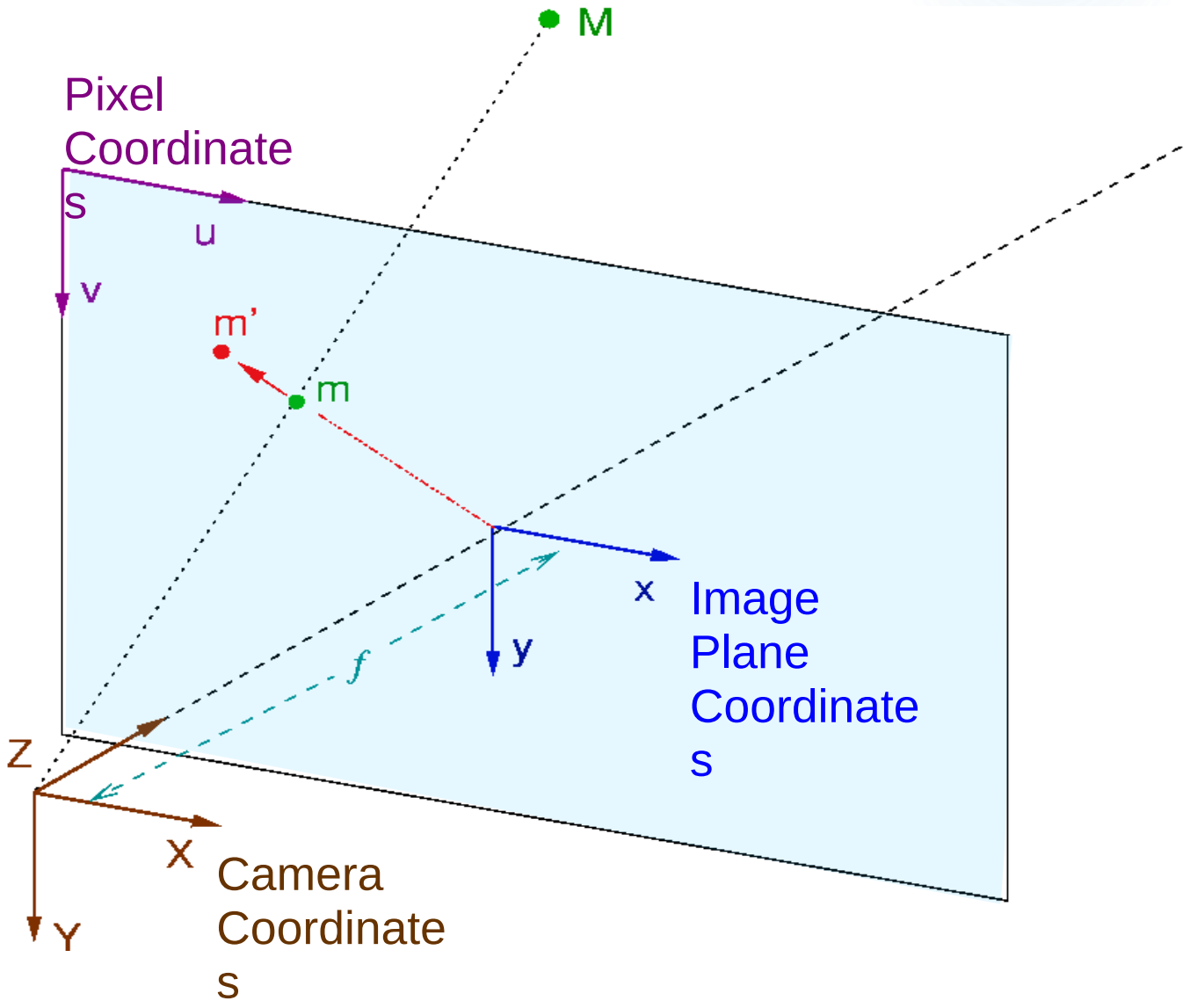
Intrinsic Parameters

- Finally, in general, we consider a calibration matrix of the form

$$\mathrm{K} = \begin{bmatrix} \alpha_x & s & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

- where s is referred to as the skew parameter (will be zero for most normal cameras).

Pinhole Camera Model



Extrinsic Parameters

- Recall from before our matrix equation for pinhole projection

$$\mathbf{u}' = \mathrm{K}[\mathrm{I} \mid 0]\mathbf{X}'$$

- We can translate/rotate by 6DOF rigid-boy transformation ${}^{c}_{w}\mathrm{T}$ the world point into the camera frame before projecting

$${}^{c}\mathbf{X}' = {}^{c}_{w}\mathrm{T}\,{}^{w}\mathbf{X}'$$

- Combing the two

$$\mathbf{u}' = \underbrace{\mathrm{K}\left[{}^{c}_{w}\mathrm{R} \mid {}^{w}\mathbf{t}_{cw}\right]}_{\mathrm{P}}\,{}^{w}\mathbf{X}'$$

3£4 *projection matrix*
(accounts for both intrinsics & extrinsics)

Projection Matrix

- The resulting $3£4$ projection matrix P accounts for both the intrinsic AND extrinsic parameters

$$\mathbf{u}' = \mathrm{K}[\mathrm{R} \mid \mathbf{t}]\mathbf{X}'$$

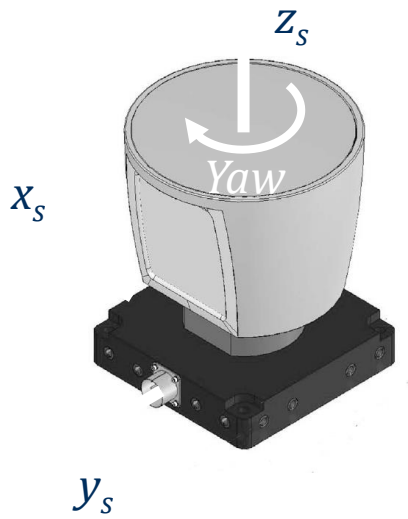
$$= \mathrm{P}\mathbf{X}'$$

where

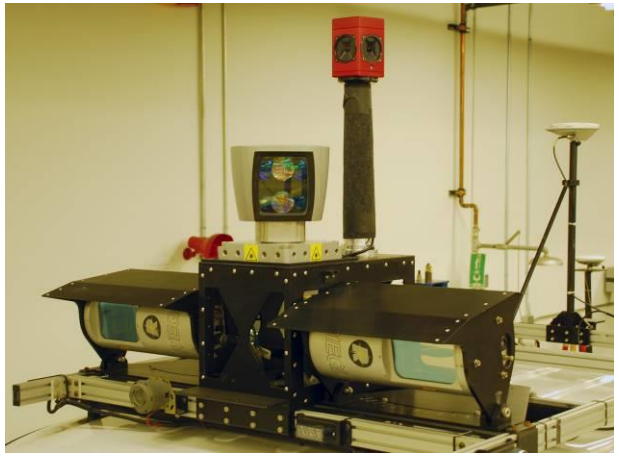
$$\mathrm{P} = \mathrm{K}[\mathrm{R} \mid \mathbf{t}]$$

$3£4$ *projection matrix*
(accounts for both intrinsics & extrinsics)

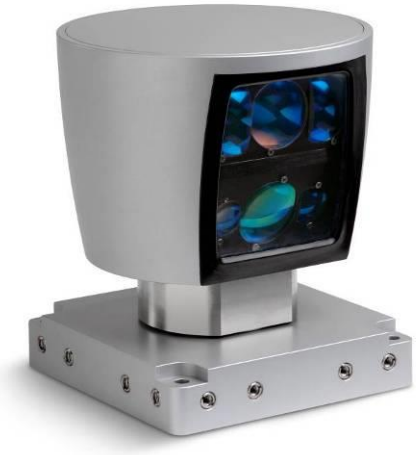
Lidar-Camera Projection



3-D Mapping on the XAV-250



- Ladybug 3 spherical camera
- Velodyne HDL-64E LIDAR
- Riegl LMS-Q120 LIDAR
- Delphi ACC3 radar
- Applanix POS-LV 420 INS

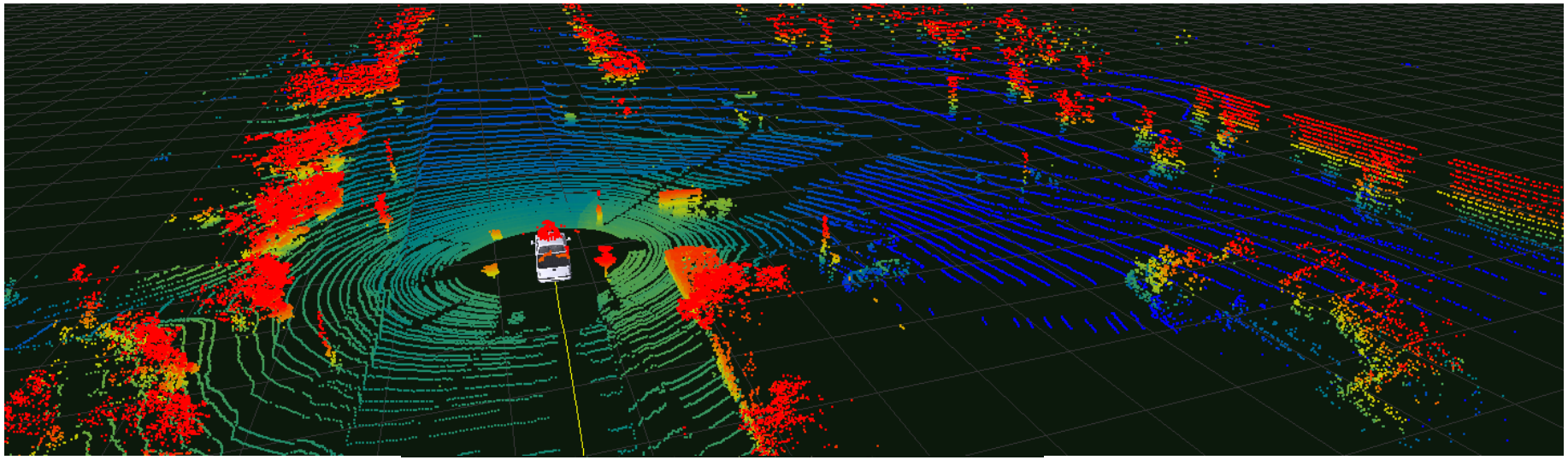


- 64 beams, 360° FOV
- +2.5° to –24° elevation
- 10Hz refresh rate
- >1,000,000 points/s
- 120m range, 2cm accuracy

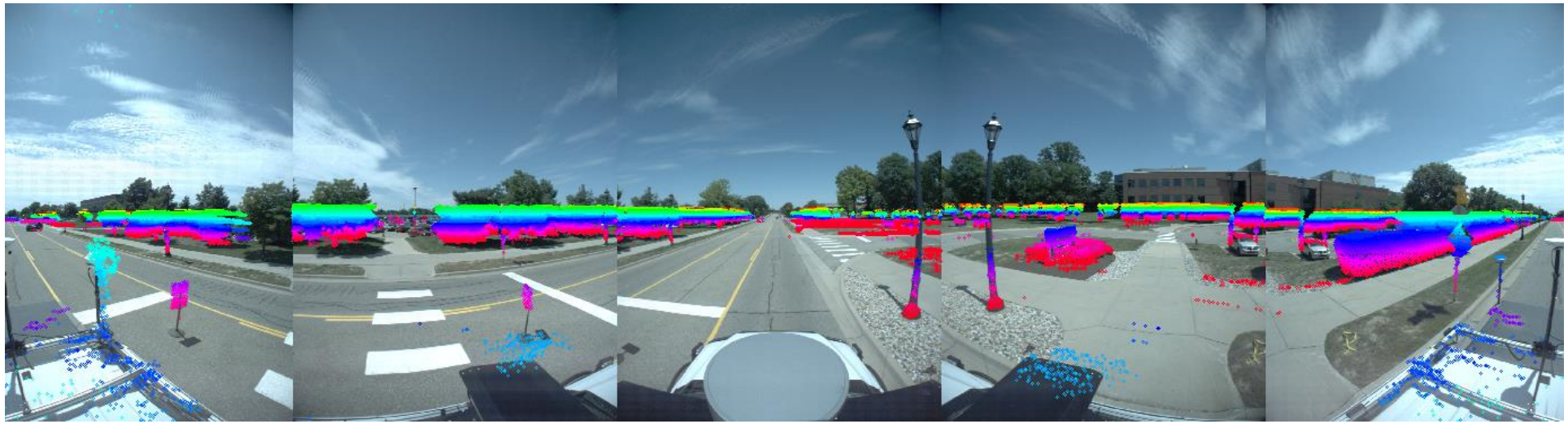
- 6 1.125” Sony CCD imagers
- 12MP resolution (6×1600×1200)
- 15 frames/s at full resolution
- 800Mb/s 1394b interface
- Embedded JPEG compression

Data Acquisition

LIDAR vs. Camera View



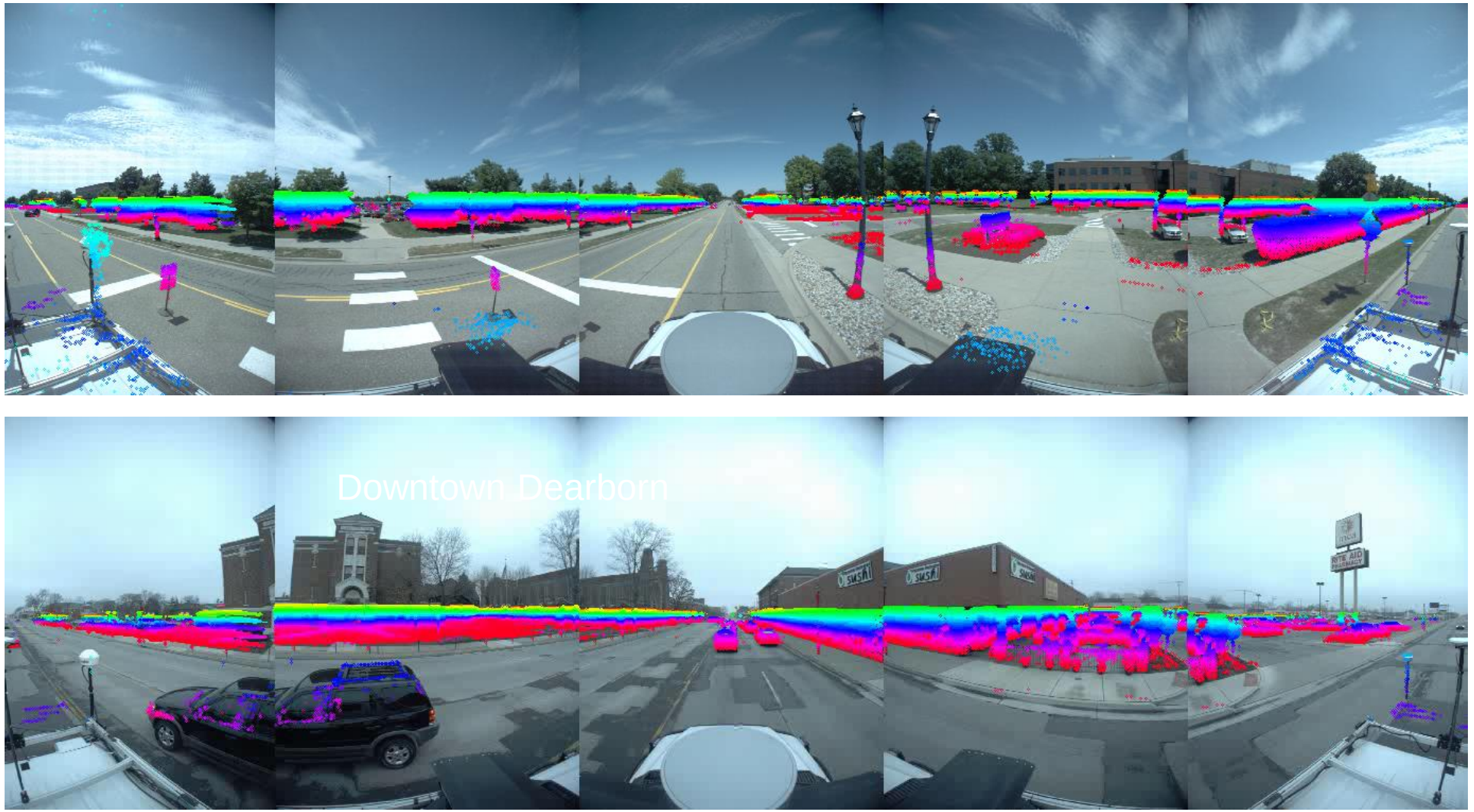
Co-Registration of Laser and Camera Sensing Using Pin-Hole Camera Model



$$ {}^{c_i}\mathbf{u}' = \mathrm{K}_i \left[{}^{c_i}_{\ell}\mathrm{R} \mid {}^{c_i}\mathbf{t}_{c_i\ell} \right] {}^{\ell}\mathbf{X}' $$

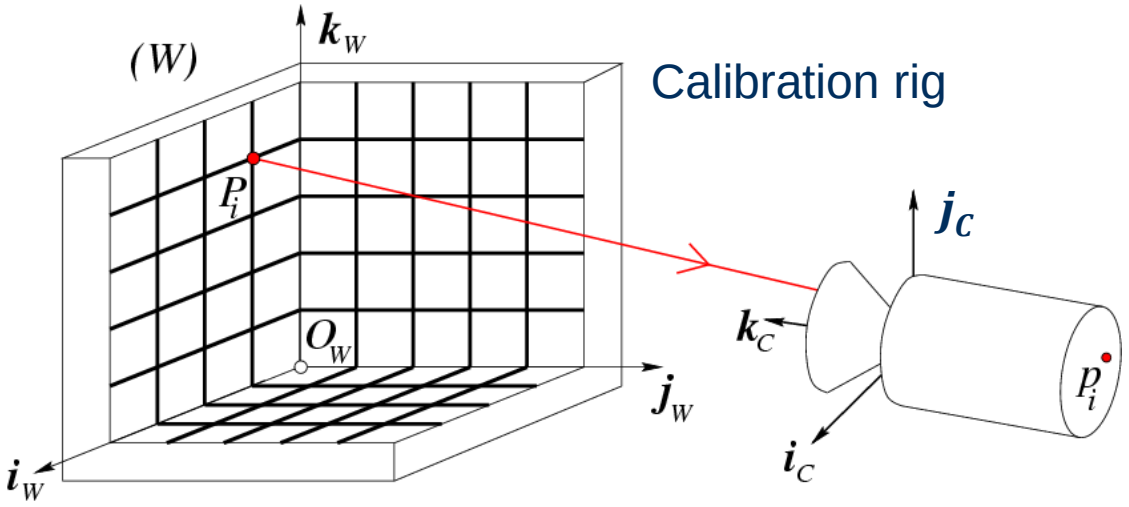
Fused LIDAR and Camera Imagery

(LIDAR Color-coded by Height Above Ground Plane)



Camera Calibration (Single-View)

- How do we compute the intrinsic/extrinsic parameters?

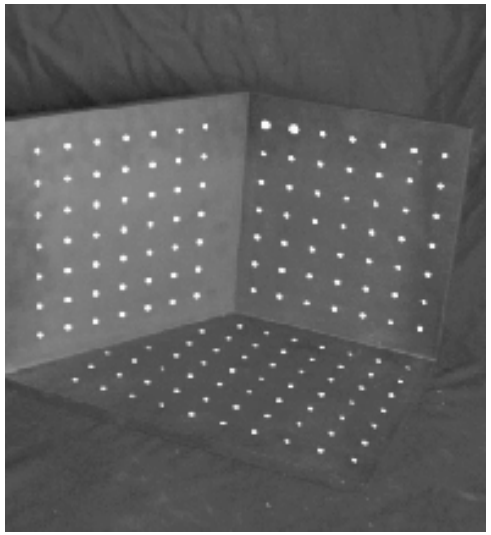


- $P_1 \ldots P_n$ with known positions in $[O_w, i_w, j_w, k_w]$
- $p_1, \ldots p_n$ known positions in the image

How many correspondences do I need?

Camera Calibration (Single-View)

- We actually build one of these rigs…
- Extract points from image
  - Use known target geometry



- Optimization problem: find parameter settings that project 3D scene points to the correct image coordinates.

$$X = f(P)$$

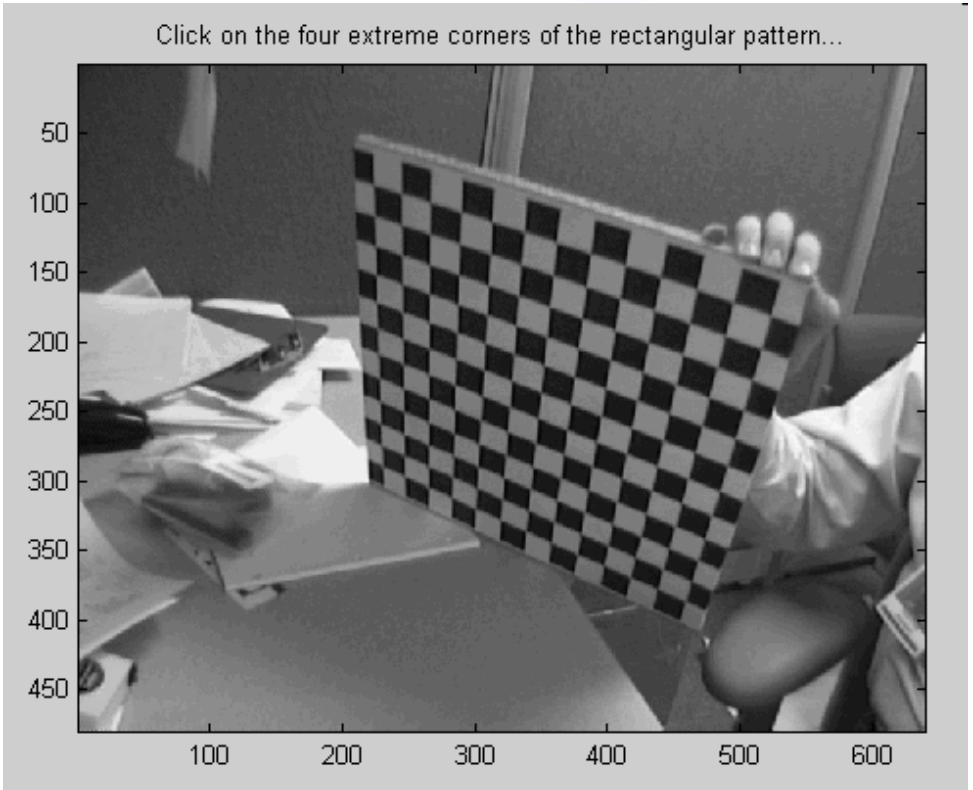
measurement

parameter

R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation, 3(4):323–344, Aug. 1987.*

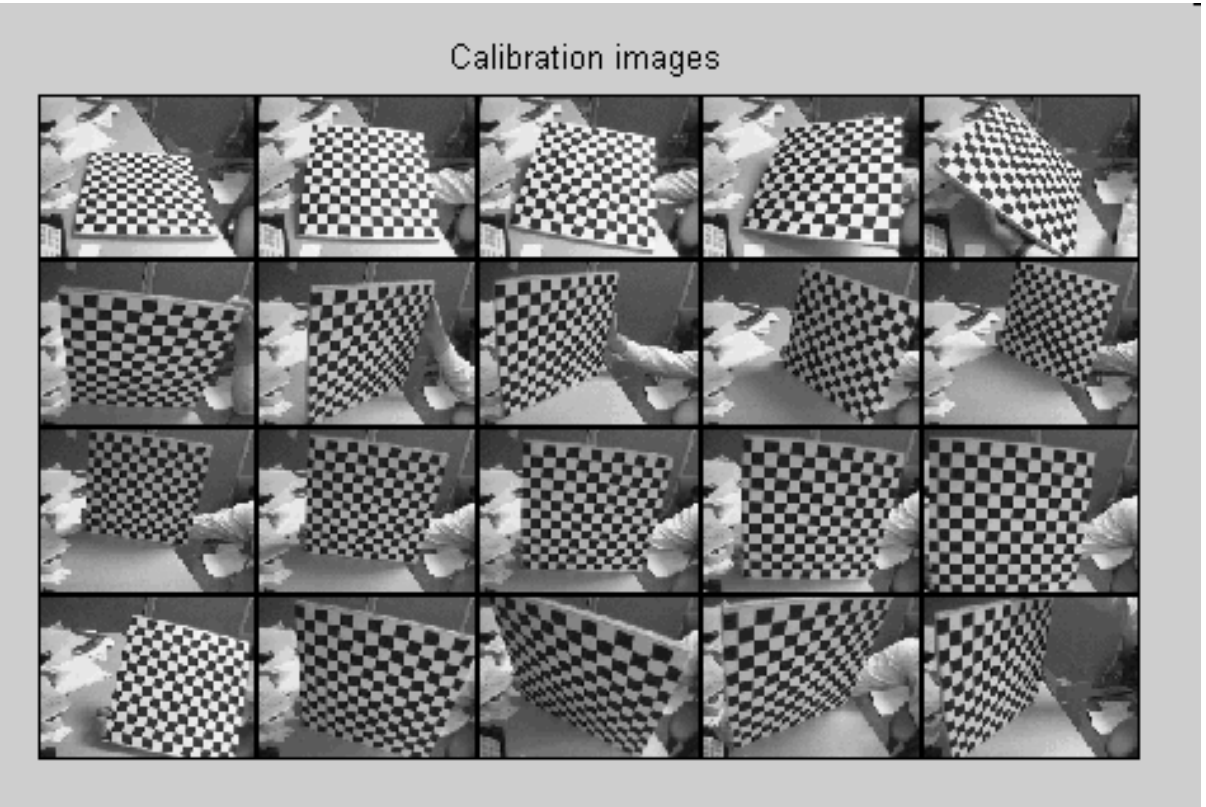
Planar Calibration

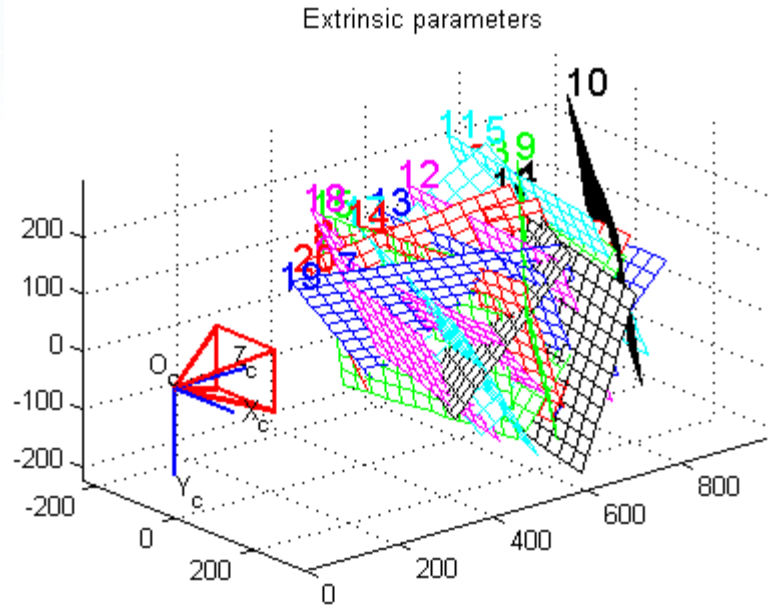
- 3D target calibration requires precision machining
- 2D target pattern much easier to come by in practice
  - Laser printer + book cover
- What do we sacrifice?
  - Less constraints per observation, requires more views
    - $N \geq 3$ views

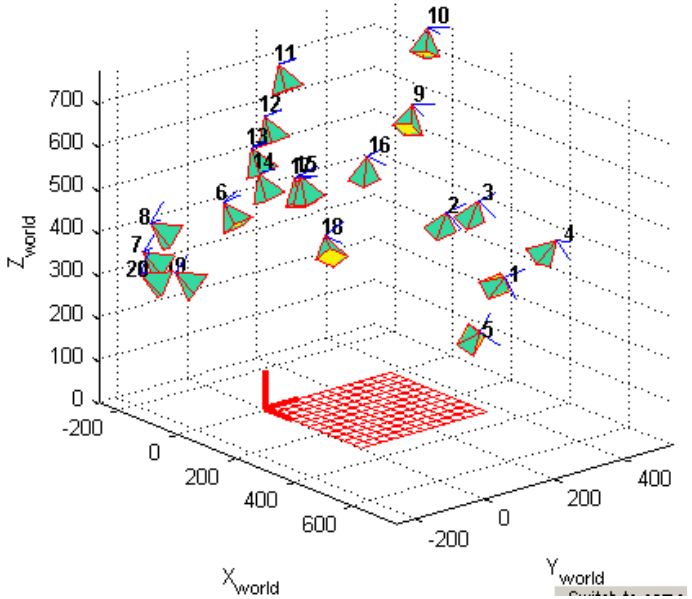


Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations Proceedings of the IEEE International Conference on Computer Vision, 1999, p.666-673

MATLAB Camera Calibration Toolbox







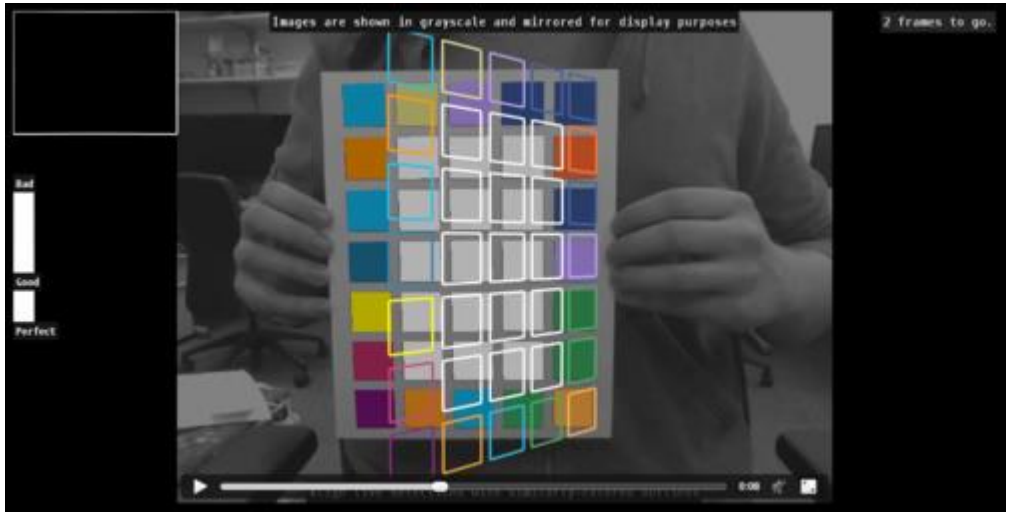
http://www.vision.caltech.edu/bouguetj/calib_doc/

AprilCal

http://april.eecs.umich.edu/wiki/index.php/Camera_suite

A. Richardson, J. Strom and E. Olson, “AprilCal: Assisted and repeatable camera calibration,” Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013

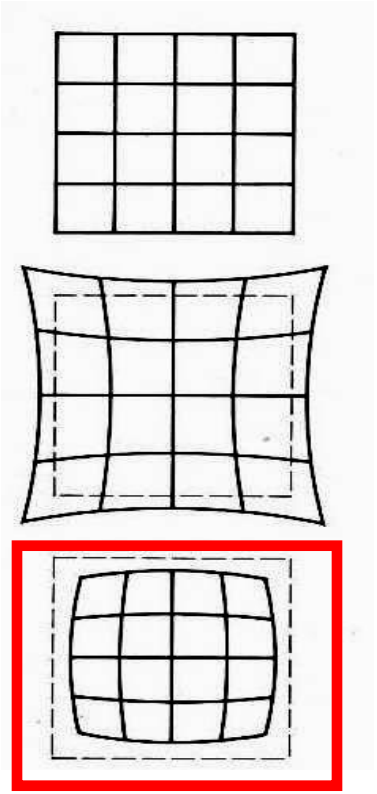
- AprilCal is a variant on the planar calibration formulation. Uses AprilTags for automatic extraction of calibration points on the planar target. Guides the user in choosing a good set of views for observing the calibration parameters.



Lens Distortion

$$\mathbf{x}_d = (1 + k_{r_0} r^2 + k_{r_1} r^4 + k_{r_2} r^6)\mathbf{x}_n + \mathbf{dx}$$

- Caused by imperfect lenses
- Deviations are most noticeable for rays that pass through the edge of the lens



No distortion

Pin cushion

$k_{r_0} > 0$

Barrel

$k_{r_0} < 0$

Lens Distortion Models

- Let $\mathbf{x}_n$ be the normalized (pinhole) projection

$$\mathbf{x}_n = \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}$$

- Let $r^2 = x^2 + y^2$ be the radial distance from the COP. After including lens distortion, the new normalized point coordinate $\mathbf{x}_d$ is defined as:

$$\mathbf{x}_d = \underbrace{(1 + k_{r_0} r^2 + k_{r_1} r^4 + k_{r_2} r^6)}_{\text{Radial distortion term}} \mathbf{x}_n + \underbrace{\mathbf{dx}}_{\text{tangential distortion term}}$$

D.C. Brown, Close-Range Camera Calibration, Photogrammetric Engineering, pages 855-866, Vol. 37, No. 8, 1971.

Lens Distortion Models…

- Where the tangential distortion term is given by

$$\mathbf{dx} = \begin{bmatrix} 2k_{t_0}xy + k_{t_1}(r^2 + 2x^2) \\ k_{t_0}(r^2 + 2y^2) + 2k_{t_1}xy \end{bmatrix}$$

- This yields the following distorted pixel coordinates

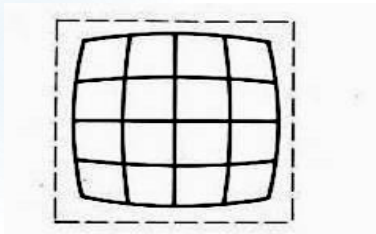
$$\begin{bmatrix} u_d \\ v_d \\ 1 \end{bmatrix} = \mathrm{K} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \qquad \mathrm{K} = \begin{bmatrix} \alpha_x & s & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/parameters.html

Ladybug3 Imagery (Barrel Distortion)

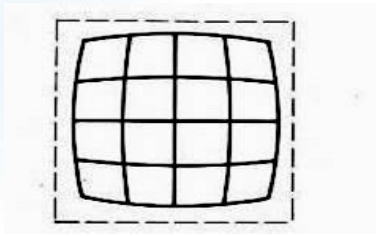
Barrel

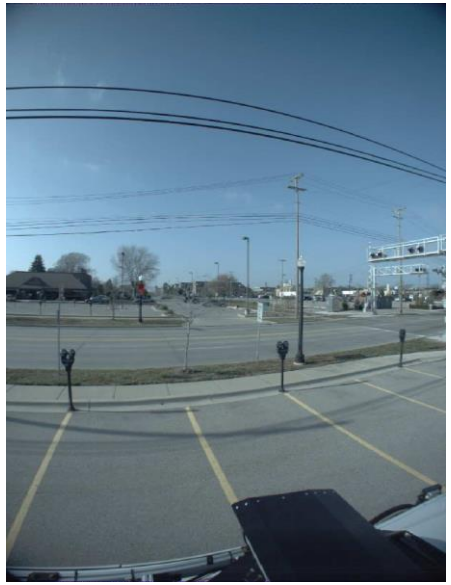
$k_{r_0} < 0$



Ladybug3 Imagery (Barrel Distortion)

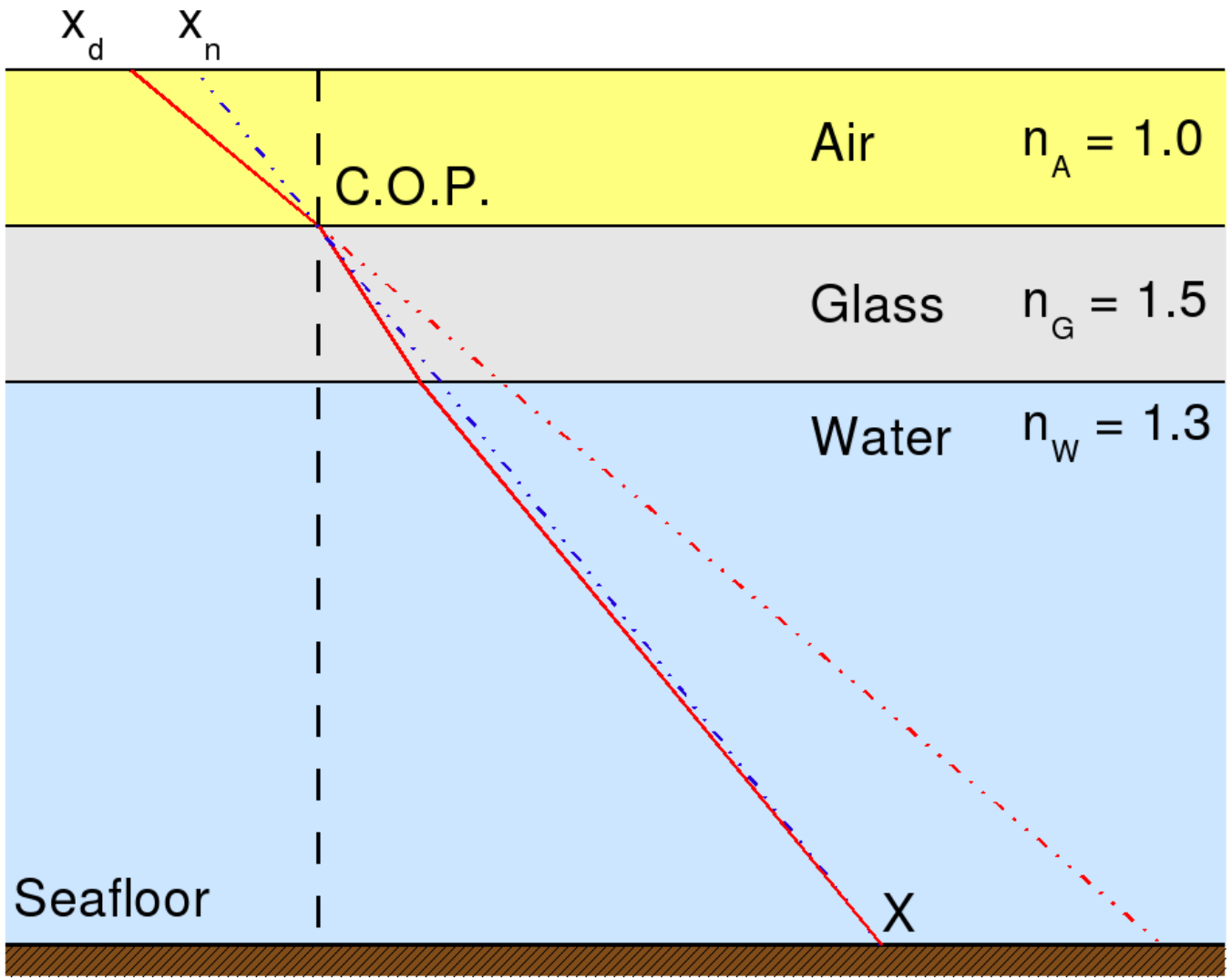
$k_{r_0} < 0$





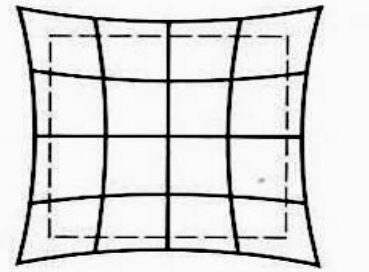
Underwater Flat Plate Distortion

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{v_1}{v_2} = \frac{n_2}{n_1}$$

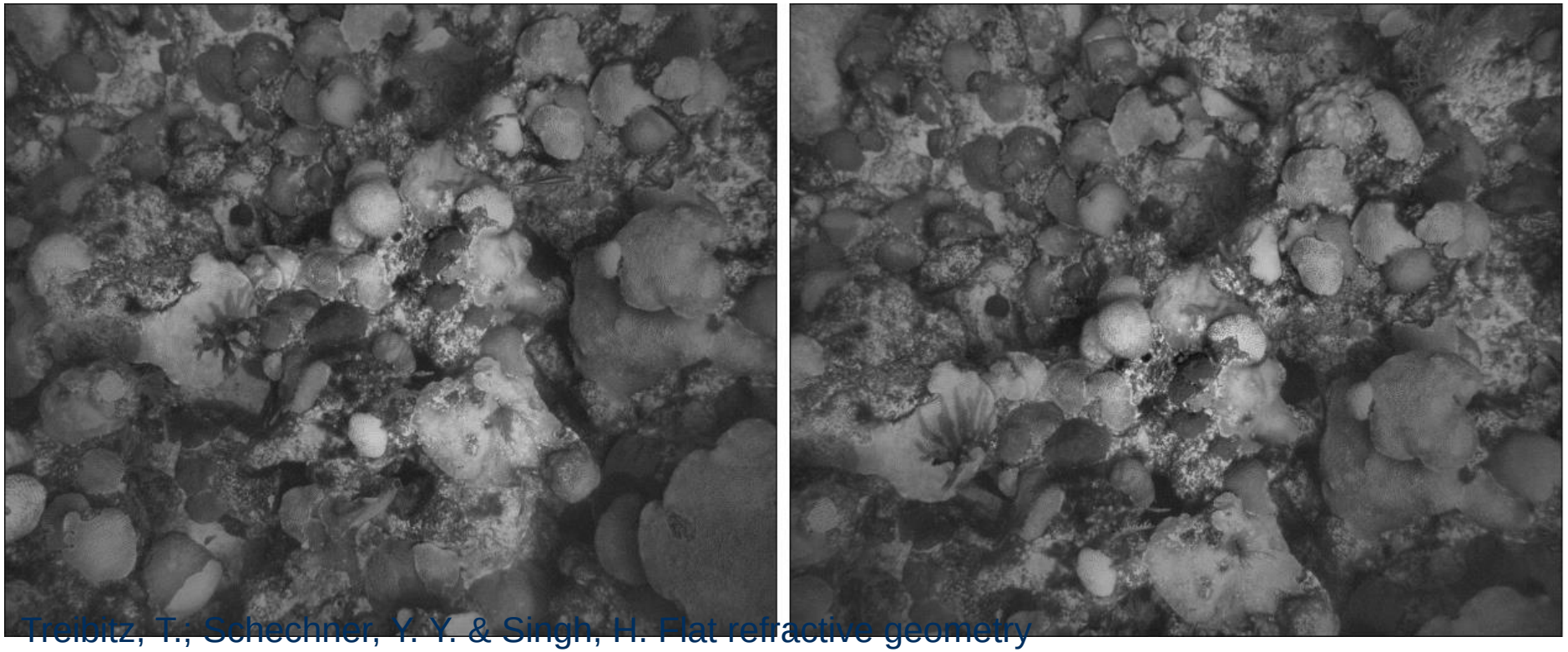


Radially Compensated U/W Imagery

$k_{r_0} > 0$



~Pin-cushion radial distortion for lens close to flat plate.



Treibitz, T.; Schechner, Y. Y. & Singh, H. Flat refractive geometry
Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, **2008**, 1-8

Camera limitations

- Scale ambiguity

Cameras Overview

- Camera Projection Models
  - Intrinsics
  - Extrinsics
  - Lens Distortion

- Image Feature Detection
  - Harris Corners
  - SIFT Features

- Image Registration Models
  - Homography and Transformations
  - Essential Matrix
  - Fundamental Matrix

Motivation: Build a Panorama

M. Brown and D. G. Lowe. Recognising Panoramas. ICCV 2003

How do we build panorama?

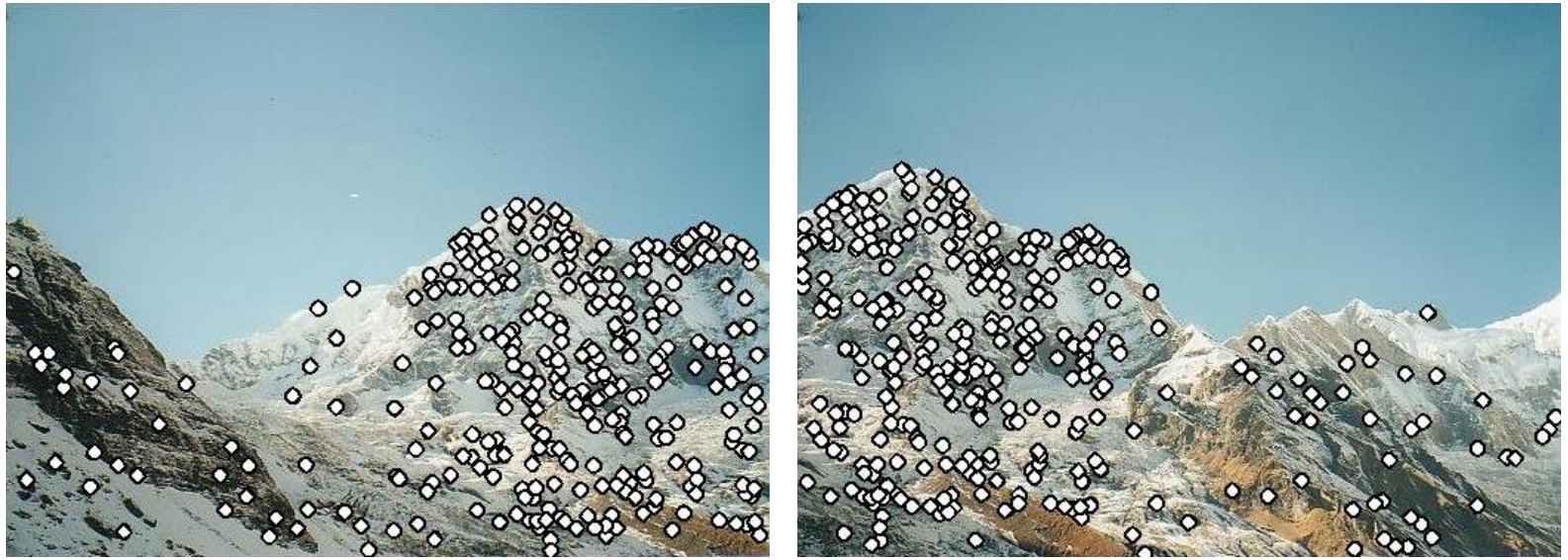
Courtesy: S. Seitz and R. Szeliski

- We need to match (align) images

Matching with Features

Courtesy: S. Seitz and R. Szeliski

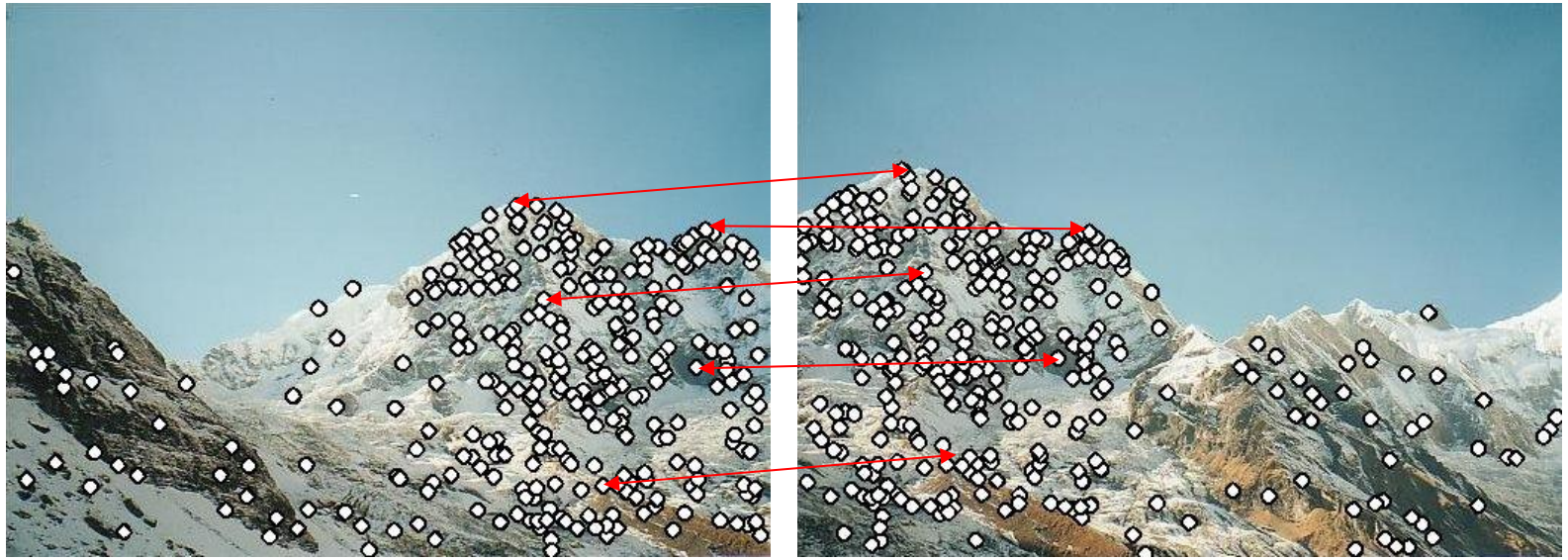
- Detect feature points in both images



Matching with Features

Courtesy: S. Seitz and R. Szeliski

- Detect feature points in both images
- Find corresponding pairs



Matching with Features

Courtesy: S. Seitz and R. Szeliski

- Detect feature points in both images
- Find corresponding pairs
- Use these pairs to align images

Courtesy: S. Seitz and R. Szeliski

Matching with Features

- Problem 1:
  - Detect the *same* point *independently* in both images

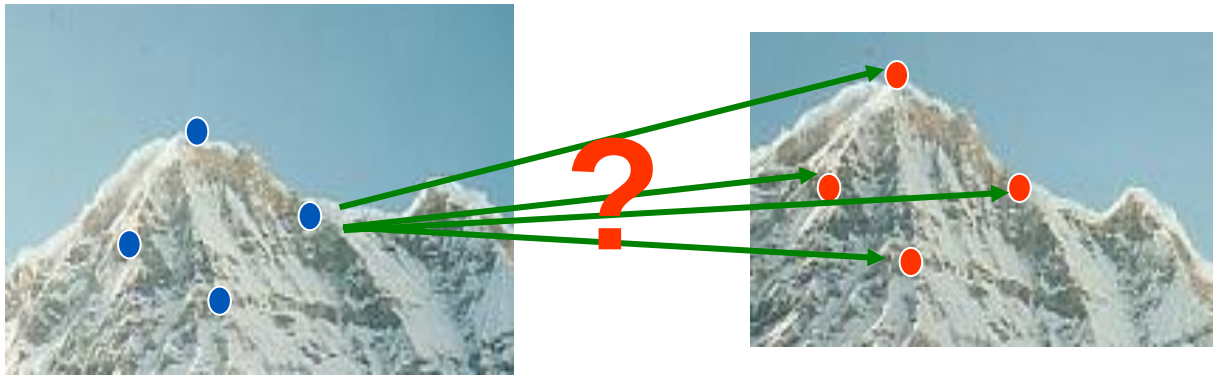
no chance to match!

We need a repeatable detector

Matching with Features

Courtesy: S. Seitz and R. Szeliski

- Problem 2:
  - For each point correctly recognize the corresponding one



We need a reliable and distinctive descriptor

Courtesy: S. Seitz and R. Szeliski

Matching with Features

- Problem 3:
  - Need to estimate transformation between images, despite erroneous correspondences.

We need a robust estimation framework

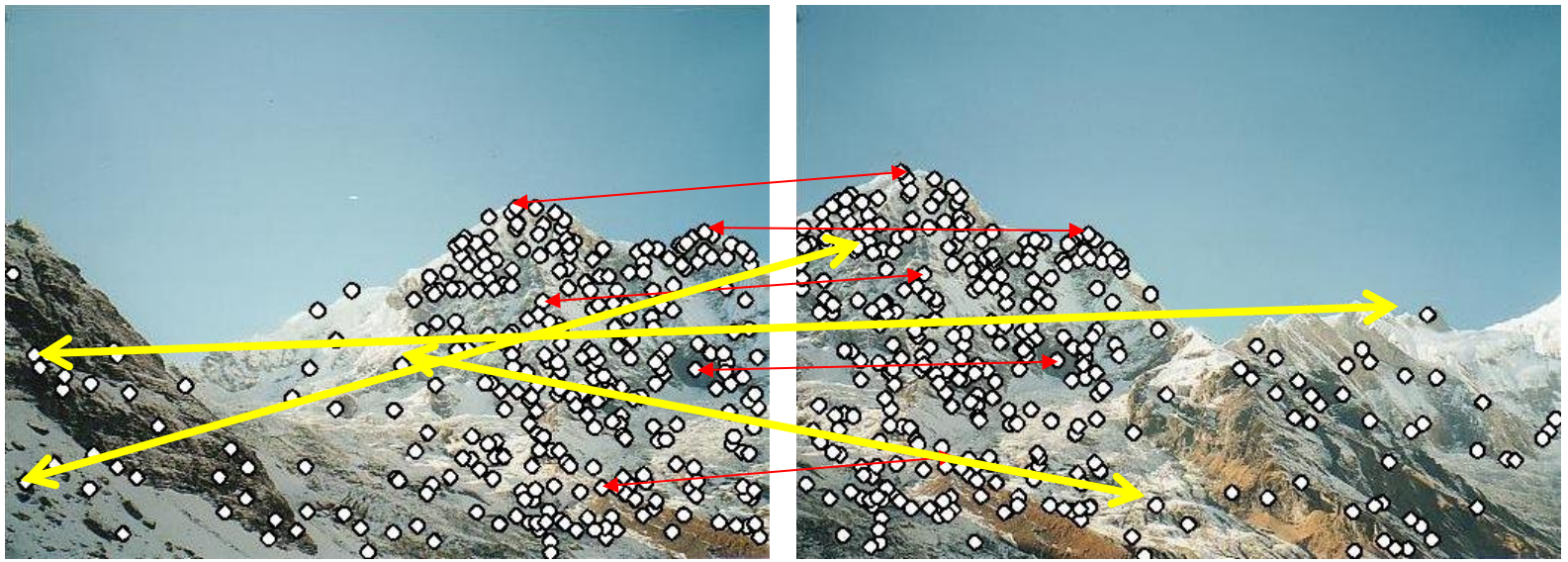


Image matching

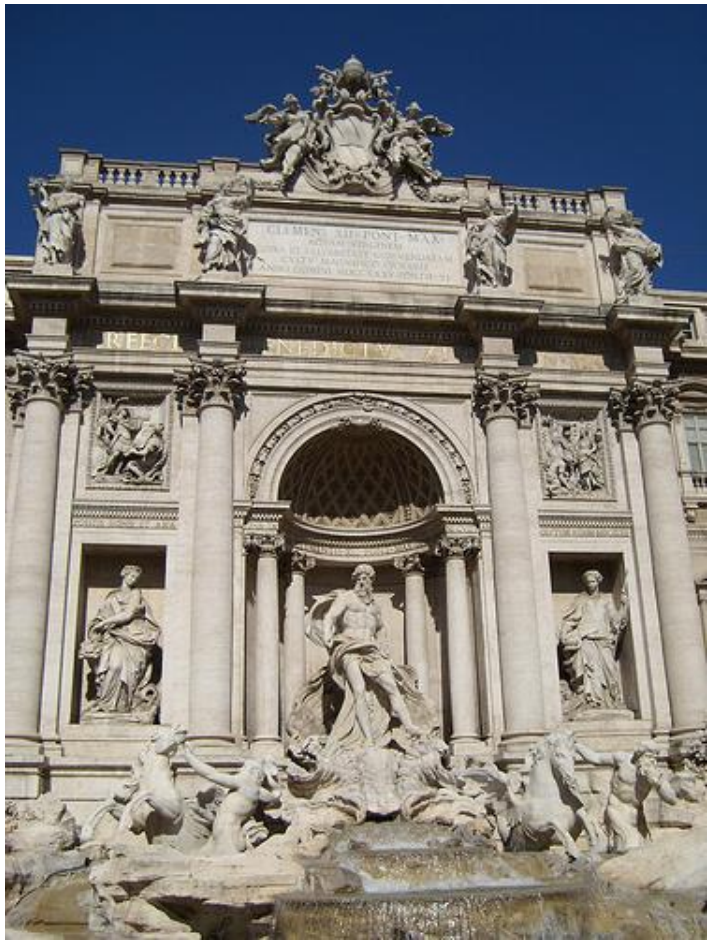
Flickr photos of Trevi Fountain in Rome.
Usernames from Flickr

Courtesy: S. Seitz and R. Szeliski

Diva Sian

Perspective view, lighting and scale are roughly similar.

(There's hope!)



swashford

Harder case

Diva Sian

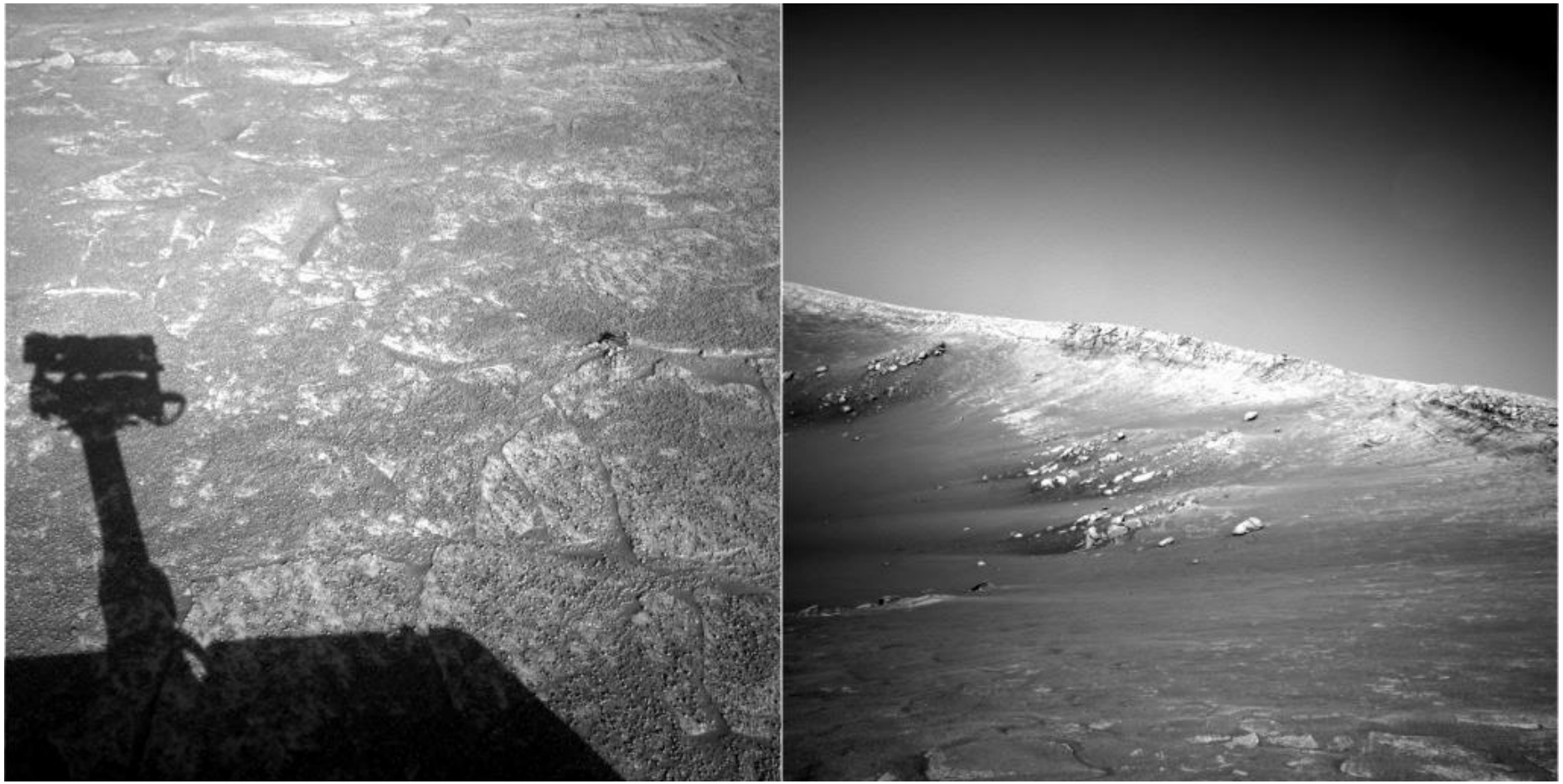
scgbt

Harder case because of lighting changes and perspective view point changes.

(Hmm, hopeful but with a twinge of doubt?)

Harder still?

Courtesy: S. Seitz and R. Szeliski



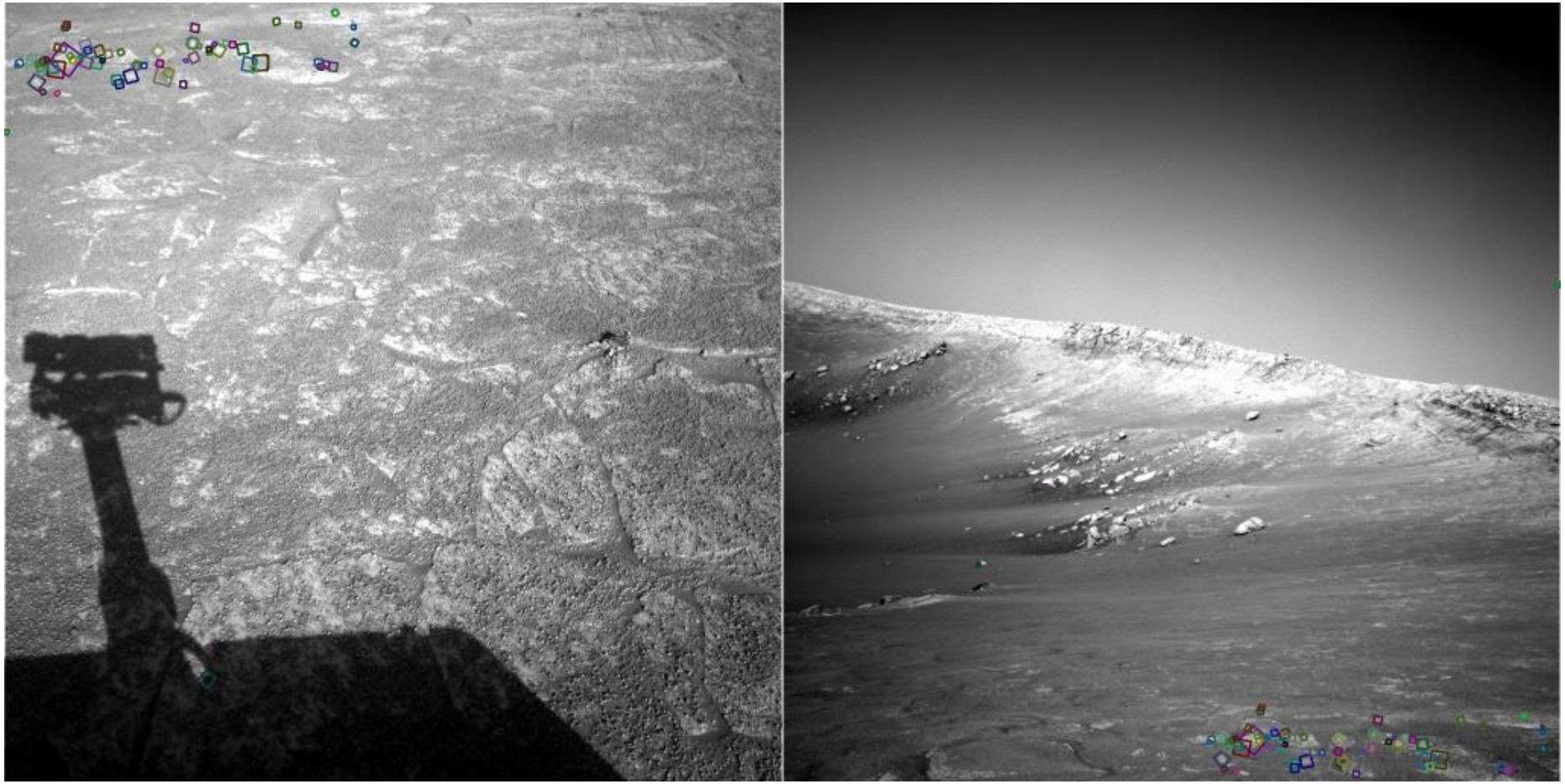
Low-overlap
Perspective
Scale
Ilumination

NASA Mars Rover images

(Really, are you serious?)

Answer below (look for tiny colored squares…)

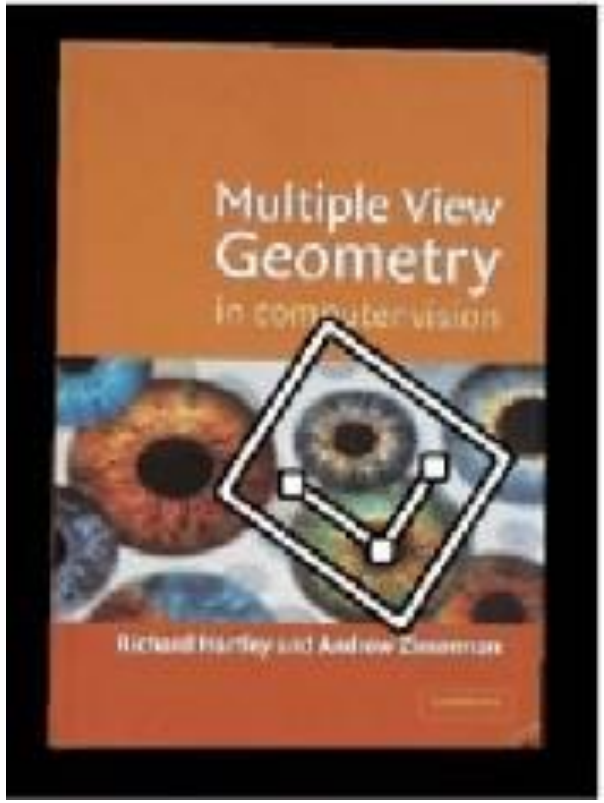
Courtesy: S. Seitz and R. Szeliski



NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

Courtesy: S. Seitz and R. Szeliski

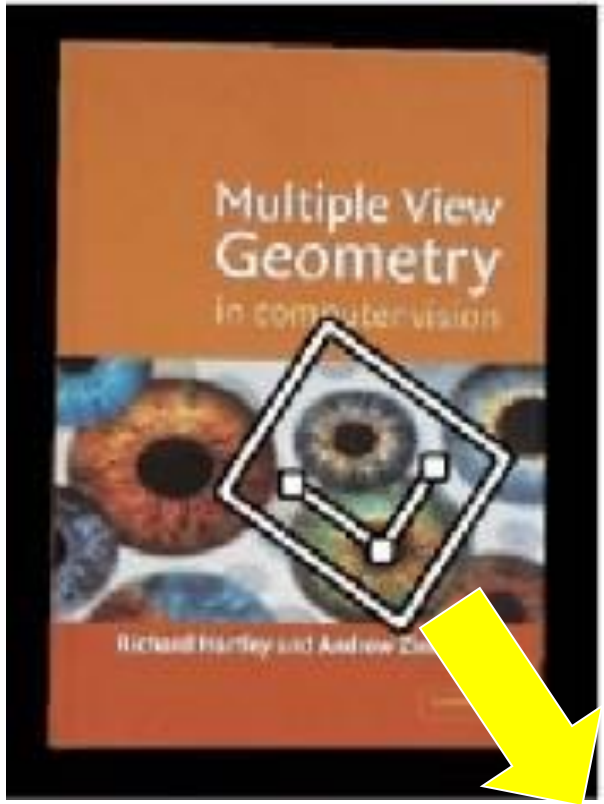
Image Matching

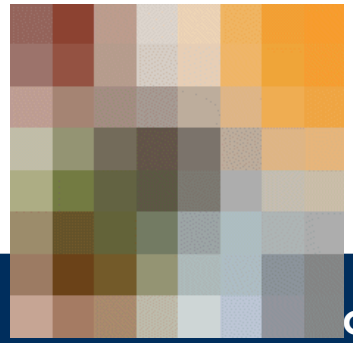


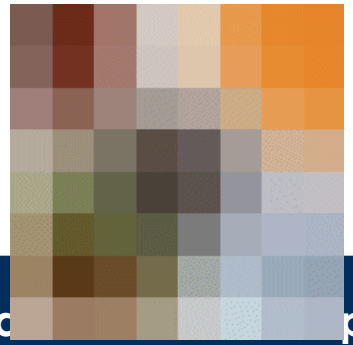
1) At an interesting point, let's define a coordinate system (x,y axis)
2) Use the coordinate system to pull out a patch at that point

Image Matching

Courtesy: S. Seitz and R. Szeliski







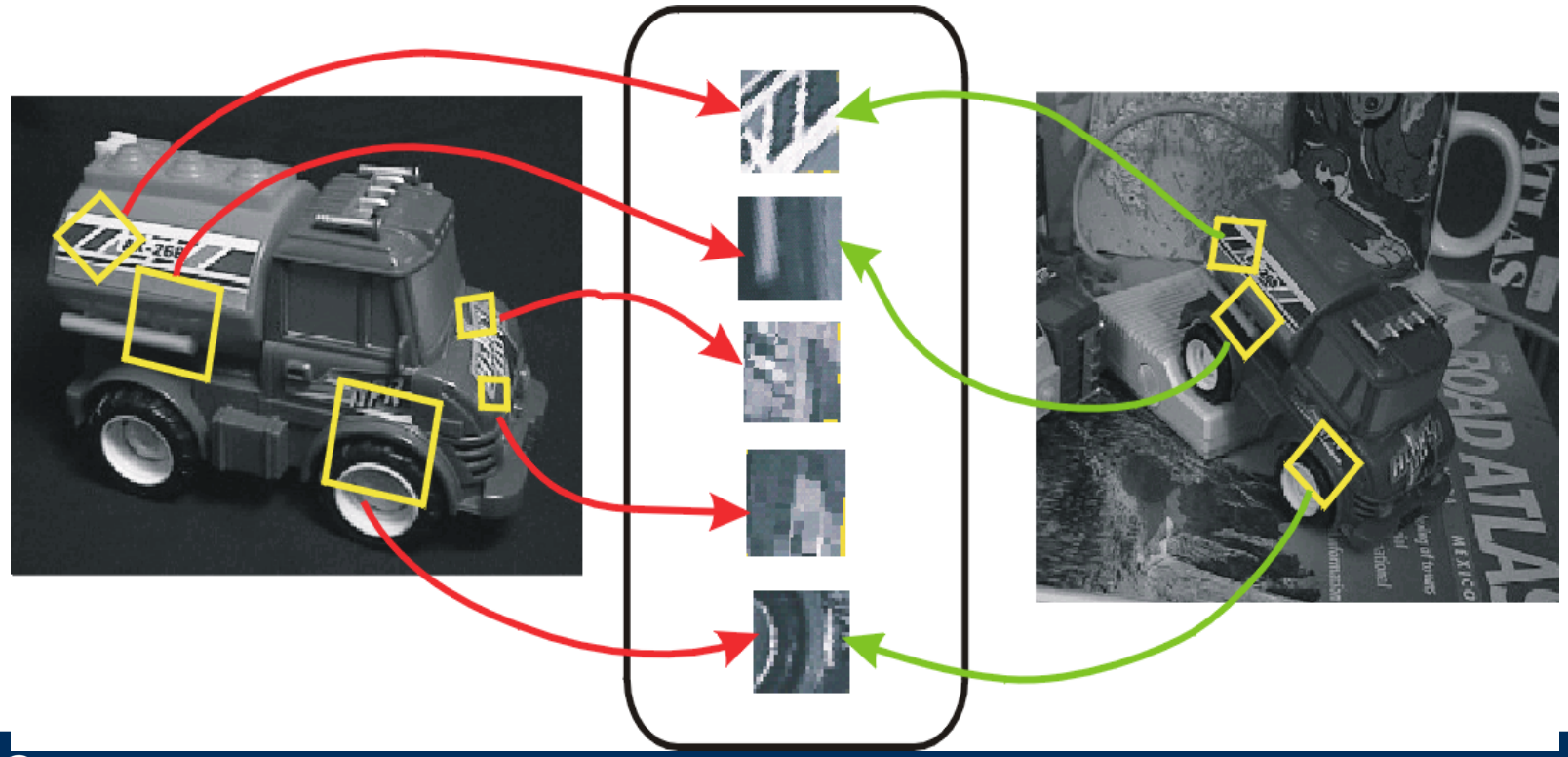
Invariant local features

Courtesy: S. Seitz and R. Szeliski

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

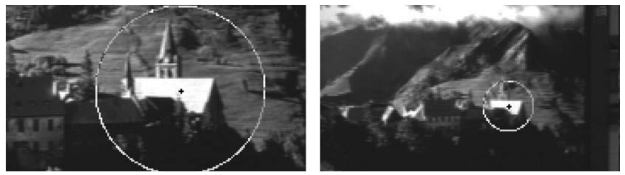
-Buzzword is “invariance”

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



Robust image patch matching

- 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

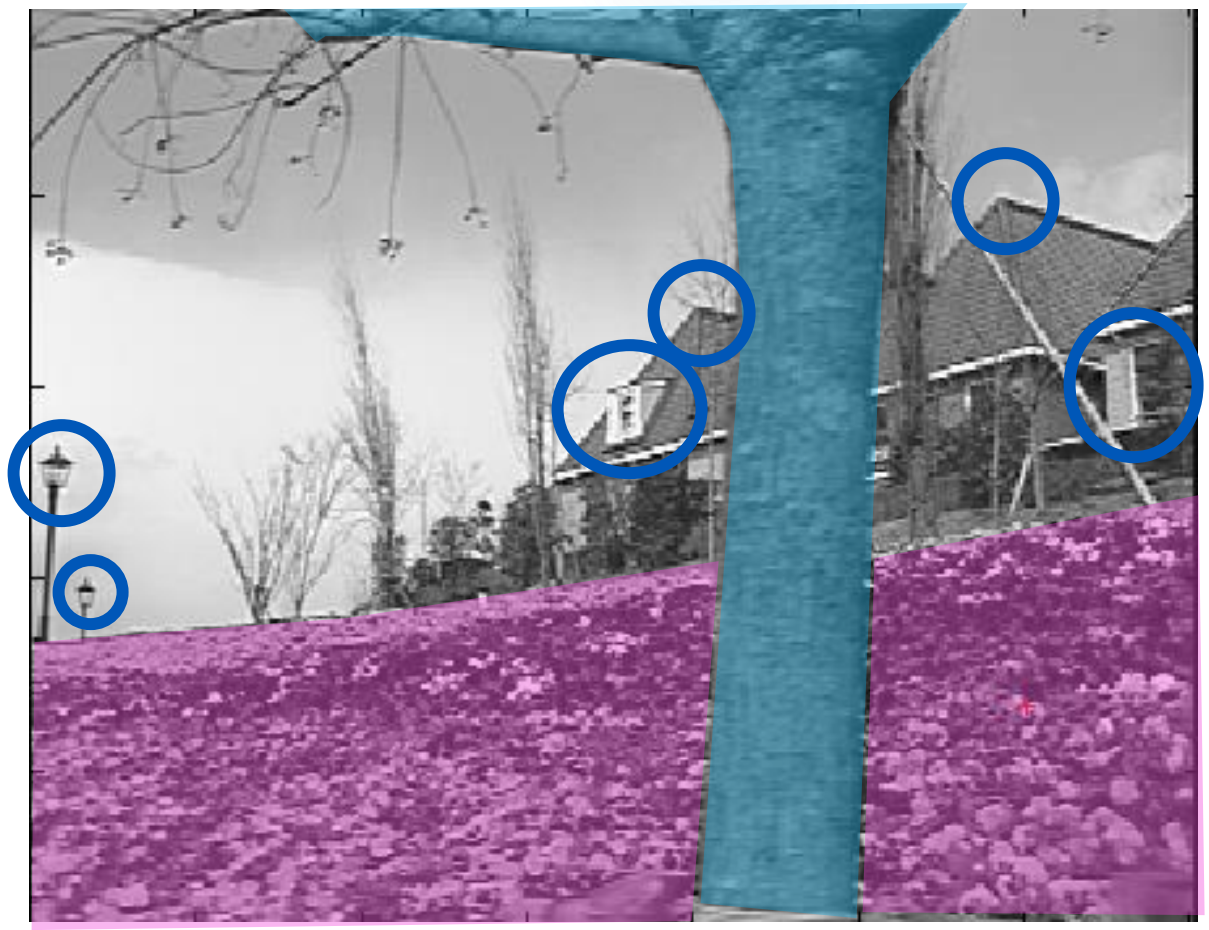
  - Brightness invariance
    - Robust to minor changes in illumination

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

What makes a good feature?

Courtesy: S. Seitz and R. Szeliski

- 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)?

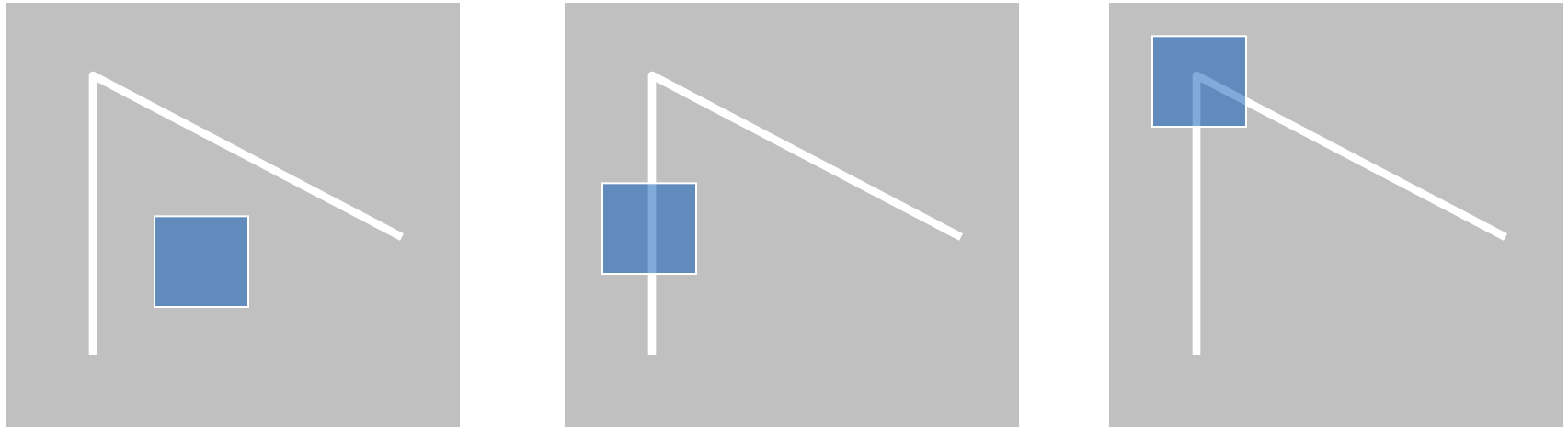
Want uniqueness

- Look for image regions that are unusual
  - Lead to unambiguous matches in other images

- How to define “unusual”?

Local measures of uniqueness

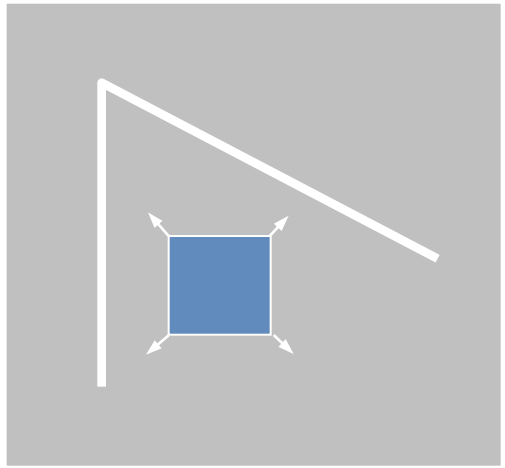
- Suppose we only consider a small window of pixels
  - What defines whether a feature is a good or bad candidate?



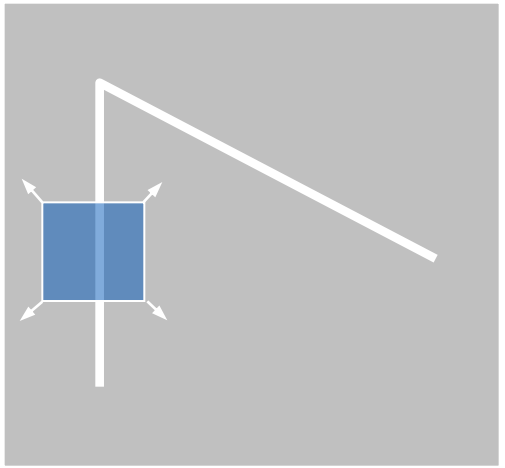
Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

Feature detection

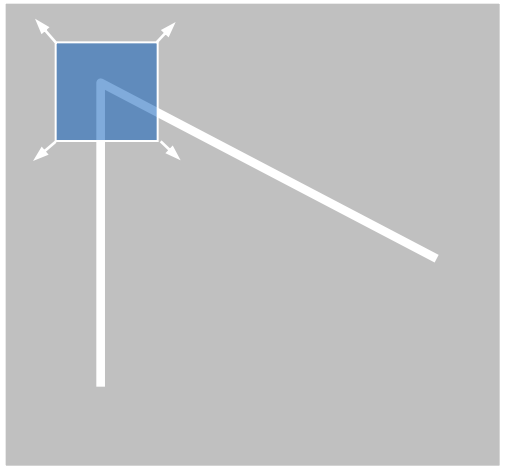
- Local measure of feature uniqueness
  - How does the window change when you shift it?
  - Shifting the window in any direction causes a big change



"flat" region:
no change in all directions



"edge"
no change along the edge direction

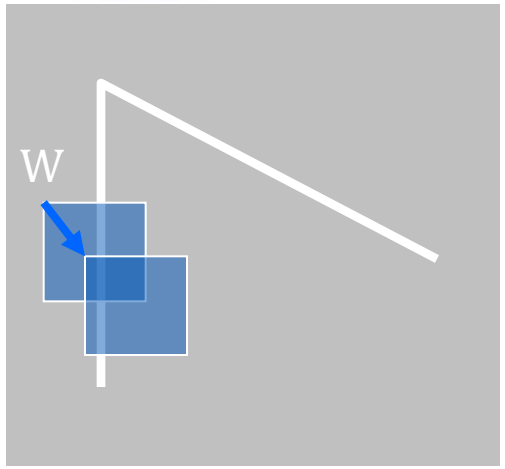


"corner"
significant change in all directions

Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

Feature detection: the math

- Consider shifting the window W by (u,v)
  - how do the pixels in W change?
  - compare each pixel ***before*** and ***after*** by summing up the squared differences (SSD)
  - this defines a SSD "error" of $E(u,v)$:



$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

Harris Corner: Summary

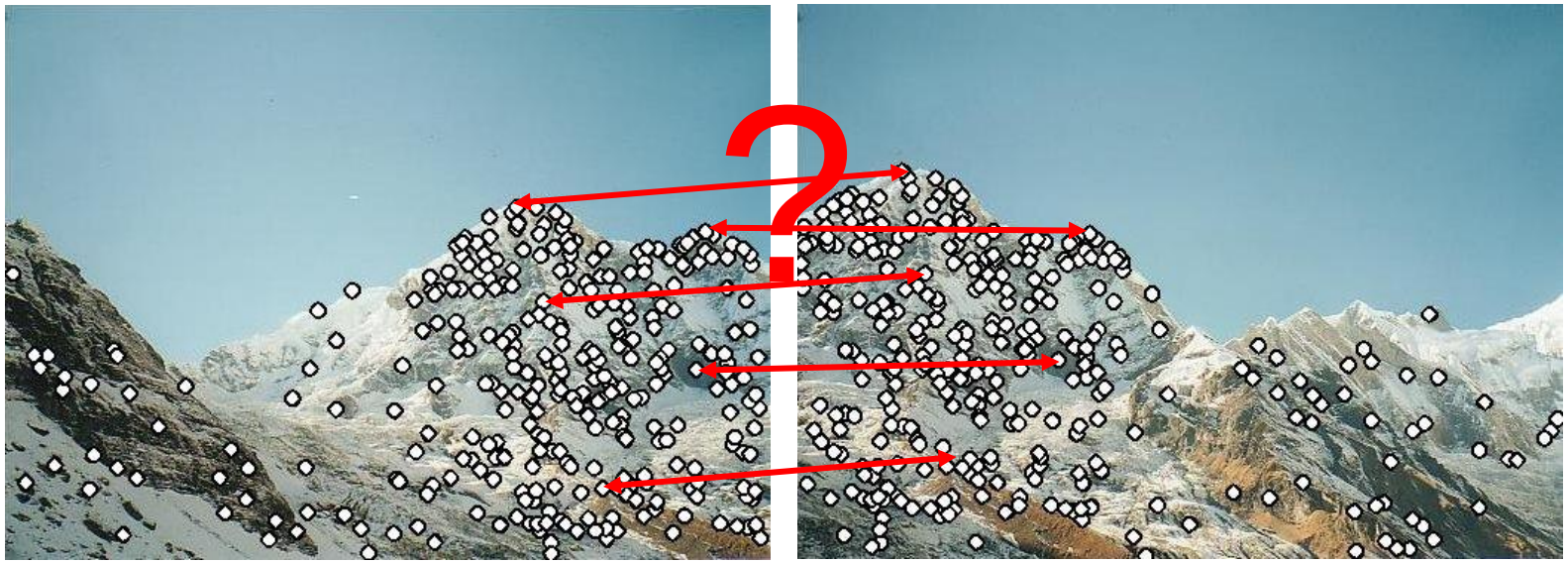
- Principled method of detecting corners
- Look for image patches whose gradients span the whole space
- With simple modifications, scale invariant, isotropic, fast.

Corners

- Now that we have corners, what can we do with them?
  - Landmarks for SLAM
  - Use them to track moving objects
  - Find registration marks on calibration targets

Feature descriptors

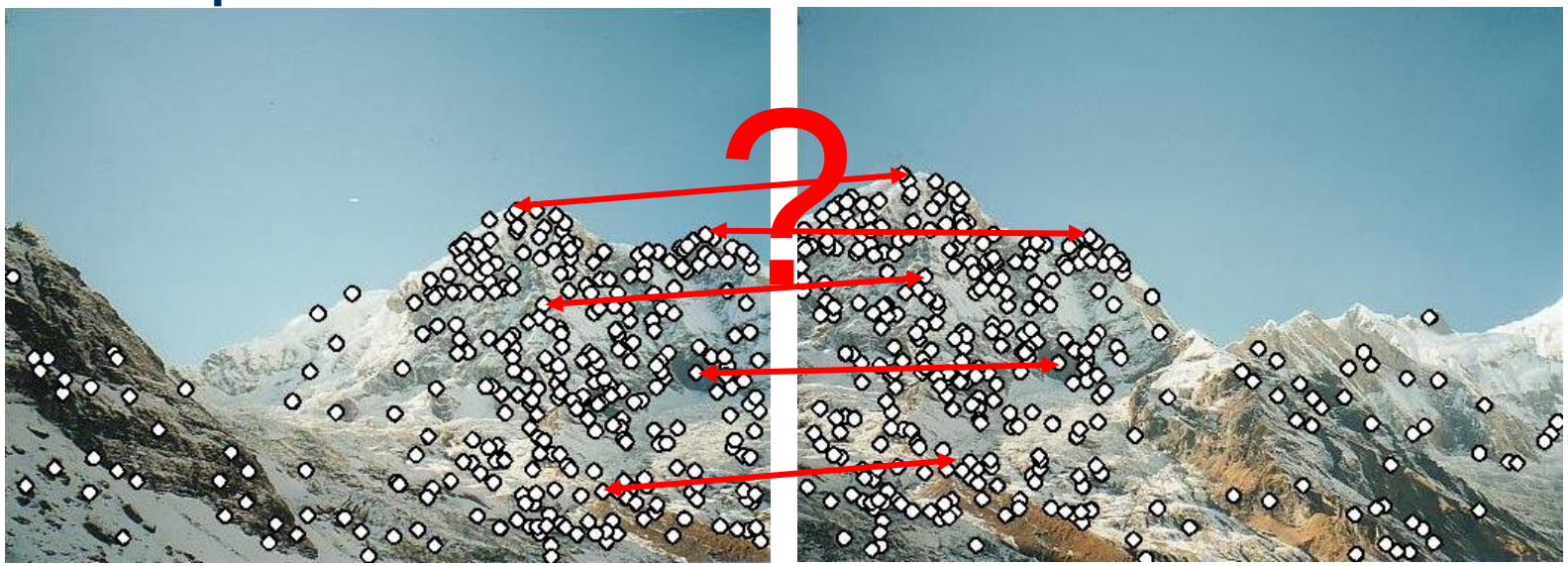
We now know how to detect good points
Next question: **How to match them?**



Feature descriptors

We now know how to detect good points

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



Lots of possibilities (this is a popular research area)

- Simple option: match square windows around the point
- Commonly used approach: SIFT
  - David Lowe, UBC http://www.cs.ubc.ca/~lowe/keypoints/

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 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
   - Harris is invariant to translation and rotation
   - Scale is trickier
     - common approach is to detect features at many scales using a Gaussian pyramid (e.g., MOPS)
     - More sophisticated methods find “the best scale” to represent each feature (e.g., SIFT)
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…

Image patch matching

- Extract regions from images, compute sum of absolute/squared (SAD/SSE) differences
  - These will have a fairly small error


  - These will have a large error: may not match


Rotation invariance for feature descriptors

Find dominant orientation of the image patch

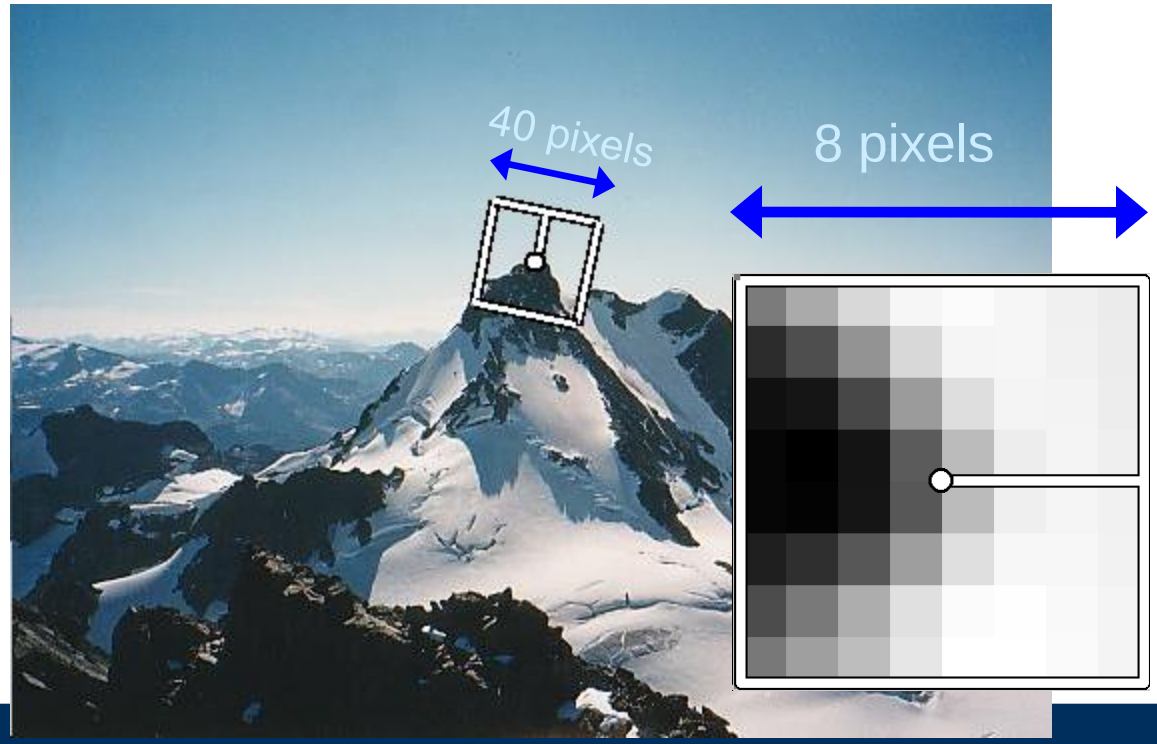
- This is given by $\mathbf{x}_+$, the eigenvector of H corresponding to λ_+
  - λ_+ is the *larger* eigenvalue
- Rotate the patch according to this angle

Figure by Matthew Brown

Multiscale Oriented PatcheS descriptor

Take 40x40 square window around detected feature

- Scale to 1/5 size (using prefiltering)
- Rotate to horizontal
- Sample 8£8 square window centered at feature
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window (both window $\mathbf{I}$ and $a\mathbf{I}+b$ will match)



Adapted from slide by Matthew Brown

Detections at multiple scales

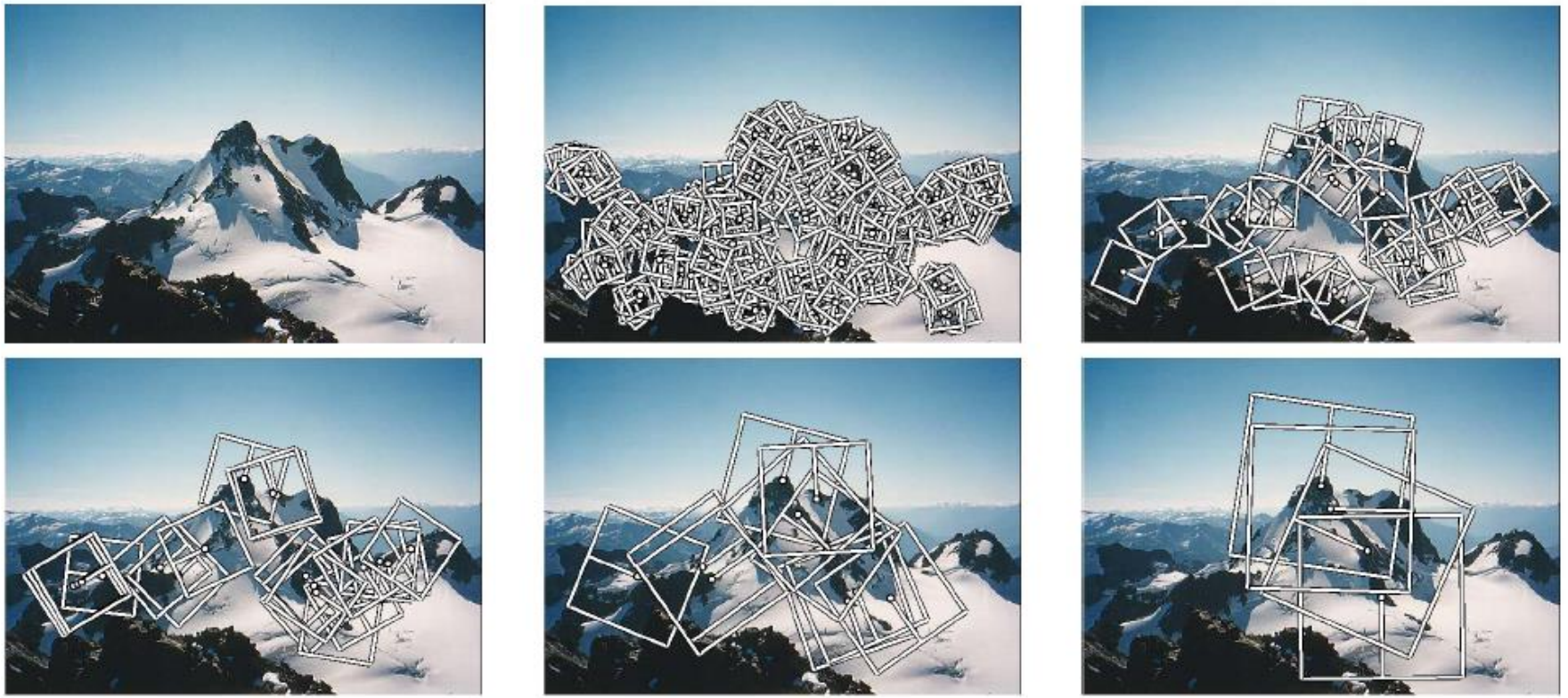
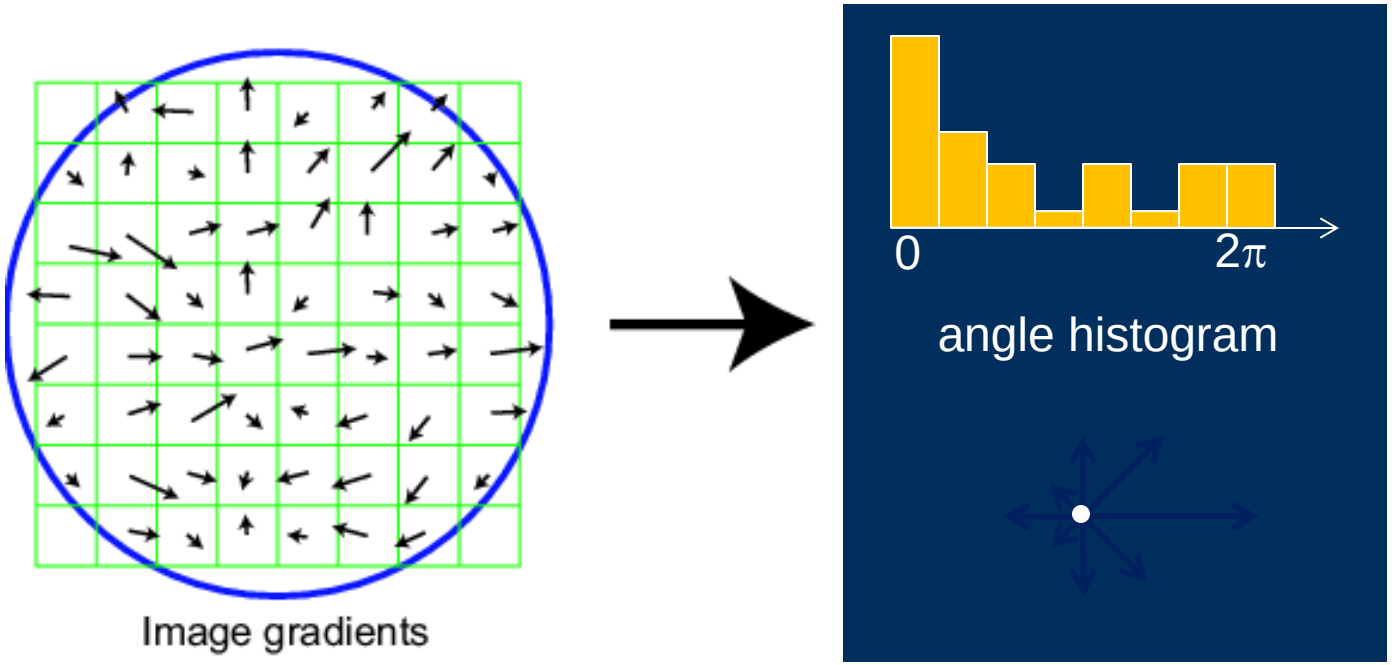


Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

Scale Invariant Feature Transform

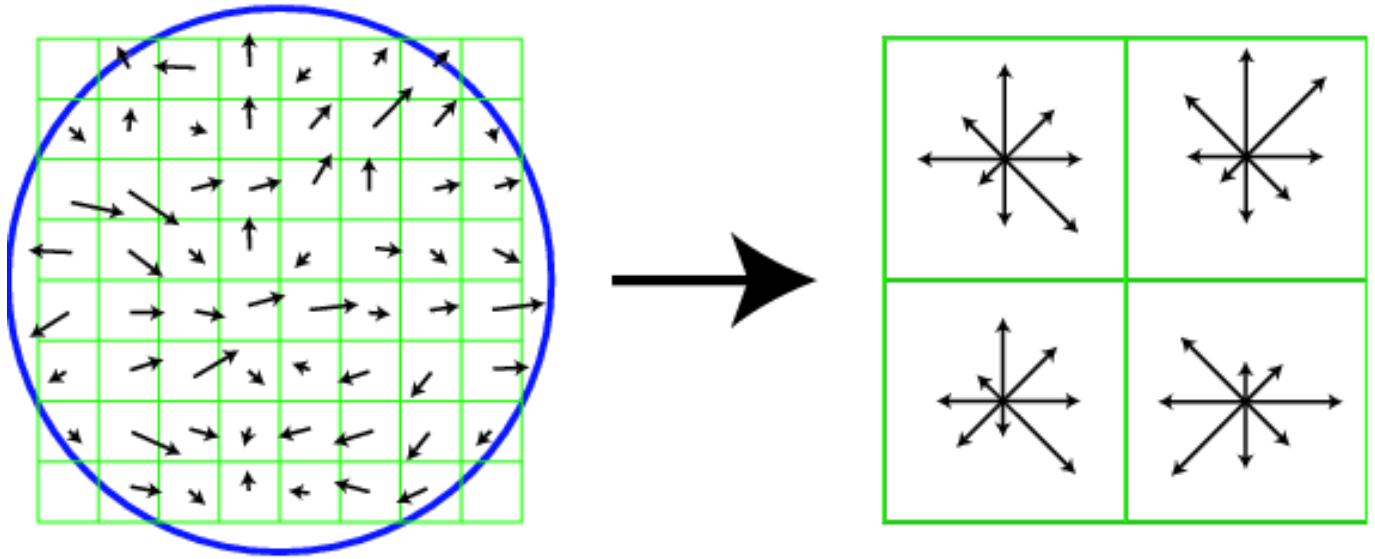
- Algorithm outline:
  - Detect interest points (aka corners)
  - For each interest point
    - Determine dominant orientation
    - Build histograms of gradient directions
    - Output feature *descriptor*

Scale Invariant Feature Transform



Adapted from slide by David Lowe

SIFT 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/

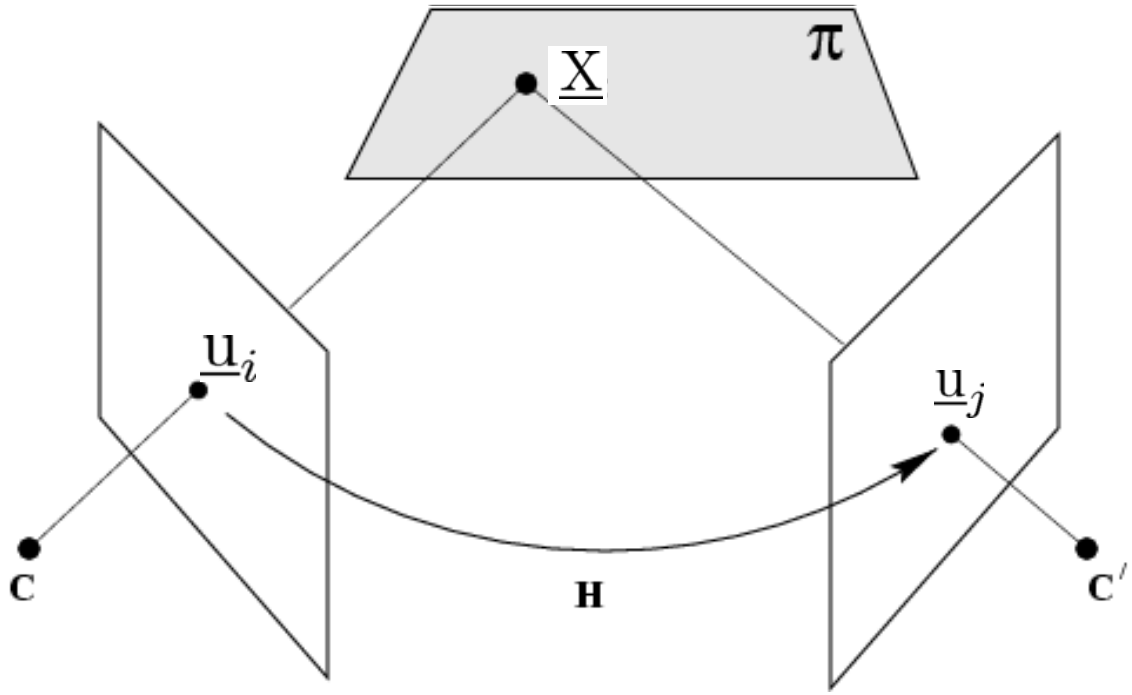
Cameras Overview

- Camera Projection Models
  - Intrinsics
  - Extrinsics
  - Lens Distortion
- Image Feature Detection
  - Harris Corners
  - SIFT Features
- Image Registration Models
  - Homography
  - Essential Matrix
  - Fundamental Matrix

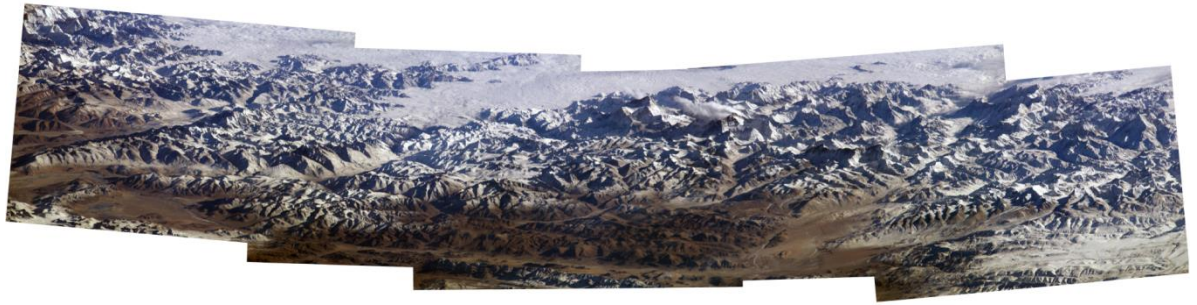
2D Planar Homographies

- Homography, H
  - Planar structure
  - 8-DOF (in general)

$$\underline{u}_j = \mathrm{H}\underline{u}_i$$



- Images of a strictly planar scene
- Far away objects (single dominant plane)



- Purely rotating camera

Planar Homography Example

World Plane

Center view (actual image)

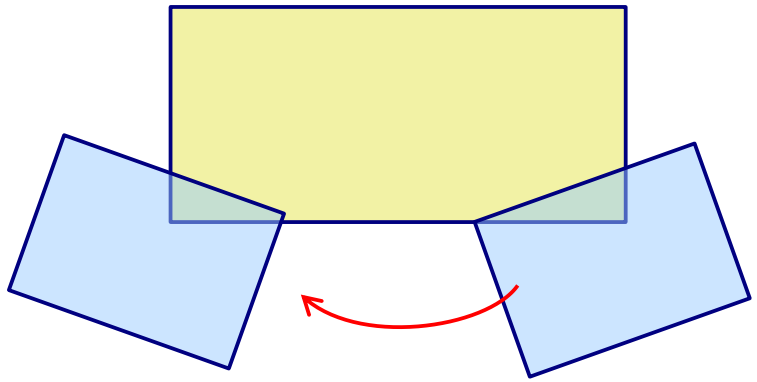
Images

Left view (actual image)

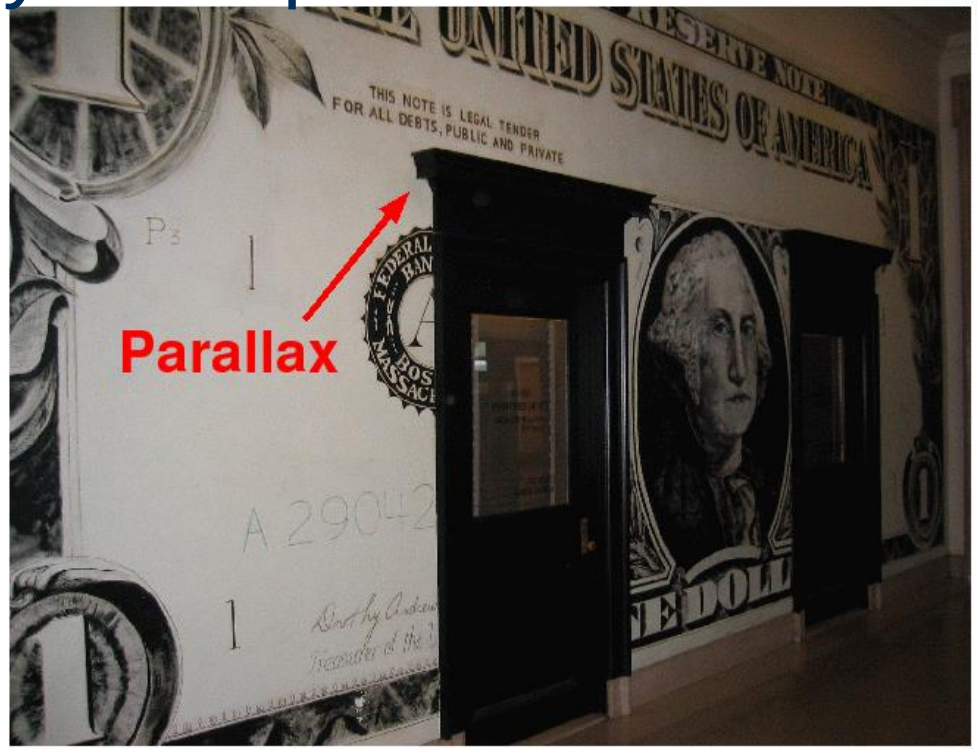
Right view (actual image)

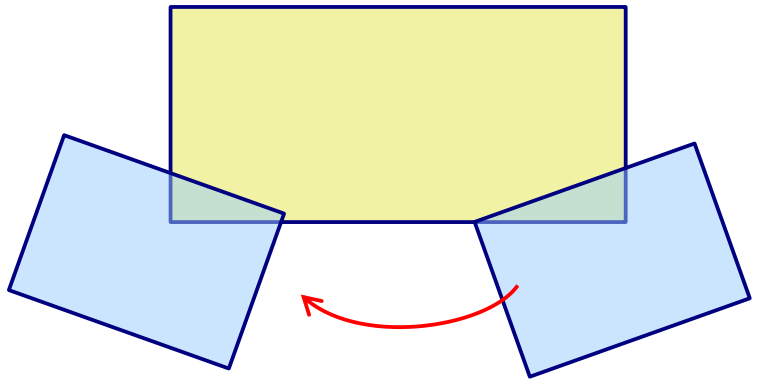
Planar Homography Example

Right to left synthetic view via a homography



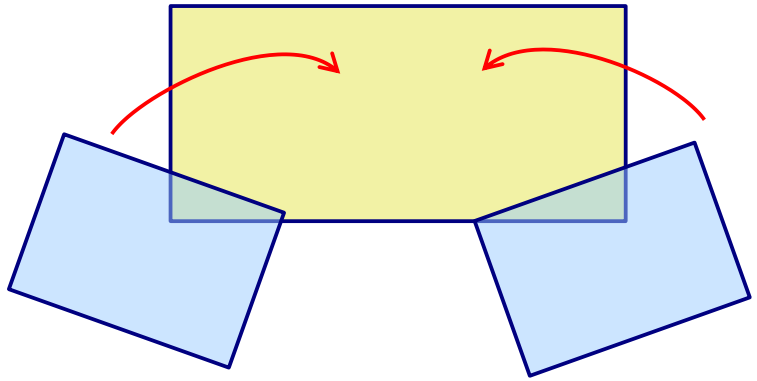
Planar Homography Example





Planar Homography Example

Right to center and left to center synthetic views via their respective homographies



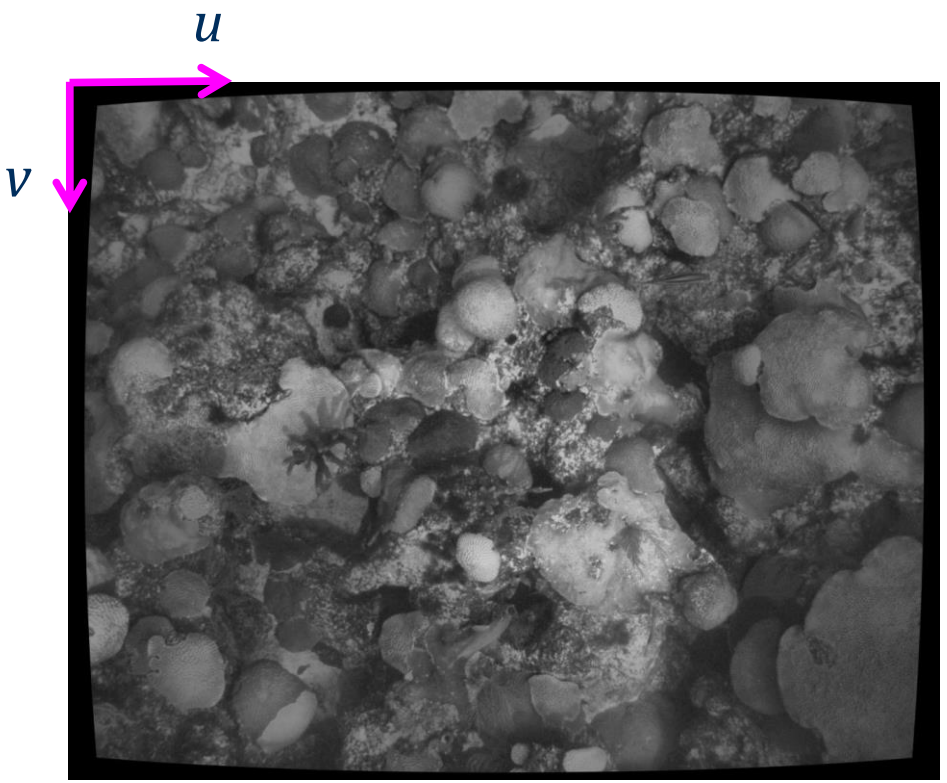
Pure Rotation Homography Example

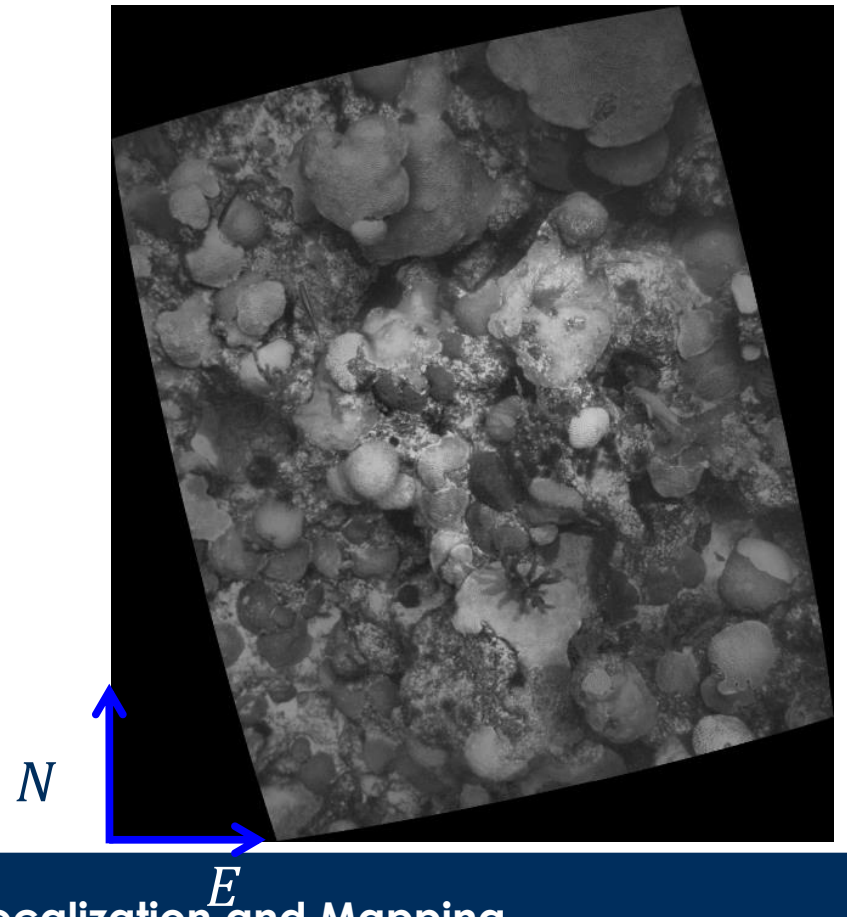
- Warp image to a canonical North/East/Down view based upon measured attitude

$$\mathrm{H}_\infty = \mathrm{K}\mathrm{R}\mathrm{K}^{-1}$$

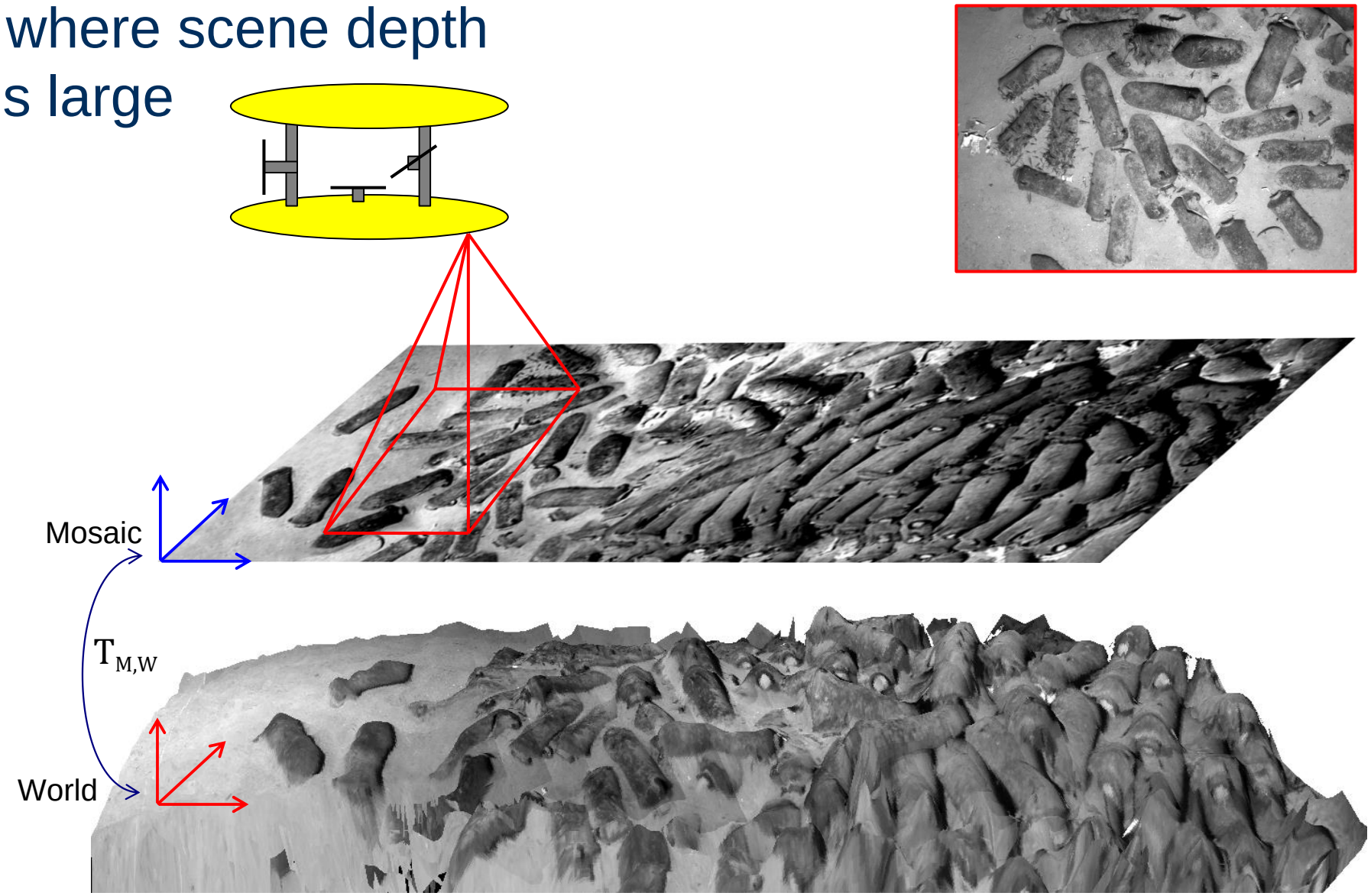
camera calibration matrix

rotation matrix

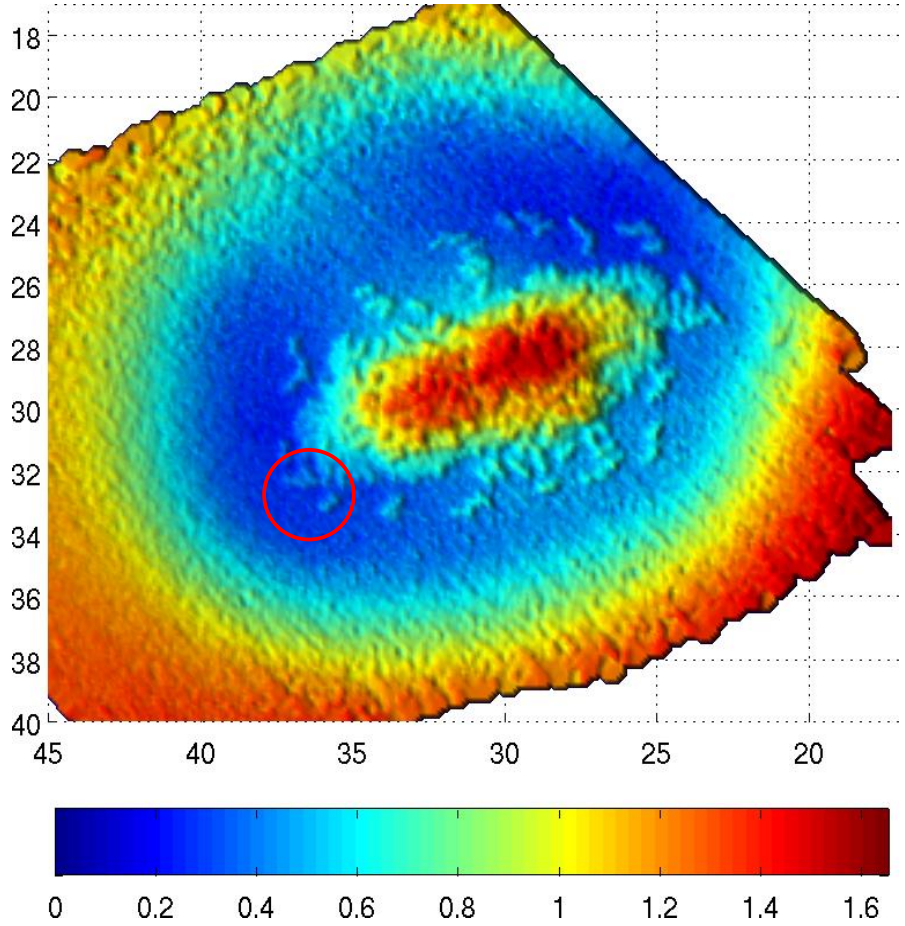


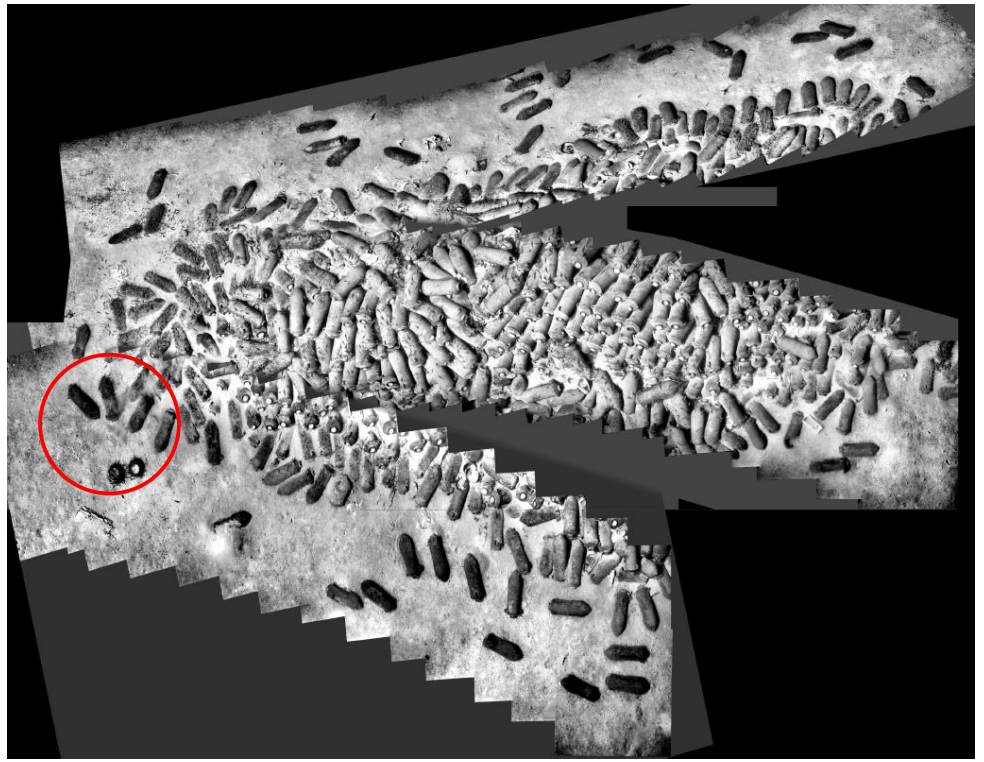


Homography does not model imagery where scene depth change is large



Failure of Planar Seafloor Assumption



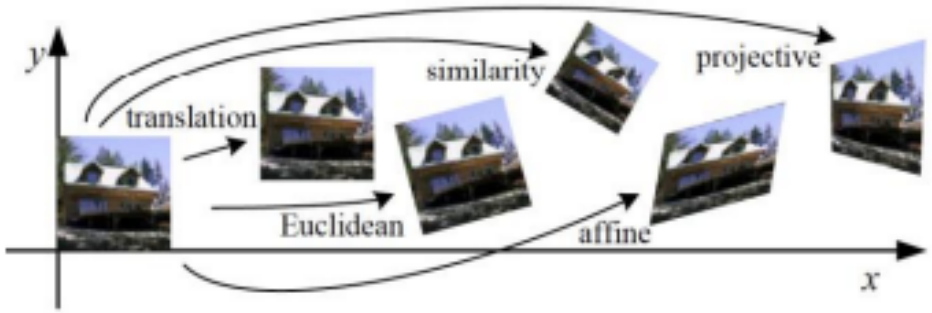


2D transformation models

- Similarity (translation, scale, rotation)

- Affine

- Projective (homography)



Group	Matrix	Distortion	Invariant properties
Projective 8 dof	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$		Concurrency, collinearity, **order of contact**: intersection (1 pt contact); tangency (2 pt contact); inflections (3 pt contact with line); tangent discontinuities and cusps. cross ratio (ratio of ratio of lengths).
Affine 6 dof	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Parallelism, ratio of areas, ratio of lengths on collinear or parallel lines (e.g. midpoints), linear combinations of vectors (e.g. centroids). The line at infinity, $\mathbf{l}_\infty$.
Similarity 4 dof	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Ratio of lengths, angle. The circular points, $\mathbf{I}, \mathbf{J}$ (see section 2.7.3).
Euclidean 3 dof	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Length, area

Table 2.1. **Geometric properties invariant to commonly occurring *planar* transformations.** *The matrix $\mathtt{A} = [a_{ij}]$ is an invertible 2×2 matrix, $\mathtt{R} = [r_{ij}]$ is a 2D rotation matrix, and (t_x, t_y) a 2D translation. The distortion column shows typical effects of the transformations on a square. Transformations higher in the table can produce all the actions of the ones below. These range from Euclidean, where only translations and rotations occur, to projective where the square can be transformed to any arbitrary quadrilateral (provided no three points are collinear).*

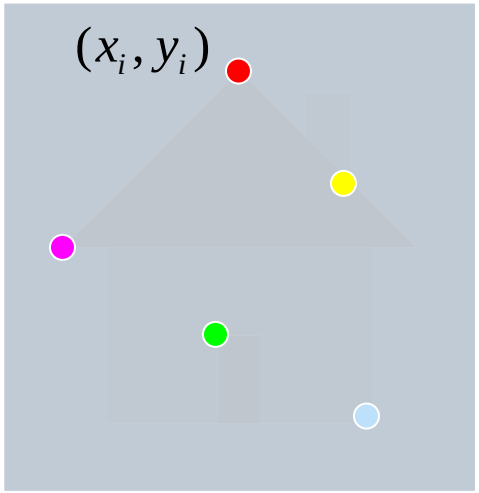
Cite: Cambridge University Press 978-0-521-54051-3 - Multiple View Geometry in Computer Vision: Second Edition Richard Hartley and Andrew Zisserman Frontmatter

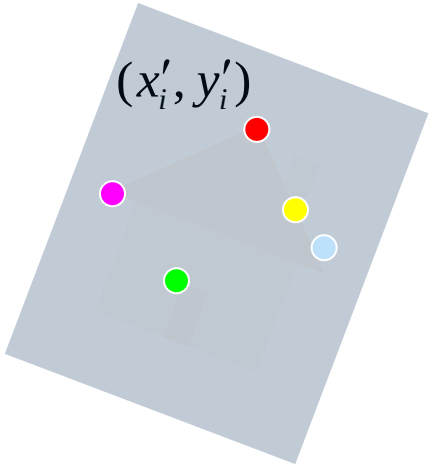
Let’s start with affine transformations

- Simple fitting procedure (linear least squares)
- Approximates viewpoint changes for roughly planar objects and roughly orthographic cameras
- Can be used to initialize fitting for more complex models

Fitting an affine transformation

- Assume we know the correspondences, how do we get the transformation?





$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = \begin{bmatrix} m_1 & m_2 & t_1 \\ m_3 & m_4 & t_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} & \cdots & & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & \cdots & & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x'_i \\ y'_i \\ \cdots \end{bmatrix}$$

Fitting an affine transformation

$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

- Linear system with six unknowns
- Each match gives us two linearly independent equations: need at least three to solve for the transformation parameters
- Can solve $\mathbf{Ax}=\mathbf{b}$ using pseduo-inverse:
  $\mathbf{x} = (\mathrm{A}^{\mathrm{T}}\mathrm{A})^{-1}\mathrm{A}^{\mathrm{T}}\mathbf{b}$; or $\mathbf{x} = \mathrm{pinv}(\mathrm{A})*\mathbf{b}$; in Matlab

Dealing with outliers

- The set of putative matches still contains a very high percentage of outliers
- How do we fit a geometric transformation to a small subset of all possible matches?
- Possible strategies:
  - RANSAC
  - Incremental alignment
  - Hough transform

Strategy: RANSAC

- RANSAC loop:

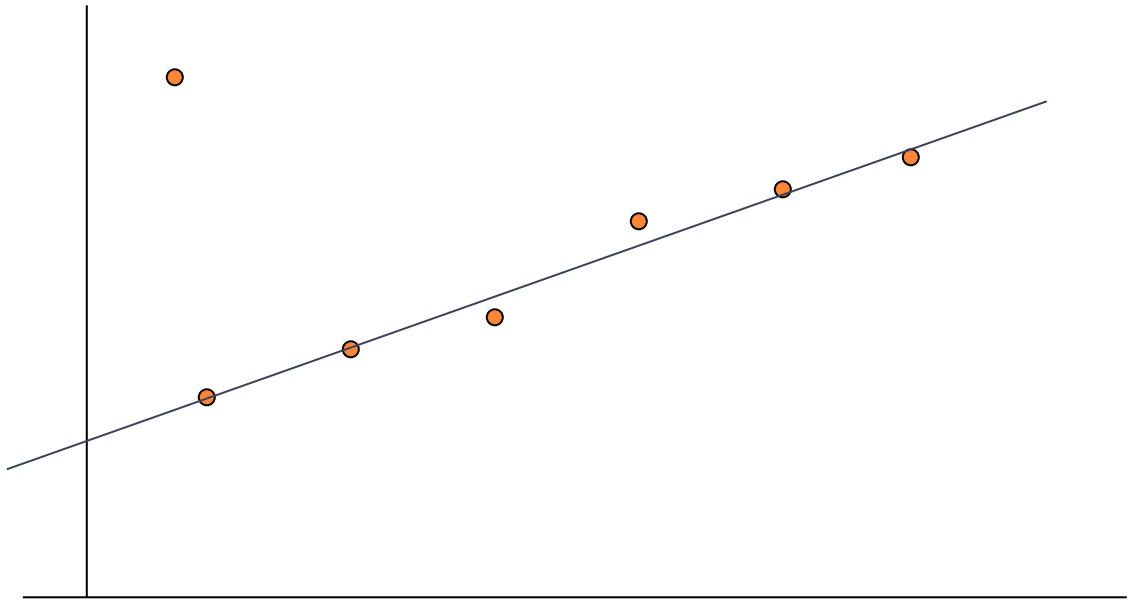
1. Randomly select a *seed group* of matches
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers

- 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

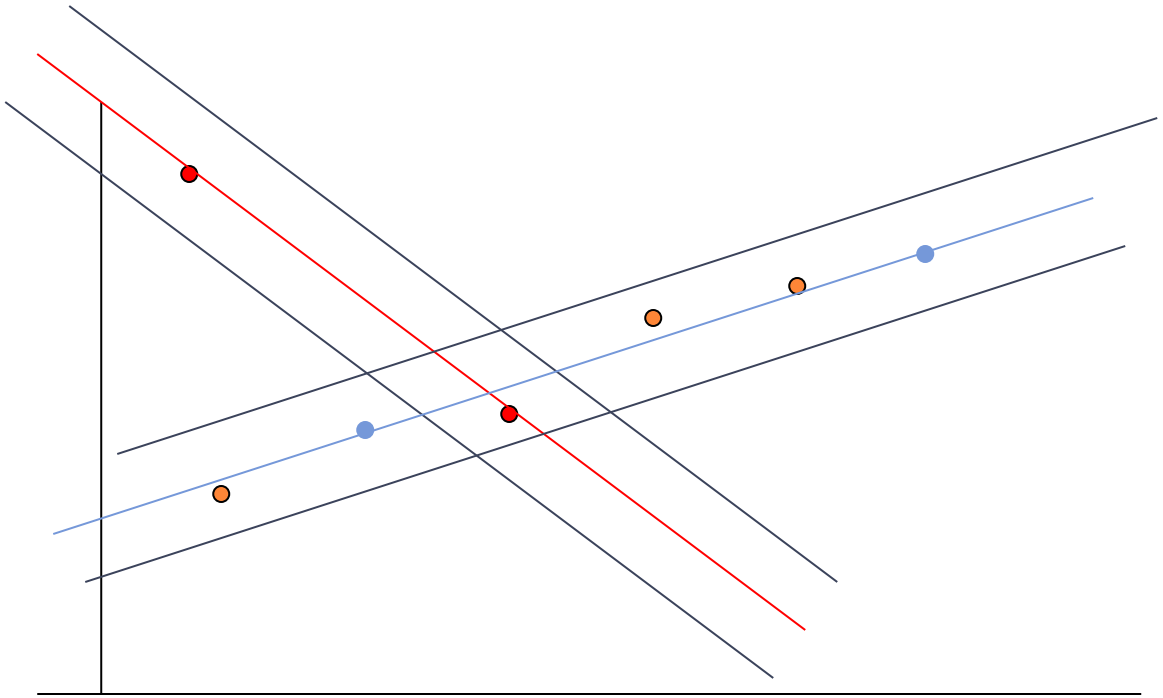
- Fitting a straight line



Main Idea

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

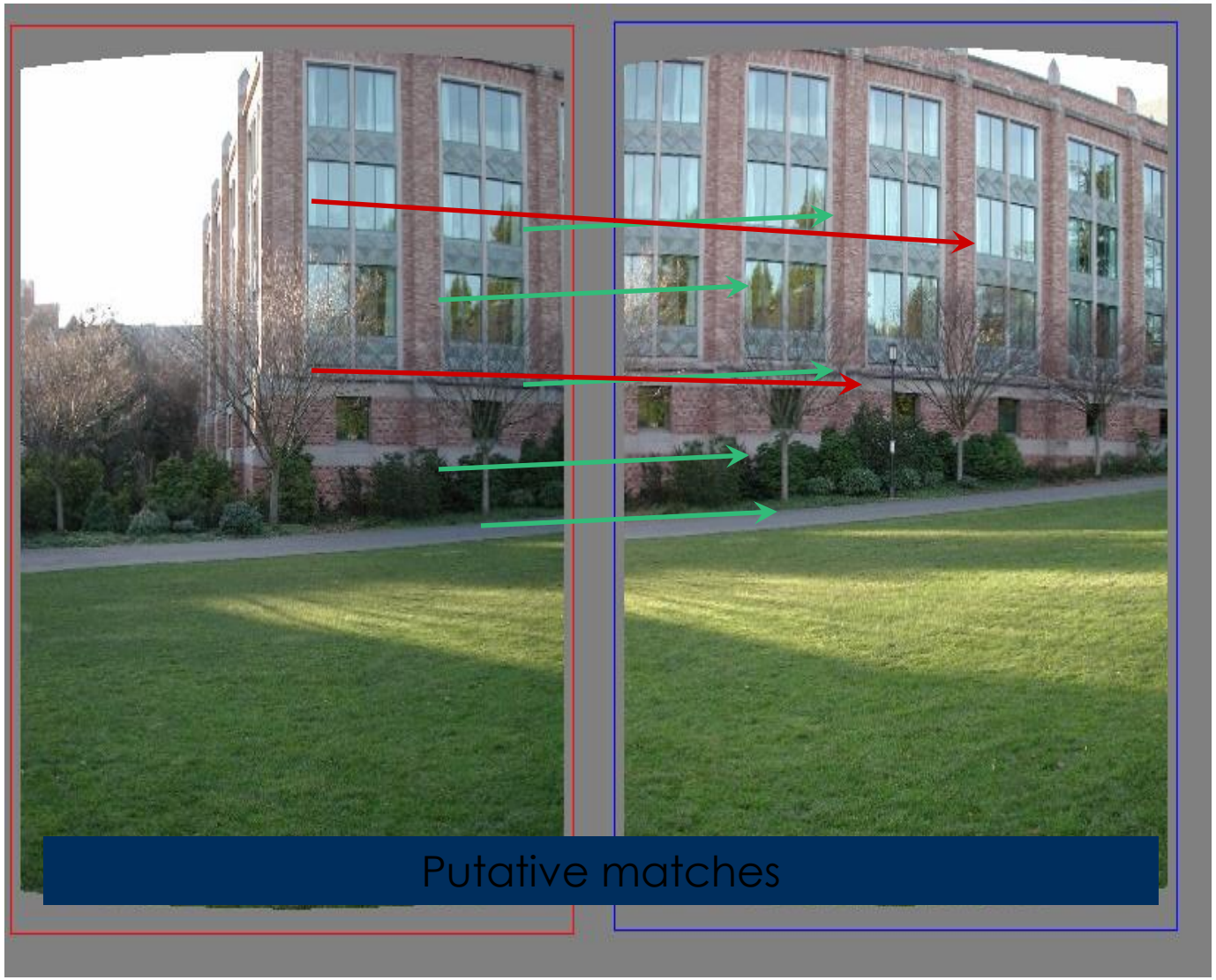
Why will this work ?



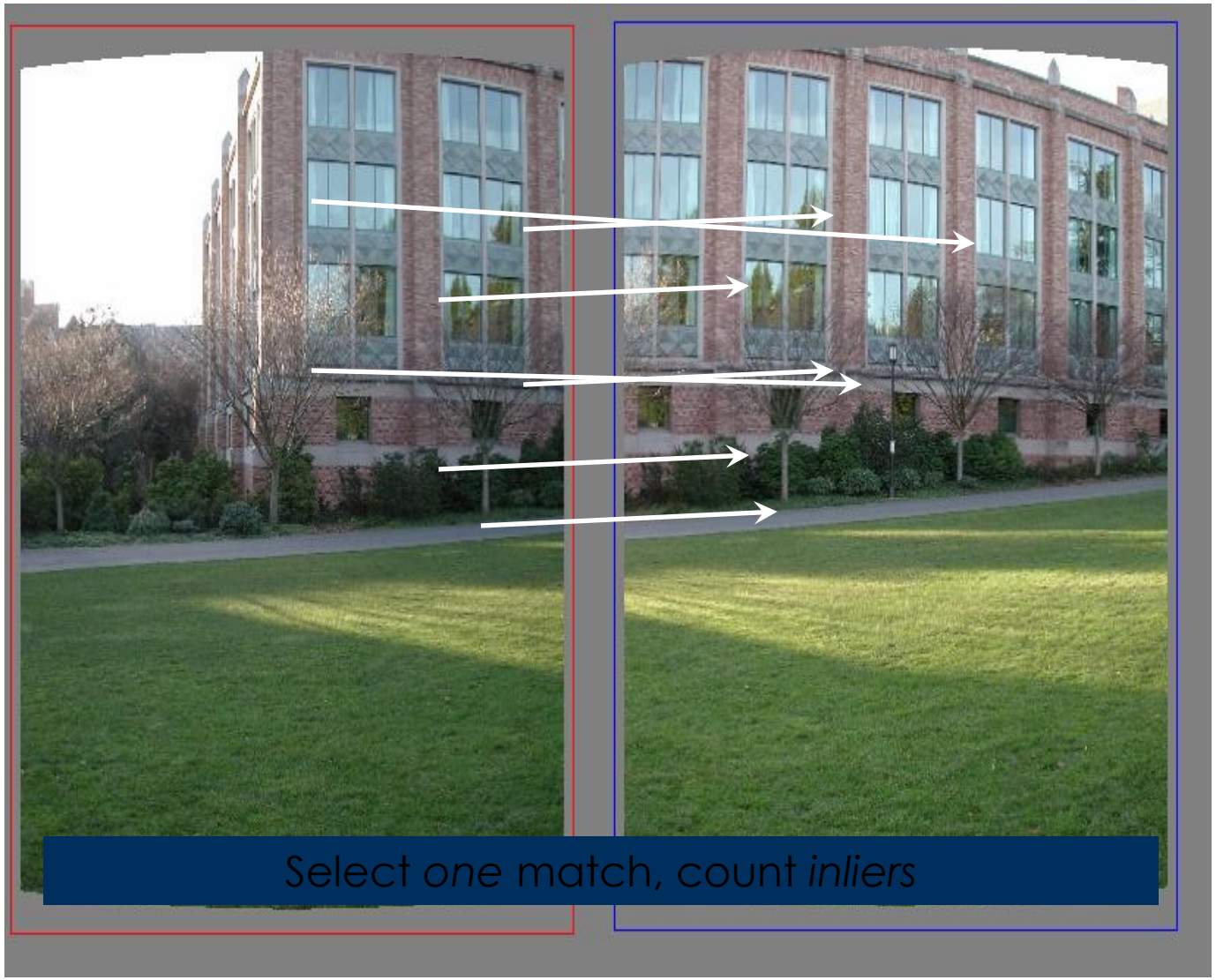
Best Line has most support

- More support ! better fit

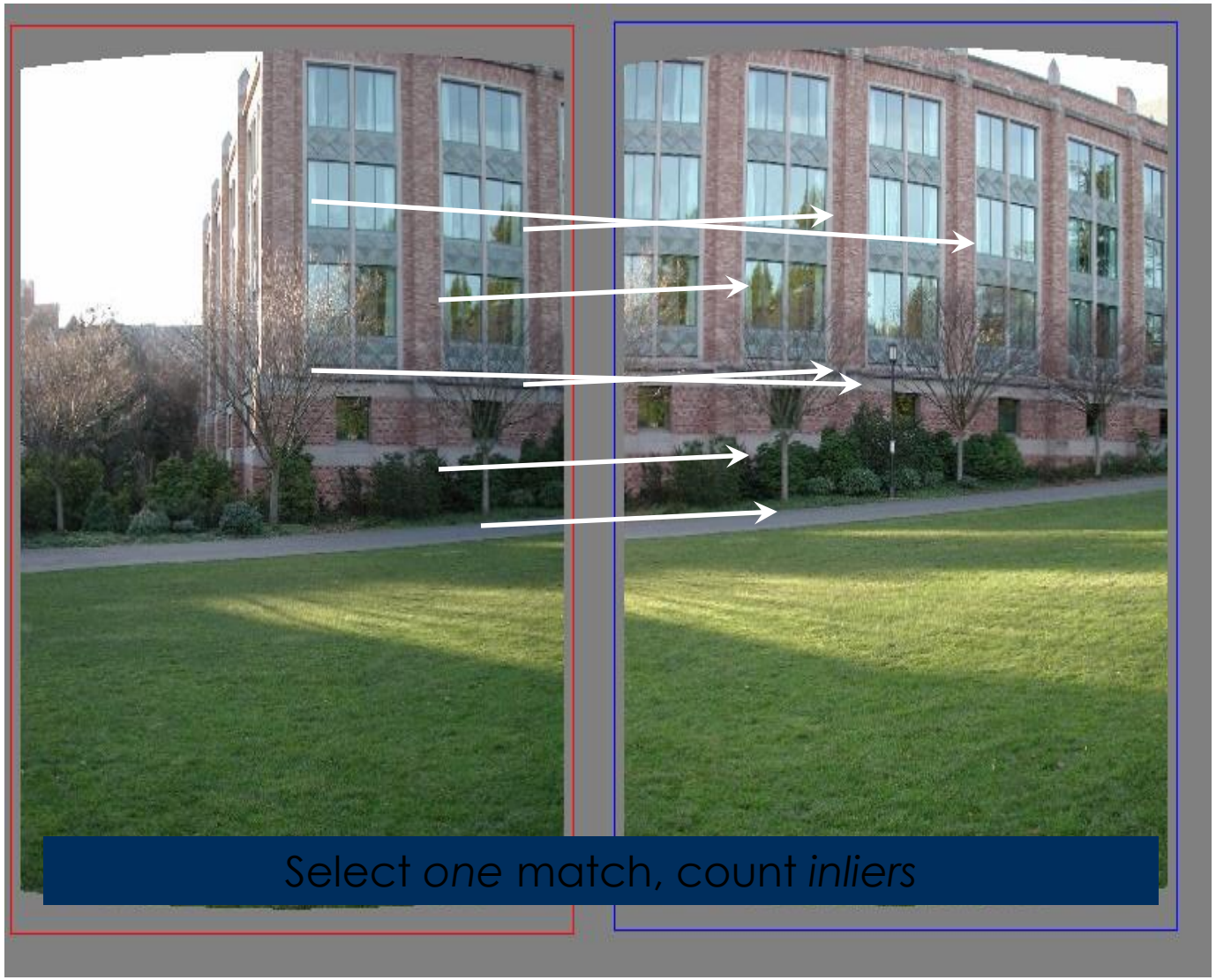
RANSAC example: Translation



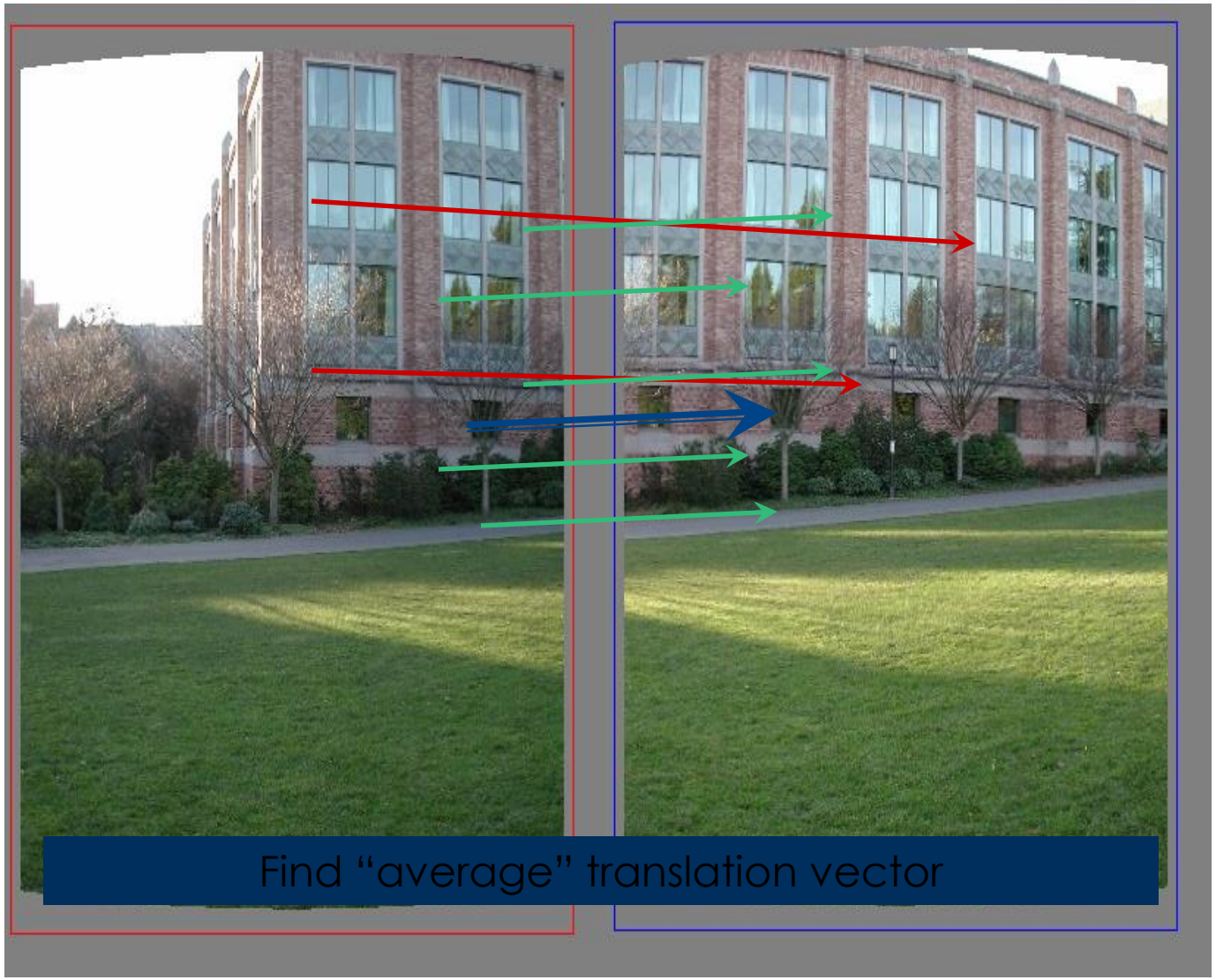
RANSAC example: Translation



RANSAC example: Translation



RANSAC example: Translation



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 $|S_i|$

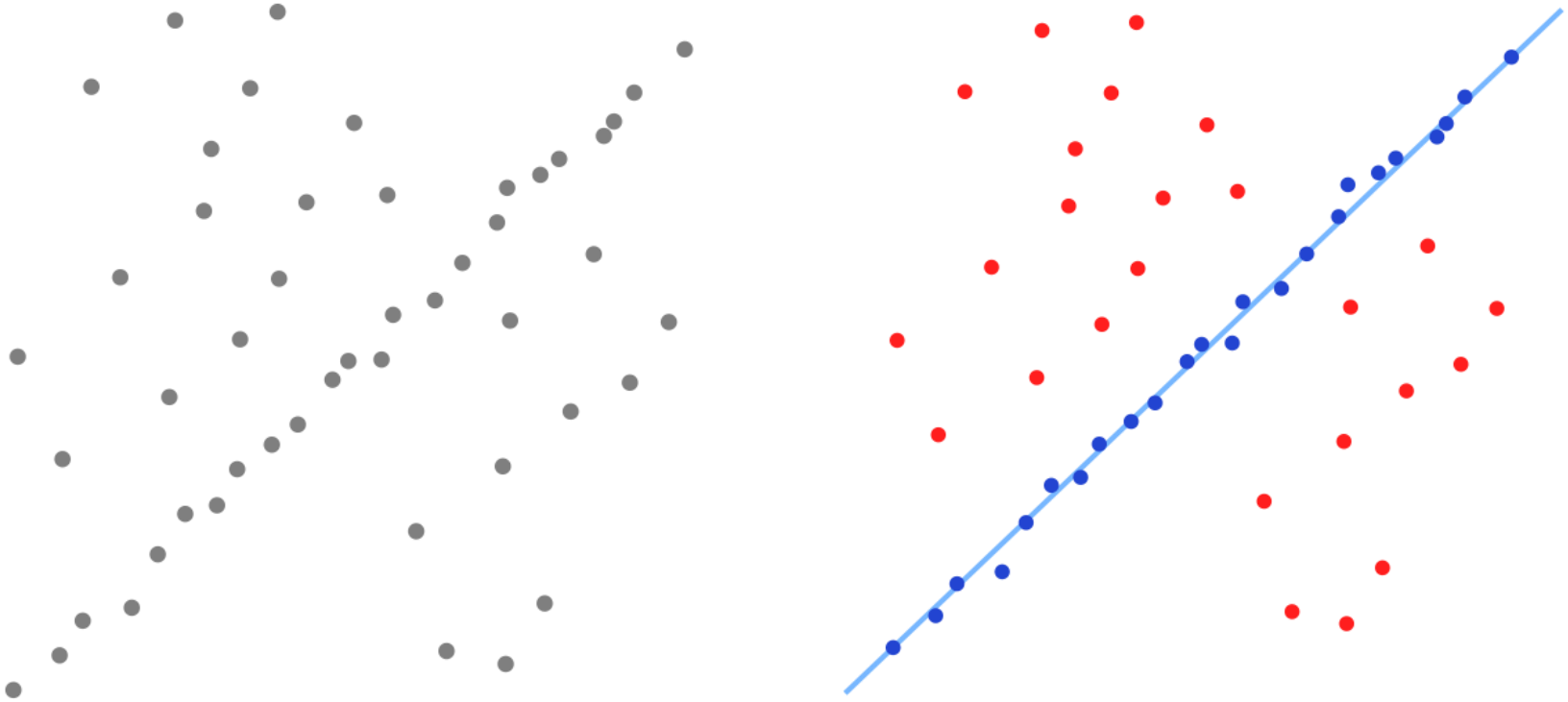
How many samples ?

- We want: at least one sample with all inliers
  - Can't guarantee: probability p
  - e.g., $p = 0.99$

Calculate N

- Let e = *% of outliers*, and s = *# of req'd data pts to fit model*
- $P(sample\ \boldsymbol{an}\ inlier) = ?$
- $P(sample\ a\ \boldsymbol{set}\ with\ all\ inliers) = ?$
- $P(sample\ a\ set\ with\ at\ least\ one\ outlier) = ?$
- $P(N\ trials\ with\ at\ least\ one\ outlier) = ?$
- With probability p, we want at least one trial with all inliers:
  $1 - P(N\ trials\ with\ at\ least\ one\ outlier) \geq p$
- Hence, the required number of trials is ?
  - $N \geq log(1\text{-}p)/log(1\text{-}(1\text{-}e)^s)$

RANSAC: Line Fitting



RANSAC for estimating affine transformation

- Repeat N times:
- Draw s points uniformly at random
- Estimate affine transformation from these s points
- Find inliers among the remaining features (i.e., distance between feature position and their transformed position is less than t)
- If there are d or more inliers, accept the transformation and refit using all inliers

Choosing the parameters

- Initial number of points s *(s=3 for affine transformation)*
  - Typically minimum number needed to fit the model
- Distance threshold t *(L2 distance btw. match and transformed match)*
  - Choose t so probability for inlier is p (e.g., 0.95)
  - Zero-mean Gaussian noise with std. dev. σ:

Codimension m	Model	t^2
1	line, fundamental matrix	$3.84\ \sigma^2$
2	homography, camera matrix	$5.99\ \sigma^2$
3	trifocal tensor	$7.81\ \sigma^2$

Table 3.2. *The distance threshold* $t^2 = F_m^{-1}(\alpha)\sigma^2$ *for a probability of* $\alpha = 0.95$ *that the point (correspondence) is an inlier.*

Choosing the parameters

- Initial number of points s *(s=3 for affine transformation)*
  - Typically minimum number needed to fit the model
- Distance threshold t *(L2 distance btw. match and transformed match)*
  - Choose t so probability for inlier is p (e.g., 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples N
  - Choose N so that, with probability p, at least one random sample is free from outliers (e.g., p=0.99) (outlier ratio: e)

$$\left(1-\left(1-e\right)^s\right)^N = 1-p$$

$$N = \log\left(1-p\right) / \log\left(1-\left(1-e\right)^s\right)$$

	proportion of outliers e						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

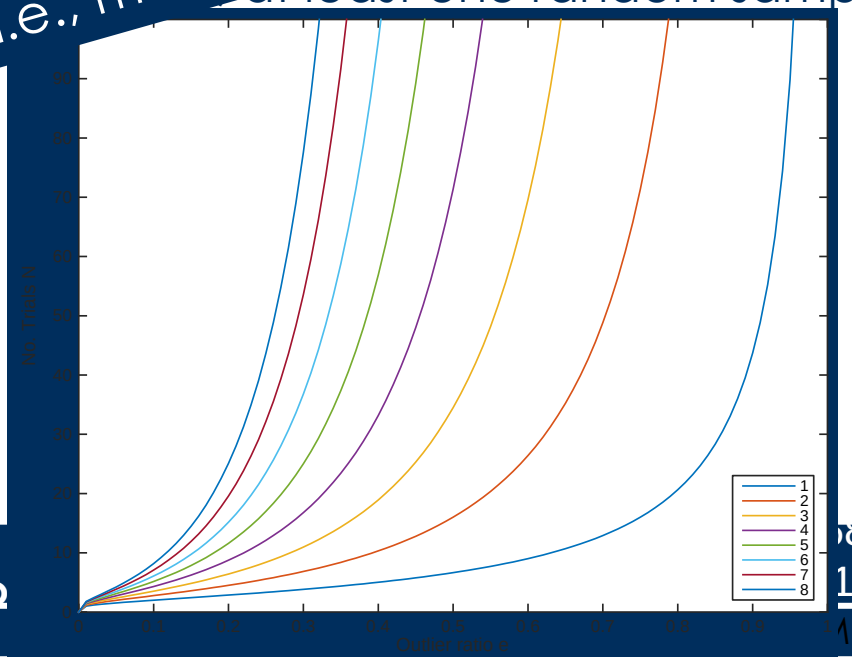
Choosing the parameters

- Initial number of points s *(s=3 for affine transformation)*
  - Typically minimum number needed to fit the model
- Distance threshold t *(L2 distance btw. match and transf[illegible]d match)*
  - Choose t so probability for inlier is p (e.g. [illegible]
  - Zero-mean Gaussian noise wit[illegible]
- Number of sam[illegible]
  - Choo[illegible] least one random sample is

Remarks
N = f(outliers%, model size s), not the number of points in the dataset!
N increases steeply with s (i.e., model size)

$$\left(1-\left(1-e\right)^{s}\right)^{N}=1-p$$

$$N=\log\left(1-p\right)/\log\left(1-\left(1-e\right)^{s}\right)$$



Choosing the parameters

- Initial number of points s
  - Typically minimum number needed to fit the model
- Distance threshold t
  - Choose t so probability for inlier is p (e.g., 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples N
  - Choose N so that, with probability p, at least one random sample is free from outliers (e.g., $p=0.99$) (outlier ratio: e)
- Consensus set size d
  - Should match expected inlier ratio

Source: M. Pollefeys

Adaptive RANSAC

- Eliminates the guess of outlier ratio

- $N = \infty$, 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
  - We can often do better than brute-force sampling

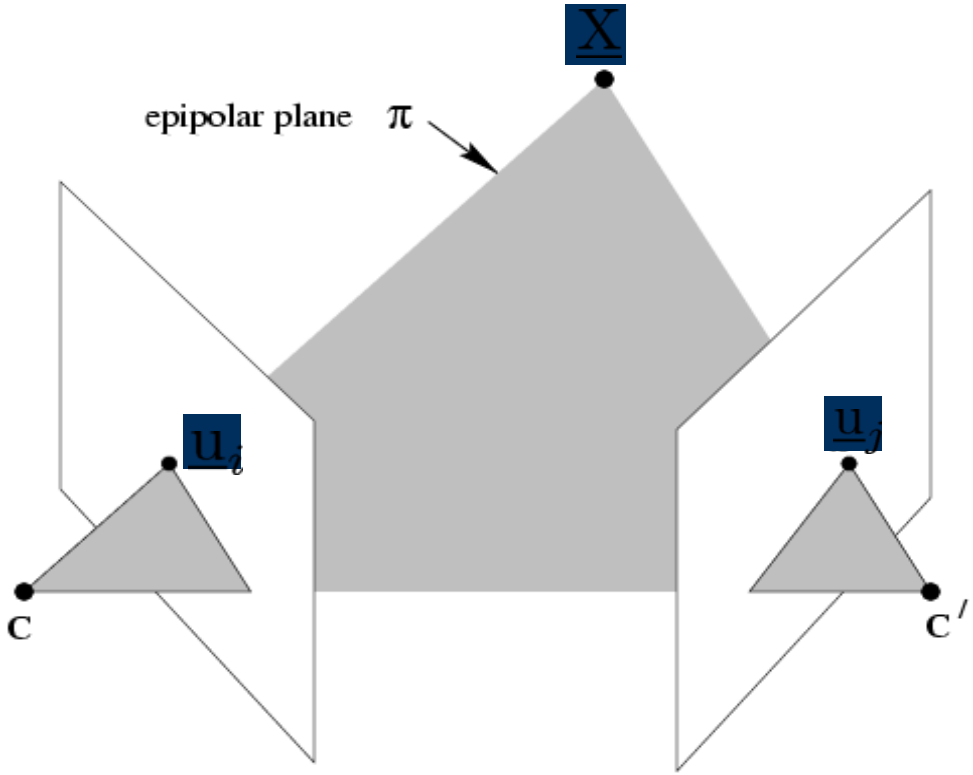
Essential/Fundamental Matrix Models

- Essential Matrix, E
  - Calibrated cameras
  - 5-dof

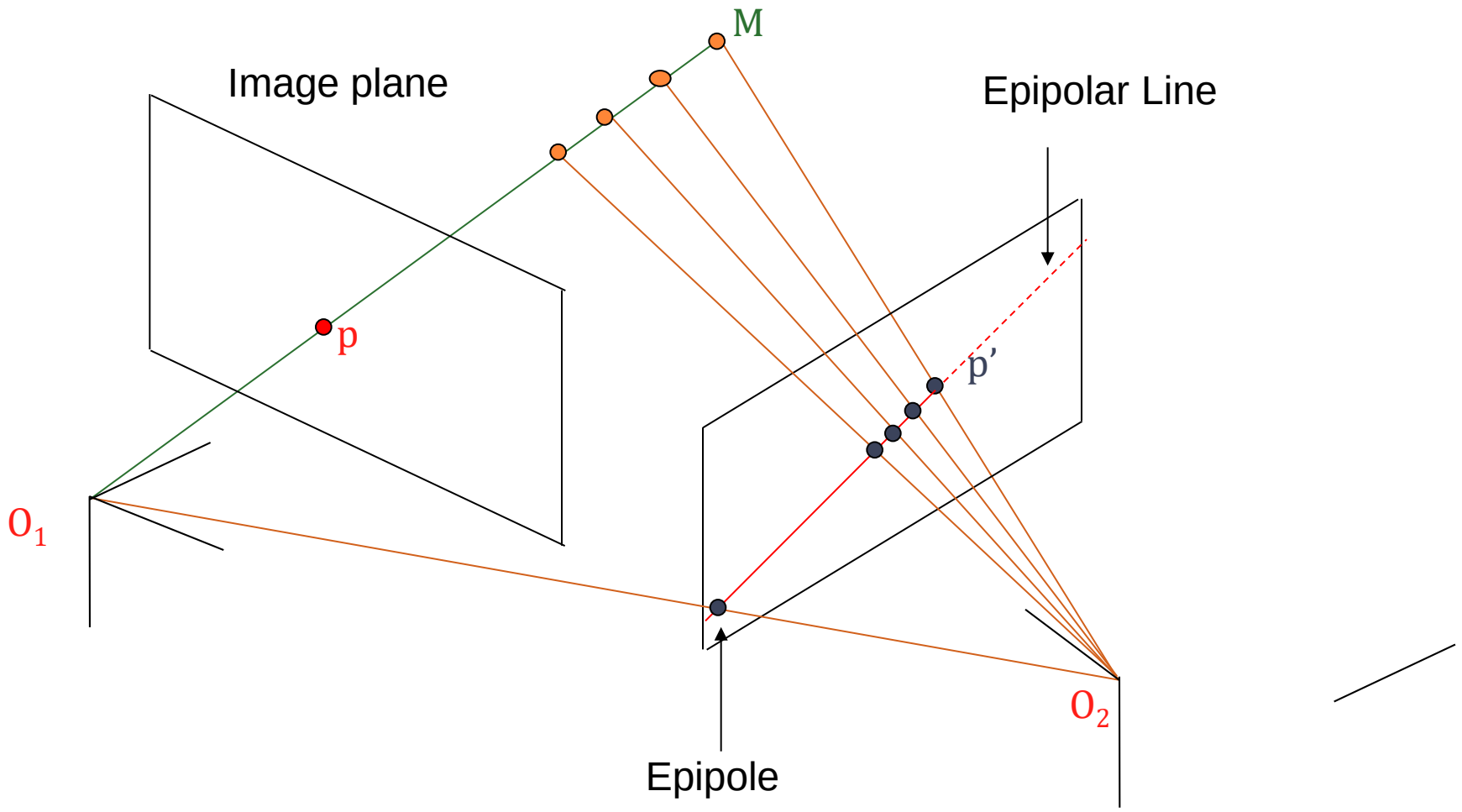
$$\underline{x}_j^\top E \underline{x}_i = 0$$

- Fundamental Matrix, F
  - Uncalibrated cameras
  - 7-dof

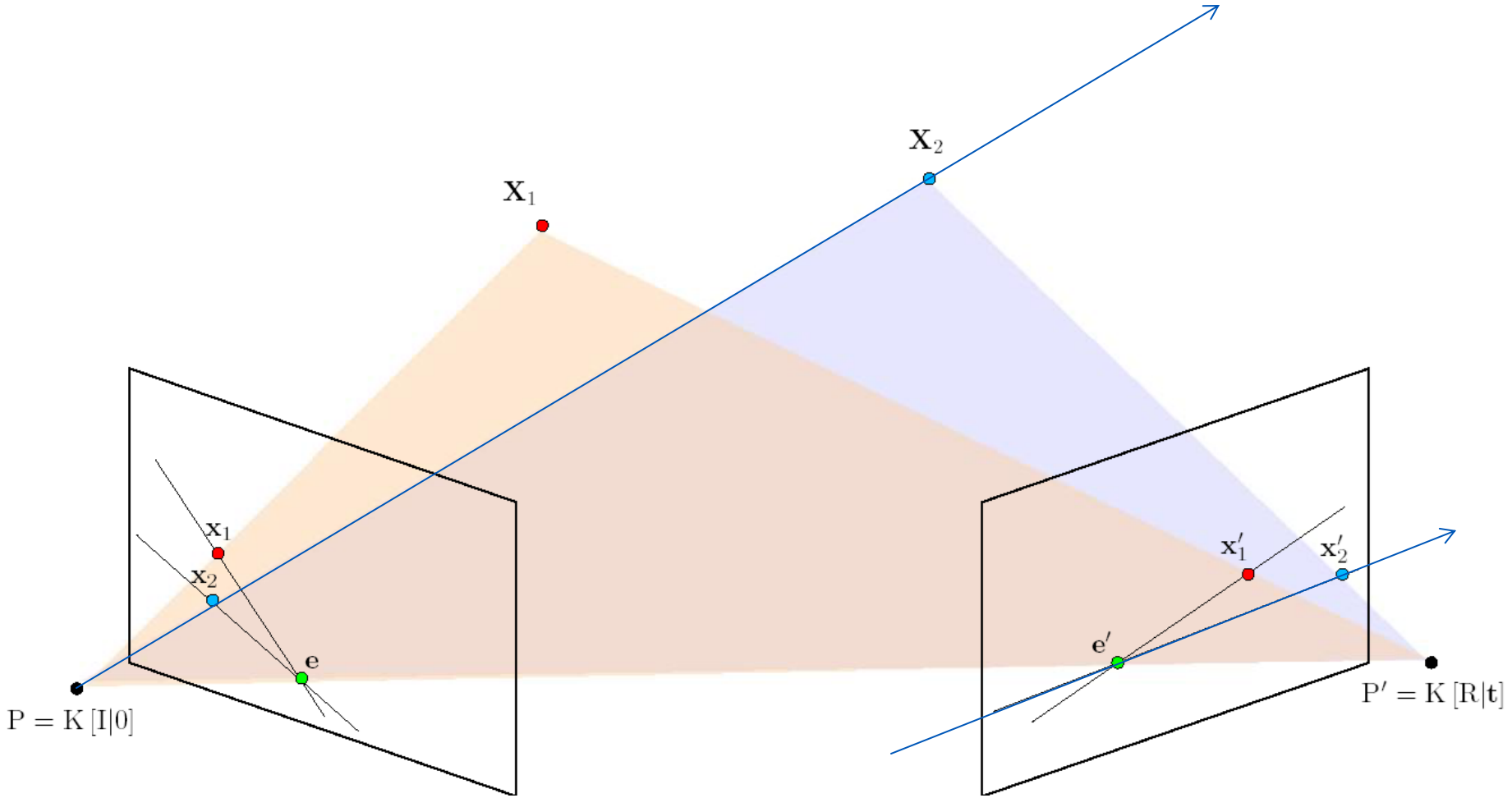
$$\underline{u}_j^\top F \underline{u}_i = 0$$

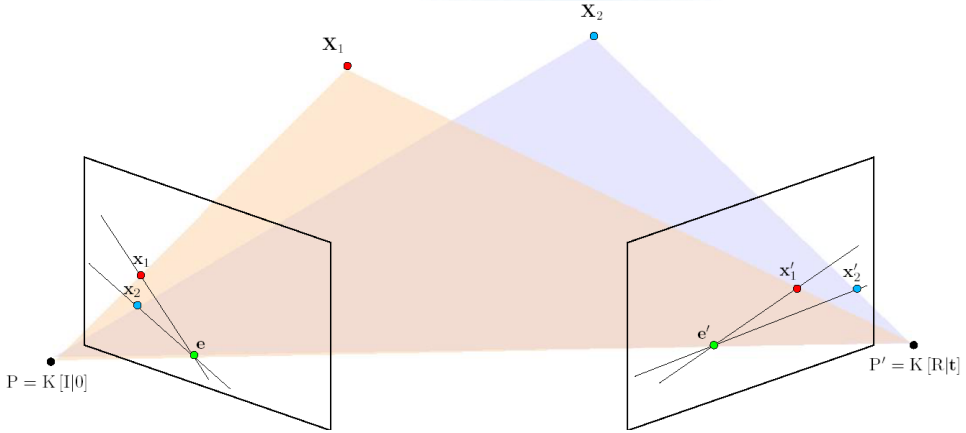


Stereo Constraints



Two-view Epipolar Geometry





- Vectors $\mathbf{x}_1$, $\mathbf{x}_1$', and $\mathbf{t}$ all lie within the same plane; similarly for $\mathbf{x}_2$, $\mathbf{x}_2$', and $\mathbf{t}$.

- Hence, the scalar triple product of any two corresponding image points is zero.

$$\mathbf{x}_i' \cdot \mathbf{t} \times R\mathbf{x}_i = 0$$

Essential Matrix
(Longuet-Higgins, 1981)

- This can be written in matrix/vector form as

$$\mathbf{x}_i'^{\top} [\mathbf{t}_\times] R\mathbf{x}_i = 0$$

using the skew-symmetric matrix

$$[\mathbf{t}_\times] = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

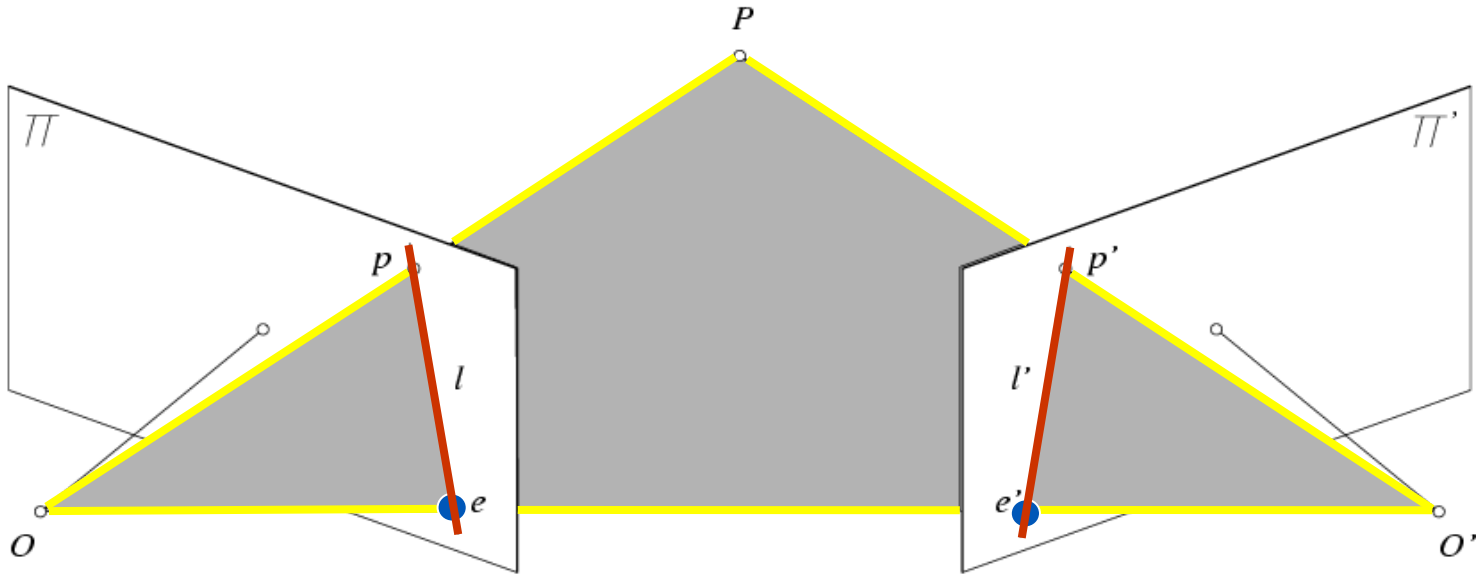
Properties of the Essential Matrix

$$\boldsymbol{p}^T \mathcal{E} \boldsymbol{p}' = 0 \quad \text{with} \quad \mathcal{E} = [\boldsymbol{t}_\times]\mathcal{R}$$

- $\mathcal{E}$ p' is the epipolar line associated with p'.
- $\mathcal{E}^T$p is the epipolar line associated with p.
- $\mathcal{E}$ e'=0 and $\mathcal{E}^T$e=0.
- $\mathcal{E}$ is singular.
- $\mathcal{E}$ has two equal non-zero singular values (Huang and Faugeras, 1989).

Courtesy
Marc Pollefeys

Epipolar Constraint: Uncalibrated Case



$$\hat{\boldsymbol{p}}^T \mathcal{E} \hat{\boldsymbol{p}}' = 0$$

$$\boldsymbol{p} = \mathcal{K}\hat{\boldsymbol{p}}$$

$$\boldsymbol{p}' = \mathcal{K}'\hat{\boldsymbol{p}}'$$

$$\Rightarrow \quad \boldsymbol{p}^T \mathcal{F} \boldsymbol{p}' = 0 \quad \text{with} \quad \mathcal{F} = \mathcal{K}^{-T} \mathcal{E} \mathcal{K}'^{-1}$$

$$\Downarrow$$

Fundamental Matrix
(Faugeras and Luong, 1992)

Courtesy
Marc Pollefeys

Properties of the Fundamental Matrix

$$\boldsymbol{p}^T \mathcal{F} \boldsymbol{p}' = 0 \quad \text{with} \quad \mathcal{F} = \mathcal{K}^{-T} \mathcal{E} \mathcal{K}'^{-1}$$

- $\mathcal{F}$ p' is the epipolar line associated with p'.
- $\mathcal{F}^T$ p is the epipolar line associated with p.
- $\mathcal{F}$ e'=0 and $\mathcal{F}^T$ e=0.
- $\mathcal{F}$ is singular.

3D registration models

$$\mathbf{x}_i'^{\top}[\mathbf{t}_\times]R\mathbf{x}_i = 0$$

- Essential Matrix

$$E = [\mathbf{t}_\times]R$$

- 5-DOF
- Calibrated camera
- Constraint in image coordinates

$$\mathbf{x}_i'^{\top}E\mathbf{x}_i = 0$$

- Fundamental Matrix

$$F = K^{-\top}[\mathbf{t}_\times]RK'^{-1}$$

- 7-DOF
- Uncalibrated camera
- Constraint in pixel coordinates

$$\mathbf{u}_i'^{\top}F\mathbf{u}_i = 0$$

The Eight-Point Algorithm (Longuet-Higgins, 1981)

$$(u, v, 1)\begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{pmatrix}\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0$$

$$(uu', uv', u, vu', vv', v, u', v', 1)\begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = 0$$

$$\begin{pmatrix} u_1u'_1 & u_1v'_1 & u_1 & v_1u'_1 & v_1v'_1 & v_1 & u'_1 & v'_1 \\ u_2u'_2 & u_2v'_2 & u_2 & v_2u'_2 & v_2v'_2 & v_2 & u'_2 & v'_2 \\ u_3u'_3 & u_3v'_3 & u_3 & v_3u'_3 & v_3v'_3 & v_3 & u'_3 & v'_3 \\ u_4u'_4 & u_4v'_4 & u_4 & v_4u'_4 & v_4v'_4 & v_4 & u'_4 & v'_4 \\ u_5u'_5 & u_5v'_5 & u_5 & v_5u'_5 & v_5v'_5 & v_5 & u'_5 & v'_5 \\ u_6u'_6 & u_6v'_6 & u_6 & v_6u'_6 & v_6v'_6 & v_6 & u'_6 & v'_6 \\ u_7u'_7 & u_7v'_7 & u_7 & v_7u'_7 & v_7v'_7 & v_7 & u'_7 & v'_7 \\ u_8u'_8 & u_8v'_8 & u_8 & v_8u'_8 & v_8v'_8 & v_8 & u'_8 & v'_8 \end{pmatrix}\begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \end{pmatrix} = -\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$$\sum_{i=1}^{n} (\boldsymbol{p}_i^T \mathcal{F} \boldsymbol{p}'_i)^2$$

under the constraint

$$|\mathcal{F}| = 1.$$

Courtesy Marc Pollefeys

The Normalized Eight-Point Algorithm (Hartley, 1995)

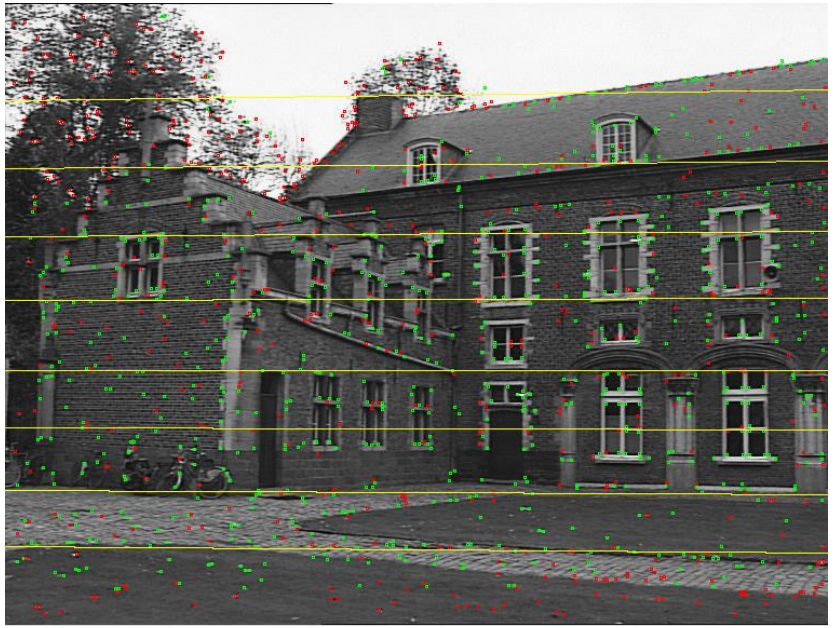
- Center the image data at the origin, and scale it so the RMS distance between the origin and the data points is sqrt(2) pixels: $q = T p$, $q' = T' p'$.

- Use the eight-point algorithm to compute $\mathcal{F}$ from the points q and q' .

- Enforce the rank-2 constraint
  - e.g. SVD decomposition and set $s_3 = 0$

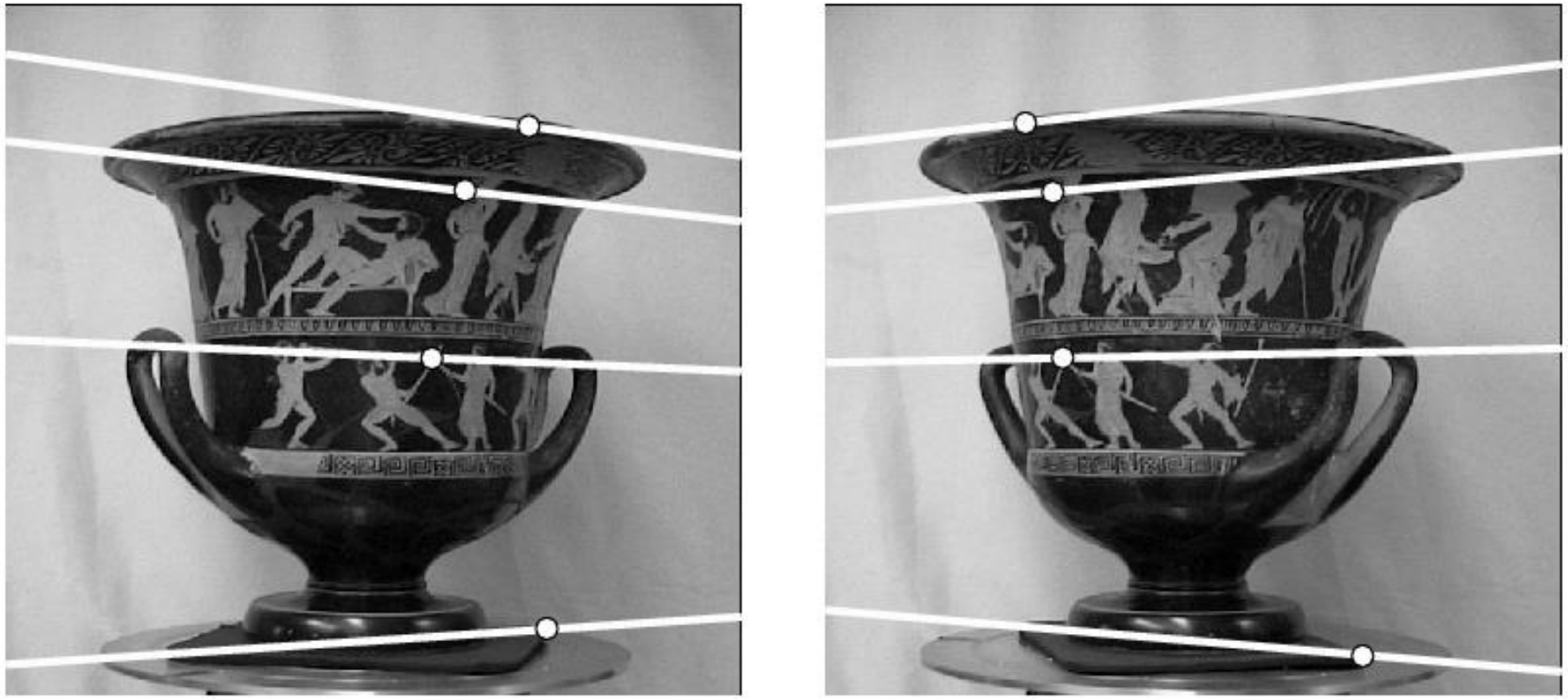
- Output $T \mathcal{F} T'$.

Epipolar geometry example

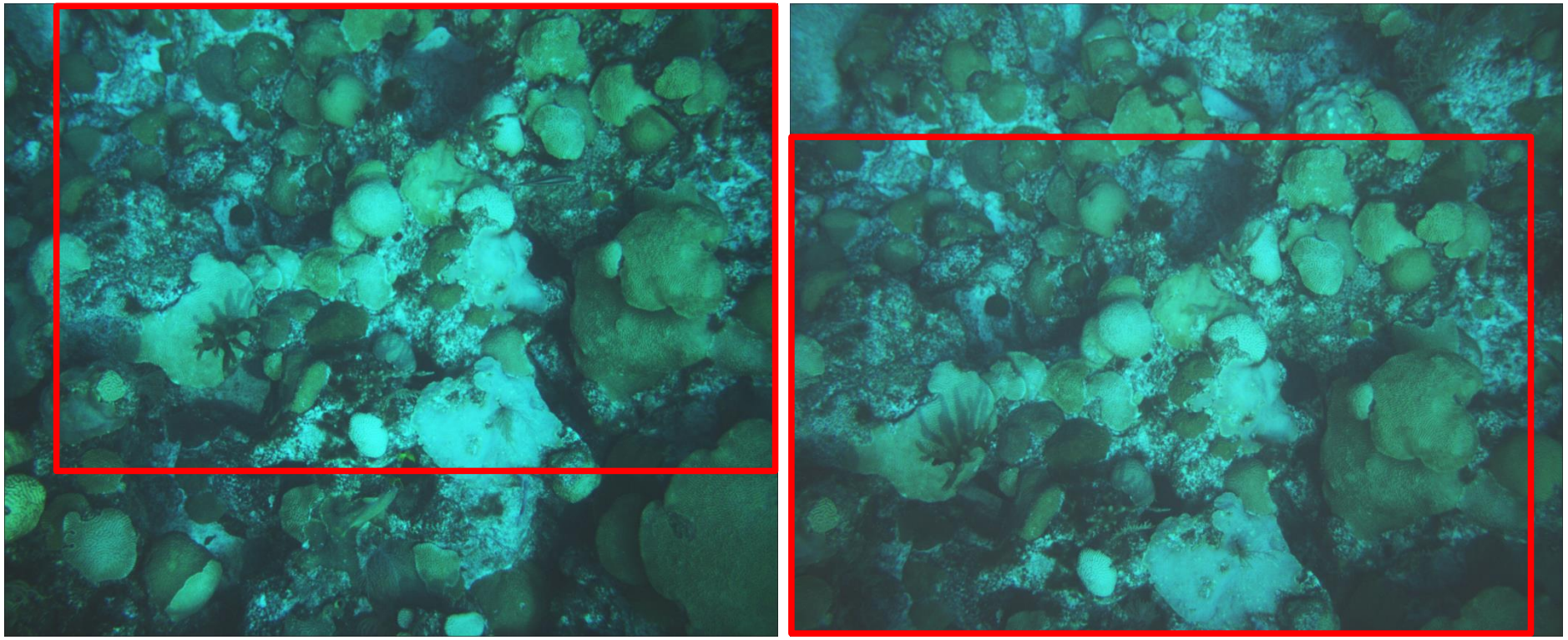


Courtesy
Marc Pollefeys

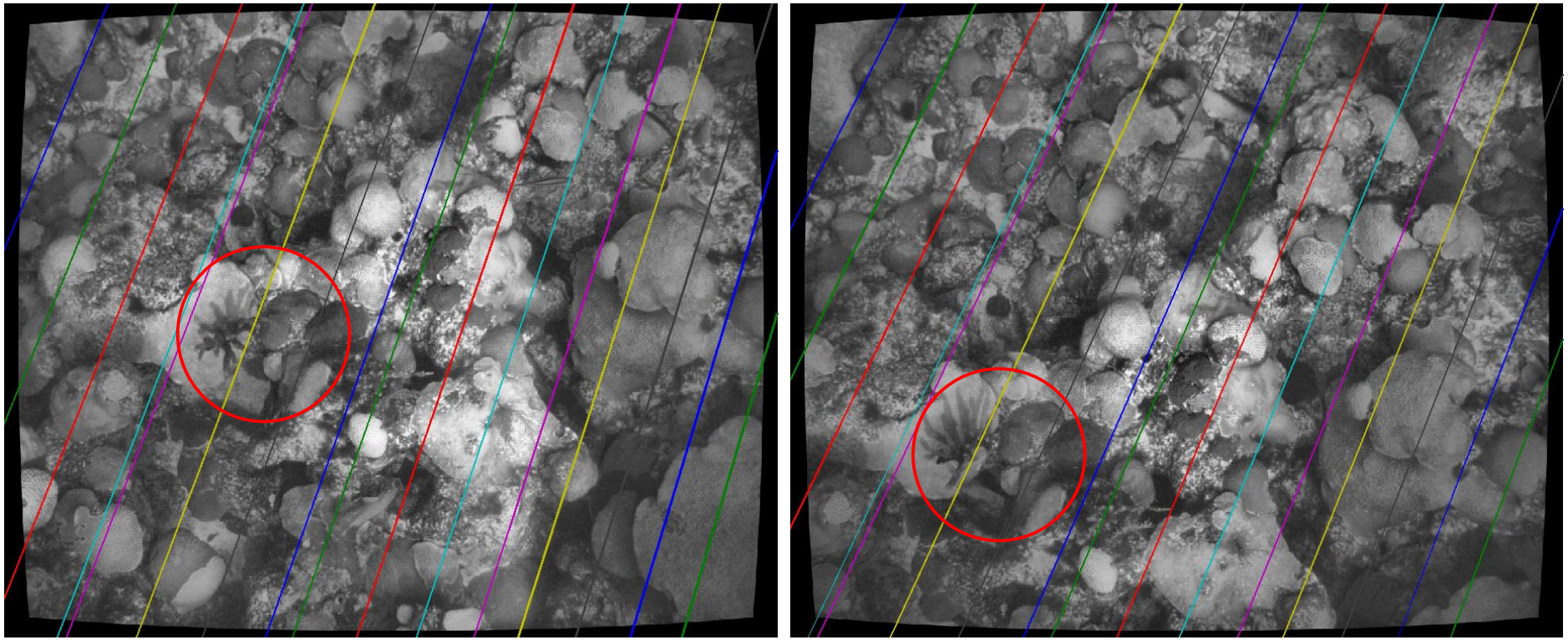
Example: Epipolar Lines for Converging Cameras



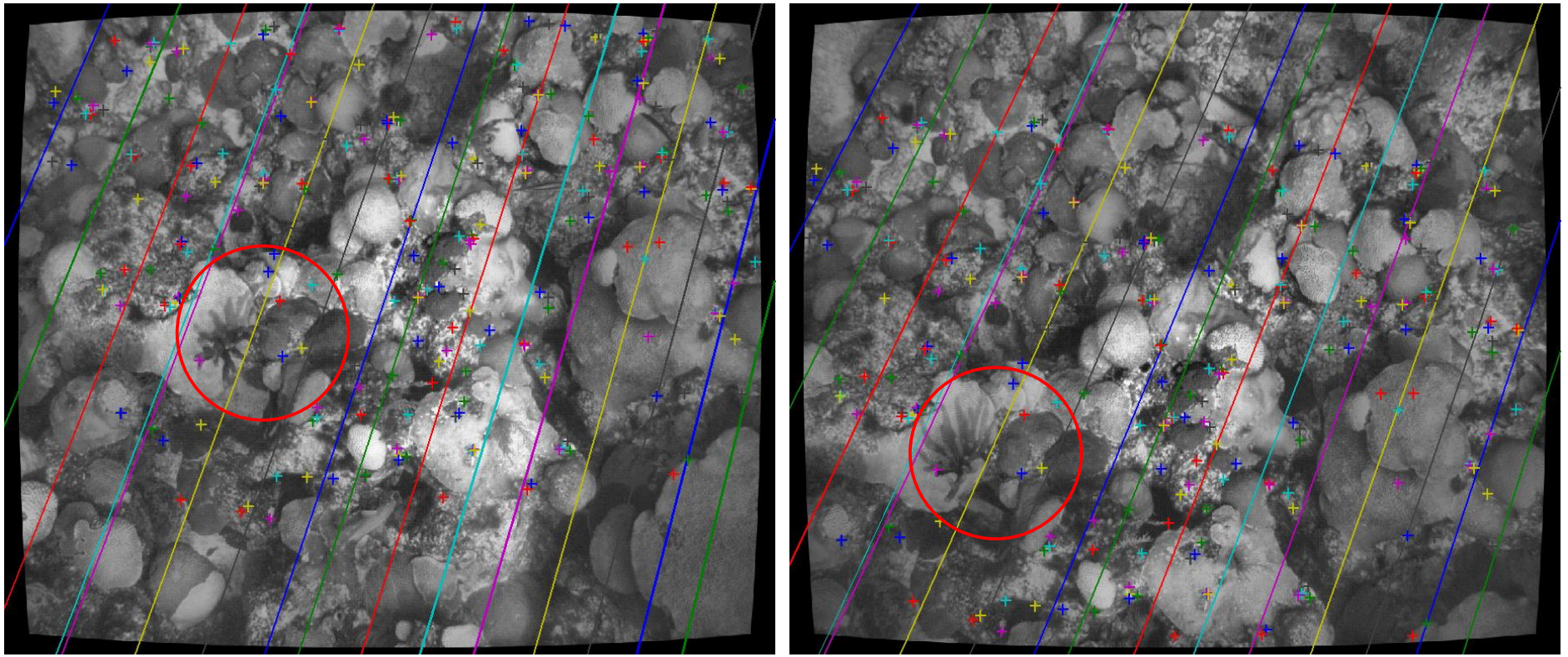
Epipolar Geometry Example



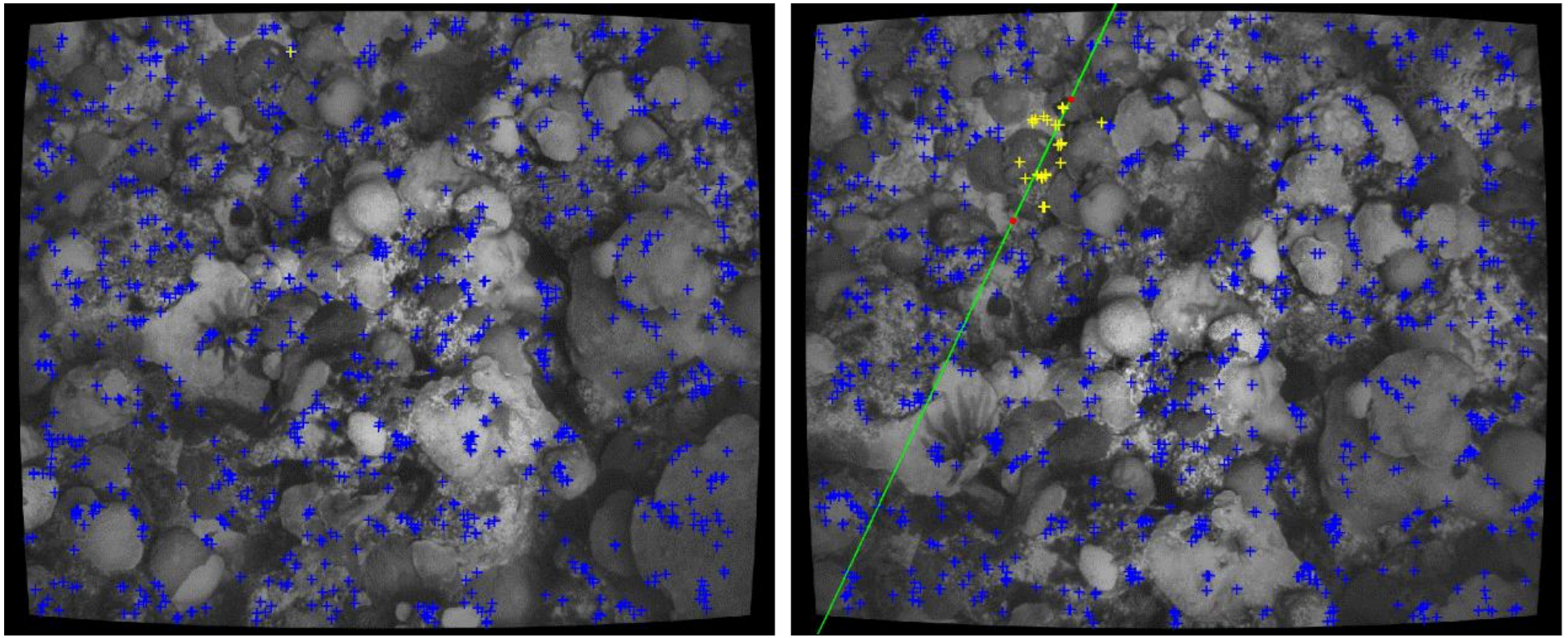
Navigation Instantiated Epipolar Geometry $E = [t]_x R$



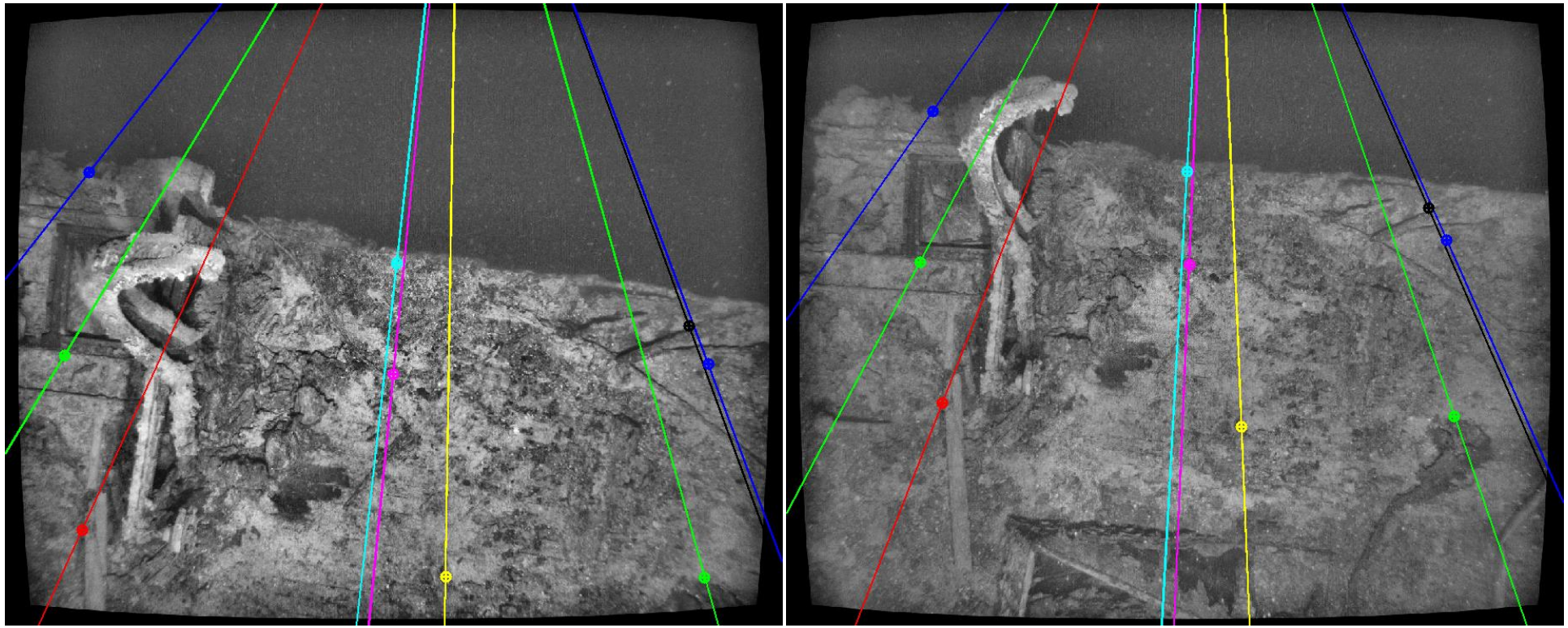
Epipolar Geometry Example



Can exploit pose prior in establishing similarity-based correspondences

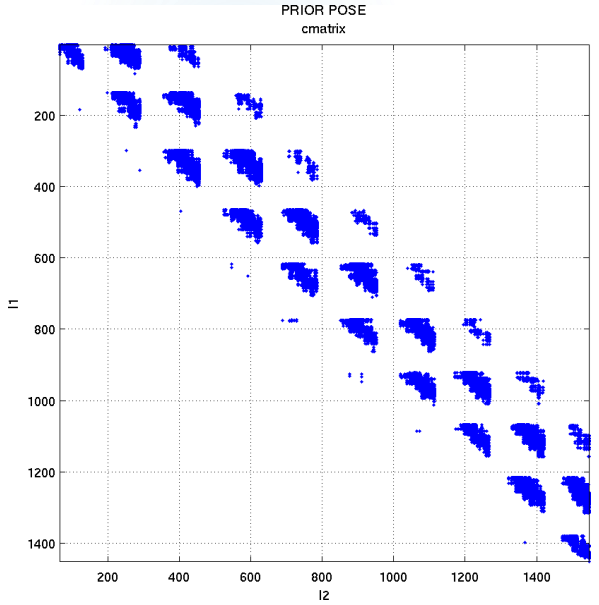


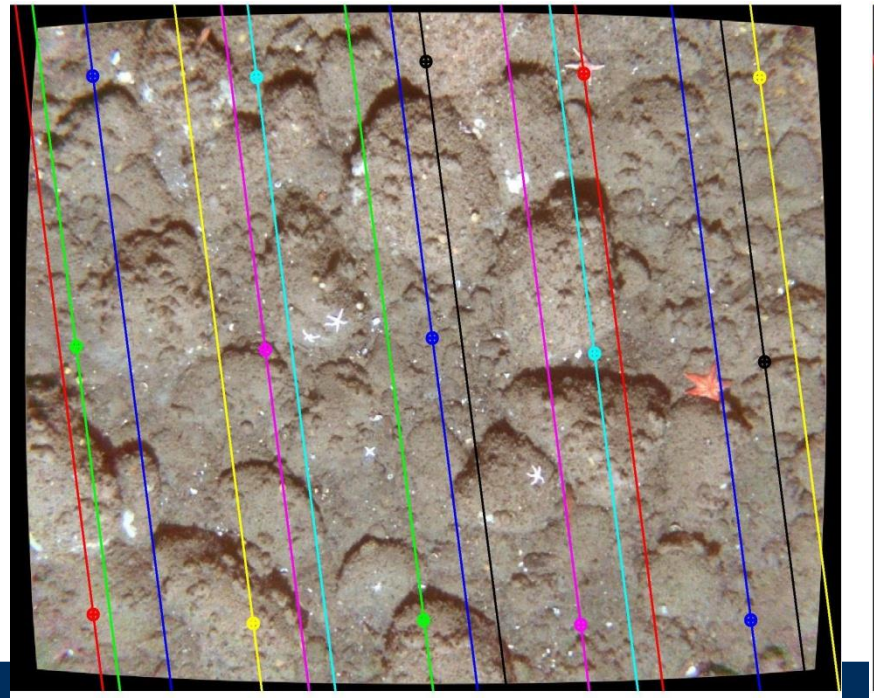
Epipolar Demo (RMS Titanic)

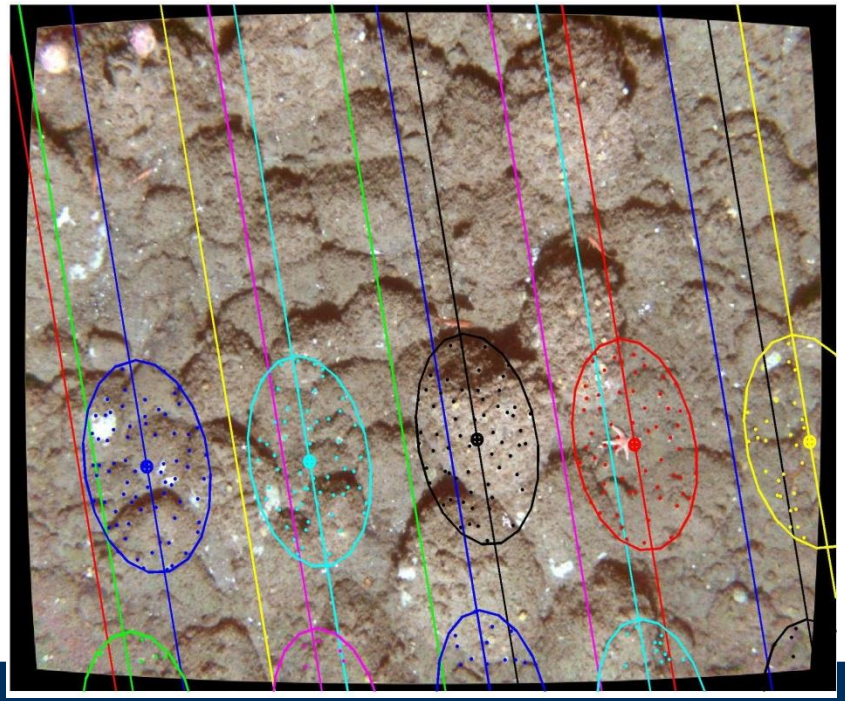


Can extend epipolar constraint to include effect of pose uncertainty

$$\mathbf{u}' = \frac{\mathrm{H}_\infty \underline{\mathbf{u}} + \mathrm{K}\mathbf{t}/Z}{\mathbf{H}_\infty^{3T} \underline{\mathbf{u}} + t_z/Z} \qquad \mathrm{H}_\infty = \mathrm{KRK}^{-1}$$

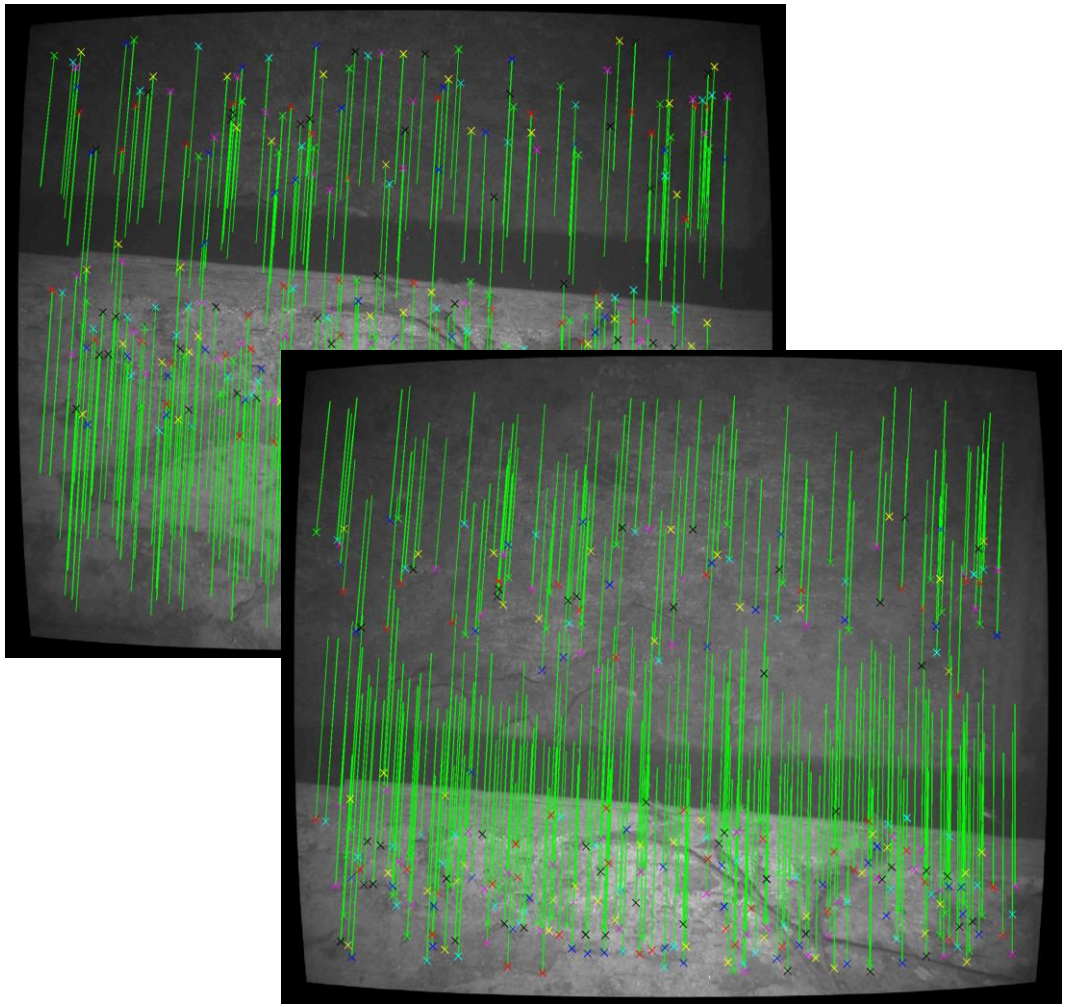


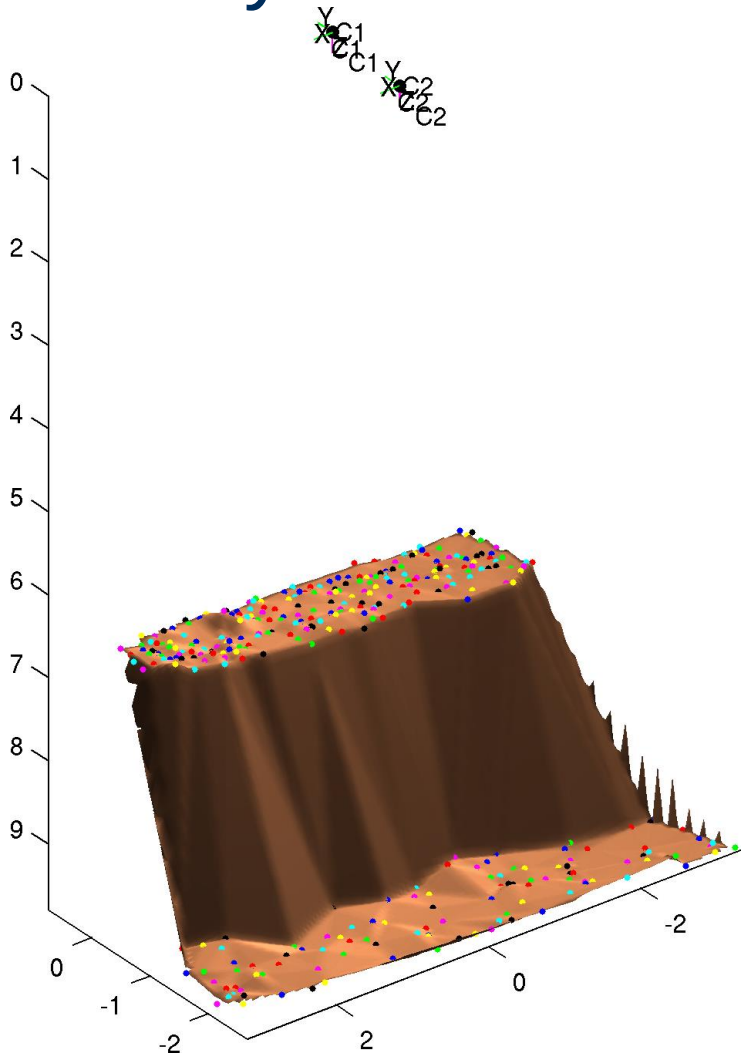




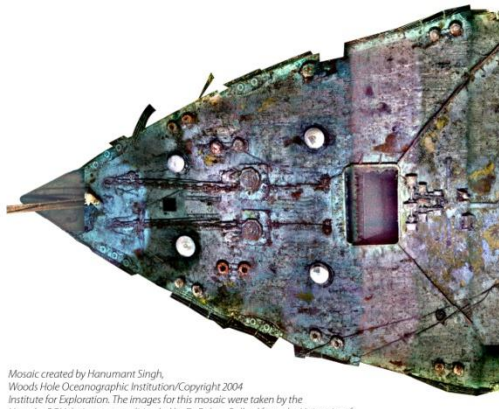
R. Eustice, O. Pizarro and H. Singh, *Visually augmented navigation for autonomous underwater vehicles*, *IEEE Journal of Oceanic Engineering*, 33(2):103-122, 2008.

Essential/Fundamental matrix registration allow for camera motion and structure recovery

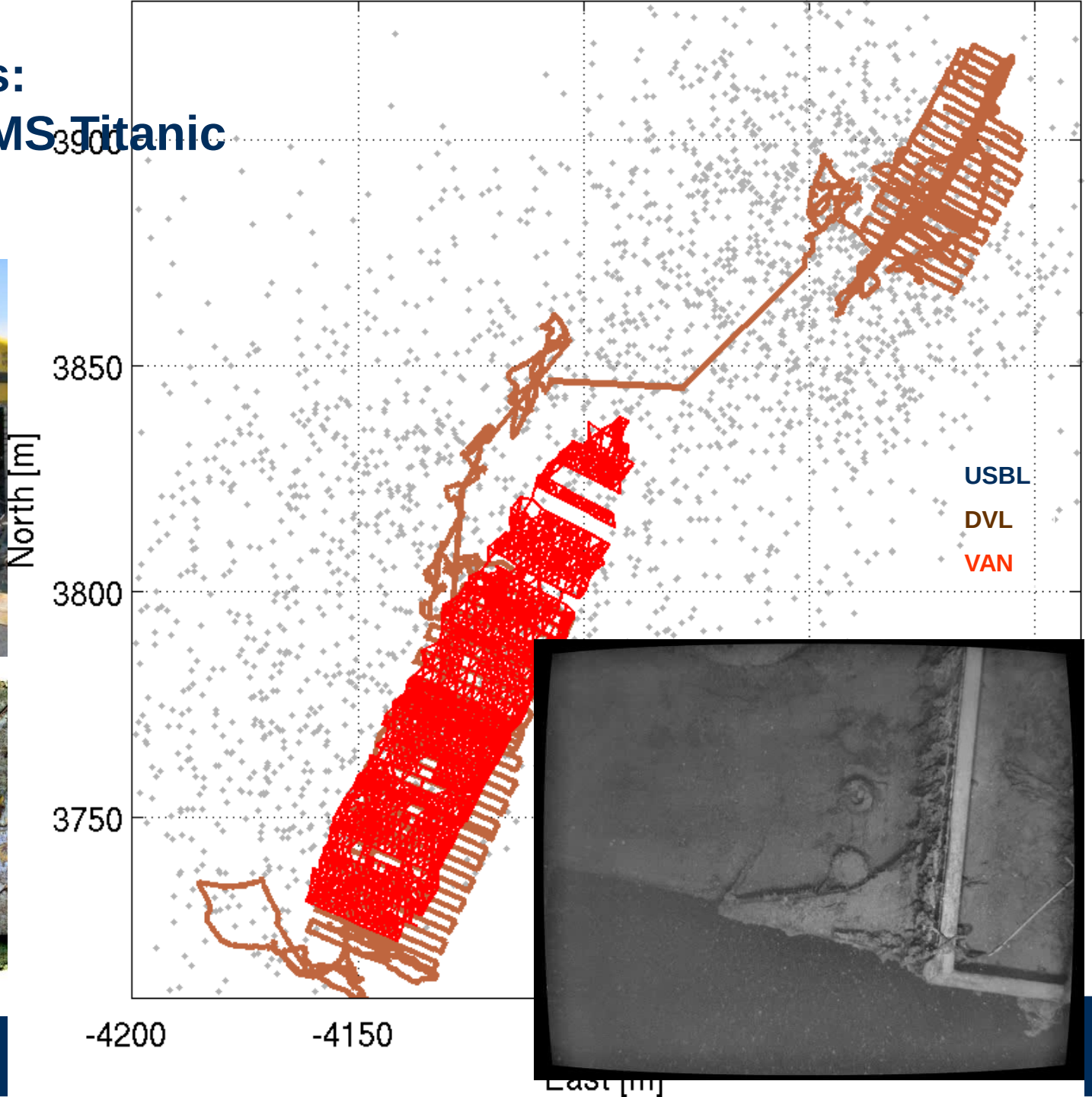




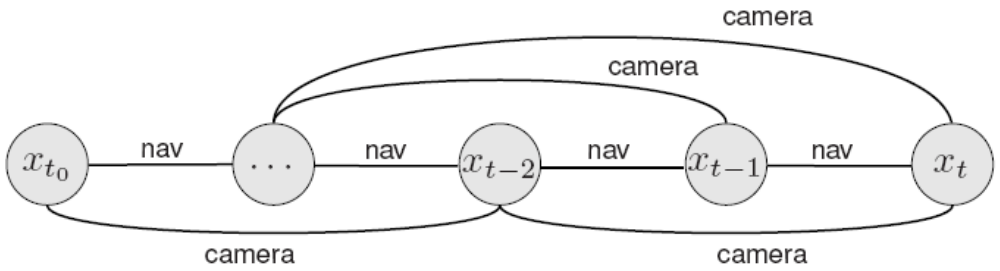
Real-World VAN Results: Visually Mapping the RMS Titanic

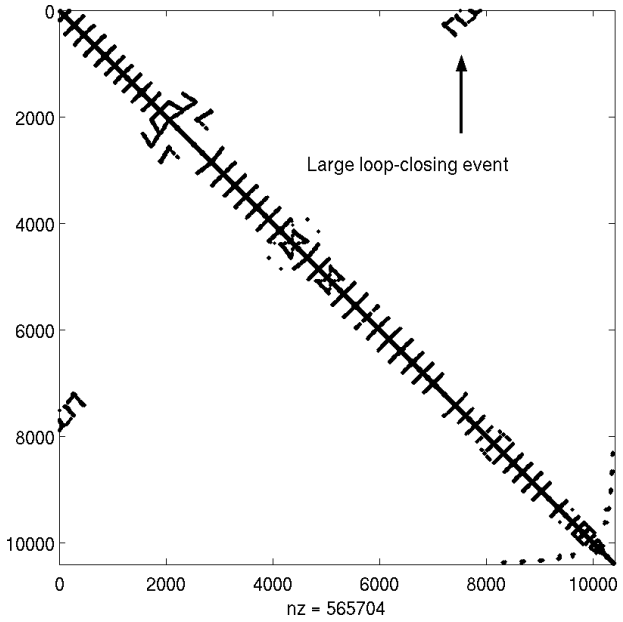


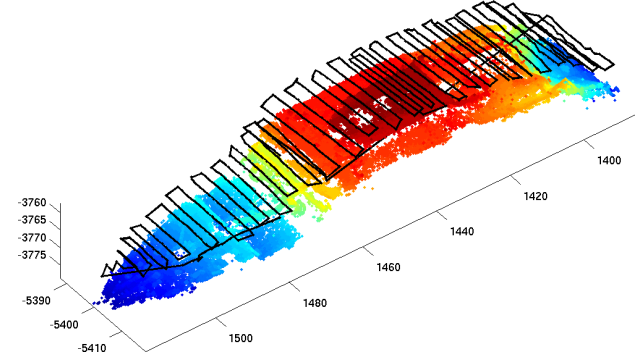
Mosaic created by Hanumant Singh,
Woods Hole Oceanographic Institution/Copyright 2004
Institute for Exploration. The images for this mosaic were taken by the
Hercules ROV during an expedition led by Dr Robert Ballard from the University of
Rhode Island's Institute for Archaeological Oceanography.

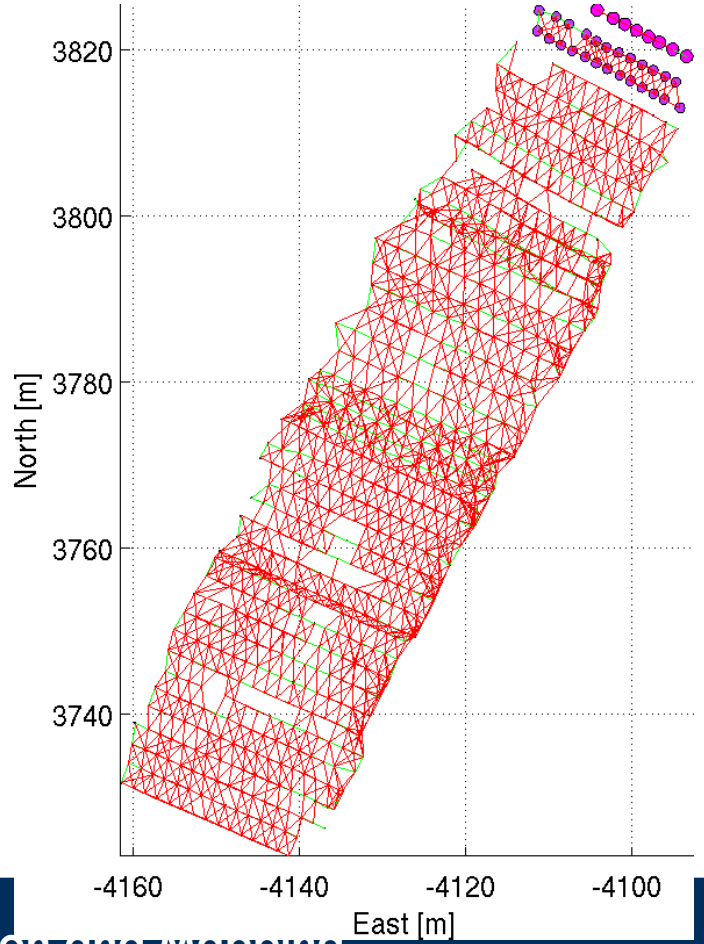


Camera Constraint Network









Camera-Produced Map