CS 4610/5335 – Lecture 17
Robot Vision (3-D)

Lawson L.S. Wong
Northeastern University
3/30/22

Material adapted from:
1. Robert Platt, CS 4610/5335
2. Peter Corke, Robotics, Vision and Control
3. Wolfram Burgard, U. Freiburg Mobile Robotics Course
4. Kavita Bala, Cornell CS 4670/5760

Announcements

Ethics assignment due 4/1

Ex5 expected to be posted tonight (approximately)
- Planned due 4/10 (Sun), late days allowed

Upcoming schedule:

4/4	(Mon)	Ethics 2
4/6	(Wed)	Advanced perception
4/11	(Mon)	Project check-ins

…

Robotics talk of interest!

Christoforos Mavrogiannis
Postdoctoral Research Associate
University of Washington CS&E

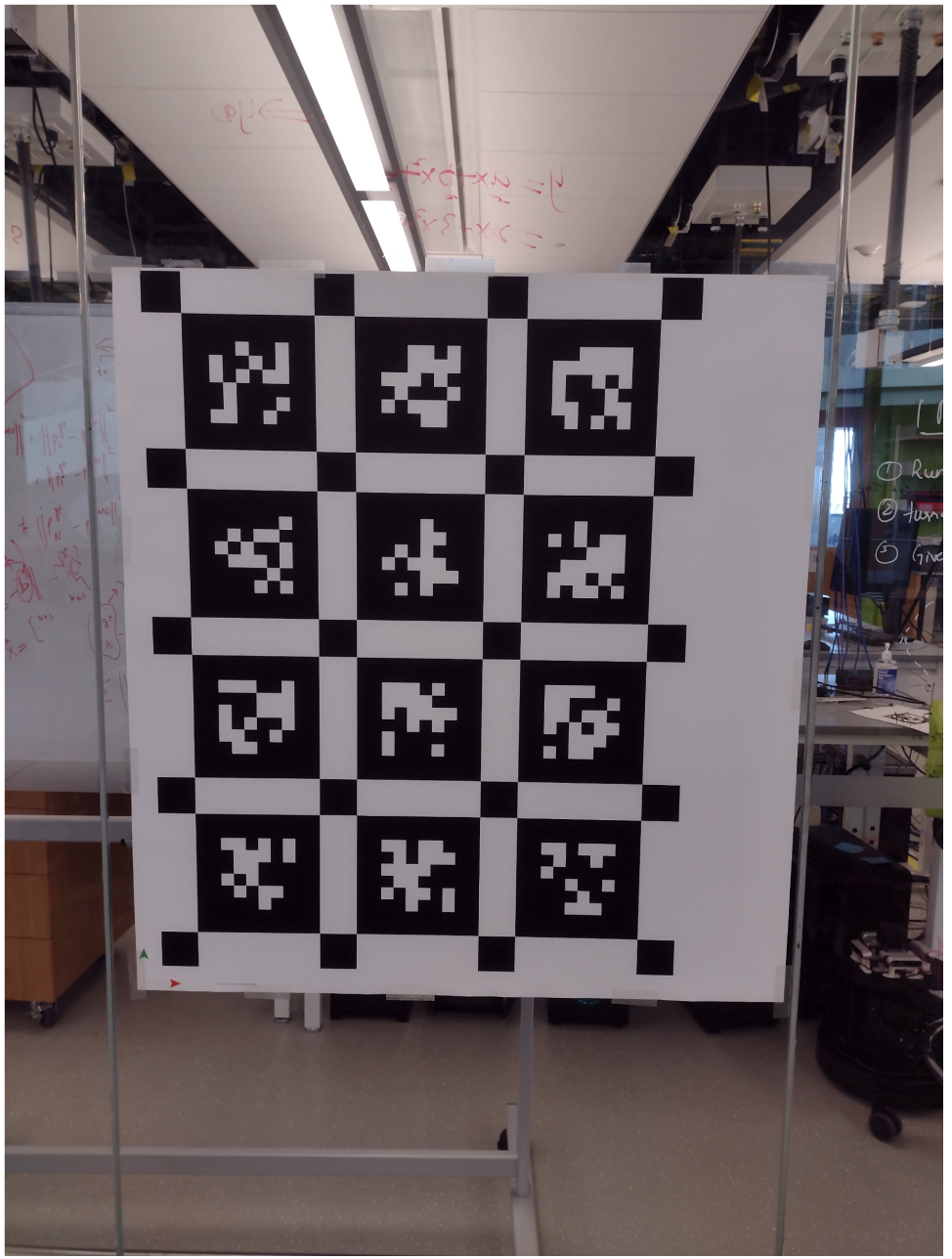
Building robots that humans accept

Friday, April 1
10:30 – 11:30 AM
366 West Village H
also on Zoom: 929 4529 4640 (282448)

“My research strives to develop highly functional, safe, and comfortable robots that humans accept.
… I argue that the path towards acceptance requires imbuing robots with a deeper understanding of how users perceive and react to them.
… I will share insights on robot navigation in dynamic environments, a fundamental task with many applications.”

Ex4 discussion

The first rule of perception



The rest of perception

Read RVC Part IV

In particular: Ch. 12 and 13 for 2-D vision (last time)
Ch. 14 for 3-D vision (this time)

See videos on https://robotacademy.net.au

Outline

2-D image alignment
- Feature descriptors: Corner detectors, SIFT, SURF
- Applications

3-D perception – point clouds
- Sensors and strategies

Aligning point clouds
- Iterative closest point

Extracting primitive shapes
- Surface normals
- Planes
- RANSAC
- More shapes (time permitting)

Corner detection

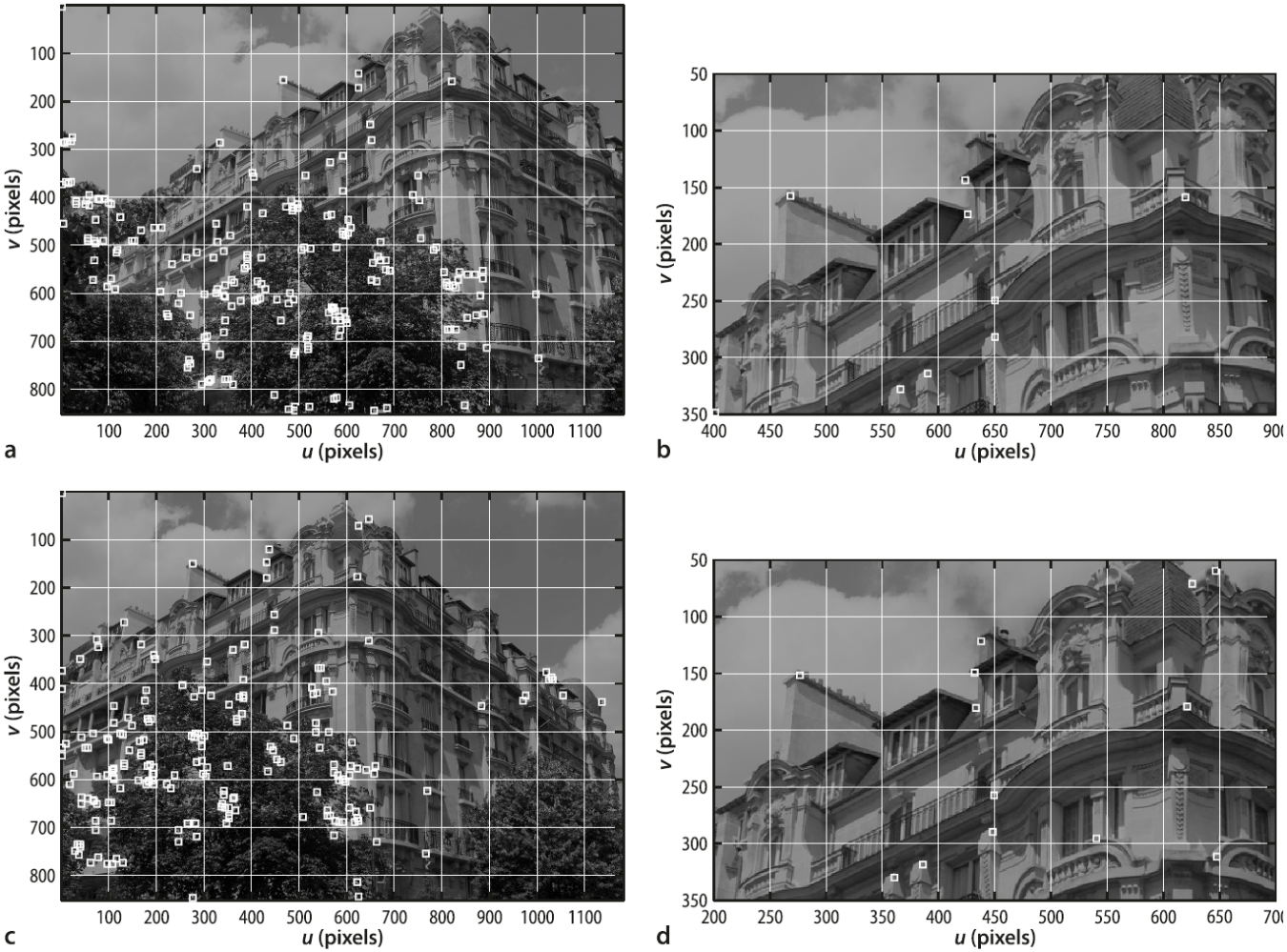


Fig. 13.21. Harris corner detector applied to two views of the same building. **a** View one; **b** zoomed in view one; **c** view two; **d** zoomed in view two. Notice that quite a number of the detected corners are attached to the same world features in the two views

Corner detection

If the weighting matrix is a Gaussian kernel $\boldsymbol{W} = \boldsymbol{G}(\sigma_I)$ and we replace the summation by a convolution then

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{G}(\sigma_I) * \boldsymbol{I}_u^2 & \boldsymbol{G}(\sigma_I) * \boldsymbol{I}_u \boldsymbol{I}_v \\ \boldsymbol{G}(\sigma_I) * \boldsymbol{I}_u \boldsymbol{I}_v & \boldsymbol{G}(\sigma_I) * \boldsymbol{I}_v^2 \end{pmatrix} \tag{13.14}$$

which is a symmetric 2×2 matrix referred to variously as the structure tensor, autocorrelation matrix or second moment matrix.

Corner detection

If the weighting matrix is a Gaussian kernel $\boldsymbol{W} = \boldsymbol{G}(\sigma_I)$ and we replace the summation by a convolution then

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{G}(\sigma_I) * \boldsymbol{I}_u^2 & \boldsymbol{G}(\sigma_I) * \boldsymbol{I}_u \boldsymbol{I}_v \\ \boldsymbol{G}(\sigma_I) * \boldsymbol{I}_u \boldsymbol{I}_v & \boldsymbol{G}(\sigma_I) * \boldsymbol{I}_v^2 \end{pmatrix} \tag{13.14}$$

which is a symmetric 2×2 matrix referred to variously as the structure tensor, autocorrelation matrix or second moment matrix.

An interest point (u, v) is one for which $s(\cdot)$ is high for *all* directions of the vector (δ_u, δ_v). That is, in whatever direction we move the window it rapidly becomes dissimilar to the original region. If we consider the original image $\boldsymbol{I}$ as a surface the eigenvalues of $\boldsymbol{A}$ are the principal curvatures of the surface at that point. If both eigenvalues are small then the surface is flat, that is the image region has approximately constant local intensity. If one eigenvalue is high and the other low, then the surface is ridge shaped which indicates an edge. If both eigenvalues are high the surface is sharply peaked which we consider to be a corner.

Corner detection (details)

If the weighting matrix is a Gaussian kernel $\boldsymbol{W} = \boldsymbol{G}(\sigma_I)$ and we replace the summation by a convolution then

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{G}(\sigma_I) * \boldsymbol{I}_u^2 & \boldsymbol{G}(\sigma_I) * \boldsymbol{I}_u \boldsymbol{I}_v \\ \boldsymbol{G}(\sigma_I) * \boldsymbol{I}_u \boldsymbol{I}_v & \boldsymbol{G}(\sigma_I) * \boldsymbol{I}_v^2 \end{pmatrix} \tag{13.14}$$

which is a symmetric 2×2 matrix referred to variously as the structure tensor, autocorrelation matrix or second moment matrix. It captures the intensity structure of the local neighborhood and its eigenvalues provide a rotationally invariant description of the neighborhood. The elements of the $\boldsymbol{A}$ matrix are computed from the image gradients, squared or multiplied, and then smoothed using a weighting matrix. The latter reduces noise and improves the stability and reliability of the detector. The gradient images $\boldsymbol{I}_u$ and $\boldsymbol{I}_v$ are typically calculated using a derivative of Gaussian kernel method (Sect. 12.5.1.3) with a smoothing parameter σ_D.

Corner detection (details)

An interest point (u, v) is one for which $s(\cdot)$ is high for *all* directions of the vector (δ_u, δ_v). That is, in whatever direction we move the window it rapidly becomes dissimilar to the original region. If we consider the original image $\boldsymbol{I}$ as a surface the eigenvalues of $\boldsymbol{A}$ are the principal curvatures of the surface at that point. If both eigenvalues are small then the surface is flat, that is the image region has approximately constant local intensity. If one eigenvalue is high and the other low, then the surface is ridge shaped which indicates an edge. If both eigenvalues are high the surface is sharply peaked which we consider to be a corner.◀

$$s\left(u, v, \delta_u, \delta_v\right) = \sum_{(i,j)\in\mathcal{W}} \left(\boldsymbol{I}\left[u+\delta_u+i, v+\delta_v+j\right] - \boldsymbol{I}\left[u+i, v+j\right]\right)^2$$

Corner detection (details)

The Shi-Tomasi detector considers the strength of the corner, or *cornerness*, as the minimum eigenvalue

$$C_{\mathrm{ST}}(u, v) = \min(\lambda_1, \lambda_2) \tag{13.15}$$

where λ_i are the eigenvalues of $\boldsymbol{A}$. Points in the image for which this measure is high are referred to as "*good features to track*". The Harris detector is based on this same insight but defines corner strength as

$$C_{\mathrm{H}}(u, v) = \det(\boldsymbol{A}) - k\,\mathrm{tr}(\boldsymbol{A}) \tag{13.16}$$

and again a large value represents a strong, distinct, corner. Since $\det(\boldsymbol{A}) = \lambda_1\lambda_2$ and $\mathrm{tr}(\boldsymbol{A}) = \lambda_1 + \lambda_2$ the Harris detector responds when both eigenvalues are large and elegantly avoids computing the eigenvalues of $\boldsymbol{A}$ which has a somewhat higher computational cost. A commonly used value for k is 0.04.

Scale-invariant feature transform (SIFT)

Figure 4: Top row shows model images for 3D objects with outlines found by background segmentation. Bottom image shows recognition results for 3D objects with model outlines and image keys used for matching.

Figure 5: Examples of 3D object recognition with occlusion.

Scale-invariant feature transform (SIFT)

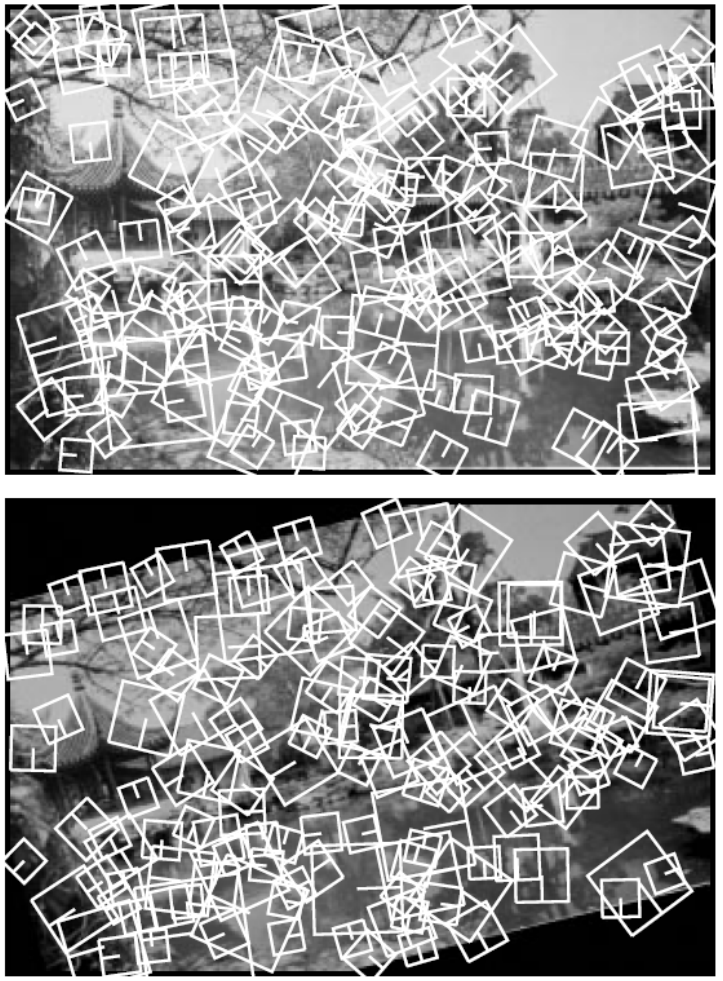


Figure 1: The second image was generated from the first by rotation, scaling, stretching, change of brightness and contrast, and addition of pixel noise. In spite of these changes, 78% of the keys from the first image have a closely matching key in the second image. These examples show only a subset of the keys to reduce clutter.

Image transformation	Match %	Ori %
A. Increase contrast by 1.2	89.0	86.6
B. Decrease intensity by 0.2	88.5	85.9
C. Rotate by 20 degrees	85.4	81.0
D. Scale by 0.7	85.1	80.3
E. Stretch by 1.2	83.5	76.1
F. Stretch by 1.5	77.7	65.0
G. Add 10% pixel noise	90.3	88.4
H. All of A,B,C,D,E,G.	78.6	71.8

Figure 2: For various image transformations applied to a sample of 20 images, this table gives the percent of keys that are found at matching locations and scales (Match %) and that also match in orientation (Ori %).

SIFT: Scale-invariant keypoint detection

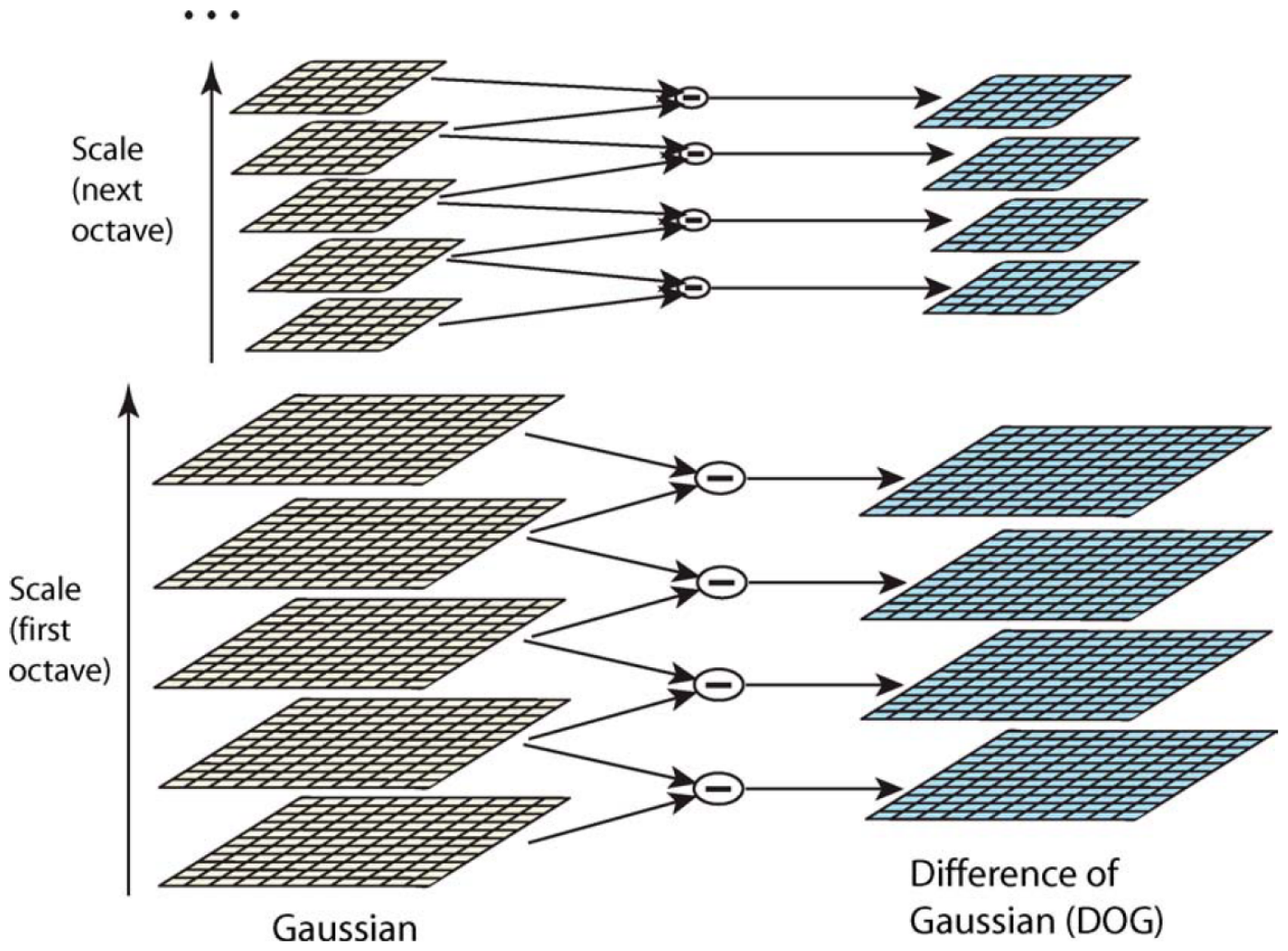


Figure 1. For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated.

SIFT: Keypoint selection

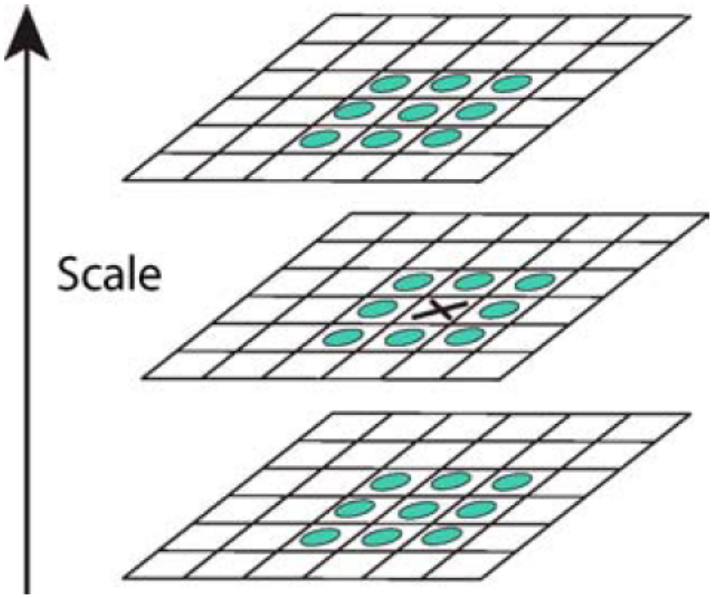


Figure 2. Maxima and minima of the difference-of-Gaussian images are detected by comparing a pixel (marked with X) to its 26 neighbors in 3 × 3 regions at the current and adjacent scales (marked with circles).

SIFT: Keypoint selection

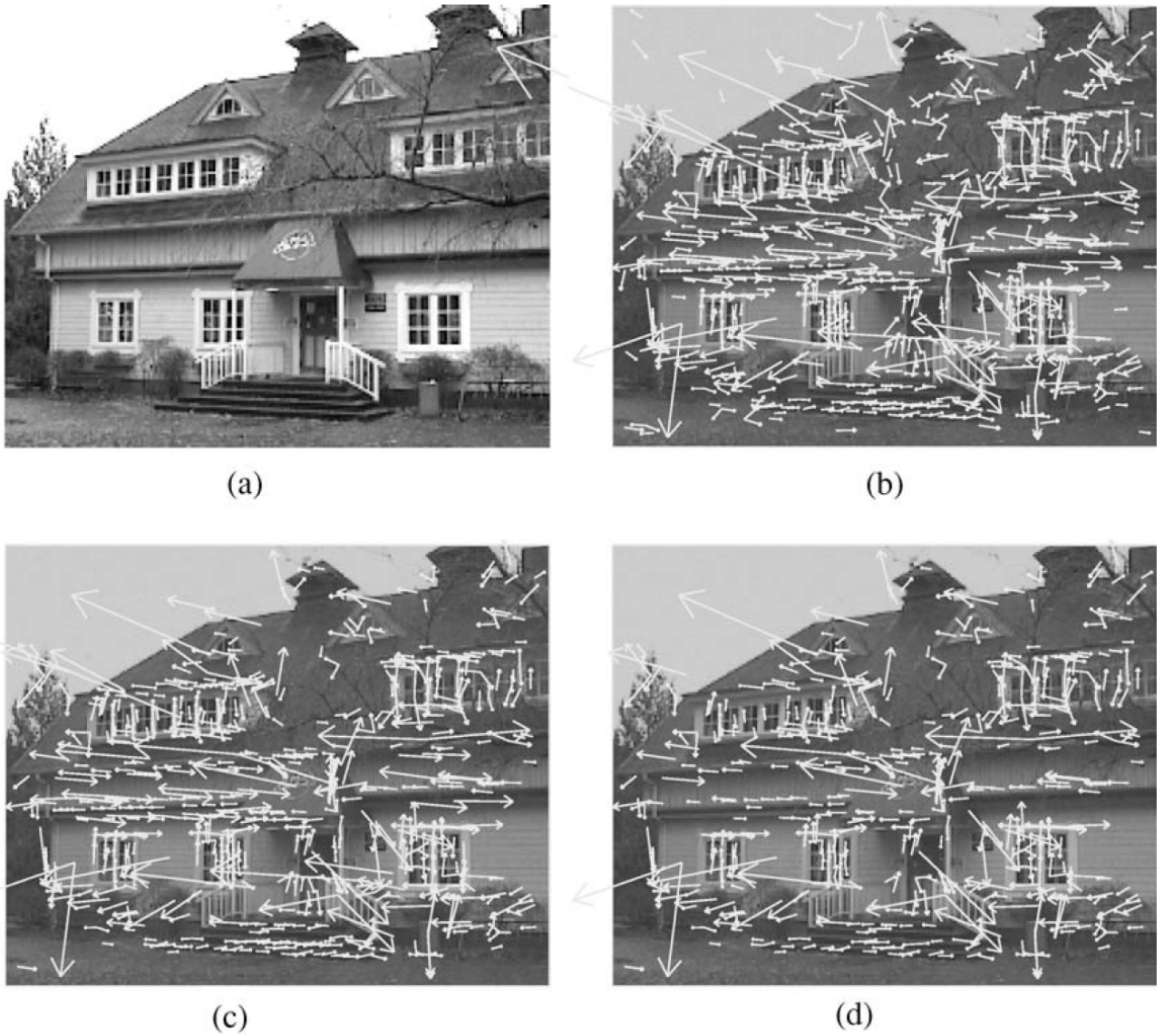


Figure 5. This figure shows the stages of keypoint selection. (a) The 233 × 189 pixel original image. (b) The initial 832 keypoints locations at maxima and minima of the difference-of-Gaussian function. Keypoints are displayed as vectors indicating scale, orientation, and location. (c) After applying a threshold on minimum contrast, 729 keypoints remain. (d) The final 536 keypoints that remain following an additional threshold on ratio of principal curvatures.

SIFT: Keypoint descriptor

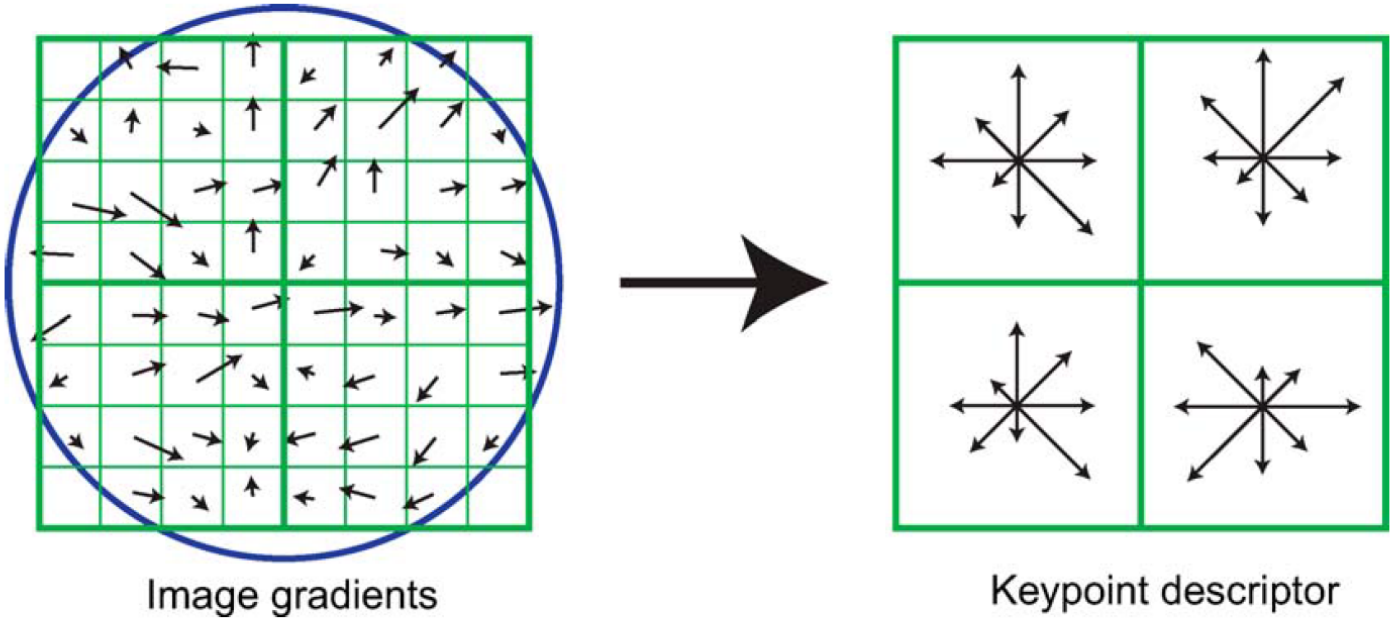


Figure 7. A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2×2 descriptor array computed from an 8×8 set of samples, whereas the experiments in this paper use 4×4 descriptors computed from a 16×16 sample array.

Speeded-up robust features (SURF)

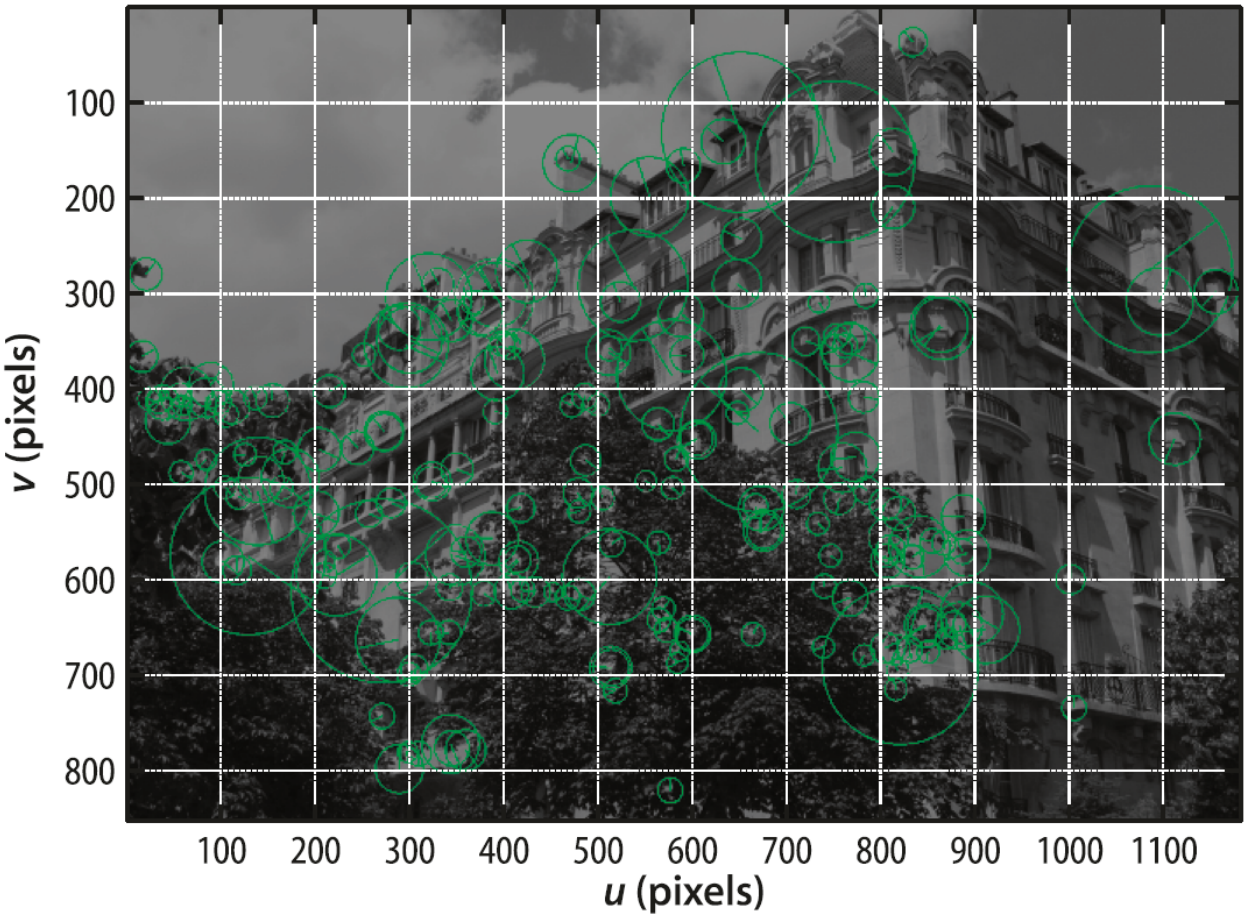


Fig. 13.28.
SURF descriptors showing the support region (scale) and orientation as a radial line

Speeded-up robust features (SURF)

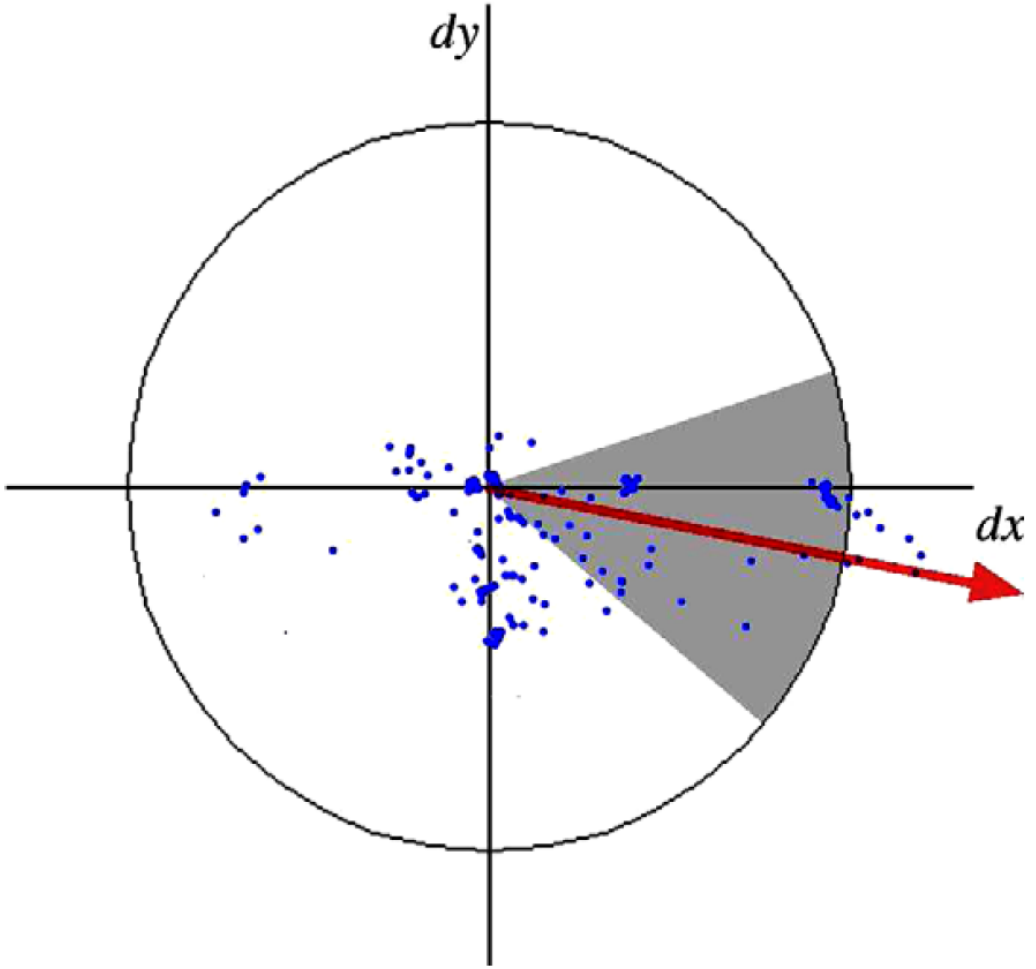


Fig. 10. Orientation assignment: a sliding orientation window of size $\frac{\pi}{3}$ detects the dominant orientation of the Gaussian weighted Haar wavelet responses at every sample point within a circular neighbourhood around the interest point.

Speeded-up robust features (SURF)

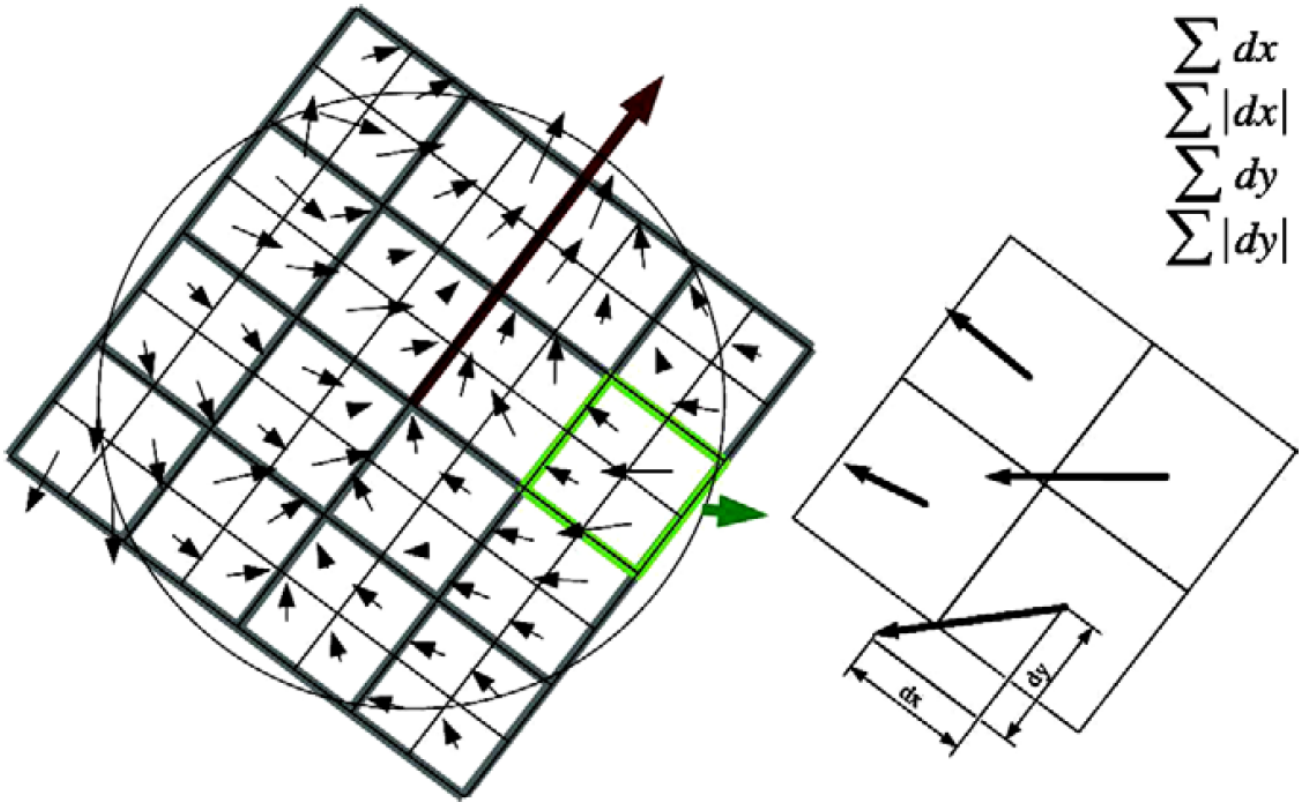


Fig. 12. To build the descriptor, an oriented quadratic grid with 4×4 square sub-regions is laid over the interest point (left). For each square, the wavelet responses are computed from 5×5 samples (for illustrative purposes, we show only 2×2 sub-divisions here). For each field, we collect the sums d_x, $|d_x|$, d_y, and $|d_y|$, computed relatively to the orientation of the grid (right).

Speeded-up robust features (SURF)

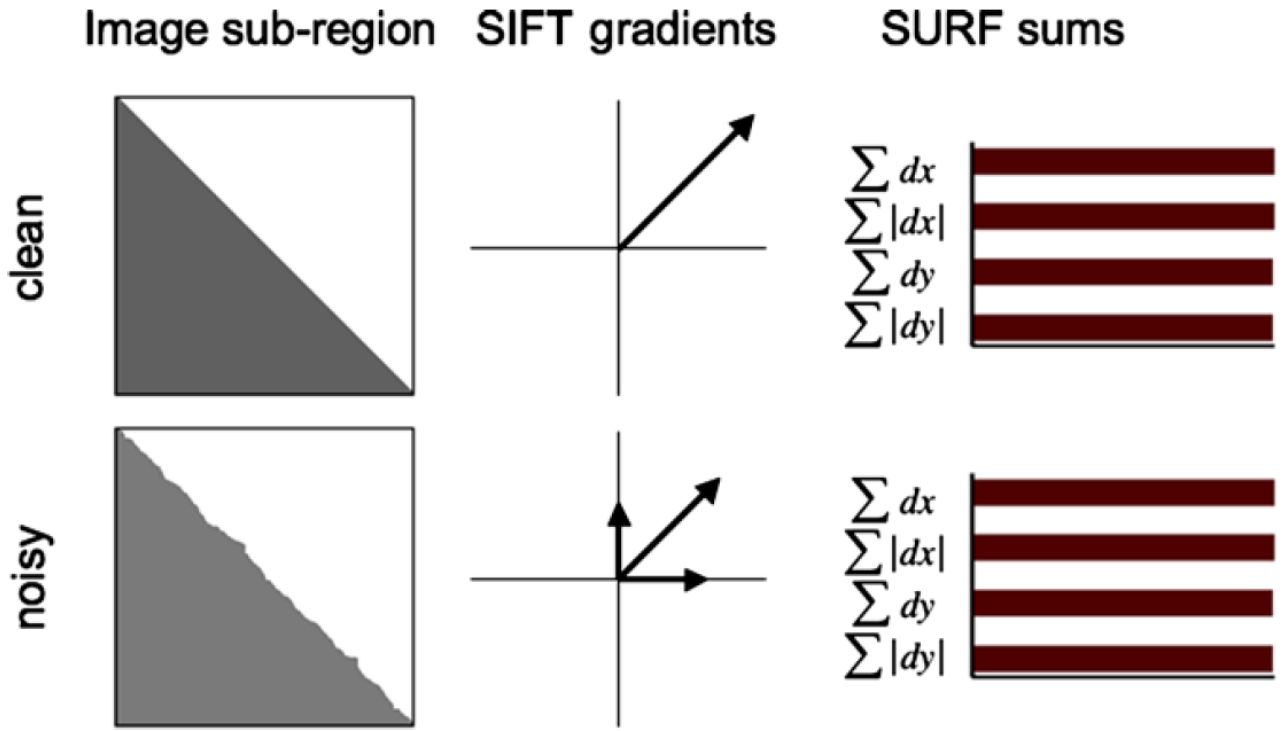


Fig. 14. Due to the global integration of SURF's descriptor, it stays more robust to various image perturbations than the more locally operating SIFT descriptor.

Feature descriptors (details)

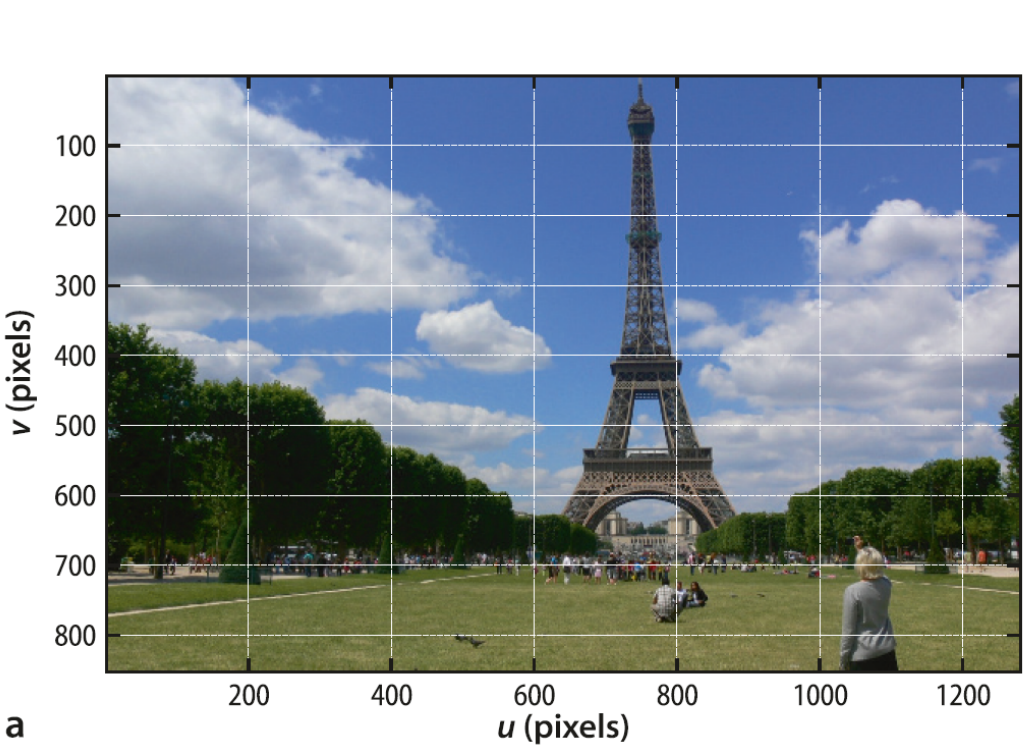
The SURF algorithm is more than just a scale-invariant feature detector, it also computes a very robust *descriptor*. The descriptor is a 64-element vector that encodes the image gradient in subregions of the support region in a way which is invariant to brightness, scale and rotation. This enables feature descriptors to be unambiguously matched to a descriptor of the same world point in another image even if their scale and orientation are quite different. The difference in position, scale and orientation of the matched features gives some indication of the relative camera motion between the two views. Matching features between scenes is crucial to the problems that we will address in the next chapter.

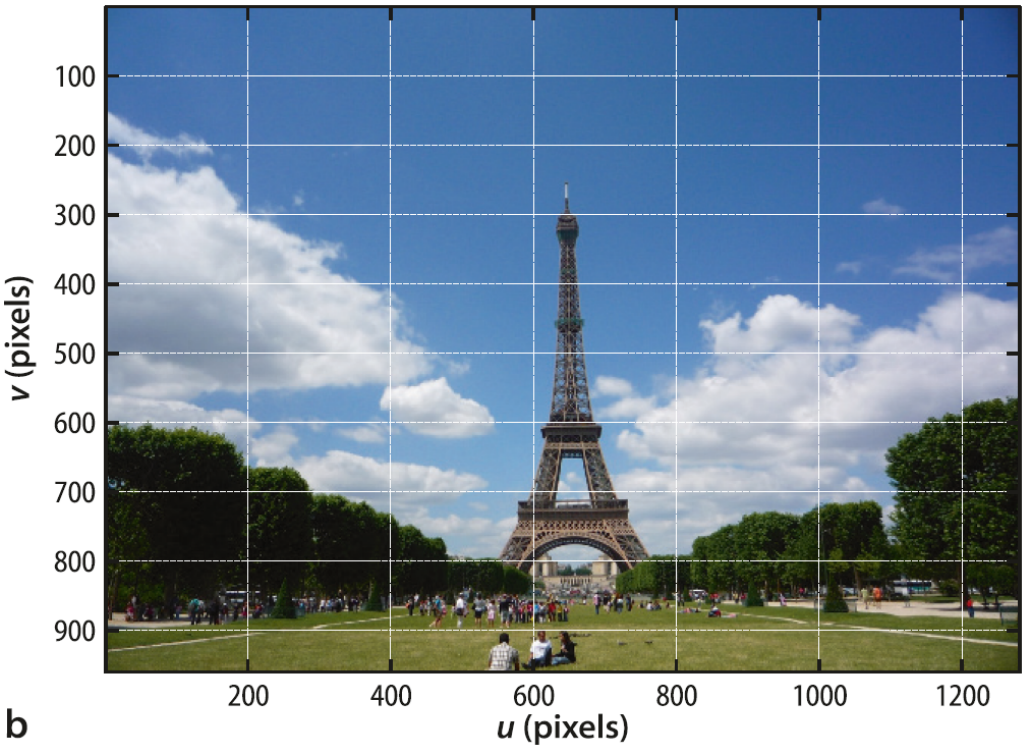
Feature descriptors (details)

The SURF algorithm is more than just a scale-invariant feature detector, it also computes a very robust *descriptor*. The descriptor is a 64-element vector that encodes the image gradient in subregions of the support region in a way which is invariant to brightness, scale and rotation. This enables feature descriptors to be unambiguously matched to a descriptor of the same world point in another image even if their scale and orientation are quite different. The difference in position, scale and orientation of the matched features gives some indication of the relative camera motion between the two views. Matching features between scenes is crucial to the problems that we will address in the next chapter.

Detectors versus descriptors. When matching world feature points, or landmarks, between different views we must first *find* points that are distinctive. This is the job of the detector and results in a coordinate (u, v) and perhaps a scale factor or orientation. The second task is to *describe* the region around the point in a way that allows it to be matched as decisively as possible with the region around the corresponding point in the other view. This is the descriptor which is typically a long vector formed from pixel values, histograms, gradients, histograms of gradient and so on. There are many detectors to choose from: Harris and variants, Shi-Tomasi, FAST, AGAST, MSER etc.; as well as many descriptors: ORB, BRISK, FREAK, CenSurE (aka STAR), HOG, ACF etc. Some algorithms such as SIFT and SURF define both a detector and a descriptor. The SIFT descriptor is a form of HOG descriptor.

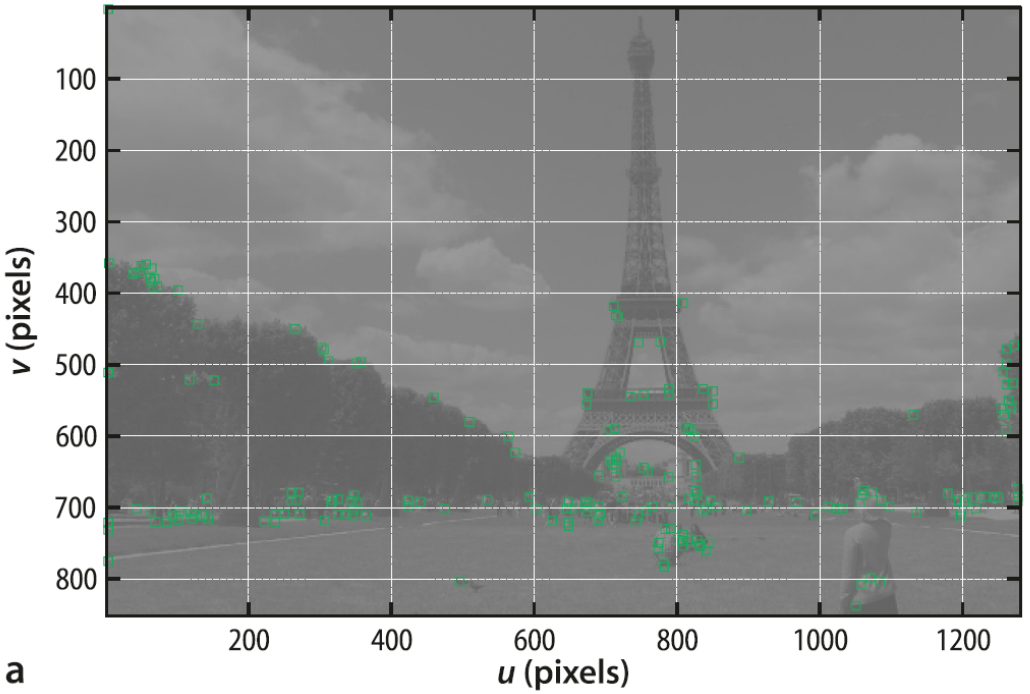
Feature matching

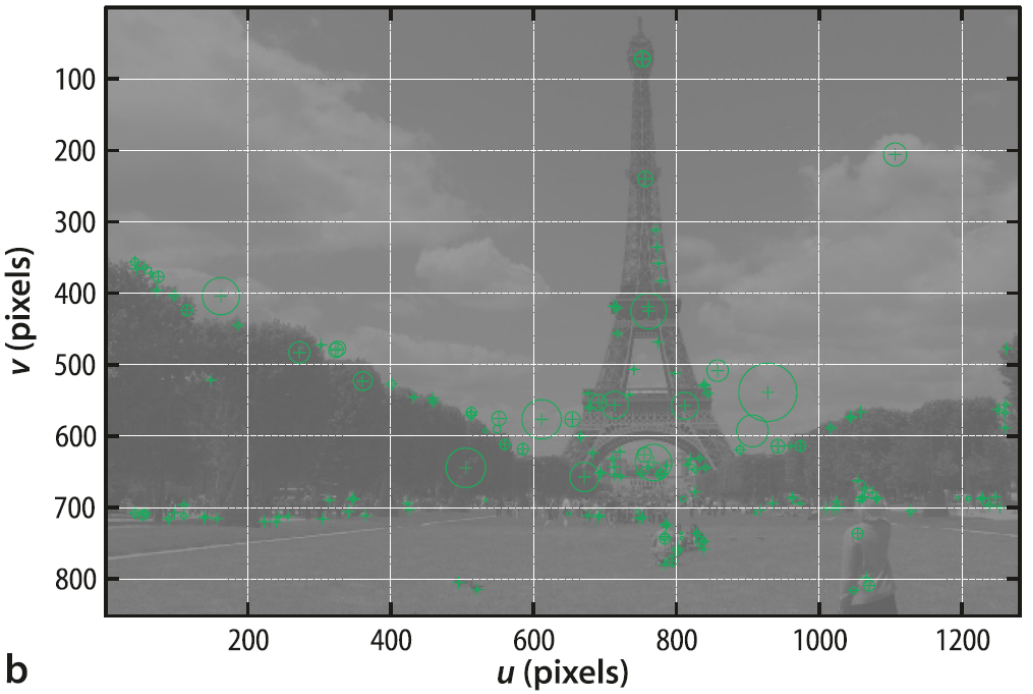




Feature matching

Fig. 14.2. Corner features computed for Fig. 14.1a. **a** Harris corner features; **b** SURF corner features showing scale





Feature matching

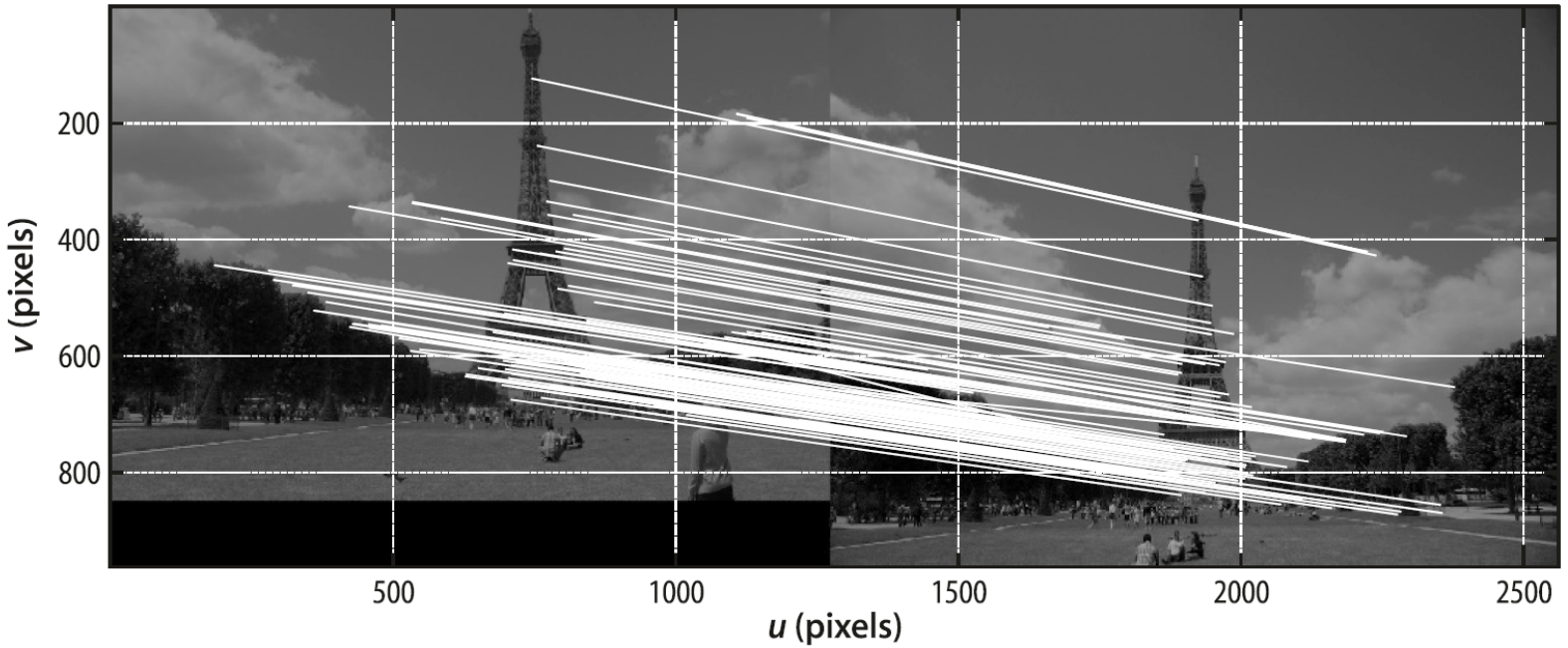


Fig. 14.3.
Feature matching. Subset (100 out of 1 664) of matches based on SURF descriptor similarity. We note that a few are clearly incorrect

Application: Perspective correction

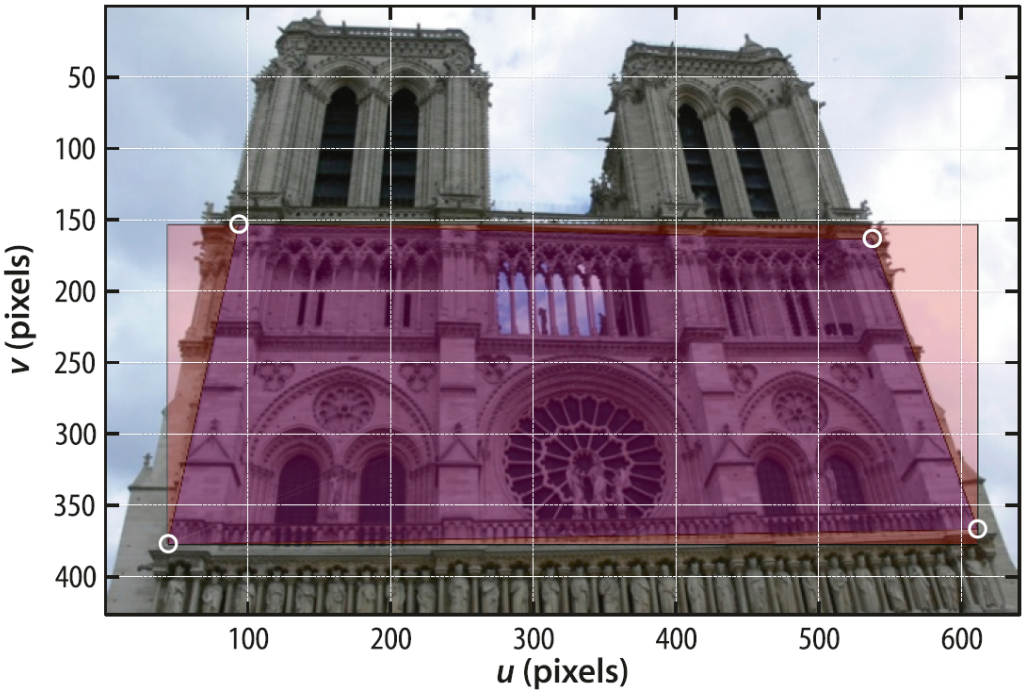


Fig. 14.45. Photograph taken from the ground shows the effect of foreshortening which gives the building a trapezoidal appearance (also known as keystone distortion). Four points on the approximately planar face of the building have been manually picked as indicated by the *white ○-markers* (Notre Dame de Paris)

Application: Perspective correction

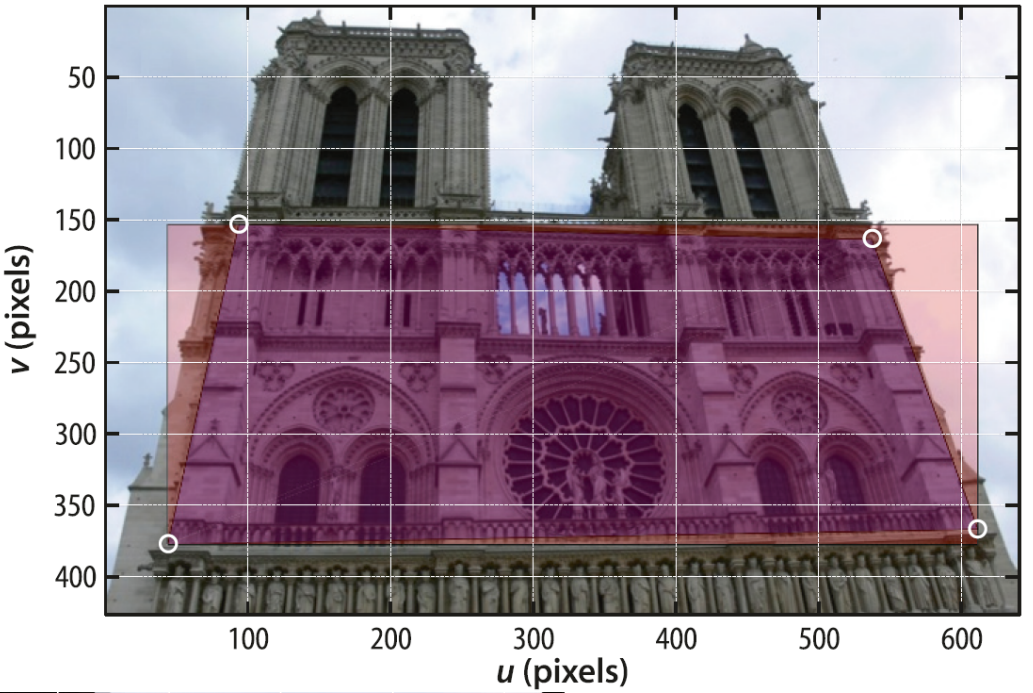


Fig. 14.45.
Photograph taken from the ground shows the effect of foreshortening which gives the building a trapezoidal appearance (also known as keystone distortion). Four points on the approximately planar face of the building have been manually picked as indicated by the *white ○-markers* (Notre Dame de Paris)

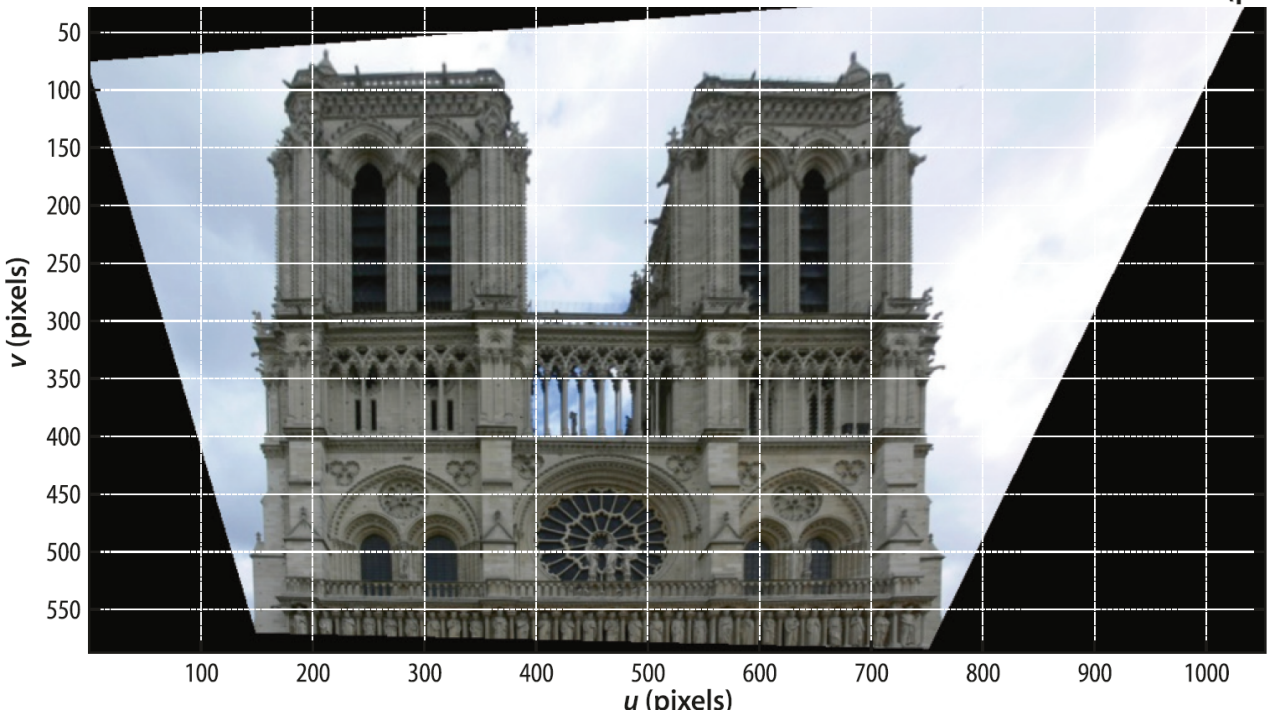


Fig. 14.46.
A fronto-parallel view synthesized from Fig. 14.45. The image has been transformed so that the marked points become the corners of a rectangle in the image

Application: Mosaicing

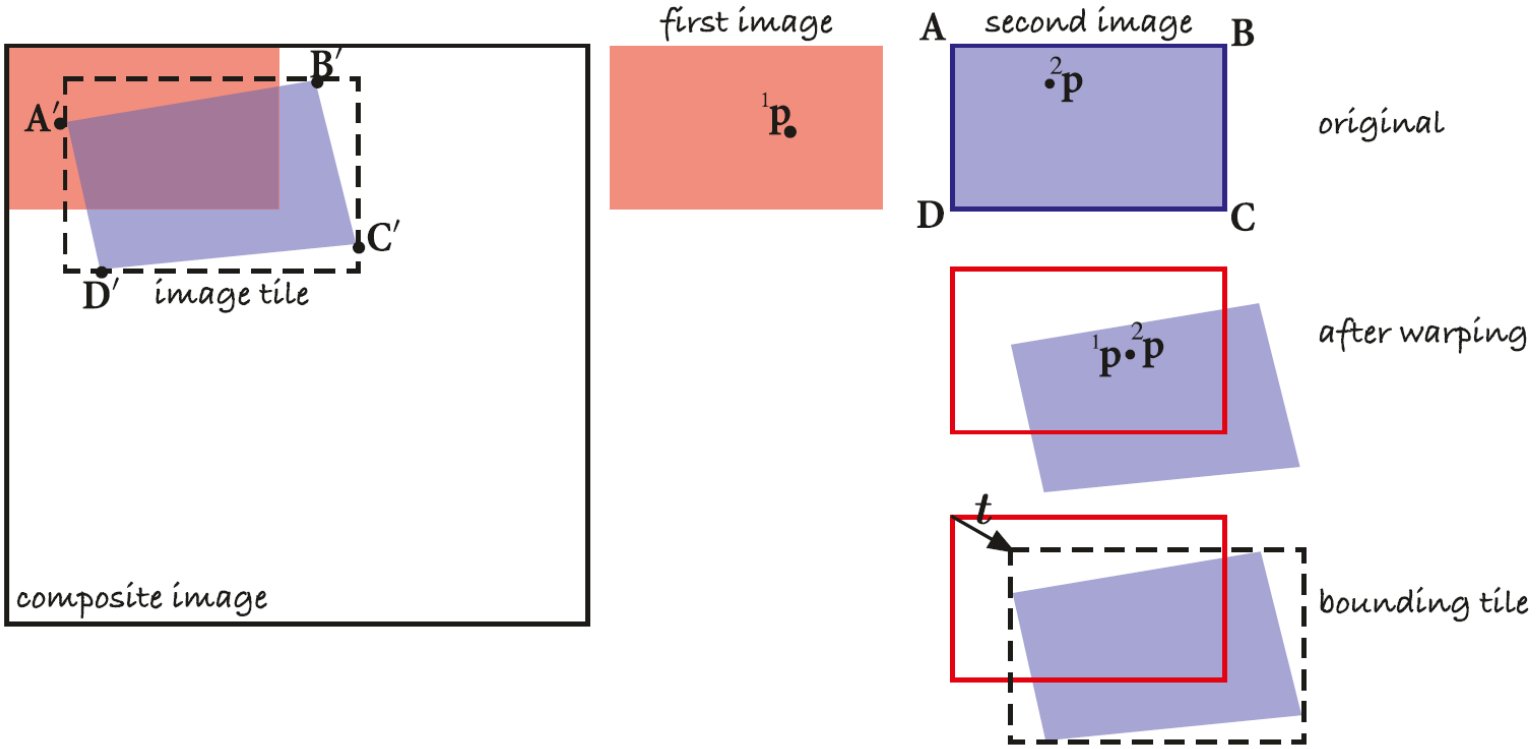


Fig. 14.48.
The first image in the sequence is shown as *red*, the second as *blue*. The second image is warped into the image tile and then blended into the composite image

Application: Mosaicing

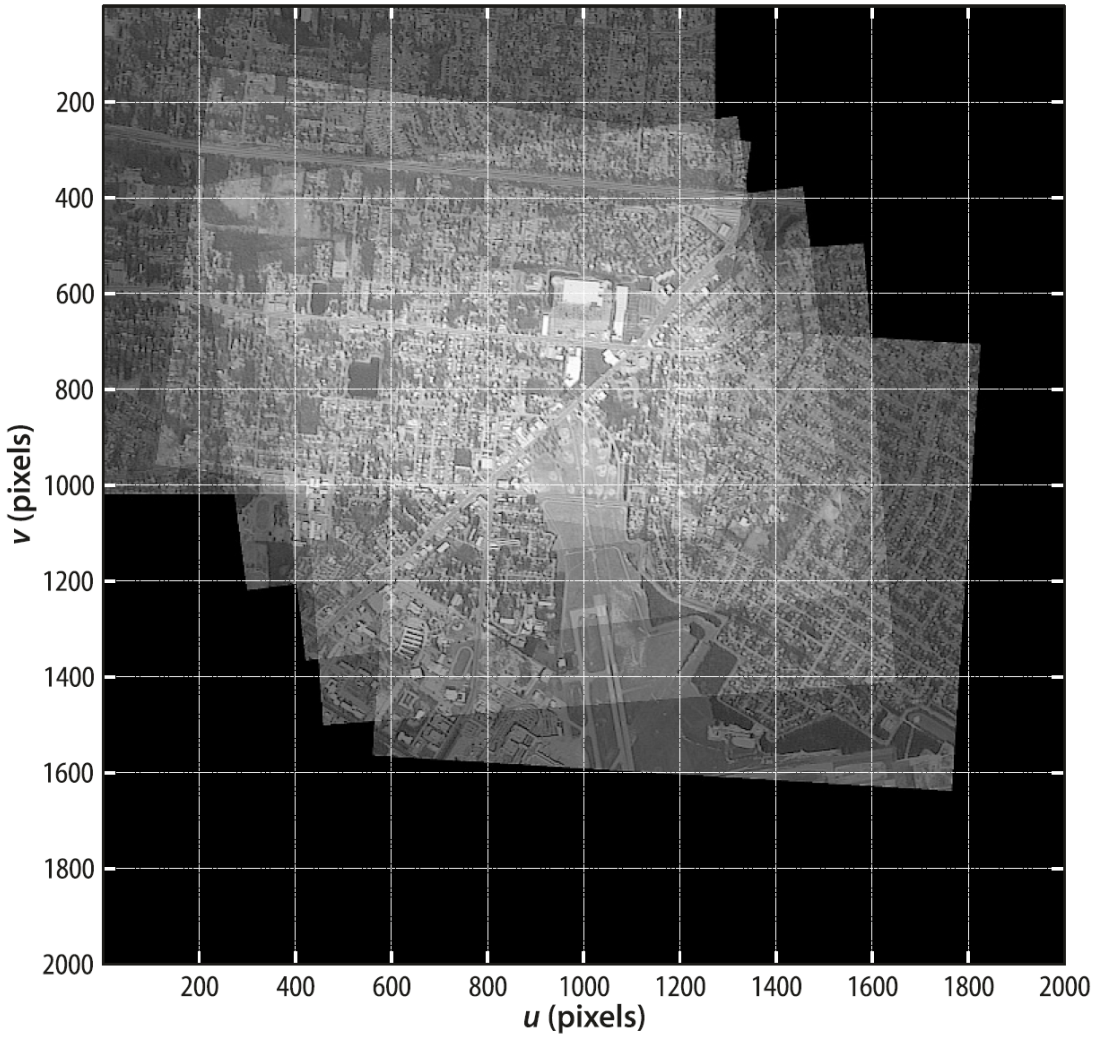


Fig. 14.49.
Example image mosaic. At the bottom of the frame we can clearly see three overlapping views of the airport runway which shows good alignment between the frames

Application: Visual odometry

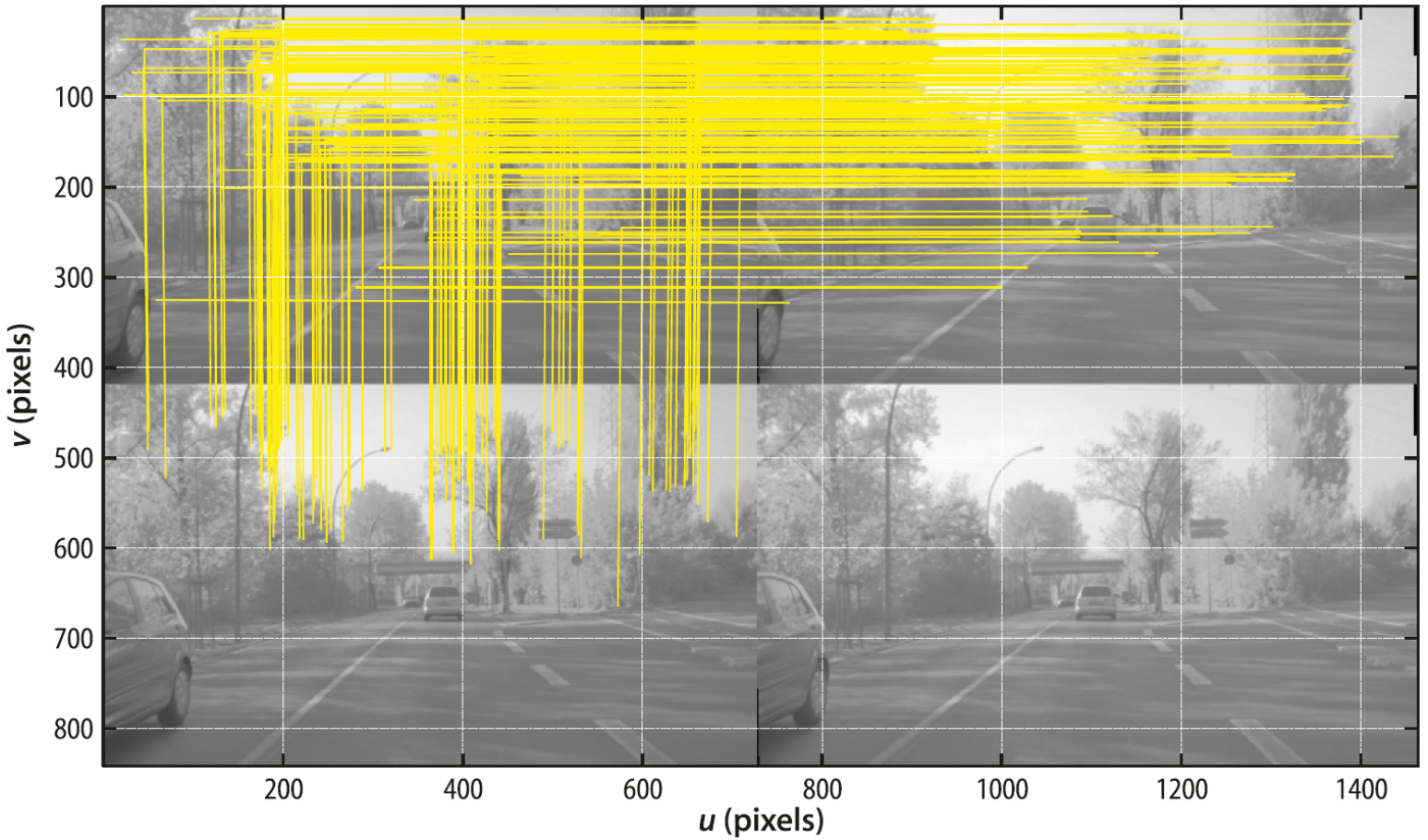


Fig. 14.57. Feature correspondence for visual odometry. The top row is a stereo pair at the current time step, and the bottom row is a stereo pair at the previous time step. Epiplolar consistent correspondences between three of the image images are shown in *yellow*

Outline

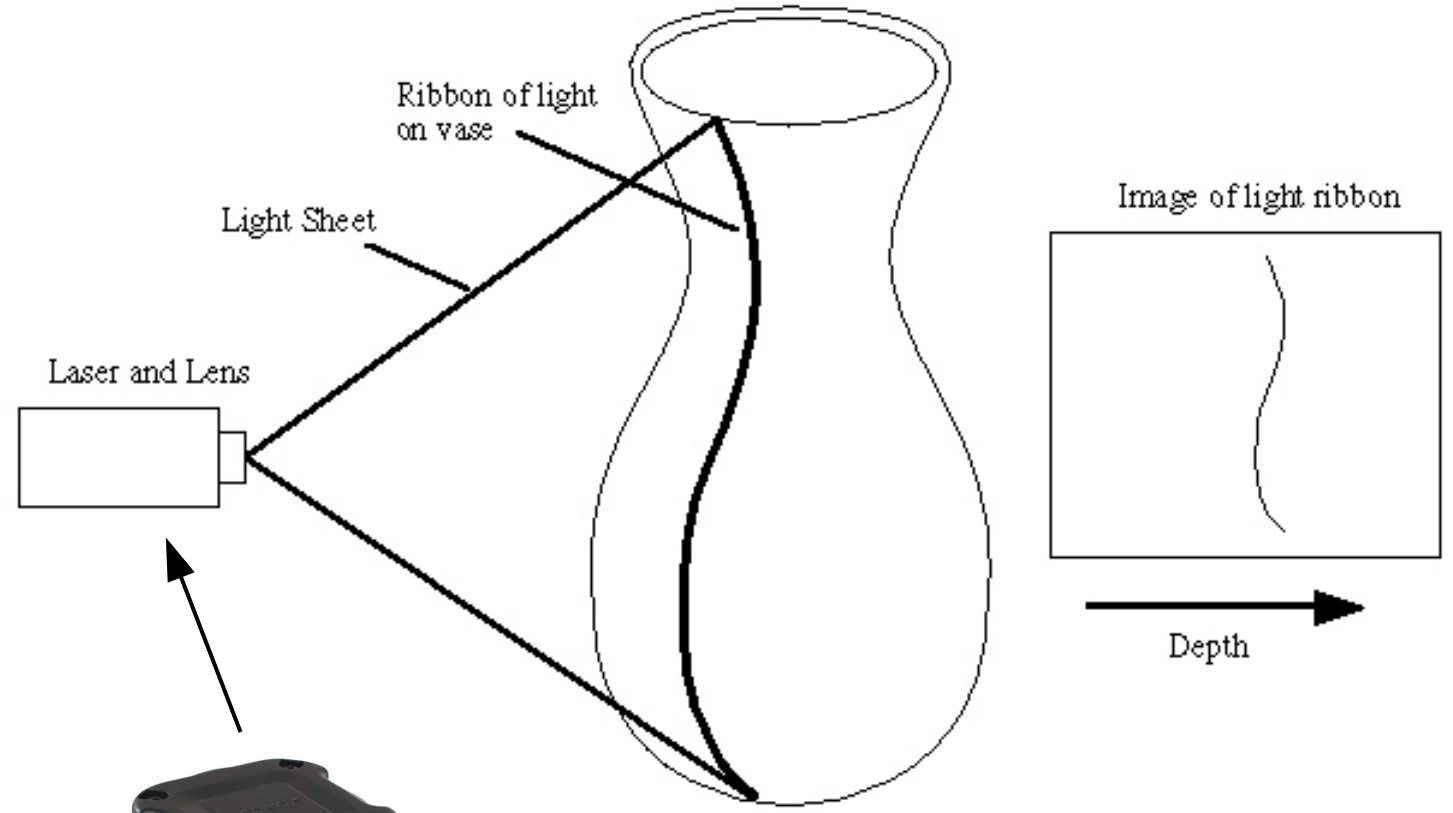
✔ 2-D image alignment
- Feature descriptors: Corner detectors, SIFT, SURF
- Applications

3-D perception – point clouds
- Sensors and strategies

Aligning point clouds
- Iterative closest point

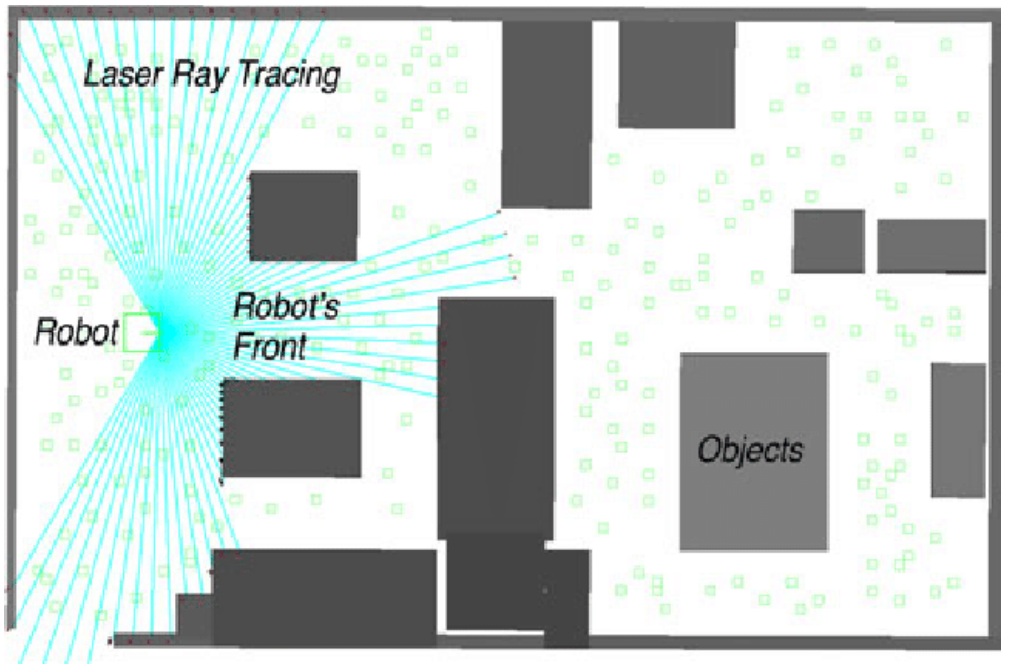
Extracting primitive shapes
- Surface normals (planes)
- RANSAC
- More shapes (time permitting)

Laser range scanner

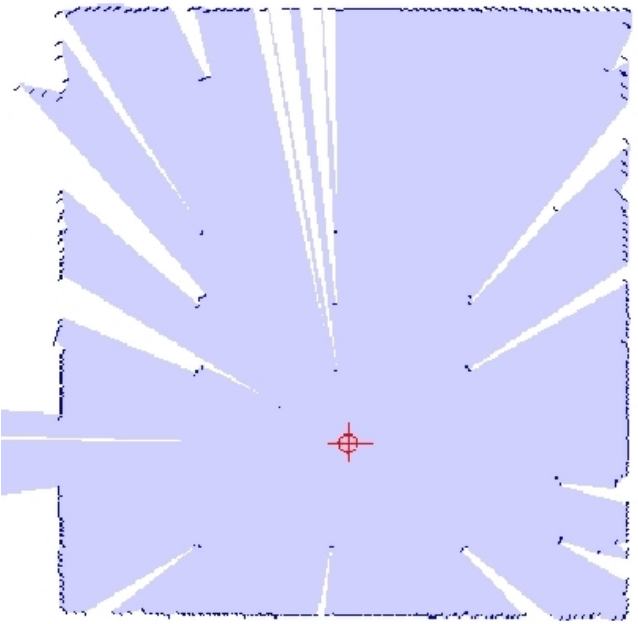


Hokuyo UTM-30LX-EW Scanning Laser Range Finder

Laser range scanner

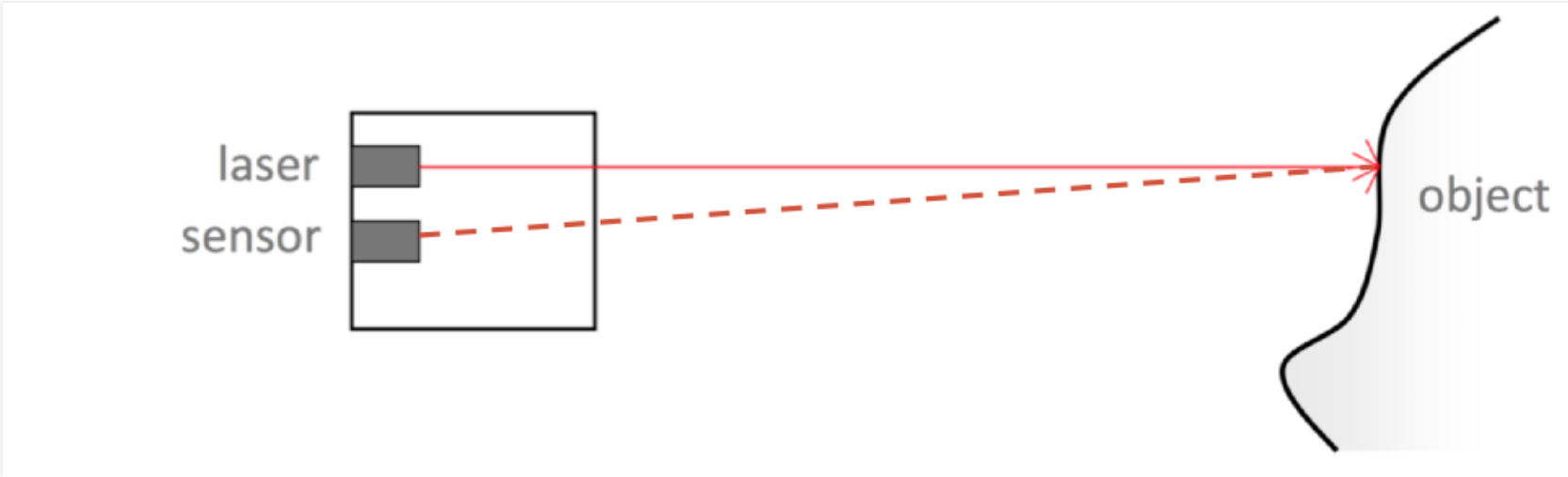


Scan geometry



2D map created using laser range scanner

Laser range scanner



1. Emit a short short pulse of laser
2. Capture the reflection.
3. Measure the time it took to come back.
4. Need a very fast clock.
5. Main advantage: can be done over long distances.
6. Used in terrain scanning.

Slide from Course INF 555 slides, Ecole Polytechnique, Paris

Laser range scanner

source: Michael Wand

Stereo vision

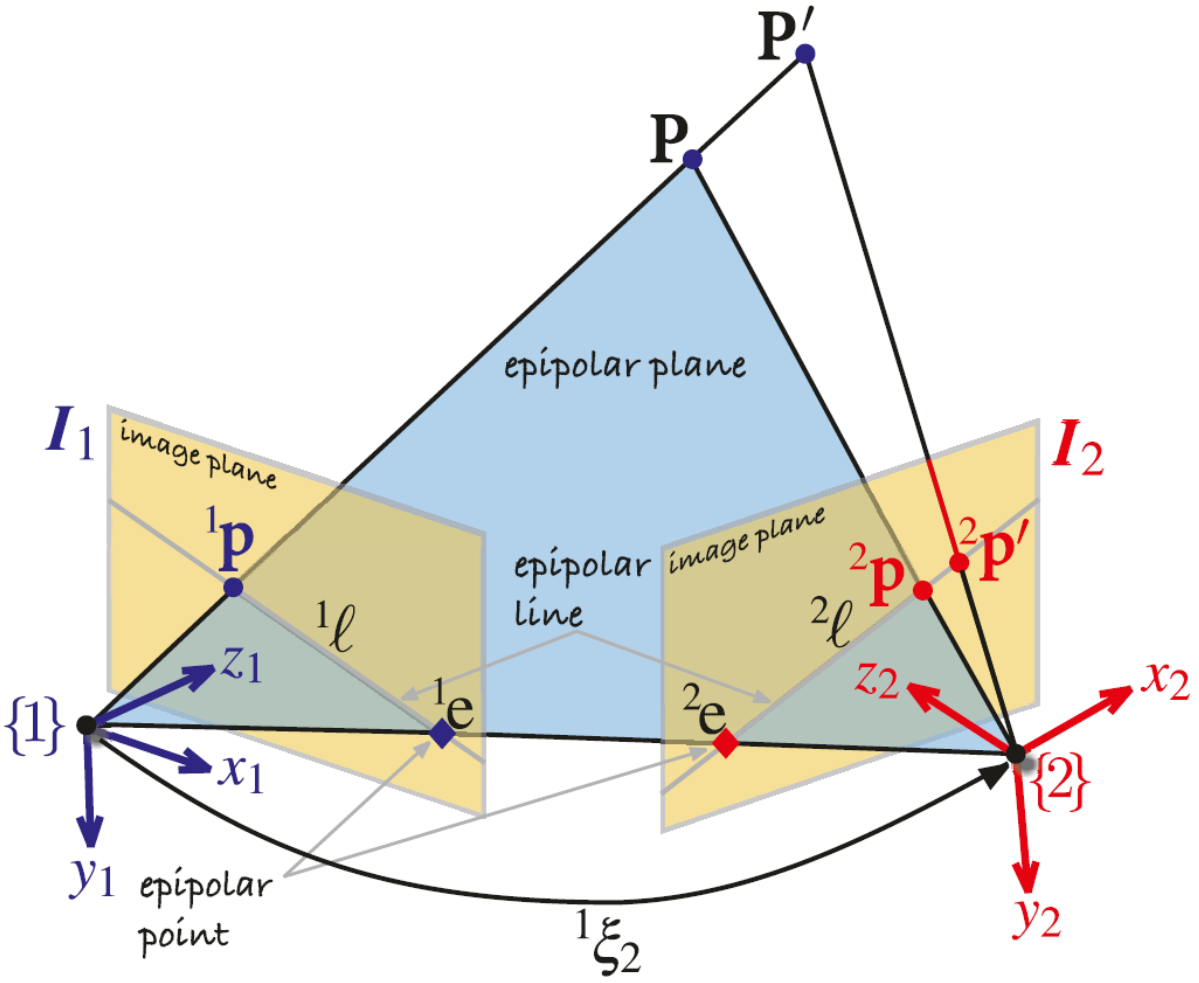
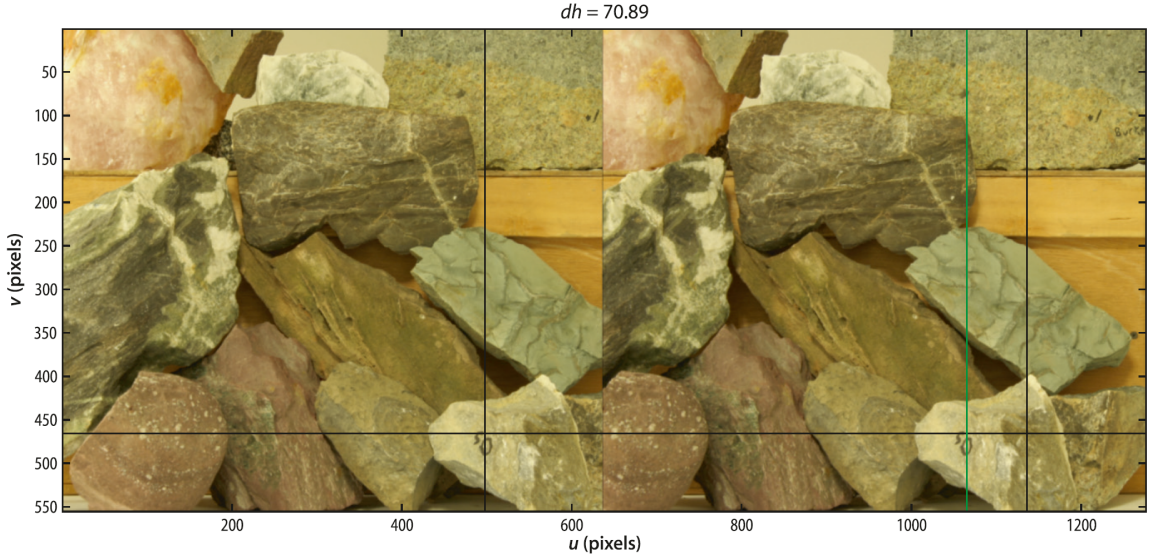


Fig. 14.17.
Epipolar geometry for stereo vision. We can see clearly that as the depth of the world point increases, from **P** to **P**′, the projection moves along the epipolar line in the second image plane

Stereo vision



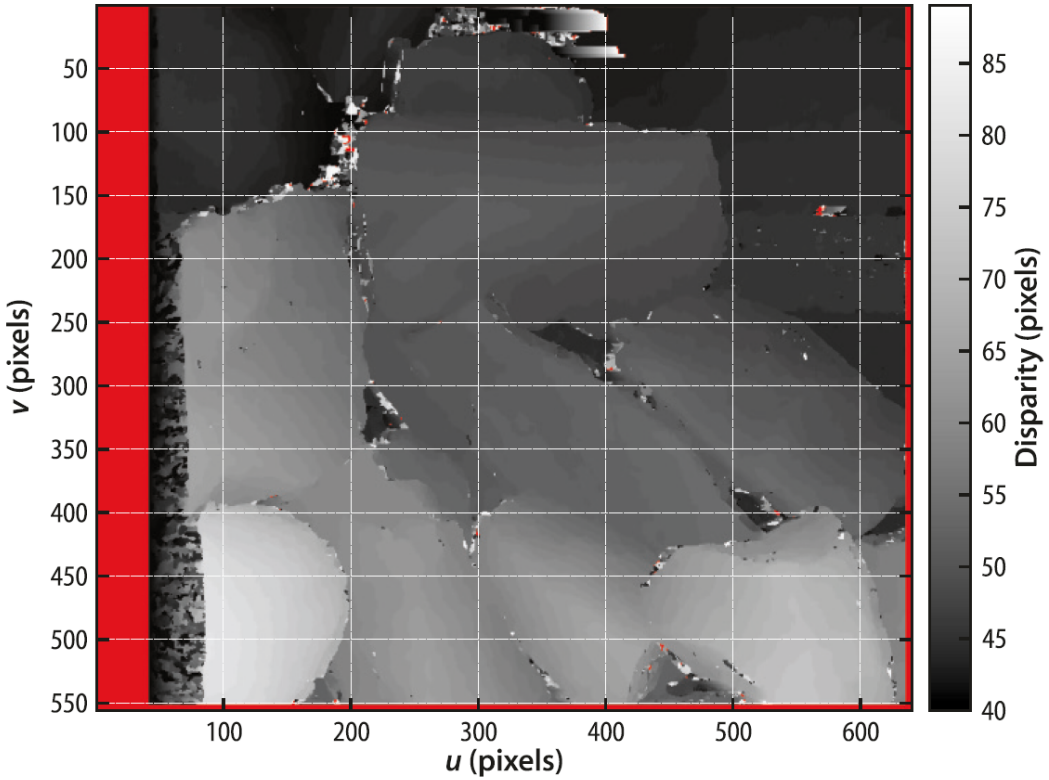


Fig. 14.22.
Disparity image for the rock pile stereo pair, where *brighter* means higher disparity or shorter range. *Red* indicates `Inf` or `NaN` values in the result where disparity could not be computed. Note the quantization in grey levels since we search for disparity in steps of one pixel

Stereo vision

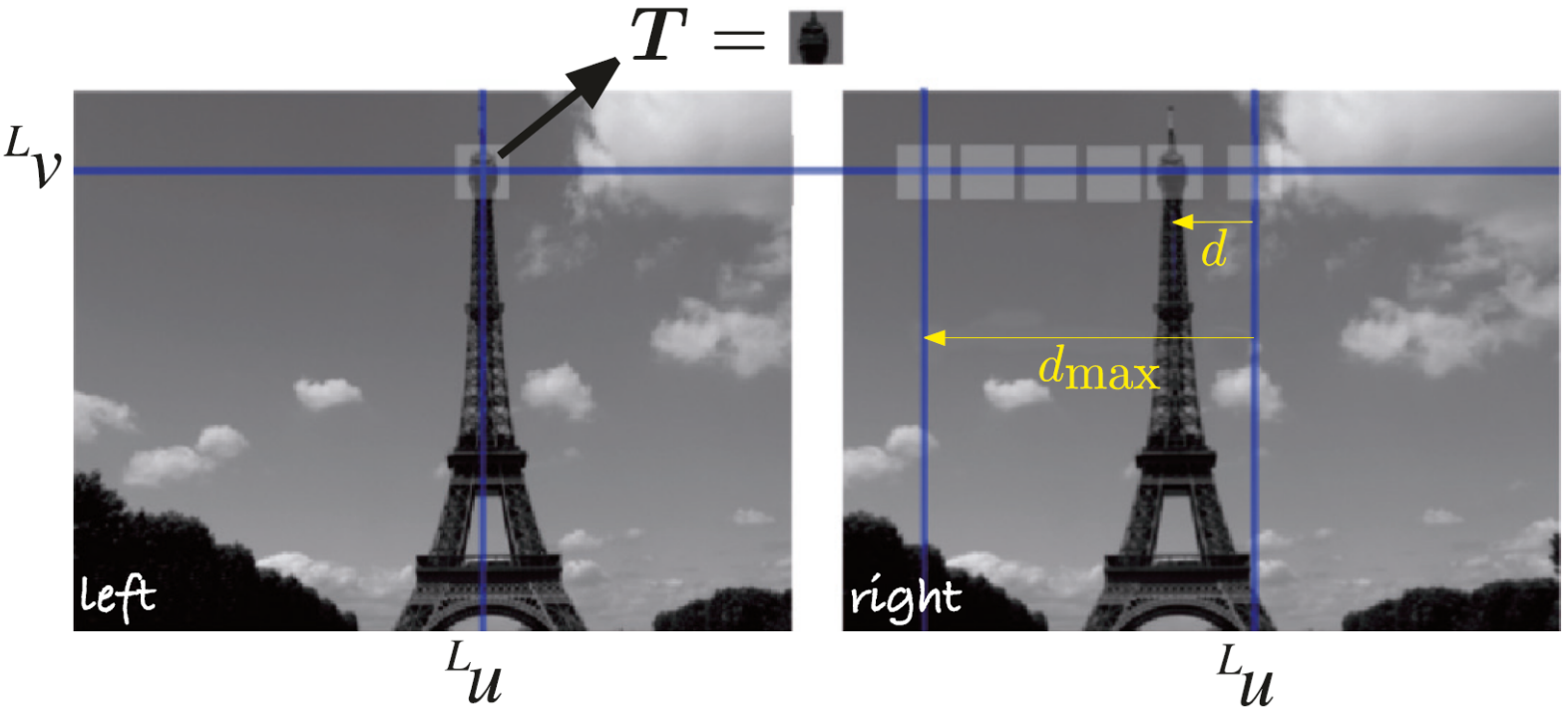


Fig. 14.21.
Stereo matching. A search window in the right image, starting at $u = {}^{L}u$, is moved leftward along the epipolar line $v = {}^{L}v$ until it matches the *template* window T from the left image

Stereo vision

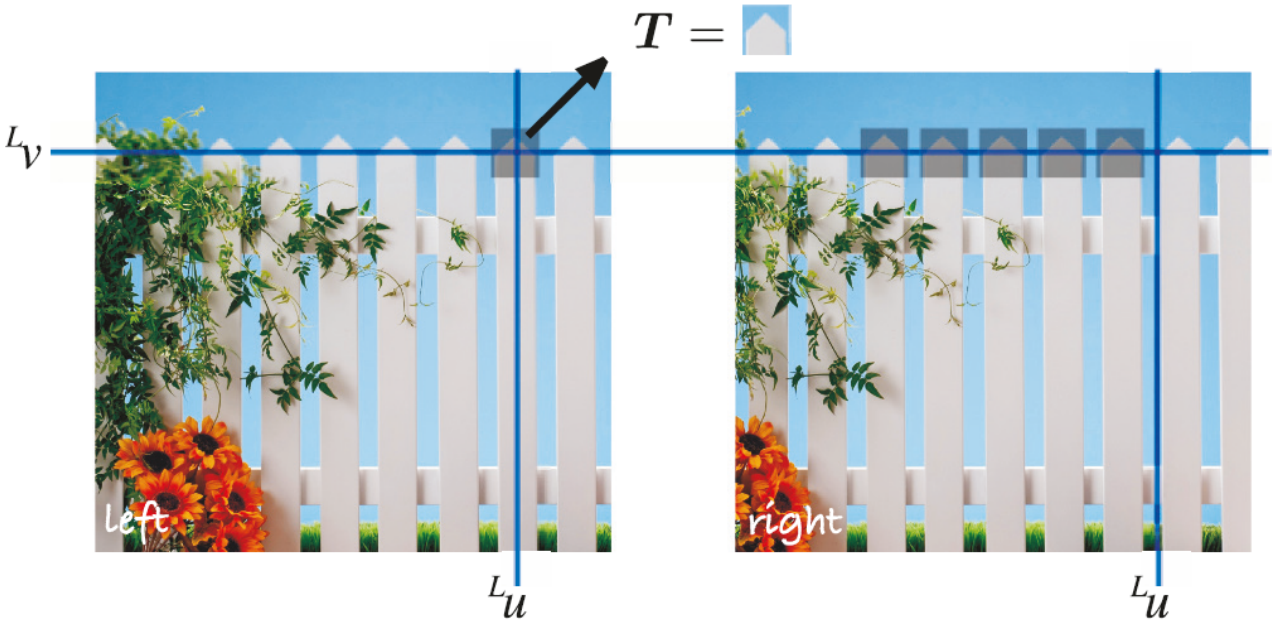


Fig. 14.25. Picket fence effect. The template will match well at a number of different disparities. This problem occurs in any scene with repeating patterns

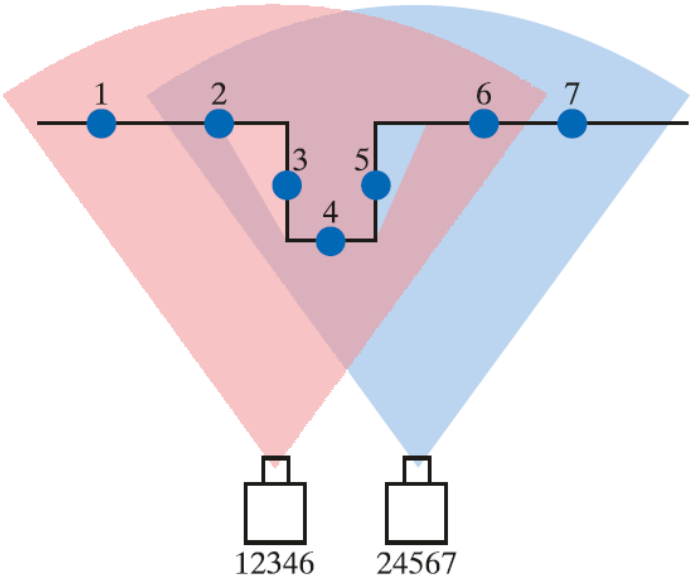


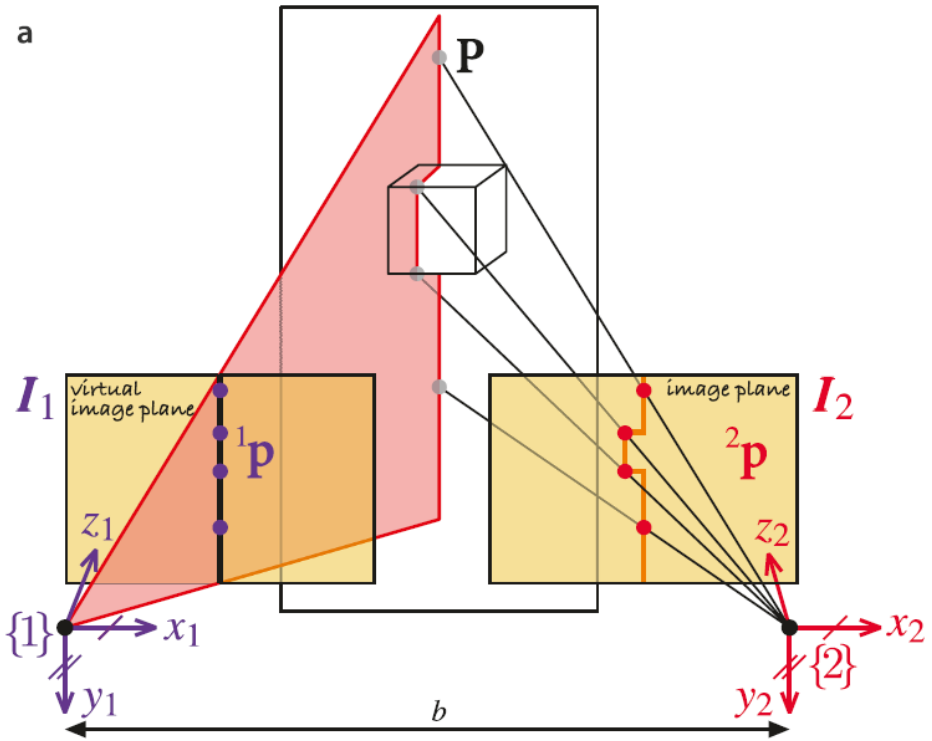
Fig. 14.26. Occlusion in stereo vision. The field of view of the two cameras are shown as *colored sectors*. *Points 1* and *7* fall outside the overlapping view area and are seen by only one camera each. *Point 5* is occluded from the left camera and *point 3* is occluded from the right camera. The order of points seen by each camera is given underneath it

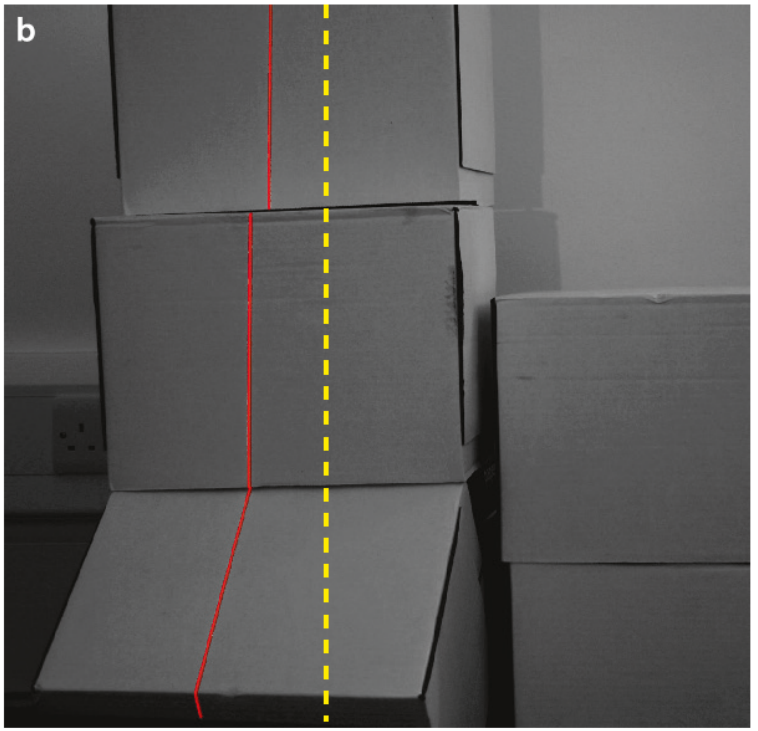
Structured light

Fig. 14.43. a Geometry of structured light showing a light projector on the left and a camera on the right; four corresponding points are marked with *dots* on the left and right images and the scene; **b** a real structured light scenario showing the light stripe falling on a simple 3D scene. The superimposed *dashed line* represents the stripe position for a plane at infinity. Disparity, left shift of the projected line relative to the dashed line, is inversely proportional to depth

14.6 Structured Light

An old, yet simple and effective, method of estimating the 3D structure of a scene is structured light. It is conceptually similar to stereo vision but we replace the left camera with a projector that emits a vertical plane of light as shown in Fig. 14.43a. This is equivalent, in a stereo system, to a left-hand image that is a vertical line. The image of the line projected onto the surface viewed from the right-hand camera will be a distorted version of the line, as shown in Fig. 14.43b. The disparity between the virtual left-hand image and the actual right-hand image is a function of the depth of points along the line.





Structured light

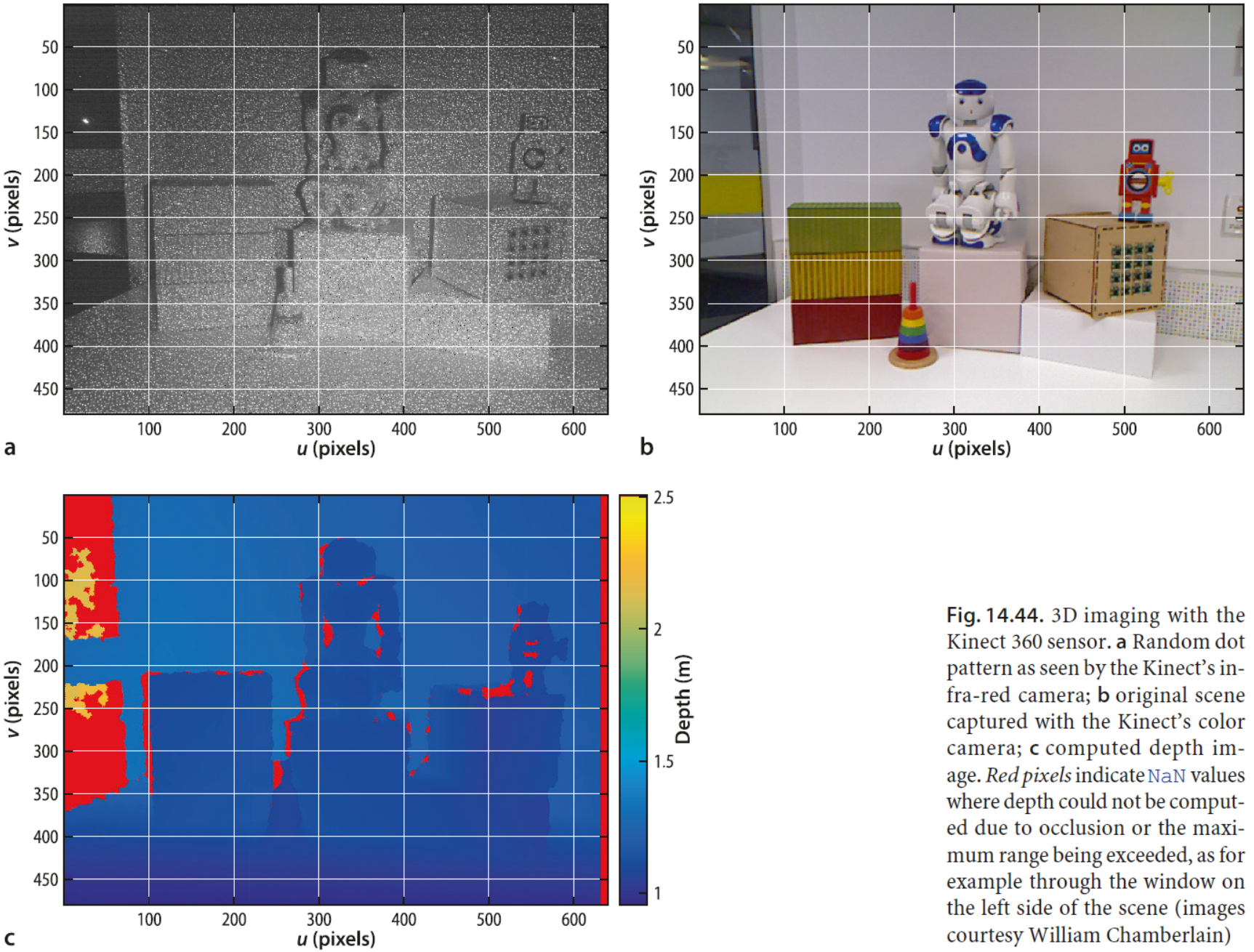
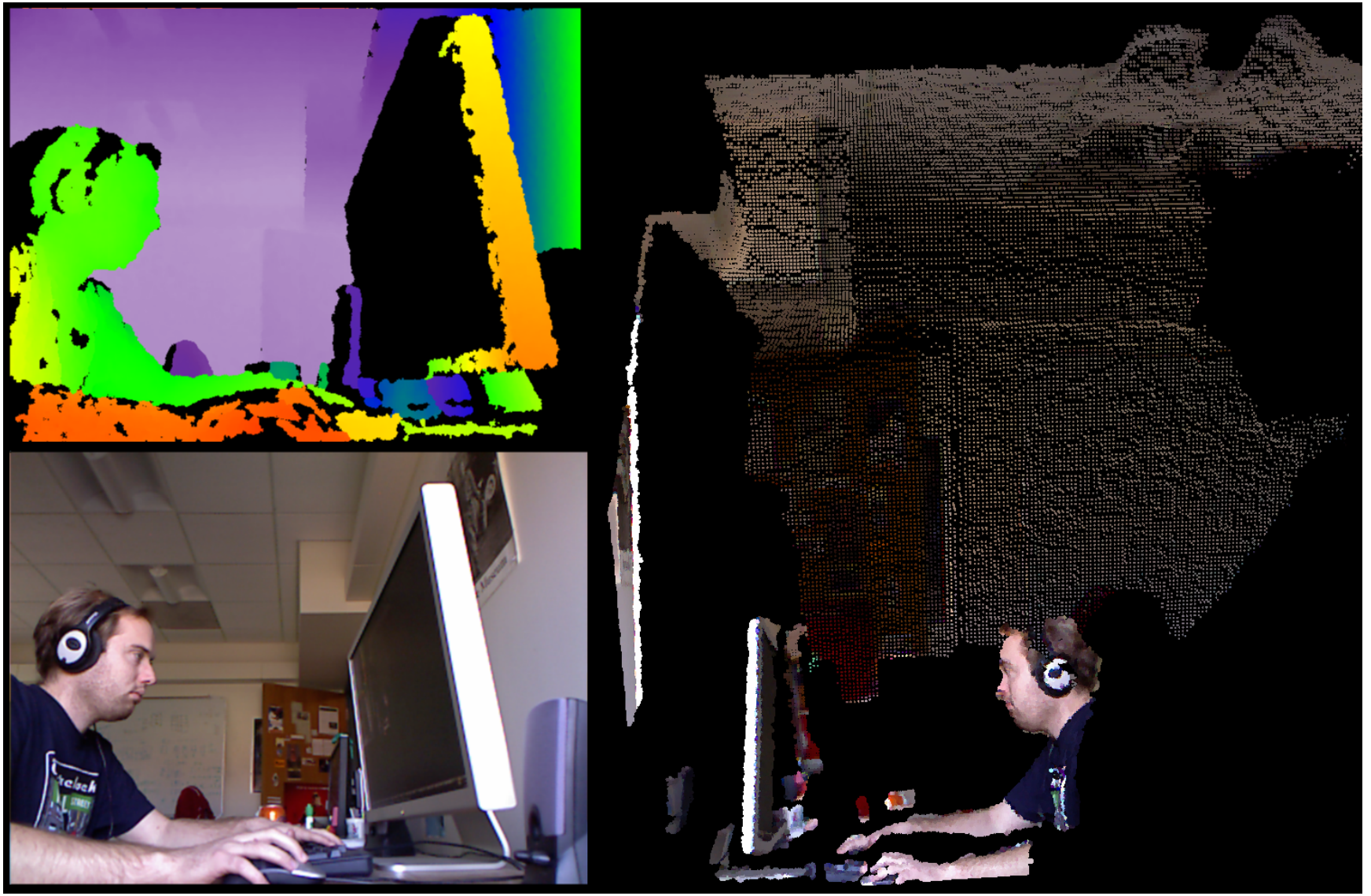


Fig. 14.44. 3D imaging with the Kinect 360 sensor. **a** Random dot pattern as seen by the Kinect's infra-red camera; **b** original scene captured with the Kinect's color camera; **c** computed depth image. *Red pixels* indicate `NaN` values where depth could not be computed due to occlusion or the maximum range being exceeded, as for example through the window on the left side of the scene (images courtesy William Chamberlain)

RGB-D image



Outline

✔ 2-D image alignment
- Feature descriptors: Corner detectors, SIFT, SURF
- Applications

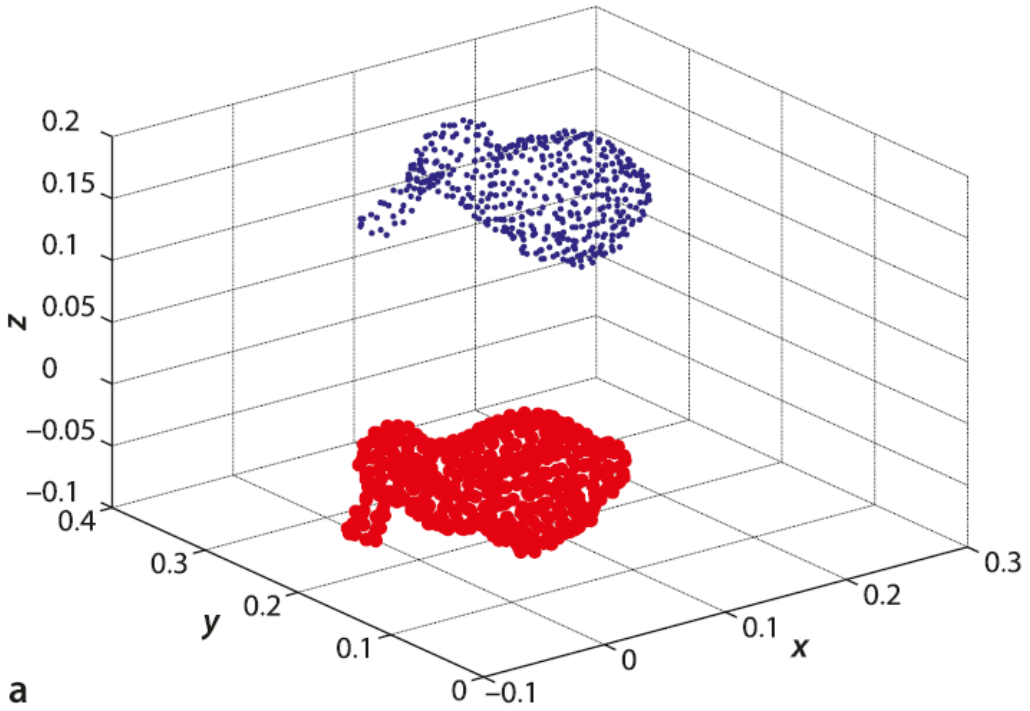
✔ 3-D perception – point clouds
- Sensors and strategies

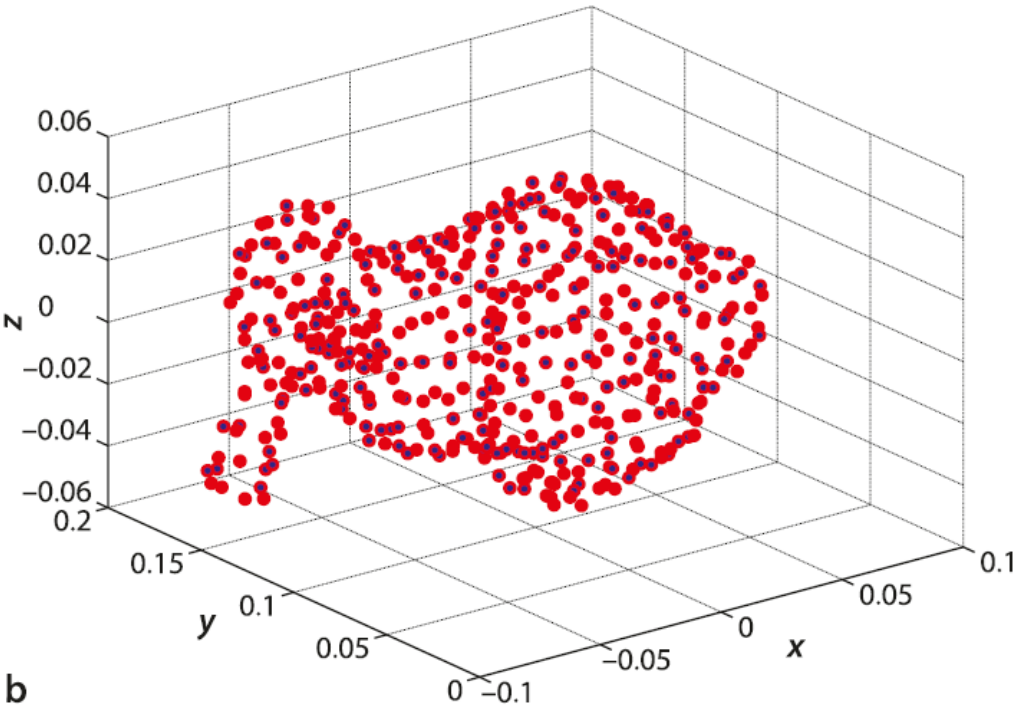
Aligning point clouds
- Iterative closest point

Extracting primitive shapes
- Surface normals (planes)
- RANSAC
- More shapes (time permitting)

Aligning point clouds

Fig. 14.42. Iterated closest point (ICP) matching of two point clouds: model (*red*) and data (*blue*) **a** before registration, **b** after registration; observed data points have been transformed to the model coordinate frame using the inverse of the identified transformation (Stanford bunny model courtesy Stanford Graphics Laboratory)





Iterative closest point (ICP)

- ICP is a powerful algorithm for calculating the displacement between scans.
- The major problem is to determine the correct data associations.
- Given the correct data associations, the transformation can be computed efficiently using SVD.

Iterative closest point (ICP)

- Given: two corresponding point sets:

$$X = \{x_1, \ldots, x_n\}$$

$$P = \{p_1, \ldots, p_n\}$$

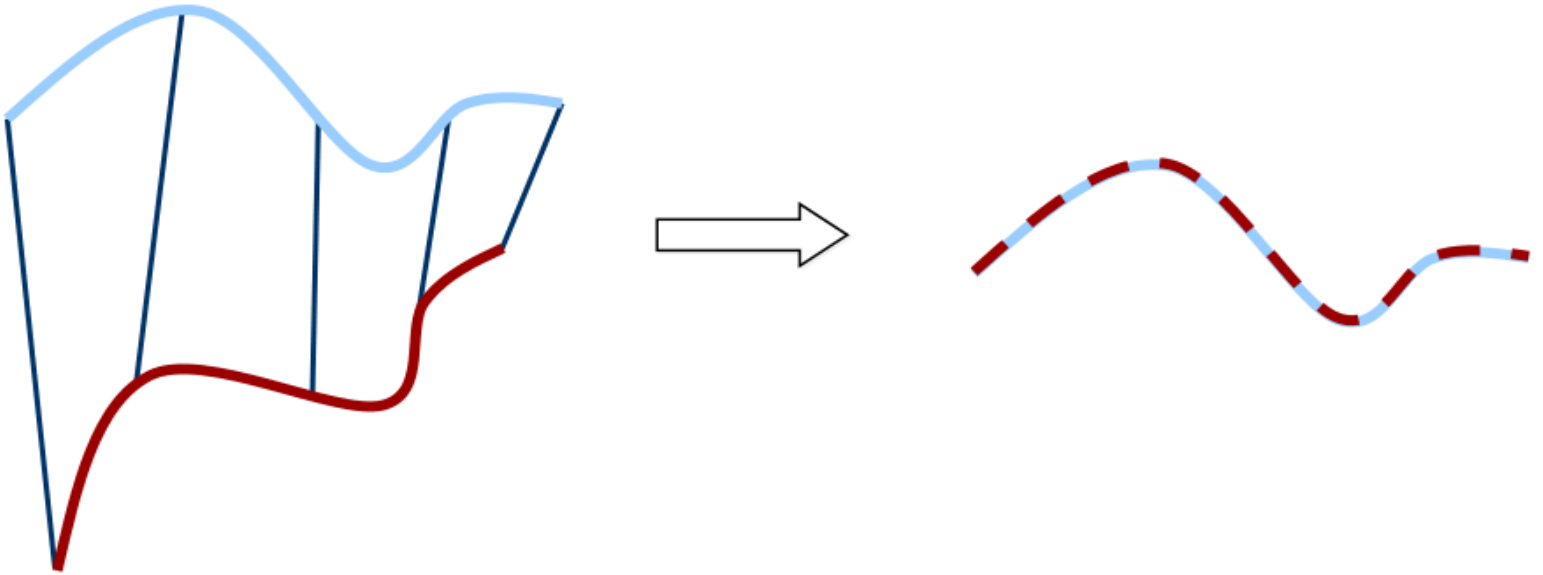
- Wanted: translation t and rotation R that minimizes the sum of the squared error:

$$E(R,t) = \frac{1}{N_p} \sum_{i=1}^{N_p} ||x_i - Rp_i - t||^2$$

Where x_i and p_i are corresponding points.

ICP: Key idea

- If the correct correspondences are known, the correct relative rotation/translation can be calculated in closed form.



ICP step 1: Center the two point clouds

$$\mu_x = \frac{1}{N_x}\sum_{i=1}^{N_x} x_i \quad \text{and} \quad \mu_p = \frac{1}{N_p}\sum_{i=1}^{N_p} p_i$$

are the centers of mass of the two point sets.

Idea:

- Subtract the corresponding center of mass from every point in the two point sets before calculating the transformation.
- The resulting point sets are:

$$X' = \{x_i - \mu_x\} = \{x'_i\}$$

and

$$P' = \{p_i - \mu_p\} = \{p'_i\}$$

ICP step 2: Use SVD to get minimizing *t* and *R*

Let $W = \sum_{i=1}^{N_p} x'_i p'^T_i$

denote the singular value decomposition (SVD) of W by:

$$W = U \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} V^T$$

where $U, V \in \mathbb{R}^{3\times 3}$ are unitary, and

$\sigma_1 \geq \sigma_2 \geq \sigma_3$ are the singular values of W.

ICP step 2: Use SVD to get minimizing *t* and *R*

Theorem (without proof):

If rank(W) = 3, the optimal solution of E(R,t) is unique and is given by:

$$R = UV^T$$

$$t = \mu_x - R\mu_p$$

The minimal value of error function at (R,t) is:

$$E(R,t) = \sum_{i=1}^{N_p} (||x_i'||^2 + ||y_i'||^2) - 2(\sigma_1 + \sigma_2 + \sigma_3)$$

ICP step 2: Use SVD to get minimizing *t* and *R*

Theorem (without proof):

If rank(W) = 3, the optimal solution of E(R,t) is unique and is given by:

$$R = UV^T$$

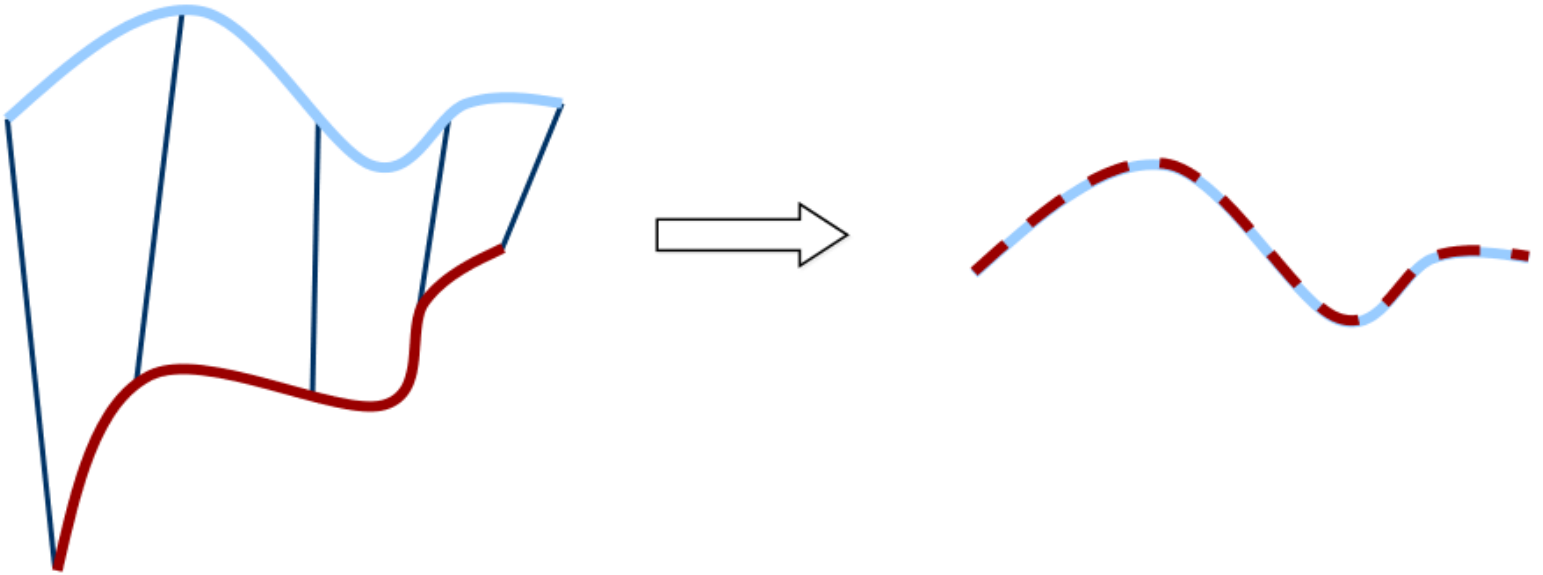
$$t = \mu_x - R\mu_p$$

The minimal value of error function at (R,t) is:

$$E(R,t) = \sum_{i=1}^{N_p} (||x_i'||^2 + ||y_i'||^2) - 2(\sigma_1 + \sigma_2 + \sigma_3)$$

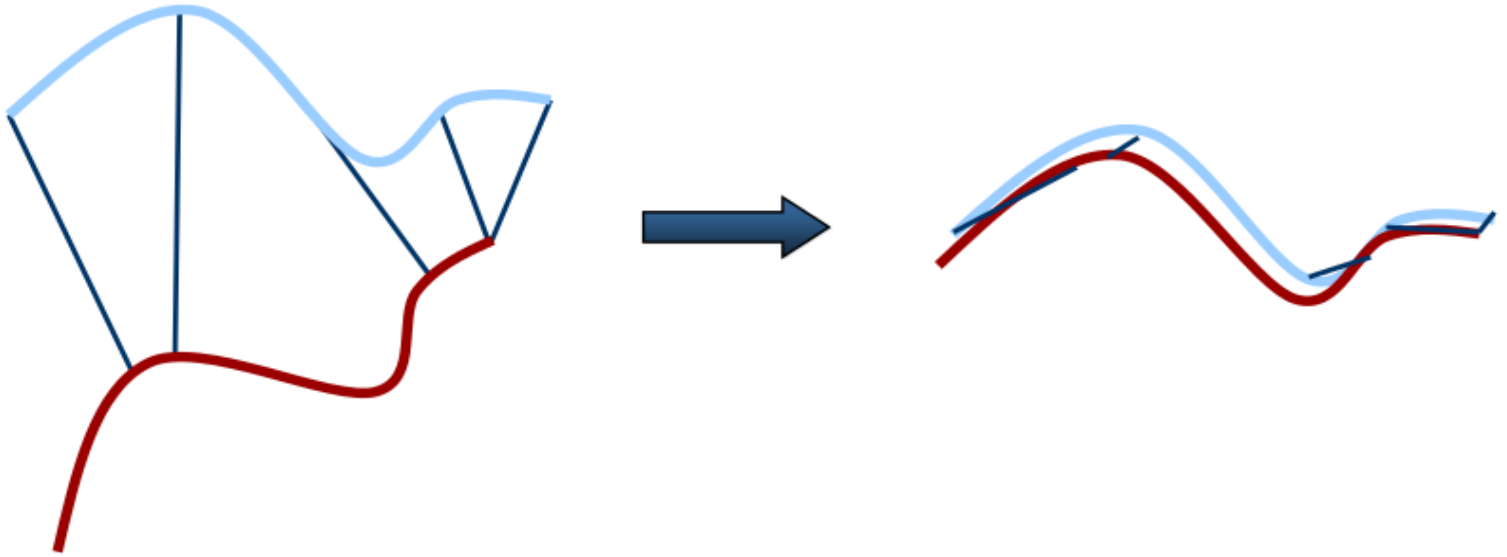
ICP: Iterative data association

- If the correct correspondences are known, the correct relative rotation/translation can be calculated in closed form.



ICP: Iterative data association

- If correct correspondences are not known, it is generally impossible to determine the optimal relative rotation/translation in one step



8

ICP: Pseudocode

Input: Two point sets, *X* and *P*

Output: Translation *t* and rotation matrix *R*
that best aligns the two point sets

1. Start with a “good” alignment
2. Repeat until *t* and *R* are small:
3. **For every point in *X*, find its closest neighbor in *P***
4. Find min *t* and *R* for that correspondence assignment
5. Translate and rotate *P* by *t* and *R*
6. Return the net translation *t* and rotation matrix *R*

Converges if the point sets are initially well-aligned
(Besl & McKay, 1992)

Outline

✔ 2-D image alignment
- Feature descriptors: Corner detectors, SIFT, SURF
- Applications

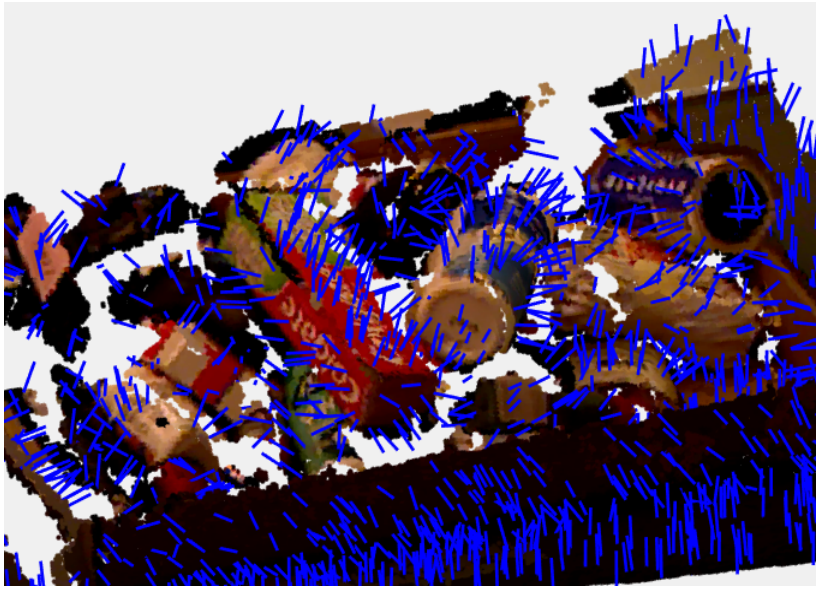
✔ 3-D perception – point clouds
- Sensors and strategies

✔ Aligning point clouds
- Iterative closest point

Extracting primitive shapes
- Surface normals (planes)
- RANSAC
- More shapes (time permitting)

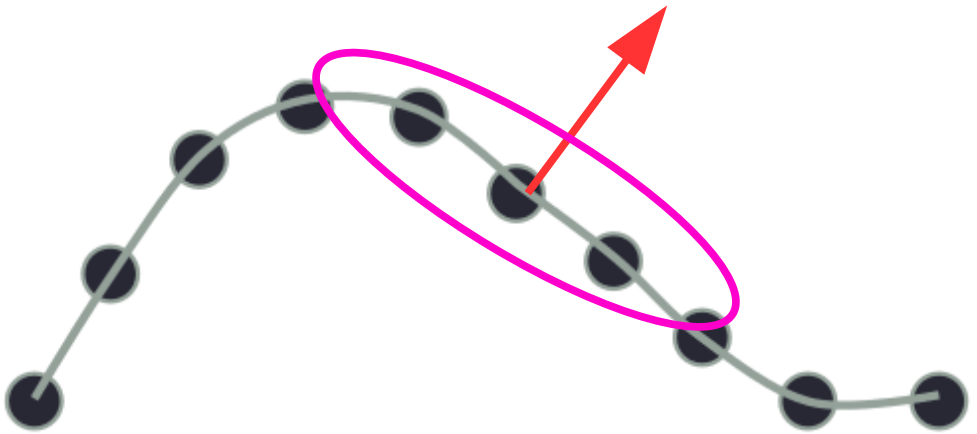
Surface normals

Point cloud



Point cloud w/ surface normals
(normals are subsampled)

Surface normals



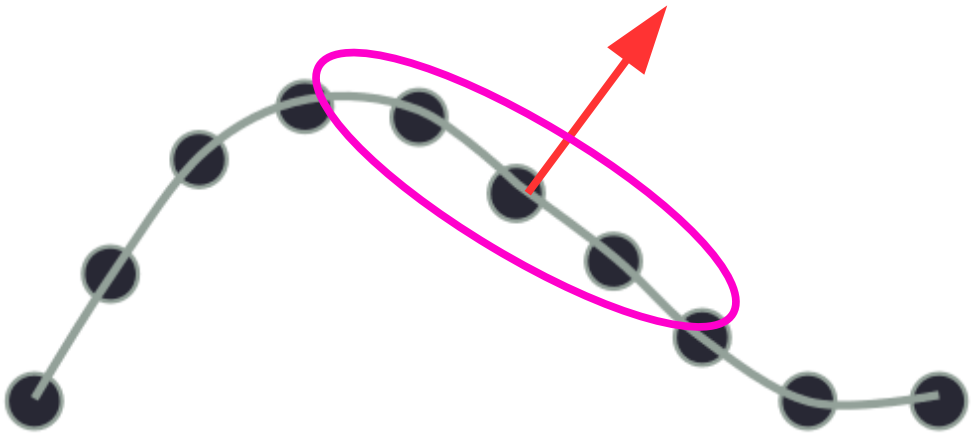
How do we calculate the surface normal given only points?

Answer:

1. Calculate the sample covariance matrix of the points within a local neighborhood of the surface normal

2. Take eigenvalues/eigenvectors of that covariance matrix

Surface normals



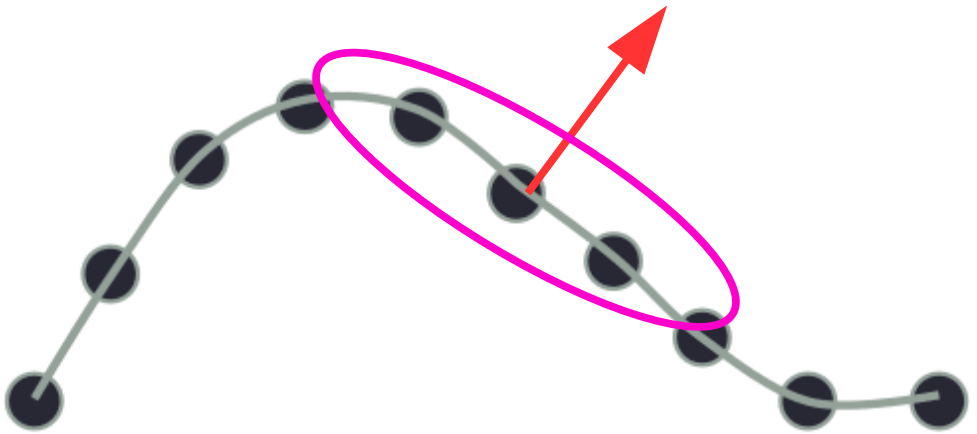
Let *C* denote the set of points in the point cloud

Suppose we want to calculate the surface normal at $x \in C$

Let $B_r(x) \subseteq \mathbb{R}^3$ denote the r-ball about *x*

$N_r(x) = B_r(x) \cap C$ is the set of points in the cloud within *r* of *x*

Surface normals

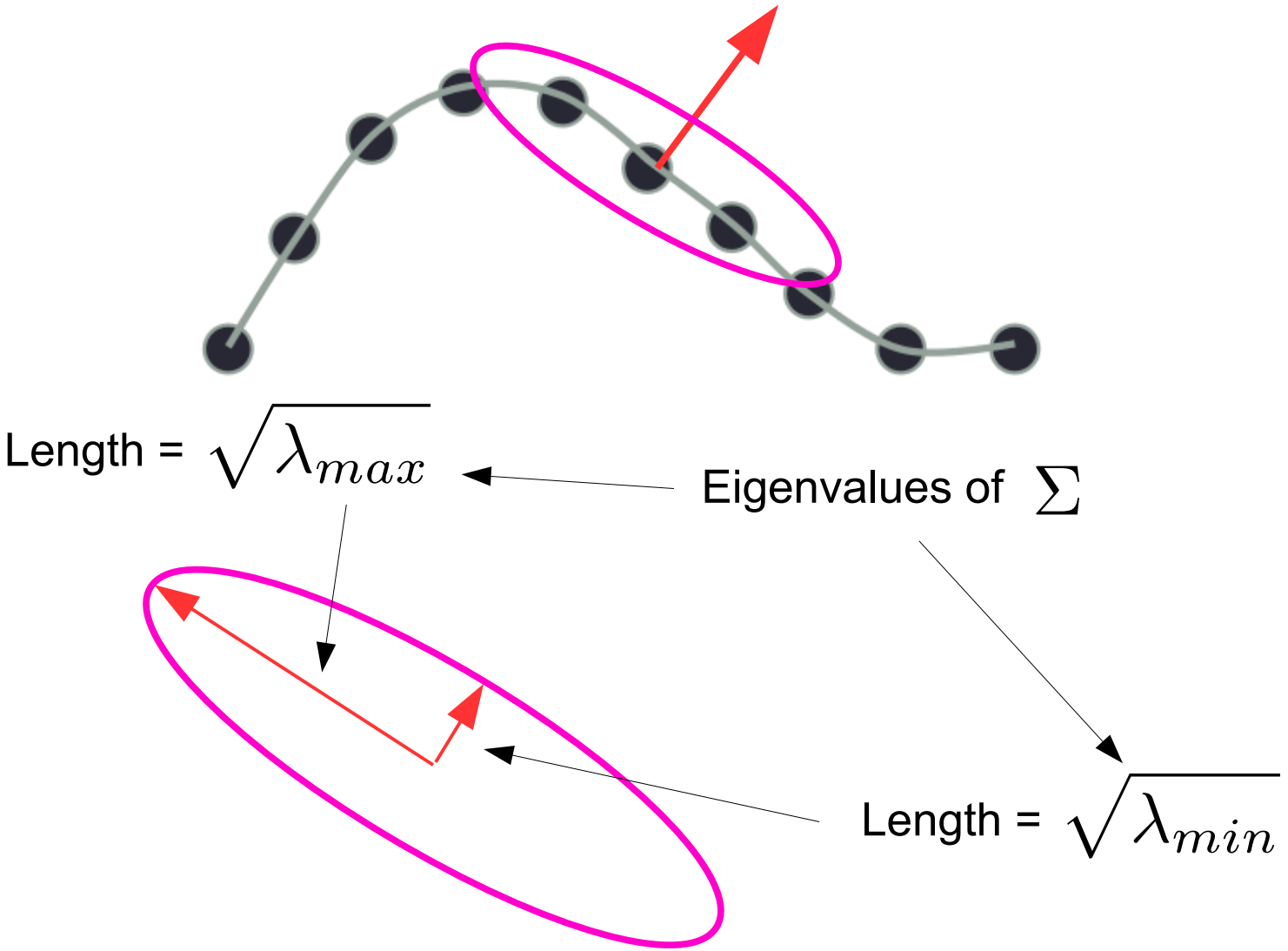


Calculate the sample covariance matrix of the points in $N_r(x)$

$$\Sigma = \sum_{p \in N_r(x)} (p - \bar{p})(p - \bar{p})^T$$

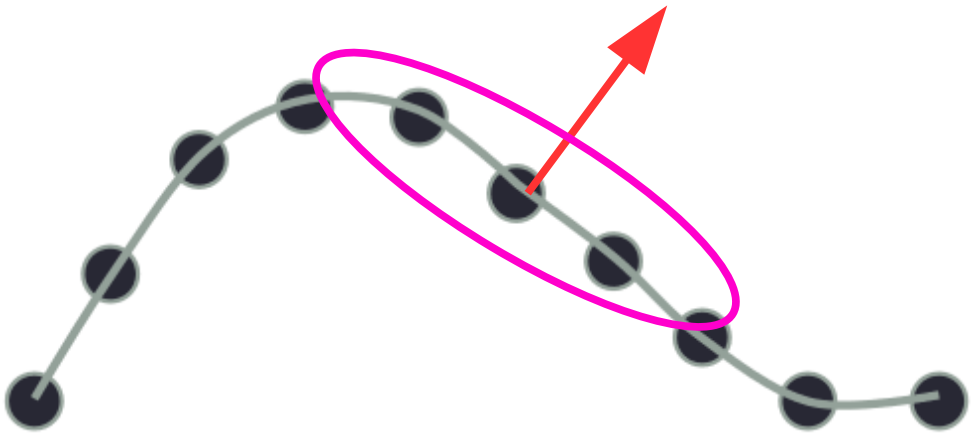
$$\bar{p} = \frac{1}{|N_r(x)|} \sum_{p \in N_r(x)} p$$

Surface normals



Principle axes of ellipse point in directions of corresponding eigenvectors

Surface normals



Surface normal is in the direction of the eigenvector corresponding to the smallest eigenvalue of $\sum$

There should be two large eigenvalues and one small eigenvalue.

Summary: Computing surface normals

1. Calculate points within r-ball about x: $N_r(x) = B_r(x) \cap C$

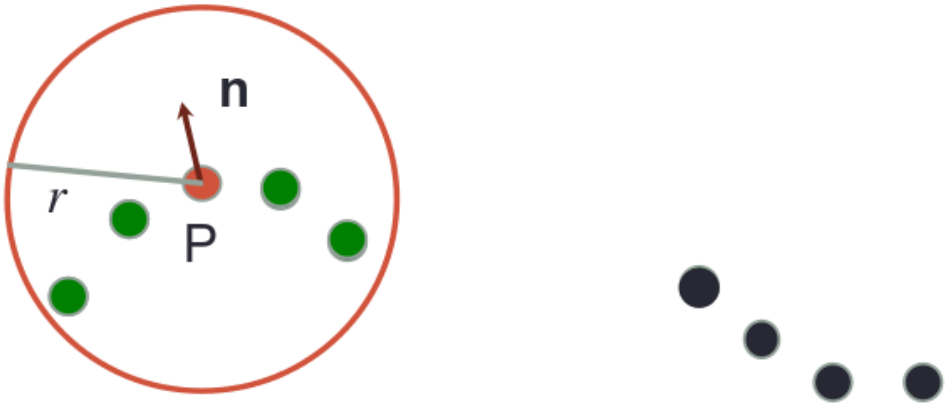
2. Calculate covariance matrix: $\Sigma = \sum_{p \in N_r(x)} (p - \bar{p})(p - \bar{p})^T$

3. Calculate eigenvectors: v_1, v_2, v_3
   and eigenvalues: $\lambda_1, \lambda_2, \lambda_3$ (\lambda_3 is smallest)

4. v_3 is parallel or antiparallel to surface normal

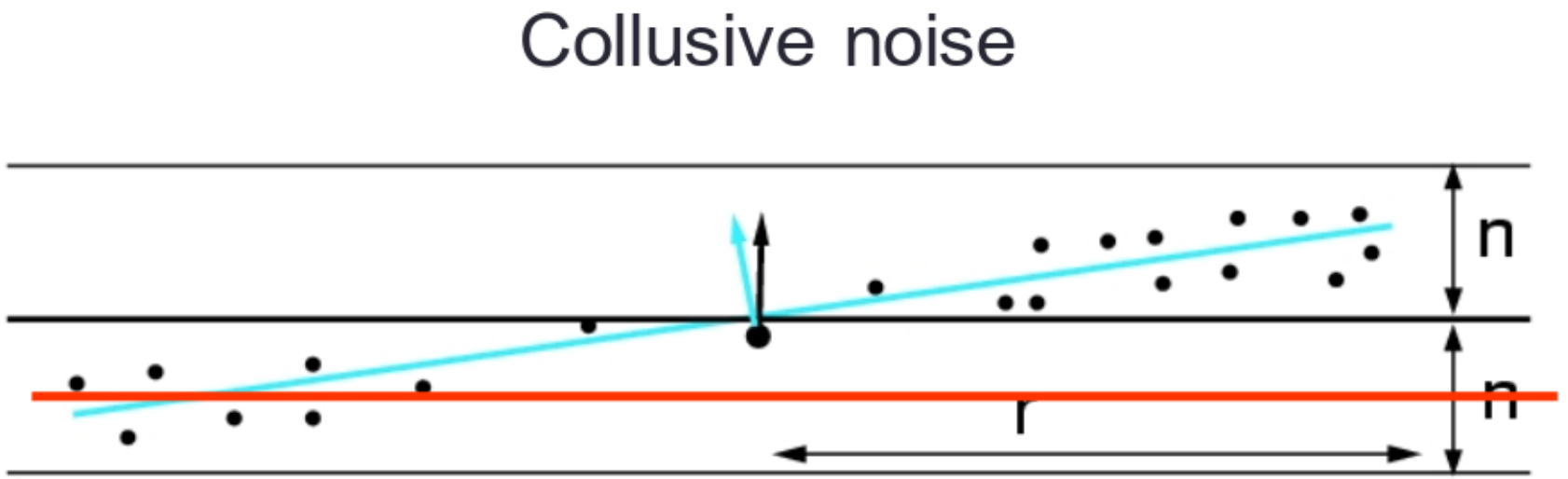
• Calculating surface normals

How large a point neighborhood to use when calculating $\sum$?



Because points can be uneven, don't use k-nearest neighbor.
– it's important to select a radius *r* and stick w/ it.
– which what value of *r* to use?

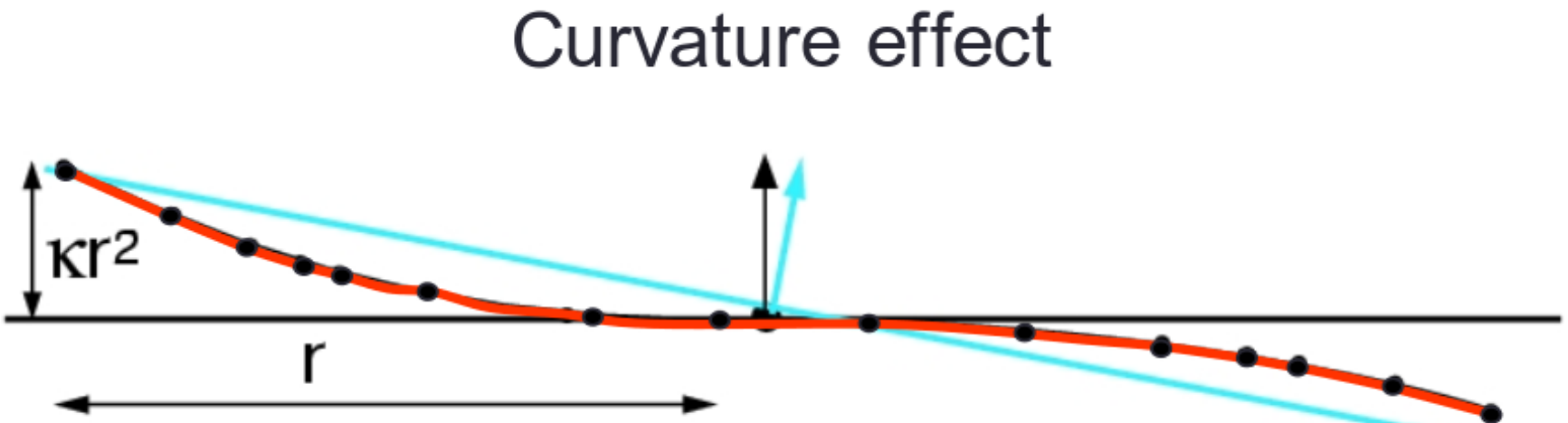
• Calculating surface normals



Because of noise in the data, small *r* may lead to underfitting.

Images from Course INF 555 slides, Ecole Polytechnique, Paris

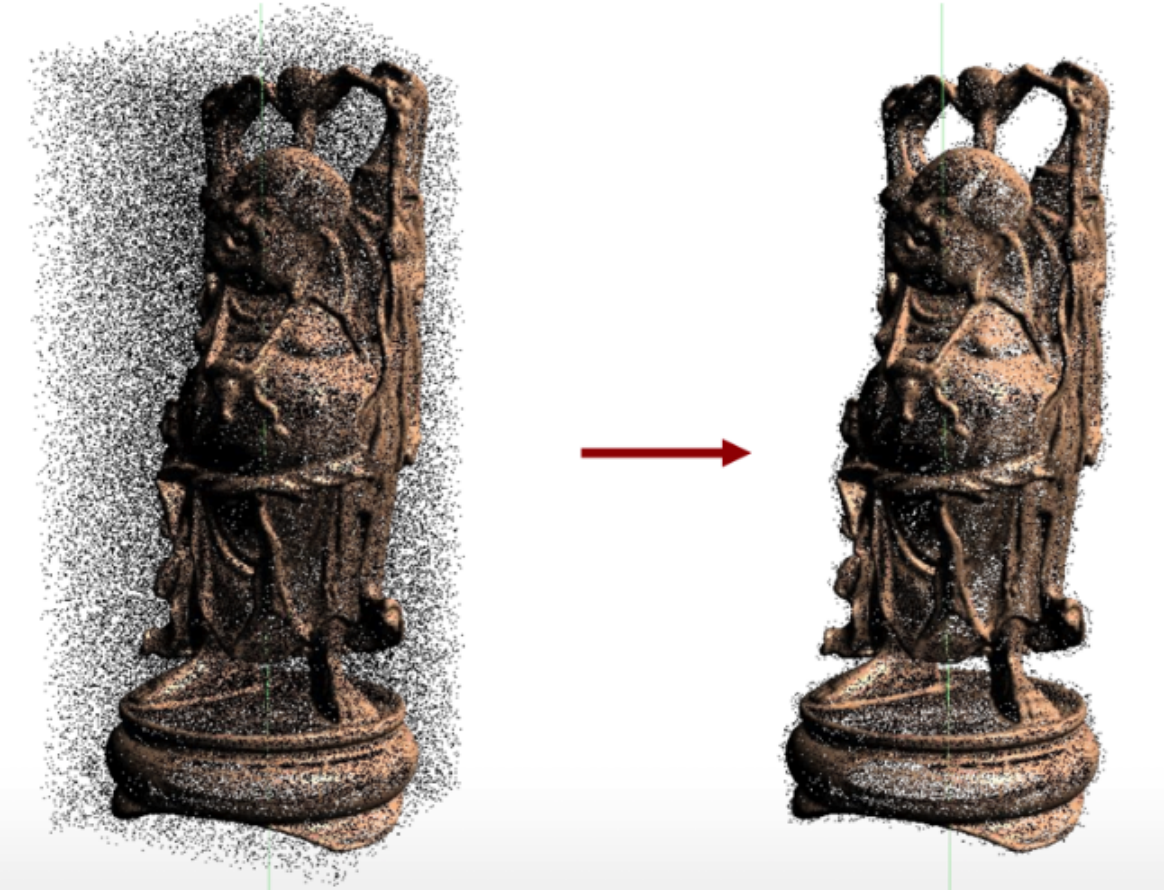
- Calculating surface normals



Due to curvature, large *r* can lead to estimation bias.

Images from Course INF 555 slides, Ecole Polytechnique, Paris

• Outlier removal

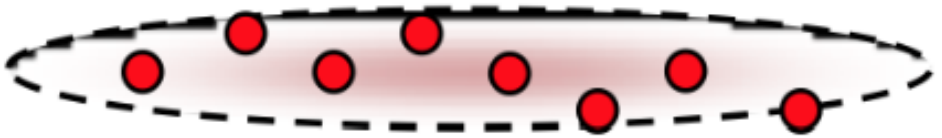


Similar approach as in normal estimation:
1. calculate local covariance matrix
2. estimate Eigenvectors/Eigenvalues
3. use that information somehow...

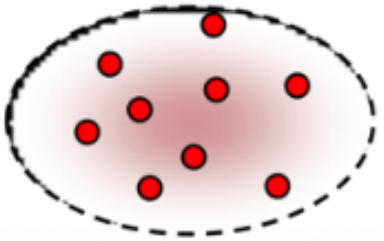
Images from Course INF 555 slides, Ecole Polytechnique, Paris

• Outlier removal

If points lie on a plane or line, then $\frac{\lambda_{min}(\Sigma)}{\lambda_{max}(\Sigma)}$ is small



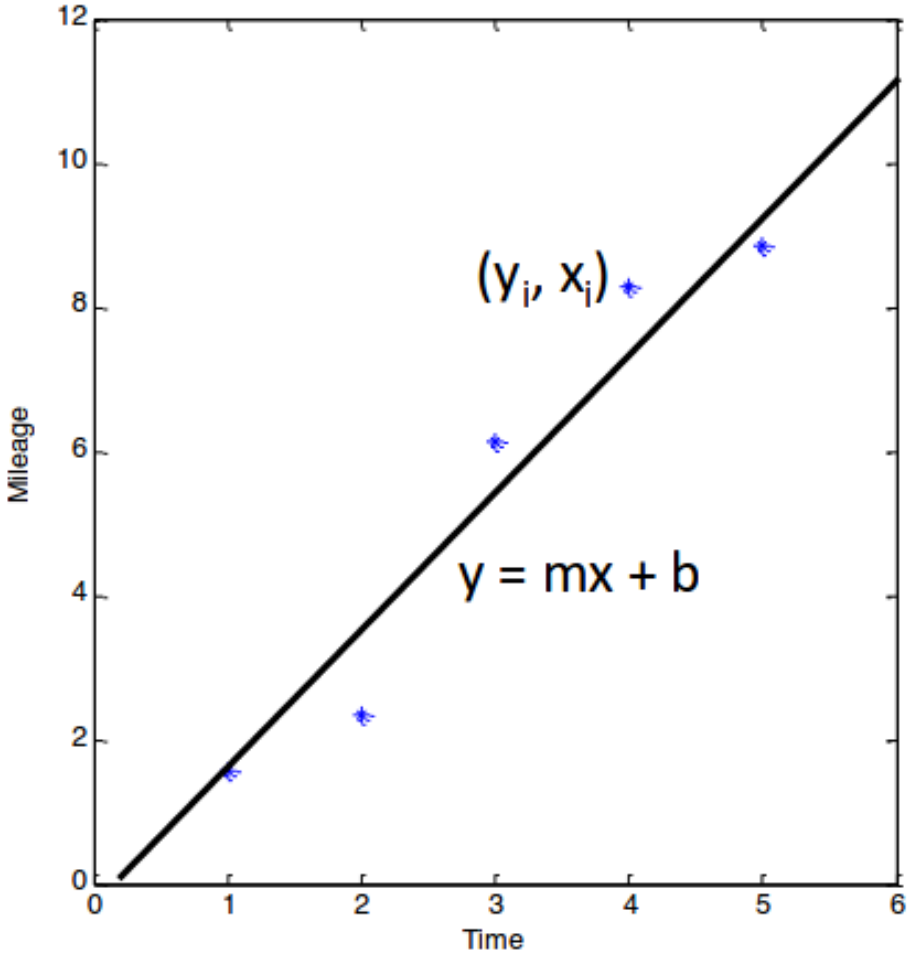
If points are uniformly random, then $\frac{\lambda_{min}(\Sigma)}{\lambda_{max}(\Sigma)}$ is close to 1



Outlier removal: delete all points for which $\frac{\lambda_{min}(\Sigma)}{\lambda_{max}(\Sigma)}$ is above a threshold

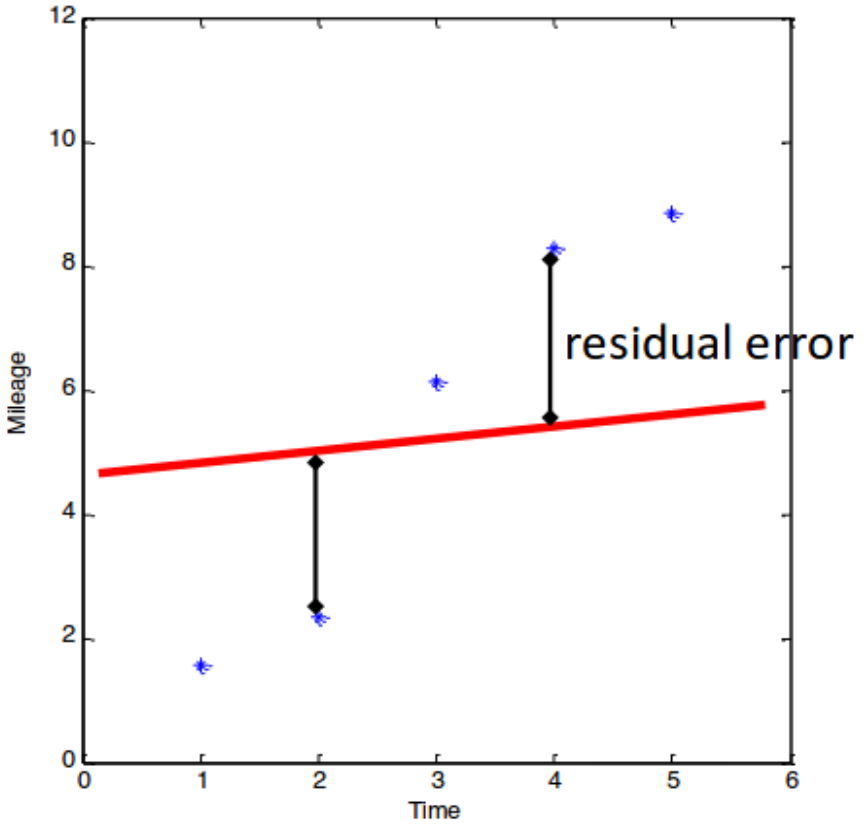
Images from Course INF 555 slides, Ecole Polytechnique, Paris

- Another approach to alignment: RANSAC



This slide from: Kavita Bala, Cornell U.

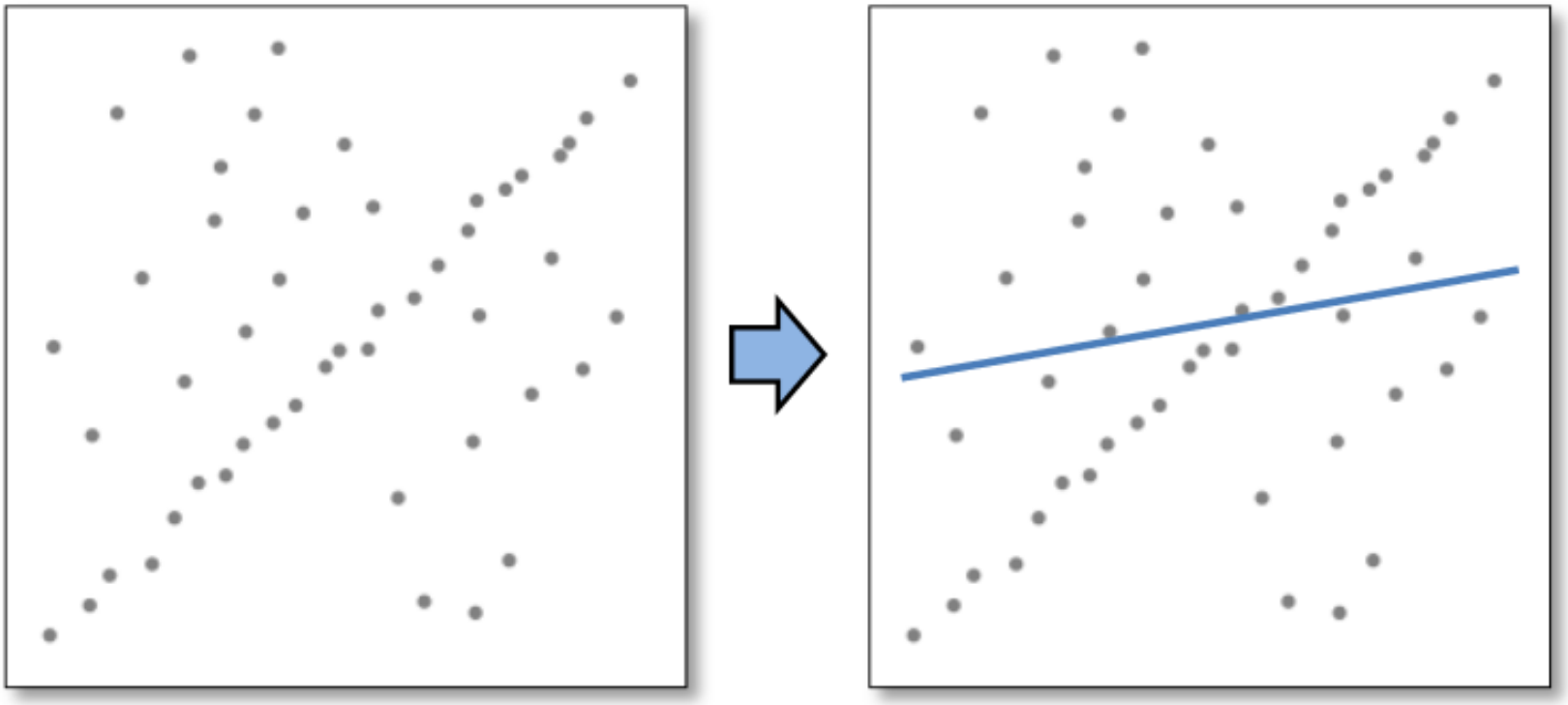
• RANSAC



$$\text{Cost}(m, b) = \sum_{i=1}^{n} |y_i - (mx_i + b)|^2$$

This slide from: Kavita Bala, Cornell U.

- How does regression work here?



Problem: Fit a line to these datapoints

Least squares fit

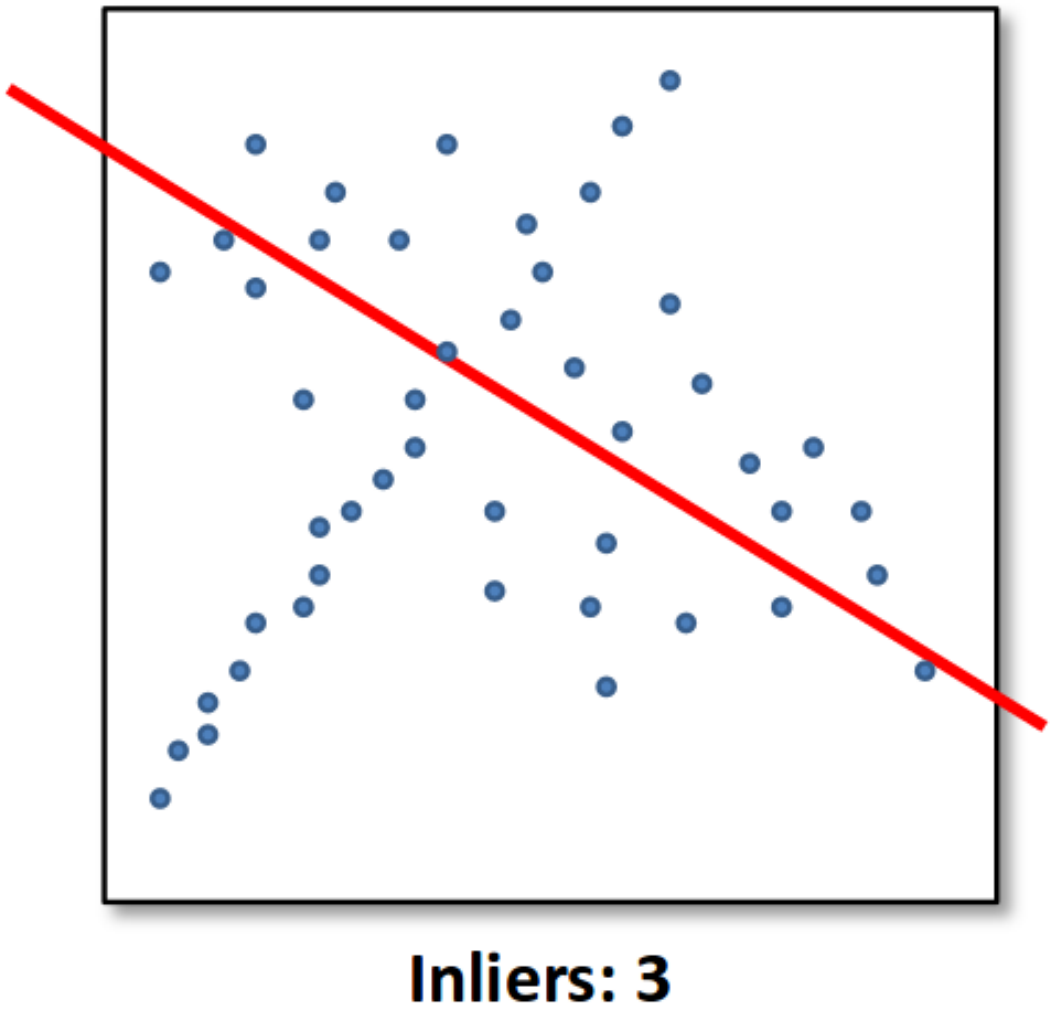
• RANSAC key idea

- Given a hypothesized line
- Count the number of points that "agree" with the line
  - "Agree" = within a small distance of the line
  - I.e., the **inliers** to that line

- For all possible lines, select the one with the largest number of inliers

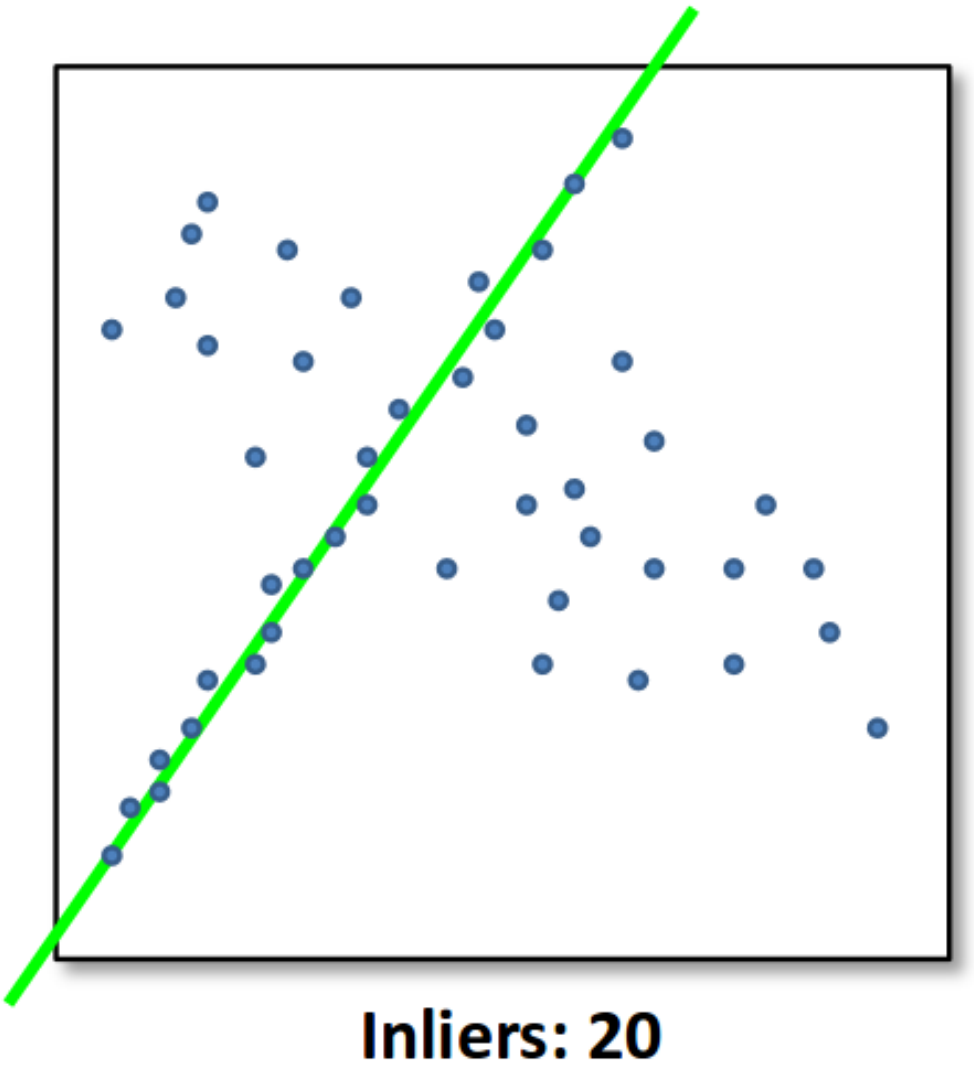
This slide from: Kavita Bala, Cornell U.

- Counting inliers



This slide from: Kavita Bala, Cornell U.

- Counting inliers



This slide from: Kavita Bala, Cornell U.

- How do we find the best line?

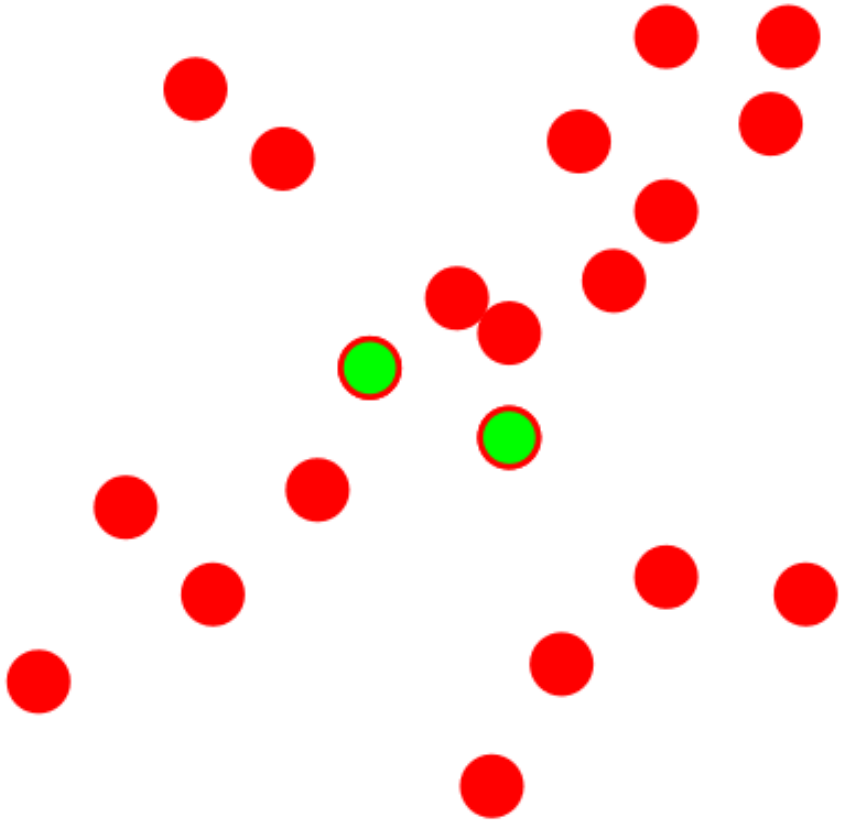
- Unlike least-squares, no simple closed-form solution

- Hypothesize-and-test
  - Try out many lines, keep the best one
  - Which lines?

This slide from: Kavita Bala, Cornell U.

RANSAC

Line fitting example



Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

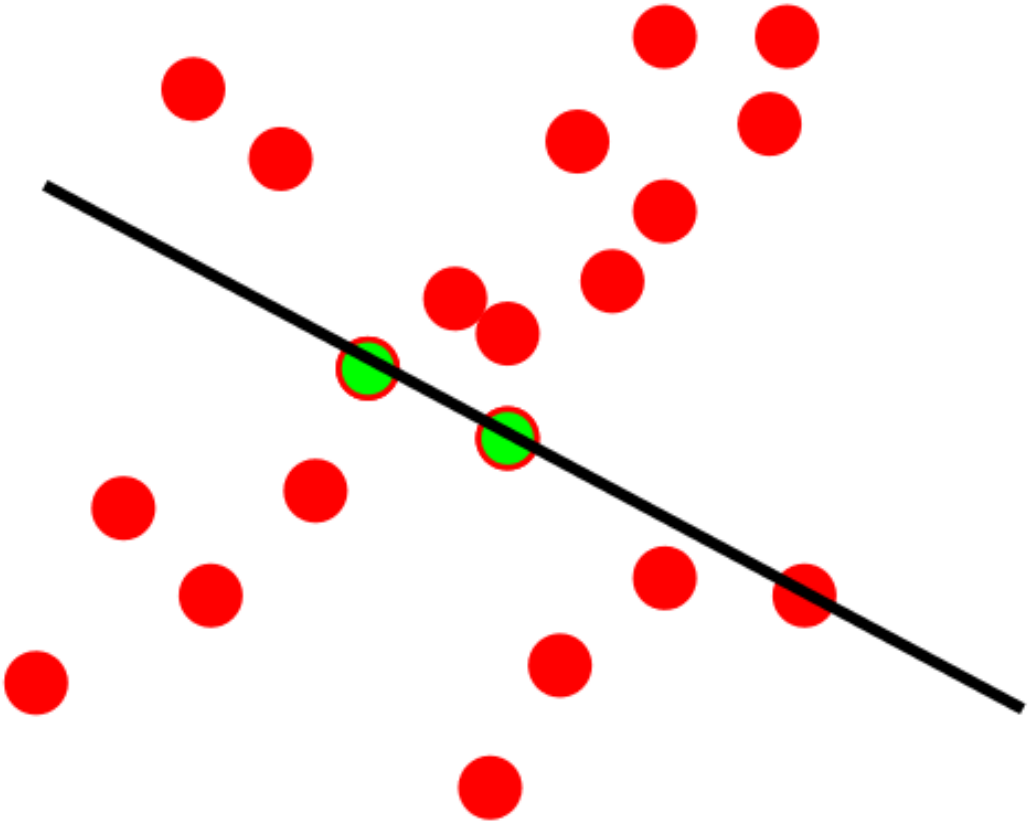
Repeat 1-3 until the best model is found with high confidence

This slide from: Kavita Bala, Cornell U.

Illustration by Savarese

RANSAC

Line fitting example



Algorithm:

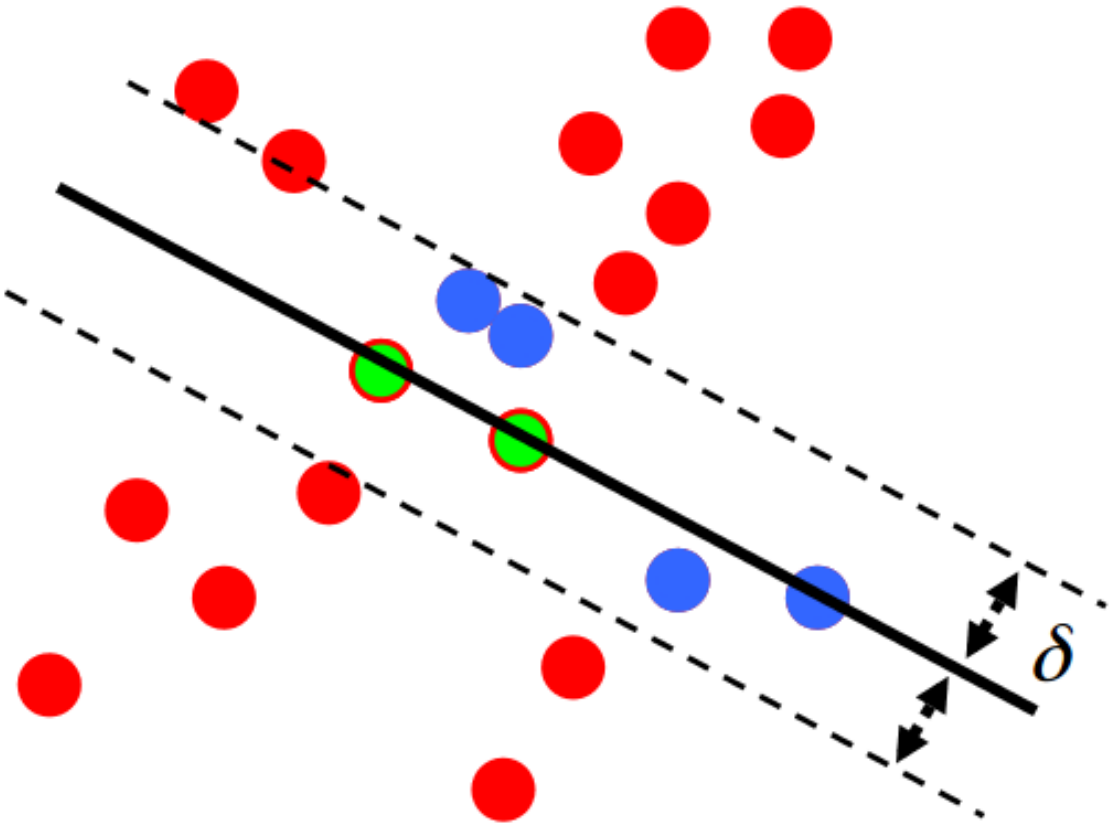
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

This slide from: Kavita Bala, Cornell U.

RANSAC

Line fitting example



$$N_I = 6$$

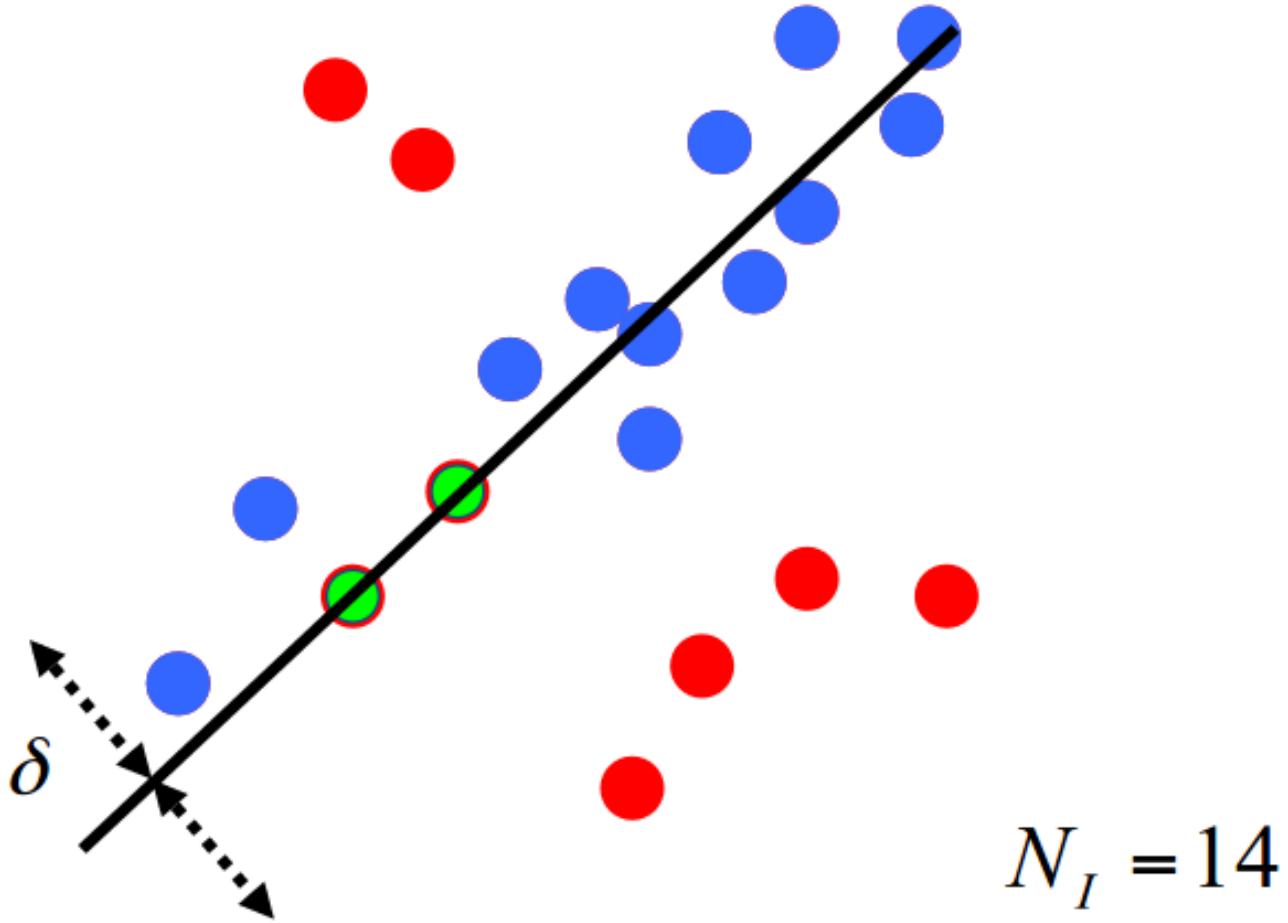
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

This slide from: Kavita Bala, Cornell U.

RANSAC



Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

This slide from: Kavita Bala, Cornell U.

• RANSAC Summary

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- Cons
  - Parameters to tune
  - Sometimes too many iterations are required
  - Can fail for extremely low inlier ratios
  - We can often do better than brute-force sampling

This slide from: Kavita Bala, Cornell U.

Outline

✔ 2-D image alignment
- Feature descriptors: Corner detectors, SIFT, SURF
- Applications

✔ 3-D perception – point clouds
- Sensors and strategies

✔ Aligning point clouds
- Iterative closest point

Extracting primitive shapes
- Surface normals (planes)
- RANSAC
- More shapes (time permitting)

Feedback

Piazza thread: 3/30 Lec 17 Feedback

Please post your answers to the following anonymously.

1. What did you like today?

2. What was unclear?

3. Any further thoughts / comments on Ex4?

4. Ready for the final month of the semester?

5. Any additional feedback / comments?