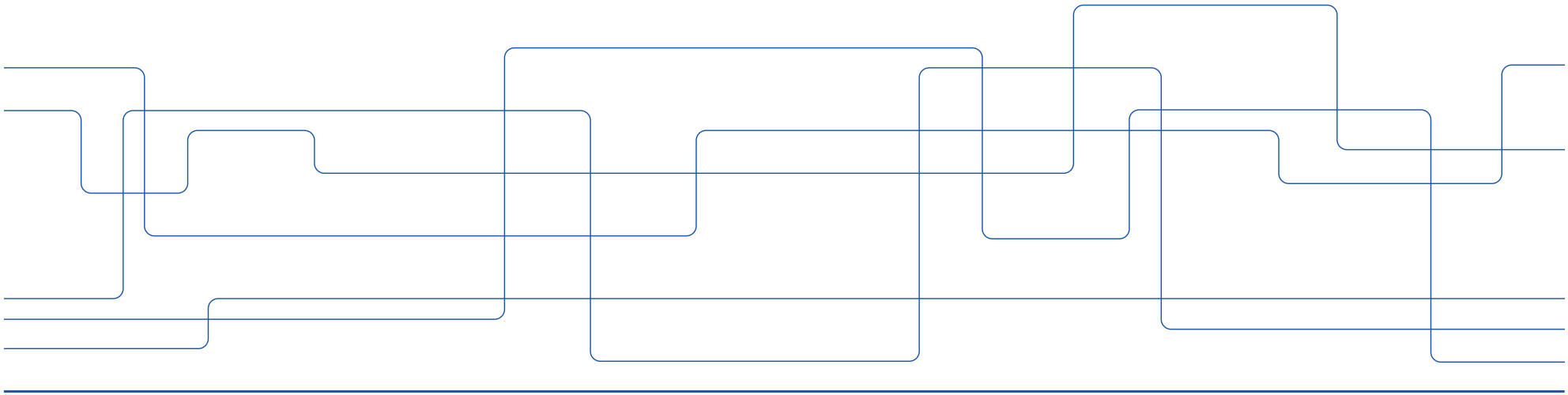
Model fitting and representation

Mårten Björkman



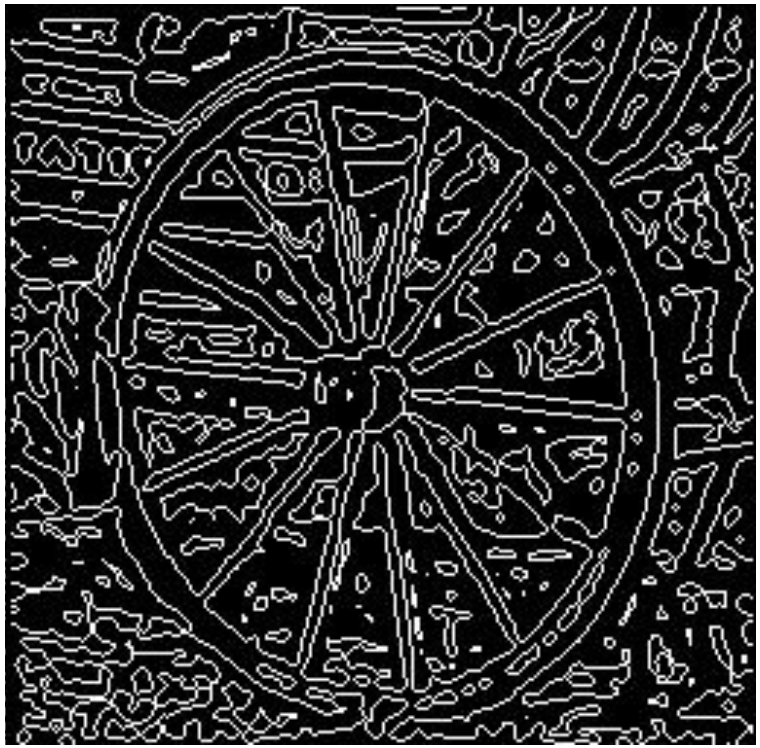
Edge detection (in short)

Edge detection in three steps:

1. Compute gradients (f_x, f_y) from derivatives.
2. Find edge points by maximizing gradient magnitudes $|(f_x, f_y)|$.
3. Finally, link edge points to edge segments.

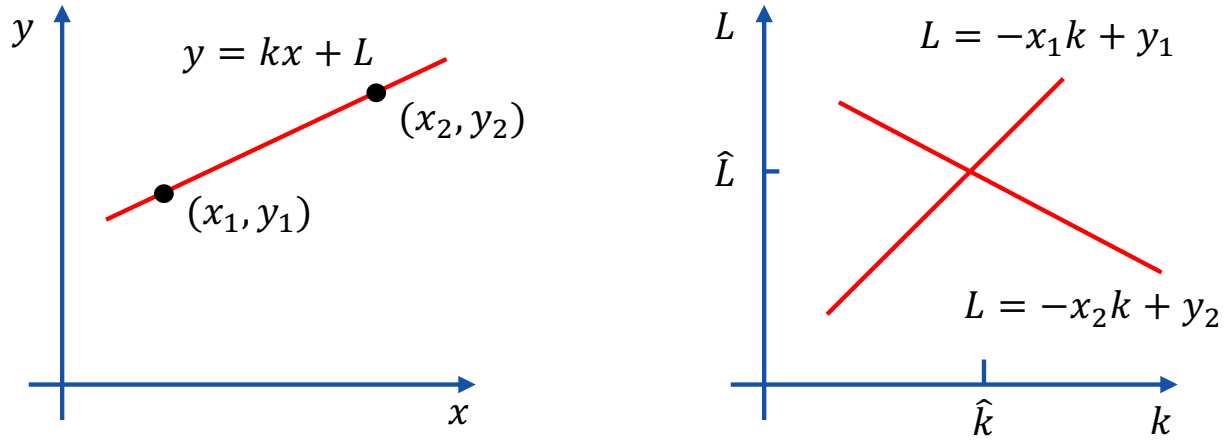
How do we go from that to a more compact description? Find lines!

Problem: edge segments might be fragmented into many pieces.



Hough transform for line detection

- Line = a collection of edge points placed along the same direction.
- Hough Transform can detect all possible lines spanned by edge points.
- The line strength depends on how many points lie on that line.



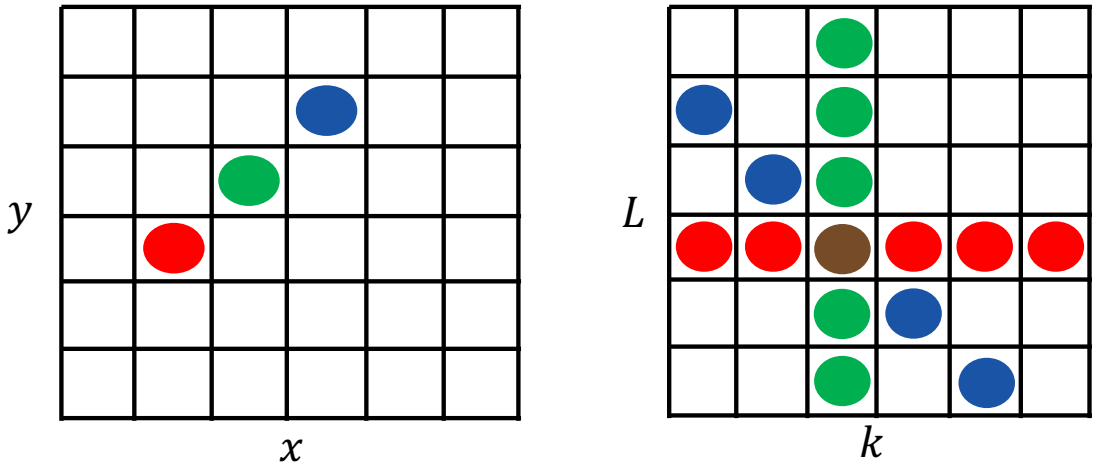
- A point in (x, y)-space is mapped to a line in (L, k)-space.
- A line in (x, y)-space is mapped to a point in (L, k)-space.

From edge points to lines

- Each point (x, y) could have come from many lines defined in (L, k)-space,

$$L(k) = -kx + y$$

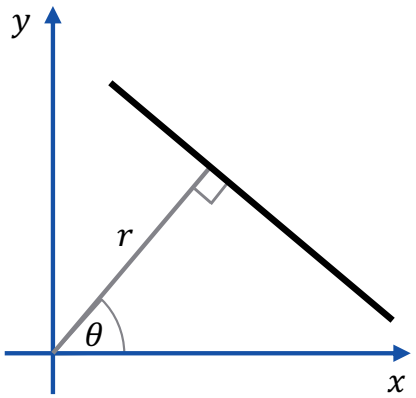
- Idea: Vote on different combinations of (L, k) in a discrete accumulator space.
- The estimated line parameters $(\hat{L}, \hat{k})$ are given by intersection point in accumulator space that gets most votes.

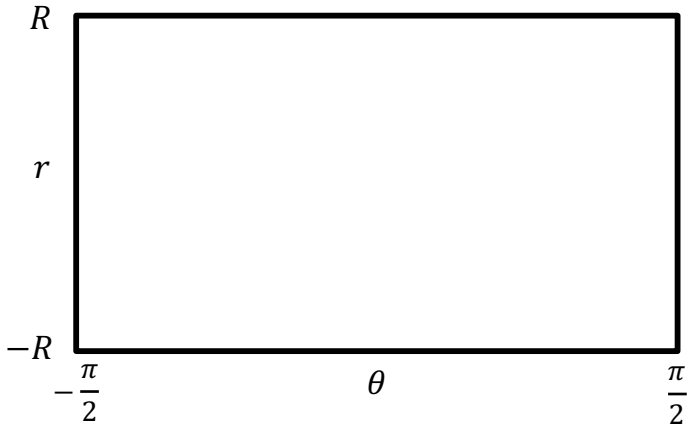


Problem: parameterization in duality space

- Observation: vertical lines correspond to $k \longrightarrow \pm\infty$
- Better method: use parameterization $x\cos\theta + y\sin\theta = r$

  θ = orientation of line, r = perpendicular distance to origin

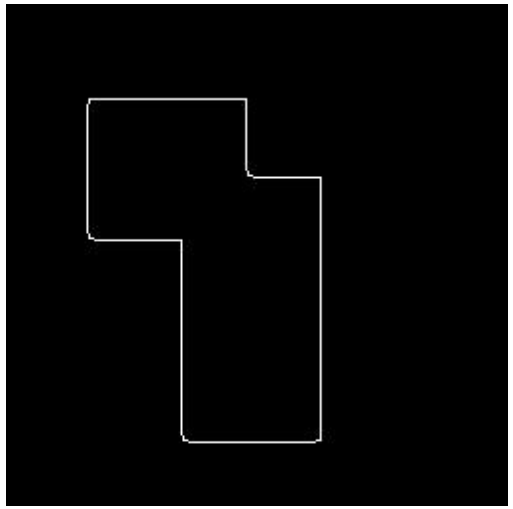




- Point (x, y) in the image $\Longrightarrow$ curve $r(\theta) = x\cos\theta + y\sin\theta$ in the Hough (accumulator) space.

Lines represented in Hough space

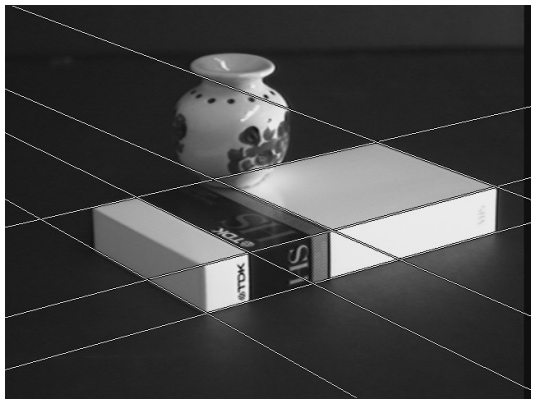
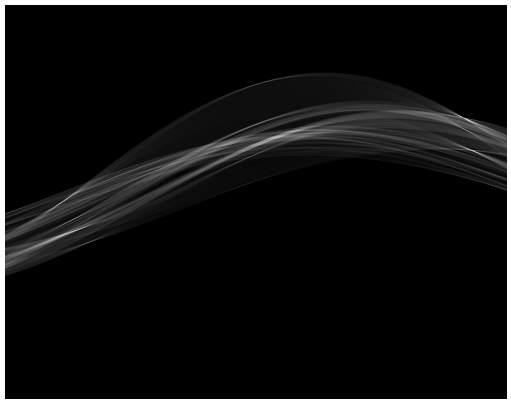
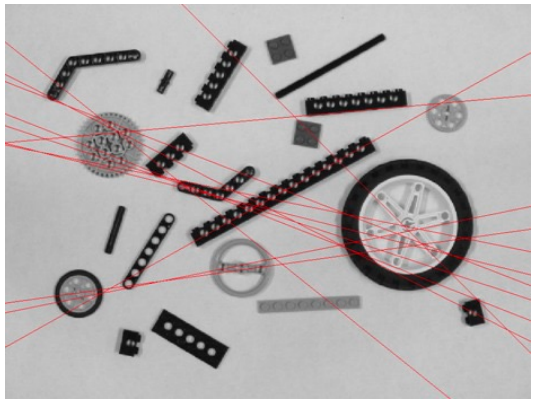
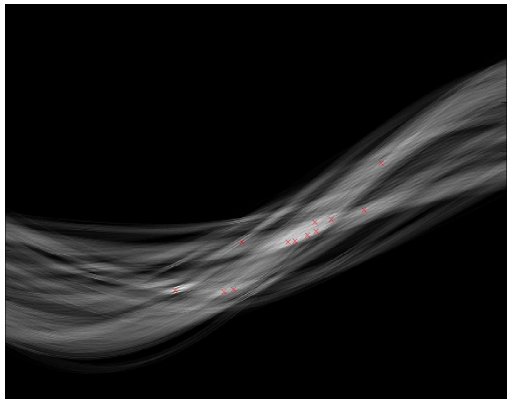
Which line corresponds to which point in Hough space?



Question: How fine should θ and r be quantized?

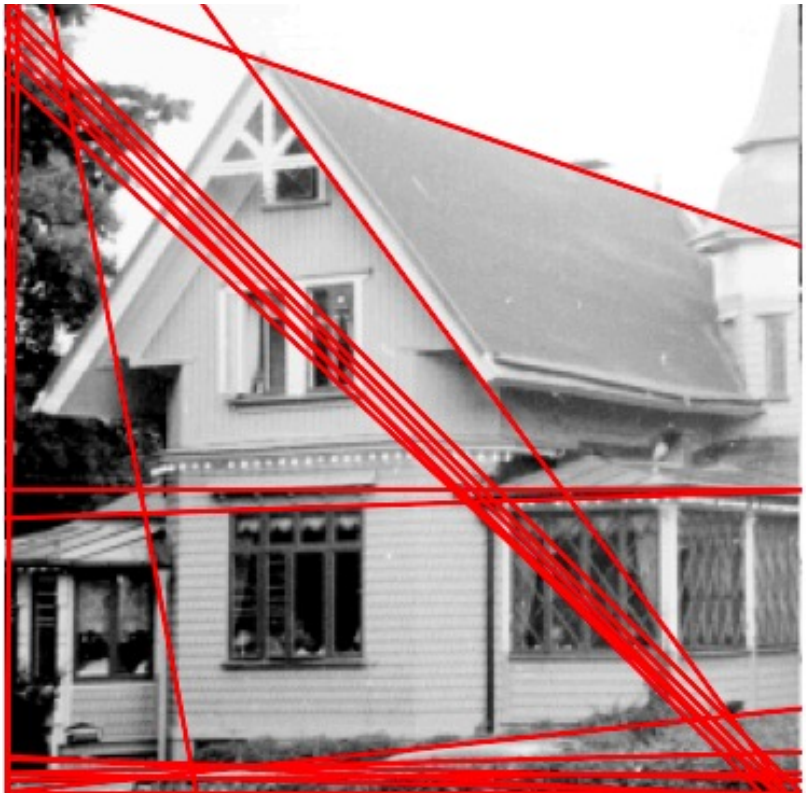
- Too coarse: poor resolution and imprecise line directions.
- Too fine: too many accumulators and too few samples per accumulator.

Hough transform (example)

Note: Edge points may come from texture, not just real lines

Example from lab 2

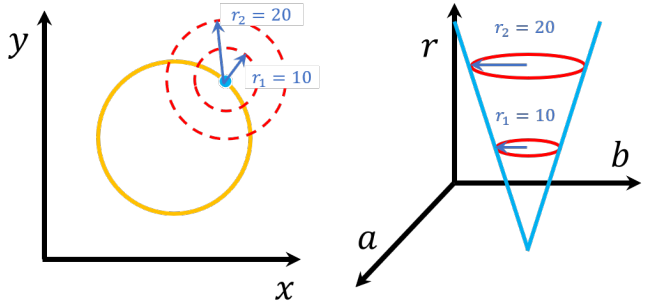


Lane detection using Hough transform

Hough transform (improvements)

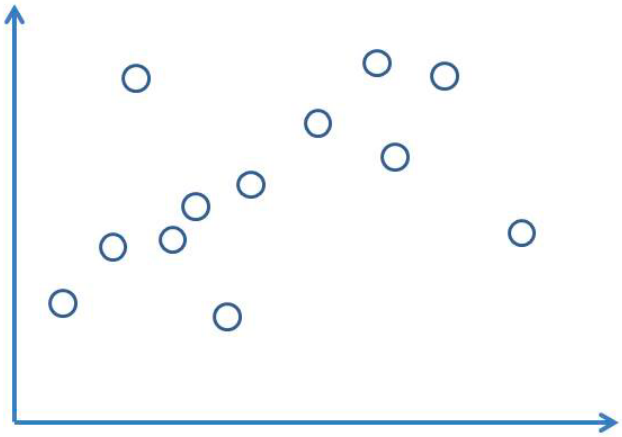
- Increment with gradient magnitude or some other weighting.
  - Reduces critical dependency on thresholds (rarely improves).
- Use information about gradient direction from edge detection.
  - Increment only for those r values that seem reasonable (more effective).
- Increment accumulator not only point-wise but also with some windowing function (Gaussian like)
  - Equivalent to smoothing after voting is done (more effective).

- Hough Transforms can also be extended
  - Line: 2 parameters $\Longrightarrow$ 2D Hough space (fine)
  - Circle: 3 parameters $\Longrightarrow$ 3D Hough space (doable)
  - Ellipse: 5 parameters $\Longrightarrow$ 5D Hough space (infeasible)



RANSAC: Random Sampling Consensus

- Alternative: Random Sampling Consensus (RANSAC)
  1. Randomly pick a minimal set of points (lines=2, circle=3, ellipse=5)
  2. Compute a model (e.g. $y = kx + L$) from the points
  3. Count the number of points 'close enough' to the model.
  4. Repeat 1-3 a given number S of iterations.
  5. Pick the solution with highest number of matching points in 3.



RANSAC (line example)

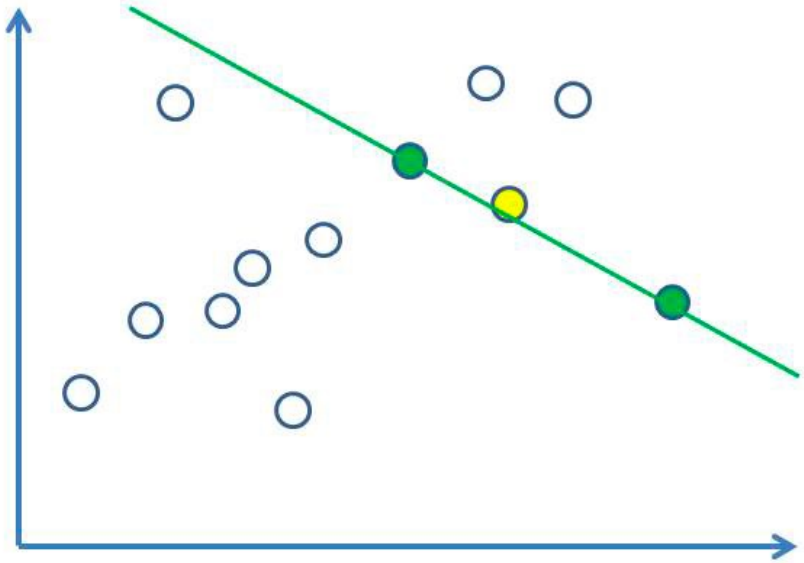
- Step 2: Find the K = 2 unknown parameters

$$\begin{cases} kx_1 + L = y_1 \\ kx_2 + L = y_2 \end{cases} \Rightarrow \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \end{pmatrix} \begin{pmatrix} k \\ L \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \Rightarrow$$

$$\begin{pmatrix} k \\ L \end{pmatrix} = \frac{1}{x_1 - x_2} \begin{pmatrix} 1 & -1 \\ -x_2 & x_1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \frac{1}{x_1 - x_2} \begin{pmatrix} y_1 - y_2 \\ x_1 y_2 - x_2 y_1 \end{pmatrix}$$

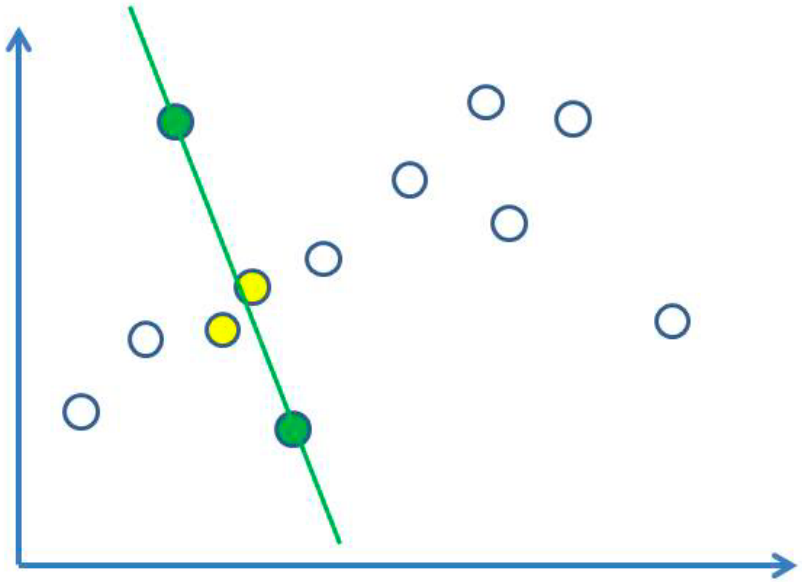
- Step 3: Count the number of points (x_i, y_i) for which $|kx_i + L - y_i| < \varepsilon$, where ε is some threshold (e.g. 2 pixels).

RANSAC (line example)



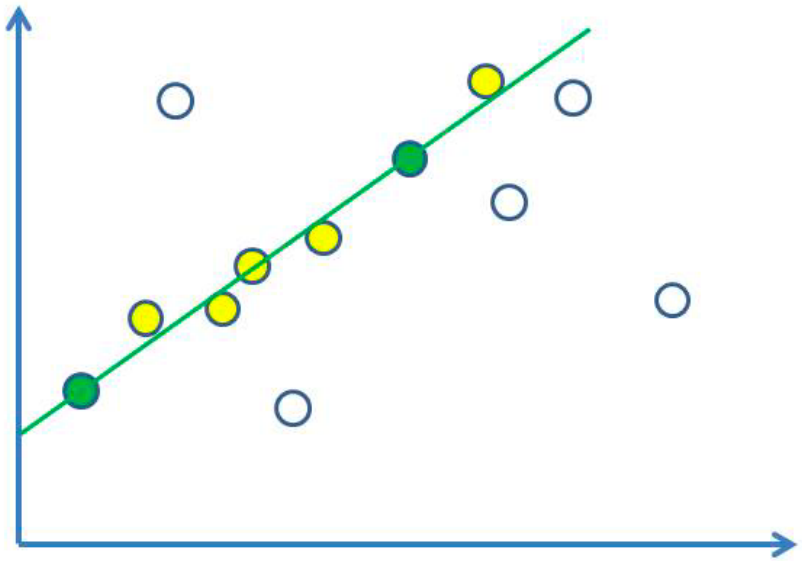
Three inliers, nine outliers.

RANSAC (line example)



Four inliers, eight outliers. Better!

RANSAC (line example)



Seven inliers, five outliers. Even better!

RANSAC: Number of trials?

- All K sampled points must lie on the shape (e.g. line) to be found.
- How many trials are needed for this to happen with a certainty of P, if a fraction of p of points belong to the shape?
  - Probability of failure in one trial $= (1 - p^K)$
  - Complete failure = probability of failure after S trials $\Longrightarrow (1 - p^K)^S = (1 - P)$
  - The number of required trials is

$$S = \frac{\log(1 - P)}{\log(1 - p^K)}$$

Example 1 (line): $p = 10\%, P = 99\% \Longrightarrow S = 460$

Example 2 (ellipse): $p = 5\%, P = 99\% \Longrightarrow S \approx 15 \times 10^6$

- Number of required trials quickly becomes very large.

Matching planes with homographies

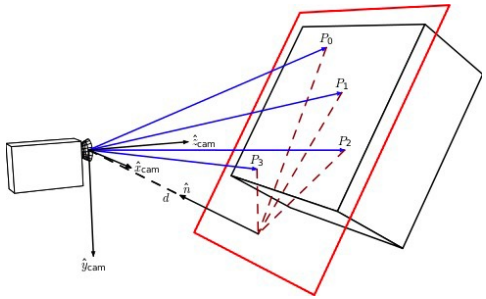
- Projections as normally not invertible.

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \simeq \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

- However, for a plane defined by $Z_i = 0$, it can (usually) be inverted.

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \simeq \begin{bmatrix} p_{00} & p_{01} & p_{03} \\ p_{10} & p_{11} & p_{13} \\ p_{20} & p_{21} & p_{23} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix}$$

- The relation is a 3×3 homography matrix that is invertible, if the camera centre does not lie on the plane.

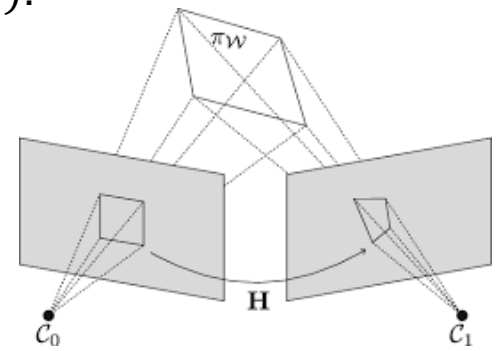


Matching planes with homographies

- Assume a plane in 3D space is viewed by two different cameras.
- Then a point $(x_i, y_i)^T$ on the plane in one camera can be transformed to a point in the other $(x_i', y_i')^T$ through a homography H,

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \simeq \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

- You can see that mapping as a mapping $(x_i, y_i) \longrightarrow (X_i, Y_i)$ followed by another mapping $(X_i, Y_i) \longrightarrow (x_i', y_i')$.



Matching planes with homographies

1. Detect point features in both images (e.g. SIFT or SURF)

2. Match features between the two images.

3. Use RANSAC to find homography and mismatches (outliers)

Solving for homographies

$$\begin{bmatrix} wx_i' \\ wy_i' \\ w \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$x_i' = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}, \quad y_i' = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$\begin{cases} x_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{00}x_i + h_{01}y_i + h_{02} \\ y_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12} \end{cases}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y_i'y_i & -y_i \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = 0$$

One match leads to two equations, but we have nine unknown in total.

Solving for homographies

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1 \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_n & -x_n'y_n & -x_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n'x_n & -y_n'y_n & -y_n \end{bmatrix}}_{A\ [2n \times 9]} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{h\ [9 \times 1]} = \underbrace{0}_{[2n \times 1]}$$

- Four ($n = 4$) matches are needed, plus the extra constraint $h^T h - 1 = 0$.
- Define a constrained least square problem with a Lagrangian multiplier λ:

$$\textit{minimize } f(h, \lambda) = |Ah - 0|^2 - \lambda(h^T h - 1) = h^T A^T A h - \lambda(h^T h - 1)$$

$$f_h(h, \lambda) = 2A^T A h - 2\lambda h = 0 \Longrightarrow (A^T A)h = \lambda h$$

- Solution: $\hat{h}$ = eigenvector of smallest eigenvalue of $A^T A$.

Application: panorama stitching

It the scene is far enough away, it can still be viewed as a plane.

Reasoning about shape in binary images

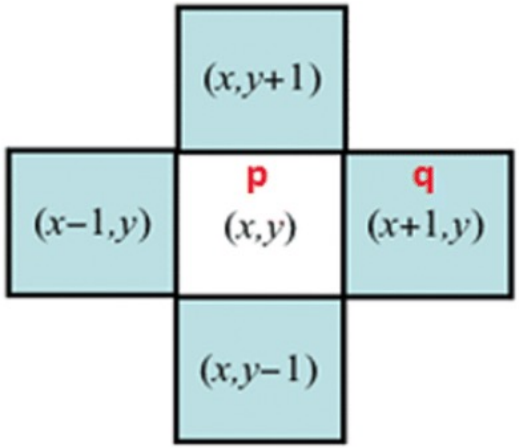
- Images with two colours, black (0) and white (1 or 255).
  - Commonly referred to as 'background' and 'foreground'.
- Typically obtained from thresholding or image segmentation.

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T \\ 0 & \text{otherwise} \end{cases}$$

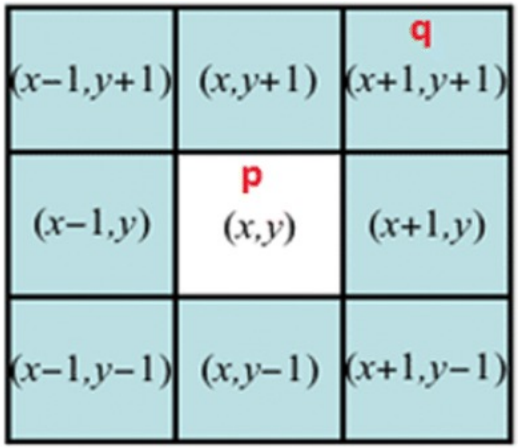
- In most cases it can not be done as easily as this, but sometimes you are lucky.

Neighbourhood concepts

- Many image processing operations work of neighbouring pixels. We need to define what it means that two pixels are neighbours.
- Results often differ (slightly) depending on the definition.



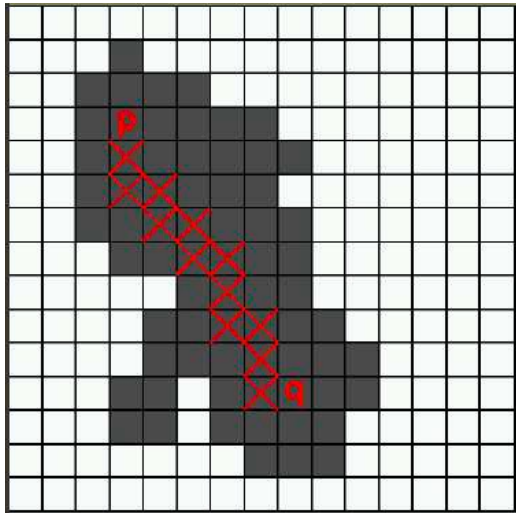
4-neighbourhood



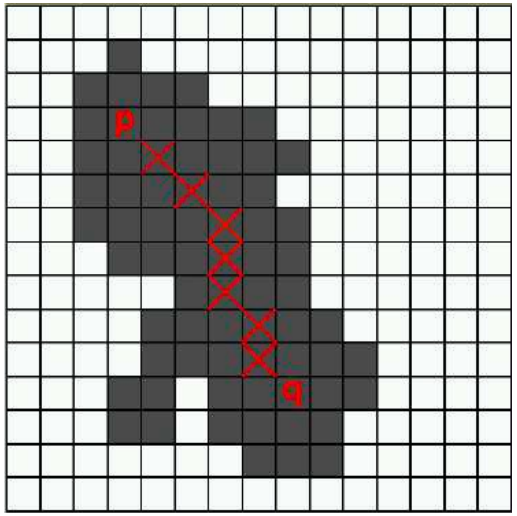
8-neighbourhood

Connectivity

- Path: A path from p to q is a set of points $p_0, \dots, p_n$, such that each point p_i is a neighbour of p_{i+1}.
- Connectivity: p is connected to q in S (some set of pixels), if there is a path from p to q completely in S.



4-neighbourhood



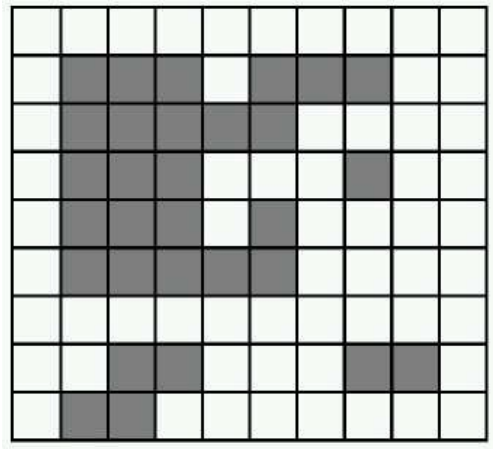
8-neighbourhood

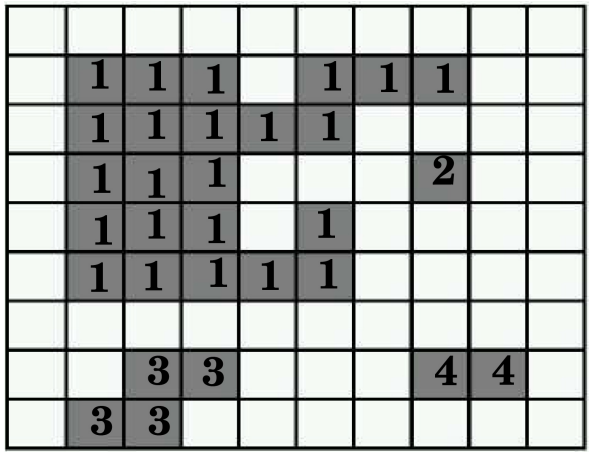
Connected components

For every p, the set of all points q connected to p is said to be its connected component.

Recursive procedure that scans entire image:

1. for each unlabelled foreground pixel, assign it a new label L
2. recursively assign label L to all neighbouring foreground pixels
3. stop if there is no unlabelled foreground pixels

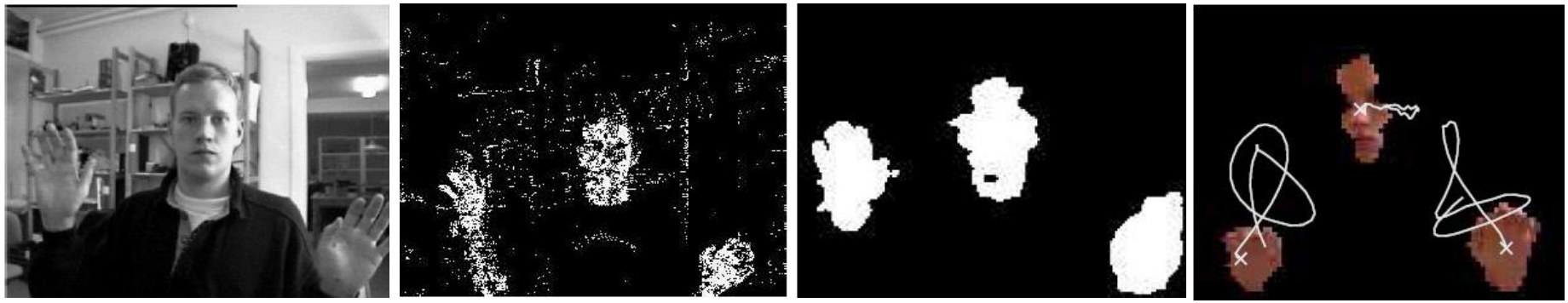




Connected component labelling

Regions (connected components) are often denoted by labels.

- statistics of regions (size, shape, grey-level statistics)
- size filtering (suppress objects of size < threshold)

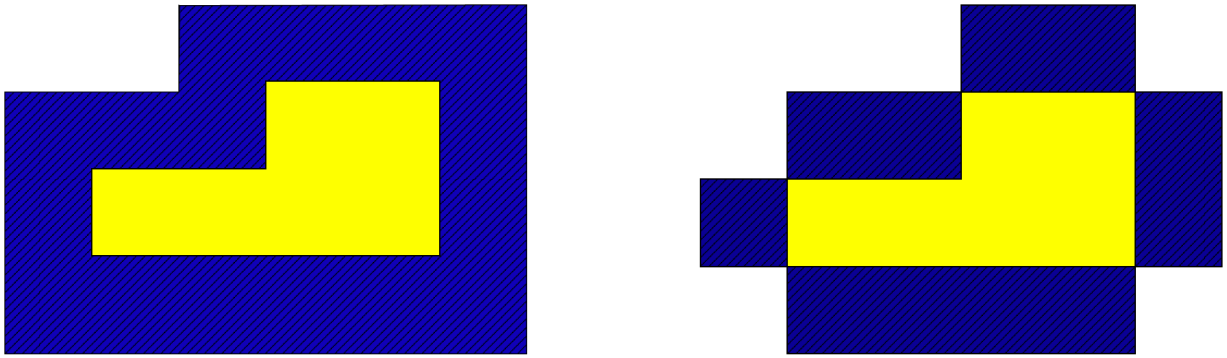


Connected components video example

Pixels are classified based on their colours, resulting in connected components.
The largest such component is then tracked.

Duality of 4-connectivity and 8-connectivity

- Outer boundary: background points with a neighbour on the object.
- If the connected component is defined using 4-connectivity, the outer boundary is 8-connected, and vice versa.



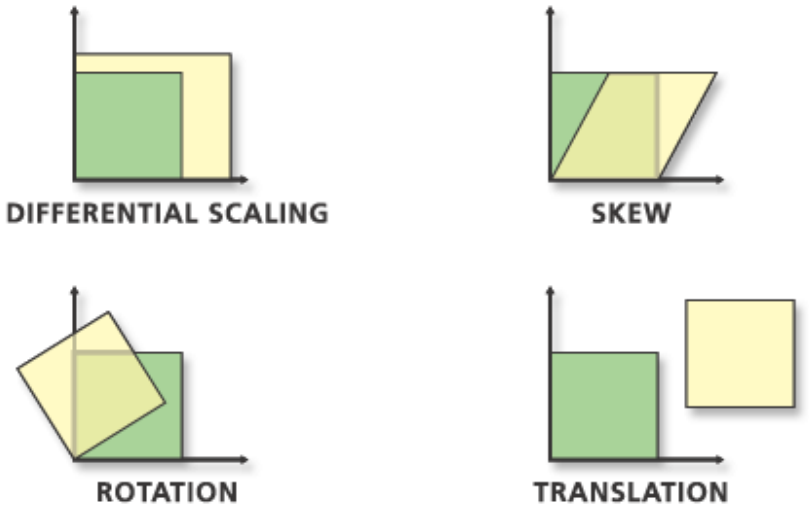
- Jordan curve theorem: Each closed curve divides plane into one region inside and one region outside.
- In fact, many region based methods, only really store the boundary.

Representation of shape

Task: Describe different shapes detected in an image.

- Shape descriptors are just ’numbers’ that describe a shape.
- A shape may not be reconstructable from a descriptor, but descriptors for different shapes should be different enough for shapes to be discriminated.
- Typically used for matching between two images or between images and some known shapes.
- Can be learned with neural networks (representation learning), but may be difficult without a lot of training data.

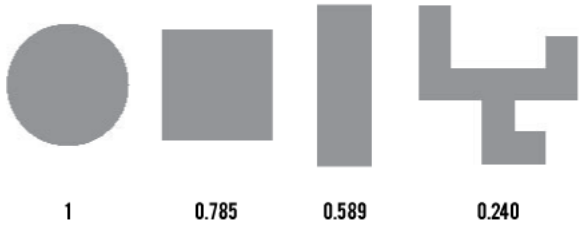
What invariances do we want?

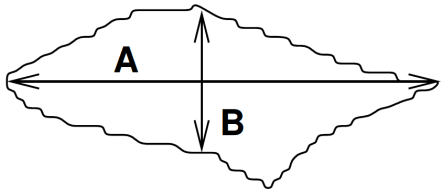


- Depending on task, context and images.
  - Translation invariance (for sure)
  - Scale invariance (definitely)
  - Rotation invariance (sometimes, but what about 6 and 9?)
  - Skew invariance (possibly)

Simple descriptors

- Sometimes a very simple descriptor can be most efficient.
- The first measures that comes to mind
  - Size (number of pixels)
  - Bounding box (width and height)
  - Aspect ratio
- Compactness $= \frac{area}{\text{circumference}^2}$
- Eccentricity $= \frac{length\ of\ maximum\ chord\ A}{length\ of\ maximum\ chord\ B \perp A}$





Moment descriptors of regions

- Moments:

$$m_{pq} = \sum_{x,y} x^p y^q f(x,y),$$

$$\text{where} \quad f(x,y) = \begin{cases} 1 & \text{in region} \\ 0 & \text{outside} \end{cases}$$

- Centred moments:

$$\mu_{pq} = \sum_{x,y} (x - \bar{x})^p (y - \bar{y})^q f(x,y)$$

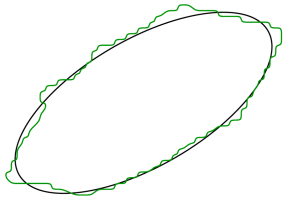
with centre of gravity

$$\bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}}$$

- Combine moments for rotation, scale invariance ....
- Examples: $\mu_{20} + \mu_{02}$, $\mu_{20}\mu_{02} - \mu_{11}^2$, $(\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2$

Moments: special case

- Moment descriptors of orders 0-2 $\Longrightarrow$ ellipse approximation



- Centre of gravity:

$$\bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}}$$

- Covariance matrix:

$$C = \begin{pmatrix} C_{xx} & C_{xy} \\ C_{xy} & C_{yy} \end{pmatrix}$$

$$C_{xx} = \mu_{20} = m_{20} - \bar{x}m_{10},\ C_{xy} = \mu_{11} = m_{11} - \bar{x}m_{01},\ C_{yy} = \mu_{02} = m_{02} - \bar{y}m_{01}$$

Moments: special case

- Lengths of main axes of the ellipse:

$$\begin{vmatrix} C_{xx} - \lambda & C_{xy} \\ C_{xy} & C_{yy} - \lambda \end{vmatrix} = 0$$

$$\lambda^2 - \underbrace{(C_{xx} + C_{yy})}_{trace\ C}\lambda + \underbrace{(C_{xx}C_{yy} - C_{xy}^2)}_{det\ C} = 0$$

$$\lambda_{1,2} = \frac{\mathrm{trace}(C)}{2} \pm \sqrt{\frac{\mathrm{trace}(C)^2}{4} - \det(C)}$$

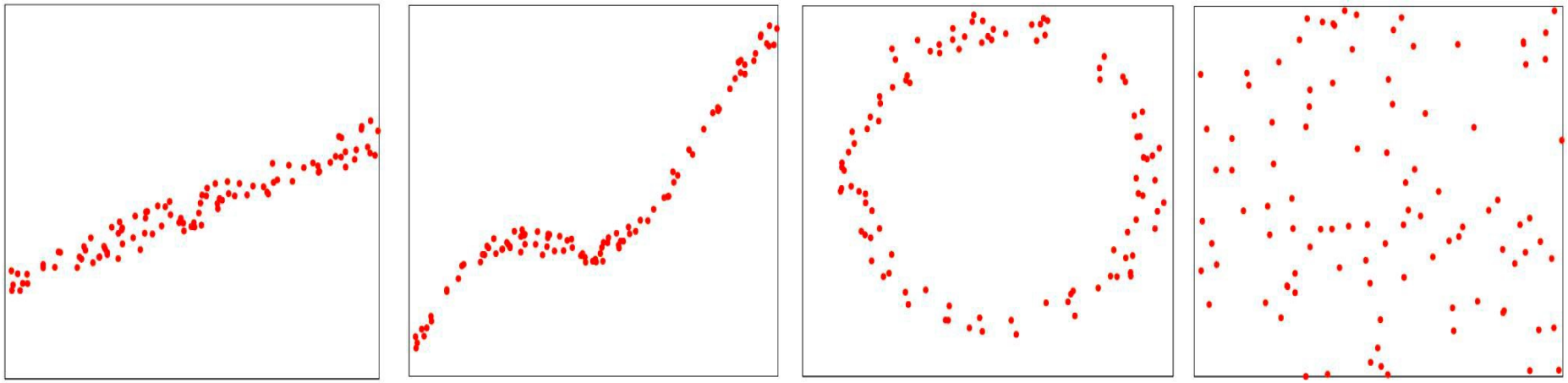
- Orientation:

$$\theta = \frac{1}{2}\tan^{-1}\left(\frac{2C_{xy}}{C_{xx} - C_{yy}}\right)$$

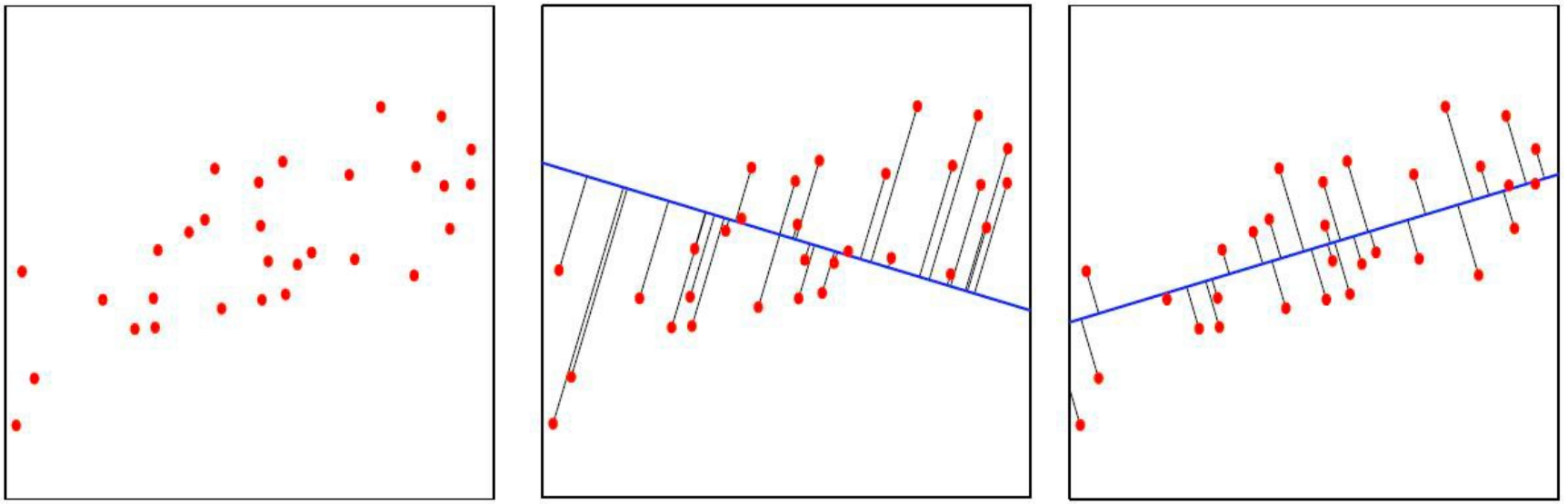
- Note: $\lambda_{1,2}$ is rotationally invariant, but orientation θ is not.

Dimensionality of data

What is the true dimensionality of this data?



Fit data to low-dimensional model



Dimensionality reduction (embedding)

- Assign instances to real-valued vectors, in a space that is of much lower dimension (even 2D or 3D for visualization).
- Approximately preserve similarity/distance relationships between instances.
- Some techniques:

  Linear:

  – Principal components analysis (PCA)

  Non-linear:

  – Independent components analysis
  – Self-organizing maps
  – Gaussian process latent variable models
  – Deep auto-encoders

Basic idea

- Project high-dimensional data onto a lower dimensional subspace using linear or non-linear transformations.

$$x = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \Rightarrow \text{reduce dimensionality} \Rightarrow y = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_K \end{bmatrix} \quad (K << N)$$

- The goal is to reduce the dimensionality of data, while retaining as much as possible of the variation present in the dataset.

Principal Component Analysis (PCA)

- Find a basis in a lower dimensional sub-space.
- Approximate vectors by projected them to the lower dim space.

  (1) Original space representation

  $$x = a_1 v_1 + a_2 v_2 + \cdots + a_N v_N$$

  where $v_1, v_2, \ldots v_N$ is a base in original N-dim space.

  (2) Lower dimensional sub-space representation

  $$\hat{x} = b_1 u_1 + b_2 u_2 + \cdots + b_K u_K$$

  where $u_1, u_2, \ldots, u_K$ is a base in original K-dim sub-space.
- Note: if $K = N$, then $\hat{x} = x$.

Example: $K = N$

$$v_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, v_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, v_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \text{(standard basis)}$$

$$x_v = \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix} = 3v_1 + 3v_2 + 3v_3$$

$$u_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, u_2 = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}, u_3 = \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix} \text{(another basis)}$$

$$x_u = \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix} = 3u_1 + 0u_2 + 0u_3$$

Thus $x_v = x_u$, but represented with another base.

PCA: Methodology

- Suppose $x_1, x_2, \ldots, x_M$ are $N \times 1$ vectors.
- Step 1: Compute the mean

$$\bar{x} = \frac{1}{M} \sum_{i=1}^{M} x_i$$

- Step 2: Subtract the mean

$$\phi_i = x_i - \bar{x}$$

- Step 3: From the matrix $A = [\phi_1 \phi_2 \ldots \phi_M]$, compute sample covariance matrix

$$C = \frac{1}{M} \sum_{i=1}^{M} \phi_i \phi_i^T = \frac{1}{M} A A^T$$

- Step 4: Compute the eigenvalues of C: $\lambda_1 > \lambda_2 > \cdots > \lambda_N$.

PCA: Methodology

- Step 5: Compute the eigenvectors of C: $u_1, u_2, \dots, u_N$. The eigenvectors form a basis for $x - \bar{x}$.

$$x - \bar{x} = b_1 u_1 + b_2 u_2 + \cdots + b_N u_N = \sum_{i=1}^{N} b_i u_i$$

$$b_i = u_i^T (x - \bar{x})$$

- Step 6 (dimensionality reduction step): Keep only the terms corresponding to the K largest eigenvalues:

$$\hat{x} - \bar{x} = \sum_{i=1}^{K} b_i u_i \,, K \ll N$$

- The representation of $\hat{x} - \bar{x}$ in basis $u_1, u_2, \dots, u_K$ is thus $\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_K \end{bmatrix}$

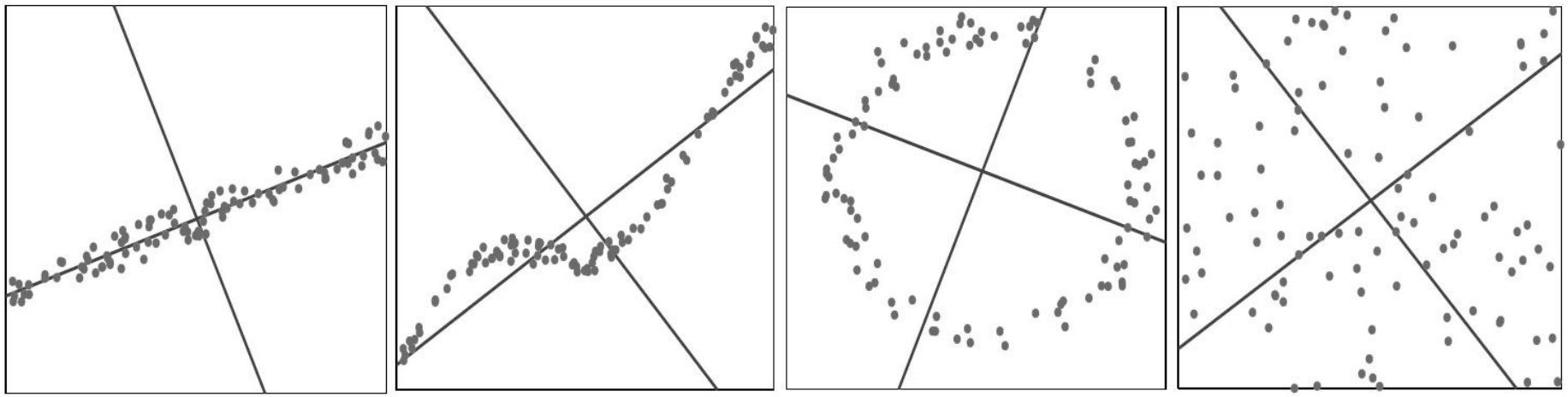
PCA: Methodology

- The linear transformation $\mathbb{R}^N \rightarrow \mathbb{R}^K$ that performs dimensionality reduction is:

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_K \end{bmatrix} = \begin{bmatrix} u_1{}^T \\ u_2^T \\ \vdots \\ u_K^T \end{bmatrix} (x - \bar{x}) = U^T (x - \bar{x})$$

- PCA projects the data along the directions, where the data varies the most.
- These directions are determined by the eigenvectors of the covariance matrix corresponding to the largest eigenvalues.
- The magnitude of the eigenvalues corresponds to the variance of the data along the eigenvector directions.

Example



What would the eigenvalues be in these cases? Does a PCA make sense?

Think bigger! - Face images

- Assume you have 92×112 grayscale images of faces. Each image can be considered as a point in high-dimensional face space.

- For each image, stack all pixels into a large vector of size $92 \cdot 112 = 10304$ and apply principal component analysis.

Eigenfaces

- We can rearrange the eigenvectors into (eigenface) images.
- The upper-left image is the mean, then eigenfaces in decreasing order of eigenvalues.

Summary of good questions

- How does a Hough transform work for lines?
- How many accumulators should you use?
- How does RANSAC work?
- What is a homography?
- How do you create a connected component?
- In what ways can two shape descriptors be different?
- Can you give two examples of two shape descriptors?
- How do you compute moment descriptors?
- What do you like to preserve with dimensionality reduction?
- How does a PCA work?

Recommended reading

- Gonzalez & Woods: Chapters 10.2 and 11.3 - 11.5
- Szeliski: Chapters 5.2.3, 7.4 and 8.1