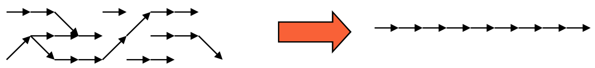
Grouping

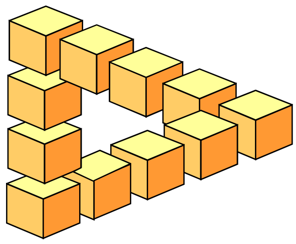
To make sense of image elements, they first have to be grouped into larger structures.

<u>Example</u>: Grouping noisy edge elements into a straight edge



<u>Essential problem</u>:

Obtaining globally valid results by local decisions



- locally compatible
- globally incompatible

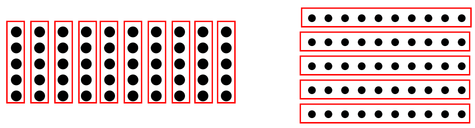
<u>Important methods</u>:

- Fitting
- Clustering
- Hough Transform
- Relaxation

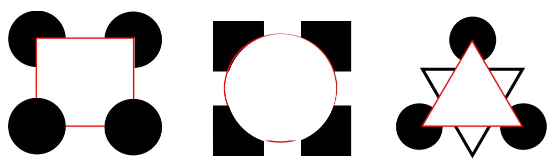
1

Cognitive Grouping

The human cognitive system shows remarkable grouping capabilities



grouping into rows or columns according to a distance criterion



grouping into virtual edges

• • • • • •

grouping into virtual motion

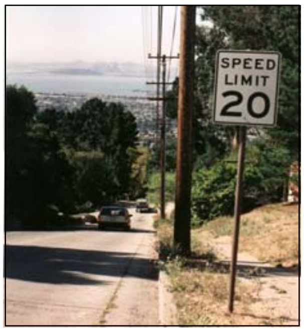
It is worthwhile wondering which cognitive grouping rules should also be followed by machine vision

2

Fitting Straight Lines

Why do we want to discover straight edges or lines in images?

- **Straight edges occur abundantly in the civilized world.**
- **Approximately straight edges are also important to model many natural phenomena, e.g. stems of plants, horizon at a distance.**
- **Straightness in scenes gives rise to straighness in images.**
- **Straightness discovery is an example of constancy detection which is at the heart of grouping (and maybe even interpretation).**



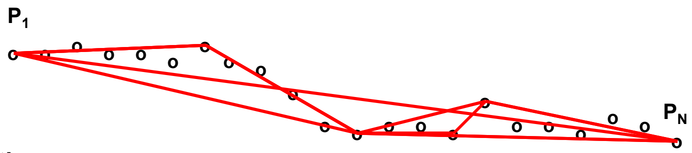
We will treat several methods for fitting straight lines:

- **Iterative refinement**
- **Mean-square minimization**
- **Eigenvector analysis**
- **Hough transform**

3

Straight Line Fitting by Iterative Refinement

Example: Fitting straight segments to a given object motion trajectory



Algorithm:

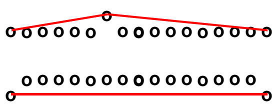
A: First straight line is P_1P_N

B: Is there a straight line segment P_iP_k with an intermediate point P_j ($i < j < k$) whose distance from P_iP_k is more than d? If no, then terminate.

C: Segment P_iP_k into P_iP_j and P_jP_k and go to B.

Advantage: simple and fast

Disadvantages:
- strong effect of outliers
- not always optimal

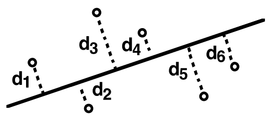


4

Straight Line Fitting by Eigenvector Analysis (1)

Given: $(x_i\ y_i)$ $i = 1 \dots N$

Wanted: Coefficients c_0, c_1 for straight line $y = c_0 + c_1 x$ which minimizes $\Sigma\, d_i^2$



Observation:

The optimal straight line passes through the mean of the given points. Why?

Let $(x'y')$ be a coordinate system with the x' axis parallel to the optimal straight line.

optimal straight line	$x' = x_0'$
error	$\Sigma\, d_i^2 = \Sigma\, (x_i' - x_0')^2$
condition for optimum	$\delta/\delta x_0 \{\Sigma\, (x_i' - x_0')^2\} = -2 \bullet \Sigma\, (x_i' - x_0') = 0$
	$x_0' = 1/N \bullet \Sigma\, x_i'$

A new coordinate system may be chosen with the origin at the mean of the given points:

$$x_j' = x_j - \frac{\sum x_i}{N} \qquad y_j' = y_j - \frac{\sum y_i}{N}$$

Optimal straight line passes through origin, only direction is unknown.

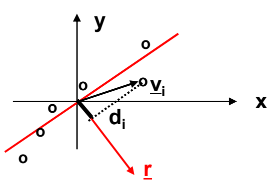
5

Straight Line Fitting by Eigenvector Analysis (2)

After coordinate transformation the new problem is:

Given: points $\underline{v}_i^T = [x_i\ y_i]$ with $\Sigma\, \underline{v}_i = \underline{0}$ $i = 1 \dots N$

Wanted: direction vector $\underline{r}$ which minimizes $\Sigma\, d_i^2$



Minimize $$d^2 = \sum_{i=1}^{N} d_i^2 = \sum_{i=1}^{N} (\underline{r}^T \underline{v}_i)^2 = \sum_{i=1}^{N} (\underline{r}^T \underline{v}_i)(\underline{v}_i^T \underline{r}) = \underline{r}^T S \underline{r}$$

S ← scatter matrix

Minimization with Lagrange multiplier λ:

$\underline{r}^T S \underline{r} + \lambda \underline{r}^T \underline{r}$ => minimum subject to $\underline{r}^T \underline{r} = 1$

Minimizing $\underline{r}$ is eigenvector of S, minimum is eigenvalue of S.

For a 2D scatter matrix there exist 2 orthogonal eigenvectors:

$\underline{r}_{min}$	orthogonal to optimal straight line
$\underline{r}_{max}$	parallel to optimal straight line

6

Straight Line Fitting by Eigenvector Analysis (3)

Computational procedure:

- Determine mean m of given points with $m_x = 1/N\ \Sigma\ x_i$, $m_y = 1/N\ \Sigma\ y_i$, i = 1 ... N
- Determine scatter matrix S = $\begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} = \begin{bmatrix} \Sigma (x_i-m_x)^2 & \Sigma (x_i-m_x)(y_i-m_y) \\ \Sigma (x_i-m_x)(y_i-m_y) & \Sigma (y_i-m_y)^2 \end{bmatrix}$
- Determine maximal eigenvalue

$$\lambda_{1,2} = \frac{S_{11}+S_{22}}{2} \pm \sqrt{\left(\frac{S_{11}+S_{22}}{2}\right)^2 - |S|} \qquad \lambda_{max} = \max\{\lambda_1, \lambda_2\}$$

- Determine direction of eigenvector corresponding to λ_{max}

$S_{11}\, r_x + S_{12}\, r_y = \lambda_{max}\, r_x$ by definition of eigenvector => r_y/r_x

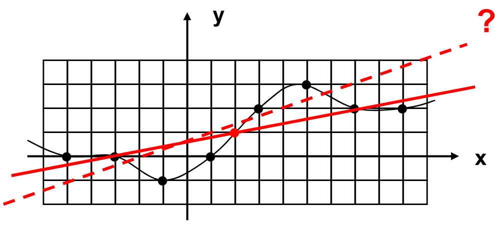
- Determine optimal straight line

$(y- m_y) = (x-m_x)\ (r_y/r_x) = (x-m_x)\ (\lambda_{max} - S_{11})/S_{12}$

7

Example for Straight Line Fitting by Eigenvector Analysis

What is the best straight-line approximation of the contour?



Given points: { (-5 0) (-3 0) (-1 -1) (1 0) (3 2) (5 3) (7 2) (9 2) }

Center of gravity: $m_x = 2$ $m_y = 1$

Scatter matrix: $S_{11} = 168$ $S_{12} = S_{21} = 38$ $S_{22} = 14$

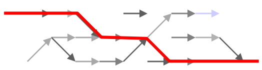
Eigenvalues: $\lambda_1 = 176{,}87$ $\lambda_2 = 5{,}13$

Direction of straight line: $r_y/r_x = 0{,}23$

Straight line equation: $y = 0{,}23\ x + 0{,}54$

8

Grouping by Search



What is the "best path" which could represent a boundary in a given field of edgels?

The problem can be formulated as a search problem:

What is the best path from a starting point to an end point, given a cost function $c(x_1, x_2, \dots, x_N)$?

The variables $x_1 \dots x_N$ are decision variables whose values determine the path.

Unfortunately, the total cost $c(x_1, \dots, x_N)$ is in general not minimized by local minimal cost decisions min $c(x_i)$, e.g. following the path of maximal edgel strength.

Hence search for a global optimum is necessary, e.g.

- hill climbing
- A* search
- **Dynamic Programming**

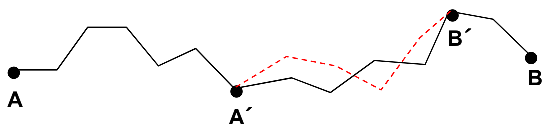
Dynamic Programming (1)

Dynamic Programming is an optimization method which can be applied if the global cost $c(x_1, x_2, \dots, x_N)$ obeys the <u>principle of optimality</u>:

If $a_1, a_2, \dots, a_N$ minimize $c(x_1, x_2, \dots, x_N)$,
then $a_i, a_{i+1}, \dots, a_k$ minimize $c(a_i, x_{i+1}, x_{i+2}, \dots, x_{k-1}, a_k)$

Hence, for a globally optimal path every subpath has to be optimal.

<u>Example</u>: In street traffic, an optimal path from A to B usually implies that all subpaths from A´ to B´ between A and B are also optimal.



Dynamic Programming avoids cost computations for all value assignments for $x_1, x_2, \dots, x_N$.
If each x_i, $i = 1 \dots N$, has K possible values, only $N{*}K^2$ cost computations are required instead of K^N.

Dynamic Programming (2)

Suppose $c(x_1, x_2, \ldots, x_N) = c(x_1, x_2) + c(x_2, x_3) + \ldots + c(x_{N-1}, x_N)$, then the optimality principle holds.

Dynamic Programming:

Step 1: Minimize $c(x_1, x_2)$ over x_1 => $f_1(x2)$
Step 2: Minimize $f_1(x2)+c(x_2, x_3)$ over x_2 => $f_2(x3)$
Step 3: Minimize $f_2(x3)+c(x_3, x_4)$ over x_3 => $f_3(x4)$
•
•
•
Step N: Minimize $f_{N-1}(x_N)$ over x_N => $f_N = \min c(x_1, x_2, \ldots, x_N)$

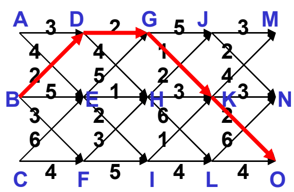
Example of a cost function for boundary search:
"Punish accumulated curvature and reward accumulated edge strengths"

$$c(x_1,\ldots,x_N) = \sum_{k=1..N} (1 - s(x_k)) + \alpha \sum_{k=1..N-1} q(x_k, x_{k+1})$$

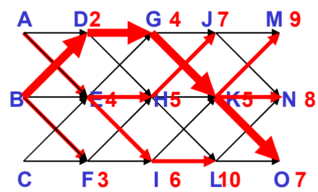
$s(x_k)$ edge strength
$q(x_k, x_{k+1})$ curvature

Dynamic Programming (3)

<u>Example</u>: Find optimal path from left to right



optimaler Pfad?



optimaler Pfad!

- Find best paths from A, B, C to D, E, F, record optimal costs at D, E, F
- Find best paths from D, E, F to G, H, I, record optimal costs at G, H, I

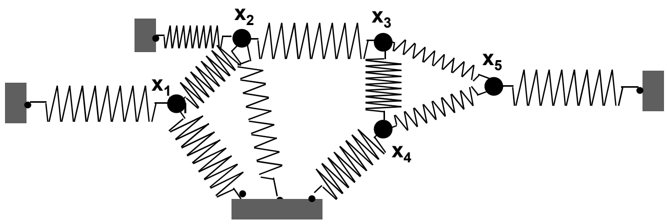
etc.

- Trace back optimal path from right to left

Grouping by Relaxation

Relaxation methods seek a solution by stepwise minimization ("relaxation") of constraints.

Analogy with spring system:

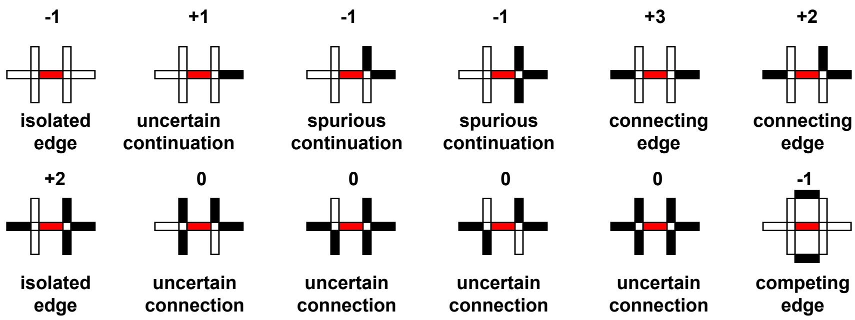


Variables x_i take on values (= positions) where springs are maximally relaxed corresponding to a state of global minimal energy. Hence relaxation is often realized by "energy minimization".

Contexts for Edge Relaxation

Iterative modification of edge strengths using context-dependent compatibility rules.

Context types:



Each context contributes with weight $w_j = w_0 \cdot \{-1 \ldots +2\}$ to an interative modification of the edge strength of the central element.

Modification Rule for Edge Relaxation

P_i^k	edge strength in position i after iteration k
Q_{ij}^k	strength of context j for position i after iteration k
w_j	weight factor of context j

$Q_{ij}^k = \prod P_m^k \bullet \prod (1-P_n^k)$ edge context strength

m, n ranging over all supporting and not supporting edge positions of context j, respectively.

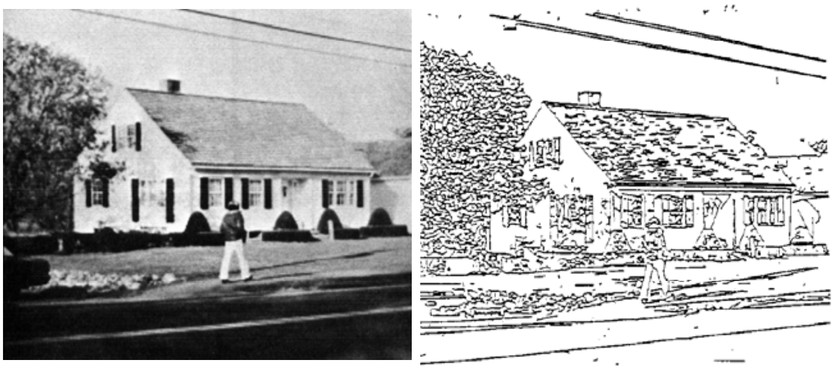
$$P_i^{k+1} = P_i^k \frac{1+\Delta P_i^k}{1+P_i^k \Delta P_i^k}$$ edge strength modification rule

$$\Delta P_i^k = \sum_{j=1}^{N} w_j Q_{ij}^k$$ edge strength increment

There is empirical evidence (but no proof) that for most edge images this relaxation procedure converges within 10 ... 20 iterations.

15

Example of Edge-finding by Relaxation



Landhouse scene from VISIONS project, 1982

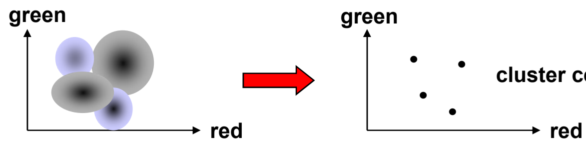
16

Histogram-based Segmentation with Relaxation (1)

<u>Basic idea</u>:

Use relaxation to introduce a local similarity constraint into histogram-based region segmentation.

A Determine cluster centers by multi-dimensional histogram analysis



B Label each pixel by cluster-membership probabilities p_i , 1 = 1 ... N

$$p_i = \frac{1/d_i}{\sum_{k=1}^{N} 1/d_k}$$

d_i is Euclidean distance between the feature vector of the pixel and cluster center $\underline{c}_i$

Histogram-based Labelling with Relaxation (2)

C Iterative relaxation of the $p_i(j)$ of all pixels j:

- equal labels of neighbouring pixels support each other
- unequal labels of neighbouring pixels inhibit each other

$$q_i(j) = \sum_{k \in D(j)} [w^+ p_i(k) - w^- (1 - p_i(k))]$$

D(j) is neighbourhood of pixel j

$$p_i'(j) = \frac{p_i(j) + q_i(j)}{\sum_n (p_n(j) + q_n(j))}$$

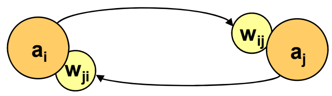
new probability $p_i'(j)$ of pixel j to belong to cluster i

D Region assignment of each pixel according to its maximal membership probability max p_i

E Recursive application of the procedure to individual regions

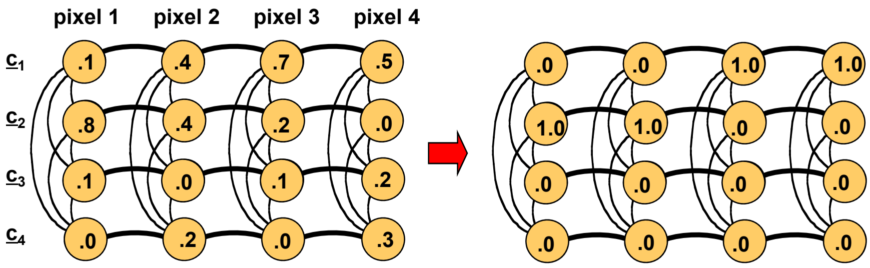
Relaxation with a Neural Network

Principle:



cells influence each other´s activation via exciting or inhibiting weights

Relaxation labelling of 4 pixels:



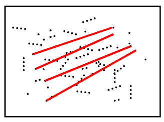
bidirectional inhibiting connection

bidirectional exciting connection

19

Hough Transform (1)

Robust method for fitting straight lines, circles or other geometric figures which can be described analytically.

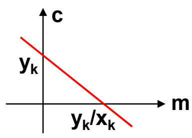


Given: Edge points in an image

Wanted: Straight lines supported by the edge points

An edge point (x_k, y_k) supports all straight lines $y = mx + c$ with parameters m and c such that $y_k = mx_k + c$.

The locus of the parameter combinations for straight lines through (x_k, y_k) is a straight line in parameter space.



Principle of Hough transform for straight line fitting:

- Provide accumulator array for quantized straight line parameter combinations
- For each edge point, increase accumulator cells for all parameter combinations supported by the edge point
- Maxima in accumulator array correspond to straight lines in the image

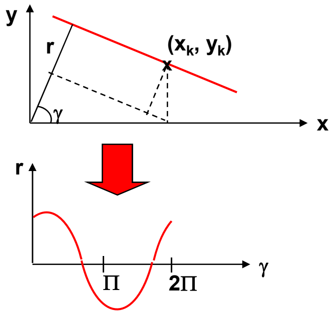
20

Hough Transform (2)

For straight line finding, the parameter pair (r, γ) is commonly used because it avoids infinite parameter values:

$$x_k \cos\gamma + y_k \sin\gamma = r$$

Each edge point (x_k, y_k) corresponds to a sinusoidal in parameter space:



Important improvement by exploiting direction information at edge points:

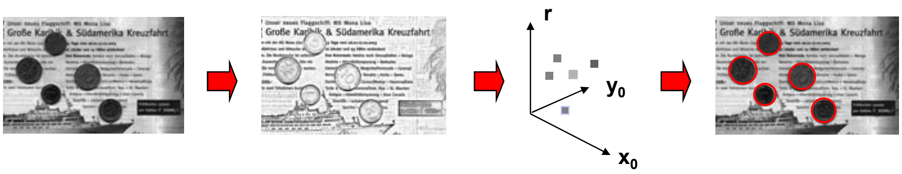
(x_k, y_k, φ) ⇒ $x_k \cos\gamma + y_k \sin\gamma = r$ **restricted to** $\varphi - \delta \le \gamma \le \varphi + \delta$

gradient direction (↑ φ) **direction tolerance** (↑ δ)

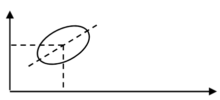
Hough Transform (3)

Same method may be applied to other parameterizable shapes, e.g.

- **circles** $(x_k - x_0)^2 + (y_k - y_0)^2 = r^2$ **3 parameters x_0, y_0, r**



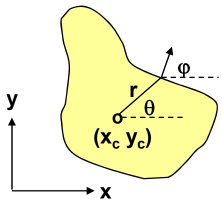
- **ellipses** $$\left(\frac{(x_k - x_0)\cos\gamma + (y_k - y_0)\sin\gamma}{a}\right)^2 + \left(\frac{(y_k - y_0)\cos\gamma - (x_k - x_0)\sin\gamma}{b}\right)^2 = 1$$ **5 parameters x_0, y_0, a, b, γ**



Accumulator arrays grow exponentially with number of parameters
=> quantization must be chosen with care

Generalized Hough Transform

- shapes are described by edge elements ($r\ \theta\ \varphi$) relative to an arbitrary reference point ($x_c\ y_c$)
- φ is used as index into ($\rho\ \theta$) pairs of a shape description
- edge point coordinates ($x_k\ y_k$) and gradient direction φ_k determine possible reference point locations
- likely reference point locations are determined via maxima in accumulator array



φ_1: $\{(r_{11}\ \theta_{11})\ (r_{12}\ \theta_{12})\ \dots\}$
φ_2: $\{(r_{21}\ \theta_{11})\ (r_{22}\ \theta_{12})\ \dots\}$
⋮
φ_N: $\{(r_{N1}\ \theta_{11})\ (r_{N2}\ \theta_{12})\ \dots\}$

$(x_k\ y_k\ \varphi_k) \Rightarrow \{(x_c\ y_c)\} = \{\ (x_k - r_i(\varphi_k) \cos \theta_i(\varphi_k),\ \ (y_k - r_i(\varphi) \sin \theta_i(\varphi_k))\ \}$

↳ counter cell in accumulator array

23