SEGMENTATION AND CONTOUR FINDING IN IMAGES

Hermann Ney

Human Language Technology and Pattern Recognition
Lehrstuhl für Informatik VI
Computer Science Department
RWTH Aachen – University of Technology
D-52056 Aachen, Germany

OVERVIEW

segmentation of one–dim. signals and and contours:

- **segmentation of one–dim. signals**
  - **problem formulation: squared error criterion**
  - **optimization method: dynamic programming**
  - **extensions: error criterion: absolute value error**
    probabilistic model (maximum likelihood estimation)
  - **model: general case**
  - **model: straight line**

- **segmentation of image contours (= two–dim. signals)**
  - **problem formulation**
  - **error criterion: vertical vs. orthogonal least squares**
  - **model: straight lines (= polygon approximation)**
  - **model: circular arcs**
  - **use of prior knowledge**

OVERVIEW

contour finding and image matching:

- **contour finding in images**
  - **problem formulation – one–dim. contour:** $x \rightarrow y(x)$
  - **two–dim. contour:** $n \rightarrow [x_n, y_n]$
  - **use of prior knowledge**
- **image registration and matching**
  - **HMM (Hidden Markov Model): matching for one–dim. signals**
  - **Pseudo 2–D HMM**
  - **Moore's algorithm**
  - **Sakoe's algorithm**

REFERENCES: dynamic programming

J.S. Bridle, N.C. Sedgewick, "A method for segmenting acoustic patterns, with applications to automatic speech recognition," Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Hartford, CN, pp. 656-659, May 1977.

V.A. Kovalevsky. Sequential Optimization in Pattern Recognition and Pattern Description. Proc. IFIP Congr., Int. Federation for Information Processing, Edinburgh, North-Holland Pub. Co., Amsterdam, pp. 1603-1607, 1968

U. Montanari. On the Optimal Detection of Curves in Noisy Pictures. Communications of the ACM, 14(5), pp. 335-345, May 1971

A. Martelli. An Application of Heuristic Search Methods to Edge and Contour Detection. Communications of the ACM, 19(2), pp. 73-83 February 1976.

H. Ney: Konturbestimmung in Bildern mit dynamischer Programmierung. DAGM-Symposium "Modelle und Strukturen", DAGM, Hamburg, Informatik-Fachberichte, Springer, Berlin, Vol. 49, pp. 319-326, Okt. 1981.

V. Steinbiss, R. Ullrich, H. Ney: Curve Fitting for the Recognition of Line Drawings. Fourth European Signal Processing Conf., Grenoble, pp. 1449-1452, Sep. 1988.

L. Welling, H. Ney. "A model for efficient formant estimation". Proc. Int. Conf. on Acoustics, Speech and Signal Processing, Atlanta, GA, pp. 797-800, May 1996.

REFERENCES: 2-D Methods

O. Agazzi and S. Kuo. Pseudo Two-Dimensional Hidden Markov Models for Document Recognition. AT&T Technical Journal, pp. 60–72, September 1993.

S. Kuo and O. Agazzi. Keyword Spotting in Poorly Printed Documents Using Pseudo 2-D Hidden Markov Models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 16(8):842–848, August 1994.

E. Levin and R. Pieraccini. Dynamic Planar Warping for Optical Character Recognition. In ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech and Signal Processing, Vol. III, pp. 149–152, March 1992.

R. K. Moore. A Dynamic Programming Algorithm for the Distance Between Two Finite Areas. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1(1):86–88, January 1979.

S. Uchida and H. Sakoe. Piecewise Linear Two-Dimensional Warping. In 15th International Conf. on Pattern Recognition, Barcelona, Spain, Vol. 3, pp. 538–541, September 2000.

REFERENCES: Least squares

T. Pavlidis: Structural Pattern Recognition. Springer-Verlag, Berlin, 1977. pp. 179 ff.

G. Casella, R. L. Berger: Statistical Inference. Wadsworth & Brooks/Cole, Pacific Grove, CA, 1990. pp. 584 ff.

ONE-DIM. SIGNALS

constant function:

We are given a sequence of one–dim. measurements:

$$x_n \in \mathbb{R}, \quad n = 1, ..., N$$

We want to segment this sequence into $k = 1, ..., K$ segments.
Each segment k is represented by a constant but unknown value a_k.
The quantitative criterion is based on using the squared error:

$$\sum_{n=n_{k-1}+1}^{n_k} [x_n - a_k]^2$$

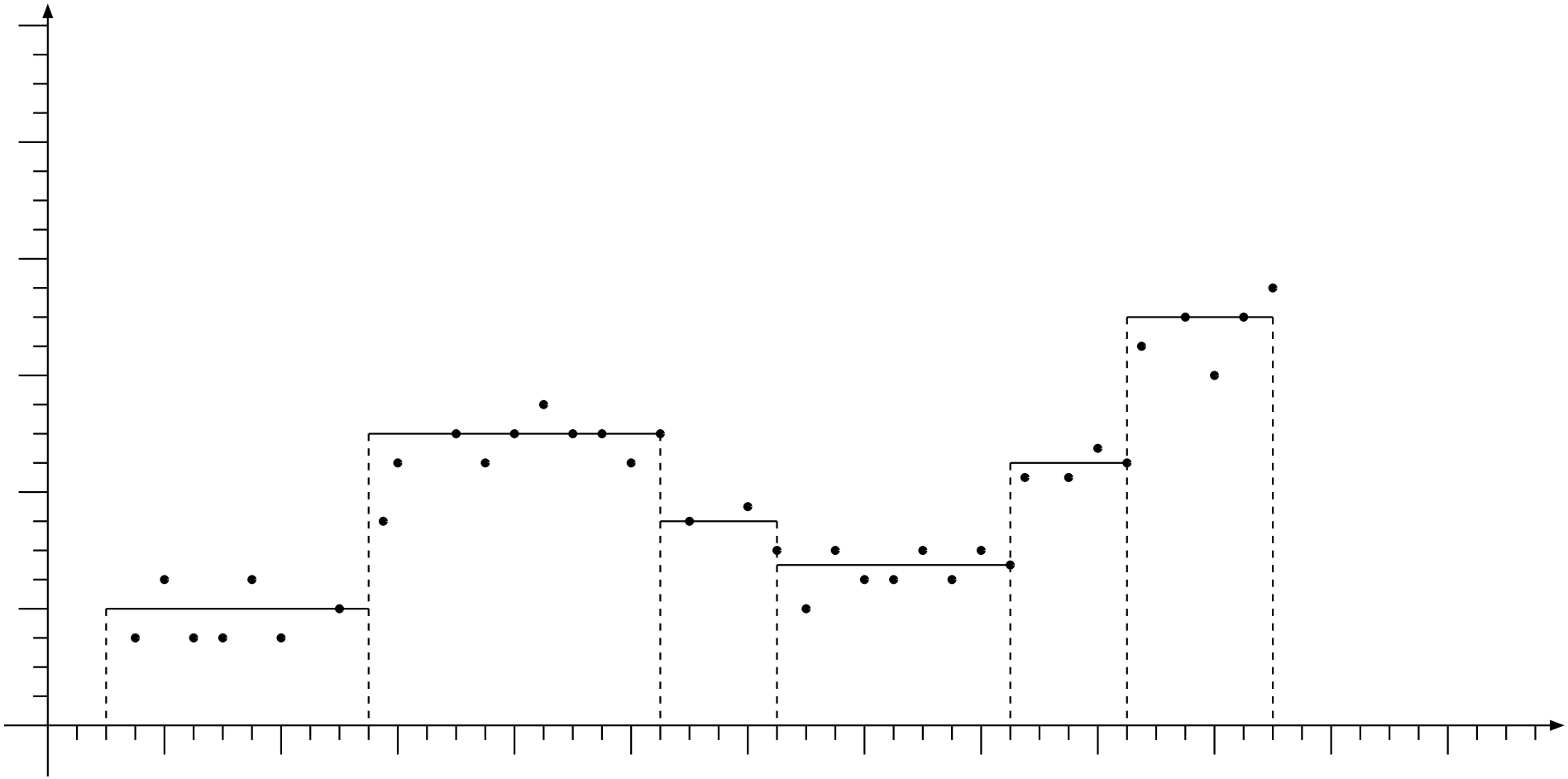
The global criterion then is to optimize over the unknown segment boundaries

$$n_1, ..., n_k, ..., n_K \quad := \quad n_1^K$$

with the definitions:

$$0 \equiv n_0 < n_1 < ... < n_{k-1} < n_k < ... < n_K \equiv N$$

segmentation of one–dim. signals



The criterion is to select the boundaries such that the sum over all local squared errors is a minimum:

$$\min_{n_1^K} \sum_{k=1}^{K} \min_{a_k} \left\{ \sum_{n=n_{k-1}+1}^{n_k} [x_n - a_k]^2 \right\}$$

where also the constant value a_k for each segment k is unkown.

number of possible segmentations:

$$\binom{N-1}{K-1} \approx \frac{(N-1)^{K-1}}{(K-1)!} \qquad [\text{ for } N \gg K]$$

we will design an algorithm (based on dynamic programming) with time complexity:

$$K \cdot N^2$$

constant function: local optimization over a_k

optimal value a_k for each segment k:

for each segment with boundaries $n = n_{k-1} + 1$ and $n = n_k$, we have:

$$\min_{a_k} \sum_{n=n_{k-1}+1}^{n_k} [x_n - a_k]^2$$

To carry out the minimization over a_k, we observe:

$$\begin{aligned}
\sum_{m=1}^{M} [y_m - A]^2 &= \sum_m y_m^2 - 2\,A \sum_m y_m + M \cdot A^2 \\
&= M \cdot \left[A - \frac{1}{M} \sum_m y_m \right]^2 + \sum_m y_m^2 - \frac{1}{M} \left(\sum_m y_m \right)^2
\end{aligned}$$

and we obtain:

$$\begin{aligned}
\arg\min_A \sum_{m=1}^{M} [y_m - A]^2 &= \frac{1}{M} \sum_m y_m =: \bar{y} \\
\min_A \sum_{m=1}^{M} [y_m - A]^2 &= \sum_m y_m^2 - \frac{1}{M} \left(\sum_m y_m \right)^2 = \sum_m y_m^2 - M\,\bar{y}^2
\end{aligned}$$

We introduce the auxiliary function $G(k,n)$:

$$G(k,n) \quad := \quad \min_{n_1^k : n_k = n} \left\{ \sum_{\kappa=1}^{k} \min_{a_\kappa} \sum_{i=n_{\kappa-1}+1}^{n_\kappa} [x_i - a_\kappa]^2 \right\}$$

i.e. the error of the best decomposition of the partial sequence x_1^n into k segments.

We decompose this quantity for k segments by going back to $(k-1)$ segments:

$$G(k,n) \;=$$

$$= \quad \min_{n_1^k : n_k = n} \left\{ \sum_{\kappa=1}^{k-1} \min_{a_\kappa} \sum_{i=n_{\kappa-1}+1}^{n_\kappa} [x_i - a_\kappa]^2 \;+\; \min_{a_k} \sum_{i=n_{k-1}+1}^{n_k} [x_i - a_k]^2 \right\}$$

$$= \quad \min_{m} \left\{ \min_{n_1^{k-1} : n_{k-1} = m} \left(\sum_{\kappa=1}^{k-1} \min_{a_\kappa} \sum_{i=n_{\kappa-1}+1}^{n_\kappa} [x_i - a_\kappa]^2 \right) \;+\; \min_{a_k} \sum_{i=m+1}^{n} [x_i - a_k]^2 \right\}$$

$$= \quad \min_{m} \left\{ G(k-1,m) + \min_{a_k} \sum_{i=m+1}^{n} [x_i - a_k]^2 \right\}$$

The approach we have applied is the following:

> **The original problem has been decomposed into smaller problems. We solve these smaller problems and compose the global solution from the solution to the smaller problems.**

other algorithms based on dynamic programming:

- **CYK parsing for context free grammars (CYK = Cocke–Younger–Kasami)**
- **Viterbi algorithm for finding the best path in Hidden Markov models**
- **algorithm for knapsack problem**
- **...**

To reconstruct the optimal segmentation, we introduce so-called backpointers $B(k,n)$ which keep track of the optimal segmentation when going from $(k-1)$ to k segments.
Thus we obtain the recursion:

$$G(k,n) = \min_m \left\{ G(k-1,m) + \min_{a_k} \sum_{i=m+1}^{n} [x_i - a_k]^2 \right\}$$

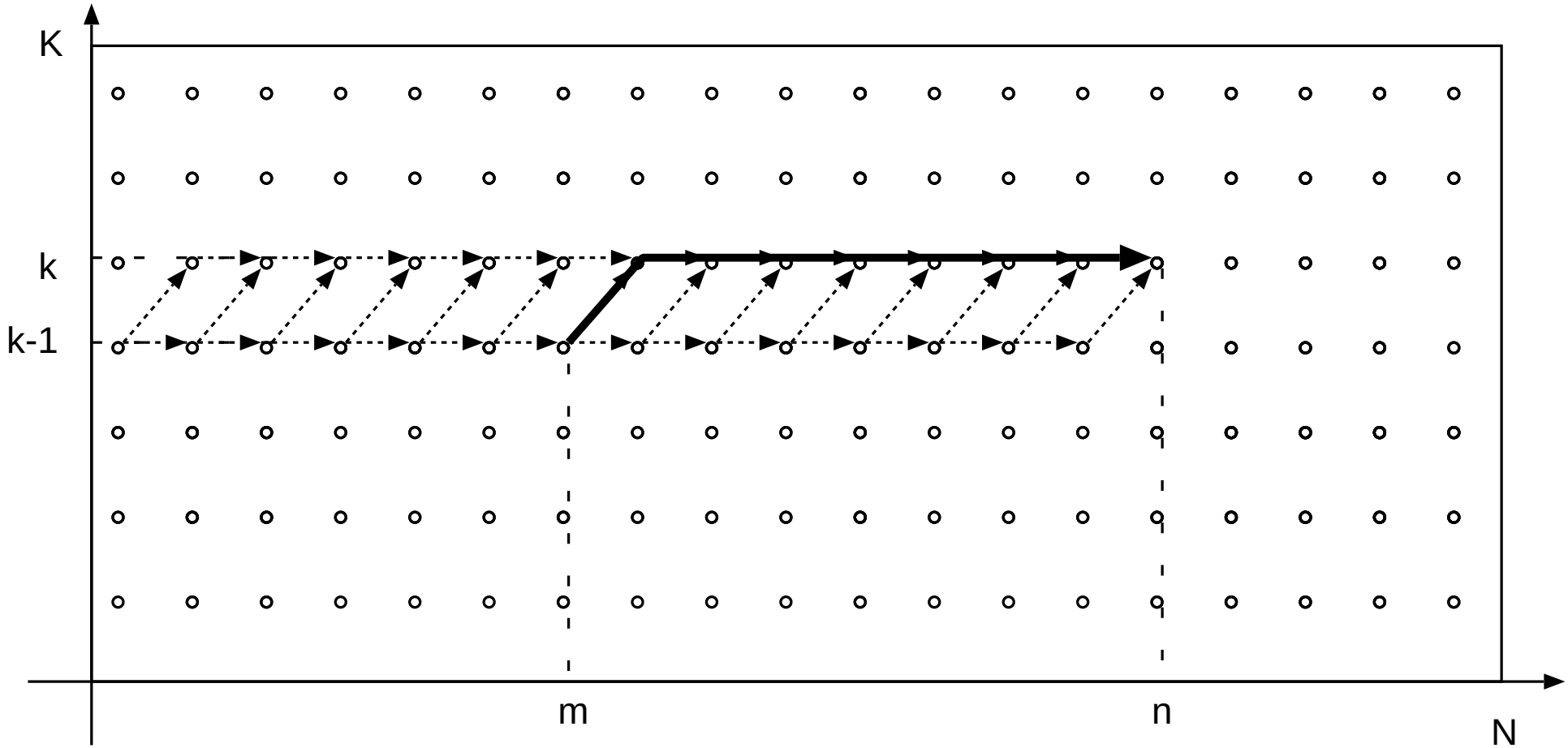
$$B(k,n) = \arg \min_m \left\{ G(k-1,m) + \min_{a_k} \sum_{i=m+1}^{n} [x_i - a_k]^2 \right\}$$

Note:

- This recursion should be implemented in the so-called ITERATIVE variant, i.e. by using an array or table for $G(k,n)$ (and $B(k,n)$).
- Alternative implementation:
  MEMOIZATION := recursive implementation + lookup table

resulting complexity (without the sum, i.e. using running sums): $K \cdot N^2$
modification: maximum segment length ΔN: $K \cdot N \cdot \Delta N$

segmentation of one–dim. signals: dynamic programming



initialization: $G(0,0) = 0, \quad G(0,n) = \infty$

for each segment $k = 1, ..., K$ **do**

for each observation $n = 1, ..., N$ **do**

$$G(k,n) := \min_m \left\{ G(k-1,m) + \min_{a_k} \sum_{i=m+1}^{n} [x_i - a_k]^2 \right\}$$

$$B(k,n) := \arg\min_m \left\{ G(k-1,m) + \min_{a_k} \sum_{i=m+1}^{n} [x_i - a_k]^2 \right\}$$

optimum value: $G(K,N)$

traceback: $n_K := N$

for each segment $k = K, ..., 1$ **do**

$n_{k-1} := B(k, n_k)$

note: use running sums for $\sum_i [x_i - a_k]^2$

running sums

To efficiently calculate $\sum_{i=m+1}^{n} [x_i - a]^2$ for varying m, n, a,
we compute only once (for all $n = 1, \ldots, N$):

$$S_n = \sum_{i=1}^{n} x_i \qquad\qquad Q_n = \sum_{i=1}^{n} x_i^2$$

then, we observe:

$$\begin{aligned}
\sum_{i=m+1}^{n} [x_i - a]^2 &= \sum_{i=m+1}^{n} [x_i^2 - 2ax_i + a^2] \\
&= \sum_{i=m+1}^{n} x_i^2 - 2a \sum_{i=m+1}^{n} x_i + a^2 \\
&= Q_n - Q_m - 2a(S_n - S_m) + a^2
\end{aligned}$$

segmentation of one–dim. signals: absolute value error

The squared error is replaced by the absolute value error:

$$\min_{n_1^K} \sum_{k=1}^{K} \min_{a_k} \left\{ \sum_{n=n_{k-1}+1}^{n_k} |x_n - a_k| \right\}$$

- **advantage: more robust with respect to outliers**
- **disadvantage: the optimum value a_k is the median, which is expected to require more computation**

probability distribution:
we assume a functional dependence for each model of segment k:

$$p_k(x|\lambda_k)$$

with a suitable parameter set λ_k.

The criterion is to find the segment boundaries n_1^K such that the sum over the negative logarithms of the probability model is a minimum:

$$\min_{n_1^K} \sum_{k=1}^{K} \min_{\lambda_k} \left\{ \sum_{n=n_{k-1}+1}^{n_k} [-\log p_k(x_n|\lambda_k)] \right\}$$

Here, the negative logarithm of the probability model $[-\log p_k(x_n|\lambda_k)]$ is interpreted as local error $[x_n - \hat{x}_n(k;\lambda_k)]^2$.

This is equivalent to maximizing the product of the probabilities over the unknown segment boundaries n_1^K and the unknown parameter sets λ_1^K:

$$\max_{n_1^K} \prod_{k=1}^{K} \max_{\lambda_k} \left\{ \prod_{n=n_{k-1}+1}^{n_k} p_k(x|\lambda_k)] \right\}$$

It is easy to see that we obtain the same type of dynamic programming recursion.

note: there is NO constraint on the continuity of the segments at the boundaries.

probability distribution:

$$\begin{aligned} p_k(x|\lambda_k) &= \mathcal{N}(x|\mu_k, \sigma_k^2) \\ &= \frac{1}{\sqrt{2\pi\sigma_k^2}} \cdot \exp\left[-\frac{1}{2\sigma_k^2}(x-\mu_k)^2\right] \\ -\log \mathcal{N}(x|\mu_k, \sigma_k^2) &= \frac{1}{2}\log\left(2\pi\sigma_k^2\right) + \frac{1}{2\sigma_k^2}(x-\mu_k)^2 \end{aligned}$$

case I: unknown means μ_k and known variances σ_k^2:

$$\min_{n_1^K} \sum_{k=1}^{K} \min_{\mu_k} \left\{ \sum_{n=n_{k-1}+1}^{n_k} [-\log \mathcal{N}(x_n|\mu_k, \sigma_k^2)] \right\}$$

case II: unknown means μ_k and unknown variances σ_k^2:

$$\min_{n_1^K} \sum_{k=1}^{K} \min_{\mu_k, \sigma_k^2} \left\{ \sum_{n=n_{k-1}+1}^{n_k} [-\log \mathcal{N}(x_n|\mu_k, \sigma_k^2)] \right\}$$

case III: unknown means μ_k and (known or unknown) equal variances $\sigma_k^2 = \sigma^2 \forall k$:

squared error model (constant offset $\frac{1}{2}\log\left(2\pi\sigma^2\right)$ and factor $\frac{1}{2\sigma^2}$ leave decisions unchanged)

general case:
we assume a functional dependence for each model of segment k:

$$n \quad \rightarrow \quad \hat{x}_n(k; \lambda_k)$$

with a suitable parameter set λ_k.

The criterion is to find the segment boundaries n_1^K such that the sum over all local squared errors is a minimum:

$$\min_{n_1^K} \sum_{k=1}^{K} \min_{\lambda_k} \left\{ \sum_{n=n_{k-1}+1}^{n_k} [x_n - \hat{x}_n(k; \lambda_k)]^2 \right\}$$

It is easy to see that we obtain the same type of dynamic programming recursion:

$$G(k,n) \quad = \quad \min_m \left\{ G(k-1,m) + \min_{\lambda_k} \sum_{i=m+1}^{n} [x_i - \hat{x}_i(k; \lambda_k)]^2 \right\}$$

As before, we want to segment the sequence x_1^N into $k = 1, ..., K$ segments. Each segment k is represented by a straight line with unknown parameters a_k and b_k:

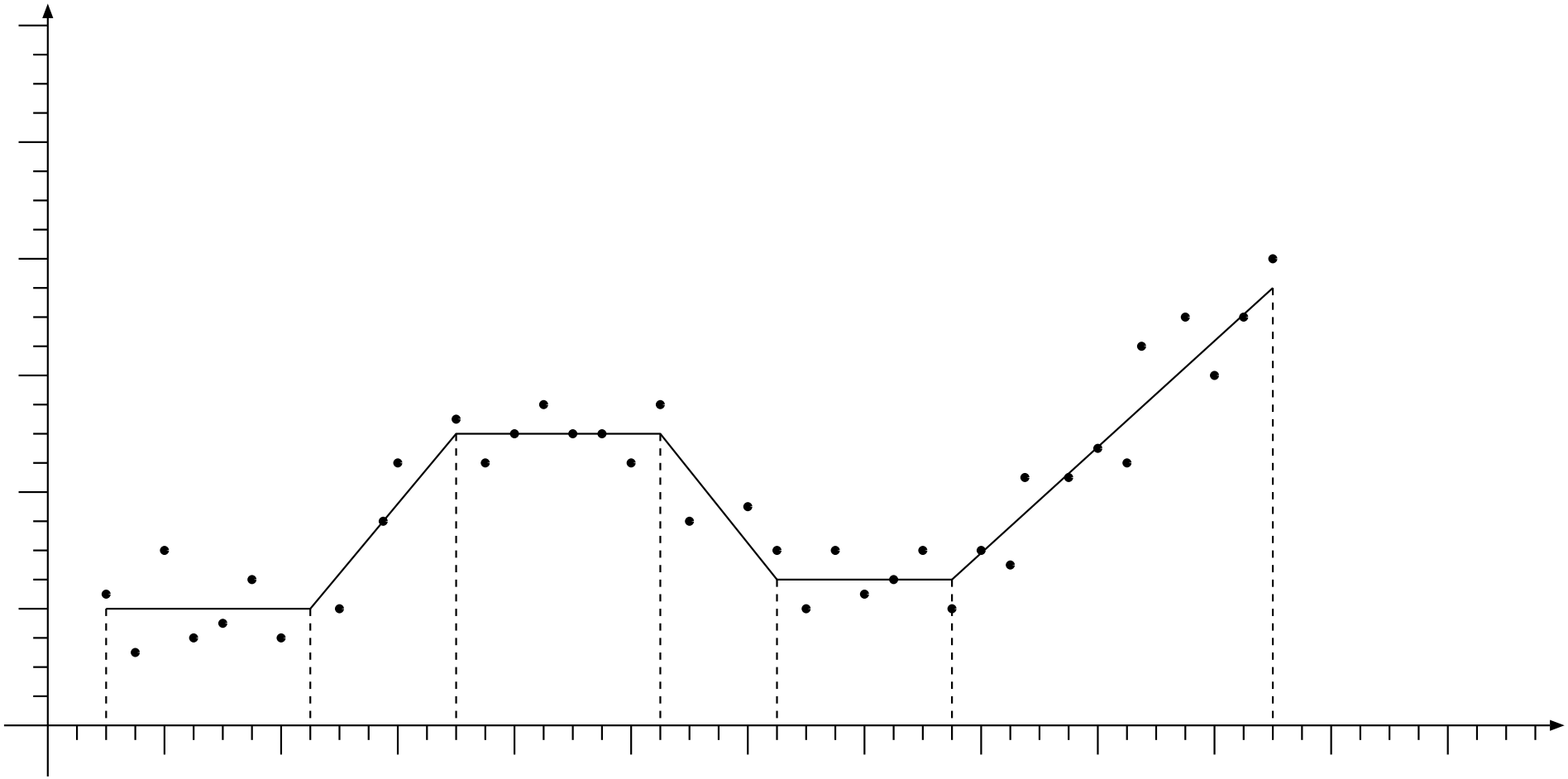
$$\hat{x}_n(k; a_k, b_k) \quad := \quad a_k + b_k \cdot n$$

The global criterion then is:

$$\min_{n_1^K} \sum_{k=1}^{K} \min_{a_k, b_k} \left\{ \sum_{n=n_{k-1}+1}^{n_k} [x_n - (a_k + b_k \cdot n)]^2 \right\}$$

implementation: see later (more general case: two–dim. signal with vertical least squares)

segmentation of one–dim. signals: straight line



SEGMENTATION OF IMAGE CONTOURS: TWO-DIM. SIGNAL

We are given the sequence of points (x_1^N, y_1^N) in the two–dim. plane.

We want to segment this sequence into $k = 1, ..., K$ segments.
Each segment k is represented by a straight line with unknown parameters a_k and b_k:

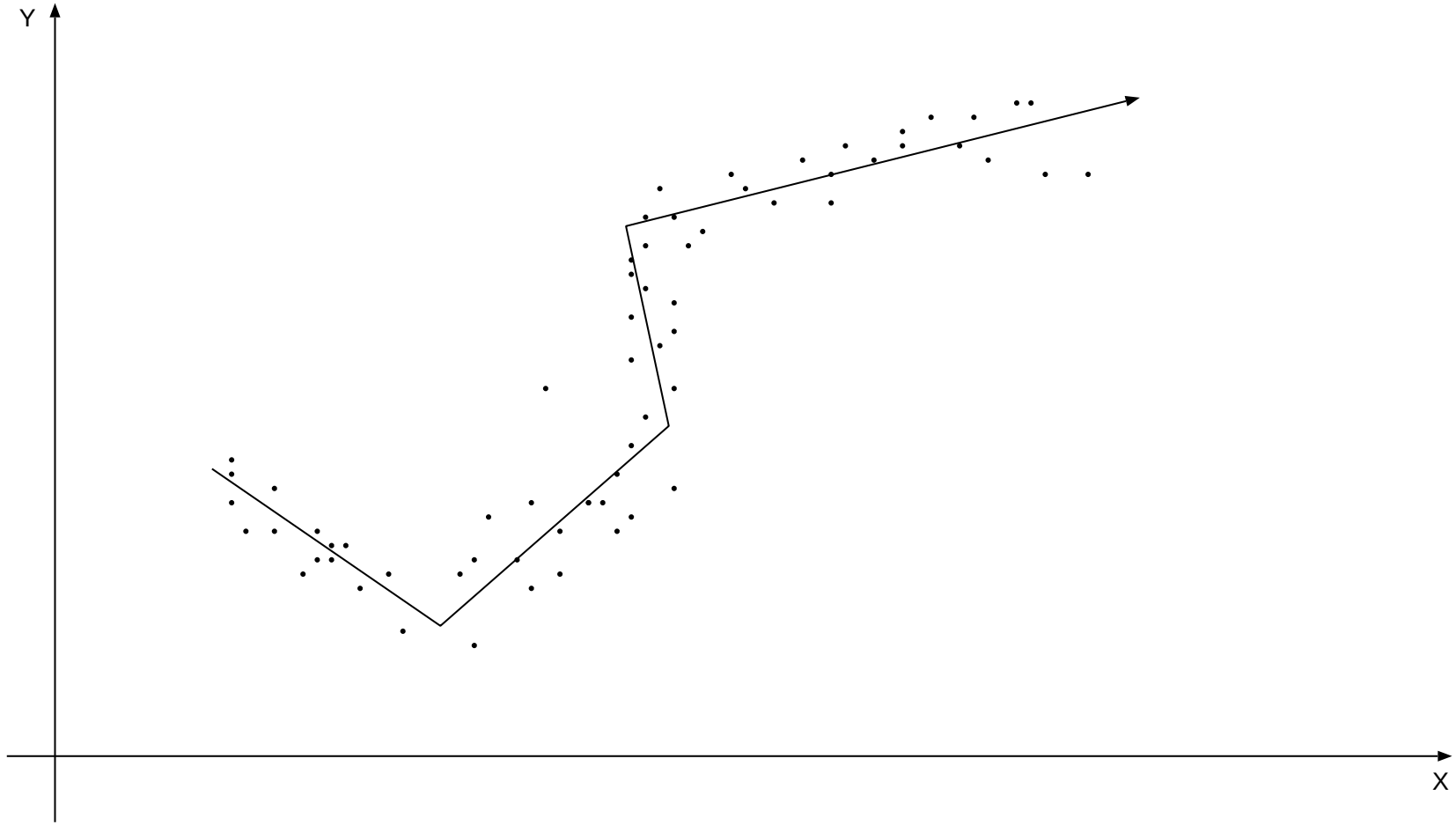
$$y_k(x) \quad := \quad a_k + b_k \cdot x$$

The global criterion then is:

$$\min_{n_1^K} \sum_{k=1}^{K} \min_{a_k, b_k} \left\{ \sum_{n=n_{k-1}+1}^{n_k} \left([y_n - y_k(x_n)]^2 \right) \right\}$$

[Note: This criterion does not take into account the difference at the segment boundaries $y_{k+1}(x_{n_k}) - y_k(x_{n_k})$. However, incorporation of this difference into the criterion is possible at the expense of a slightly more complicated search, but this extension will not be dealt with here.]

segmentation of image contours (two–dim. signals)



vertical least squares: local optimization

We re–write the vertical least squares error:

$$\sum_m [y_m - \hat{y}_m]^2 \quad = \quad \sum_m [y_m - (a + b\, x_m)]^2$$

For fixed parameter b, minimizing over a results in:

$$a \quad = \quad \bar{y} - b\, \bar{x}$$

Thus we obtain:

$$\begin{aligned} \min_{a,b} \left\{ \sum_m [y_m - \hat{y}_m]^2 \right\} \quad &= \quad \min_b \left\{ \sum_m [(y_m - \bar{y}) - b\, (x_m - \bar{x})]^2 \right\} \\ &= \quad \min_b \left\{ S_{yy} - 2\, b\, S_{xy} + b^2\, S_{xx} \right\} \end{aligned}$$

with the definitions of the elements of the scatter (empirical covariance) matrix:

$$\begin{aligned} S_{xx} \quad &:= \quad \sum_m (x_m - \bar{x})^2 \qquad S_{yy} \;:= \; \sum_m (y_m - \bar{y})^2 \\ S_{xy} \quad &:= \quad \sum_m (x_m - \bar{x})\, (y_m - \bar{y}) \end{aligned}$$

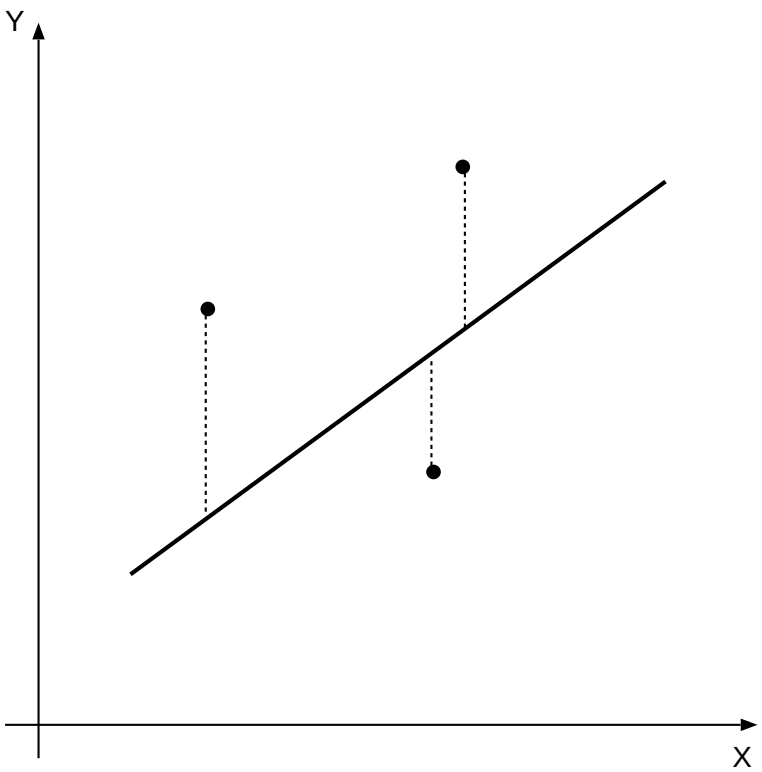
vertical least squares: local optimization

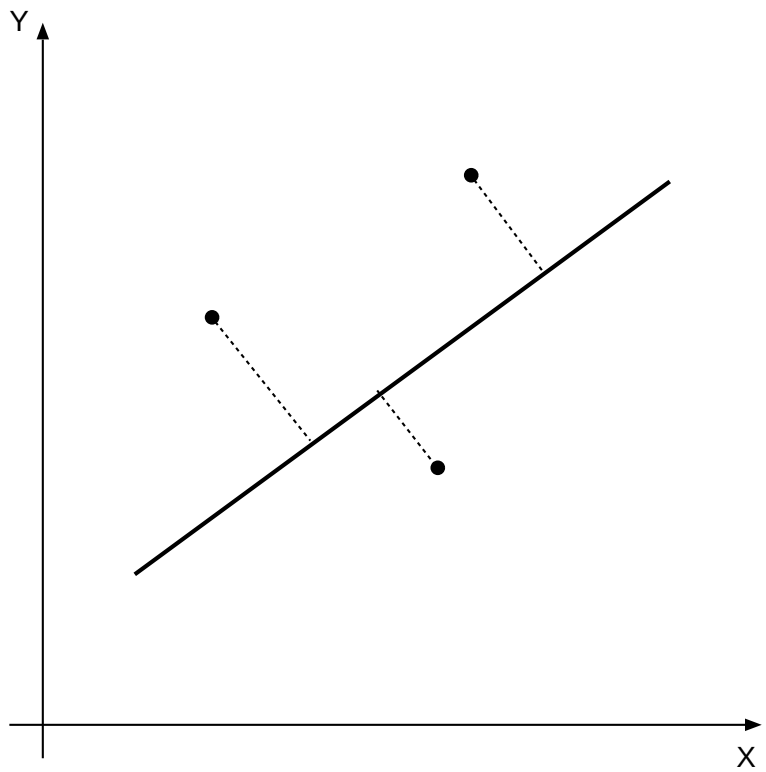
We re–write the vertical least squares error:

$$
\begin{aligned}
\min_b \left\{ S_{yy} - 2\,b\,S_{xy} + b^2\,S_{xx} \right\} &= \\
&= \min_b \left\{ S_{xx} \cdot \left[\frac{S_{yy}}{S_{xx}} - 2\,b\,\frac{S_{xy}}{S_{xx}} + b^2 \right] \right\} \\
&= \min_b \left\{ S_{xx} \cdot \left[\left(b - \frac{S_{xy}}{S_{xx}} \right)^2 + \frac{S_{yy}}{S_{xx}} - \left(\frac{S_{xy}}{S_{xx}} \right)^2 \right] \right\} \\
&= S_{xx} \cdot \left[\frac{S_{yy}}{S_{xx}} - \left(\frac{S_{xy}}{S_{xx}} \right)^2 \right] \\
&= S_{yy} - \frac{S_{xy}^2}{S_{xx}} \\
&= \text{local error of segment}
\end{aligned}
$$

implementation: running sums for S_{xx}, S_{xy}, S_{yy}

segmentation of image contours: orthogonal least squares





First, we define orthogonal least squares.

Each segment k is represented by a straight line with unknown parameters a_k and b_k:

$$y \quad := \quad a_k + b_k \cdot x$$

The global criterion then is:

$$\min_{n_1^K} \sum_{k=1}^{K} \min_{a_k, b_k} \left\{ \sum_{n=n_{k-1}+1}^{n_k} \left([x_n - \hat{x}_n(k)]^2 + [y_n - \hat{y}_n(k)]^2 \right) \right\}$$

where for each observed point (x_n, y_n), the point $(\hat{x}_n, \hat{y}_n)$ denotes the closest point on the line $y = a_k + b_k\, x$, i.e. the projection of each point (x_n, y_n) onto the straight line.

orthogonal least squares: local optimization

We re–write the orthogonal least squares error:

$$\begin{aligned}\sum_m \left([x_m - \hat{x}_m]^2 + [y_m - \hat{y}_m]^2\right) &= \ldots \\ &= \frac{1}{1+b^2} \sum_m [y_m - (a + b\, x_m)]^2\end{aligned}$$

For fixed parameter b, minimizing over a results in:

$$a = \bar{y} - b\,\bar{x}$$

Thus we obtain:

$$\begin{aligned}\min_{a,b} \left\{ \sum_m \left([x_m - \hat{x}_m]^2 + [y_m - \hat{y}_m]^2\right) \right\} &= \\ = \min_b \left\{ \frac{1}{1+b^2} \sum_m [(y_m - \bar{y}) - b\,(x_m - \bar{x})]^2 \right\} \\ = \min_b \left\{ \frac{1}{1+b^2} \left[S_{yy} - 2\,b\,S_{xy} + b^2\,S_{xx}\right] \right\}\end{aligned}$$

with the same definitions of the elements of the scatter (empirical covariance) matrix as before

orthogonal least squares: local optimization

elements of the scatter (empirical covariance) matrix:

$$S_{xx} := \sum_m (x_m - \bar{x})^2 \qquad S_{yy} := \sum_m (y_m - \bar{y})^2$$

$$S_{xy} := \sum_m (x_m - \bar{x})\,(y_m - \bar{y})$$

By taking the derivative with respect to b, we obtain the quadratic equation:

$$b^2 + b\,\frac{S_{xx} - S_{yy}}{S_{xy}} - 1 = 0$$

There are two solutions:

$$b = \frac{-(S_{xx} - S_{yy}) \pm \sqrt{(S_{xx} - S_{yy})^2 + 4\,S_{xy}^2}}{2\,S_{xy}}$$

From the two solutions, we need the solution with the "+" sign for the minimum.

Alternative approach:

The problem we have formulated is equivalent to the representation error in PCA (= Principal Component Analysis = Karhunen–Loève transformation). In PCA, we have to find the smallest eigenvalue $\lambda > 0$ and the corresponding eigenvector $u \in \mathbb{R}^2$:

$$S\,u \quad = \quad \lambda\,u$$

with the scatter matrix $S = \begin{pmatrix} S_{xx} & S_{xy} \\ S_{xy} & S_{yy} \end{pmatrix} \in \mathbb{R}^{2\times 2}$

The eigenvalues are obtained from the characteristic equation:

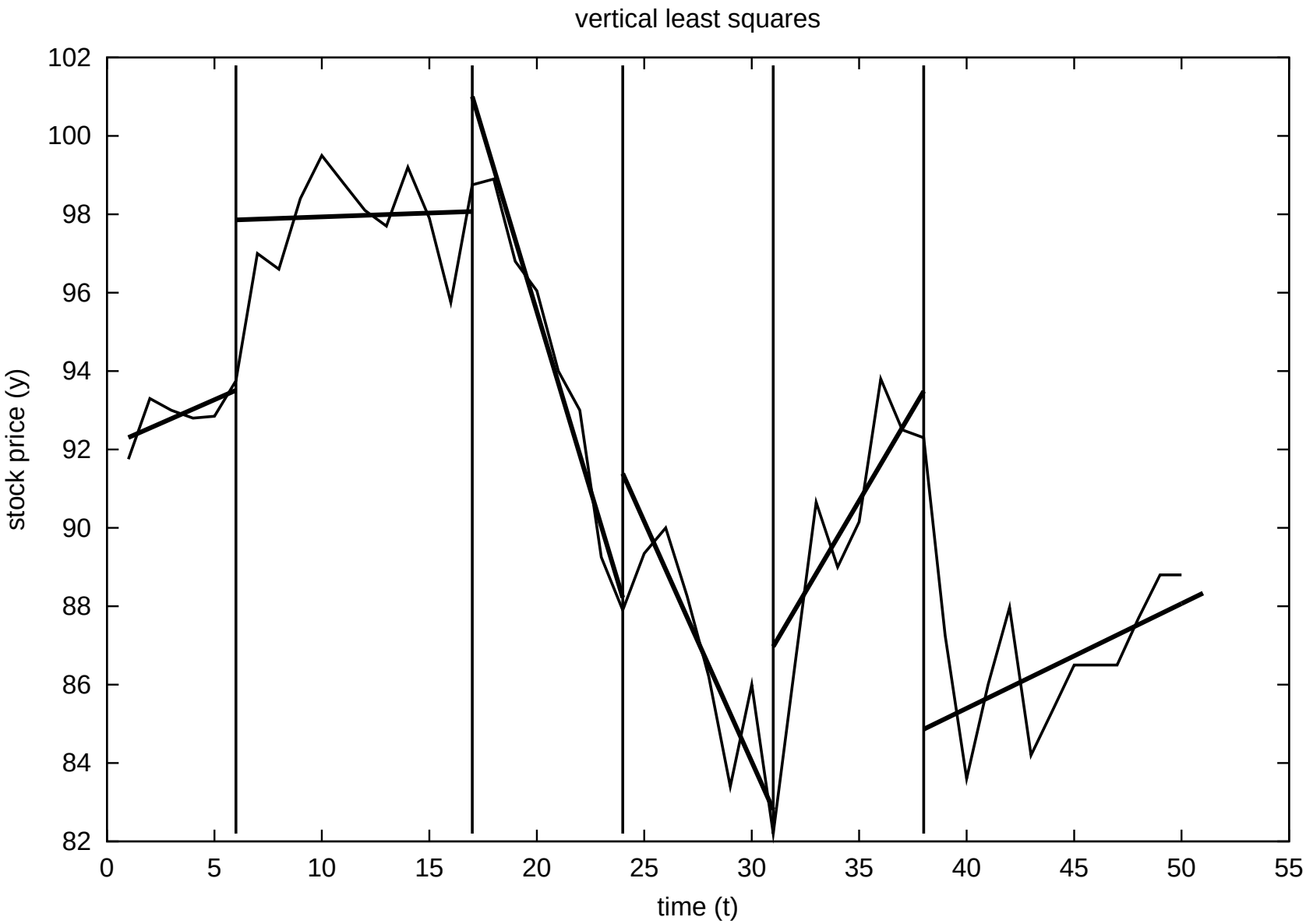
$$\begin{aligned} \det(S - \lambda I) &= 0 \\ (S_{xx} - \lambda)\,(S_{yy} - \lambda) \;-\; S_{xy}^2 &= 0 \\ \lambda^2 - \lambda\,(S_{xx} + S_{yy}) \;+\; S_{xx}S_{yy} \;-\; S_{xy}^2 &= 0 \end{aligned}$$

We need the smaller of the two eigenvalues:

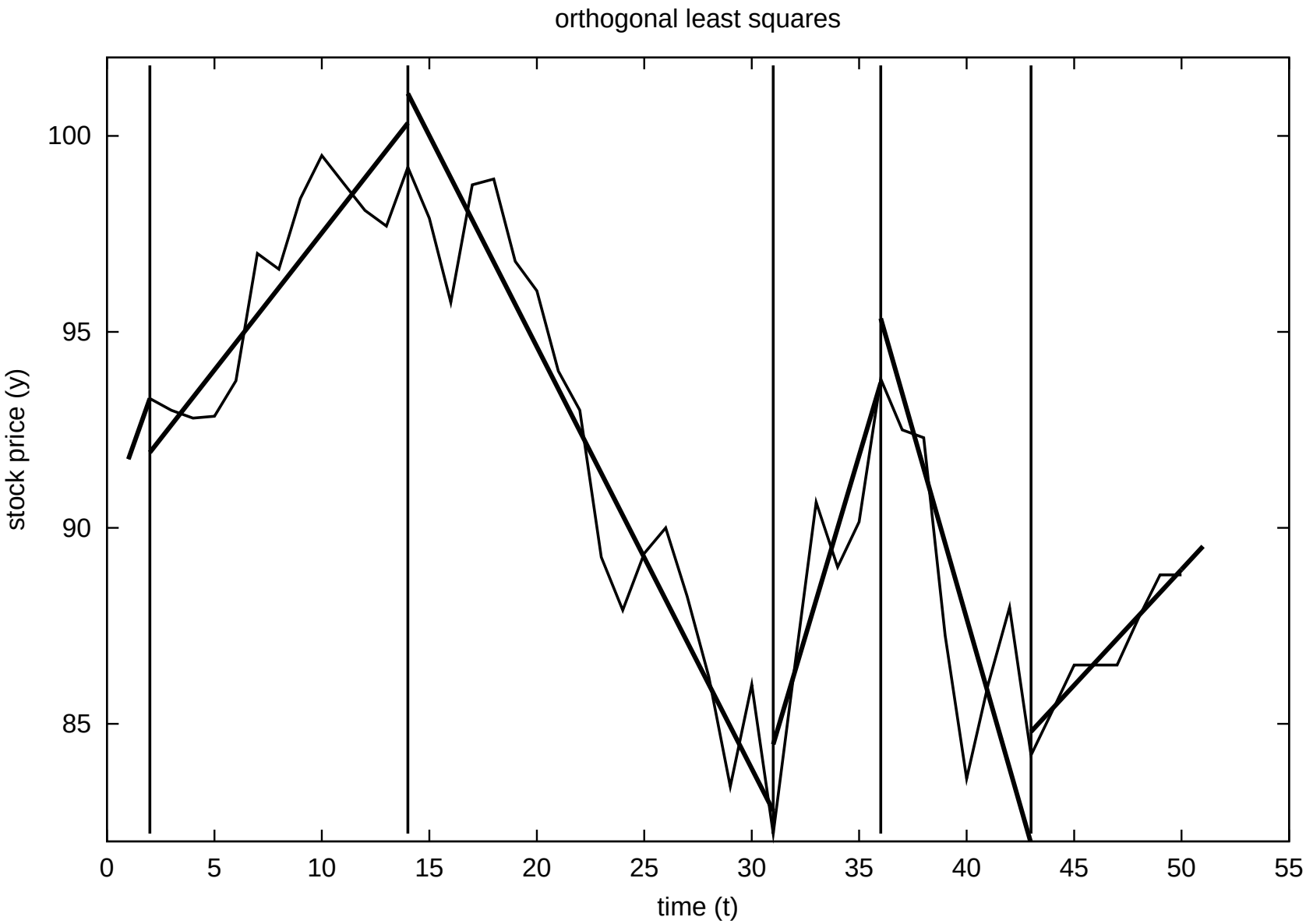
$$\lambda \quad = \quad \frac{1}{2}\left(S_{xx} + S_{yy} - \sqrt{(S_{xx} - S_{yy})^2 \;+\; 4\,S_{xy}^2}\right)$$

local error of segment: $= \lambda$

segmentation of one–dim. signals: examples – stock prices



segmentation of one–dim. signals: examples – stock prices



As before, we want to segment the sequence of points (x_1^N, y_1^N) into $k = 1, ..., K$ segments. Each segment k is represented by a circular arc with unknown center (a_k, b_k) and radius r_k:

$$(x - a_k)^2 + (y - b_k)^2 \quad = \quad r_k^2$$

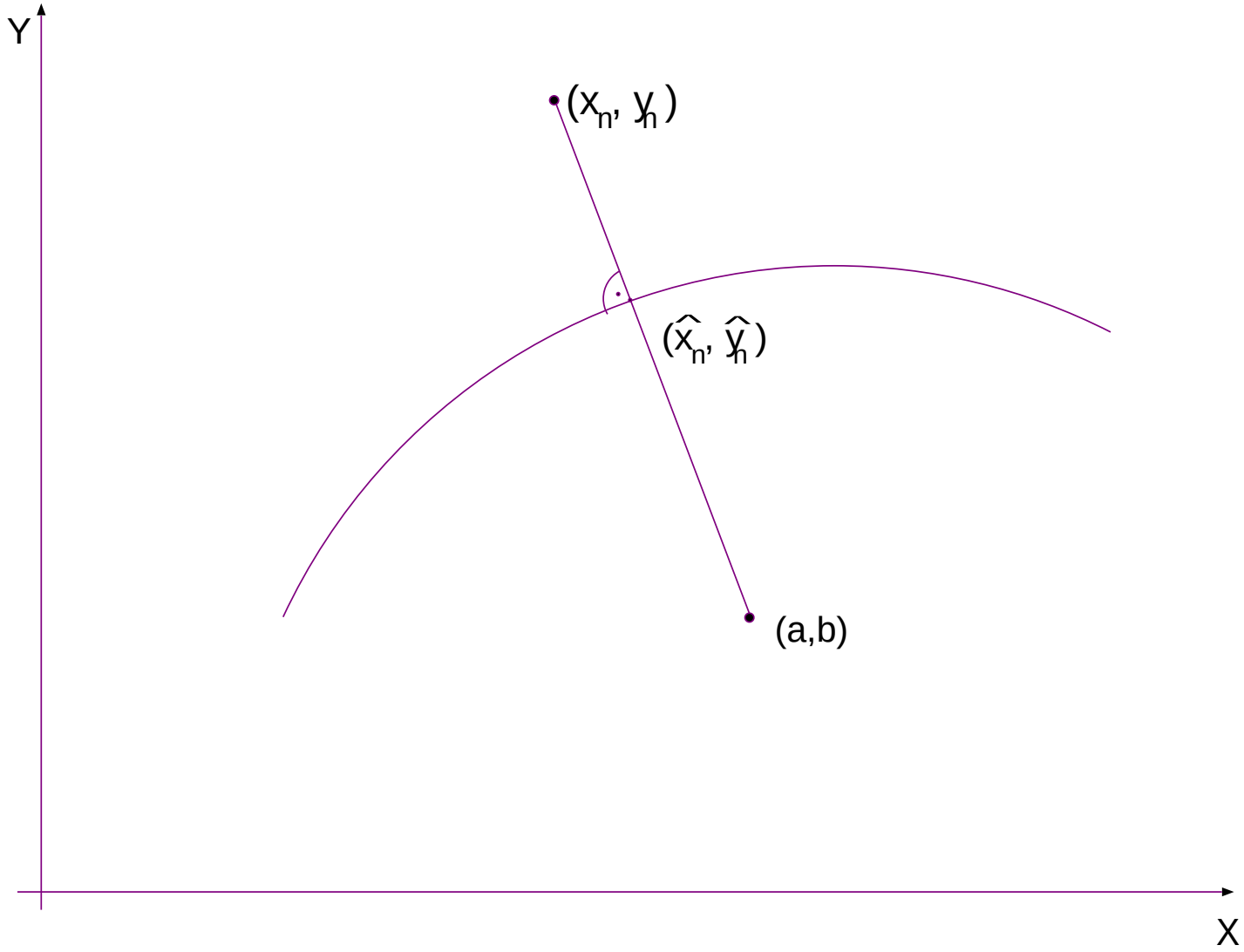
The global criterion then is:

$$\min_{n_1^K} \sum_{k=1}^{K} \min_{a_k, b_k, r_k} \left\{ \sum_{n=n_{k-1}+1}^{n_k} \left([x_n - \hat{x}_n(k)]^2 + [y_n - \hat{y}_n(k)]^2 \right) \right\}$$

For each segment k, the sum of squared errors can be re-written:

$$\sum_n [x_n - \hat{x}_n(k)]^2 + [y_n - \hat{y}_n(k)]^2 \quad = \quad \sum_n \left[r_k - \sqrt{(x_n - a_k)^2 + (y_n - b_k)^2} \right]^2$$

segmentation of image contours: circular arcs



We consider a circular arc with unknown center (a, b) and unknown radius r:

$$\begin{aligned} r_n &:= \sqrt{(x_n - a)^2 + (y_n - b)^2} \\ r_n^2 &= (r + e_n)^2 = r^2 + 2\,r\,e_n + e_n^2 \end{aligned}$$

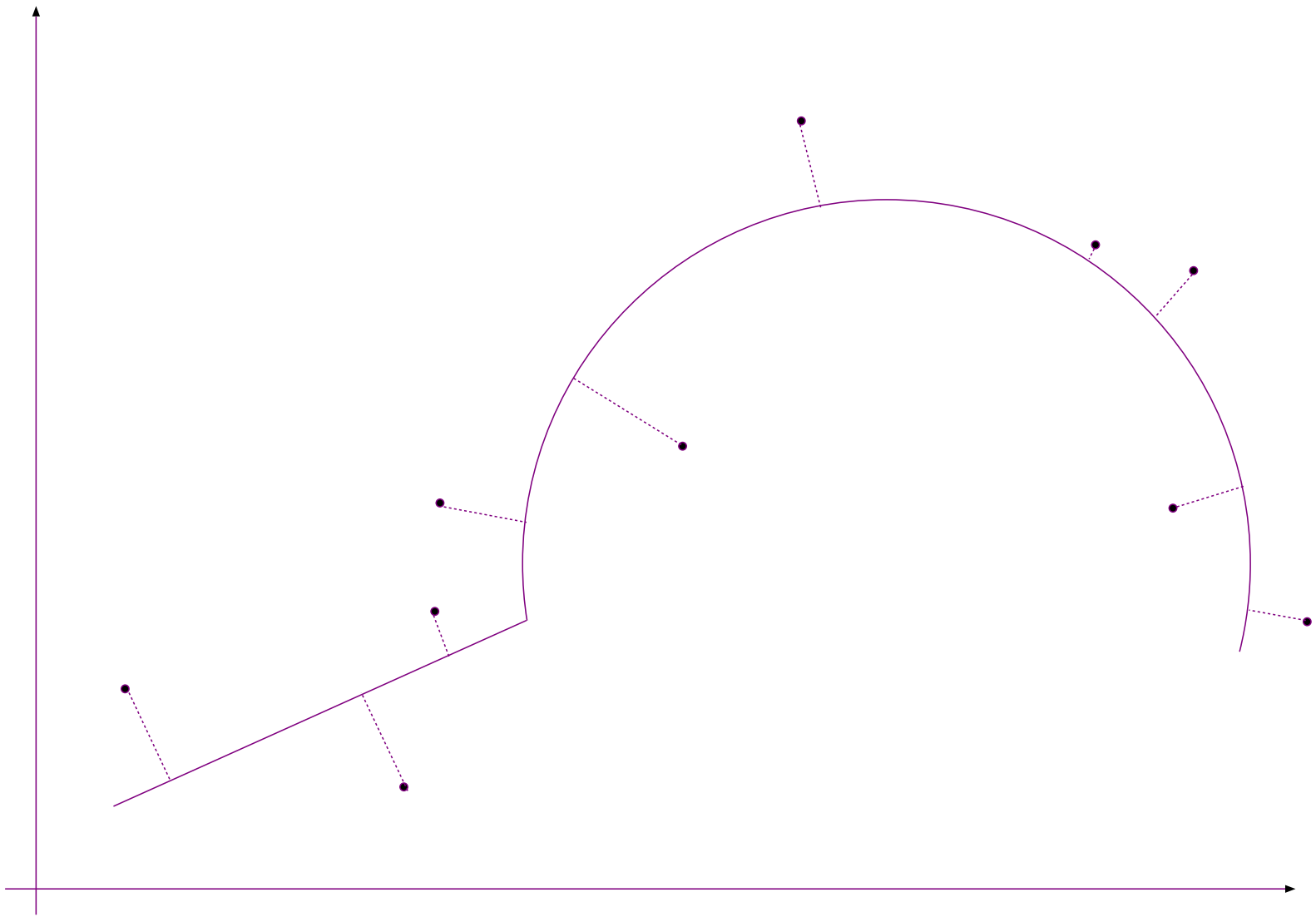
With the assumption $|e_n| \ll r$, we obtain the approximation:

$$e_n \cong \frac{r_n^2 - r^2}{2\,r}$$

For the total error of the circular arc, we obtain:

$$\begin{aligned} \sum_n e_n^2 &= \sum_n [r_n - r]^2 \\ &\cong \sum_n \left[\frac{r_n^2 - r^2}{2\,r}\right]^2 \end{aligned}$$

segmentation of image contours: circular arcs



Approximation:

to find the optimal parameters a, b, r, we neglect the denominator and consider the numerator only:

$$\min_{a,b,r} \left\{ \sum_n \left[(x_n - a)^2 + (y_n - b)^2 - r^2 \right]^2 \right\}$$

The optimal value of the squared radius is given by the mean value of the r_n^2:

$$\begin{aligned} \hat{r}^2 &= \hat{r}^2(a,b) \\ &:= \langle\ (x_n - a)^2 + (y_n - b)^2\ \rangle \\ &= < x_n^2 > -2\,a < x_n > +a^2 + < y_n^2 > -2\,b < y_n > +b^2 \end{aligned}$$

Using this value, we obtain for the numerator:

$$\min_{a,b,r}\left\{\sum_n \left[(x_n-a)^2+(y_n-b)^2-r^2\right]^2\right\} =$$

$$= \min_{a,b}\left\{\sum_n \left[(x_n-a)^2+(y_n-b)^2-\hat{r}^2(a,b)\right]^2\right\}$$

$$= \min_{a,b}\left\{\sum_n \left[x_n^2 - < x_n^2 > -2\,a(x_n - < x_n >) \;+\; y_n^2 - < y_n^2 > -2\,b(y_n - < y_n >)\right]^2\right\}$$

where the squares a^2 and b^2 have cancelled.

The resulting system of equations for a and b is LINEAR and thus can be easily solved.

note:

- use running sums (again) for efficiency
- quality of approximation ?? (better for larger r)

global criterion for general case:

$$\min_{n_1^K} \sum_{k=1}^{K} \min_{\lambda_k} \left\{ \sum_{n=n_{k-1}+1}^{n_k} \left([x_n - \hat{x}_n(k;\lambda_k)]^2 + [y_n - \hat{y}_n(k;\lambda_k)]^2 \right) \right\}$$

prior knowledge $c(K)$ (= cost function) about number of segments K:

$$\min_{K} \left\{ c(K) + \min_{n_1^K} \sum_{k=1}^{K} \min_{\lambda_k} \left\{ \sum_{n=n_{k-1}+1}^{n_k} \left([x_n - \hat{x}_n(k;\lambda_k)]^2 + [y_n - \hat{y}_n(k;\lambda_k)]^2 \right) \right\} \right\}$$

prior knowledge about smooth (single !) parameters λ_k (for sets of parameters use $||\lambda_k - \lambda_{k-1}||^2$
using constants γ_k:

$$\min_{n_1^K} \sum_{k=1}^{K} \min_{\lambda_k} \left\{ \gamma_k \cdot (\lambda_k - \lambda_{k-1})^2 + \sum_{n=n_{k-1}+1}^{n_k} \left([x_n - \hat{x}_n(k;\lambda_k)]^2 + [y_n - \hat{y}_n(k;\lambda_k)]^2 \right) \right\}$$

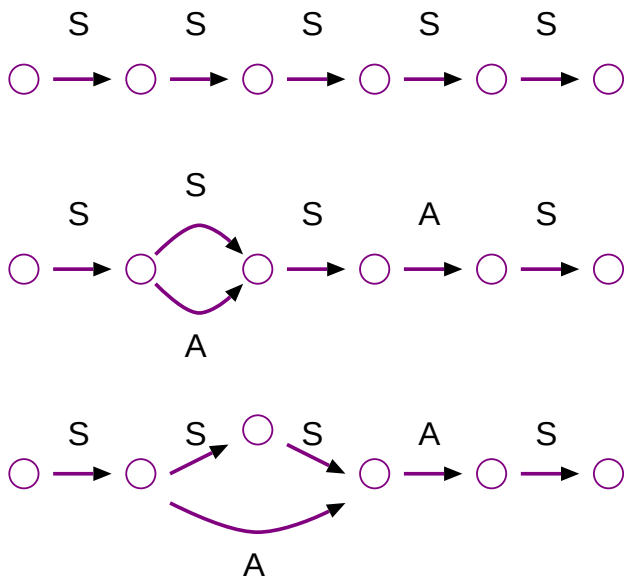
prior knowledge:

put constraints on the sequence of segment types

solution:

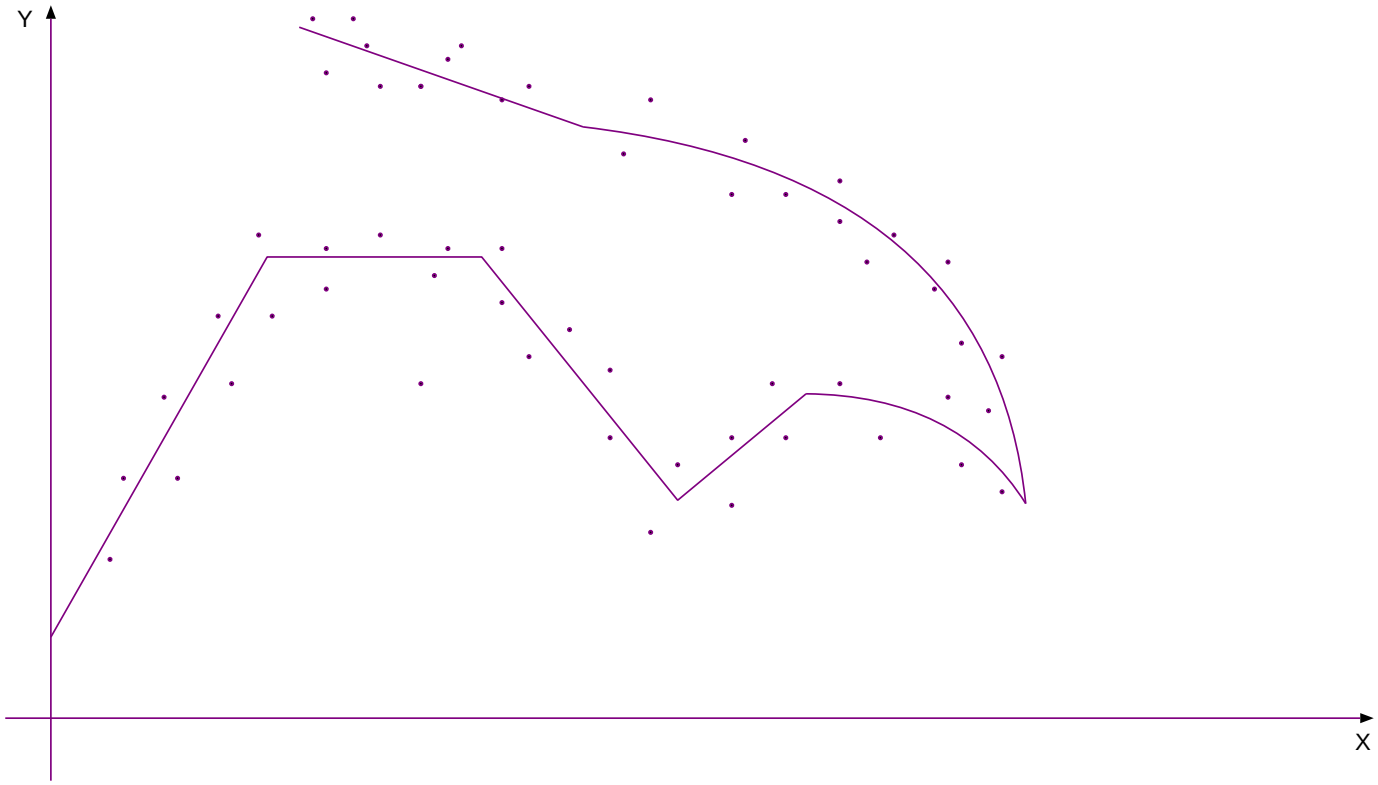
use left–to–right graph of segment types (e.g. straight lines, arcs, ...)

represented by finite–state network:

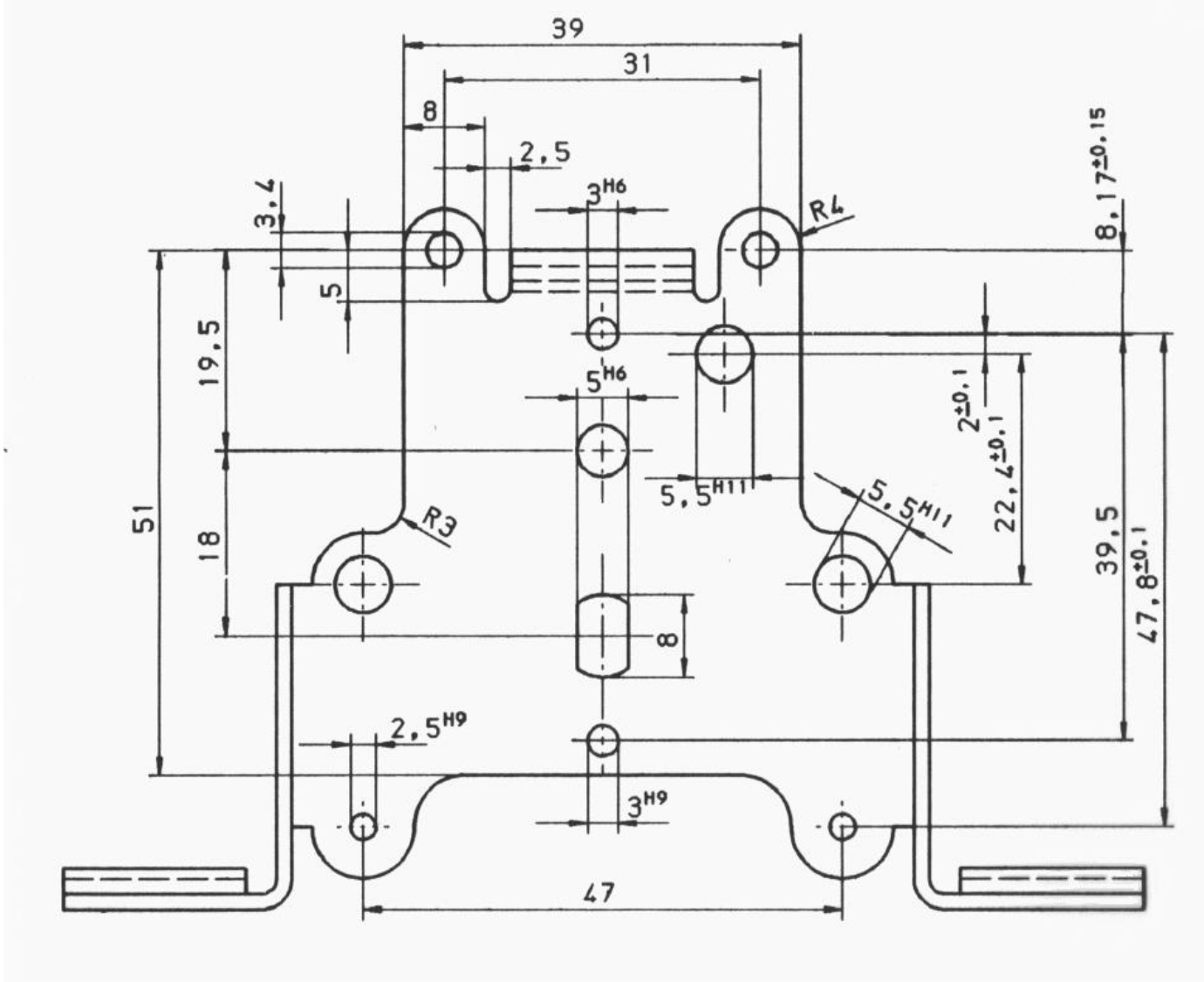


S = straight, A = (circular) arc

segmentation of image contours: prior knowledge

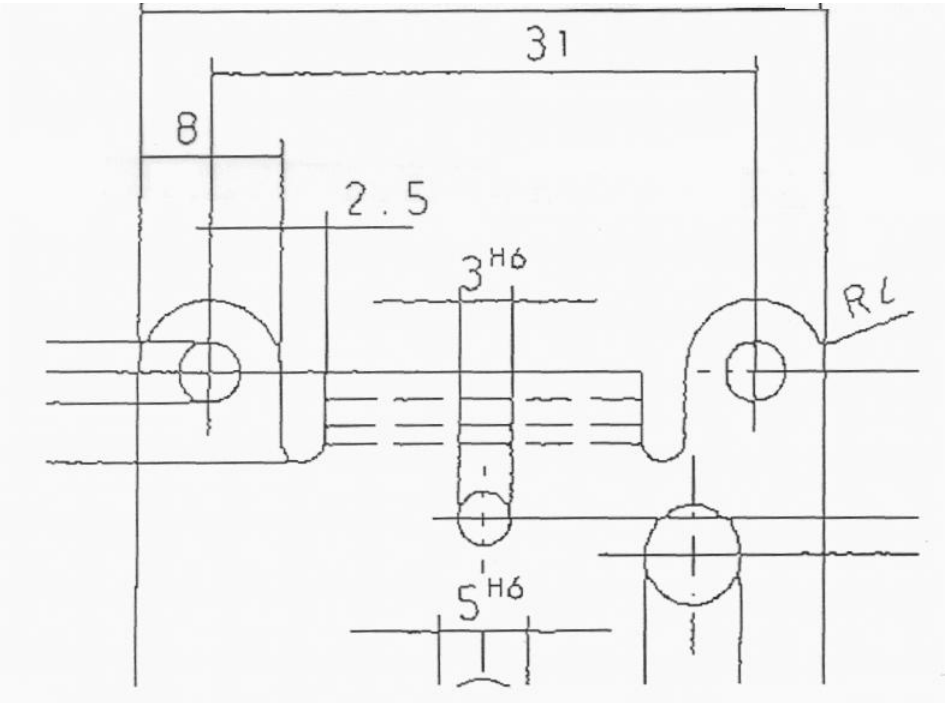


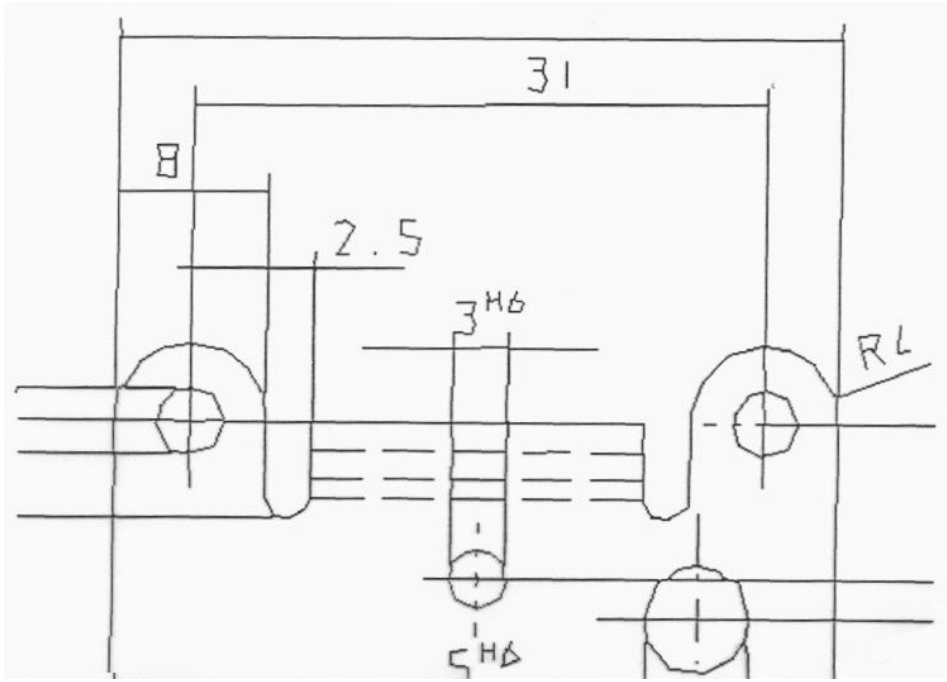
contour finding: example



example: technical line drawing

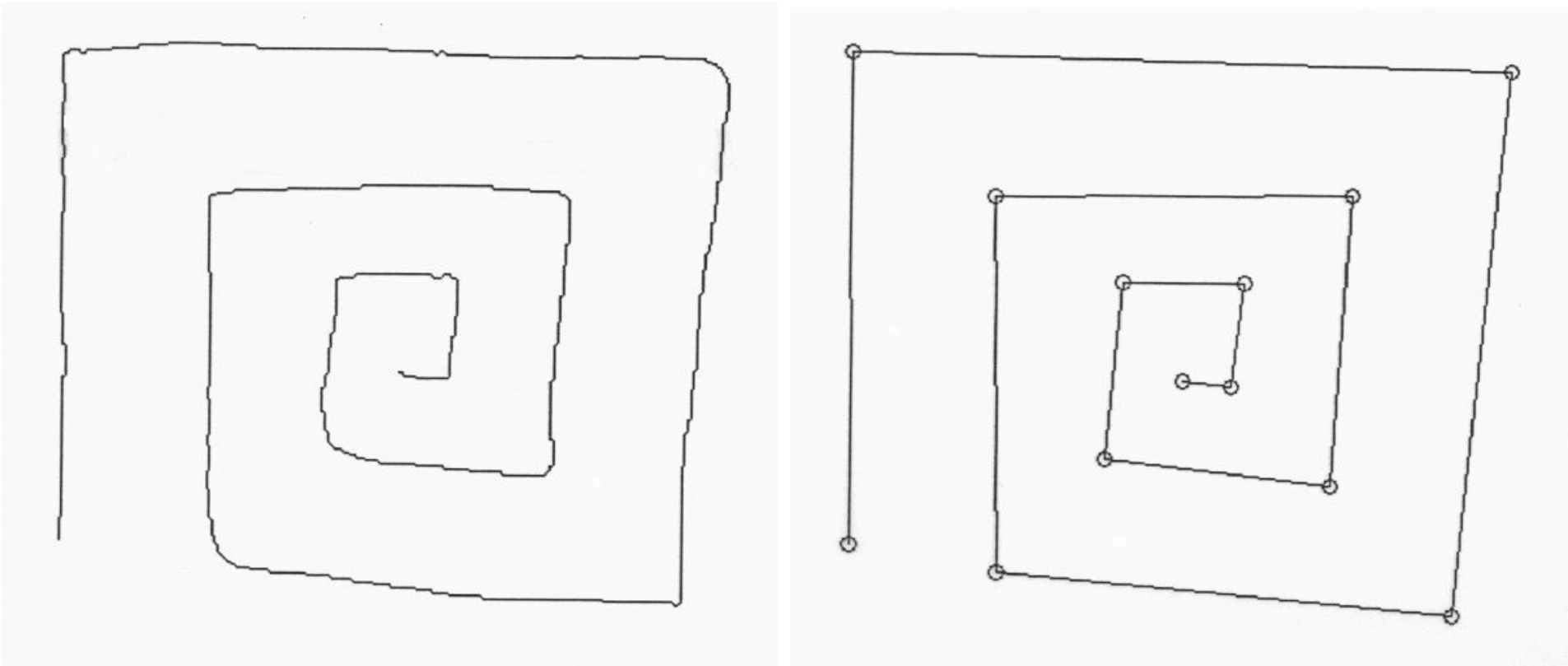
contour finding: example





application of curve fitting to a mechanical construction drawing: skeletized input image (left) and result of curve fitting (right)

contour finding: example



application of curve fitting to a freehand drawing: drawing (left) and result of curve fitting (right)

IMAGE CONTOUR

Image Contour

image greylevel as a function of coordinates (x, y):

$$(x, y) \quad \rightarrow \quad F(x, y)$$

notion of image contour:

- gradient $f(x, y)$ of image $F(x, y)$:

$$f(x, y) \quad := \quad f_{max} - \sqrt{\left(\frac{\partial F(x, y)}{\partial x}\right)^2 + \left(\frac{\partial F(x, y)}{\partial y}\right)^2}$$

  or some other suitable type of gradient
  note: suitable discretization of derivatives

- continuity constraint for the contour:

$$(x_n, y_n) \quad \in \quad R(x_{n-1}, y_{n-1})$$

  where $R(x, y)$ defines a suitable neigbourhood of (x, y),
  e.g. the 8–pixel neighbourhood
  (in the following, let $R^{-1}(x, y) := \{(x', y') : (x, y) \in R(x'y')\}$ denote the set of possible predecessors for a point (x, y);
  usually the neighborhood is symetric, i.e. $R = R^{-1}$)

Image Contour: Illustration of Neighbourhood

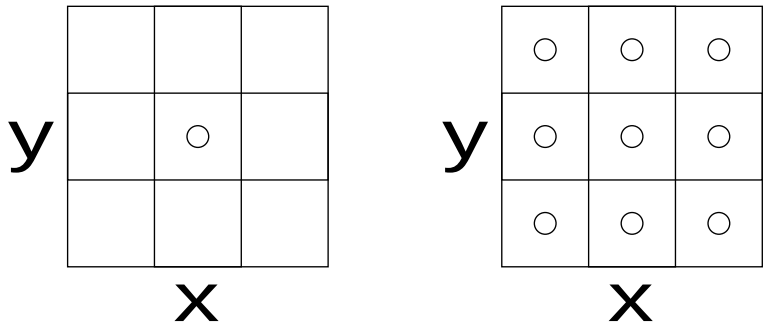


Image Contour

optimization criterion:

$$\min_{N} \left\{ c(N) + \min_{(x_1^N, y_1^N)} \left\{ \sum_{n=1}^{N} f(x_n, y_n) \right\} \right\}$$

using some suitable cost function $c(N)$ for the length N of the contour, e.g.

$$c(N) \quad = \quad \gamma \cdot (N - N_0)^2$$

Image Contour: Length Normalization

To counteract the positivity of the cost and make the optimization problem well defined, we have to specify a region R_B for the beginning point (x_1, y_1) and a region R_E for the end point (x_N, y_N):

$$(x_1, y_1) \in R_B \qquad (x_N, y_N) \in R_E$$

alternative cost function: use bonus f_0:

$$\min_N \left\{ \min_{(x_1^N, y_1^N)} \left\{ \sum_{n=1}^N [f(x_n, y_n) - f_0] \right\} \right\}$$

alternative cost function: normalize with respect to N:

$$\min_N \left\{ \min_{(x_1^N, y_1^N)} \left\{ \frac{1}{N} \sum_{n=1}^N f(x_n, y_n) \right\} \right\}$$

Image Contour

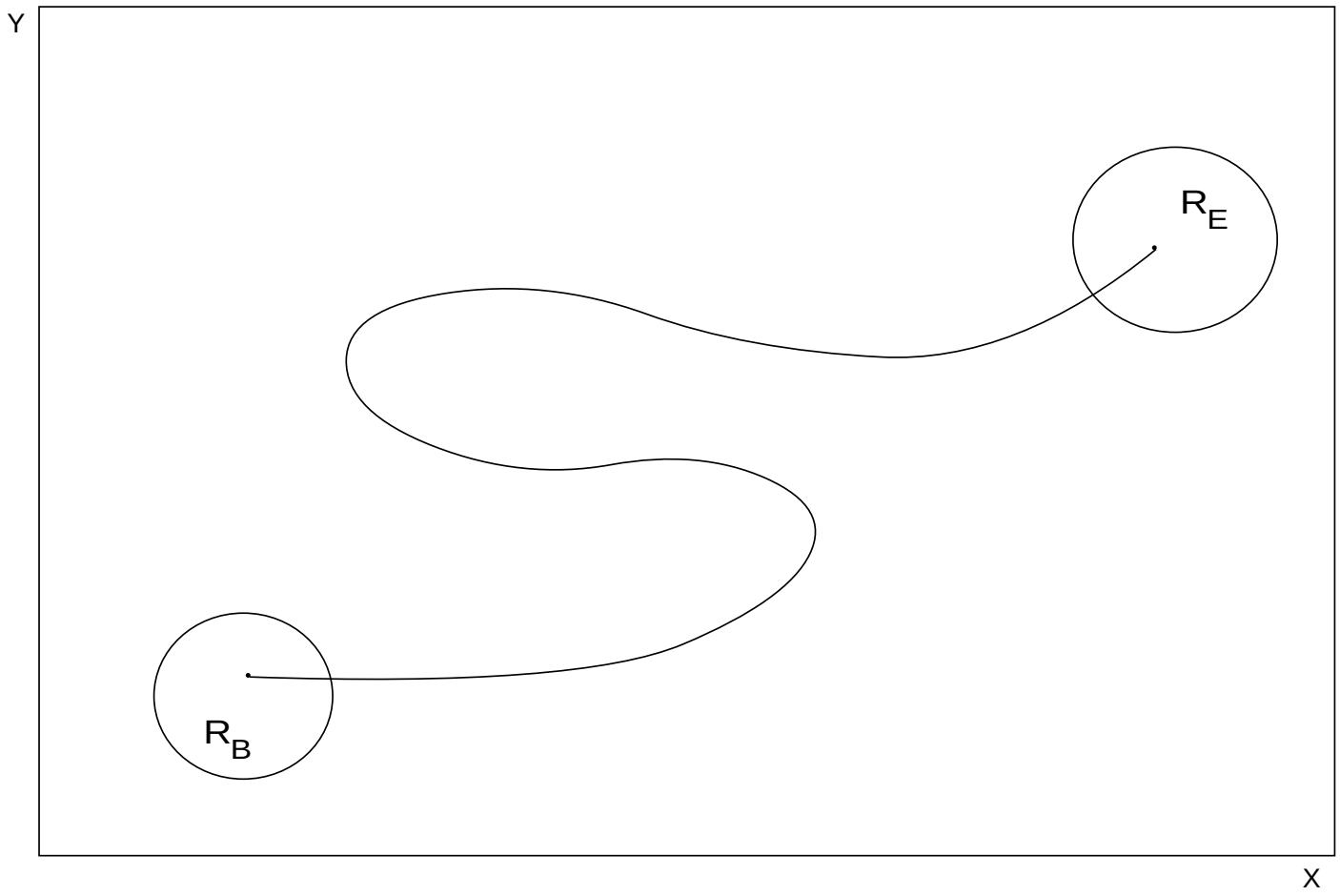


Image Contour: DP

optimization criterion:

$$\min_{N} \left\{ c(N) + \min_{(x_1^N, y_1^N)} \left\{ \sum_{n=1}^{N} f(x_n, y_n) \right\} \right\}$$

DP recursion with backpointers:

$$\begin{aligned} G_n(x,y) &= \min_{(x',y') \in R^{-1}(x,y)} \left\{ G_{n-1}(x',y') + f(x,y) \right\} \\ B_n(x,y) &= \arg \min_{(x',y') \in R^{-1}(x,y)} \left\{ G_{n-1}(x',y') + f(x,y) \right\} \end{aligned}$$

start traceback from optimal end hypothesis:

$$\min_{n} \left\{ c(n) + \min_{(x,y) \in R_E} G_n(x,y) \right\}$$

Image Contour: DP

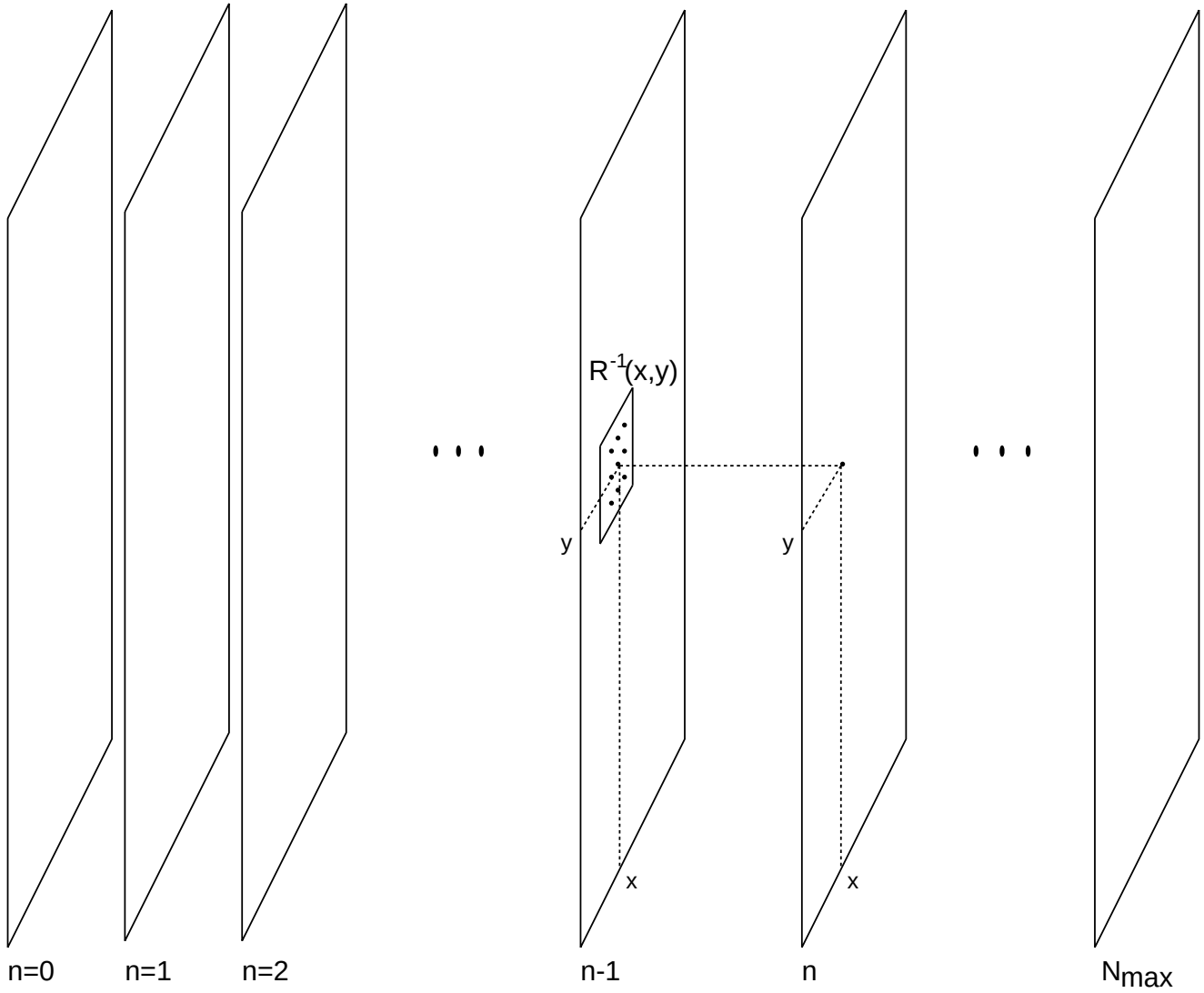


Image Contour: DP

initialization: $G_0(x,y) := 0 \quad (x,y) \in R_B$

$G_0(x,y) := \infty \quad (x,y) \notin R_B$

for each length position $n = 1, ..., N_{max}$ **do**

  for each pixel (x,y) **do**

    $$G_n(x,y) := \min_{(x',y') \in R^{-1}(x,y)} \{G_{n-1}(x',y') + f(x,y)\}$$

    $$B_n(x,y) := \arg \min_{(x',y') \in R^{-1}(x,y)} \{G_{n-1}(x',y') + f(x,y)\}$$

start traceback from optimum end hypothesis:

$$[N; x_N, y_N] := \arg \left[\min_n \min_{(x,y) \in R_E} \{c(n) + G_n(x,y)\} \right]$$

for each length position $n = N, ..., 1$ **do**

  $(x_{n-1}, y_{n-1}) := B_n(x_n, y_n)$

Image Contour: DP Beam Search

computational complexity: $N_{max} \cdot X\,Y$

reduction of computational cost: DP beam search

compute score of best hypothesis of length n:

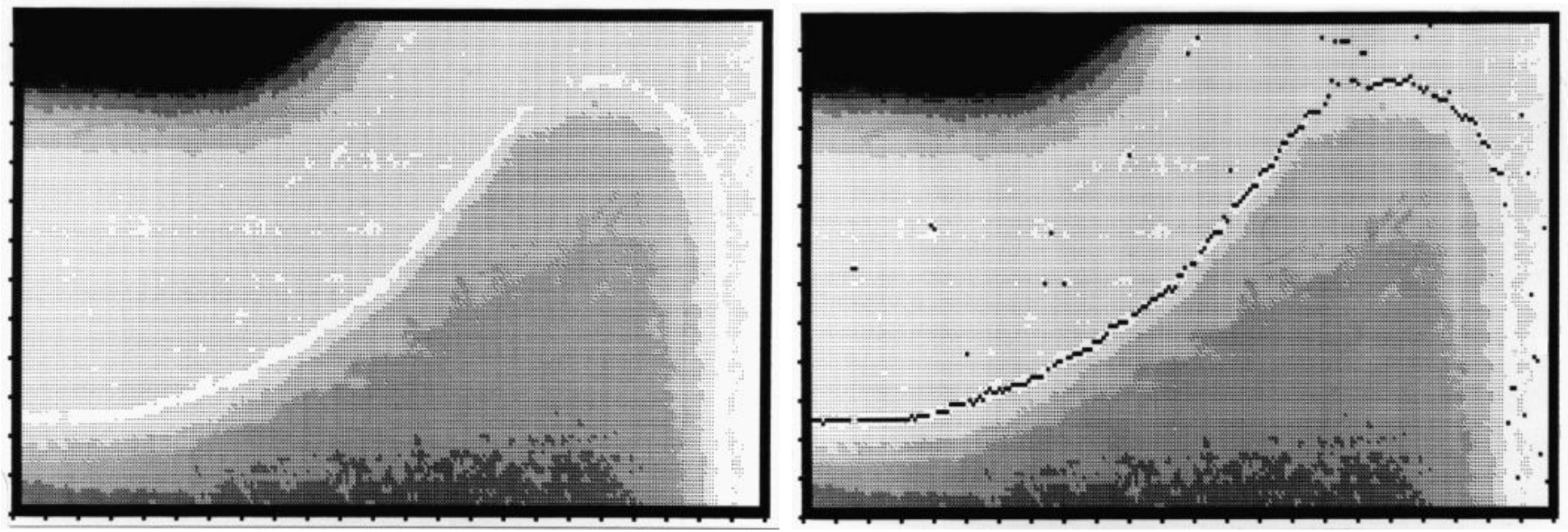
$$G_n^{min} \quad = \quad \min_{(x,y)} \{G_n(x,y)\}$$

extend only hypotheses (x,y) that are in an ϵ neighborhood:

$$\{(x,y) : G_n(x,y) \leq G_n^{min} + \epsilon\}$$

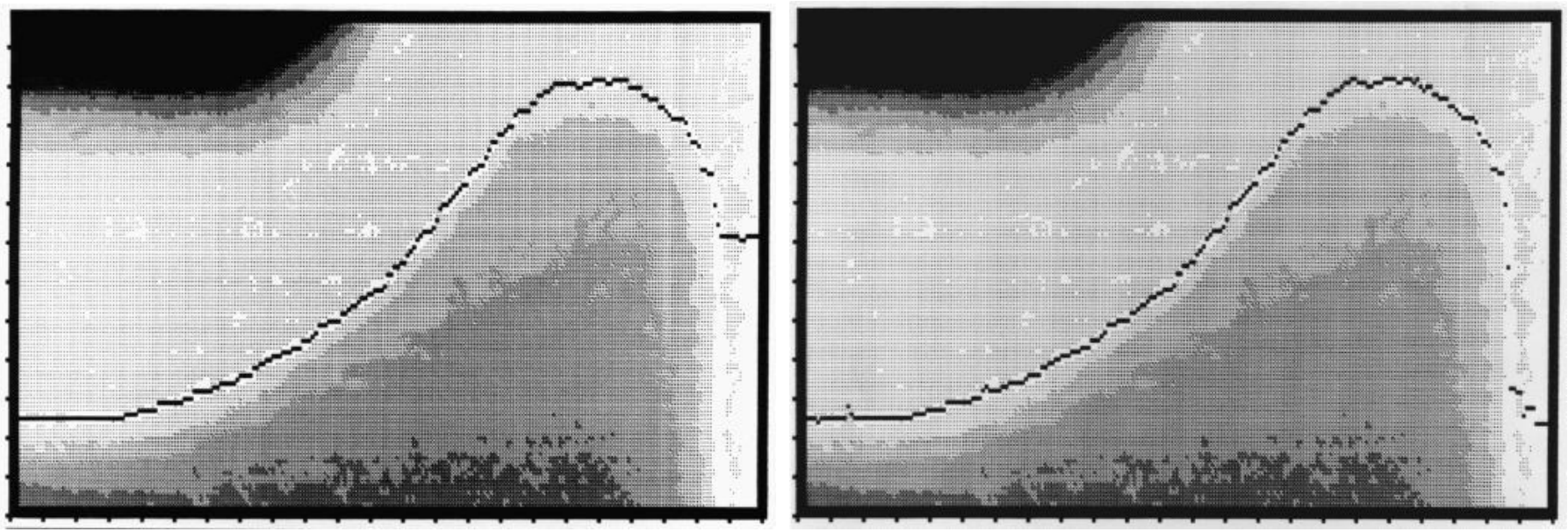
Image Contour: Example

Results for contour finding: (a) gradient image; (b) minimum for each column; (c,d) contours found (algorithm I, $w = 20/w = 5$); (e,f) contours found (algorithm II, $B = 0/B = 140$)



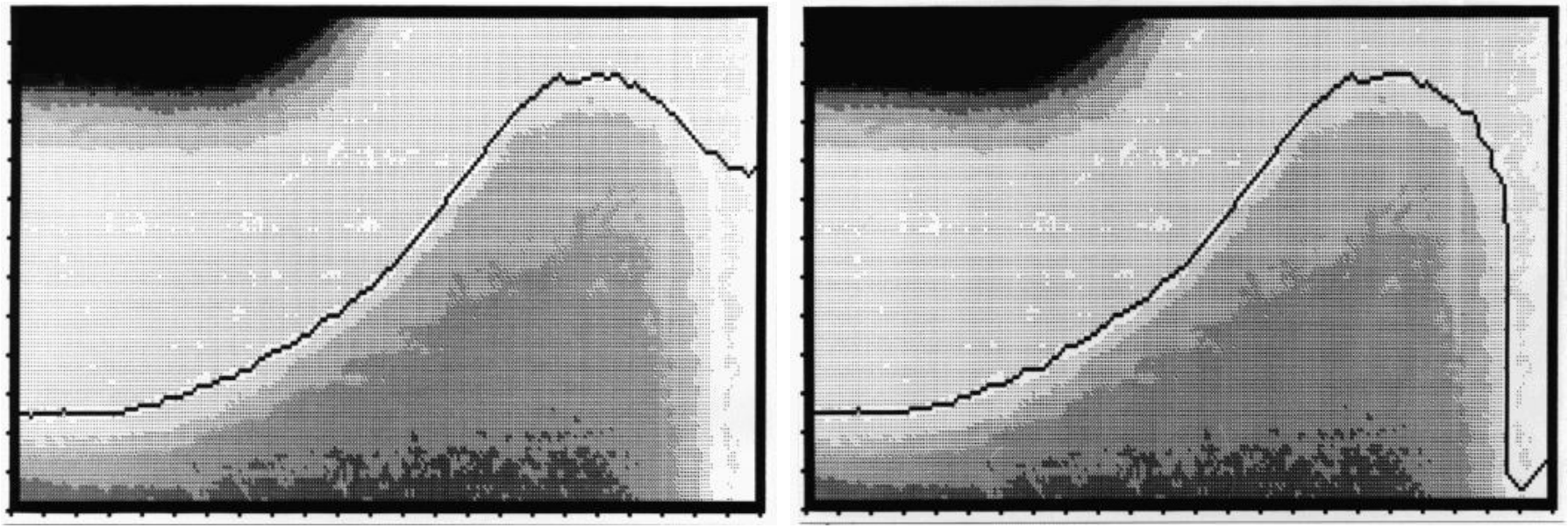
(a), (b)

Image Contour: Example



(c), (d)

Image Contour: Example



(e), (f)

model with prior knowledge:
reference contour with constraints on directions $(x_n - x_{n-1}, y_n - y_{n-1}), n = 1, ..., N$

local cost function:

$$\begin{aligned} c_n(x_n - x_{n-1}, y_n - y_{n-1}) \;&:= \\ = \quad \gamma_n \cdot \left[(x_n - x_{n-1} - \delta x_n)^2 + (y_n - y_{n-1} - \delta y_n)^2\right] \end{aligned}$$

i.e. for each reference pixel n**, we assume directions** δx_n **and** δy_n **and a weight factor** γ_n

optimization criterion using length model $c(N)$**:**

$$\min_N \left\{ c(N) + \min_{(x_1^N, y_1^N)} \sum_{n=1}^{N} \left\{ c_n(x_n - x_{n-1}, y_n - y_{n-1}) + f(x_n, y_n) \right\} \right\}$$

DP recursion:

$$G_n(x, y) \quad = \quad \min_{(x', y') \in R^{-1}(x, y)} \left\{ G_{n-1}(x', y') + c_n(x - x', y - y') + f(x, y) \right\}$$

Image Contour: Extensions

disadvantage of present model: length of contour must be specified beforehand.

improved model:
reference contour with forward, skip and loop transitions (like HMM), denoted by index $m = 1, ..., M$

DP recursion with TWO indices n **and** m**:**

$$
\begin{aligned}
G_{nm}(x,y) \;:= & \\
= \quad & \min_{\delta=0,1,2} \left\{ \min_{(x',y') \in R^{-1}(x,y)} \left\{ G_{n-1,m-\delta}(x',y') + c_m(x-x', y-y') + f(x,y) \right\} \right\} \\
= \quad & f(x,y) + \min_{(x',y') \in R^{-1}(x,y)} \left\{ c_m(x-x', y-y') + \min_{\delta=0,1,2} \left\{ G_{n-1,m-\delta}(x',y') \right\} \right\}
\end{aligned}
$$

extension: penalties $C(\delta)$

$$
\begin{aligned}
G_{nm}(x,y) \;:= & \\
= \quad & \min_{\delta=0,1,2} \left\{ C(\delta) + \min_{(x',y') \in R^{-1}(x,y)} \left\{ G_{n-1,m-\delta}(x',y') + c_m(x-x', y-y') + f(x,y) \right\} \right\}
\end{aligned}
$$

Image Contour: Extensions

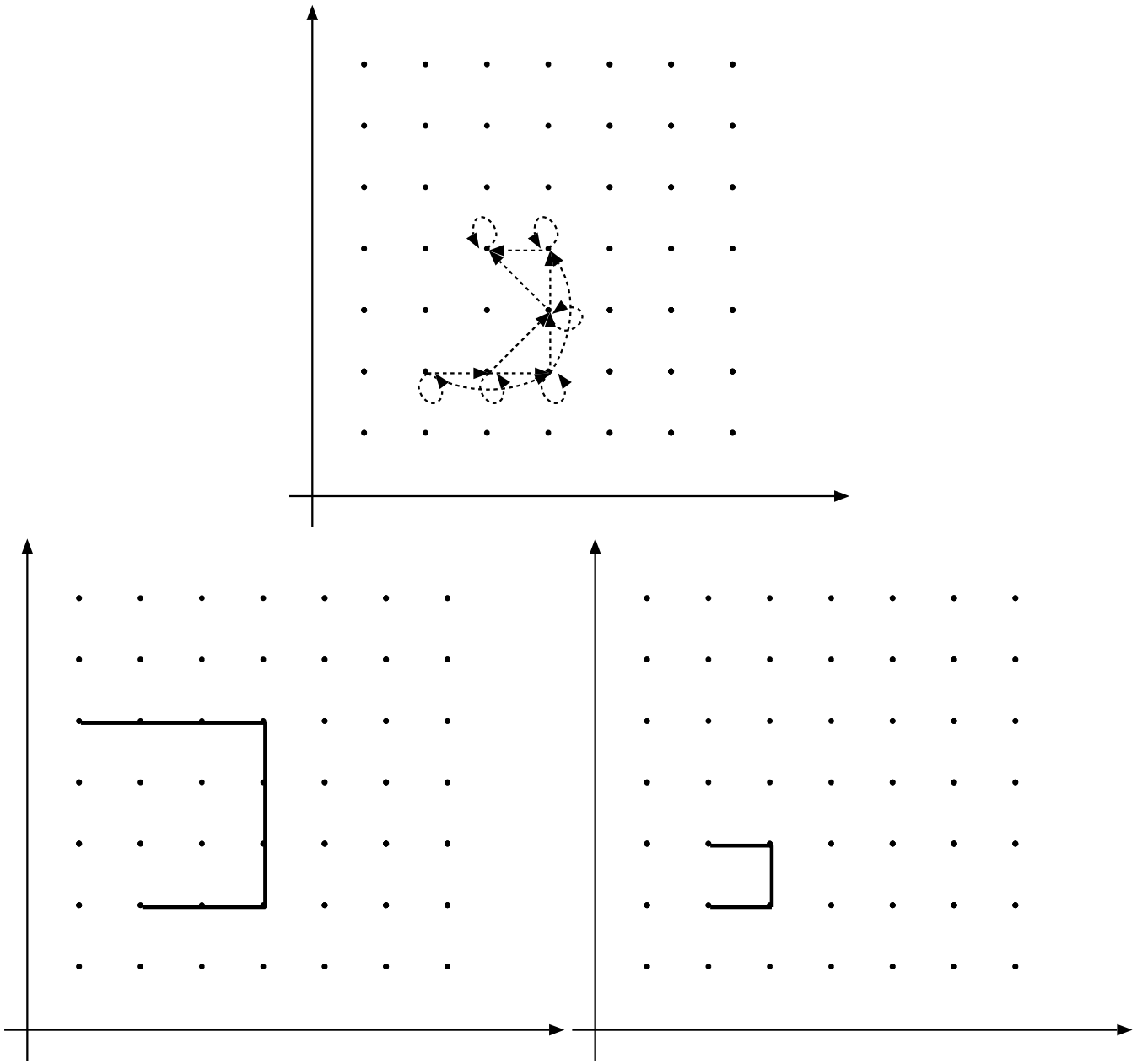
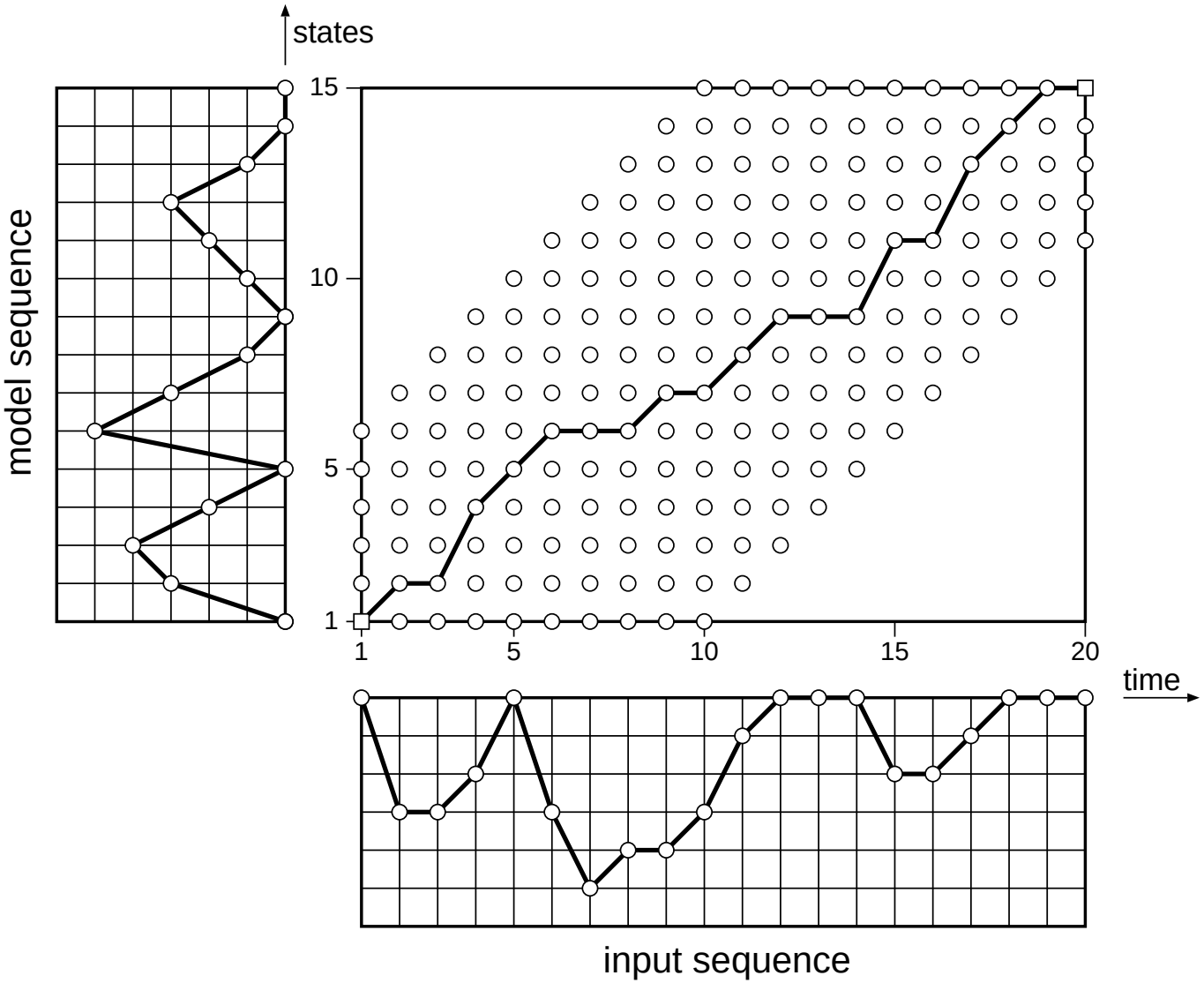


Image Registration and Matching

- **one–dimensional registration and matching:**
  = nonlinear time alignment (HMM = Hidden Markov model)

- **two–dimensional registration and matching:**
  – pseudo 2-D algorithm
  – Moore's algorithm
  – Sakoe's algorithm

Time Alignment Problem



Optimization Problem and Best Path

goal: comparing two time contours with NO fixed time axis

two time dependent signals:

- **reference signal:** $\mu_s \in \mathbb{R}, s = 1, ..., S$ **(**s **= state = IDEAL time points)**
- **observed signal:** $x_t \in \mathbb{R}, t = 1, ..., T$

task: find optimal time alignment

$$t \rightarrow s = s_t$$

quantitative criterion:

$$\min_{s_1^T} \left\{ \sum_{t=1}^{T} \left[\mathcal{T}(s_t - s_{t-1}) + (\mu_{s_t} - x_t)^2 \right] \right\}$$

with local distance:

$$(\mu_{s_t} - x_t)^2$$

and distortion penalty:

$$\mathcal{T}(s_t - s_{t-1})$$

Time Alignment: Dynamic Programming

auxiliary quantity:

$$D(\tau, s) \quad = \quad \min_{s_1^\tau : s_\tau = s} \left\{ \sum_{t=1}^{\tau} \left[\mathcal{T}(s_t - s_{t-1}) + (\mu_{s_t} - x_t)^2 \right] \right\}$$

DP recursion:

$$D(t, s) \quad = \quad \min_{s'} \left\{ D(t-1, s') + \left[\mathcal{T}(s - s') + (\mu_s - x_t)^2 \right] \right\}$$

monotonicity of time scale:

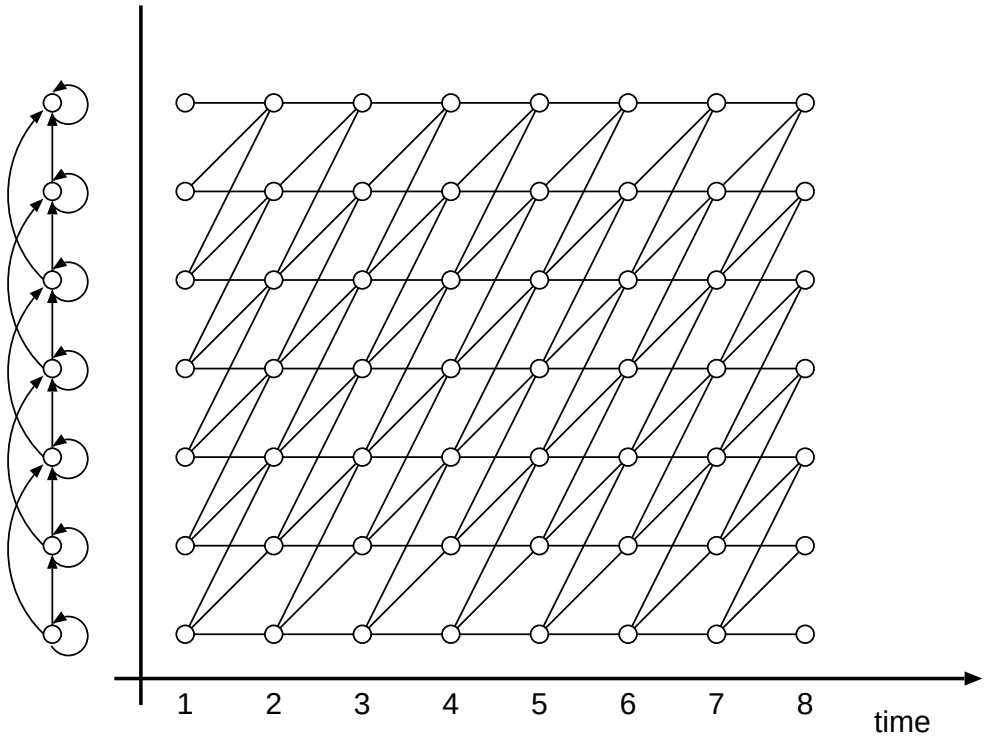
do not allow backward transitions, i.e. $\mathcal{T}(s - s') = \infty$ **for** $s < s'$

typical model with three types of transitions:

loop, forward, skip

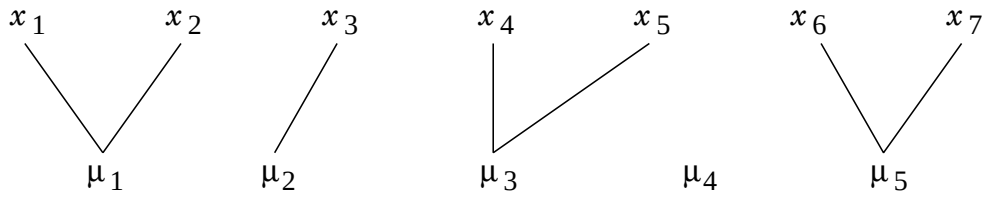
Optimization Problem and Best Path

For a reference signal μ_1^S and an observed signal x_1^T, the set of possible time alignment paths can be expressed by the so-called "trellis" or "lattice":



The time alignment problem amounts to finding the optimal path through the trellis form the bottom left to the upper right corner.

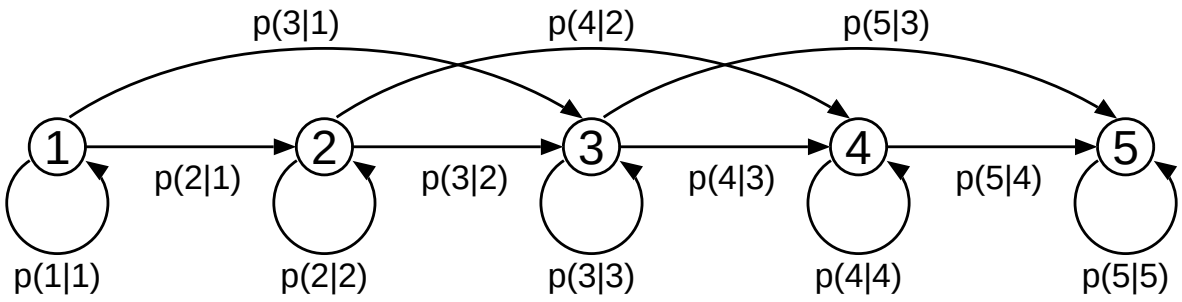
This best state sequence associates a state with each observation sample Some states may be skipped, some may have more than one acoustic vector associated with them. Such a mapping is called a "time-alignment".



Idea: Represent each "part" of a signal by a state of a (stochastic) finite state machine.

At each time-frame t the machine will "produce" an observation according to a distribution $p(x_t|s_t)$ (emission probability) which depends on its current state s_t.

Then the machine will go to the next state s_{t+1} with probability $p(s_{t+1}|s_t)$ (transition probability).



The probability of the sequence pair (x_1^T, s_1^T) is then

$$p(x_1^T, s_1^T) = \prod_{t=1}^{T} \left[p(s_t|s_{t-1}) \cdot p(x_t|s_t) \right]$$

Note: "stochastic finite state automaton", "stochastic regular grammar", and "Hidden Markov Model" are all equivalent formulations.

Hidden Markov models: Typical Choice

emission probability $p(x_t|s)$:

$$p(x_t|s) \quad = \quad \mathcal{N}(x_t|\mu_s, \sigma_s^2)$$

with a Gaussian distribution with state dependent mean μ_s and variance σ_s^2

transition probability $p(s|s')$: big table

The Viterbi Approximation

Strictly speaking it is necessary to sum over all state sequences:

$$p(x_1^T) = \sum_{\{s_1^T\}} \prod_{t=1}^{T} \left[p(s_t|s_{t-1}) \cdot p(x_t|s_t) \right]$$

In most practical applications it is sufficient to consider only the most probable state sequence (Viterbi approximation):

$$\begin{aligned}
\operatorname*{argmax}_{\{s_1^T\}} \prod_{t=1}^{T} \left[p(s_t|s_{t-1}) \cdot p(x_t|s_t) \right] &= \\
= \quad \operatorname*{argmin}_{\{s_1^T\}} \left\{ -\log \prod_{t=1}^{T} \left[p(s_t|s_{t-1}) \cdot p(x_t|s_t) \right] \right\} \\
= \quad \operatorname*{argmin}_{\{s_1^T\}} \sum_{t=1}^{T} \left[-\log p(s_t|s_{t-1}) - \log p(x_t|s_t) \right]
\end{aligned}$$

2-DIM. IMAGE MATCHING

Image matching: example

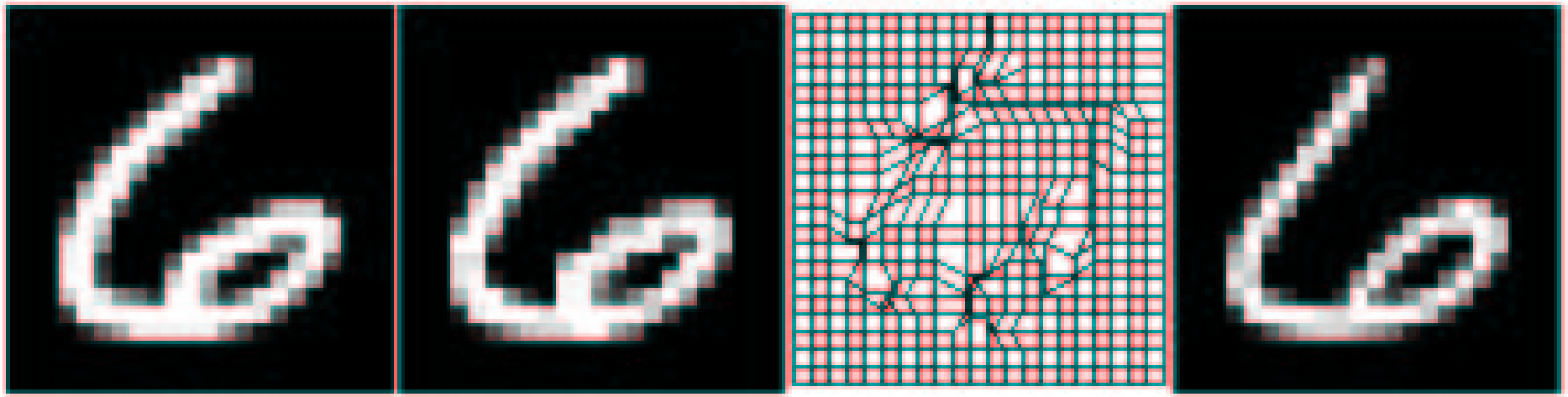


Image matching: concept

- map one image onto another one
- satisfy constraints (no 'crossing', no excessive stretching)
- we will formalize this concept in three steps and present DP algorithms for each resulting model:
  1. apply 1-D HMM to images:
     each column is regarded one fixed vector and the columns are matched
  2. add flexibility in second axis ('pseudo-2D'):
     use 1-D HMMs for each column separately (no dependency between columns)
  3. introduce dependency:
     use a real 2-D model (drawback: complexity)

Note: We do not restrict the mappings to affine or rigid transformations here. This other type of matching is usually performed by some global search over the transformation parameters with a suitable criterion (e.g. least squares or information theoretic measures).

From 1-D to 2-D

Now: comparing two images with image planes flexible, allow deformation

Two position dependent signals = images:

- reference image: $\mu_{xy} \in \mathbb{R},\ x = 1, ..., X,\ y = 1, ..., Y$
- observed image: $a_{ij} \in \mathbb{R},\ i = 1, ..., I,\ j = 1, ..., J$

Task: find optimal image alignment

$$(i, j) \rightarrow (x, y) = (x_{ij}, y_{ij})$$

1-D signal natural to regard as sequence $t \mapsto t + 1$ over $t = 1, ..., T - 1$

This is not the case for 2-D signals.

From 1-D to 2-D

First step: consider only ONE axis as flexible, second axis fixed
$\rightarrow$ problem is the same as 1-D time alignment with vector-valued signals:

$$(i,j) \rightarrow (x,y) = (x_i, j)$$

Quantitative criterion:

$$\min_{x_1^I} \left\{ \sum_{i=1}^{I} \left[\mathcal{T}(x_i - x_{i-1}) + \sum_{j=1}^{J} (\mu_{x_i j} - a_{ij})^2 \right] \right\}$$

(assumption here: $Y = J$, i.e. images of same height)

compare HMM from before:

$$\min_{s_1^T} \left\{ \sum_{t=1}^{T} \left[\mathcal{T}(s_t - s_{t-1}) + (\mu_{s_t} - x_t)^2 \right] \right\}$$

Pseudo-2-D HMM

Now introduce flexibility in second axis:

Consider each column vector of the image as 1-D signal and use best alignment.

$$(i,j) \rightarrow (x,y) = (x_i, y_{ij})$$

Quantitative criterion:

$$\min_{x_1^I} \left\{ \sum_{i=1}^{I} \left[\mathcal{T}(x_i - x_{i-1}) + \min_{y_{i1}^J} \left\{ \sum_{j=1}^{J} [\mathcal{T}(y_{ij} - y_{i,j-1}) + (\mu_{x_i y_{ij}} - a_{ij})^2] \right\} \right] \right\}$$

No interdependence between the HMMs for the columns is assumed,
each image column is considered independently.

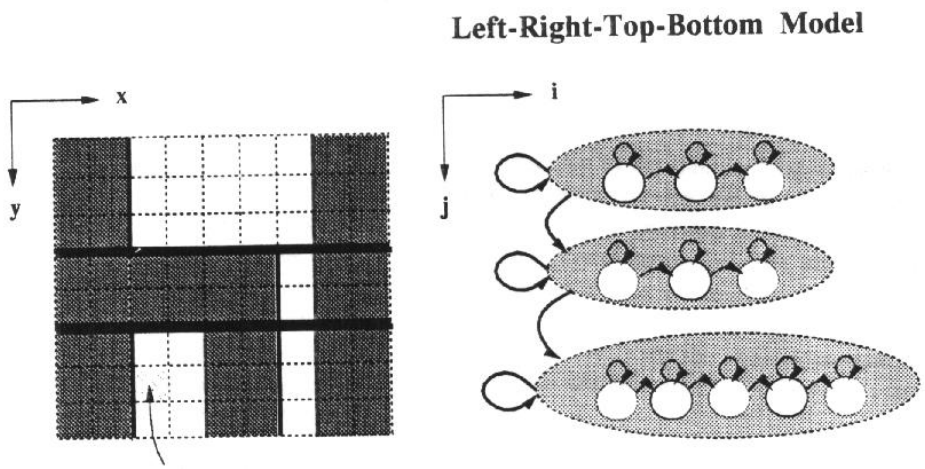
$\rightarrow$ called <u>pseudo</u>-2-D HMM

Pseudo-2-D HMM

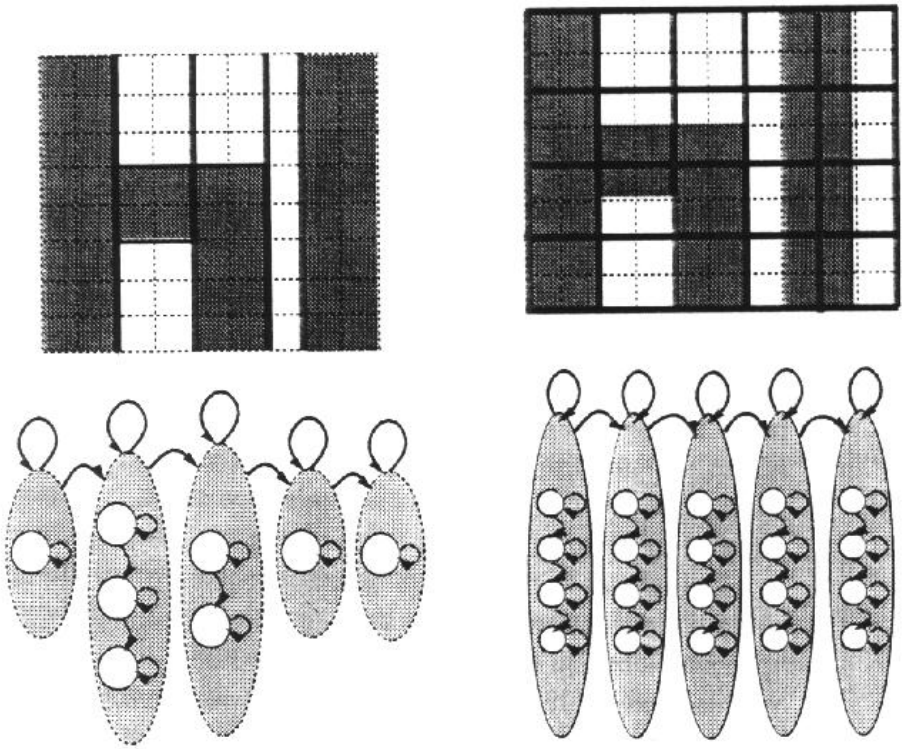
DP Recursion:

$$
\begin{aligned}
D(i,x) &= \min_{x'} \left\{ D(i-1,x') + \left[\mathcal{T}(x-x') + D_{(i,x)}(J,Y) \right] \right\} \\
D_{(i,x)}(j,y) &= \min_{y'} \left\{ D_{(i,x)}(j-1,y') + \left[\mathcal{T}(y-y') + (\mu_{xy} - a_{ij})^2 \right] \right\}
\end{aligned}
$$

Pseudo-2-D HMM



Pseudo-2-D HMM for the word "hl"



"rotated" structures

→ independent DP for columns and rows

Pseudo-2-D HMM

application: keyword spotting

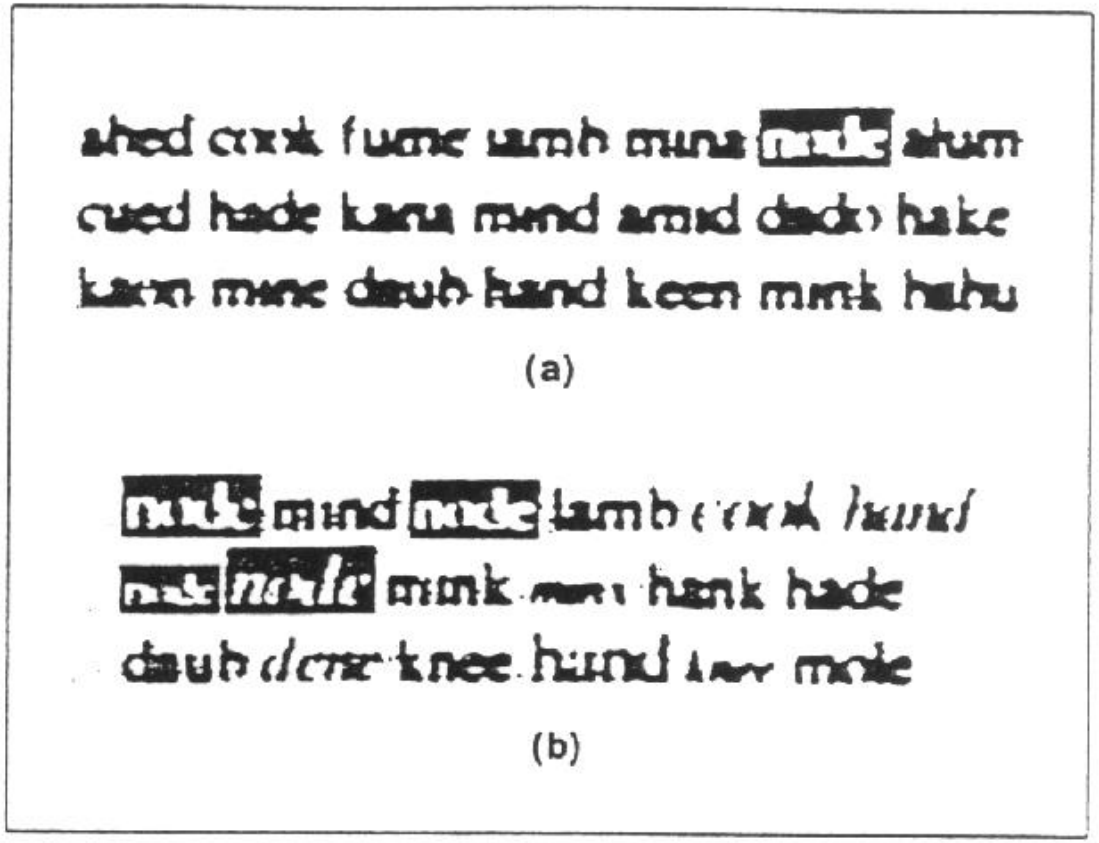


Figure 7. In (a), the keyword "node" is correctly spotted on this document, which contained, in one test, 2,650 key words and 16,000 extraneous words. In (b), the keyword "node" also was successfully spotted in a document that included both size and slant transformations of the word.

Moore's algorithm

Pseudo-2-D maybe not suited, consider generalization of edit distance

1-D edit distance:

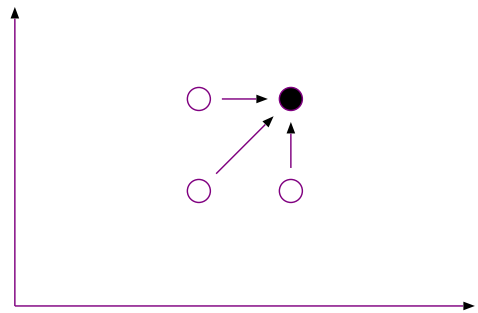
the minimum cost of changing one sequence into the other by the substitution, deletion, and insertion of symbols in either sequence

recursive definition of $D(i,j)$ (cost of matching the sequences $a_1 a_2 \ldots a_i$ and $b_1 b_2 \ldots b_j$)

$$\begin{aligned} D(i,j) \quad = \quad & \min\{D(i-1,j) + d(a_i,\epsilon), \\ & D(i,j-1) + d(\epsilon,b_j), \\ & D(i-1,j-1) + d(a_i,b_j)\} \end{aligned}$$

for each matching step, there are 3 possibilities:

- skip the last symbol of a or,
- skip the last symbol of b or,
- match the last symbol of a with the last symbol of b.



here: replace symbols by greyvalues $\rightarrow$ use $d(a,b) = (a-b)^2$

Moore's algorithm

Straightforward generalization is difficult, because skipping of an element from a 1-D signal leaves a 1-D signal in a canonical way, but skipping of a pixel from a 2-D images does not leave an image.

extension to 2-D: 15 possibilities (in general: $2^{2D} - 1$ – hypercube in 2D-dimensional space has 2^{2D} vertices, one vertex can result from all others; in 1-D: square has 4 vertices)
$\rightarrow$ means skipping of rows/columns

formula very complicated, informally: given two images A and B we have the possibilities to

1. skip the right column of A
2. skip the right column of B
3. skip the lower row of A
4. skip the lower row of B
5. skip the right column of A and the lower row of B
6. skip the lower row of A and the right column of B
7. match the right columns of A and B
8. match the lower rows of A and B
9. match the right columns of A and B, skip the lower row of A
10. match the right columns of A and B, skip the lower row of B
11. match the lower rows of A and B, skip the right column of A
12. match the lower rows of A and B, skip the right column of B
13. skip the lower row and the right column of A
14. skip the lower row and the right column of B
15. match the lower rows and the right columns of A and B

matching of 1-D sub-patterns with 1-D edit distance

Moore's algorithm

Easy to implement

Difficult to interpret:

- **which deformations are allowed at which cost?**

No interdependence between matched or skipped columns / rows

Symmetric, allows skipping of pixels (and complete columns and rows) in both images

planar HMM

If pseudo-2-D not suited maybe "real" 2-D HMM (or planar HMM)

Interdependence between alignment in both directions

Task: find optimal image alignment

$$(i,j) \rightarrow (x,y) = (x_{ij}, y_{ij})$$

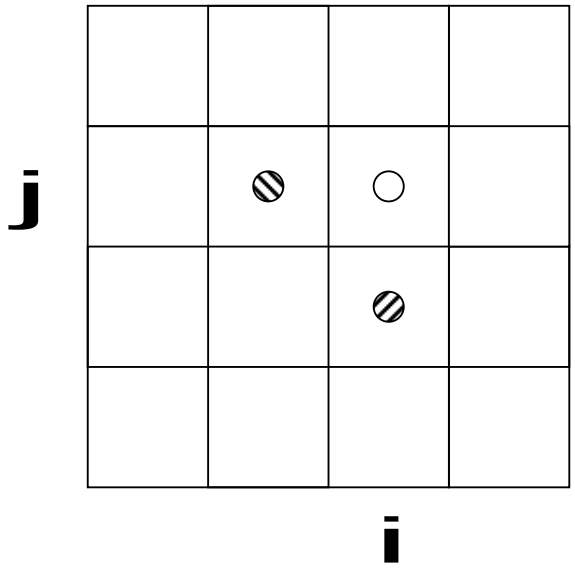
Quantitative criterion:

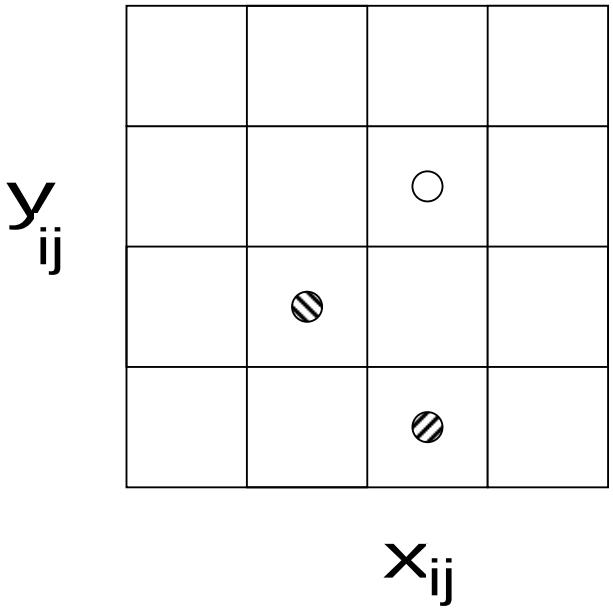
$$\min_{x_{11}^{IJ}, y_{11}^{IJ}} \left\{ \sum_{i=1}^{I} \sum_{j=1}^{J} \left[\mathcal{T}_1(x_{ij} - x_{i-1,j}) + \mathcal{T}_2(x_{ij} - x_{i,j-1}) + \right. \right.$$
$$\left. \left. \mathcal{T}_2(y_{ij} - y_{i-1,j}) + \mathcal{T}_1(y_{ij} - y_{i,j-1}) + (\mu_{x_{ij},y_{ij}} - a_{ij})^2 \right] \right\}$$

DP algorithm is exponential in $\min(I,J)$

Can be shown to be NP-complete under some general assumptions

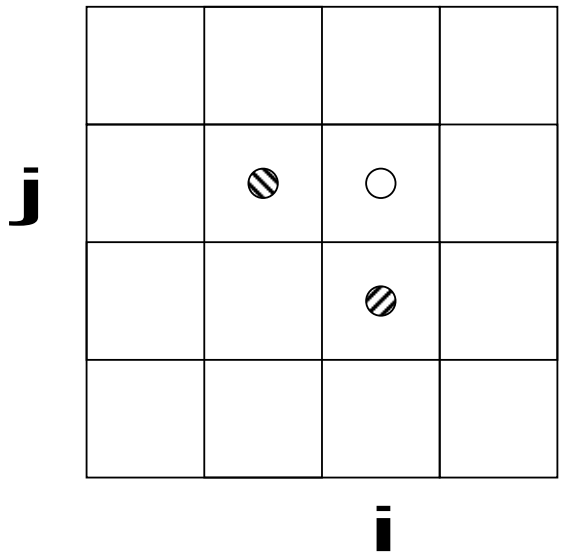
2-D Dependencies

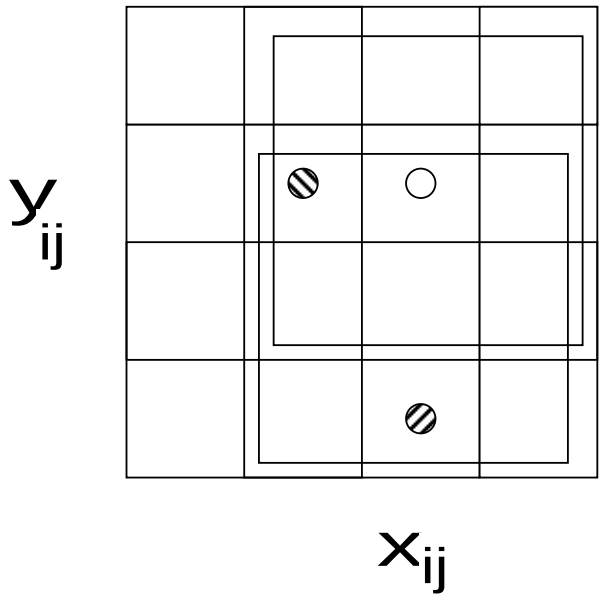




Sakoe's algorithm

$$\mathcal{T}_1(\delta) = \begin{cases} |\delta - 1| & \text{for } \delta \in \{0, 1, 2\} \\ \infty & \text{otherwise} \end{cases} \qquad \mathcal{T}_2(\delta) = \begin{cases} |\delta| & \text{for } \delta \in \{-1, 0, 1\} \\ \infty & \text{otherwise} \end{cases}$$





constraints on displacements necessary for "monotonicity and continuity"

boundary conditions: $x_{1j} = y_{i1} = 1 \qquad x_{Ij} = X \qquad y_{iJ} = Y$

2-D-interdependence $\rightarrow$ exponential complexity even with DP

DP algorithm:

- xy_{ij}^J denotes last J pixel alignments at position (i,j) (called "warped wake"), needed because of 2-D-interdependence:

$$
\begin{aligned}
xy_{ij}^J : \quad (i-1, j+1) &\mapsto (x_{i-1,j+1}, y_{i-1,j+1}) \\
&\ldots \\
(i-1, J) &\mapsto (x_{i-1,J}, y_{i-1,J}) \\
(i, 1) &\mapsto (x_{i,1}, y_{i,1}) \\
&\ldots \\
(i, j) &\mapsto (x_{i,j}, y_{i,j})
\end{aligned}
$$

- $WW(i,j)$ denotes set of all warped wakes xy_{ij}^J in position (i,j):

$$
\begin{aligned}
WW(i', j') &= \{xy_{i'j'}^J\} \\
&= \Big\{\{x_{ij}, y_{ij}\} : (i = i' \quad \wedge \; j \in \{1, \ldots, j'\}) \vee \\
&\qquad\qquad (i = i' - 1 \; \wedge \; j \in \{j'+1, \ldots, J\})\Big\}
\end{aligned}
$$

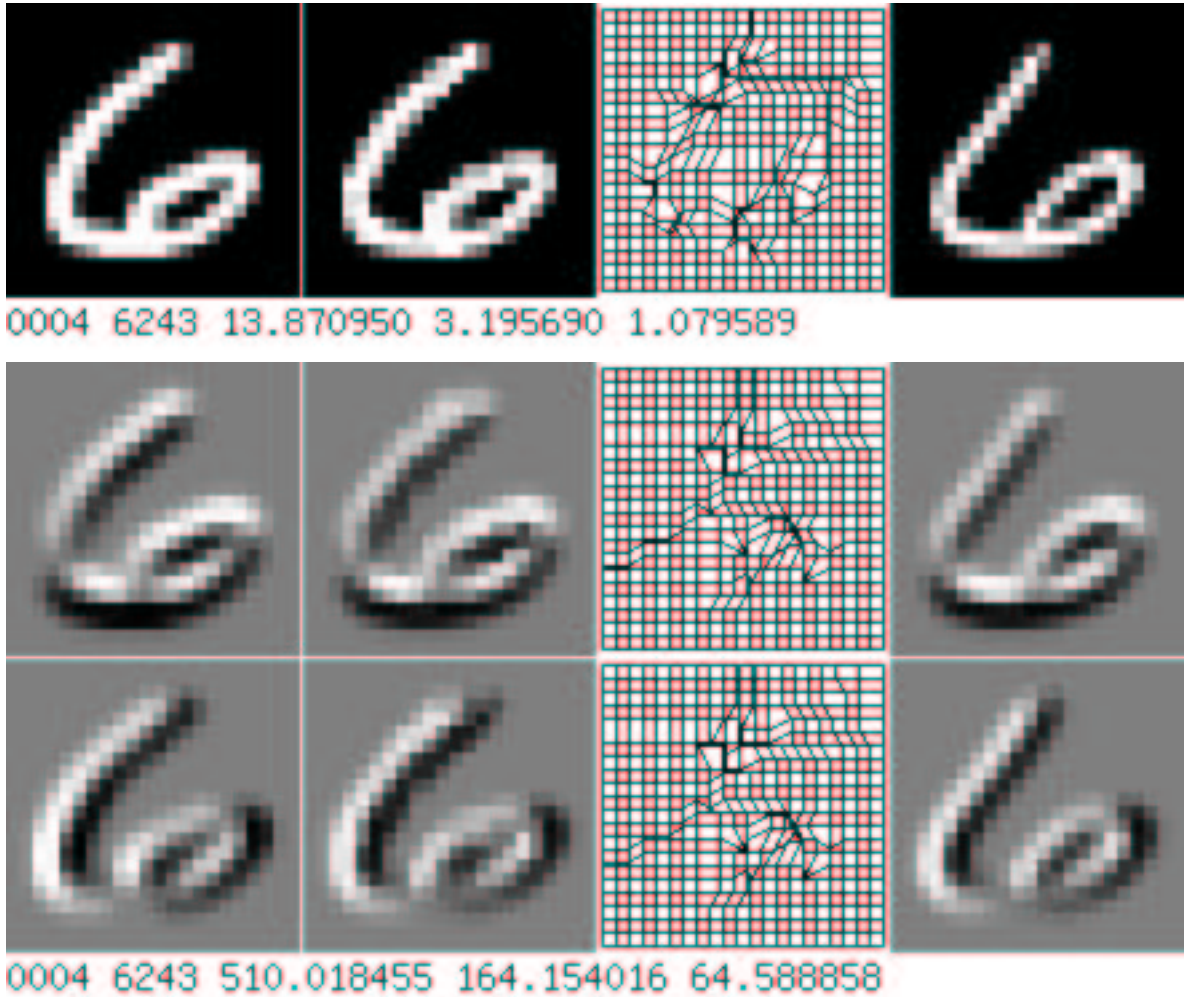
Sakoe's algorithm

DP algorithm (cont'd):

- $Pred(xy_{ij}^J)$ denotes set of possible predecessors of xy_{ij}^J
- auxiliary quantity $D(i,j,xy_{ij}^J)$ is minimal cost for alignment up to (i,j) with the last J pixels aligned according to xy_{ij}^J

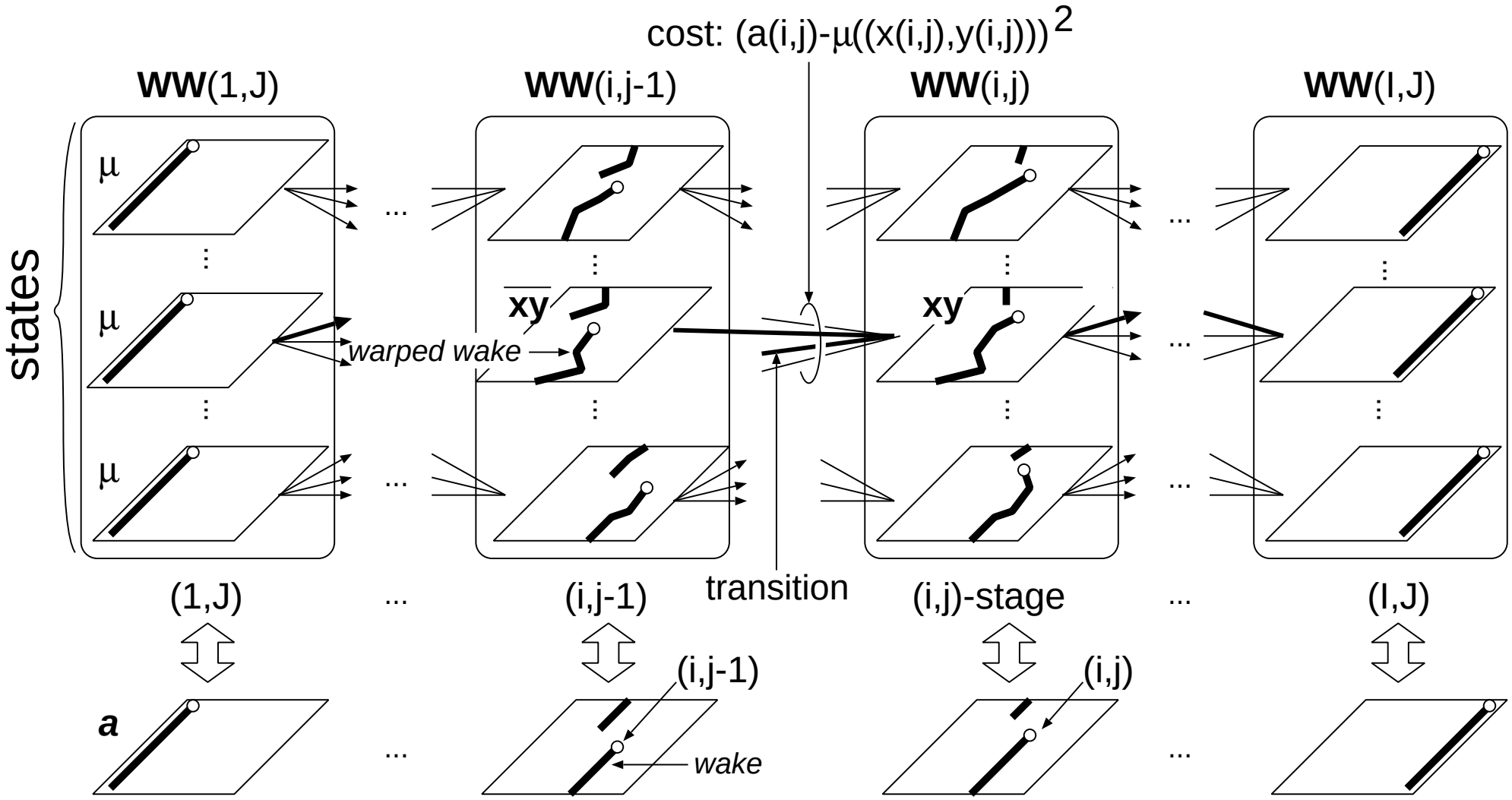
$$D(i,j,xy_{ij}^J) = (\mu_{xy_{ij}^J(i,j)} - a_{ij})^2 + \min_{\overline{xy} \in Pred(xy_{ij}^J)} \begin{cases} D(i-1,J,\overline{xy}) + \mathcal{T}(\ldots) & \text{for } j = 1 \\ D(i,j-1,\overline{xy}) + \mathcal{T}(\ldots) & \text{for } j > 1 \end{cases}$$

Sakoe's algorithm

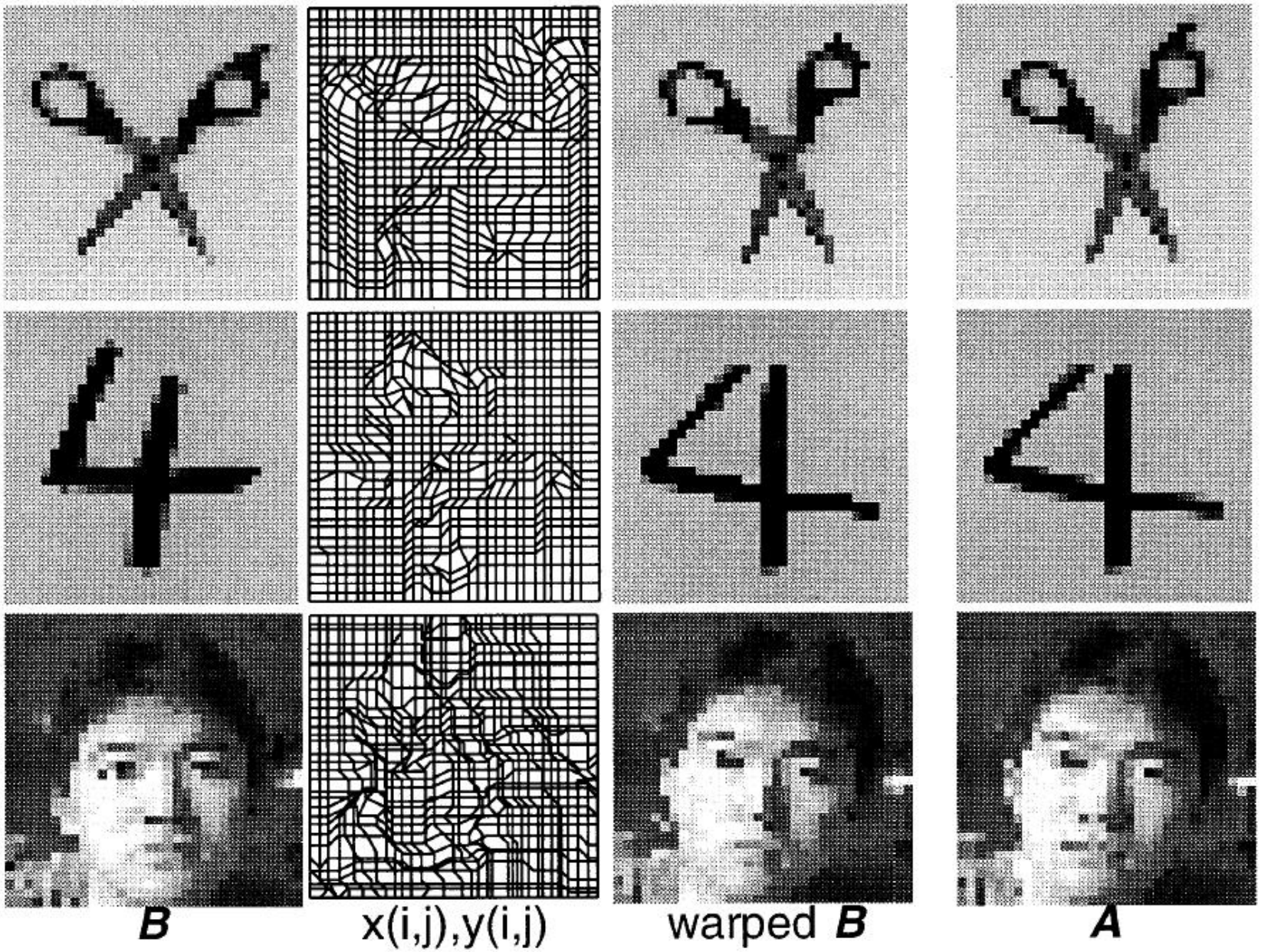


Examples: image a, corresponding pixels of transformed image μ, transformation grid, image μ
top: gray values, bottom: use of derivatives

Sakoe's algorithm



Sakoe's algorithm



no dependency / zero order model

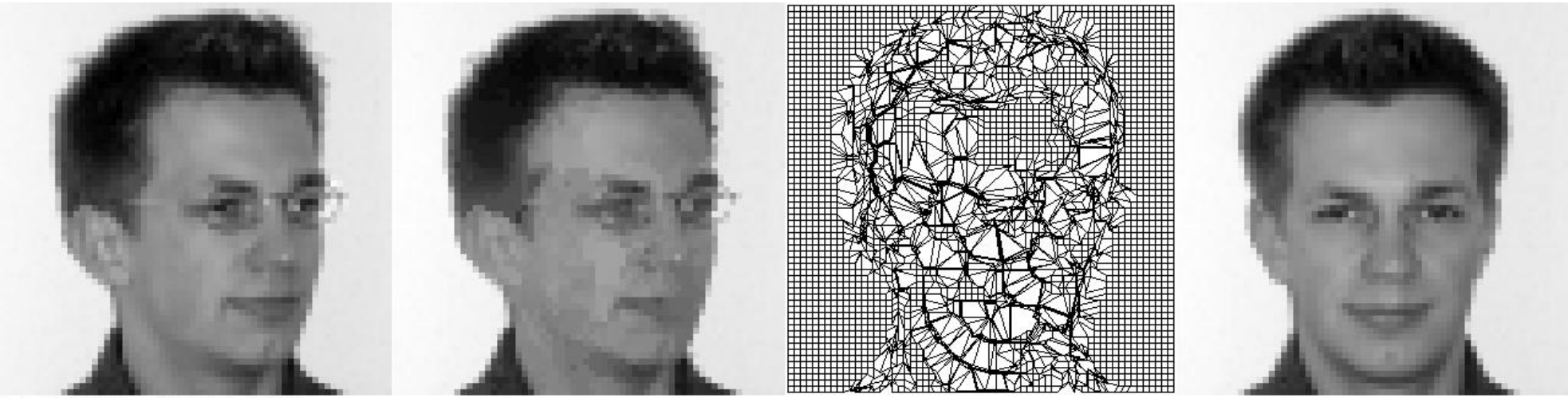
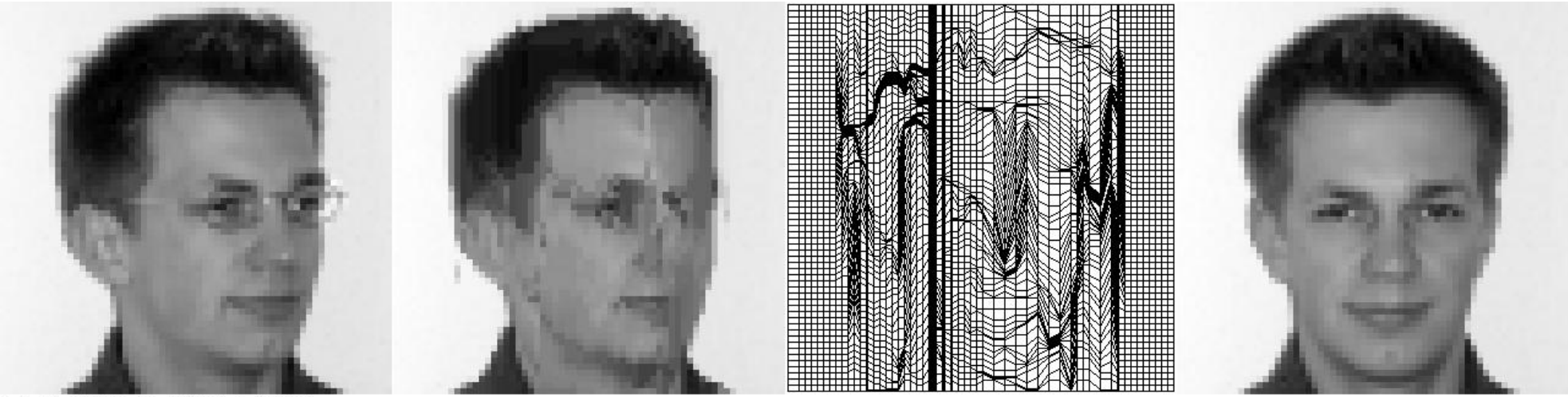


image a, corresponding pixels of transformed image μ, transformation grid, image μ

(warprange = max. displacement = 3)

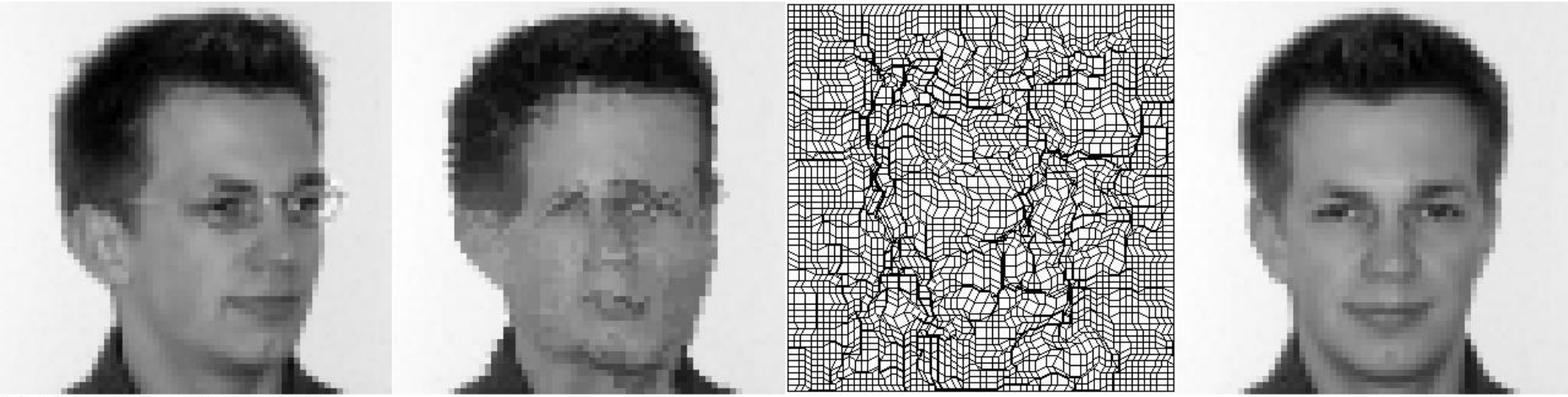
pseudo 2-D HMM



0 1 76.056820 10.284805 7.478805

image a, corresponding pixels of transformed image μ, transformation grid, image μ

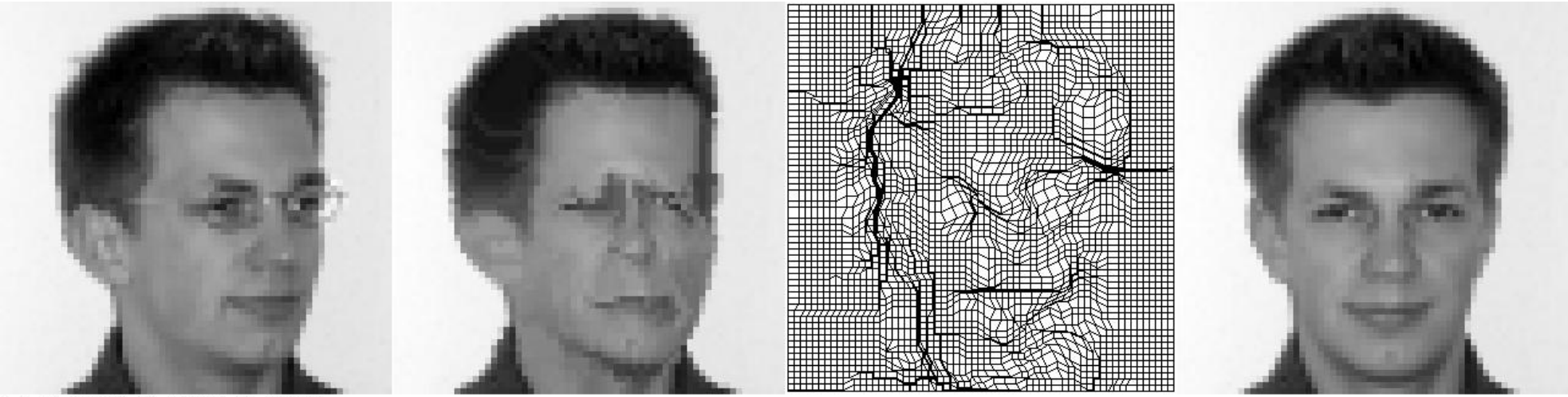
2D model - simulated annealing



0 1 76.056820 21.576448 16.568948

image a, corresponding pixels of transformed image μ, transformation grid, image μ

2D model - Sakoe's algorithm

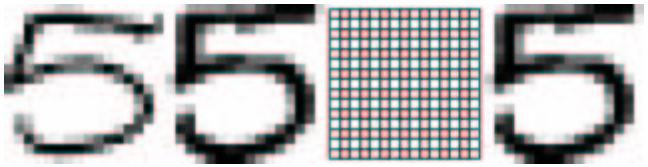


0 1 76.077094 14.863986 11.832733

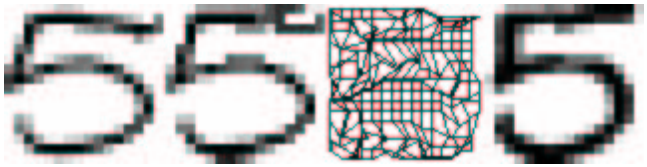
image a, corresponding pixels of transformed image μ, transformation grid, image μ

2-D matching: examples

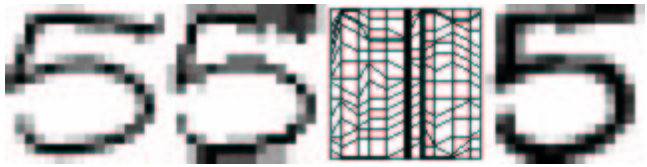
comparison of different algorithms:



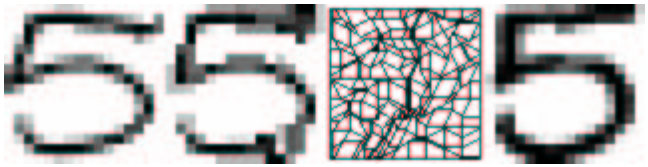
$D_{\text{euk}}(A, B) = \mathbf{26.87}$
relative time: **1**



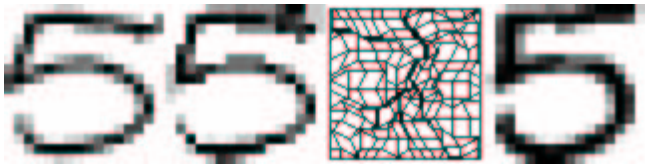
$D_{\text{IDM}}(A, B) = \mathbf{5.90}$
relative time: **7**



$D_{\text{P2D}}(A, B) = \mathbf{12.50}$
relative time: **180**

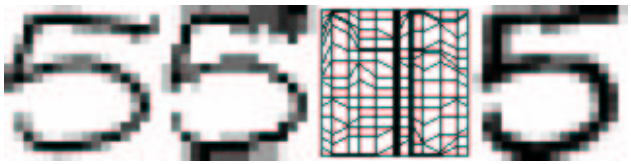


$D_{\text{WW}}(A, B) = \mathbf{11.40}$
beamsize: **5 000**
relative time: **460 000**

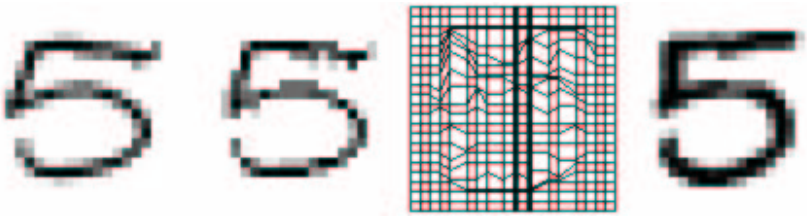


$D_{\text{SA}}(A, B) \approx \mathbf{6.84}$
iterations: **3 000 000**
relative time: **480 000**

relaxation of border constraints:



$D_{\text{P2D}}(A, B) = \mathbf{12.50}$



$D_{\text{P2D}}(A, B) = \mathbf{2.85}$

2-D general case

find optimal alignment $(x, y) : (i, j) \mapsto (x_{ij}, y_{ij})$ between two images a and μ

criterion: greyvalue-difference and cost of mapping

$$\min_{(x_{11}^{IJ}, y_{11}^{IJ})} \left\{ C((x_{11}^{IJ}, y_{11}^{IJ})) + \sum_{i,j} d(a_{ij}, \mu_{x_{ij} y_{ij}}) \right\}$$

further includes

- image distortion model
- tangent distance
- ...