

Labor Matlab für die industrielle und medizinische Bildverarbeitung

Prof. Dr.-Ing. Bodo Rosenhahn

Institut für Informationsverarbeitung



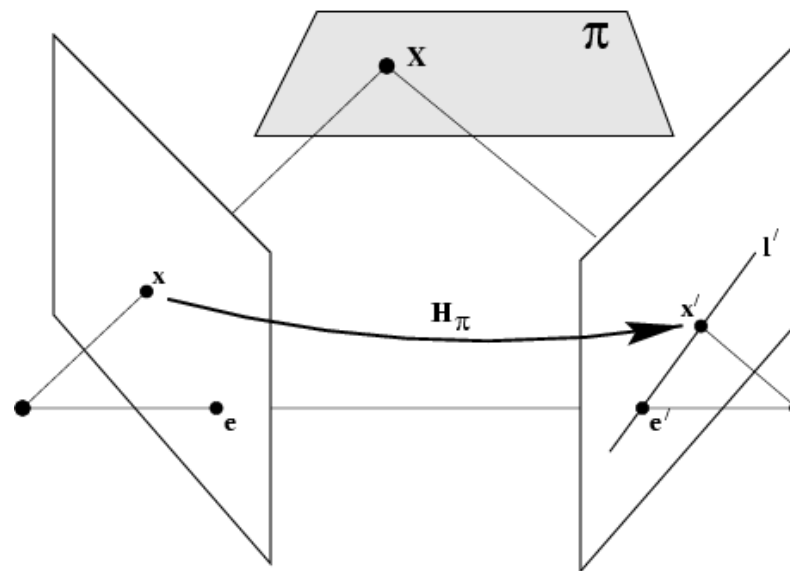
Einleitung

- 19.10. Introduction (1h VL, 3 L), Accountvergabe (Präsenz)
- 26.10. Local operators (Harris, etc.) (1h VL, 3L)
- 02.11. Global Operators (Hough Transform) (1h VL, 3L)
- 09.11. Region Growing / Watershed Segmentation (1h VL, 3L)
- 16.11. Bayes Classifier (1h VL, 3L)
- 23.11. K-Means / Mean shift (1h VL, 3L)
- 30.11. Shape Context (1h VL, 3L)
- 07.12. Morphological Operators (1h VL, 3L)
- 14.12. Disparity estimation (DTW) (1h VL, 3L)
- 21.12. Restarbeiten vor Weihnachten (4L)
- 11.01. Calibration and Triangulation (1h VL, 3L)
- 18.01. PCA (1h VL, 3L)
- 25.01. Tracking (1h VL, 3L)

Heute

1. Dichte Korrespondenzschätzung
2. Dynamische Programmierung

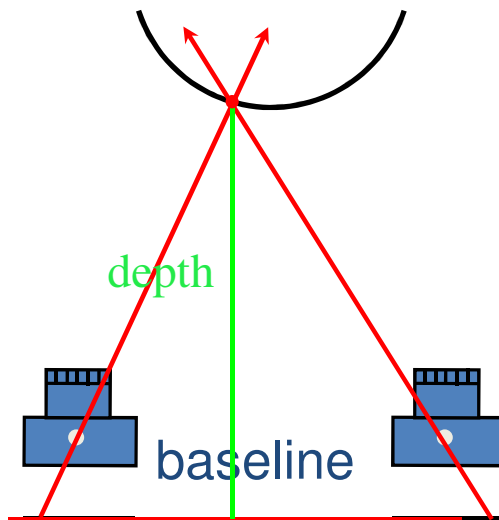
Die Fundamentalmatrix



$$x'^T F x = 0$$

Suchraumreduzierung

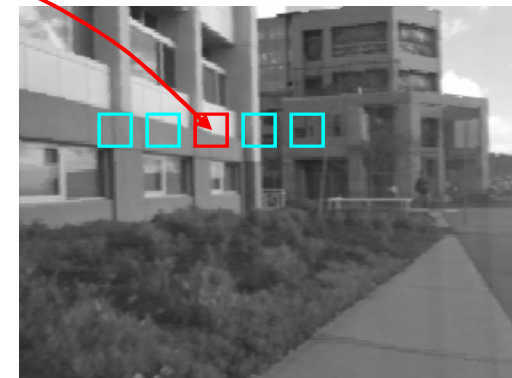
- **Idee:** Rektifizierte Bilder + Verwendung des Epipolarconstraints um den Suchraum einzuschränken



Links



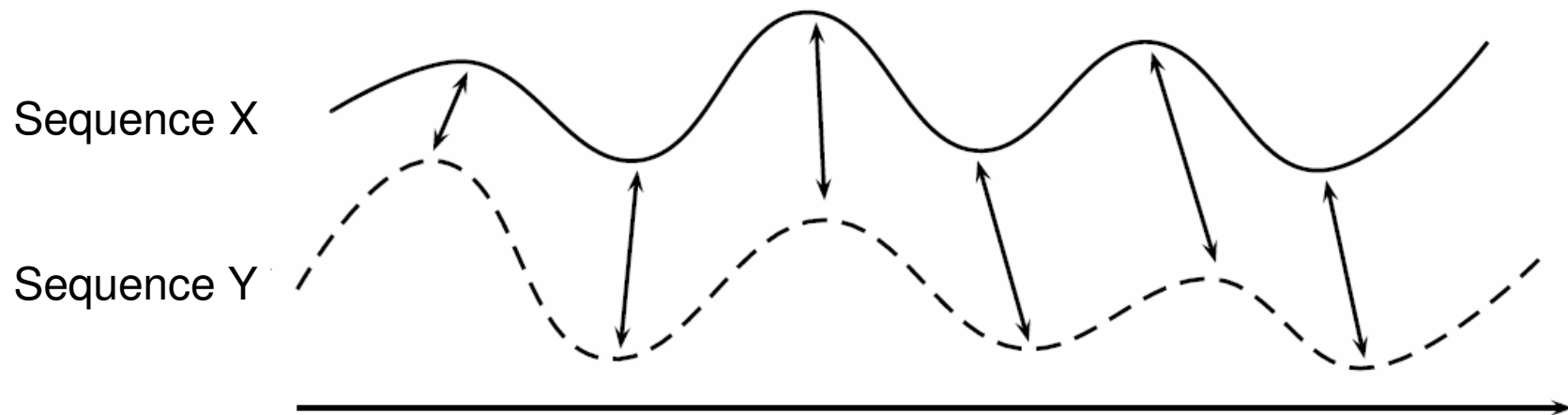
Rechts



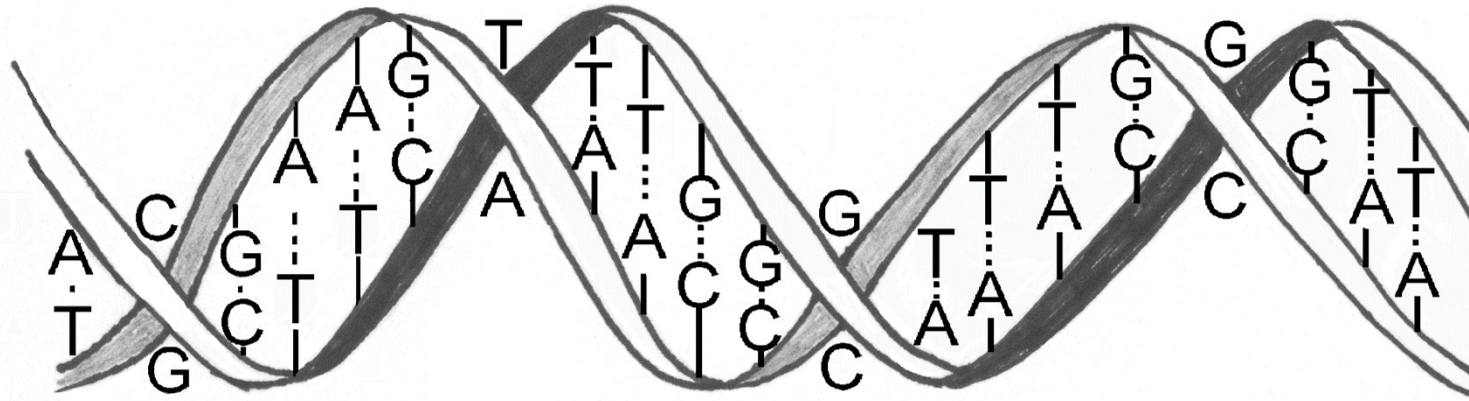
www.cs.ubc.ca/~lowe/425/slides/6-Stereo.ppt

Anwendungen: Matching

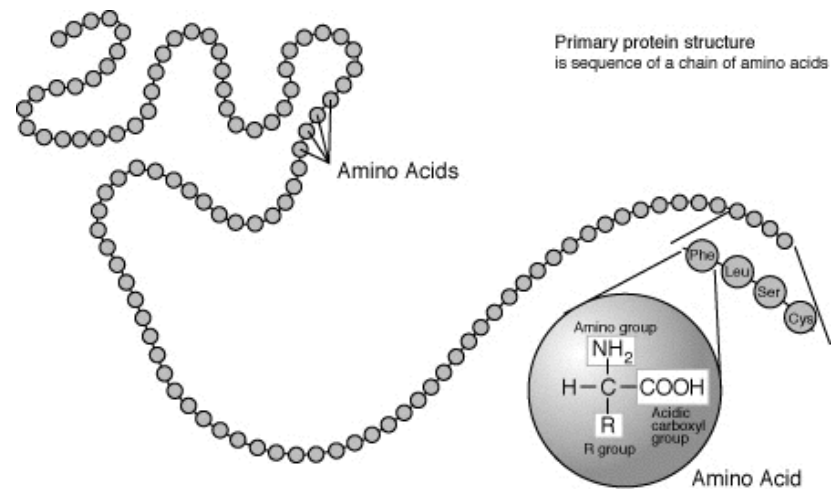
Matching von Sequenzen



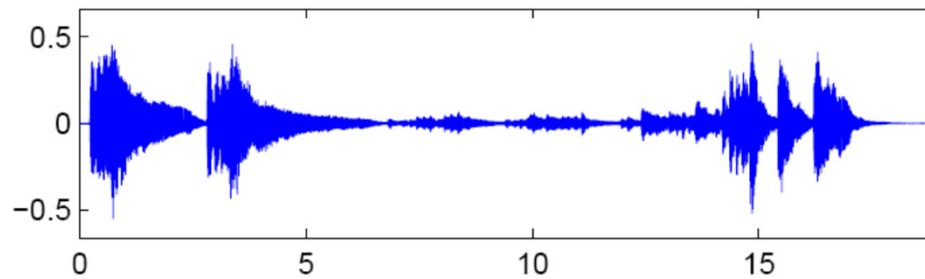
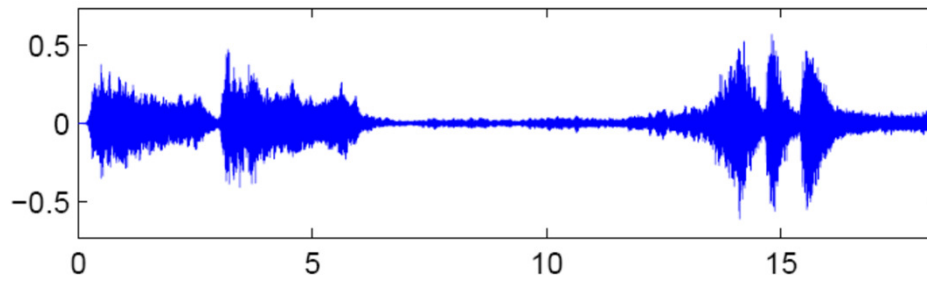
Andere Anwendungen



Analyse von DNA-Sequenzen



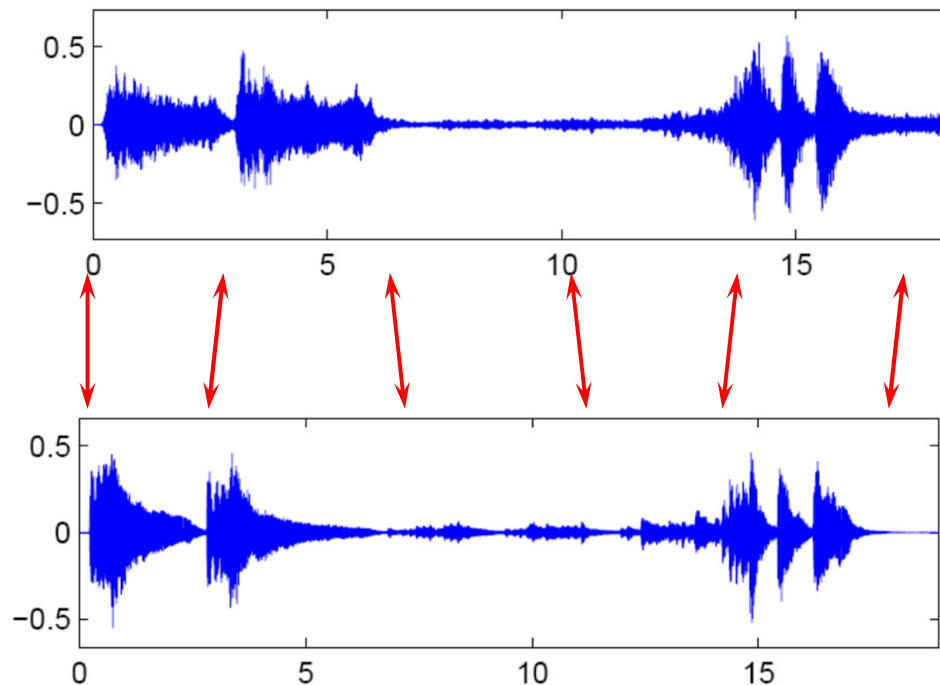
Alignment



Music = Sequence of Audio Samples

Mueller et.al.

Alignment

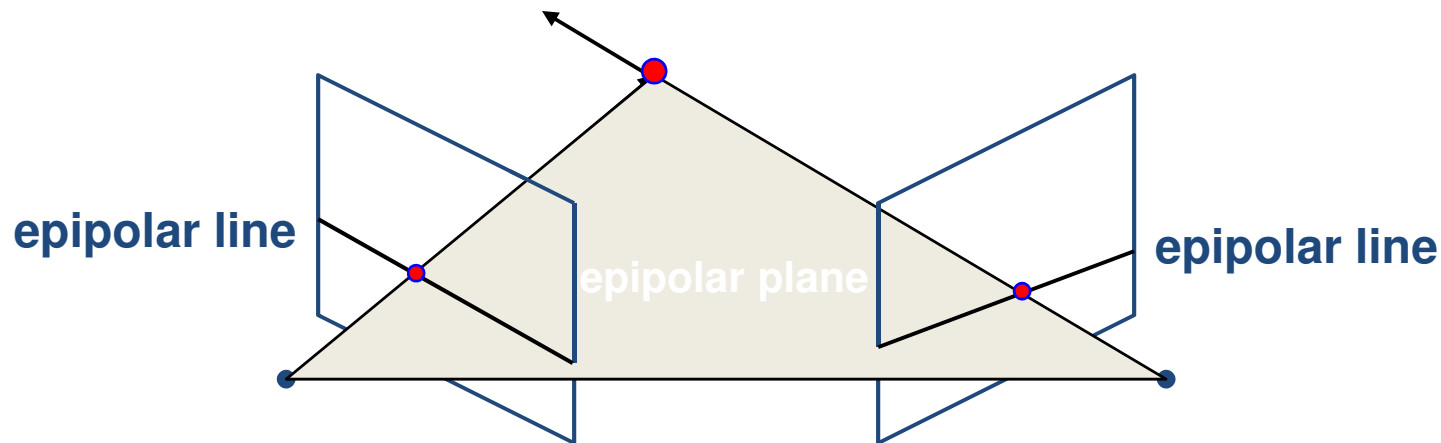


Warping

Music = Sequence of Audio Samples

Mueller et.al.

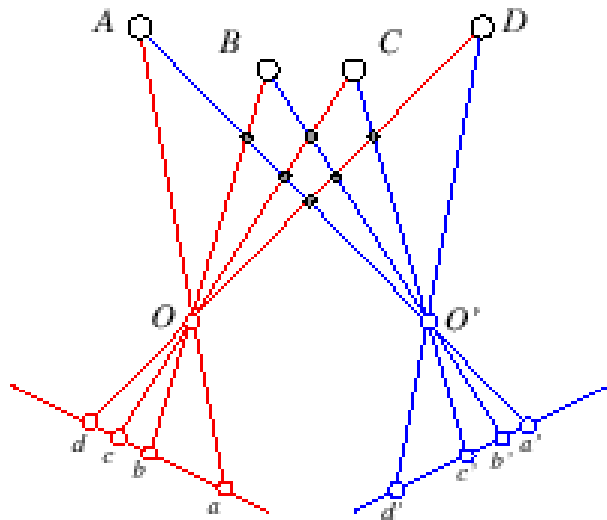
Der Epipolar Constraint



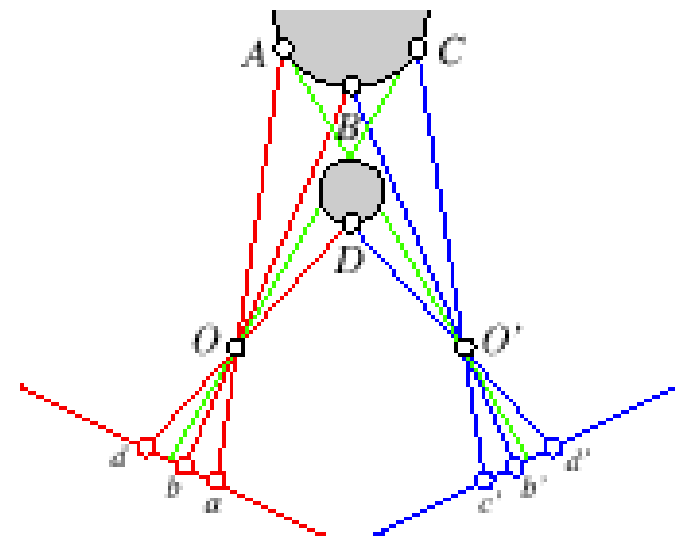
- Epipolar Constraint
 - Korrespondierende Punkte liegen entlang einer Epipolargeraden
 - Der Suchraum für das Korrespondenzproblem wird auf eine 1D-Suche entlang der korrespondierenden Epipolargeraden reduziert. (schnelle Verfahren möglich)

Korrespondenzproblem:

- Prinzipiell ist die Ordnung der Pixel uneindeutig, da sie abhängig von der Tiefe der Pixel ist !



Anordnung erfüllt



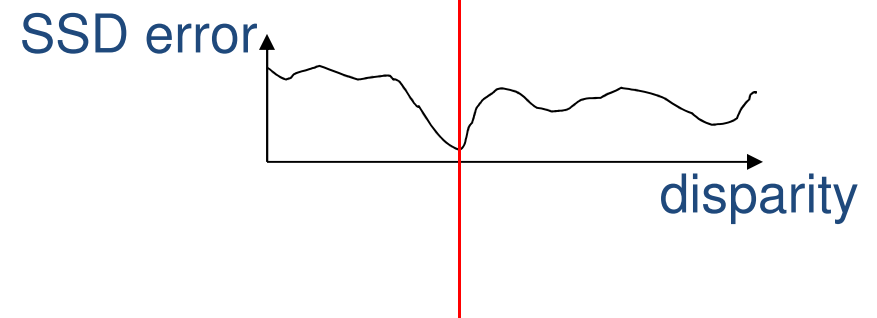
Anordnung nicht erfüllt

Korrelations - Analyse

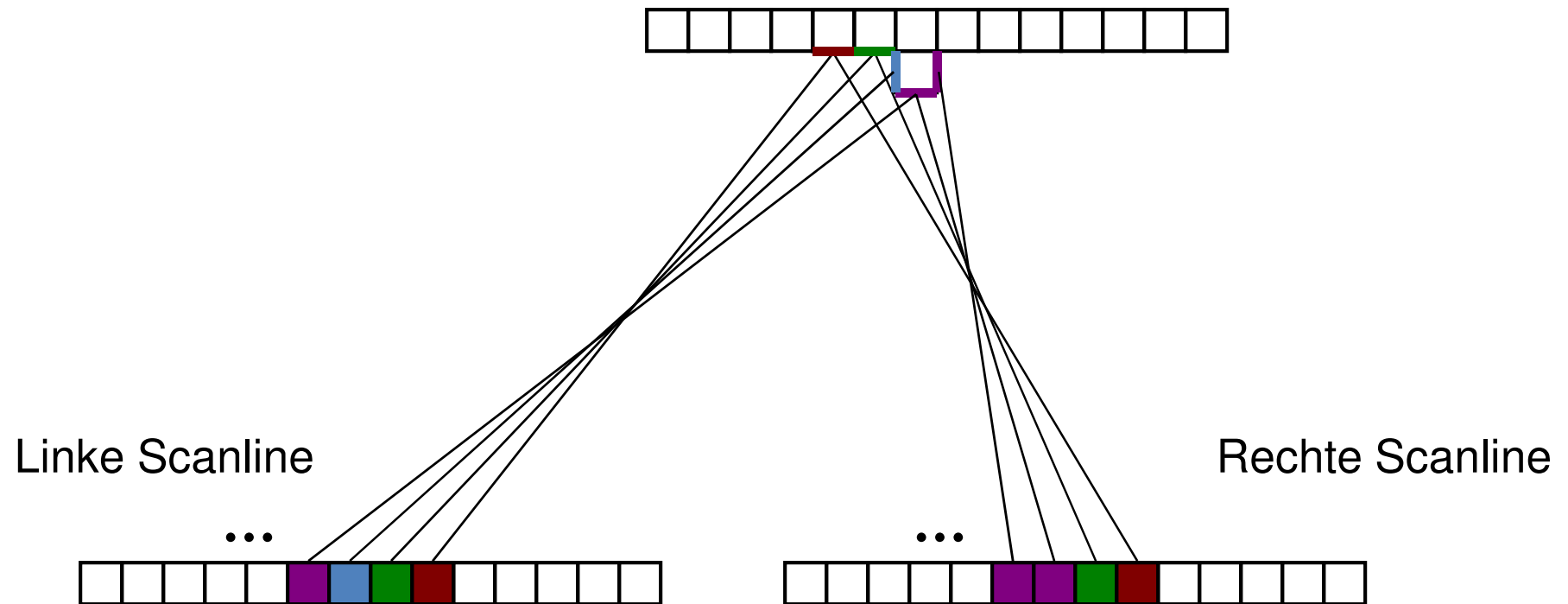
Links

Rechts

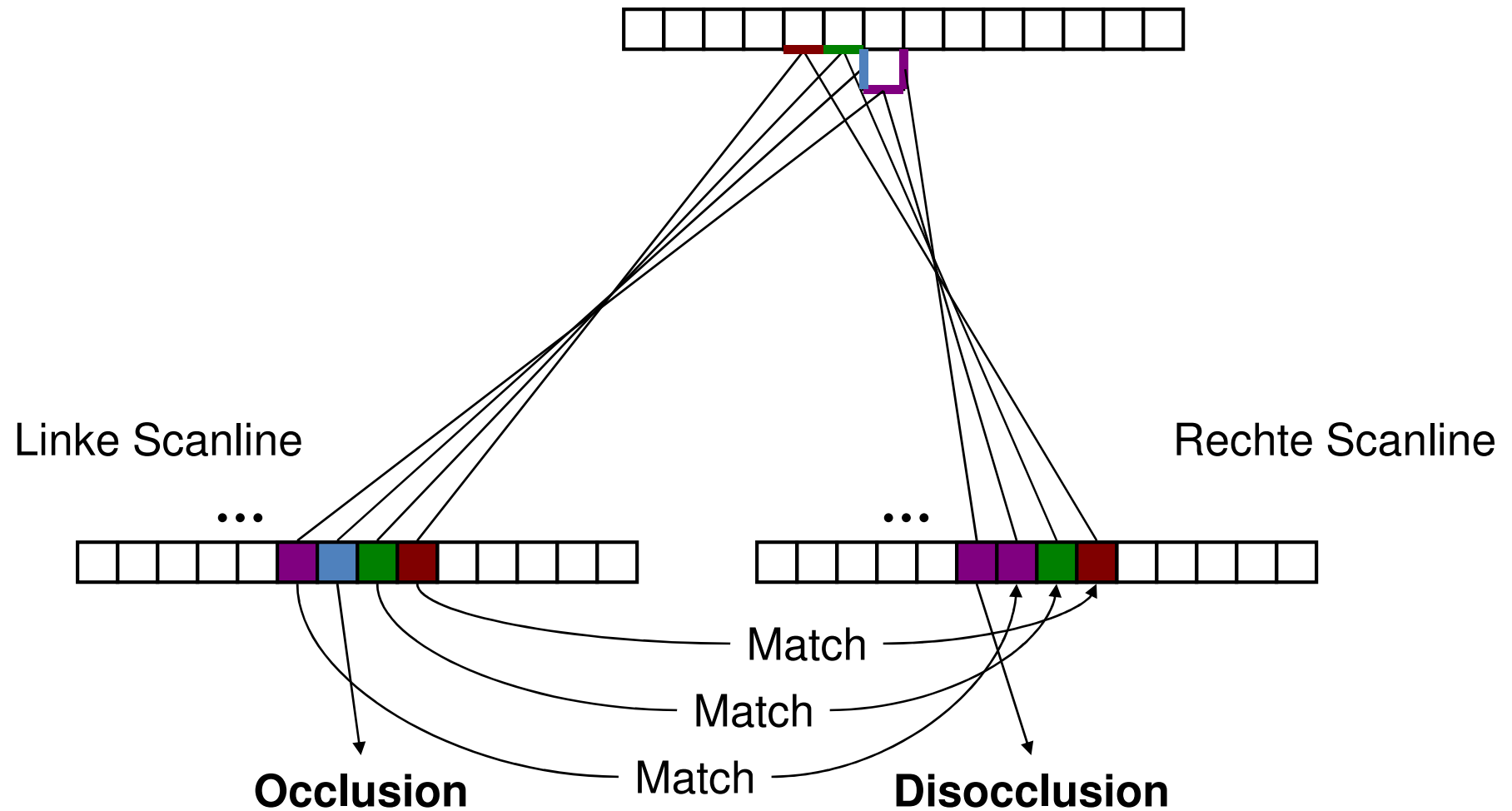
Scanlinie



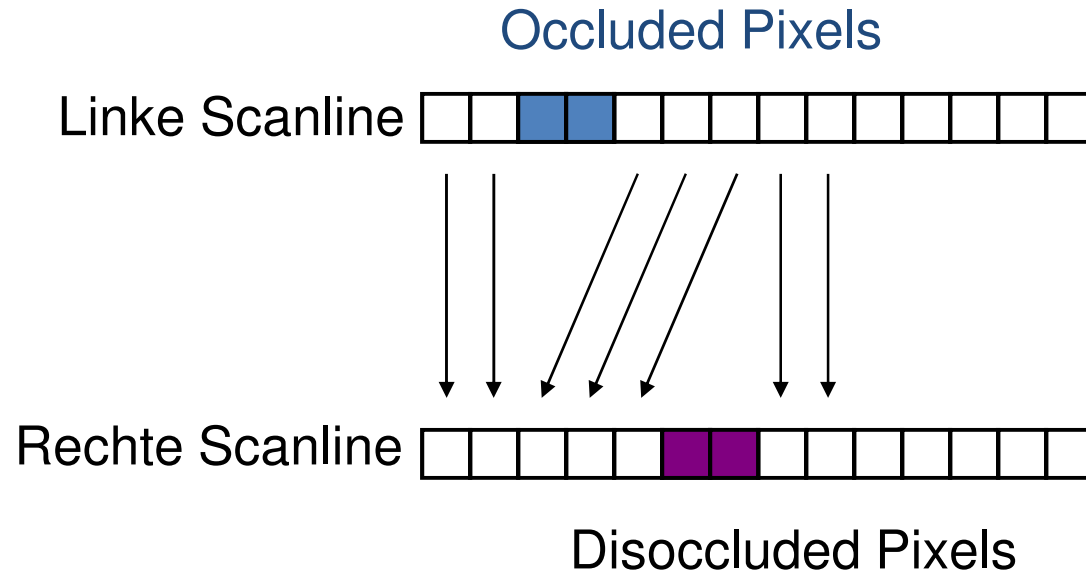
Stereo Korrespondenzen



Stereo Korrespondenzen



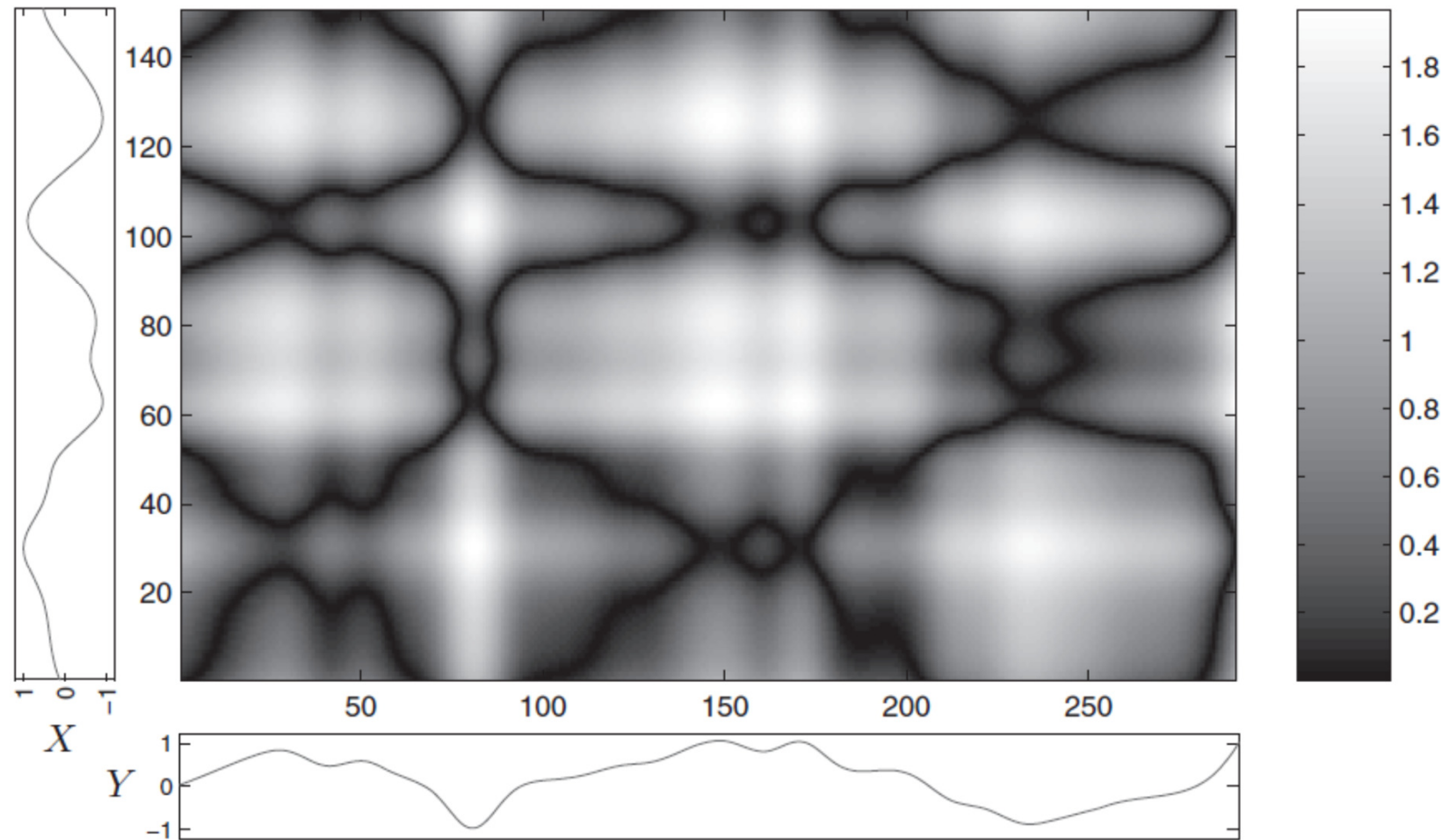
Suche über alle Korrespondenzen



3 Fälle:

- Sequential
 - Occluded
 - Disocclude
- addiere Kosten des Matches (klein, wenn Intensitäten übereinstimmen)
 - addiere Kosten zu “kein Match” (große Kosten)
 - addiere Kosten zu “kein Match” (große Kosten)

Kostenmatrix

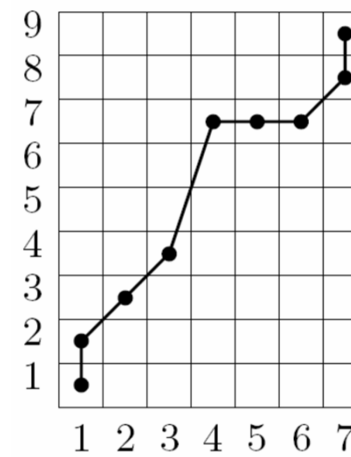
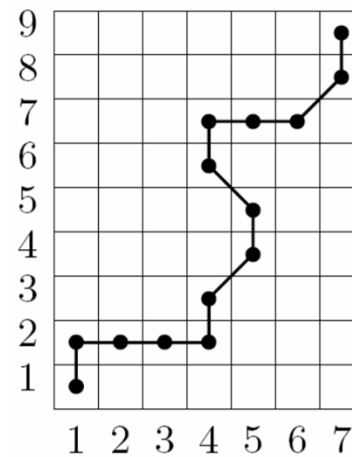
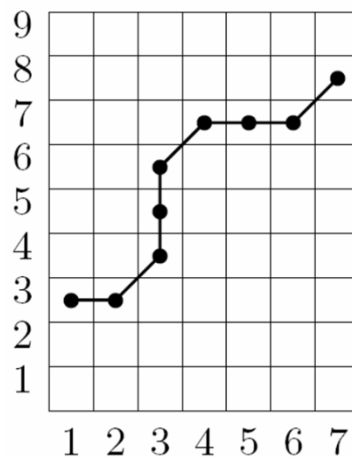


Warping Pfad

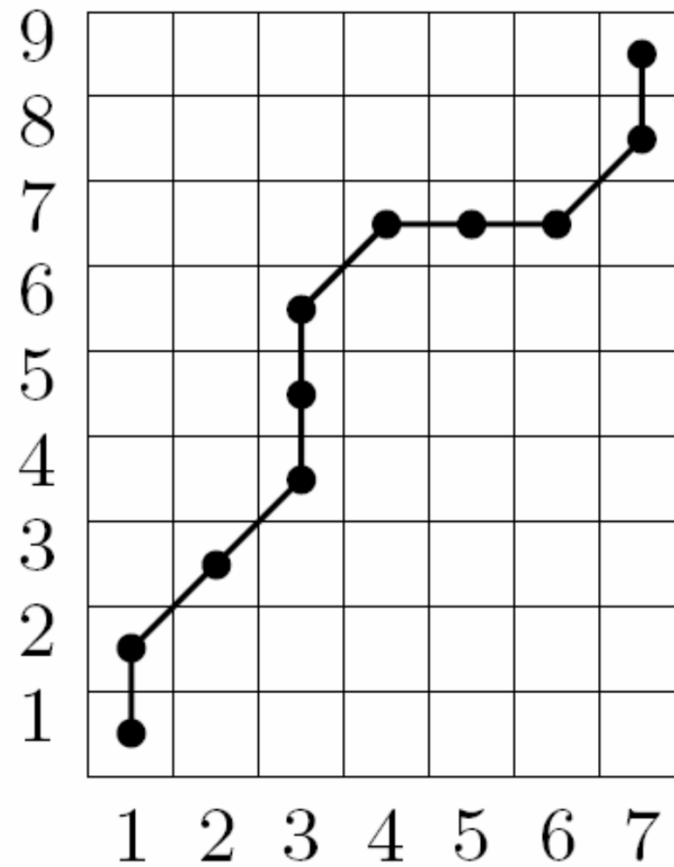
Definition 4.1. An (N, M) -warping path (or simply referred to as warping path if N and M are clear from the context) is a sequence $p = (p_1, \dots, p_L)$ with $p_\ell = (n_\ell, m_\ell) \in [1 : N] \times [1 : M]$ for $\ell \in [1 : L]$ satisfying the following three conditions.

- (i) Boundary condition: $p_1 = (1, 1)$ and $p_L = (N, M)$.
- (ii) Monotonicity condition: $n_1 \leq n_2 \leq \dots \leq n_L$ and $m_1 \leq m_2 \leq \dots \leq m_L$.
- (iii) Step size condition: $p_{\ell+1} - p_\ell \in \{(1, 0), (0, 1), (1, 1)\}$ for $\ell \in [1 : L - 1]$.

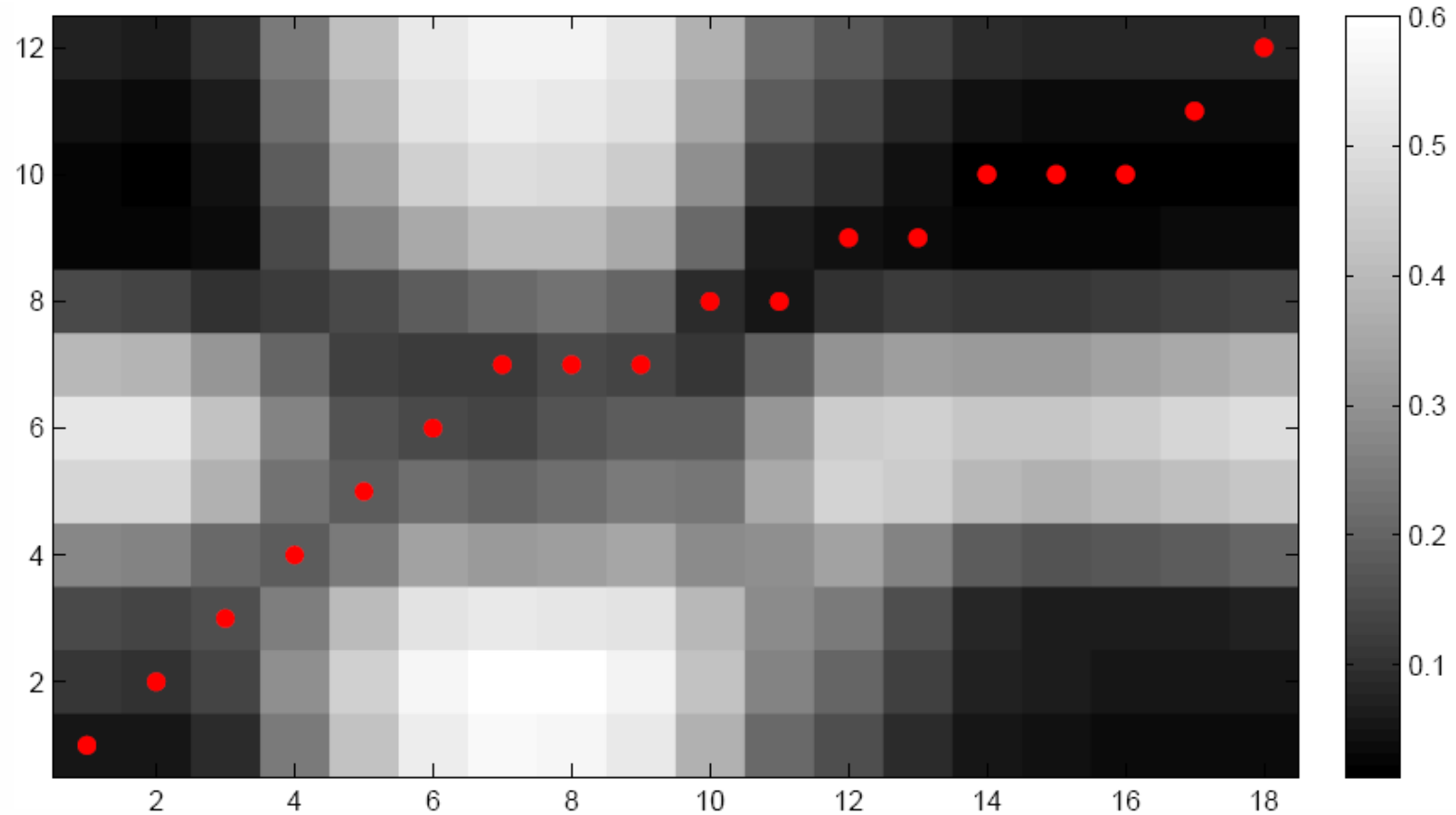
Gegenbeispiele zu (i)-(iii)



Warping-Pfad



Minimale Pfadsuche



Dynamic Time Warping

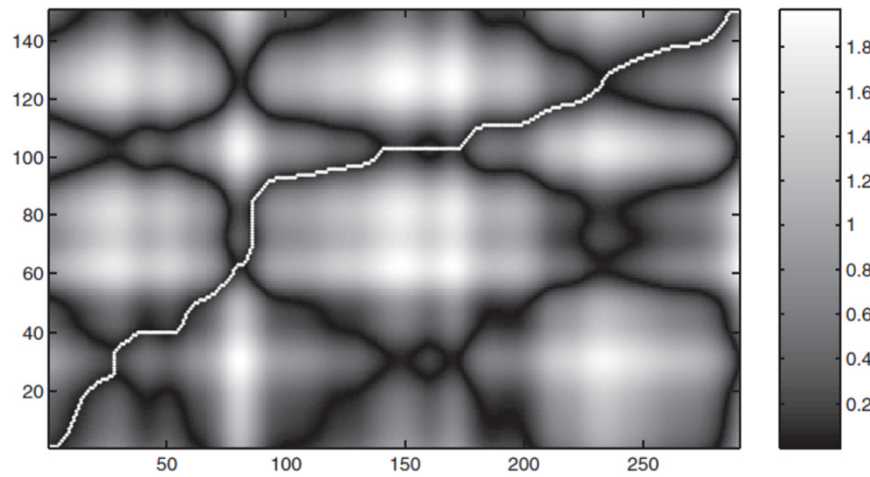
Dynamic time warping ist ein Verfahren, um die Ähnlichkeit zwischen zwei Sequenzen (die in Zeit und Geschwindigkeit variieren) zu messen.

Algorithmus:

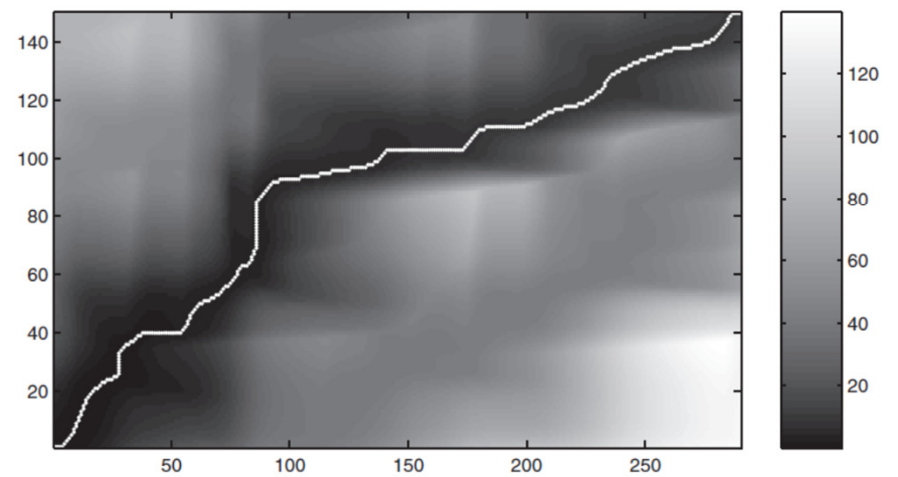
```
int DTWDistance(char s[1..n], char t[1..m], int d[1..n,1..m]) {  
  declare int DTW[0..n, 0..m]  
  declare int i, j, cost  
  
  for i := 1 to m  
    DTW[0,i] := infinity  
  for i := 1 to n  
    DTW[i,0] := infinity  
  DTW[0,0] := 0  
  for i := 1 to n  
    for j := 1 to m  
      cost := d[s[i],t[j]]  
      DTW[i,j] := cost + minimum(DTW[ i-1, j ], // insertion  
                                DTW[ i , j-1 ], // deletion  
                                DTW[ i-1, j-1 ]) // match  
  
  return DTW[n,m] }
```

Beispiel

(a)



(b)



1. Distanzmatrix

$$t = [1, 3, 4, 5]$$

$$s = [1, 2, 4, 5]$$

$$DTW = \begin{bmatrix} 0 & 2 & 5 & 9 \\ 1 & 1 & 3 & 6 \\ 4 & 2 & 1 & 2 \\ 8 & 4 & 2 & 1 \end{bmatrix}$$

Zeitaufwand: $n \cdot m$ (Dimension der Matrix)

2. Schritt: Analyse der Distanzmatrix

2. Schritt: Backtracking

- Analyse der DTW - Matrix durch Backtracking

Effiziente Lösung: dynamisches Programmieren

Dynamische Programmierung ist ein Paradigma zum algorithmischen Lösen von Optimierungsproblemen. Der Begriff wurde in den 1940er Jahren von dem amerikanischen Mathematiker Richard Bellman eingeführt, der diese Methode auf dem Gebiet der Kontrolltheorie anwendete. In diesem Zusammenhang wird auch oft von *Bellmans Prinzip der dynamischen Programmierung* gesprochen.

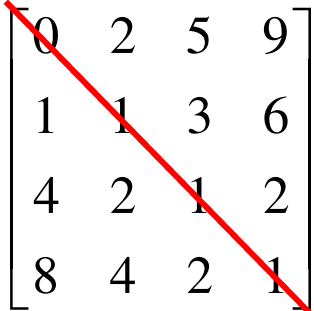
Dynamische Programmierung kann dann erfolgreich eingesetzt werden, wenn das Optimierungsproblem aus vielen gleichartigen Teilproblemen besteht, und eine optimale Lösung des Problems sich aus optimalen Lösungen der Teilprobleme zusammensetzt

http://de.wikipedia.org/wiki/Dynamische_Programmierung

Dynamische Programmierung

$$t = [1, 2, 3, 4, 5]$$

$$s = [1, 2, 4, 5]$$

$$DTW = \begin{bmatrix} 0 & 2 & 5 & 9 \\ 1 & 1 & 3 & 6 \\ 4 & 2 & 1 & 2 \\ 8 & 4 & 2 & 1 \end{bmatrix}$$


Für die Analyse: starte rechts unten

Problem: Uneindeutigkeiten

Beispielkostenmatrix

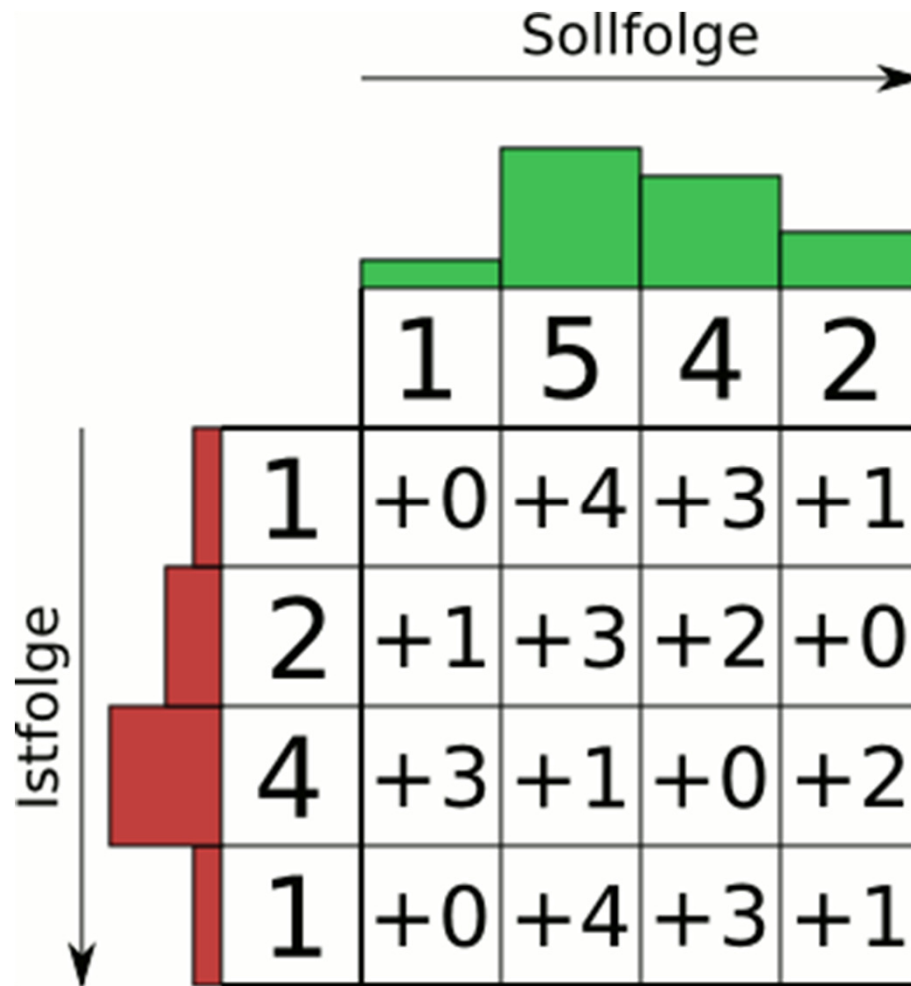
	1	2	3	0	0	2	3	2	0
1	0	1	3	4	5	6	8	9	10
2	1	0	1	3	5	5	6	6	8
3	3	1	0	3	6	6	5	6	9
3	5	2	0	3	6	7	5	6	9
3	7	3	0	3	6	7	5	6	9
0	8	5	3	0	0	2	5	7	6
0	9	7	6	0	0	2	5	7	6
2	10	7	7	2	2	0	1	1	3
3	12	8	7	5	5	1	0	1	4

Lösung: Bestrafungsterme für Occluder

$$M[i, j] = \min(M[i-1, j] + \text{Verdeckungskosten}, M[i, j-1] + \text{Verdeckungskosten}, M[i-1, j-1] + \text{Kosten}(i, j))$$

Verdeckungskosten sollten abhängig vom Bildinhalt gewählt werden !

Animation



http://upload.wikimedia.org/wikipedia/de/8/84/Animation_Dynamic_Time_Warping.gif

Stereo - Algorithmus

“A Maximum Likelihood Stereo Algorithm”, Cox, Hingorani, Rao, Maggs, Computer Vision & Image Understanding, 63, 3, pp. 542-567.

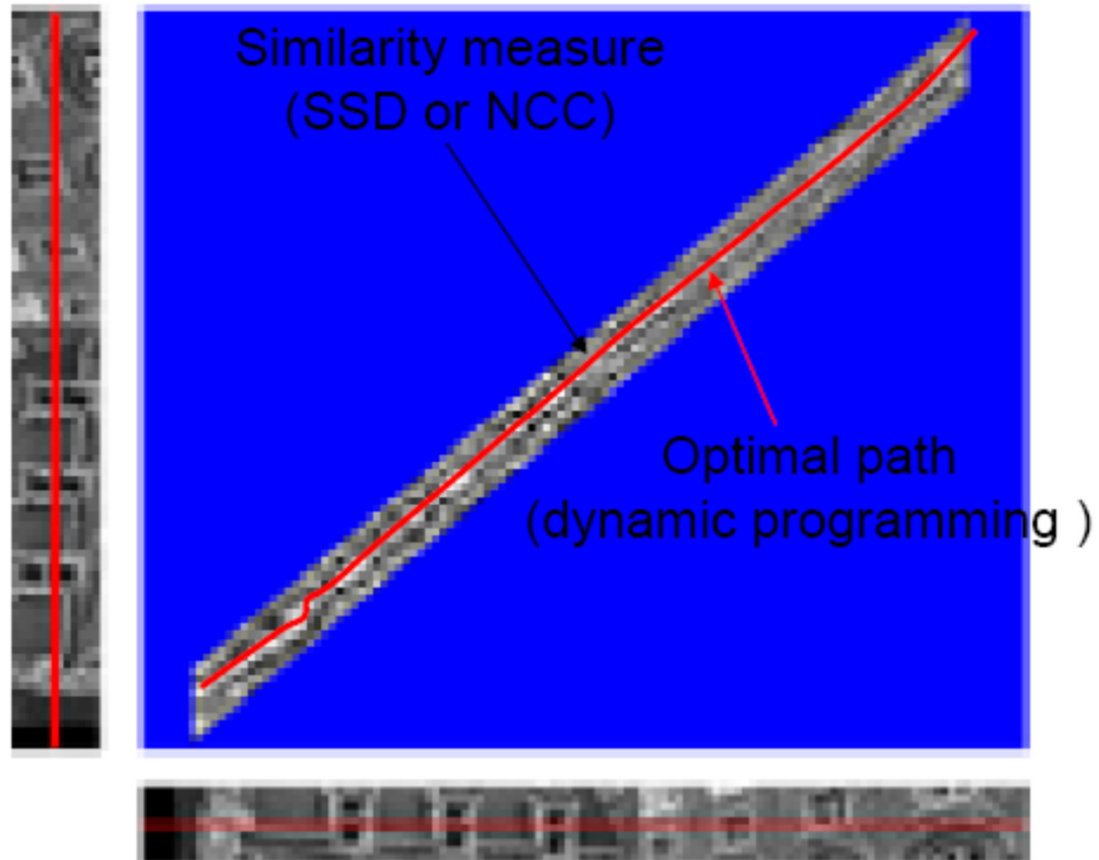
```
Occlusion =  $\left[ \ln \left( \frac{P_D}{1-P_D} \frac{\phi}{|(2\pi)^d \mathbf{S}_s^{-1}|^{\frac{1}{2}}} \right) \right]$ 
for (i=1; i ≤ N; i++) { C(i,0) = i*Occlusion }
for (i=1; i ≤ M; i++) { C(0,i) = i*Occlusion }
for (i=1; i ≤ N; i++) {
    for (j=1; j ≤ M; j++) {
        min1 = C(i-1,j-1)+c(z1,i,z2,j);
        min2 = C(i-1,j)+Occlusion;
        min3 = C(i,j-1)+Occlusion;
        C(i,j) = cmin = min(min1,min2,min3);
        if (min1==cmin) M(i,j) = 1;
        if (min2==cmin) M(i,j) = 2;
        if (min3==cmin) M(i,j) = 3;
    }
}
```

DTW

```
p=N;
q=M;
while(p!=0 && q!=0){
    switch(M(p,q)){
        case 1:
            p matches q
            p--;q--;
            break;
        case 2:
            p is unmatched
            p--;
            break;
        case 3:
            q is unmatched
            q--;
            break;
    }
}
```

Backtracking

Beispiel



DTW - Matlab Beispiel

D:\home\rosenhahn\Vorlesung\HannoverSzenenanalyse\MatlabExamples\DTW-VL

DTW_Classical.m

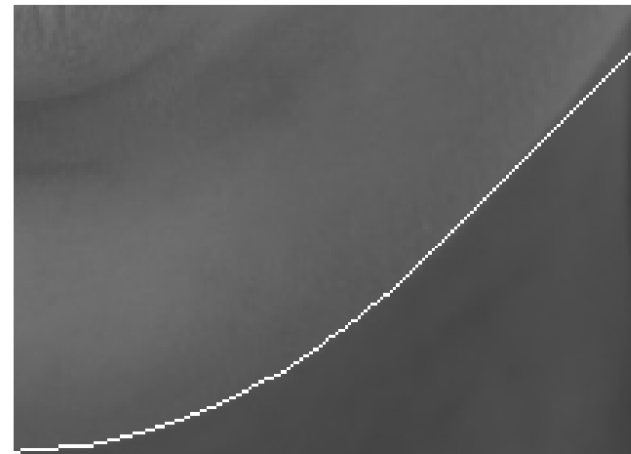
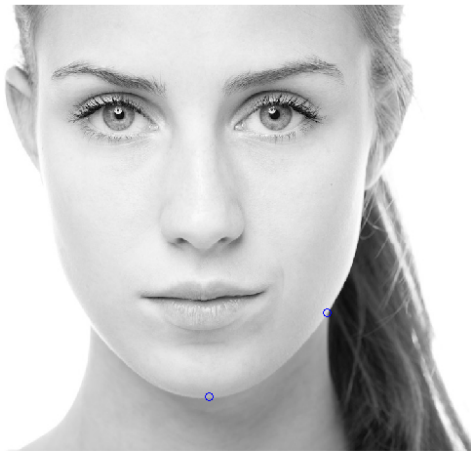
Test_DTW_Vorlesung.m

(© Meinard Mueller MPI Saarbrücken)

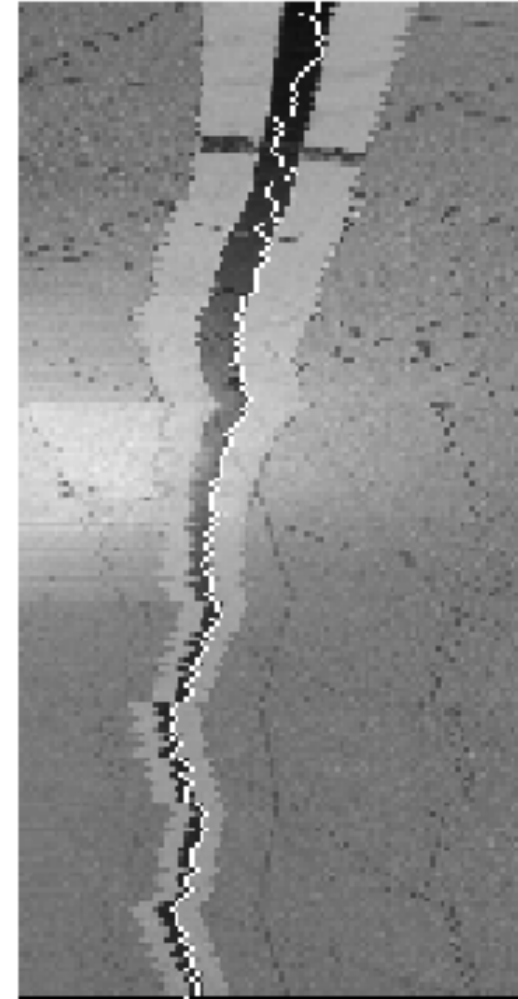
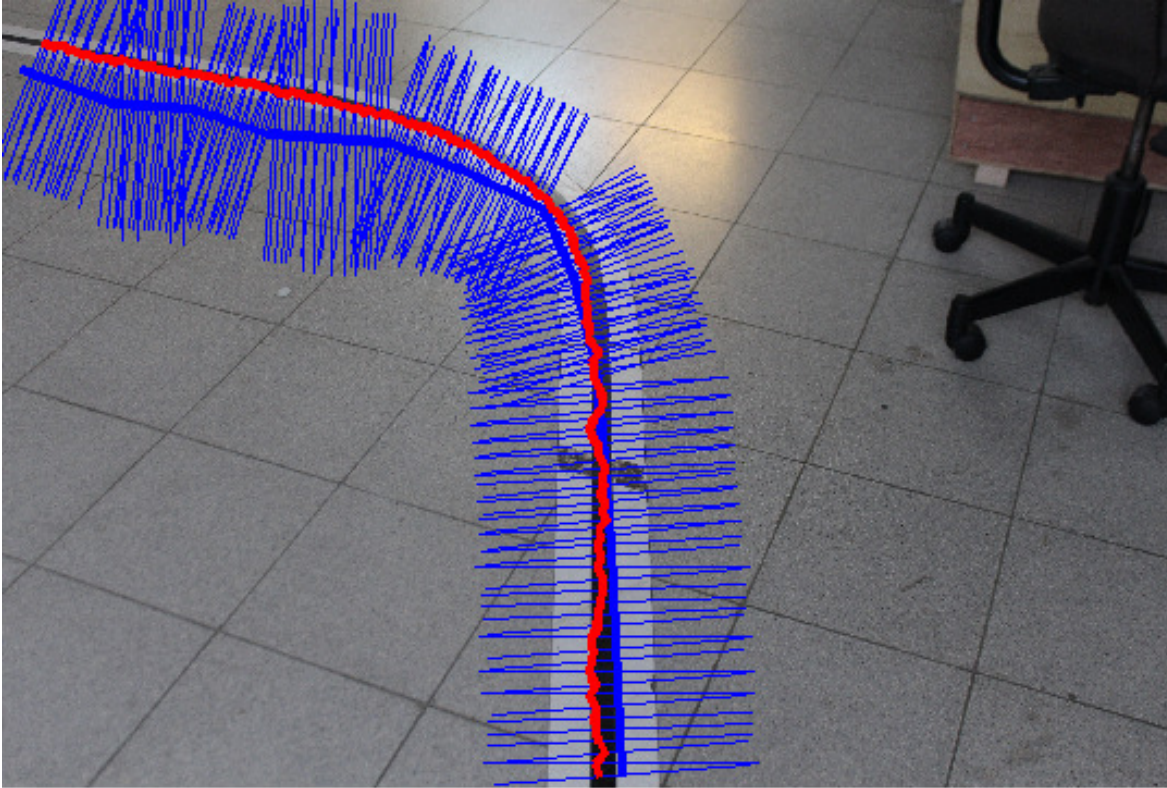
Stereo - Matlab Beispiel



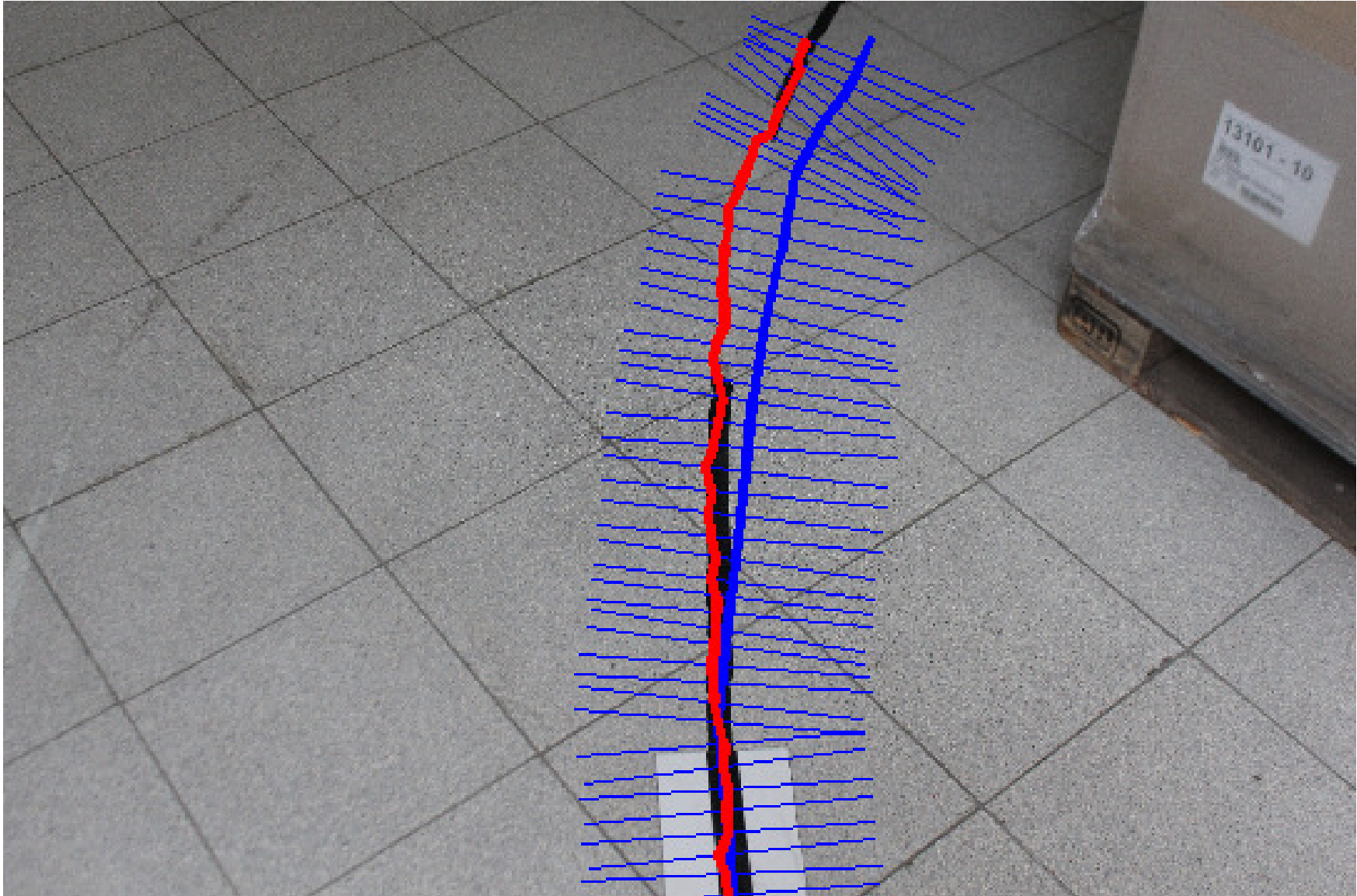
Shortest Path Beispiel (Konturfindung)



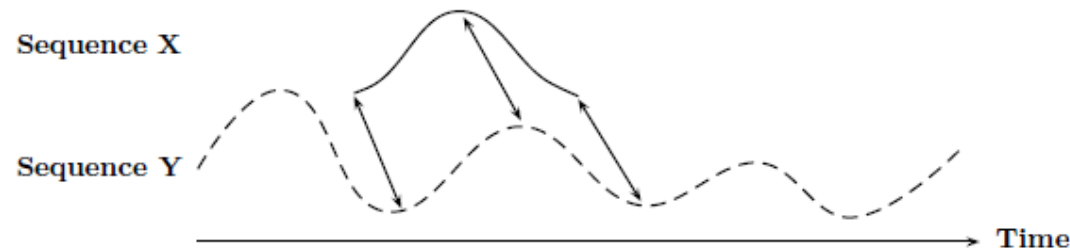
Shortest Path Beispiel (Konturfindung)



Shortest Path Beispiel (Konturfindung)



Subsequence DTW



Algorithm: COMPUTESIMILARSUBSEQUENCES

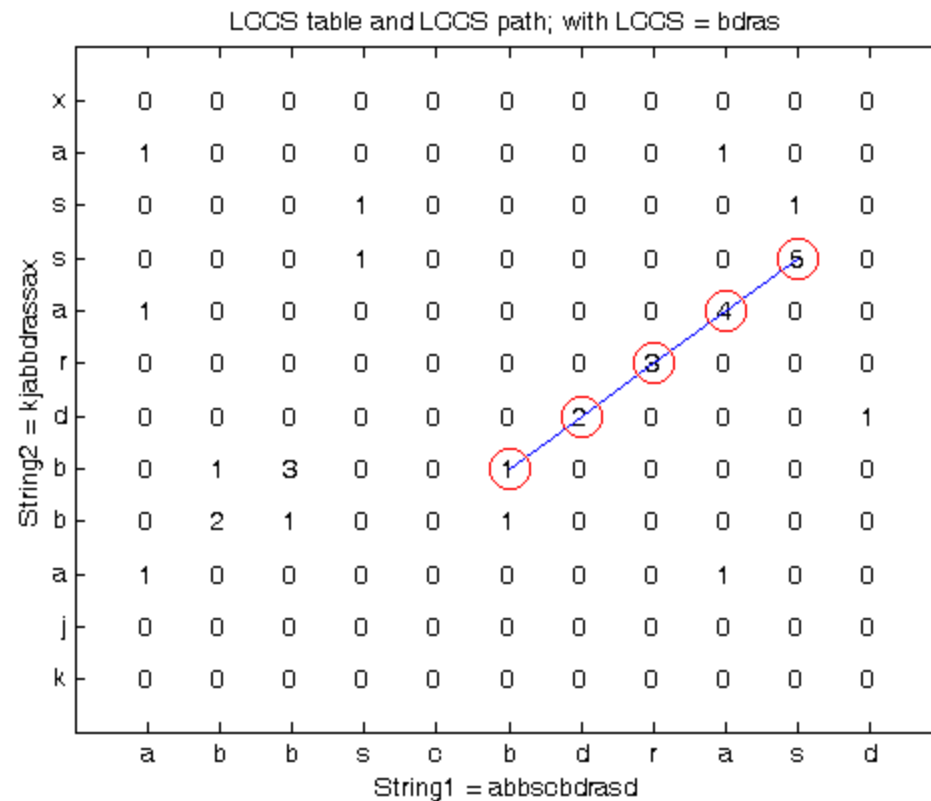
Input: $X = (x_1, \dots, x_N)$ query sequence
 $Y = (y_1, \dots, y_M)$ database sequence
 $\tau \in \mathbb{R}$ cost threshold

Output: Ranked list of all (essential distinct) subsequences of Y that have a DTW distance to X below the threshold τ .

- (0) Initialize the ranked list to be the empty list.
- (1) Compute the accumulated cost matrix D w.r.t. X and Y ,
- (2) Determine the distance function Δ as in (4.12).
- (3) Determine the minimum $b^* \in [1 : M]$ of Δ .
- (4) If $\Delta(b^*) > \tau$ then terminate the procedure.
- (5) Compute the corresponding DTW-minimizing index $a^* \in [1 : M]$.
- (6) Extend the ranked list by the subsequence $Y(a^* : b^*)$.
- (7) Set $\Delta(b) := \infty$ for all b within a suitable neighborhood of b^* .
- (8) Continue with Step (3).

LCCS

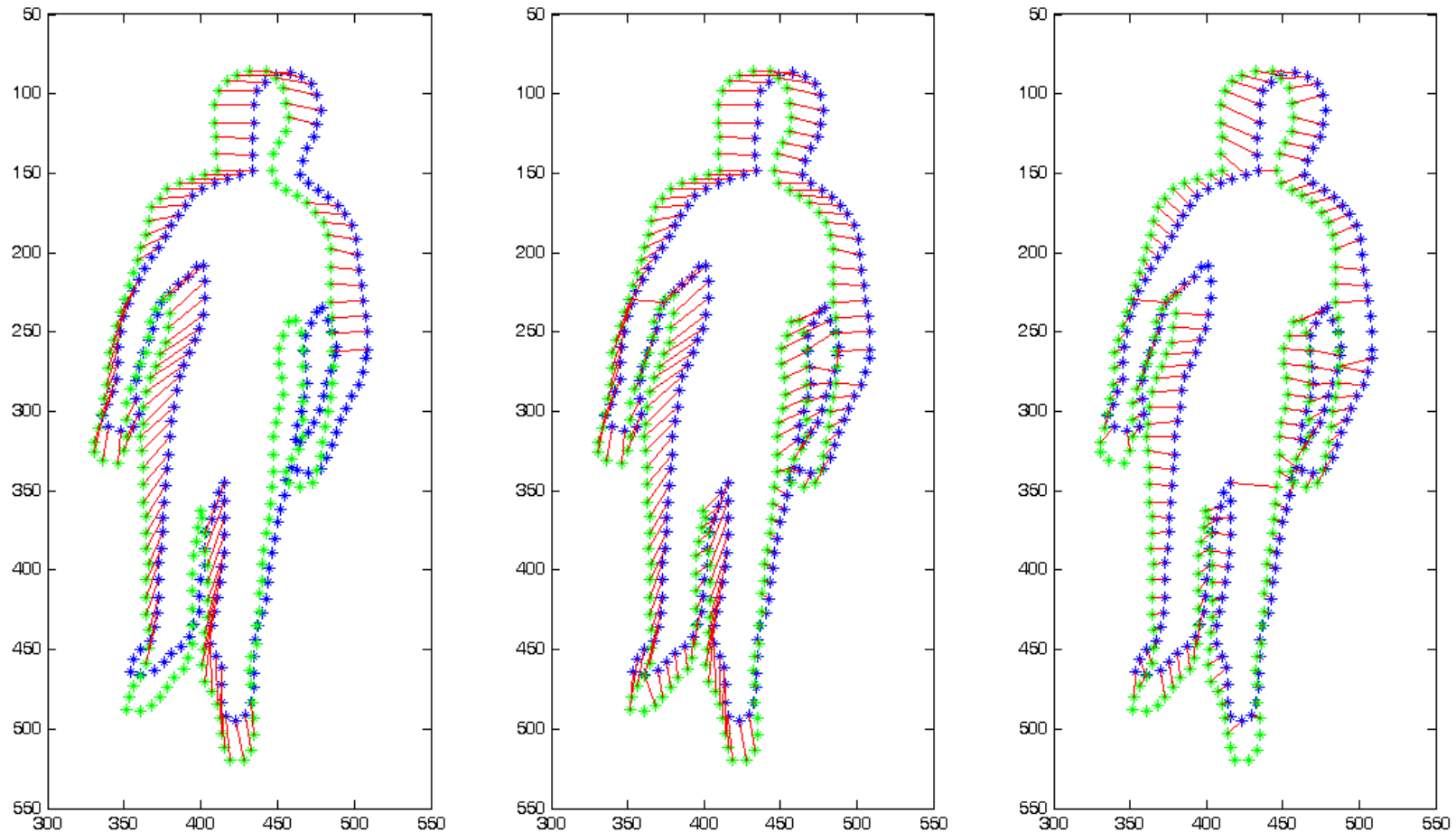
Longest Common Consecutive Subsequence



D:\home\rosenhahn\Vorlesung\HannoverTracking\MatlabExamples\LCCS

LCCS

Longest Common Consecutive Subsequence as prior in Cost matrix



D:\home\rosenhahn\Vorlesung\HannoverTracking\MatlabExamples\LCCS\gradients.m

Zusammenfassung

- Dichte Korrespondenzfindung mittels DTW
- Allgemeine Anwendbarkeit für verschiedene Problemstellungen
- Konzept: Dynamische Programmierung