EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

MATHEMATISCH-
NATURWISSENSCHAFTLICHE FAKULTÄT

Medieninformatik / Human-Computer Interaction

Grundlagen der Multimediatechnik

Dynamic Time Warping
– ein Algorithmus, viele Anwendungen –

21.01.2022, Prof. Dr. Enkelejda Kasneci

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Termine und Themen

22.10.2021	Einführung
29.10.2021	Menschliche Wahrnehmung – visuell, akustisch, haptisch, …
05.11.2021	Informationstheorie, Textcodierung und -komprimierung
12.11.2021	Bildverbesserung
19.11.2021	Bildanalyse
26.11.2021	Grundlagen der Signalverarbeitung
03.12.2021	Bildkomprimierung
10.12.2021	Videokomprimierung
17.12.2022	Audiokomprimierung
14.01.2022	Videoanalyse
21.01.2022	Dynamic Time Warping
28.01.2022	Gestenanalyse
04.02.2022	Tiefendatengenerierung
11.02.2022	FAQ mit den Tutoren
17.02.2022	Klausur, 14-16 Uhr, N10+N11

Dynamic Time Warping (Dynamische Zeitnormierung)

- **Mustervergleich zweier Sequenzen mit variierenden Geschwindigkeit**
  - Abbildung auf möglichst ähnliche Merkmale durch zeitliche Streckung bzw. Stauchung der Zeitsequenzen

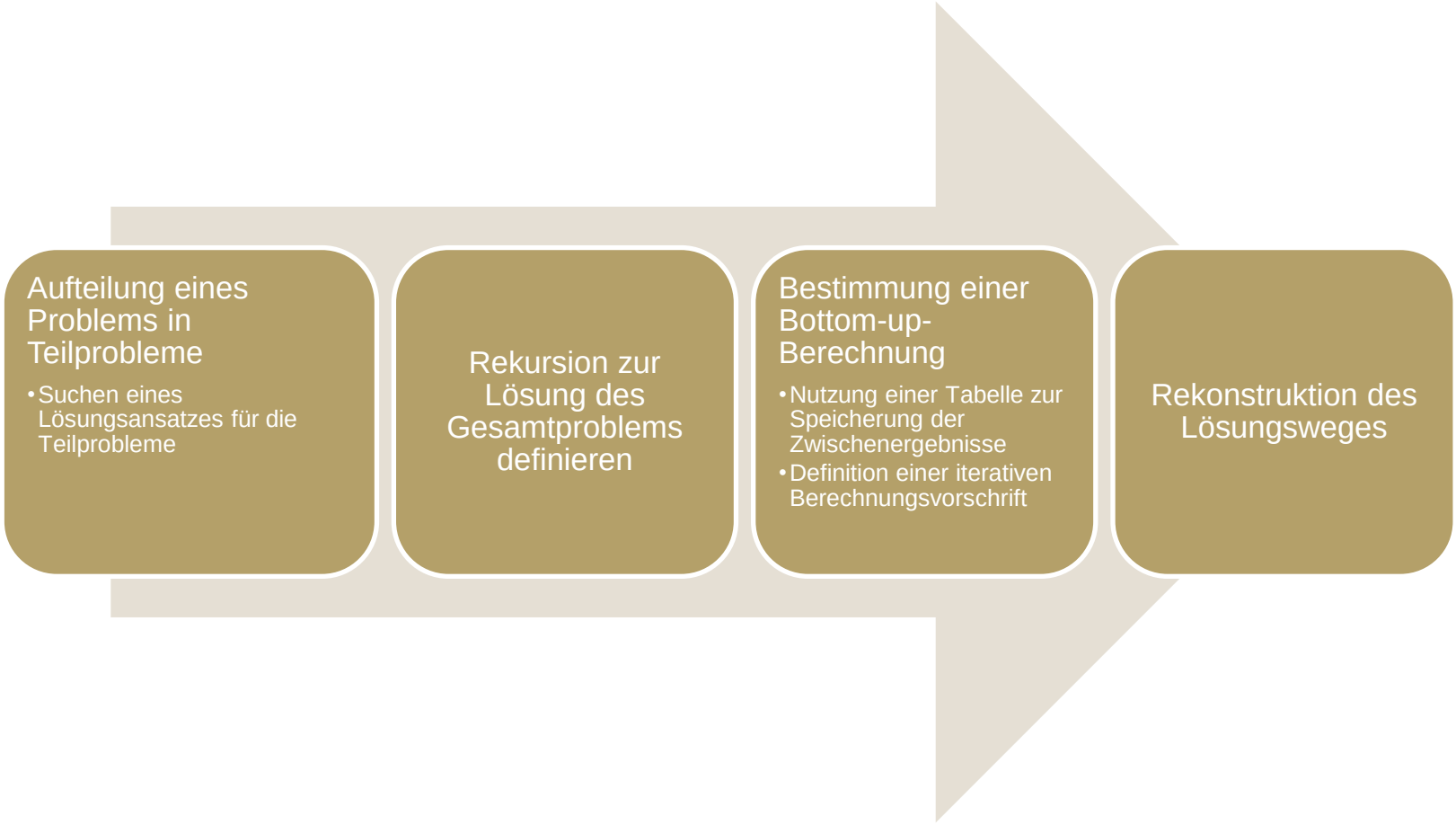
- Berechnung mittels **dynamischen Programmierens**

- **Viele Anwendungen**
  - Editierdistanzen
  - Synchronisierung von Musik
  - Spracherkennung
  - Gestenerkennung
  - Information Retrieval (Informationsrückgewinnung)
  - Data Mining
  - Bioinformatik (DNA-Alignment, RNA-Faltung)

Dynamisches Programmieren

- Bottom-up-Programmierparadigma
  - **Zerlegen** in geeignete **Teilprobleme**
  - **Lösung** der **Teilprobleme** und
  - **Wiederverwendung** zur **Berechnung** der **nächstgrößeren Teillösungen**
- **Ursprung**: Richard Bellman
  - Optimale Lösung eines Gesamtproblems setzt sich aus optimalen Teillösungen zusammen **(Bellmansches Optimalitätsprinzip)**
  - Systematisches Ausfüllen einer Tabelle zur Wiederverwendung von Teillösungen bezeichnet er als „Programmieren“
- Gut einsetzbar, wenn es viele überlappende Teillösungen gibt, die nicht erneut berechnet werden müssen

Herleitung eines generischen Algorithmus für dynamisches Programmieren



Beispiel: Editierdistanz (Levenshtein)

- **Editierdistanz (Levenshtein-Distanz)**: $d(wort_1, wort_2)$
  - Maß für den Unterschied zwischen zwei Zeichenketten, d.h. die
  - **minimale Anzahl** von **Einfüge-**, **Lösch-** und **Ersetz-Operationen**, um die erste Zeichenkette in die zweite umzuwandeln

- **Beispiel:** Editierdistanz zwischen Apfel und Pferd

→ Editieroperationen: A löschen, L durch R ersetzen, D einfügen

→ Editierdistanz *d*(„APFEL“, „PFERD“) = 3

Rekursive Definition der Editierdistanz

Editierdistanz $d(i,j)$ für Teilwörter $A = a_1, \dots, a_i$ und $B = b_1, \dots, b_j$

$$
\begin{aligned}
d(0,j) &= j \text{ (komplettes Wort erzeugen)} \\
d(i,0) &= i \text{ (komplettes Wort löschen)} \\
d(i,j) &= \min(\, d(i,j-1)+1, \text{ (Einfügen)} \\
&\qquad\quad d(i-1,j)+1, \text{ (Löschen)} \\
&\qquad\quad d(i-1,j-1)+1, \text{falls } a_i \neq b_j \text{ (Ersetzen)} \\
&\qquad\quad d(i-1,j-1)+0, \text{falls } a_i = b_j \text{ (Übernehmen)} \\
&\qquad\quad)
\end{aligned}
$$

- Naive Implementierung führt zu exponentieller Laufzeit!
- Aber: Es gibt nur $i \times j$ unterschiedliche Zwischenwerte

Aufstellung einer Tabelle von Zwischenwerten

- Randbereich: $d(0,j) = j$ und $d(i,0) = i$

L	5					
E	4					
F	3					
P	2					
A	1					
	0	1	2	3	4	5
		P	F	E	R	D

Aufstellung einer Tabelle von Zwischenwerten

Innenbereich: $d(i,j) = \min($ $d(i,j-1)+1,$ (Einfügen)

$d(i-1,j)+1,$ (Löschen)

$d(i-1,j-1)+1, \text{falls } a_i \neq b_j$ (Ersetzen)

$d(i-1,j-1)+0, \text{falls } a_i = b_j$ (Übernehmen)

$)$

L	5					
E	4					
F	3					
P	2					
A	1					
	0	1	2	3	4	5
		P	F	E	R	D

Aufstellung einer Tabelle von Zwischenwerten

Innenbereich: $d(i,j) = \min($ $d(i,j-1)+1,$ (Einfügen)

$d(i-1,j)+1,$ (Löschen)

$d(i-1,j-1)+1, \text{falls } a_i \neq b_j$ (Ersetzen)

$d(i-1,j-1)+0, \text{falls } a_i = b_j$ (Übernehmen)

$)$

L	5					
E	4					
F	3					
P	2					
A	1	1				
	0	1	2	3	4	5
		P	F	E	R	D

Aufstellung einer Tabelle von Zwischenwerten

Innenbereich: $d(i,j) = \min($ $d(i,j-1)+1,$ (Einfügen)
$d(i-1,j)+1,$ (Löschen)
$d(i-1,j-1)+1, \text{falls } a_i \neq b_j$ (Ersetzen)
$d(i-1,j-1)+0, \text{falls } a_i = b_j$ (Übernehmen)
$)$

L	5					
E	4					
F	3					
P	2					
A	1	1	2			
	0	1	2	3	4	5
		P	F	E	R	D

Aufstellung einer Tabelle von Zwischenwerten

Innenbereich: $d(i,j) = \min($ $d(i,j-1)+1,$ (Einfügen)
$d(i-1,j)+1,$ (Löschen)
$d(i-1,j-1)+1, \text{falls } a_i \neq b_j$ (Ersetzen)
$d(i-1,j-1)+0, \text{falls } a_i = b_j$ (Übernehmen)
$)$

L	5					
E	4					
F	3					
P	2					
A	1	1	2	3	4	5
	0	1	2	3	4	5
		P	F	E	R	D

Aufstellung einer Tabelle von Zwischenwerten

Innenbereich: $d(i,j) = \min($ $d(i,j-1)+1,$ (Einfügen)
$d(i-1,j)+1,$ (Löschen)
$d(i-1,j-1)+1, \text{falls } a_i \neq b_j$ (Ersetzen)
$d(i-1,j-1)+0, \text{falls } a_i = b_j$ (Übernehmen)
$)$

L	5					
E	4					
F	3					
P	2	1				
A	1	1	2	3	4	5
	0	1	2	3	4	5
		P	F	E	R	D

Aufstellung einer Tabelle von Zwischenwerten

Innenbereich: $d(i,j) = \min($ $d(i,j-1)+1,$ (Einfügen)
$d(i-1,j)+1,$ (Löschen)
$d(i-1,j-1)+1, \text{falls } a_i \neq b_j$ (Ersetzen)
$d(i-1,j-1)+0, \text{falls } a_i = b_j$ (Übernehmen)
$)$

L	5					
E	4					
F	3					
P	2	1	2	3	4	5
A	1	1	2	3	4	5
	0	1	2	3	4	5
		P	F	E	R	D

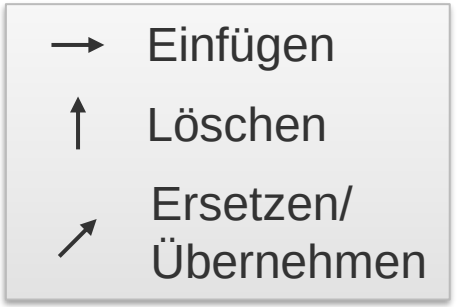
Aufstellung einer Tabelle von Zwischenwerten

Ergebnis: Editierdistanz $d(i,j) = 3$

L	5	4	3	2	2	3
E	4	3	2	1	2	3
F	3	2	1	2	3	4
P	2	1	2	3	4	5
A	1	1	2	3	4	5
	0	1	2	3	4	5
		P	F	E	R	D

Wie erhält man Editieroperationen?

- Rückwärts-Rekonstruktion ausgehend von $d(i,j)$
  - Jeweils **Minimum** der **3 angrenzenden Nachbarn suchen** und fortfahren
  - $min = d(i-1, j-1)$: Ersetzen/Übernehmen
  - $min = d(i-1, j)$: Löschen
  - $min = d(i, j-1)$: Einfügen



	A löschen	P überneh.	F überneh.	E überneh.	L → R	D einfügen
L	5	4	3	2	2	3
E	4	3	2	1	2	3
F	3	2	1	2	3	4
P	2	1	2	3	4	5
A	1	1	2	3	4	5
	0	1	2	3	4	5
		P	F	E	R	D

Wie erhält man Editieroperationen? Anderer Rückwärtspfad

- Rückwärts-Rekonstruktion ausgehend von $d(i,j)$
  - Jeweils **Minimum** der **3 angrenzenden Nachbarn suchen** und fortfahren
  - $min = d(i-1, j-1)$: Ersetzen/Übernehmen
  - $min = d(i-1, j)$: Löschen
  - $min = d(i, j-1)$: Einfügen

→ Einfügen
↑ Löschen
↗ Ersetzen/ Übernehmen

	A löschen	P überneh.	F überneh.	E überneh.	R einfügen	L → D
L	5	4	3	2	2	3
E	4	3	2	1	2	3
F	3	2	1	2	3	4
P	2	1	2	3	4	5
A	1	1	2	3	4	5
	0	1	2	3	4	5
		P	F	E	R	D

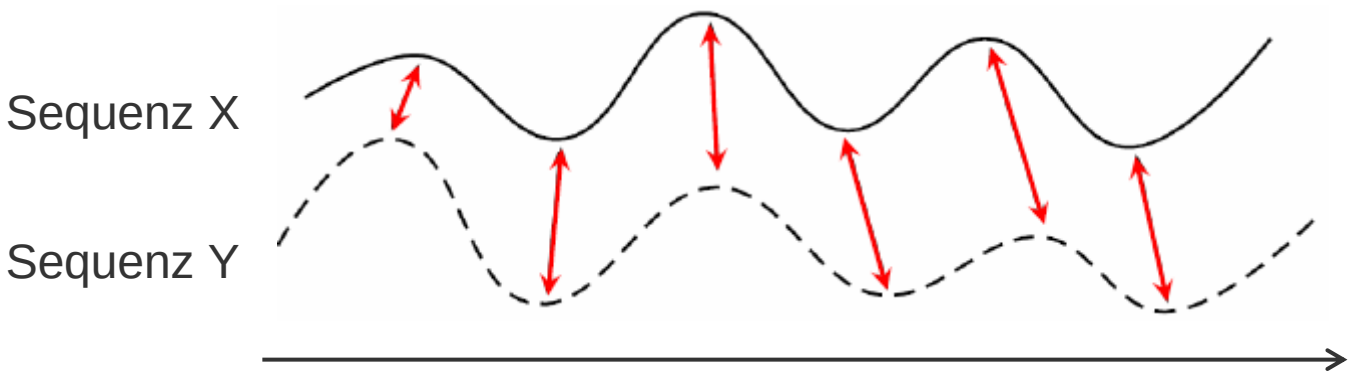
Rekursive Definition der Editierdistanz mit a-Umlaut

Editierdistanz $d(i,j)$ für Teilwörter $A = a_1, \ldots, a_i$ und $B = b_1, \ldots, b_j$

$$d(0,j) = j \text{ (komplettes Wort erzeugen)}$$
$$d(i,0) = i \text{ (komplettes Wort löschen)}$$
$$\begin{aligned} d(i,j) = \min(\; & d(i,j-1) + 1, && \text{(Einfügen)} \\ & d(i-1,j) + 1, && \text{(Löschen)} \\ & d(i-1,j-1) + 1, \text{falls } a_i \neq b_j && \text{(Ersetzen)} \\ & d(i-1,j-1) + 0, \text{falls } a_i = b_j && \text{(Übernehmen)} \\ & d(i-1,j-2) + 0, \text{falls } a_i = \text{"ä"} \;\&\; b_{j-1} = \text{"a"} \;\&\; b_j = \text{"e"} \\ ) & \end{aligned}$$

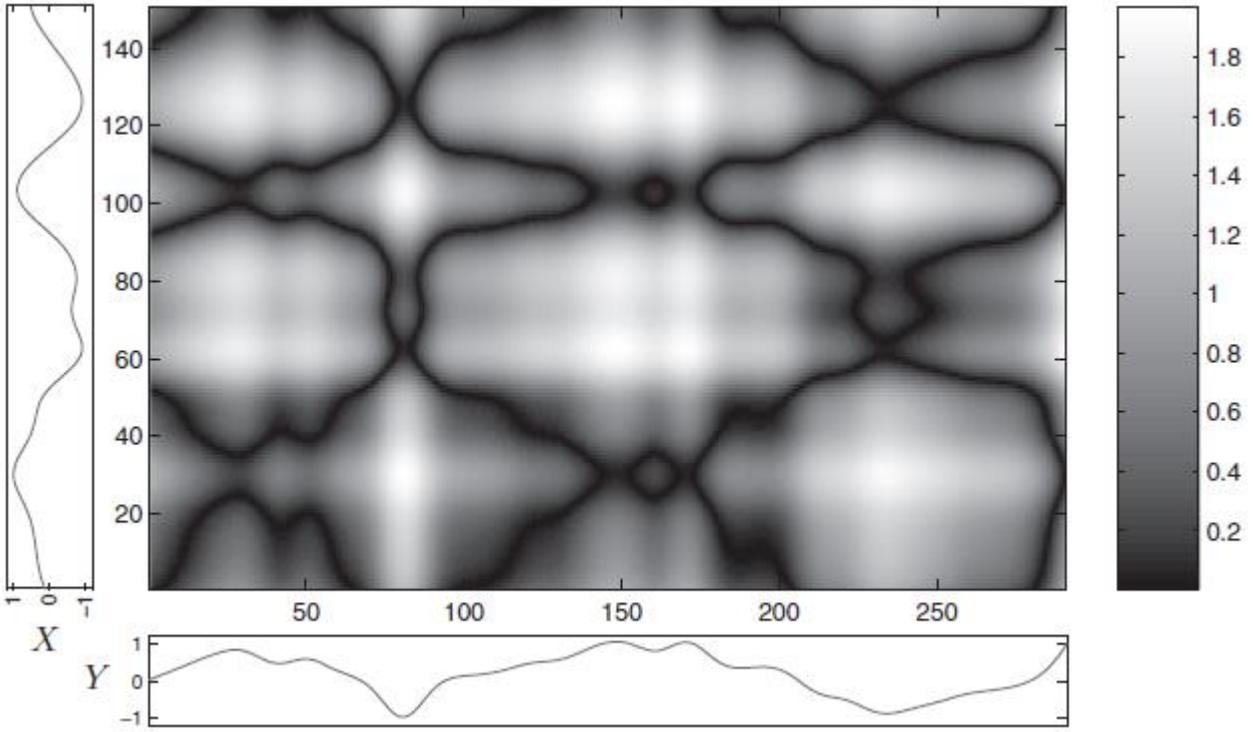
Dynamic Time Warping

- Anpassen bzw. Ausrichtung (engl. Alignment) zeitlicher oder geometrischer Sequenzen



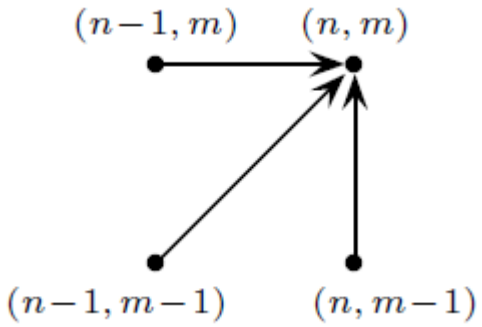
Kostenmatrix zweier reellwertiger Sequenzen X, Y

- Kostenmatrix $C^{n\times m} := c(x_i, y_j) = |x_i - y_j|$

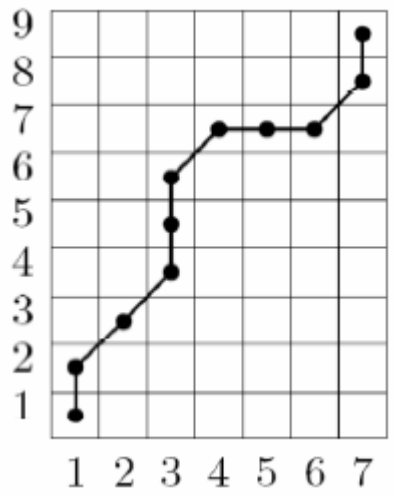


Warping-Pfad

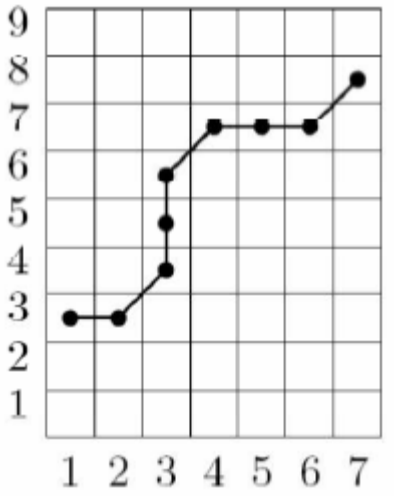
- Ein **Anpassungs- oder Ausrichtungspfad (engl. warping path)** beschreibt eine **„Abbildung“ zweier Sequenzen aufeinander**, so dass gilt:
  - **Randbedingung (Boundary-Bedingung)**
    - Vollständigkeit (definiert an beiden Randpunkten der Sequenzen)
  - **Monotoniebedingung**
  - **Schrittweitenbedingung**



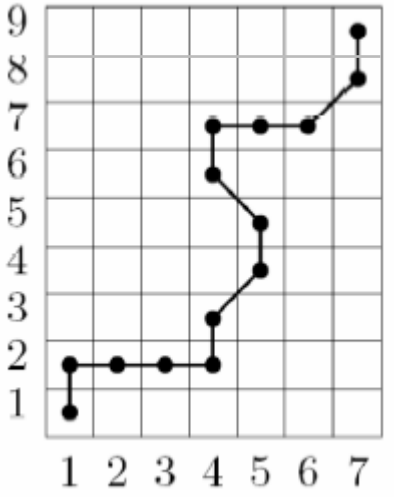
Warping-Pfad (Beispiele und Gegenbeispiele)



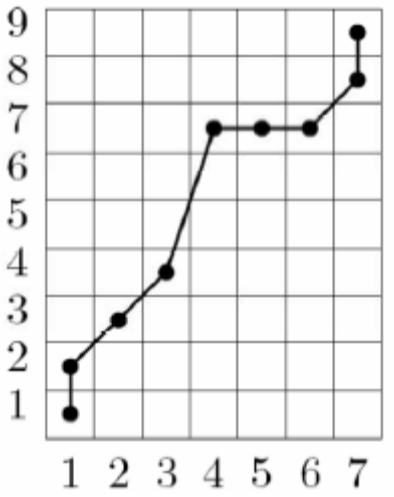
alle Bedingungen erfüllt!



Boundary-Bedingung verletzt!



Monotoniebedingung verletzt!



Schrittweiten-bedingung verletzt!

Optimaler Warping-Pfad

- Gesamtkosten eines Warping-Pfad $p = \{(x_{n_1}, x_{m_1}), \dots, (x_{n_L}, x_{m_L})\}$

$$c_p(X, Y) := \sum_{\ell=1}^{L} c(x_{n_\ell}, y_{m_\ell})$$

- Ein **optimaler Warping-Pfad p^*** zwischen X und Y wird als DTW-Distanz bezeichnet und berechnet sich durch

$$\begin{aligned} \mathrm{DTW}(X, Y) &:= c_{p^*}(X, Y) \\ &= \min\{c_p(X, Y) \mid p \text{ is an } (N, M)\text{-warping path}\} \end{aligned}$$

Berechnung des optimalen Warping-Pfads mit dynamischen Programmieren

- Aufteilung in Teilprobleme (rekursiv mit dem Ziel einer Bottom-up-Berechnung)

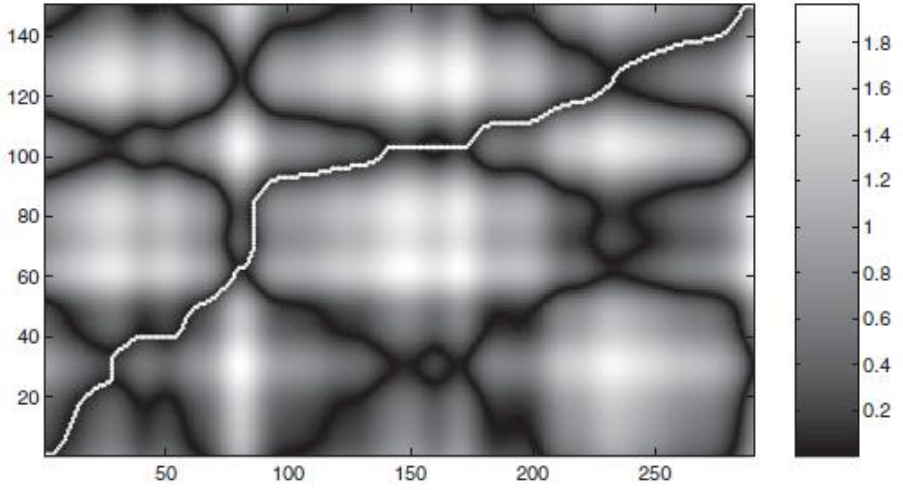
  $$d(i,j) = \min(\, d(i,j-1), d(i-1,j), d(i-1,j-1)\,) + c\left(x_i, y_j\right)$$

- Setzen der Randwerte:

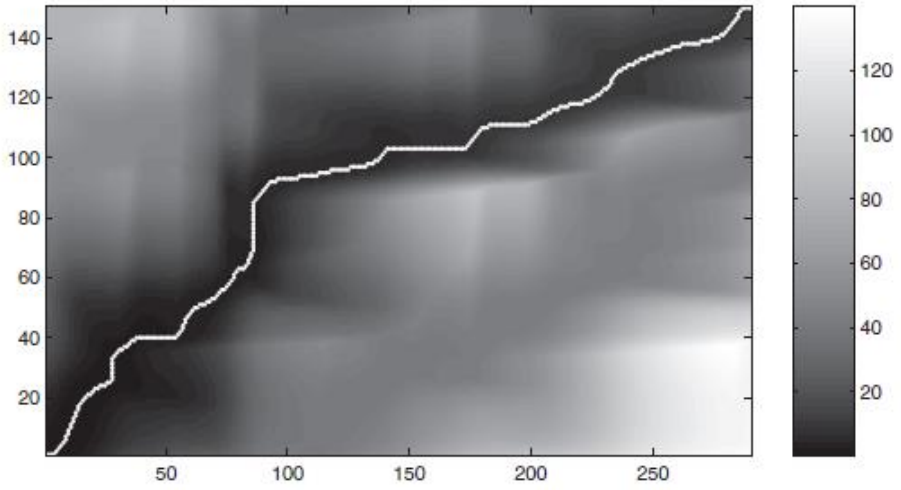
  $$d(0,0) = 0, d(n,0) = \infty,\ d(0,m) = \infty$$

- Speicherung der Zwischenergebnisse
  - Verwendung der obigen Berechnungsvorschrift zur sukzessiven Berechnung einer akkumulierten Kostenmatrix D

Optimaler Warping-Pfad mit Kostenmatrix und akkumulierter Kostenmatrix



Warping-Pfad mit Kostenmatrix



Warping-Pfad mit akkumulierter Kostenmatrix

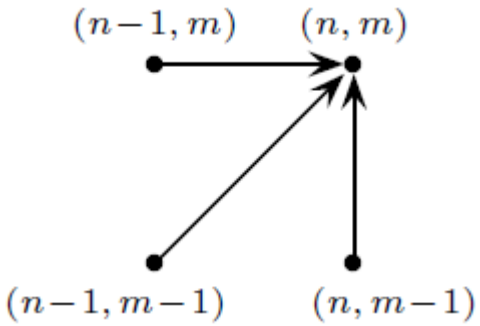
Berechnung des optimalen Warping-Pfads

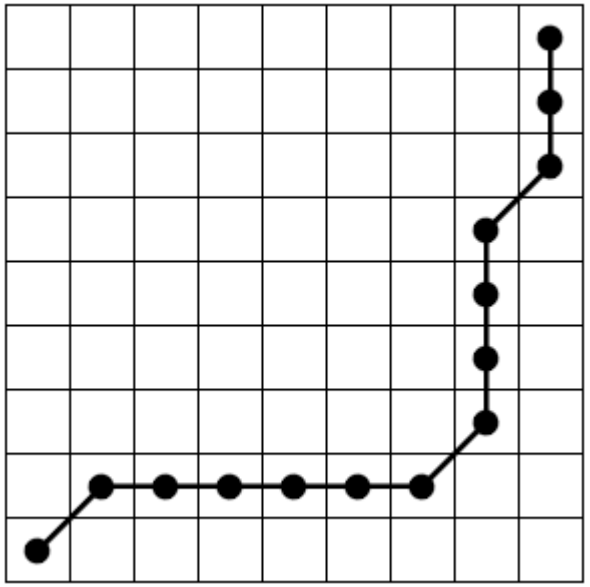
- **Eingabe:** Akkumulierte Kostenmatrix D
- **Ausgabe:** Optimaler Warping-Pfad $p = \{(x_{n_1}, x_{m_1}), \dots, (x_{n_L}, x_{m_L})\}$

- **Algorithmus:** Berechne optimalen Warping-Pfad rückwärts beginnend mit $(x_{n_l}, x_{m_l}) = (x_{n_L}, x_{m_L})$

$$p_{l-1} = \begin{cases} (1, m_L - 1), \text{falls } n_L = 1 \\ (n_L - 1, 1), \text{falls } m_L = 1 \\ argmin(\, d(n_L, m_L - 1), d(n_L - 1, m_L), d(n_L - 1, m_L - 1), \text{sonst} \end{cases}$$

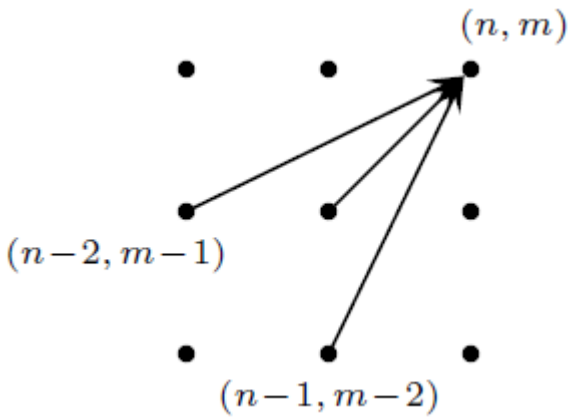
Standard Schrittweitenbedingung

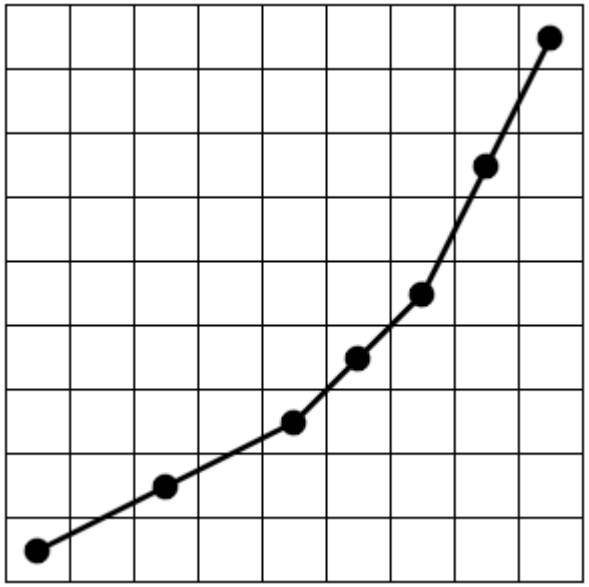




- **Problem:** kann in lange Abschnitte mit Steigung 1 oder 0 degenerieren

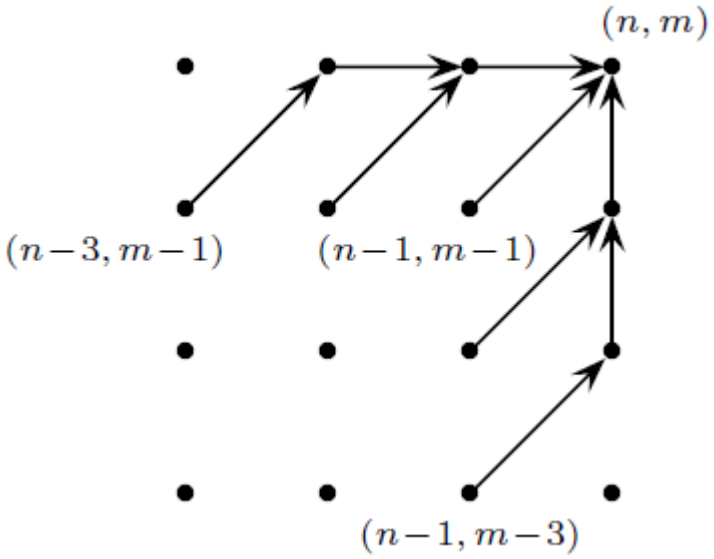
Alternative Schrittweitenbedingungen

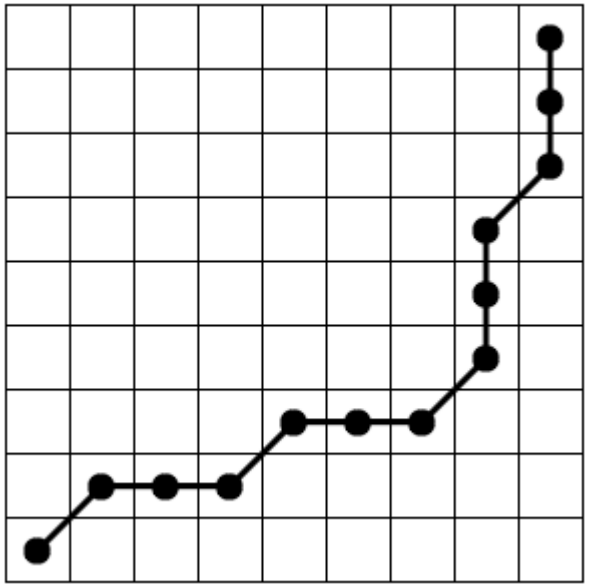




- **Problem:** Sequenzpunkte können übersprungen (ausgelassen) werden

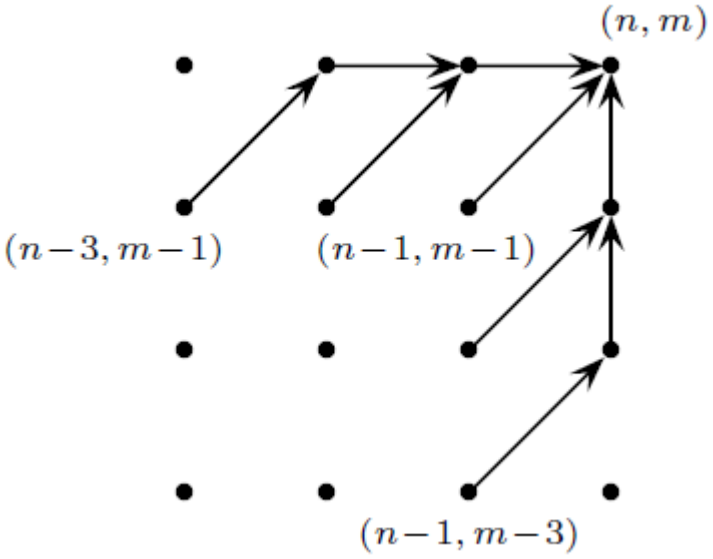
Alternative Schrittweitenbedingungen

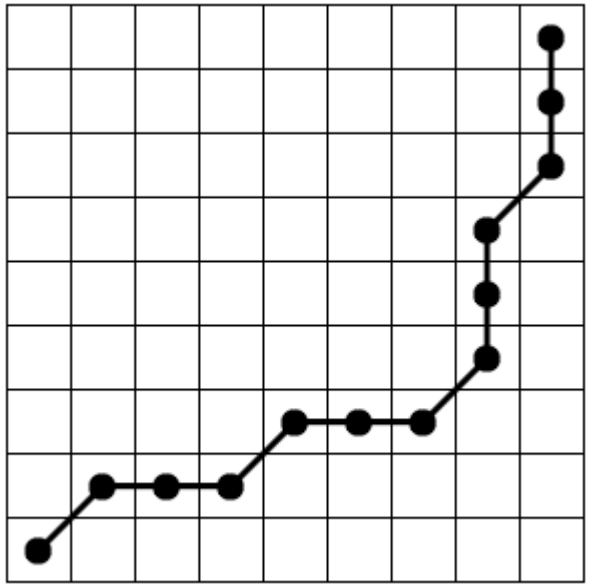




- **Vorteile**
  - Maximale Steigung ergibt sich aus der Definition der Schrittweitenbeschränkung
  - Keine ausgelassenen Sequenzpunkte

Berechnung mit maximaler Schrittweitenbeschränkung





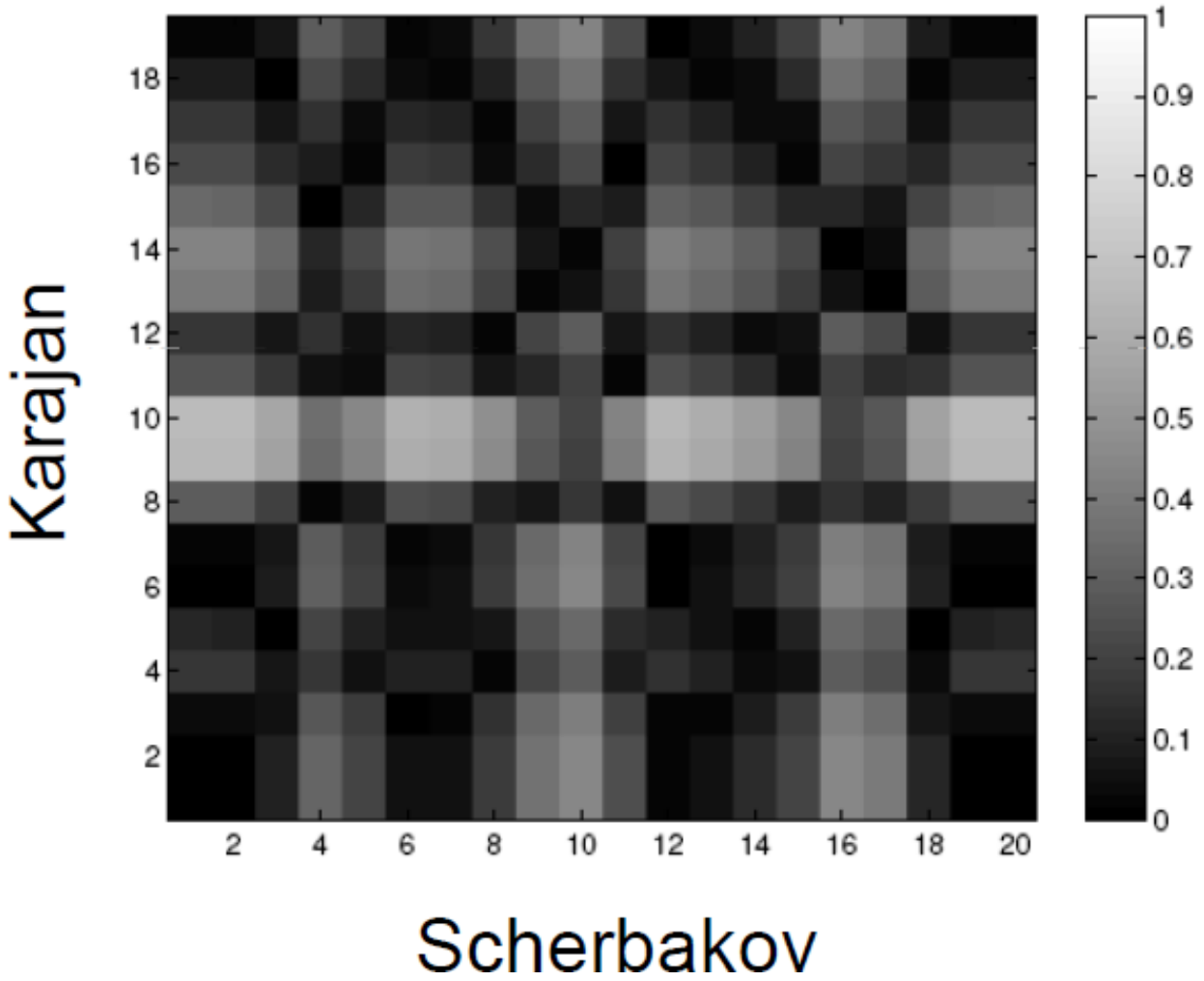
$$D(n,m) = \min \begin{cases} D(n-1,m-1) + c(x_n, y_m) \\ D(n-2,m-1) + c(x_{n-1}, y_m) + c(x_n, y_m) \\ D(n-1,m-2) + c(x_n, y_{m-1}) + c(x_n, y_m) \\ D(n-3,m-1) + c(x_{n-2}, y_m) + c(x_{n-1}, y_m) + c(x_n, y_m) \\ D(n-1,m-3) + c(x_n, y_{m-2}) + c(x_n, y_{m-1}) + c(x_n, y_m) \end{cases}$$

Gewinnung von Audiomerkmalen

- **Spektrogramm**
  - Darstellung des **zeitlichen Verlaufes** des **Frequenzspektrums** eines Signals
  - **Achtung:**
    - Fourier-Transformation stellt keine Informationen über das zeitliche Auftreten von Frequenzanteilen eines Signals bereit
    - funktioniert nur bei Verwendung eines zeitlichen Signalfensters, d.h. der Multiplikation des Signals mit einer Fensterfunktion, das überlappend über das Eingangssignal verschoben wird
      → Anwendung der Kurzzeit-Fourier-Transformation: liefert pro Fenster ein Frequenzspektrum
- **Formant**
  - **Konzentration akustischer Energie** in einem bestimmten **Frequenzbereich**, entstehen z.B. in den Resonanzspektren der Musikinstrumente oder der menschlichen Stimme

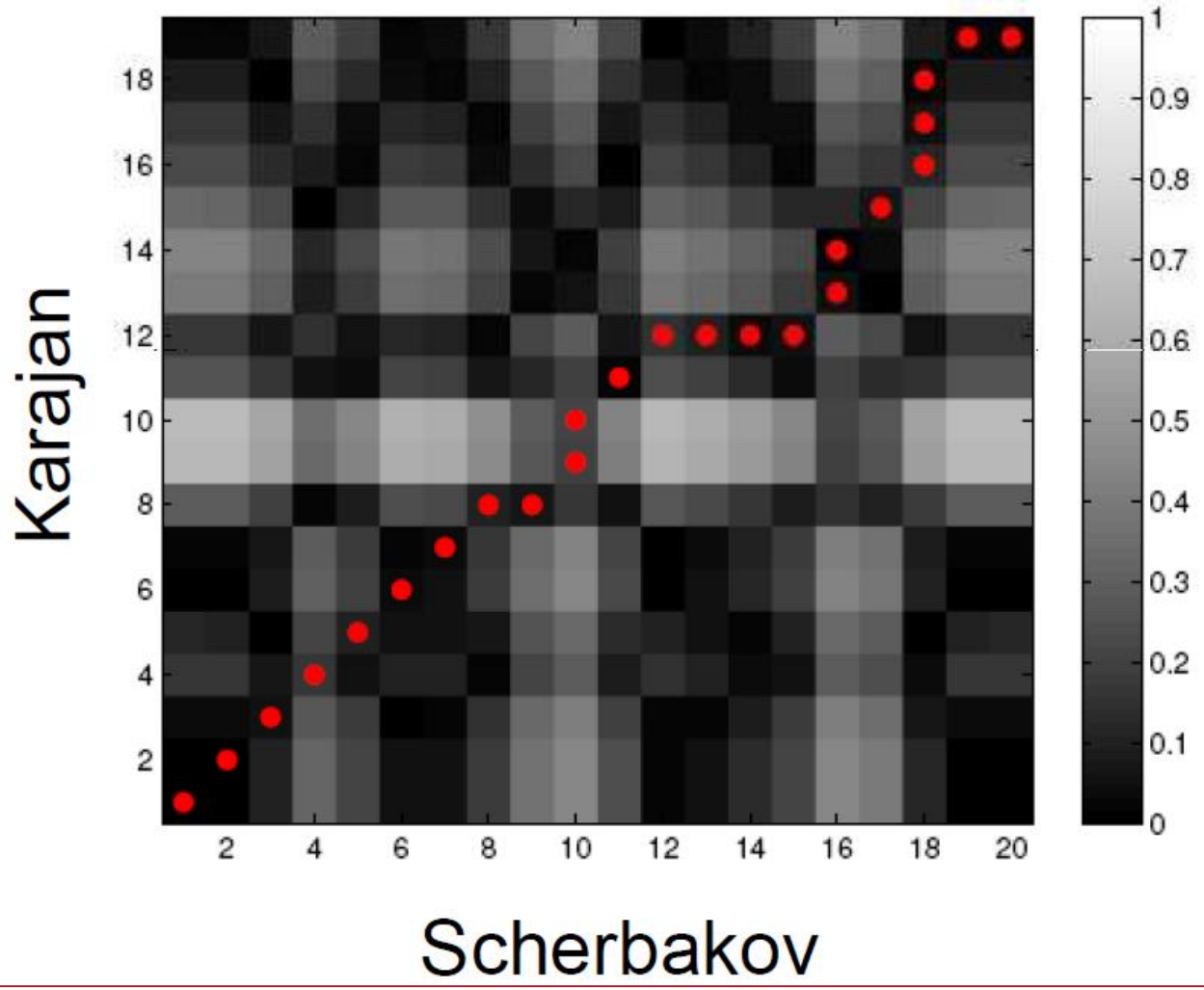
Musiksynchronisation
Audio-Audio

- Aufwand (Kostenmatrix)

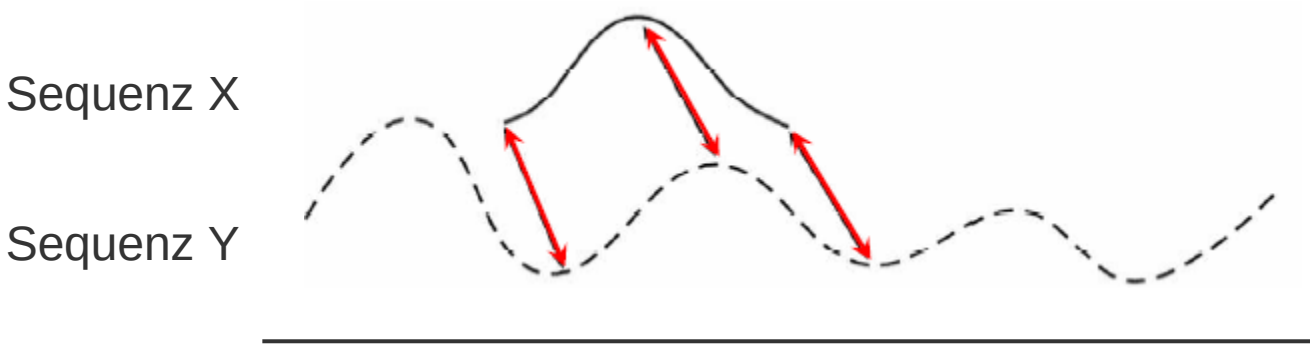


Musiksynchronisation Audio-Audio

- Kostenminimierung (Cost-Minimizing Warping Path)



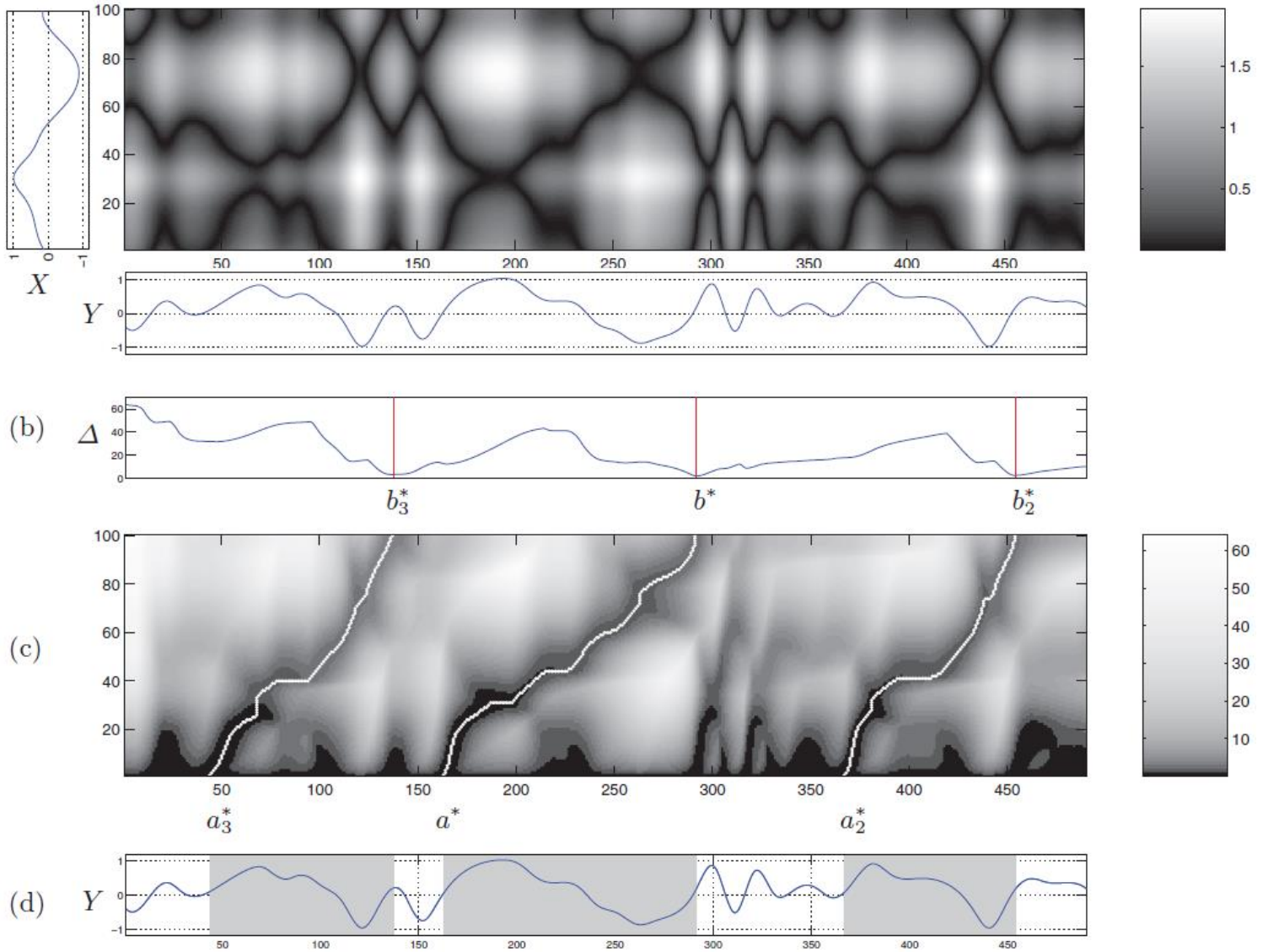
DTW über Teilsequenzen



- Berechnung:

$$(a^*, b^*) := \underset{(a,b)\,:\,1 \le a \le b \le M}{\operatorname{argmin}} \Big(\mathrm{DTW}\big(X\,,\, Y(a:b)\big) \Big)$$

DTW über Teilsequenzen

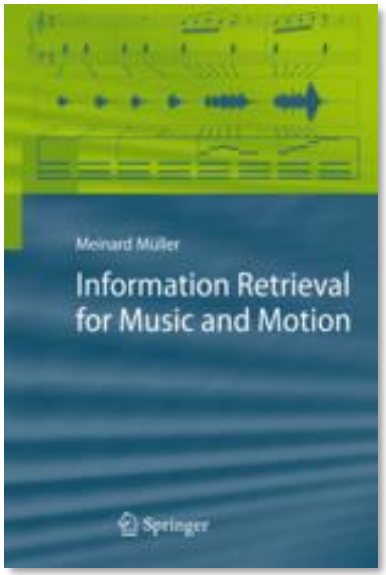


Zusammenfassung

- Dynamic Time Warping ist ein universeller Algorithmus
  - Editierdistanzen
  - Synchronisierung von Musik
  - Spracherkennung
  - Gestenerkennung
  - Information Retrieval (Informationsrückgewinnung)

- Voraussetzung für eine Anwendung
  - Identifikation geeigneter Merkmale
  - Definition eines Kostenmaßes
  - Definition einer Schrittweitenbedingung zur Begrenzung des Warping-Pfads
  - Lösung durch Rückwärts-Traversierung des Warping-Pfads

Literatur



M. Müller:

Information Retrieval for Music and Motion,

Springer-Verlag, 2007.

R. Steinmetz:

Multimedia-Technologie,

Springer-Verlag,
3. Auflage, 2000.

Quellenangabe: Bilder und Folienmaterial sind auszugsweise aus Vorlesungsmaterialien von M. Müller und R. Steinmetz entnommen.