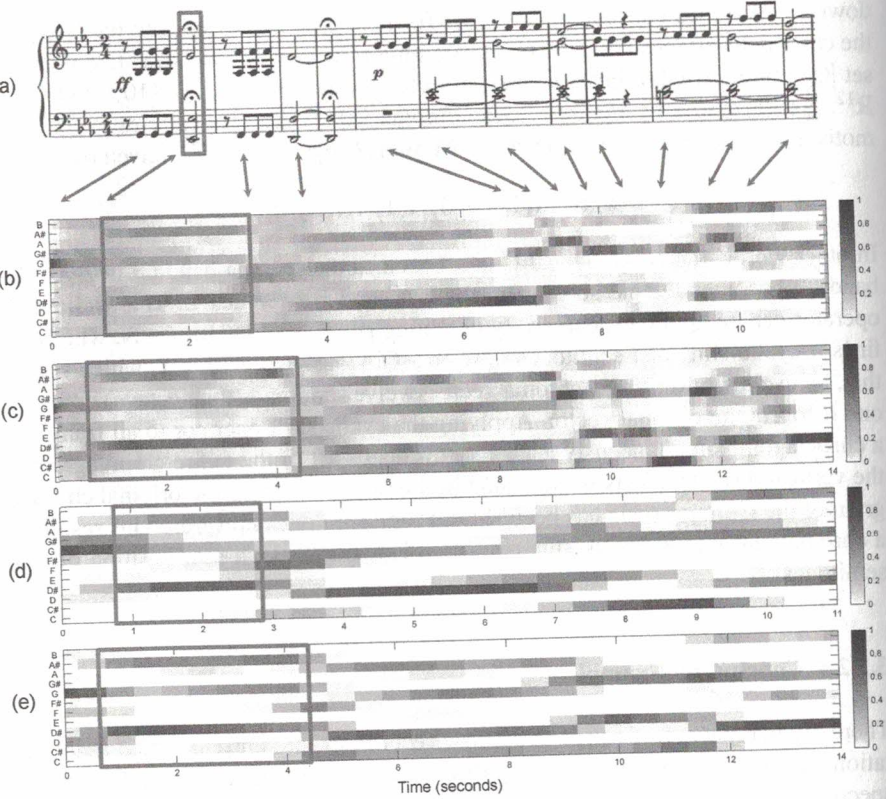

Fig. 3.9 Chroma representations for two different performances of Beethoven's Fifth Symphony. **(a)** Sheet music representation (in a piano reduced version). **(b)** Compressed and normalized chromagram for an orchestra performance. **(c)** Compressed and normalized chromagram for a piano performance. **(d)** Enhanced chromagram for an orchestra performance. **(e)** Enhanced chromagram for a piano performance.

In conclusion, we have seen that chroma features are a suitable representation for harmonic music with a broad class of pitch and tuning characteristics. Chroma representations closely correlate to the aspect of harmony, while showing a high degree of robustness to changes in timbre and dynamics. One important message of this section is that there are many ways of computing and enhancing chroma features resulting in a large number of chroma variants with different properties. We will see that there is no "best" chroma variant that performs equally well for all applications in mind. Therefore, in order to be successful, one needs to have a good understanding of both the feature design step as well as the requirements of the given application scenario.

3.2 Dynamic Time Warping

We have seen how different music representations can be made comparable by converting them into suitable feature representations. Next, we study how these feature representations can be aligned or synchronized to bring them into temporal correspondence. To this end, we introduce an important technique that is known as **dynamic time warping** (DTW).

The objective of DTW is to compare two given sequences. In our music synchronization scenario, these two sequences are, for example, chroma representations of two different versions of the same piece of music. Let us denote the first sequence by $X = (x_1, x_2, \ldots, x_N)$ with $N \in \mathbb{N}$ being the length of the sequence and the second sequence by $Y = (y_1, y_2, \ldots, y_M)$ having length $M \in \mathbb{N}$, where the elements x_n and y_m of the two sequences are chroma vectors. For example, X may be a chroma vector sequence obtained from an orchestral version of Beethoven's Fifth (as in Figure 3.9d) and Y a sequence obtained from a piano version (Figure 3.9e). Since the orchestral performance underlying the sequence X is played faster than the piano performance underlying the sequence Y, the two sequences do not have the same length even though they musically correspond to each other. In our example, as illustrated by Figure 3.10, the sequence X has length $N = 12$, whereas the sequence Y has length $M = 15$. The goal of DTW is to compensate for differences in tempo by finding a possibly nonlinear alignment between the elements of the two sequences. Intuitively speaking, this can be achieved by either skipping certain elements of a sequence or by using certain elements more than once. For example, in the alignment shown in Figure 3.10, the element x_3 is assigned to the two elements y_3 and y_4. Similarly, the elements x_5 and x_{12} are used twice in the overall alignment.

The general goal of DTW is to find an optimal alignment between two given (time-dependent) sequences under certain restrictions. Based on this alignment, the sequences can be warped in a nonlinear fashion to match each other. Originally, DTW was used to compare different speech patterns in automatic speech recognition. Closely related to concepts such as the edit distance or longest common subsequence, DTW-like procedures are now widely used in various fields such as data mining, information retrieval, and bioinformatics.

In this section, we introduce the main ideas of classical DTW as well as an efficient algorithm based on dynamic programming to compute an optimal alignment (Section 3.2.1). Then, we discuss several modifications to DTW which make it possible to influence certain local and global properties of the alignment and to further speed up the DTW computation (Section 3.2.2). A number of related algorithms and DTW variants are also discussed in the exercises and in subsequent chapters.

3.2.1 Basic Approach

As said above, the objective of DTW is to compare two sequences $X := (x_1, x_2, \ldots, x_N)$ of length $N \in \mathbb{N}$ and $Y := (y_1, y_2, \ldots, y_M)$ of length $M \in \mathbb{N}$. Going

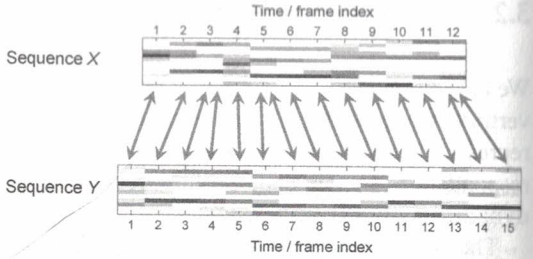

Fig. 3.10 Time alignment of two time-dependent or indexed sequences of feature vectors. Aligned points or frames are indicated by the arrows.

beyond the music synchronization scenario, these sequences may be discrete signals, feature sequences, sequences of characters, or any kind of time series. Often the indices of the sequences correspond to successive points in time that are spaced at uniform time intervals. In the following, we fix a **feature space** denoted by $\mathcal{F}$ and assume $x_n, y_m \in \mathcal{F}$ for $n \in [1:N]$ and $m \in [1:M]$. To compare two different features $x, y \in \mathcal{F}$, one needs a **local cost measure**, sometimes also referred to as a **local distance measure**, which is defined to be a function

$$c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}. \tag{3.12}$$

Typically, $c(x,y)$ is small (low cost) if x and y are similar to each other, and otherwise $c(x,y)$ is large (high cost). Evaluating the local cost measure for each pair of elements of the sequences X and Y, one obtains a **cost matrix** $\mathbf{C} \in \mathbb{R}^{N \times M}$ defined by

$$\mathbf{C}(n,m) := c(x_n, y_m) \tag{3.13}$$

for $n \in [1:N]$ and $m \in [1:M]$. In the following, a tuple (n,m) representing an entry of the matrix $\mathbf{C}$ will be referred to as a **cell** of the matrix.

Let us come back to our example from Figure 3.10. Using a sequence of twelve-dimensional chroma vectors, the feature space is $\mathcal{F} = \mathbb{R}^{12}$. There are many ways to define a distance between two elements $x, y \in \mathcal{F}$. Maybe the most well-known distance is the Euclidean distance defined by $\|x - y\|$ using the Euclidean norm of $\mathbb{R}^{12}$. Another distance is referred to as the **cosine distance**, which we use as the local cost measure c in the subsequent examples. The cosine distance between two nonzero vectors $x, y \in \mathcal{F}$ is defined by

$$c(x,y) := 1 - \frac{\langle x|y \rangle}{\|x\| \cdot \|y\|}, \tag{3.14}$$

and $c(x,y) := 0$ if either x or y is zero. Recall from (2.39) that the quotient of the inner product and the product of the norms is simply the cosine of the angle between the two vectors x and y. Therefore, $c(x,y) \in [0,1]$, $c(x,y) = 0$ in the case that x and y point in the same direction, and $c(x,y) = 1$ in the case that x and y are orthogonal. Note that, as opposed to the Euclidean distance, the cosine distance does not depend on the actual length of the vectors. Therefore, when comparing chroma vectors, the measure only considers the energy distributions across the twelve chroma bands and

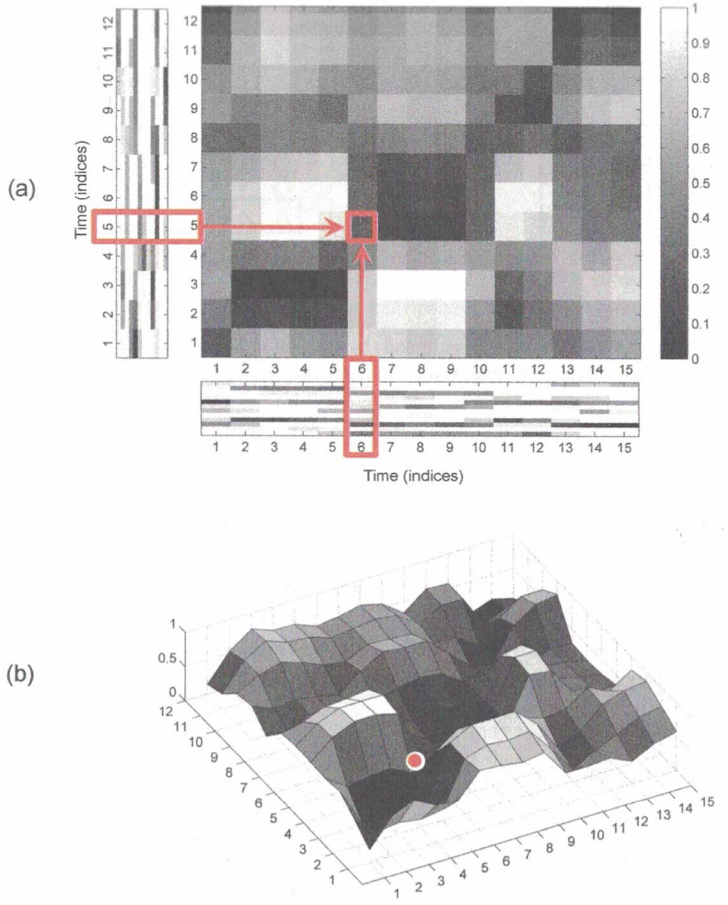

Fig. 3.11 **(a)** Cost matrix of the two chroma sequences X (vertical axis) and Y (horizontal axis) using the cosine distance (3.14) as the local cost measure c. Regions of low cost are indicated by dark colors, and regions of high cost are indicated by light colors. **(b)** 3D surface plot of the cost matrix.

disregards the actual local energy. In our music synchronization application, this property is desirable when the two versions to be compared may differ significantly in dynamics (which, for example, may be the case when comparing an orchestral and piano version of a piece of music). Furthermore, there is another practical reason why the cosine distance is beneficial: the computation of an entire cost matrix based on the cosine distance can be done efficiently using a simple matrix multiplication (see Exercise 3.16).

The cost matrix $\mathbf{C}$ obtained from our example is shown in Figure 3.11a. In this visualization, cells of low cost are depicted in dark colors and cells of high cost in light colors. Since the two sequences show a similar overall progression, except for

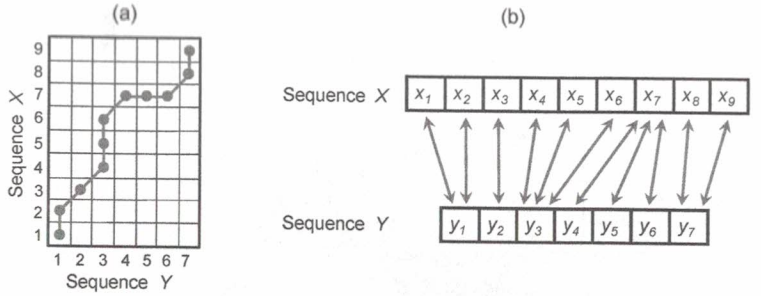

Fig. 3.12 **(a)** Illustration of a warping path and **(b)** its interpretation for some sequence X of length $N = 9$ and some sequence Y of length $M = 7$. Each cell (n,m) belonging to the warping path is indicated by a red dot and corresponds to an alignment between the elements x_n and y_m indicated by a red bidirectional arrow.

a global difference in tempo, the cost matrix has low values along the diagonal of the matrix. For example, the cell $(n,m) = (5,6)$, which indicates the distance between the vectors x_5 and y_6, has a small cost value. Now, the goal is to find an alignment between X and Y having minimal overall cost. Intuitively, such an optimal alignment runs along a "valley" of low cost within the cost matrix $\mathbf{C}$ (see Figure 3.11b for an illustration). Next, we formalize the notion of an alignment.

3.2.1.1 Warping Path

Given two sequences $X = (x_1, x_2, \ldots, x_N)$ and $Y = (y_1, y_2, \ldots, y_M)$, we have seen that a correspondence between two elements x_n and y_m can be modeled by the index pair or cell (n,m). In Figure 3.10, such a correspondence was indicated by a red bidirectional arrow. Therefore, to model a global alignment between the elements of the sequences X and Y, the idea is to consider a sequence of index pairs that fulfills certain constraints. This leads to the following definition: An (N,M)**-warping path** of length $L \in \mathbb{N}$ is a sequence

$$P = (p_1, \ldots, p_L) \tag{3.15}$$

with $p_\ell = (n_\ell, m_\ell) \in [1:N] \times [1:M]$ for $\ell \in [1:L]$ satisfying the following three conditions:

Boundary condition: $p_1 = (1,1)$ and $p_L = (N,M)$. (3.16)

Monotonicity condition: $n_1 \leq n_2 \leq \ldots \leq n_L$ and $m_1 \leq m_2 \leq \ldots \leq m_L$. (3.17)

Step size condition: $p_{\ell+1} - p_\ell \in \{(1,0),(0,1),(1,1)\}$ for $\ell \in [1:L-1]$. (3.18)

An (N,M)-warping path $P = (p_1, \ldots, p_L)$ defines an alignment between two sequences $X = (x_1, x_2, \ldots, x_N)$ and $Y = (y_1, y_2, \ldots, y_M)$ by assigning the element x_{n_ℓ}

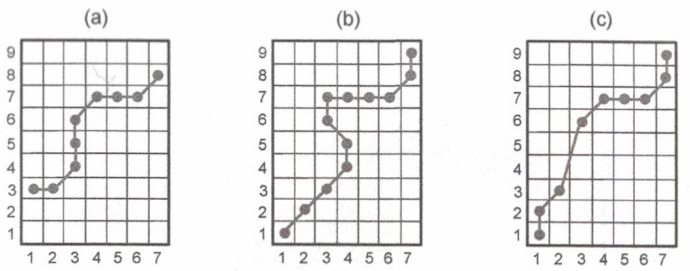

Fig. 3.13 Illustration of violations of the warping path conditions. **(a)** Violation of boundary condition. **(b)** Violation of monotonicity condition. **(c)** Violation of step size condition.

of X to the element y_{m_ℓ} of Y (see Figure 3.12 for an illustration). The boundary condition enforces that the first elements of X and Y as well as the last elements of X and Y are aligned to each other. In other words, the alignment refers to the entire sequences X and Y. The monotonicity condition reflects the requirement of faithful timing: if an element in X precedes a second element in X, then this should also hold for the corresponding elements in Y, and vice versa. Finally, the step size condition with respect to the set

$$\Sigma := \{(1,0),(0,1),(1,1)\} \tag{3.19}$$

expresses a kind of continuity condition: no element in X and Y can be omitted, and there are no replications in the alignment (in the sense that all index pairs contained in a warping path P are pairwise distinct). Note that the step size condition (3.18) implies the monotonicity condition (3.17), which nevertheless has been quoted explicitly for the sake of clarity. Figure 3.13 illustrates the conditions by some examples where the conditions are violated. In the following, if N and M are clear from the context, we simply speak of a warping path instead of an (N,M)-warping path.

3.2.1.2 Optimal Warping Path and DTW Distance

So far, in the definition of a warping path, the cost matrix does not play any role. A warping path simply encodes how to run through certain cells of a matrix with N rows and M columns starting with cell $(1,1)$ and ending with cell (N,M), while satisfying some monotonicity and step size conditions. Next, we introduce a notion that tells us something about the **quality** of a warping path. The **total cost** $c_P(X,Y)$ of a warping path P between two sequences X and Y with respect to the local cost measure c is defined as

$$c_P(X,Y) := \sum_{\ell=1}^{L} c(x_{n_\ell}, y_{m_\ell}) = \sum_{\ell=1}^{L} \mathbf{C}(n_\ell, m_\ell). \tag{3.20}$$

The intuition of this definition is that the warping path accumulates the cost of all cells it runs through. A warping path is "good" if its total cost is low, and it is "bad"

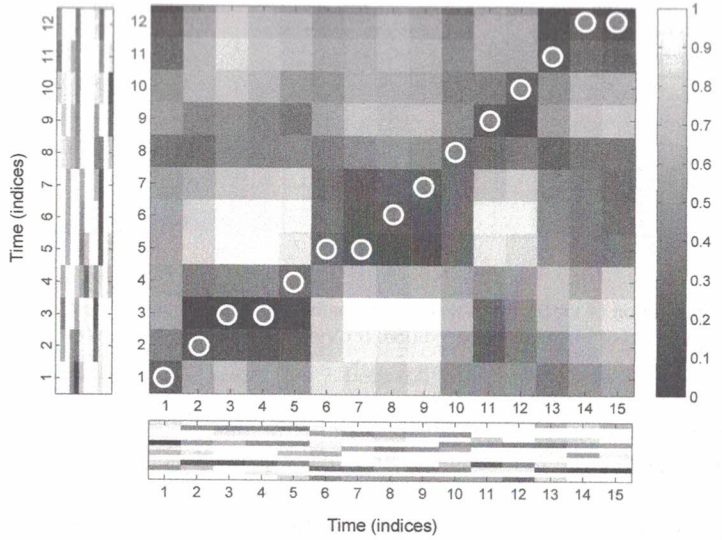

Fig. 3.14 Cost matrix from Figure 3.11 with an optimal warping path.

if its total cost is high. Now, we are interested in an **optimal warping path** between X and Y, which is defined to be a warping path P^* that has minimal total cost among all possible warping paths. Such an optimal warping path is shown in Figure 3.14 continuing the example of Figure 3.11. The cells of this warping path encode an overall optimal alignment between the chroma vectors of the two sequences, where the warping path conditions ensure that each element of sequence X is assigned to at least one element of Y and vice versa.

This leads us to the definition of the **DTW distance** denoted as $\mathrm{DTW}(X,Y)$ between the two sequences X of length N and Y of length M, which is defined as the total cost of an optimal (N,M)-warping path P^*:

$$\begin{aligned}\mathrm{DTW}(X,Y) &:= c_{P^*}(X,Y) \\ &= \min\{c_P(X,Y) \mid P \text{ is an } (N,M)\text{-warping path}\}.\end{aligned} \tag{3.21}$$

Note that in general there may exist more than one optimal warping path. For example, in the case that the cost matrix $\mathbf{C}$ is an all-zero matrix, every warping path is optimal, having a total cost of zero. Nevertheless, the DTW distance is well defined since all optimal warping paths obviously have the same total cost.

The number $\mathrm{DTW}(X,Y)$ defined in (3.21) is commonly referred to as the "DTW distance" between the sequences X and Y. However, from a mathematical point of view, the term "distance" is misused in this case. In mathematics, a **distance** is required to satisfy certain conditions, being symmetric, positive definite, and satisfying the triangle inequality. It is not hard to see that the DTW distance is symmetric, i.e., $\mathrm{DTW}(X,Y) = \mathrm{DTW}(Y,X)$, in case that the local cost measure c is symmetric (see Exercise 3.7). However, the DTW distance is in general not positive definite, where one requires that the distance between two elements is zero if and only if the elements are the same. For example, one obtains $\mathrm{DTW}(X,Y) = 0$ for the two different sequences $X := (x_1,x_2)$ and $Y := (x_1,x_1,x_2,x_2,x_2)$ in the case that $c(x_1,x_1) = c(x_2,x_2) = 0$. Intuitively, this property means that warping can be done without causing any cost. Even more surprising is the fact that the DTW distance generally does not satisfy the triangle inequality even if this holds for c. This fact will be illustrated by an example in Exercise 3.11.

3.2.1.3 Dynamic Programming Algorithm

To determine an optimal warping path P^* for two sequences X and Y, one could compute the total cost of all possible (N,M)-warping paths and then take the minimal cost. However, the number of different (N,M)-warping paths is exponential in N and M (see Exercise 3.9). Therefore, such a naive approach is computationally infeasible for large N and M. We now introduce an $O(NM)$ algorithm that is based on **dynamic programming**. The general idea behind dynamic programming is to break down a given problem into simpler subproblems and then to combine the solutions of the subproblems to reach an overall solution. In the case of DTW, the idea is to derive an optimal warping path for the original sequences from optimal warping paths for truncated subsequences. This idea can then be applied recursively. To formalize this idea, we define the prefix sequences $X(1:n) := (x_1,\ldots x_n)$ for $n \in [1:N]$ and $Y(1:m) := (y_1,\ldots y_m)$ for $m \in [1:M]$ and set

$$\mathbf{D}(n,m) := \mathrm{DTW}(X(1:n),Y(1:m)). \tag{3.22}$$

The values $\mathbf{D}(n,m)$ define an $(N \times M)$ matrix $\mathbf{D}$, which is also referred to as the **accumulated cost matrix**. Each value $\mathbf{D}(n,m)$ specifies the total (or accumulated) cost of an optimal warping path starting at cell $(1,1)$ and ending at cell (n,m). Obviously, one has $\mathbf{D}(N,M) = \mathrm{DTW}(X,Y)$. The next equations show how the accumulated cost matrix $\mathbf{D}$ can be computed recursively (see Figure 3.15 for an illustration):

$$\mathbf{D}(n,1) = \sum_{k=1}^{n} \mathbf{C}(k,1) \quad \text{for} \quad n \in [1:N], \tag{3.23}$$

$$\mathbf{D}(1,m) = \sum_{k=1}^{m} \mathbf{C}(1,k) \quad \text{for} \quad m \in [1:M], \tag{3.24}$$

$$\mathbf{D}(n,m) = \mathbf{C}(n,m) + \min \begin{cases} \mathbf{D}(n-1,m-1) \\ \mathbf{D}(n-1,m) \\ \mathbf{D}(n,m-1) \end{cases} \tag{3.25}$$

for $n \in [2:N]$ and $m \in [2:M]$.

We now give a formal proof of these equations. First, let $m = 1$ and $n \in [1:N]$. In this case, there is only one possible warping path, which assigns the single element

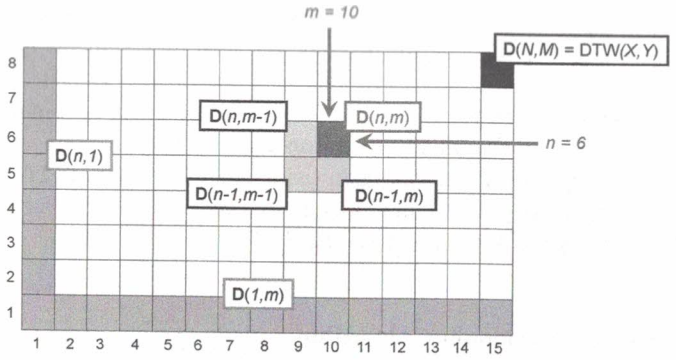

Fig. 3.15 Illustration of the recursive computation of the DTW distance. The blue cells indicate the entries $\mathbf{D}(n,1)$ and $\mathbf{D}(1,m)$ initialized by (3.23) and (3.24), respectively. The black cell indicates the final result $\mathbf{D}(N,M)$. The red cell indicates the current entry $\mathbf{D}(n,m)$ being computed by (3.25).

y_1 of $Y(1:1)$ to all elements of $X(1:n)$. Being the only possible warping path, this path is optimal. The cost of the path is obviously $\sum_{k=1}^{n}\mathbf{C}(k,1)$, which proves (3.23). Similarly, one obtains (3.24) for the case $n=1$ and $m\in[1:M]$. Now, let $n>1$ and $m>1$ and let $q=(q_1,\ldots,q_{L-1},q_L)$ denote an optimal warping path for $X(1:n)$ and $Y(1:m)$. We now show how this path can be obtained by extending a previously constructed optimal warping path. First note that the boundary condition (3.16) implies $q_L=(n,m)$. Setting $(a,b):=q_{L-1}$, the step size condition (3.18) implies $(a,b)\in\{(n-1,m-1),(n-1,m),(n,m-1)\}$. Furthermore, it follows that $(q_1,\ldots,q_{L-1})$ must be an optimal warping path for $X(1:a)$ and $Y(1:b)$ (otherwise, q would not be optimal for $X(1:n)$ and $Y(1:m)$). Since

$$\mathbf{D}(n,m)=c_{(q_1,\ldots,q_{L-1})}(X(1:a),Y(1:b))+\mathbf{C}(n,m), \tag{3.26}$$

the optimality of q implies the assertion of (3.25). This finishes the proof.

The equations (3.23) and (3.24) yield the initialization of a recursive procedure for computing $\mathbf{D}$. The values $\mathbf{D}(n,m)$ for $n>1$ and $m>1$ can then be computed via (3.25). The computation of $\mathbf{D}$ needs to be done by successively increasing n or m starting with the bottom left corner $(1,1)$ and ending with the upper right corner (N,M). This final cell (N,M) yields the DTW distance $\mathrm{DTW}(X,Y)=\mathbf{D}(N,M)$ (see Figure 3.15). The entries $\mathbf{D}(n,m)$ can be computed in different orders as long as n and m are increased monotonically. For example, one may proceed in a columnwise fashion, where the computation of the m-th column requires the values of the $(m-1)$-th column. This implies that, if one is only interested in the value $\mathrm{DTW}(X,Y)=\mathbf{D}(N,M)$, the storage requirement is $O(N)$. Similarly, one can proceed in a rowwise fashion, leading to a storage requirement of $O(M)$. However, in any case, the recursive step (3.25) is called $(N-1)\cdot(M-1)$ times. Since each step requires the minimization over three numbers as well as an addition, the overall complexity for computing $\mathbf{D}$ from a given cost matrix $\mathbf{C}$ is $O(NM)$.

Algorithm: DTW

Input: Cost matrix $\mathbf{C}$ of size $N\times M$
Output: Accumulated cost matrix $\mathbf{D}$
Optimal warping path P^*

Procedure: Initialize $(N\times M)$ matrix $\mathbf{D}$ by $\mathbf{D}(n,1)=\sum_{k=1}^{n}\mathbf{C}(k,1)$ for $n\in[1:N]$ and $\mathbf{D}(1,m)=\sum_{k=1}^{m}\mathbf{C}(1,k)$ for $m\in[1:M]$. Then compute in a nested loop for $n=2,\ldots,N$ and $m=2,\ldots,M$:

$$\mathbf{D}(n,m)=\mathbf{C}(n,m)+\min\{\mathbf{D}(n-1,m-1),\mathbf{D}(n-1,m),\mathbf{D}(n,m-1)\}.$$

Set $\ell=1$ and $q_\ell=(N,M)$. Then repeat the following steps until $q_\ell=(1,1)$:

Increase ℓ by one and let $(n,m)=q_{\ell-1}$.
If $n=1$, then $q_\ell=(1,m-1)$,
else if $m=1$, then $q_\ell=(n-1,m)$,
else $q_\ell=\mathrm{argmin}\{\mathbf{D}(n-1,m-1),\mathbf{D}(n-1,m),\mathbf{D}(n,m-1)\}$.
(If 'argmin' is not unique, take lexicographically smallest cell.)

Set $L=\ell$ and return $P^*=(q_L,q_{L-1},\ldots,q_1)$ as well as $\mathbf{D}$.

Table 3.2 DTW algorithm based on dynamic programming.

So far, we have computed the DTW distance, but we do not know how an optimal warping path looks. To determine such a path, one needs to recover the information about the minimizing cells in the recursion (3.25). Applying a **backtracking** procedure, the optimal warping path can be constructed incrementally in reverse order starting with the cell $q_1=(N,M)$. Suppose $q_\ell=(n,m)$ has been computed. In case $(n,m)=(1,1)$, we are done and set $L=\ell$. The path $P^*=(q_L,q_{L-1},\ldots,q_1)$ then defines an optimal warping path. Otherwise,

$$q_{\ell+1}=(1,m-1)\quad\text{if } n=1, \tag{3.27}$$
$$q_{\ell+1}=(n-1,m)\quad\text{if } m=1, \tag{3.28}$$
$$q_{\ell+1}=\mathrm{argmin}\begin{cases}\mathbf{D}(n-1,m-1),\\ \mathbf{D}(n-1,m),\\ \mathbf{D}(n,m-1)\end{cases} \tag{3.29}$$

if $n\in[2:N]$ and $m\in[2:M]$, where 'argmin' yields the cell leading to the minimum of the three values. Note that 'argmin' does not need to be unique, thus opening up the possibility of having more than one optimal warping path. To obtain a uniquely determined path, one may take, for example, the lexicographically smallest cell in case 'argmin' is not unique. Table 3.2 summarizes the entire procedure for computing the DTW distance as well as an optimal warping path.

We now look at a small example to illustrate how the algorithm in Table 3.2 is applied. To this end, we consider the feature space $\mathcal{F}=\mathbb{R}$ and the local measure $c:\mathcal{F}\times\mathcal{F}\to\mathbb{R}_{\geq 0}$ defined by $c(x,y)=|x-y|$, $x,y\in\mathbb{R}$. Furthermore, let $X=(1,3,3,8,1)$ and $Y=(2,0,0,8,7,2)$. Figure 3.16a shows the resulting cost

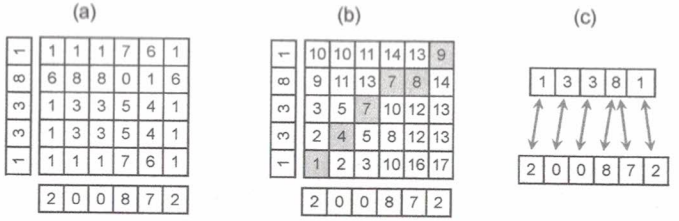

Fig. 3.16 **(a)** Cost matrix $\mathbf{C}$ for the two sequences $X = (1,3,3,8,1)$ and $Y = (2,0,0,8,7,2)$ over $\mathcal{F} = \mathbb{R}$ using the absolute differences as the local cost measure. **(b)** Accumulated cost matrix $\mathbf{D}$ with an optimal warping path. **(c)** Resulting alignment.

matrix $\mathbf{C}$, where we have indexed the rows from bottom to top and the columns from left to right (as opposed to the usual convention according to which matrices are visualized, since we think of the two dimensions being two time axes). Figure 3.16b shows the accumulated cost matrix $\mathbf{D}$ along with the optimal warping path $P^* = ((1,1),(2,2),(3,3),(4,4),(4,5),(5,6))$. This path is obtained by starting with the cell $q_1 = (N,M) = (5,6)$ of the upper right corner. Applying backtracking, one looks at all cells from which (N,M) can be reached by applying a valid step from Σ, which are the cells $(N-1,M)$, $(N,M-1)$, and $(N-1,M-1)$. From these cells, the predecessor cell q_2 is obtained by looking at the cell with minimal accumulated cost. In our example, this is the cell $q_{L-1} = (N-1,M-1) = (4,5)$, which has an accumulated cost of $\mathbf{D}(4,5) = 8$. The procedure is repeated until one reaches the cell $q_L = (1,1)$. Since in each backtracking step of our example there is only one choice for 'argmin' in (3.29), there is only one optimal warping path. The induced alignment of this path is shown in Figure 3.16c. At this point we want to emphasize that the backtracking in $\mathbf{D}$ is essential to find an optimal warping path. Such a path cannot be found by, for example, starting with the cell $(1,1)$ and then proceeding in a greedy fashion in the forward direction. One can check that such a forward approach would yield the path $((1,1),(1,2),(1,3),(2,3),(3,3),(4,4),(4,5),(5,6))$, which is not optimal.

Note that for the backtracking the entire accumulated cost matrix $\mathbf{D}$ may be needed. Therefore, as opposed to the case where one is only interested in computing $\mathrm{DTW}(X,Y)$, the storage requirement is $O(NM)$ when an optimal warping path is to be computed.

Finally, we introduce a small trick for simplifying the initialization (3.23) and (3.24). To this end, we extend the matrix $\mathbf{D}$ with an additional row and column (indexed by 0) by formally setting $\mathbf{D}(n,0) := \infty$ for $n \in [1:N]$, $\mathbf{D}(0,m) := \infty$ for $m \in [1:M]$, and $\mathbf{D}(0,0) := 0$. Then the recursion of (3.25) can be applied for $n \in [1:N]$ and $m \in [1:M]$, yielding exactly the same values for $\mathbf{D}$ as before (see Exercise 3.13). This trick will be helpful when considering modifications and variants of dynamic time warping as discussed next.

3.2.2 DTW Variants

Various modifications have been proposed in order to speed up DTW computations as well as to better control the overall course of the warping paths. In the following, we discuss some of these DTW variants.

3.2.2.1 Step Size Condition

Recall that the step size condition (3.18) expressed by the set $\Sigma = \{(1,0),(0,1),(1,1)\}$ is a kind of local continuity condition, ensuring that a warping path aligns each element of the sequence $X = (x_1,x_2,\ldots,x_N)$ to an element of $Y = (y_1,y_2,\ldots,y_M)$ and vice versa. One drawback of this condition is that a single element of one sequence may be assigned to many consecutive elements of the other sequence, which leads to vertical and horizontal sections in the warping path (see Figure 3.17a). Intuitively, in such cases the warping path is stuck at some position in one of the sequences, while moving on in the other sequence. In terms of physical time, this situation corresponds to a strong temporal deformation in the alignment of the two time series. To avoid such degenerations, one can modify the step size condition by constraining the slope of the admissible warping paths, which can be done by replacing the set Σ. For example, instead of using the original set $\Sigma = \{(1,0),(0,1),(1,1)\}$, one can use the set

$$\Sigma = \{(2,1),(1,2),(1,1)\}. \tag{3.30}$$

This leads to warping paths having a local slope within the bounds $1/2$ and 2 (see Figure 3.17b). The resulting accumulated cost matrix $\mathbf{D}$ can then be computed by the recursion

$$\mathbf{D}(n,m) = \mathbf{C}(n,m) + \min \begin{cases} \mathbf{D}(n-1,m-1), \\ \mathbf{D}(n-2,m-1), \\ \mathbf{D}(n-1,m-2) \end{cases} \tag{3.31}$$

for $n \in [1:N]$ and $m \in [1:N]$ with $(n,m) \neq (1,1)$. For the initialization, we again use the trick of extending $\mathbf{D}$, this time by two additional rows and columns (indexed by -1 and 0) and set $\mathbf{D}(1,1) := \mathbf{C}(1,1)$, $\mathbf{D}(n,-1) := \mathbf{D}(n,0) := \infty$ for $n \in [-1:N]$, and $\mathbf{D}(-1,m) := \mathbf{D}(0,m) := \infty$ for $m \in [-1:M]$. Note that, with respect to the modified step size condition, there is a warping path of finite total cost between two sequences X and Y if and only if the lengths N and M differ by less than a factor of two (see Exercise 3.14). Furthermore, note that not all elements of X need to be assigned to some element of Y and vice versa. This is illustrated by Figure 3.17b, where x_1 is assigned to y_1, x_3 is assigned to y_2, but x_2 is not assigned to any element of Y. In other words, x_2 is omitted and does not cause any cost at all.

Figure 3.17c gives a second example of a step size condition which avoids such omissions while imposing constraints on the slope of the warping path. The recursion of the resulting accumulated cost matrix $\mathbf{D}$ is given by

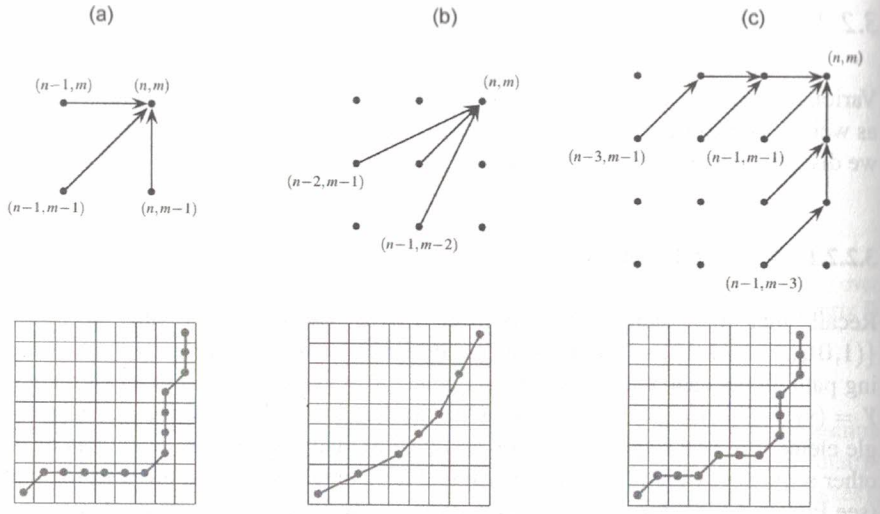

Fig. 3.17 Illustration of three different step size conditions (top), which express different local constraints on the admissible warping paths, along with some typical examples (bottom). The original step size condition based on the set $\Sigma = \{(1,0),(0,1),(1,1)\}$ is shown in (a).

$$\mathbf{D}(n,m) = \min \begin{cases} \mathbf{D}(n-1,m-1) + \mathbf{C}(n,m) \\ \mathbf{D}(n-2,m-1) + \mathbf{C}(n-1,m) + \mathbf{C}(n,m) \\ \mathbf{D}(n-1,m-2) + \mathbf{C}(n,m-1) + \mathbf{C}(n,m) \\ \mathbf{D}(n-3,m-1) + \mathbf{C}(n-2,m) + \mathbf{C}(n-1,m) + \mathbf{C}(n,m) \\ \mathbf{D}(n-1,m-3) + \mathbf{C}(n,m-2) + \mathbf{C}(n,m-1) + \mathbf{C}(n,m) \end{cases} \tag{3.32}$$

for $(n,m) \in [1:N] \times [1:M] \setminus \{(1,1)\}$. For the initialization, we extend the matrix by three additional rows and columns indexed by -2, -1, and 0. The initial values are set to $\mathbf{D}(1,1) := \mathbf{C}(1,1)$, $\mathbf{D}(n,-2) := \mathbf{D}(n,-1) := \mathbf{D}(n,0) := \infty$ for $n \in [-2:N]$, and $\mathbf{D}(-2,m) := \mathbf{D}(-1,m) := \mathbf{D}(0,m) := \infty$ for $m \in [-2:M]$. The global average slope of a resulting warping path lies between the values $1/3$ and 3. Note that this step size condition enforces that all elements of X are aligned to some element of Y and vice versa. In other words, in the recursion (3.32) all elements of X and Y generate some cost in the accumulated cost matrix $\mathbf{D}$—as opposed to the recursion (3.31). The examples in Figure 3.17 illustrate the differences of the resulting optimal warping paths computed with respect to different step size conditions.

3.2.2.2 Local Weights

To favor the vertical, horizontal, or diagonal direction in the alignment, one can introduce additional **local weights** $w_\mathrm{d}, w_\mathrm{h}, w_\mathrm{v} \in \mathbb{R}$. To compute the accumulated cost matrix $\mathbf{D}$, one uses the following initialization and recursion:

$$\mathbf{D}(1,1) := \mathbf{C}(1,1) \tag{3.33}$$

$$\mathbf{D}(n,1) = \sum_{k=1}^{n} w_\mathrm{h} \cdot \mathbf{C}(k,1) \text{ for } n \in [2:N] \tag{3.34}$$

$$\mathbf{D}(1,m) = \sum_{k=1}^{m} w_\mathrm{v} \cdot \mathbf{C}(1,k) \text{ for } m \in [2:M] \tag{3.35}$$

$$\mathbf{D}(n,m) = \min \begin{cases} \mathbf{D}(n-1,m-1) + w_\mathrm{d} \cdot \mathbf{C}(n,m) \\ \mathbf{D}(n-1,m) + w_\mathrm{v} \cdot \mathbf{C}(n,m) \\ \mathbf{D}(n,m-1) + w_\mathrm{h} \cdot \mathbf{C}(n,m) \end{cases} \tag{3.36}$$

for $n \in [2:N]$ and $m \in [2:M]$. The case $w_\mathrm{d} = w_\mathrm{h} = w_\mathrm{v} = 1$ reduces to classical DTW. Note that in the classical case one has a preference for the diagonal alignment direction, since one diagonal step (cost of one cell) corresponds to the combination of one horizontal and one vertical step (cost of two cells). To balance out this preference, one often chooses $w_\mathrm{d} = 2$ and $w_\mathrm{h} = w_\mathrm{v} = 1$. Similarly, one can introduce weights for other step size conditions.

3.2.2.3 Global Constraints

One common DTW variant is to impose **global constraints** on the admissible warping paths. Such constraints not only speed up DTW computations but also prevent "pathological" alignments by globally controlling the overall course of a warping path. More precisely, let $R \subseteq [1:N] \times [1:M]$ be a subset referred to as a global **constraint region**. Then a **warping path relative to** R is a warping path that entirely runs within the region R. The **optimal warping path relative to** R, denoted by P_R^*, is the cost-minimizing warping path among all warping paths relative to R.

Two well-known global constraint regions are the **Sakoe–Chiba band** and the **Itakura parallelogram**, as indicated by Figure 3.18. Alignments of cells can be selected only from the respective shaded region. The **Sakoe–Chiba band** runs along the main diagonal and has a fixed width (see Figure 3.18a). The **Itakura parallelogram** describes a region that constrains the slope of a warping path. More precisely, for a fixed $S \in \mathbb{R}_{>1}$, the Itakura parallelogram consists of all cells that lie within a **global** slope between the values $1/S$ and S (see Figure 3.18b). Note that a local step size condition may induce some global constraints. For example, using $\Sigma = \{(2,1),(1,2),(1,1)\}$ actually leads to warping paths that are contained in an Itakura parallelogram with $S = 2$. However, local step size conditions are stronger than global constraints, which do not enforce any **local** slope conditions on the warping paths.

For a general constraint region R, the path P_R^* can be computed similarly to the unconstrained case by formally setting $\mathbf{C}(n,m) := \infty$ for all $(n,m) \in [1:N] \times [1:M] \setminus R$. Therefore, in the computation of P_R^* only the cells that lie in R need to be evaluated. This may significantly speed up the DTW computation. For example, in

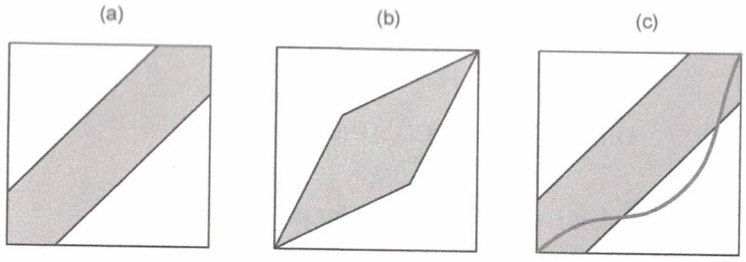

Fig. 3.18 - **(a)** Sakoe–Chiba band. **(b)** Itakura parallelogram . **(c)** Unconstraint optimal warping path P^* (red line) which does not run within the given constraint region.

case of a Sakoe–Chiba band of fixed width Δ, only $O(\Delta \cdot \max(N,M))$ computations need to be performed instead of $O(NM)$ as required in classical DTW. The savings can be substantial in particular in the case that $\Delta \ll M$ and $\Delta \ll N$. However, the usage of global constraint regions is also problematic, since the unconstrained optimal warping path P^* may traverse cells outside the specified constraint region. In this case, the constrained optimal warping path P_R^* does not coincide with P^* (see Figure 3.18c). Therefore, using a constraint region that is too strict may lead to undesirable or even completely useless alignment results. Only if the optimal unconstrained warping path P^* lies within R (which is of course not known in advance) does one obtain $P^* = P_R^*$ (see Exercise 3.17).

3.2.2.4 Multiscale DTW

We have seen that, when using the concept of global constraint regions, one needs to make sure that the optimal warping path to be computed actually lies within this region. Since this path is not known a priori, it is often difficult to find a good trade-off between choosing the constraint region as small as possible (to speed up computations) but large enough to contain the desired path. One possible strategy to increase the probability of finding the "right" path is to use data-dependent constraint regions instead of a data-independent, fixed constraint region. This idea can be realized by a multiscale approach to DTW, where the general strategy is to recursively project an optimal warping path computed at a coarse resolution level to the next highest level and then to refine the projected path. In the following, we summarize the main steps of such an approach (see Figure 3.19 for an overview).

Let $X_1 := X$ and $Y_1 := Y$ be two sequences having length $N_1 := N$ and $M_1 := M$, respectively. These two sequences represent the data at the highest resolution level, which we also refer to as level 1. The objective is to compute an optimal warping path P^* between X_1 and Y_1. The first step of multiscale DTW is to reduce the lengths of the sequences. This can be done, for example, by suitably coarsening X_1 and Y_1 and then reducing the feature sampling rate. Let us assume that we have a coarsening and downsampling procedure at hand that reduces the lengths by a factor $f_2 \in \mathbb{N}$.

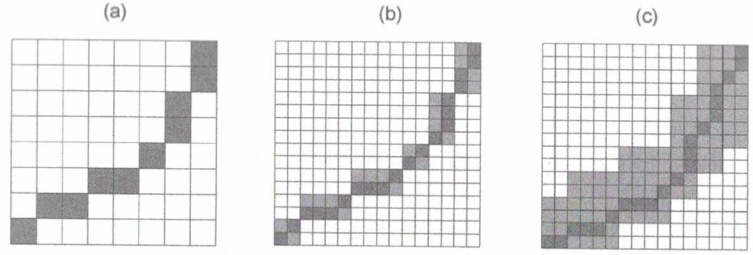

Fig. 3.19 **(a)** Optimal warping path P_2^* at level 2. **(b)** Optimal warping path P_R^* with respect to the constraint region R obtained by projecting path P_2^* to level 1. In this example, P_R^* does not coincide with the (unconstrained) optimal warping path P^*. **(c)** Optimal warping path $P_{R^\delta}^*$ using an increased constraint region $R^\delta \supset R$ with $\delta = 2$.

Furthermore, let us assume that f_2 divides N_1 and M_1, which can be achieved by suitably padding X_1 and Y_1. Let X_2 and Y_2 by the resulting feature sequences of length $N_2 := N_1/f_2$ and $M_2 := M_1/f_2$, respectively. Next, one computes an optimal warping path P_2^* of length L_2 between X_2 and Y_2 at the resulting resolution level, which we also call level 2. This path is projected onto level 1 to define a constraint region R, which consists of $L_2 \times (f_2)^2$ cells. Finally, an optimal warping path P_R^* relative to R is computed. We say that this procedure is **successful** if $P^* = P_R^*$. The overall number of cells to be computed in this procedure is $N_2M_2 + L_2(f_2)^2$, which is generally much smaller than the total number N_1M_1 of cells at level 1. This procedure can be recursively applied by introducing further levels of decreasing resolution. For a complexity analysis, we refer to Exercise 3.18.

One important issue with the multiscale approach is that the coarsened features at the different resolution levels need to be specified with great care. If the features are too coarse and the resolution too low, relevant information may be smoothed out or even lost. This may result in a poor warping path which does not lead to a meaningful constraint region for the next level. Note that a violation of the assumption $P_R^* = P^*$ at any level of the multiscale approach leads to irrecoverable errors of the overall procedure. Therefore, in practice, the recursion has to be stopped at a certain resolution level, where a full DTW needs to be computed. Also, to alleviate the problem that P_R^* may not coincide with P^*, one should increase the constraint region R—at the expense of efficiency—by a suitable neighborhood. This can be done, for example, by extending R to a constraint region R^δ, where, in addition to all cells in R, also the δ cells to the left, right, top, and bottom of all cells in R are included (see Figure 3.19c).