My Original 2D-DTW

Nobuaki Tanaka

November 17, 2025

1 Column-wise 2D DTW and River-path Extraction

1.1 Setting

We consider two surfaces A and B, given as discretized depth maps

$$A \in \mathbb{R}^{p\times q}, \qquad B \in \mathbb{R}^{r\times s}.$$

Here $A_{k,i}$ denotes the depth value at row k and column i of A. We treat each column as a vertical profile:

$$A_i := \begin{pmatrix} A_{1,i} \\ \vdots \\ A_{p,i} \end{pmatrix} \in \mathbb{R}^p, \qquad B_j := \begin{pmatrix} B_{1,j} \\ \vdots \\ B_{r,j} \end{pmatrix} \in \mathbb{R}^r.$$

Our goal is to measure the similarity between these column profiles using Dynamic Time Warping (DTW), and then to extract a consistent column-wise correspondence between A and B.

1.2 One-dimensional DTW distance

Let

$$x = (x_1, \dots, x_m)^\top \in \mathbb{R}^m, \quad y = (y_1, \dots, y_n)^\top \in \mathbb{R}^n$$

be two one-dimensional sequences. We define the local cost

$$c(k,\ell) := |x_k - y_\ell|$$

(for example, $|x_k - y_\ell|^2$ could also be used).

A warping path w is a finite sequence of index pairs

$$w = \big((i_1, j_1), \dots, (i_L, j_L)\big)$$

satisfying

- **Boundary conditions:** $(i_1, j_1) = (1,1)$ and $(i_L, j_L) = (m,n)$.
- **Monotonicity:** $i_{t+1} \ge i_t$ and $j_{t+1} \ge j_t$ for all t.
- **Step constraints:**

$$(i_{t+1} - i_t,\, j_{t+1} - j_t) \in \{(1,0), (0,1), (1,1)\}.$$

Then the DTW distance between x and y is defined as

$$\mathrm{DTW}(x, y) := \min_{w} \sum_{t=1}^{L} c\big(i_t, j_t\big).$$

1.3 Column-wise DTW distance matrix

For the depth maps A and B, we compute the DTW distance between each pair of column profiles:

$$D_{i,j} := \mathrm{DTW}(A_i, B_j), \qquad i = 1, \dots, q, \quad j = 1, \dots, s.$$

This yields a matrix

$$D \in \mathbb{R}^{q \times s},$$

where $D_{i,j}$ measures the dissimilarity between column i of A and column j of B (smaller values mean higher similarity).

We regard D as a function on the discrete grid

$$D : \ \{1, \dots, q\} \times \{1, \dots, s\} \to \mathbb{R}_{\geq 0}.$$

If we visualize $D_{i,j}$ as a height value over this grid, D defines a "distance landscape" in three dimensions: small values correspond to valleys, and large values to ridges.

1.4 Simple column-wise mapping by local minima

As a simple baseline, for each i we pick the column index $a_{\mathrm{loc}}(i)$ of B that minimizes the DTW distance:

$$a_{\mathrm{loc}}(i) := \arg \min_{1 \leq j \leq s} D_{i,j}.$$

This defines a mapping

$$a_{\mathrm{loc}} : \ \{1, \dots, q\} \to \{1, \dots, s\}, \qquad i \mapsto a_{\mathrm{loc}}(i),$$

which we interpret as "the column index of B corresponding to column i of A".

The points

$$\big(i, a_{\mathrm{loc}}(i)\big) \in \{1, \dots, q\} \times \{1, \dots, s\}$$

then form a discrete curve on the grid. If A and B are samples of smooth surfaces that are geometrically related, we can regard this curve as an approximation of a valley in the distance landscape D.

However, in the presence of noise or local artifacts, $a_{\mathrm{loc}}(i)$ may jump abruptly and fail to reflect the expected geometric continuity. To obtain a single smooth valley, we next formulate a global optimization directly on the matrix D.

2 Second-layer DTW on the column-wise distance matrix

2.1 River path as a warping path on D

We interpret the DTW distance matrix

$$D_{i,j} := \mathrm{DTW}(A_i, B_j)$$

as a discrete distance landscape on $\{1, \ldots, q\} \times \{1, \ldots, s\}$. Small values of $D_{i,j}$ form valleys (river beds), while large values form ridges. Our goal is to extract a single, coherent river path that runs along a valley of D and encodes a consistent column-wise correspondence between A and B.

We define a *river path* as a discrete sequence of grid points

$$w = \big((i_1, j_1), (i_2, j_2), \ldots, (i_L, j_L)\big),$$

satisfying the usual DTW constraints:

- **Boundary conditions:**

  $$i_1 = 1, \quad i_L = q, \qquad 1 \leq j_1 \leq s, \quad 1 \leq j_L \leq s.$$

  That is, the path starts on the first column index of A and ends on the last column index of A, while the column index of B is free on both ends.

- **Monotonicity:**

  $$i_{t+1} \geq i_t, \qquad j_{t+1} \geq j_t \qquad (t = 1, \ldots, L-1).$$

- **Step condition:**

  $$(i_{t+1} - i_t,\ j_{t+1} - j_t) \in \{(1,0), (0,1), (1,1)\},$$

  i.e., from each point the path can move only "down", "right", or "down-right" in the (i, j)-plane.

Given such a path, its cost is defined by summing the matrix values along the path,

$$C(w) := \sum_{t=1}^{L} D_{i_t, j_t}.$$

The *optimal river groove* is then defined as the minimum-cost path

$$w^* := \arg\min_{w} C(w),$$

where the minimization is taken over all admissible paths satisfying the boundary, monotonicity and step conditions above.

Once a path w^* is obtained, we can interpret it as a mapping from columns of A to columns of B. For each $i \in \{1, \ldots, q\}$, we collect all indices t such that $i_t = i$ and define, for example,

$$a(i) := \frac{1}{\#\{t : i_t = i\}} \sum_{t : i_t = i} j_t,$$

i.e., we take the average j-index along the path for a fixed i. This yields a (possibly non-integer) correspondence

$$a : \{1, \ldots, q\} \to [1, s],$$

which represents the river-bed alignment between the columns of A and those of B.

2.2 Dynamic-programming computation

The optimal river path w^* can be computed by a standard dynamic-programming scheme, which is formally identical to the usual DTW recursion. We define an accumulated cost matrix $F \in \mathbb{R}^{q\times s}$ by

$$F_{i,j} \;=\; D_{i,j} + \min\bigl(F_{i-1,j}, F_{i,j-1}, F_{i-1,j-1}\bigr),$$

with suitable boundary initialization. For example, if we fix the starting point at $(1, 1)$ and the ending point at (q, s), then

$$F_{1,1} = D_{1,1},$$

$$\begin{aligned} F_{i,1} &= D_{i,1} + F_{i-1,1} \qquad (i = 2, \ldots, q),\\ F_{1,j} &= D_{1,j} + F_{1,j-1} \qquad (j = 2, \ldots, s), \end{aligned}$$

and the above recursion is applied for $i \geq 2$, $j \geq 2$. In this case the total cost of the optimal path is simply

$$C(w^*) = F_{q,s}.$$

If we want to allow a free endpoint on the last row, i.e., any (q, j) with $1 \leq j \leq s$, we first compute F as above and then choose the best endpoint by

$$j_{\text{end}} := \arg\min_{1\leq j\leq s} F_{q,j},$$

and backtrack from (q, j_{end}) to a starting point $(1, j_{\text{start}})$ by repeatedly moving to the predecessor that achieves the minimum in the recursion. This backtracking yields the optimal river path w^*.

2.3 Greedy river-tracing variant

In practice, a simpler greedy variant can also be used. Starting from some initial grid point (i_1, j_1), for example a local minimum in the first few rows of D, we recursively move to the neighbor with the smallest local cost:

$$(i_{t+1}, j_{t+1}) := \arg\min_{(u,v)\in S(i_t,j_t)} D_{u,v},$$

where

$$S(i_t, j_t) := \bigl\{(i_t + 1, j_t),\, (i_t, j_t + 1),\, (i_t + 1, j_t + 1)\bigr\} \cap \bigl(\{1, \ldots, q\} \times \{1, \ldots, s\}\bigr).$$

This procedure follows the local valley of the distance landscape and often produces a visually reasonable river groove with much lower computational cost, at the expense of losing the global optimality guarantee of the dynamic-programming solution.

3 Interpretation and limitations

If A and B are samples of smooth surfaces that approximately correspond under some geometric transformation (e.g. translation or gentle deformation), we expect the optimal river path w^* (and the induced mapping a) to recover the column-wise alignment of A and B, up to discretization and noise.

On the other hand, the method may degrade in the following situations:

- the surfaces exhibit sharp folds or discontinuities,
- the dominant structure is along the row direction rather than the column direction,
- the noise level is very high, causing multiple competing valleys in D.

In such cases, one may need to consider higher-dimensional DTW extensions that allow warping in both row and column directions simultaneously, or to combine the present column-wise scheme with a row-wise or patch-wise DTW formulation.