Direct Image Matching by Dynamic Warping

Conference Paper · July 2004

DOI: 10.1109/CVPR.2004.50 · Source: IEEE Xplore

Hansheng Lei
26 PUBLICATIONS **491** CITATIONS
SEE PROFILE

Venu Govindaraju
University at Buffalo, The State University of New York
401 PUBLICATIONS **9,333** CITATIONS
SEE PROFILE

Direct Image Matching by Dynamic Warping

Hansheng Lei, Venu Govindaraju
CUBS, Center for Unified Biometrics and Sensors
State University of New York at Buffalo
Amherst, NY 14260
{hlei,govind}@cse.buffalo.edu

Abstract

In this paper, a new and efficient 2DDW (2-dimensional Dynamic Warping) algorithm for direct image matching is proposed. Similar to the 1-dimensional DTW (Dynamic Time Warping) for sequence matching and optimal alignment, the 2DDW is aimed to elastically matching images which may be not aligned well. However, finding the optimal alignment between two images has been proved to be NP-complete [4]. Therefore, reasonable constrains are imposed on the warping to bring down the complexity, such as continuity and monotonicity. The best complexity for continuous and monotonic 2DDW so far was reported as $O(N^2 9^N)$ in [7]. Our algorithm also guarantees continuity and monotonicity and the complexity is only $O(N^6)$.

1 Introduction

Image warping is a challenging stage in image analysis, such as image matching, image registration, pattern recognition [2]. However, optimal image warping or optimal pixel-to-pixel alignment is NP-complete [4]. Thus, a lot of efforts have been done to bring down the complexity of the warping algorithm by imposing reasonable constrains such as continuity and monotonicity [5, 7]. A continuous and monotonic 2DDW algorithm with complexity as $O(N^3 9^N)$ was reported in [7]. However, this algorithm only determines a one-way mapping from image A to B, which means that it deforms A to match template B. The underlying severe problem is: $2DDW(A, B) \neq 2DDW(A, B)$. To solve this problem, we need a bidirectional mapping, i.e., the mapping of elements between two images should be many-to-many instead of many-to-one.

The 1-dimensional DTW has been widely and successfully applied to 1-dimensional sequence matching and alignment[1]. Our motivation is whether we can extend DTW to 2-dimensional elastic image matching by making use of DTW's DP (dynamic programming) method ? Since DTW guarantees continuity and monotonicity of warping, will the 2DDW extended from DTW also preserve these properties?

The remainder of this paper is organized as follows. First, in section §2, we provide brief background of DTW to help explaining our 2DDW algorithm, which is described in section §3 in detail. We propose a graceful implementation method for the 2DDW algorithm in section §4. Finally, we evaluate it with experiments in §5 and conclude the paper in §6.

2 Dynamic Time Warping Background

Given two sequences $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_m)$, the distance DTW(X,Y) is similar to the edit distance. To calculate the DTW distance D(X,Y), we can first construct an n-by-m matrix, as shown in fig. 1. Then, we find a *path* in the matrix which starts from cell $(1, 1)$ to cell (n, m) so that the average cumulative cost along the path is minimized. If the path passes cell (i, j), then the cell (i, j) contributes $cost(x_i, y_j)$ to the cumulative cost. The cost function $cost(x_i, y_j)$ can be defined flexibly depending on application, for instance, $cost(x_i, y_j) = (x_i - y_i)^2$. This path can be determined using dynamic programming, because the following recursive equation holds:

$$\begin{aligned} D(i,j) &= cost(x_i, y_j) + min\{D(i-1, j), \\ &\quad D(i-1, j-1), D(i, j-1)\} \end{aligned} \tag{1}$$

The path may goes several cells horizontally along X or vertically along Y, which makes the matching between the two sequences not strictly one-one but one-many and many-one. This is the robustness that DTW provides to align sequences.

3 Extending 1-D DTW to 2-DDW

If we start from the 1-dimensional DTW, it will be easier to develop the 2DDW. Recall formula (1), $D(i, j)$ is re-

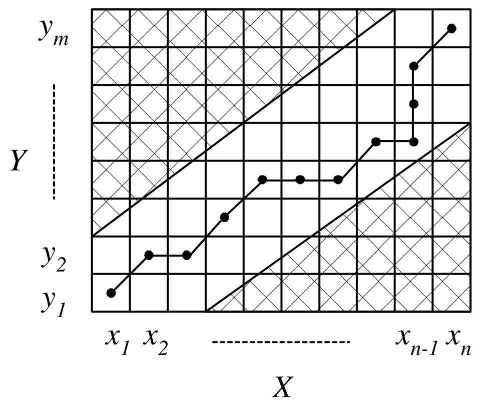

Figure 1. The path determined by DTW in the n-by-m matrix has the minimum average cumulative cost. The marked area is the constraint that path cannot go. The path indicates the optimal alignment: (x_1, y_1), (x_2, y_2), (x_3, y_2), $\cdots$, (x_{n-1}, y_{m-3}), (x_{n-1}, y_{m-2}), (x_{n-1}, y_{m-1}), (x_n, y_m).

cursively defined by $cost(i, j)$ and three previous stages: $D(i-1, j)$, $D(i-1, j-1)$,$D(i, j-1)$. Can we similarly define $D(i_1, j_1, i_2, j_2)$? The answer is positive. The challenge here is how to define $cost(i_1, j_1, i_2, j_2)$ and previous stages. Let's give the following definition first and then explain it.

$$\begin{aligned}
&D(i_1, j_1, i_2, j_2) = min\{\\
&cost_1 + D(i_1-1, j_1-1, i_2-1, j_2-1),\\
&cost_2 + D(i_1-1, j_1-1, i_2-1, j_2),\\
&cost_3 + D(i_1-1, j_1-1, i_2, j_2-1),\\
&cost_4 + D(i_1-1, j_1-1, i_2, j_2),\\
&cost_5 + D(i_1-1, j_1, i_2-1, j_2-1),\\
&cost_6 + D(i_1-1, j_1, i_2-1, j_2),\\
&cost_7 + D(i_1-1, j_1, i_2, j_2-1),\\
&cost_8 + D(i_1-1, j_1, i_2, j_2),\\
&cost_9 + D(i_1, j_1-1, i_2-1, j_2-1),\\
&cost_{10} + D(i_1, j_1-1, i_2-1, j_2),\\
&cost_{11} + D(i_1, j_1-1, i_2, j_2-1),\\
&cost_{12} + D(i_1, j_1-1, i_2, j_2),\\
&cost_{13} + D(i_1, j_1, i_2-1, j_2-1),\\
&cost_{14} + D(i_1, j_1, i_2-1, j_2),\\
&cost_{15} + D(i_1, j_1, i_2, j_2-1).\ \}
\end{aligned} \tag{2}$$

$D(i_1, j_1, i_2, j_2)$ denotes the cumulative cost of matching image I_1 at stage $[i_1, j_1]$ against I_2 at stage $[i_2, j_2]$. The starting stage is [1,1]. For one image alone, each stage $[i, j]$ has 3 possible previous stages subject to the boundary limitations: $[i-1, j-1]$, $[i-1, j]$ and $[i, j-1]$. Matching image I_1 at stage $[i_1, j_1]$ against I_2 at stage $[i_2, j_2]$ has at most $2^4 - 1$ possible previous stages.

Stage $(i_1-1, j_1-1, i_2-1, j_2-1)$ is one of the previous stages of (i_1, j_1, i_2, j_2). The matching is between partial image $I_1[1..(i_1-1), 1..(j_1-1)]$ and $I_2[1..(i_2-1), 1..(j_2-1)]$, as illustrated in fig.2 a_1) and a_2) respectively. $cost_1 = DTW(R_1, R_2) + DTW(C_1, C_2)$, where R_1 is the i_1-th row in image I_1 from column 1 through column i_1, C_1 is the j_1-th column in image I_2 from row 1 through row j_1. R_2 and C_2 are similar. $DTW(X, Y)$ calculates the 1-dimensional warping distance between sequence X and Y.

We summarize the 15 previous stages and corresponding costs in table 1.

4 Implementation of 2DDW

4.1 Calculation Order

Every stage $D([i_1, j_1], [i_2, j_2])$ is recursively defined by its previous stages. If and only if the all its previous stages have been determined, current stage can be calculated. For DTW, the calculation order is geometrically straightforward: from [1,1], then the first row, then the first column, then the second row, $\cdots$, so on and so forth until $[N, M]$. When all cumulative costs are done, tracing back the cost matrix can find the optimal path, as shown in fig.1. In 2DDW, geometrically imagining the calculation order is tedious. Fortunately, we can gracefully determine the calculation order of 2DDW by BFS (broad-first-search). Suppose image I_1 is N_1-by-M_1 and I_2 is N_2-by-M_2, we use the following procedure to determine the calculation order of stages. Each stage is denoted by four arguments as (i_1, j_1, i_2, j_2), where (i_1, j_1) indexes I_1 and (i_2, j_2) indexes I_2.

```
Procedure $Calculation\_Order(N_1, M_1, N_2, M_2)$.
1:  $Stage\_List = \emptyset$; /*to store the stage list*/
2:  $Current\_stage = (N_1, M_1, N_2, M_2)$;
3:  Add $Current\_Stage$ to Queue $Q$;
4:  while $Q$ not empty do
5:      $Current\_Stage = Remove\_Queue\_Head(Q)$;
6:      Append $Current\_Stage$ to the tail of $Stage\_List$;
7:      for each of 15 previous $stage$ of $Current\_Stage$ do
8:          if boundary allows & $stage$ has not been added to $Q$ then
9:              Add $stage$ to $Q$;
10:             Mark $stage$ indicating "Added to $Q$";
11:         end if
12:     end for
13: end while/*End procedure*/
```

Table 1. The 15 previous stages of stage (i_1, j_1, i_2, j_2) subject to the boundary limitations.

#	Stages	Cost	Matching(in fig.2)
1	$(i_1-1, j_1-1, i_2-1, j_2-1)$	$DTW(R_1,R_2)+DTW(C_1,C_2)$	$a_1 \leftrightarrow a_2$
2	$(i_1-1, j_1-1, i_2-1, j_2)$	$DTW(R_1,R_2)+DTW(C_1,C_2)$	$a_1 \leftrightarrow c_2$
3	$(i_1-1, j_1-1, i_2, j_2-1)$	$DTW(R_1,R_2)+DTW(C_1,C_2)$	$a_1 \leftrightarrow b_2$
4	(i_1-1, j_1-1, i_2, j_2)	$DTW(R_1,R_2)+DTW(C_1,C_2)$	$a_1 \leftrightarrow d_2$
5	$(i_1-1, j_1, i_2-1, j_2-1)$	$DTW(R_1,R_2)+DTW(C_1,C_2)$	$c_1 \leftrightarrow a_2$
6	(i_1-1, j_1, i_2-1, j_2)	$DTW(R_1,R_2)$	$c_1 \leftrightarrow c_2$
7	(i_1-1, j_1, i_2, j_2-1)	$DTW(R_1,R_2)+DTW(C_1,C_2)$	$c_1 \leftrightarrow b_2$
8	(i_1-1, j_1, i_2, j_2)	$DTW(R_1,R_2)$	$c_1 \leftrightarrow d_2$
9	$(i_1, j_1-1, i_2-1, j_2-1)$	$DTW(R_1,R_2)+DTW(C_1,C_2)$	$b_1 \leftrightarrow a_2$
10	(i_1, j_1-1, i_2-1, j_2)	$DTW(R_1,R_2)+DTW(C_1,C_2)$	$b_1 \leftrightarrow c_2$
11	(i_1, j_1-1, i_2, j_2-1)	$DTW(C_1,C_2)$	$b_1 \leftrightarrow b_2$
12	(i_1, j_1-1, i_2, j_2)	$DTW(C_1,C_2)$	$b_1 \leftrightarrow d_2$
13	(i_1, j_1, i_2-1, j_2-1)	$DTW(R_1,R_2)+DTW(C_1,C_2)$	$d_1 \leftrightarrow a_2$
14	(i_1, j_1, i_2-1, j_2)	$DTW(R_1,R_2)$	$d_1 \leftrightarrow c_2$
15	(i_1, j_1, i_2, j_2-1)	$DTW(C_1,C_2)$	$d_1 \leftrightarrow b_2$

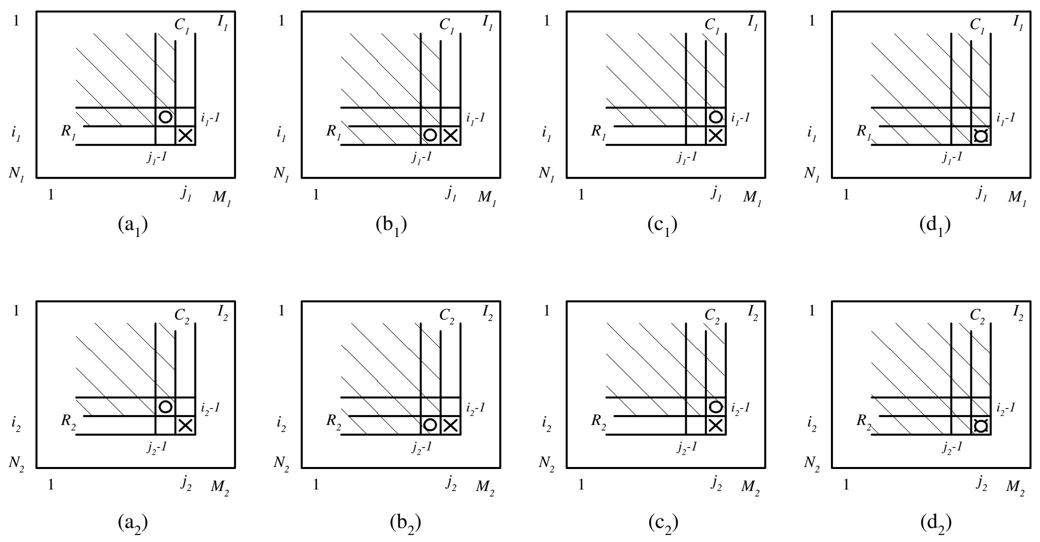

Figure 2. Stage $[i_1, j_1]$ of image I_1 has three possible previous stages: $[i_1-1, j_1-1]$,$[i_1, j_1-1]$ and $[i_1-1, j_1]$, as shown by (a_1),(b_1) and (c_1) respectively. Similarly as for image I_2. Before determining the matching I_1 at stage $[i_1, j_1]$ against I_2 at stage $[i_2, j_2]$, all pairwise matchings between $\{a_1$,b_1, c_1, d_1 $\}$ and $\{a_2$,b_2, c_2, d_2 $\}$ except $d_1 \leftrightarrow d_2$ need to be done.

In line 8, we need to check the boundary limitations. Any stage not in range $(1..N_1, 1..M_1, 1..N_2, 1..M_2)$ is not considered. This is how the recursion ends. After above processing, the $Stage\_List$ stores the order of stages with $D(N_1, M_1, N_2, M_2)$ in the front and $D(1, 1, 1, 1)$ at the tail. We can feel free to calculate each stage starting from the tail. Finally, the cumulative cost $D(N_1, M_1, N_2, M_2)$ is returned as the 2DDW distance.

4.2 Tracing alignment

We need a cost matrix with size N_1-by-M_1-by-N_2-by-M_2 to memorize the cumulative cost of each stage(Recall that we need a N-by-M cost matrix for DTW). When the final stage (N_1, M_1, N_2, M_2) is completed, we can track back into the cost matrix for the best alignment. We go back along the *minimum* previous stages utile reaching the fist stage $(1, 1, 1, 1)$. Suppose the current stage is (i_1, j_1, i_2, j_2). We scan its previous stages and see which one has is minimum one in term of cost. For an instance, suppose # 9 (see table 1) is the minimum one. The corresponding cost is $DTW(R_1, R_2)$+$DTW(C_1, C_2)$. We can determine the alignments of the elements on R_1(or C_1) and R_2 (or C_2) by DTW. In this way, we determine all the alignments.

We can see that the 2DDW is actually recursively solved by DTW. We know DTW is continuous and monotonic [3], therefore, it is not difficult to see that our 2DDW is also continuous and monotonic.

4.3 Complexity Analysis

The framework of implementing 2DDW is as follows. First, we determine the calculation order of stages by BFS. Totally, we have N^4 (for complexity discussion, suppose both I_1 and I_2 are N-by-N) stages. In this step, the complexity is $O(N^4)$. The second step is to calculate the stages one by one. The complexity of calculating each stage is $O(N^2)$ (recall that we need to calculate the DTW distance between rows or columns or both, where complexity is $O(N^2)$. Therefore, the total complexity is $O(N^4 \times N^2) = O(N^6)$. Finally, we need tracing back the cost matrix to determine the optimal alignment between the two images. The size of the cost matrix is N^4, storing a cost for each stage. Tracing the cost matrix is no more than N^4. Therefore, the total complexity is $O(N^6)$.

Just as DTW, band-constraint can be imposed onto 2DDW warping to reduce complexity. Although doing so makes the matching alignment sub-optimal, sub-optimal is enough in most cases of applications. Suppose the band-constraint width is W, which means stage $[i_1, j_1]$ in image I_1 can match stage $[i_2, j_2]$ in image I_2 with deviation up to $\leq W$, i.e., $|i_2 - i_1| \leq W$ and $|j_2 - j_1| \leq W$. The band-constraint reduces the number of stages to $O(N^2W)$ from $O(N^4)$. In addition, it reduces the complexity of DTW in calculating cost to $O(NW)$ from $O(N^2)$. Therefore, the total complexity becomes $O(N^3W)$.

Table 2. Complexity comparison of three 2D warping algorithms.

	This paper	Uchida [7]	Levin [5]
Complexity	$O(N^6)$	$O(N^3 9^N)$	$O(N^{4N})$
Complexity (w. Constraint)	$O(N^3W)$	$O(N^3W)$	$O(N^{4N})$

Compared with the complexity reported in [5] and [7], our algorithm is much more efficient. The complexities are summarized in table 2.

5 Experiments

2DDTW has two basic goals: one is to find the optimal or sub-optimal alignment between two images; another one is to calculate the distance (dissimilarity) of two images. Our experiments show that both the alignment and the dissimilarity determined by the 2DDW algorithm are robust. As an example, Fig.3 shows the distances among 4 facial images [6]. The 2DDW distances (those on the solid lines) can distinguish the two classes of faces correctly, although in each class the two faces vary in expression or pose. Obviously, the Euclidean distances (those on the dotted lines) can not distinguish them.

Fig.4 shows an example of alignment obtained by our 2DDW. The second face (fig. 4 (b)) is seriously deformed. The dynamic warping still can align the two faces [6] accurately, eye-to-eye, mouth-to-mouth,etc.

6 Conclusion

A new and efficient 2DDW algorithm with $O(N^6)$ complexity as low as is proposed. It is easy to implement by broad-firstly searching the calculation order of all stages. Similar to DTW, the 2DDW guarantees monotonicity and continuity. Applied to direct image matching, it determines robust distance and accurate alignment.

References

[1] D. Berndt and J.Clifford. Using dynamic time warping to find patterns in time series. *Working Notes of the Knowledge Discovery in Databases Workshop*, pages 359–370, 1994.

[2] C. A. Glassy and K. V. Mardia. A review of image warping methods. *Journal of Applied Statistics*, pages 155–171, 1998.

[3] E. Keogh. Exact indexing of dynamic time warping. *In 28th International Conference on Very Large Data Bases*, pages 406–417, 2002.

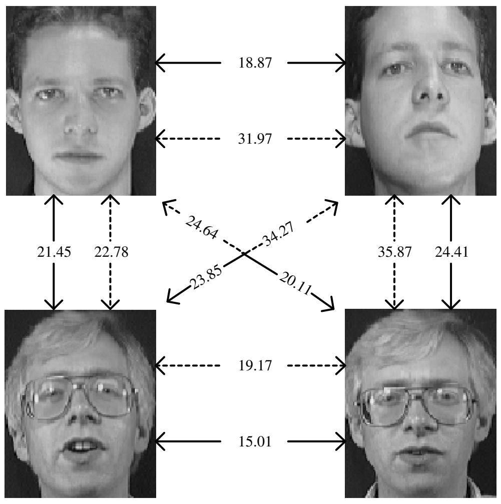

Figure 3. The distances of four faces by 2DDW and Euclidean norm. The 2DDW distances are shown on solid lines and Euclidean distances on dotted lines.

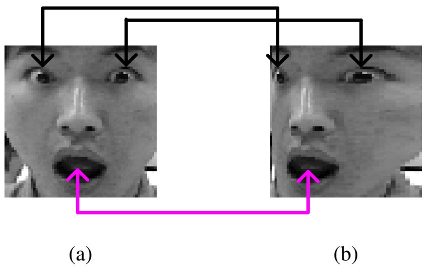

Figure 4. 2DDW aligns two faces accurately. Face (b) is deformed from (a).

[4] D. Keysers and W. Unger. Elastic image matching is np-complete. *Pattern Recognition Letters*, 24(1-3):445–453, 2003.

[5] E. Levin and R. Pieraccini. Dynamic planar warping for optical character recognition. *Proc. ICASSP*, pages 149–152, 1992.

[6] F. Samaria and A. Harter. Parameterisation of a stochastic model for human face identification. *2nd IEEE Workshop on Applications of Computer Vision*, Dec 1994.

[7] S. Uchida and H. Sakoe. An efficient two-dimensional warping algorithm. *IEICE Trans. Inf. and Syst.*, E82-D(1), 1999.