Computing Similarity Among 3D Objects Using Dynamic Time Warping

A. Angeles-Yreta and J. Figueroa-Nazuno

Centro de Investigación en Computación, Instituto Politécnico Nacional,
Unidad Profesional "Adolfo López Mateos" –Zacatenco- México, D.F.
malberto@sagitario.cic.ipn.mx, jfn@cic.ipn.mx

Abstract. A new model to compute similarity is presented. The representation of a 3D object is reviewed; sequence of vertices and index of vertices are the basic information about the *shape* of any 3D object. A linear function called *Labeling* is introduced to create a new sequence or time series from a 3D object. A method to create *randomly* 3D objects is also described. Experimental results show viability to compute similarity among 3D objects using the extracted sequences and the Dynamic Time Warping algorithm.

1 Introduction

The problem of defining and computing similarity among objects (concepts, time series, images, 3D objects, etc.) is the essence of many *Data Mining* applications.

Most of the methods of similarity search among 3D objects use a *feature extraction* technique [1]. A transformation from a 3D object to a *feature vector* is involved. The goal is to preserve, discover or select some property. This feature vector can be handled as a *time series.* Other methods consider 3D objects as images sequences (*2D view based methods*); afterward, models of similarity search among images. Also, there are methods based on histograms, even though they can be a particular case of feature extraction based methods, usually belong to another class (*Histogram based methods*). Finally, hybrid methods exist. In this work a new model to compute similarity among 3D objects is presented.

This work is organized as follow. In section 2, the *Dynamic Time Warping* algorithm used to compute similarity among sequences is described. In section 3, the *3D Object Representation* is discussed. In section 4, a linear function called *Labeling* is presented. This function *converts* the 3D object representation (sequence of vertices and index of these vertices) to a new sequence or time series useful to the properties of the Dynamic Time Warping algorithm. In section 5, a method to create *randomly* 3D objects is introduced. In section 6, *A Model to Compute Similarity* is presented. In section 7 results of *Experimental Test* that show viability of the model are presented. Finally, *Conclusions* of this work are presented in section 8.

2 Dynamic Time Warping

The Dynamic Time Warping algorithm has been applied in automatic speech recognition; is fundamentally a feature-matching scheme [2].

M. Lazo and A. Sanfeliu (Eds.): CIARP 2005, LNCS 3773, pp. 319 – 326, 2005.

Given two sequences $\boldsymbol{Q}$ and $\boldsymbol{C}$ (1), to accomplish the *alignment* (feature-match) is build a matrix of size n by m, where the (i,j) element of the matrix contains the metric $d(q_i, c_j)$ (2), in this case the Euclidean metric:

$$Q = q_1, q_2, q_3, \ldots, q_i, \ldots, q_n; \quad C = c_1, c_2, c_3, \ldots, c_j, \ldots, c_m \tag{1}$$

$$d(q_i, c_i) = \sqrt{(q_i - c_i)}^2 \tag{2}$$

The objective of the Dynamic Time Warping algorithm is to find a relation $i = \omega(j)$ that produces a warping path.

Definition 1. *Warping path*: A warping path W, is a contiguous set of matrix elements that defines a relation between two sequences, The k_{th} element of W is defined as $w_k = (i,j)_k$, so we have:

$$W = w_1, w_2, \ldots, w_k, \ldots, w_K \quad \max(m, n) \leq K < m + n - 1 \tag{3}$$

Until now, the time and space complexity of the Dynamic Time Warping algorithm is $O(nm)$ [3]. Several constraints has been proposed to reduce the complexity:

1. **Endpoint Constraints.** Requires that the endpoints match exactly; any path begin at (q_1,c_1), and end at (q_n, c_m). Another approach automatically locates endpoints .
2. **Monotonic.** The warping path should be monotonic, that is, $q_{k-1} \leq q_k$ and $c_{k-1} \leq c_k$. The features of a sequence Q must never match to features already matched in the sequence C.
3. **Global Constraints.** They imply allowed regions in the matrix; no warping path must be outside this area, even if optimal. Itakura parallelogram (left-side Fig. 1) constrains a warping path for maximum compression and expansion factors of two [2], the Window band (right-side Fig. 1) defines a **windows width** r to compress or expand the search space of a warping path. In this work the Itakura parallelogram is used.
4. **Local Constraints.** Determine alignment flexibility. In this work the warping path search is in 0°, 45° and 90°, Fig. 2 depicts this local constraint.

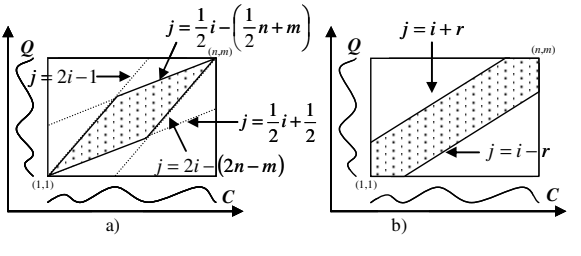


Fig. 1. a) Itakura parallelogram, and b) Window band are the most common global constraints for Dynamic Time Warping

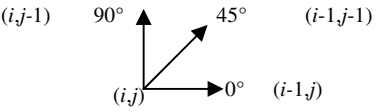


Fig. 2. The local constraint used in this work. It establishes the vicinity in the warping path search.

To avoid an exponential number of warping paths, we use only the warping path that minimizes the cost:

$$DTW(Q,C) \equiv \min\left\{\sqrt{\sum_{k=1}^{K} w_{ki}} / K\right. \tag{4}$$

The denominator K is used for the fact that warping paths may have different lengths; the sequence with the lowest match score is declared the most similar. The warping path can be found using dynamic programming, specifically:

$$\gamma(i,j) = d(q_i,c_i) + \min\{\gamma(i-1,j-1), \gamma(i-1,j), \gamma(i,j-1)\} \tag{5}$$

In Fig. 3 an example of two sequences before and after alignment, is shown, the reference (continuous line) and the sample (dotted line).

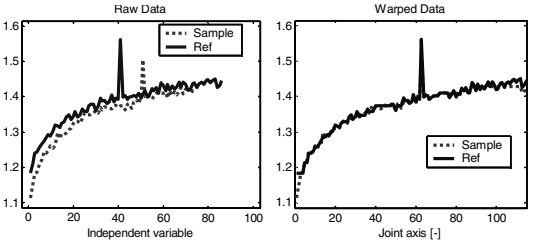


Fig. 3. Example of two sequences. Reference and sample are aligned using Dynamic Time Warping.

3 3D Object Representation

A 3D object can be represented as a graph; a graph is represented by an adjacency matrix. Most of the file formats used to represent 3D objects use an approximation of an adjacency matrix, that is, a sequence of vertices (6) and an index of vertices (7).

$$V = (v_1, v_2, \ldots, v_k), \text{ where } v_i = (x_i, y_i, z_i) \text{ and } x_i, y_i, z_i \in \Re \tag{6}$$

$$F = (v_{f_1}, v_{f_2}, \ldots, v_{f_n}), \text{ where } f_j \in [1,k] \tag{7}$$

Sequence of vertices ***V*** composes the graph nodes in a 3D space. The index of vertices ***F*** implies the order in which vertices must be drawn, and therefore the graph

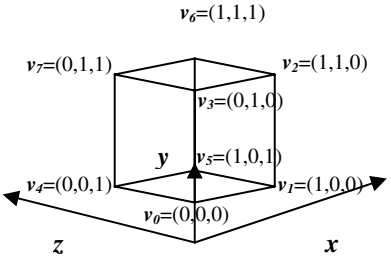


Fig. 4. The figure shows a cube, eight vertices are defined (v_0-v_7), the index of vertices is: v_0, v_1, v_2, v_3, -1, v_2, v_6, v_7, v_3, -1, v_6, v_5, v_4, v_7, -1, v_1, v_2, v_6, v_5, -1, v_0, v_4, v_7, v_3, -1, v_1, v_5, v_4, v_0

edges. Additional information such as position, rotation, cameras location, textures, etc., is ignored. Only the 3D object **shape** is considered when similarity is computed.

In Fig. 1 a cube in VRML format [4] is presented. Vertices are 3D points, the presence of -1 in the index of vertices means that the **face** sequence is almost complete and an extra vertex has to be added. For example, the first face composed by (v_0, v_1, v_2, v_3, -1), -1 has to bee substituted by v_0, that is, the first vertex in the face sequence. Several representations (file formats) uses this representation. Small modifications in index of vertices are detected. In this work this basic information is used (sequence of vertices and index of vertices). Graphics libraries, like OpenGL [5] agree this shape representation. The next code shows how to draw a 3D Object (polygon) defined by means of a sequence of vertices and an index of vertices

```
glBegin(GL_POLYGON);
  for (int i=0; i < length(F); i++)
    glVertex3f(V[F[i]].x, V[F[i].y, V[F[x].z);
glEnd();
```

The `glVertex3f` primitive puts a 3D point (vertex) in floating type. The ***F*** array is the index of vertices (7), and the structure ***V*** is the array of vertices or sequence of vertices (6). The next section presents a linear function to create another sequence or time series **based** on sequence of vertices and index of vertices.

4 Labeling 3D Objects

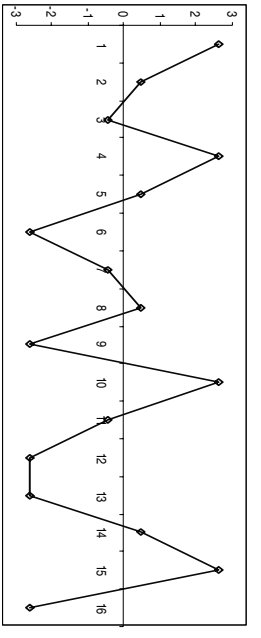
The Dynamic Time Warping is an excellent metric that **can be indexed** [3]. Given a sequence of vertices ***V*** and index of vertices ***F*** of any 3D object, a linear **function** (`Labeling`) is presented to create a **new** sequence. These sequences can be used to compute similarity among 3D objects using *Dynamic Time Warping* advantages.

```
Function Labeling(V, F):Q
  for (int i=0; i < length(F); i++)
    Q[i]=V[F[i]].x + V[F[i]].y + V[F[i]].z;
  return Q;
```

Parameters of the linear function (**`Labeling`**) are sequence of vertices ***V*** (6) and index of vertices ***F*** (7). The output is a **sequence** or **time series** ***Q*** that reflexes the

Table 1. A polyhedron of four vertices and a vertex index of length 16, the plot shows sequence Q

Vertices	Q
$v_1 = (0,1.081,1.529)$	2.61
$v_2 = (1.529,-1.081,0)$	0.448
$v_0 = (0,1.081,-1.529)$	-0.45
$v_1 = (0,1.081,1.529)$	2.61
$v_2 = (1.529,-1.081,0)$	0.448
$v_3 = (-1.529,-1.081,0)$	-2.61
$v_0 = (0,1.081,-1.529)$	-0.45
$v_2 = (1.529,-1.081,0)$	0.448
$v_3 = (-1.529,-1.081,0)$	-2.61
$v_1 = (0,1.081,1.529)$	2.61
$v_0 = (0,1.081,-1.529)$	-0.45
$v_3 = (-1.529,-1.081,0)$	-2.61
$v_3 = (-1.529,-1.081,0)$	-2.61
$v_2 = (1.529,-1.081,0)$	0.448
$v_1 = (0,1.081,1.529)$	2.61
$v_3 = (-1.529,-1.081,0)$	-2.61



movements of drawing a 3D object, the object vertices have been *labeled* with the x, y, and z addition. In Table 1 a polyhedron of four vertices (v_0-v_3) is considered.

5 Random Modifications of 3D Objects

To show the efficiency of the model (computing similarity among 3D objects using Dynamic Time Warping), a method to create 3D objects is described. Given a 3D object, cube, pyramid, etc., called **base**, random modifications to the sequence of vertices V (6) are made. The algorithm is sketched in the next code.

```
bool CObject3D::RetriveVRML(char *filename) {
 if (wml.Open(filename))    //VRML file (.wml)
  if(wml.Retrive(&V, &F)) return true;
  else return false;
 else return false; }
```

The **CObject3D** contains basic information (see section 3), that is, a sequence of vertices called V and an index of vertices called F. **RetriveVRML** method accepts a 3D object **base** (cube, prism, etc.) in VRML format, it can be modified to accept other grammar that define a sequence of vertices and an index of vertices (.3ds-The 3D Studio Format, .dfx-Autodesk's/AutoCAD, .off-Object File Format, etc.).

```
CObject3D::RandomModify(int nModify) {
 int list[V.Length]={0}; // Vertices to be modified
```

```
int count=0;              // Number of modifications
double x, y, z, min, max;
min = V.GetMin();  max = V.GetMax();

for(int i=0; i < nModify; i++)
 list[(int)myRand.IRandom(0, vertices.Length)]++;
for(i=0; i < V.Length; i++)
 if(list[i]>0) {
   count++;
   x = myRand.IRandom(min, max);
   y = myRand.IRandom(min, max);
   z = myRand.IRandom(min, max);
   V.ModifyVertix(i, x, y, z); }
   wml.SaveVRML(&V, &F, count);      }
```

RandomModify method defines a vertex list called **list** with the candidates to be modified using a pseudo-random number generator with uniform distribution and period of $2^{19,937}$-1 [6] (Mersennne Twister). Uniform distribution warranties equal probability to each vertex to be modified. **nModify** parameter gives indirect control of modifications number made to a sequence of vertices. Finally, **ModifyVertix**, updates the x, y, and z component of $v_i(6)$. Fig. 1 depicts this idea.

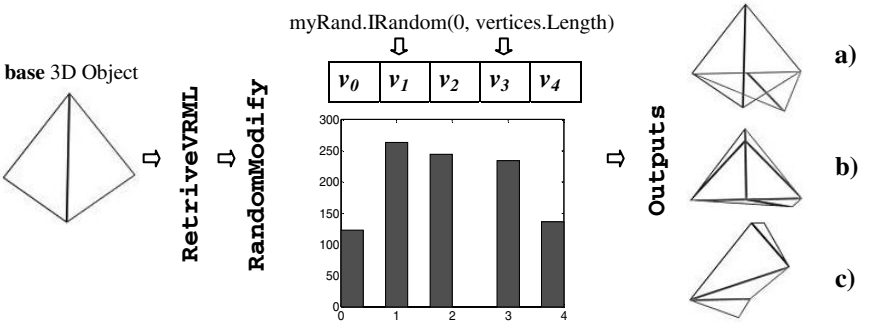


Fig. 5. A 3D Object *base* is the input to generate new 3D Objects with **a) one** modification, **b) two** modifications, and **c) three** modifications

Given a new data set of 3D Objects *randomly* created, a **Labeling** function (see section 4) can be computed over this data set to compute their similarity.

6 A Model to Compute Similarity

To compute similarity among 3D objects, a stage of pre-processing is required. The **Labeling** function creates sequences from 3D objects (see section 4), and these sequences are used to calculate a match score among these 3D objects. Fig 6 shows the model to compute similarity among 3D objects. An advantage of Dynamic Time Warping is that can be **indexed**. Future work presents results of indexing techniques.

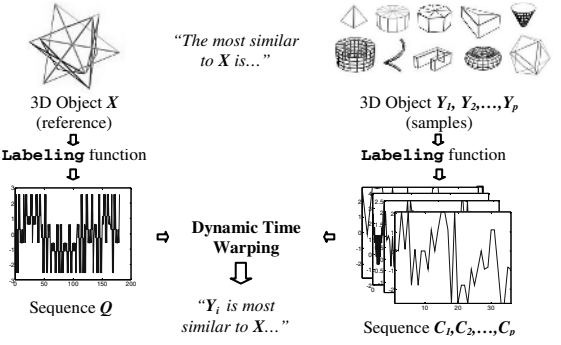


Fig. 6. For each 3D object in the database a sequence is created (see section 4). Using Dynamic Time Warping a similarity distance among 3D objects can be computed.

7 Experimental Tests

Two data sets were used in this work: 3D objects created with a typical *Computer Design System* CAD (specifically, 3D Studio Max 6) and 3D objects *randomly* created (see section 5), the pre-processing stage using the **`Labeling`** function (see section 4) was applied to all 3D objects, finally, Dynamic Time Warping metric was computed for each reference sequence (extracted from a 3D object) against every sample in the database.

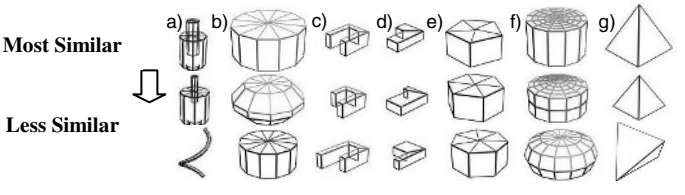


Fig. 7. a) SockAbsorber02, b) ChafCil01, c)Ext_C01, d)Ext_L01, e)Gengon01, f) Huso01, g)Pyramid01 and their two most similar 3D objects

Fig. 7 shows partial results. In Fig. 7 the less similar 3D objects to b), f) and g) respectively (marked with *), were created as part of another class (specifically, Gengon03, ChafCil05, and Polyhedron03), the proposed model can distinguish their similarity; this can be seen in Table 2.

Table 2. Dynamic Time Warping distance, ordered to most similar to less similar

	DTW		DTW		DTW		DTW		DTW		DTW		DTW
ShockAbsorber02	Distance	ChafCil01	Distance	Ext_C01	Distance	Ext_L01	Distance	Gengon01	Distance	Huso01	Distance	Piramide01	Distance
ShockAbsorber01	118.861	ChafCil02	106.2858	Ext_C02	6.672	Ext_L05	5.0388	Gengon02	13.7403	Huso02	130.4449	Piramide02	8.684
Muelle01	227.2536	**Gengon05**	120.635	Ext_C03	7.5414	Ext_L03	5.7651	Gengon03	20.4739	**ChafCil05**	205.8659	**Polyhedron03**	21.508

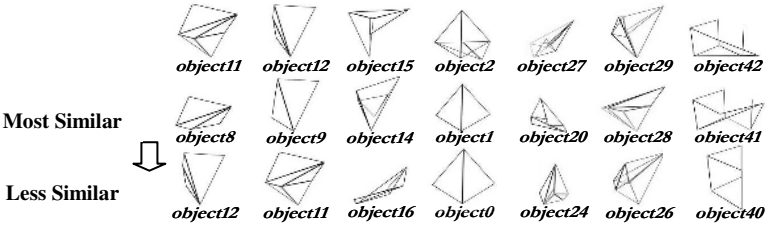


Fig. 8. 3D objects *randomly* created; *object11*, *object8*, and *object12* have 4 modifications, *object2*, *object1*, *object0*, has 1, 1, and 0 modifications respectively, the number of modifications can influence the Dynamic Time Warping distance

8 Conclusions

Experimental results show the efficiency of computing similarity among 3D objects, a linear function is required to *convert* a 3D object into a sequence and then compute Dynamic Time Warping distance among sequences. Two kinds of data sets were used in experimental test, the first one, created with a typical CAD; the other one was created from *base* 3D objects (cube, prism, and pyramid). There is not a true classification per se; the similarity between objects has to be recognized by humans. The proposed model to compute similarity among 3D objects is simpler than other approaches [1], and the results show this idea. An advantage of this model is that has been proved that Dynamic Time Warping technique can be indexed [3].

References

[1] Hlavaty, T., Skala, V.: A Survey of Methods for 3D Model Feature Extraction, *bulletin of the seminar of Geometry and Graphics in Teaching Contemporary Engineer*, Szczyrk, Poland, 2003, No: 13/03. pp. 5-8.

[2] Angeles-Yreta, A., Solís-Estrella, H. Landassuri-Moreno, V. Figueroa-Nazuno, J.: Similarity Search In Seismological Signals. *Fifth Mexican Internacional Conference on Computer Science*. Colima, México. September 2004, pp. 50-56.

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

[4] Hartman, J., Wernecke, J. The VRML 2.0 handbook: building moving worlds on the web, *Addison-Wesley*, 1996.

[5] Leech, J. Brown, P. (eds.).The OpenGL Graphics System: A Specification, *Silicon Graphics Press*, October 2004.

[6] Makoto Matsumoto, Takuji Nishimura, Mersenne Twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, *ACM Transactions on Modeling and Computer Simulation*, ACM Press, Vol. 8, 1998, pp. 3-30.