FI MU

Faculty of Informatics
Masaryk University

An Algorithm on Interpolating between Two Shapes of a Molecule

by

Aleš Křenek

FI MU Report Series FIMU-RS-97-04

May 1997

An Algorithm on Interpolating between Two Shapes of a Molecule

Aleš Křenek
Faculty of Informatics
Masaryk University
Brno, Czech Republic

Abstract

Conformational behaviour analysis produces a sequence of shapes of a molecule which are only the key points on the entire path. They differ from one another significantly and an interpolation is necessary to achieve a smooth visualization. However, standard interpolation techniques cannot be used. We introduce a hypothesis on the nature of the shape changes and derive an interpolation algorithm. Conditions required for proper function as well as some ideas how to overcome the algorithm's drawbacks are presented.

1 Chemical background

A simulation of conformational behaviour is an important part of computational chemistry. Unlike reactions this behaviour of a molecule involves changes in the shape of the molecule (called *geometry* here) only, bonds are neither created nor destroyed (the *topology* does not change).

Among all such possible geometries of a single molecule there are two categories of special interest

- *Conformers* (or conformations) are the stable ones in terms of potential energy—local minima on the potential energy hypersurface.
- *Transition states* are those of maximal potential energy on a minimal energy path connecting two conformers—saddle points on the hypersurface.

Figure 1 shows an example, conformers of propylaldehyde CH_3CH_2CHO.

Based on the assumption a molecule traverses only some minimal energy path it can be seen that describing geometries of visited conformers and transition states is sufficient to describe the entire conformation behaviour. Actually there is a recently developed software system strictly depending on that assumption [1, 2].

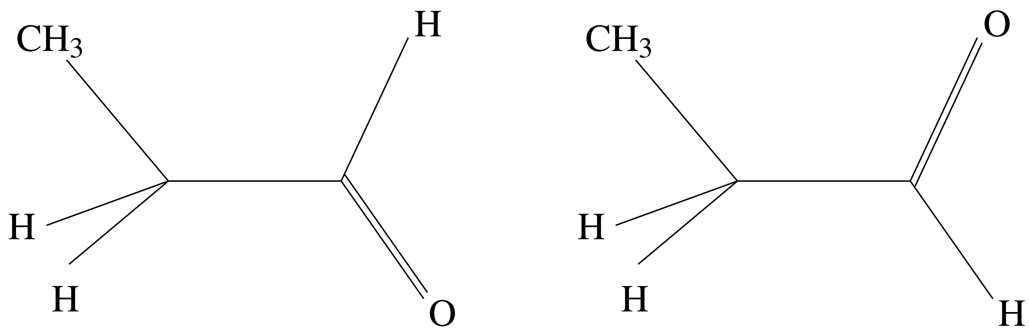

Figure 1: Conformers of propylaldehyde

First all the possible behaviour of the examined molecule is computed offline (taking upto several weeks of supercomputer time) and then a behaviour is simulated under certain conditions (temperature, pressure etc.). The resulting sequence of conformers and transition states is available for further analysis then. With this approach a global view on the behaviour is achieved in a significantly shorter time comparing to former strategies which preform the exhaustive energy calculations online.

We have developed a specialized visualization system [3, 4] for those data. The visualization itself is fairly straightforward, an animated sequence of 3D diagrams consisting of coloured balls and sticks is displayed. In theory the sequence of conformers and transitions states is sufficient to uniquely describe the behaviour. However, the geometries differ so much from one another that it requires, especially in more complex cases, considerable skills and a long experience of the user in order to be able to observe and understand the essence of the behaviour. That's why the visualization doesn't bring the required effect of an immediate insight on the data entirely.

Some sort of interpolation between the available data which would at least approximate the exact behaviour should be introduced then. As the approach used in the calculation is fairly new current molecular visualization systems (XMOL, GOPENMOL) don't perform such an interpolation—there were no data before that would require it.

2 Key observations

From the previous section emerges that besides the topology we are given a sequence of geometries of the examined molecule. The geometries are specified by Cartesian coordinates of the atoms in this case. An exact calculation of inbetweens is not affordable as several steps between a conformer and a transition state are required and the energy minimization necessary to follow the minimal energy path takes a time of about a few seconds for each step, even on powerful hardware. Therefore a computationally less expensive method, an approximating interpolation, must be used.

Unfortunately, a sequence $A \to B \to A$ where A is a conformer and B a transition state frequently occurs. This is the worst case for a traditional approach because anything more sophisticated than linear interpolation cannot be applied. However, a linear interpolation of Cartesian coordinates is not acceptable. Differences between two subsequent geometries are so great that the interpolation would cause a distortion which violates energy constraints at the first glance. We would be faced an animation which doesn't reflect a real behaviour even intuitively. It seems the information contained in the Cartesian coordinates themselves is not sufficient and other information emerging from a deeper knowledge must be exploited as well.

Looking at the energy constraints, the calculation described in [1], and the available data more closely we find out that the following holds in vast majority of cases:

- For each atom the positions of its direct neighbours almost do not change relatively to one another
- The entire change between two subsequent geometries can be composed[1] of elementary rotations

An ***elementary rotation***[2] is obtained by splitting the molecule as if a chosen bond was removed[3], keeping one part in a fixed position and rotating the other with the chosen bond being the axle. In fig. 1 the shape change involves an elemetary rotation along the horizontal C–C bond by an angle of π.

Those are essential observations. We can extract the elementary rotations from two subsequent geometries, interpolate them linearly, and recreate the inbetweens on the fly then. Therefore a smooth animation approximating the real behaviour can be achieved. Almost all real world data can be handled, at least partially if some of the above two statements is violated in some part of the molecule.

[1]The meaning of "composition" is somewhat intuitive here. An alogorithm presented in section 3.4 shows how it is done.

[2]The term is introduced only for the purposes of this paper.

[3]We do not consider cyclic structures yet. See the discussion in section 4.

3 The algorithm

3.1 Overview

The interpolation between two given geometries is done in two phases

1. A preparational phase executed only once for each pair of subsequent geometries, just after they have arrived. For each bond an angle expressing the associated elementary rotation is computed and stored. If the computation fails zero is substituted.

2. An iterating phase, parametrized by a number iterated between 0 and 1 in as many steps as is the number of required inbetween frames. Each elementary rotation's angle is multiplied with the parameter and the starting geometry is modified according to new elementary rotations computed in that way.

3.2 Fitting transformation

Before describing the interpolation algorithm in detail some previous results must be cited. Let two sets $\{\boldsymbol{x}_i\}_{i=1}^n$, $\{\boldsymbol{y}_i\}_{i=1}^n$ of 3D Cartesian coordinates be given and

$$\boldsymbol{y}_i = R\boldsymbol{x}_i + \boldsymbol{t} + \xi_i$$

hold for each $i = 1 \dots n$ where R is a fixed but unknown rotational matrix, $\boldsymbol{t}$ a fixed unknown vector of transition, and ξ some noise (an order of magnitude less signifficant than the coordinate values).

From the coordinates we can reconstruct such a matrix R and vector $\boldsymbol{t}$ (together called a ***fitting*** transformation) that

$$\sum_{i=1}^{n} \|R\boldsymbol{x}_i + \boldsymbol{t} - \boldsymbol{y}_i\|$$

is minimal. Moreover, this can be done efficiently (30 iterations of a loop containing about 80 lines of simple C code—floating point addition, multiplication, and division only). Our code is based on [5], the method is described in detail in [3, 4].

3.3 Preparational phase

Figure 2 shows a pseudocode for the phase 1. It takes a topology—graph (V, E) and two geometries A, B as its inputs. A value $angle_{\{i,j\}}$ is computed for each

```
color_i := WHITE        ∀i ∈ V
color_root := GREY
repeat
    for i ∈ V : color_i = GREY
        color_i := BLACK
        R := fit(A, B, i)
        B := RB
        for j ∈ V : {i, j} ∈ E, color_j = WHITE
            if {x ∈ V : {x, j} ∈ E} = {i}
                color_j := BLACK
                angle_{i,j} := 0
            else
                color_j := GREY
                n := 0, φ := 0
                for k ∈ V − {i} : {j, k} ∈ E
                    φ := φ + ∠(A_i A_j A_k, B_i B_j B_k)
                    n := n + 1
                end loop
                φ := φ/n
                R := rot(A_i, A_j, φ)
                err:=0
                for k ∈ V − {i} : {j, k} ∈ E
                    err := err + ‖RA_k − B_k‖
                end loop
                if err/n < LIMIT
                    angle_{i,j} := φ
                else
                    angle_{i,j} := 0
                end if
            end if
        end loop
    end loop
until {x ∈ V : color_x = GREY} = ∅
```

Figure 2: Preparational phase of the interpolation

bond $\{i,j\} \in E$. As the graph is unoriented E is a set of unordered pairs. The geometries are matrices consisting of column vectors of homogeneous 3D Cartesian coordinates, subscripted notation A_i stands for the i-th column.

The skeleton of the algorithm (lines 1–5, 8–10, 12, 13, 30–33) is a version of breadth first search of a graph. Therefore we preserve the established terms of `WHITE` unvisited vertices, `GREY` those which should be processed in the next iteration, and `BLACK` finished ones. It can be proven the algorithm terminates after all vertices have been visited. In the case of acyclic graphs (which we consider only) this means also all the edges have been traversed. As the BFS is well known in graph theory we omit a formal proof here, it can be found for instance in [6].

The main idea of the algorithm is based on the above observations. The molecule is being split bond by bond as the BFS runs. For each bond local surroundings of the two bound atoms are considered, the surroundings are examined whether they rotate relatively to each other, and if so the angle of rotation is determined.

Let's focus on essential parts now. In lines 6 and 7 we have already chosen a `GREY` vertex i. The function $fit(A,B,i)$ computes a fitting transformation[4] R such that coordinates of the vertex i and its immediate neighbours are as close as possible in A and in RB. The geometry B is transformed as a whole then. Comparing the geometries, determining rotational angles etc. is further possible. Actually the fitting can be done unambiguously if and only if i has at least two distinct neighbours and those are non-collinear with respect to i. For the root this can be achieved with a careful choice (if not, the molecule would be just a linear chain). To keep the property of i across the BFS iterations the algorithm can be easily extended. For the sake of simplicity we omit the appropriate code in fig. 2. The problem is marginal anyway.

In line 9 we check whether the next examined atom j is a leaf. If so it doesn't make sense to rotate the single atom and zero angle is stored for the bond $i-j$. Otherwise a candidate angle of the expected elementary rotation is computed for each j's neighbour (loop 15–18) and the angles are averaged (notation $\angle(A_iA_jA_k, B_iB_jB_k)$ stands for an angle between the planes given by the point triplets). The function $rot(P_1,P_2,\phi)$ computes a matrix of rotation by angle ϕ with the line P_1P_2 being an axle. If the actual rotation given by the averaged angles applied on the starting geometry brings all the j's neighbours acceptably close to their ending geometry positions the angle is accepted.

If the computation of an angle fails zero is substituted. Then, during the in-

[4] As we deal with homogeneous coordinates now the 4×4 matrix covers both the rotation R and transition $\boldsymbol{t}$ from section 3.2.

```
color_i := WHITE        ∀i ∈ V
color_root := GREY
C_root := A_root
for  i ∈ V : {i,root} ∈ E
    C_i := A_i
end loop
repeat
    for  i ∈ V : color_i = GREY
        color_i := BLACK
        R := fit(A,C,i)
        C := RC
        for  j ∈ V : {i,j} ∈ E, color_j = WHITE
            if  {x ∈ V : {x,j} ∈ E} = {i}
                color_j := BLACK
            else
                color_j := GREY
                R := rot(A_i,A_j,step ∗ angle_{i,j})
                for  k ∈ V − {i} : {j,k} ∈ E
                    C_k := RA_k
                end loop
            end if
        end loop
    end loop
until  {x ∈ V : color_x = GREY} = ∅
```

Figure 3: Iterating phase of the interpolation

terpolation (see section 3.4) the molecule doesn't seem to twist by any particular elementary rotation on the bond at all and the entire effect of the shape change (more complex than an elementary rotation) turns up on the whole between the last interpolated frame and the frame exactly derived from the ending geometry. This causes discontinuities in the visualization but allows dealing with other bonds correctly. Otherwise we should give up the pair of geometries entirely.

3.4 Iterating phase

The algorithm shown in fig. 3 takes the starting geometry A, computed angles from the preparational phase, and a number *step* between 0 and 1 as its inputs and

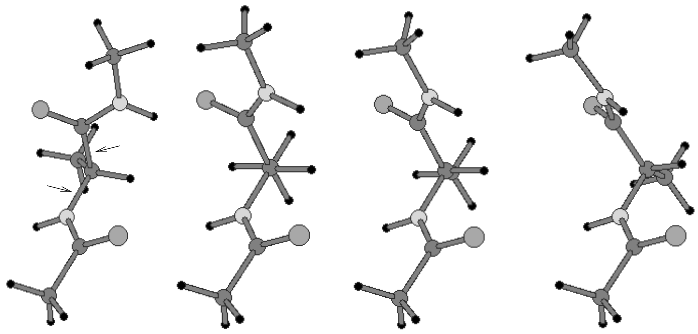

Figure 4: Generated inbetweens

computes an inbetween geometry C according to the value of *step*.

The algorithm has the BFS skeleton again. Coordinates of the root and its neighbours are taken "as are" from the starting geometry (lines 3 and 5). Then, bonds are processed in the same manner as during the preparational phase.

In each iteration the BFS together with the initialization of C guarantee that we have already computed the current coordinates of i and all its neighbours. That assures the fitting performed in lines 10 and 11 on partial current geometry makes sense.

If j is a leaf (test 13) there is nothing to do, just the starting coordinates are taken. Otherwise for each neighbour of j (loop 18–20) we rotate its starting position along $i-j$ by an angle computed as the product of the parameter and the total angle for the bond $i-j$ computed during the preparational phase. The computation can be done easily due to the fitting.

According to the method the first frame (parameter value 0) always coincide with the starting geometry and, under the assumption we have never failed during the preparational phase, the last one (parameter value 1) with the ending geometry. Relatively smooth visualization of discrete shape changes is achieved then.

Let's look at the fig. 1 again. If the algorithm is given corresponding input data in the preparational phase a value of π is assigned to the horizontal C–C bond, 0 to the others. During the interpolation the parts of the molecule rotate relatively to each other along the bond. Eventually, if the methyl CH_3 rotated as well it would be detected independently as only local surroundings are considered. That would result in a composed shape change during the interpolation.

On the other hand a linear interpolation of Cartesian coordinates is unacceptable at the first glance even intuitively.

Figure 4 shows a simple but nontrivial example of a real molecule. The first and the last geometries had been given, the others were generated with the described algorithm. Two elementary rotations are involved in this case, corresponding bonds are pointed by the arrows in the first image.

4 Further research

Currently we can handle about 90% of available data. The rest consists of the cases where more complex shape changes occur and the concept of elementary rotation is violated. Proper detection of such cases, their classification, and handling in the algorithm may be a subject to further research.

The presented algorithm is fairly robust against numerical noise present in real data. However, the actual noise ("noise" from the idealistic point of view which was inevitable for the interpolation) in our data is a more significant factor. It covers minor but significant geometry changes. In principle, this does not hurt too much as they are reflected in the visualization anyway but the discontinuities they cause are annoying. A sort of factorization of a geometry change into an idealistic elementary rotations and a residuum and consequent linear interpolation of the residuum (this should be affordable) instead of ignoring it would be a solution.

Even more annoying is current incapability of handling cyclic topologies. Actually, a shape-changing cycle is a nightmare as it ignores the concept of elementary rotations totally, the involved shape changes are more complex (describing them is beyond the scope of this paper). Currently, we only prevent the algorithm to compute elementary rotation angles on the bonds which are parts of a cycle.

5 Conclusions

The required interpolation between two subsequent geometries computed by conformational analysis presents a challenge. Based on observations of the prevailing nature of the changes a concept of *elementary rotations* was introduced and an interpolating algorithm developed. Now it is implemented as a module of a visualization system and used by a group of users. About 90% of real data are handled satisfactorily, we get a smooth animation. However, there are still unsolved problems which motivate further research.

Acknowledgement

This work has been supported by the Grant Agency of the Czech Republic, grant no. 203/94/0522. Many thanks to prof. Jaroslav Koča for the first inspiration

and help with the chemical stuff, Karel Zikan for endless discussion on the topic, Zdeněk Kříž and other users for patient testing.

References

[1] Jaroslav Koča, *Computer Program* CICADA—*travelling along conformational PES*, J. Mol. Struct. (Theochem) **308** (1994), 13–24.

[2] Jaroslav Koča, *Computer Simulation of Conformational Movement Based on Interconversion Phenomena*, J. Mol. Struct. (Theochem) **343** (1995), 125–132.

[3] Aleš Křenek, *PES Travelling Visualization*, Master's thesis, Masaryk University, Faculty of Informatics, 1995.

[4] Aleš Křenek, *Molecular models superposition and visualization*, Proc. WSCG'96 (V. Skala and N. Magnenat Thalmann, eds.), vol. 2, 1996, pp. 271–277.

[5] David J. Heisterberg, *Quatfit*, unpublished results, `ftp://kekule.osc.edu/pub/chemistry/software/SOURCES/C/quaternion_molfit`, 1990.

[6] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, *Introduction to Algorithms*, MIT Electrical Engineering and Computer Science Series, 1989, ISBN: 0-262-03141-8.

Publications in the FI MU Report Series are in general accessible
via WWW and anonymous FTP:

http://www.fi.muni.cz/informatics/reports/
ftp ftp.fi.muni.cz (cd pub/reports)

Copies may be also obtained by contacting:

Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic