# BioCD : An efficient algorithm for self-collision and distance computation between highly articulated molecular models.

Vicente Ruiz de Angulo
IRI (CSIC-UPC), Barcelona, Spain
Email: ruiz@iri.upc.edu

Juan Cortés and Thierry Siméon
LAAS-CNRS, Toulouse, France
Email: jcortes,nic@laas.fr

***Abstract*— This paper describes an efficient approach to (self) collision detection and distance computations for complex articulated mechanisms such as molecular chains. The proposed algorithm called BioCD is particularly designed for sampling-based motion planning on molecular models described by long kinematic chains possibly including cycles. The algorithm considers that the kinematic chain is structured into a number of rigid groups articulated by preselected degrees of freedom. This structuring is exploited by a two-level spatially-adapted hierarchy. The proposed algorithm is not limited to particular kinematic topologies and allows good collision detection times. BioCD is also tailored to deal with the particularities imposed by the molecular context on collision detection. Experimental results show the effectiveness of the proposed approach which is able to process thousands of (self) collision tests per second on flexible protein models with up to hundreds of degrees of freedom.**

## I. Introduction

Collision detection (CD) is a classical problem in robotics and also computer graphics. See [8], [12], [4] for recent surveys. It has been widely studied during the past decades and several efficient collision detection packages are now available [1]. An important application of CD algorithms is robot motion planning. In particular, the sampling-based planners (e.g. [9], [10]) extensively use CD techniques for checking the validity of the sampled configurations and of the local paths computed between the samples. It is well known that these planners spend most of the computing time for these validity tests. Therefore, their overall performance for exploring constrained and possibly highly dimensional configuration spaces strongly relies on efficient geometric CD techniques.

Most of the current CD approaches were designed to face the geometric complexity of scenes composed by a large number of possibly complicated obstacles. Self-collisions of the robot are generally handled by simply testing the mutual collisions between the bodies considered as a set of independent rigid objects. While such approach is sufficient for simple robots with a limited number of articulated bodies, it turns out to become inefficient when applied to more complex articulated mechanisms (e.g. humanoid robots [11]).

Motion planning techniques are applied today in diverse domains such as graphic animation [18], [16] and computational biology [2], [1], [17], [5] that involve articulated systems with many degrees of freedom (DOF) which require new classes of CD algorithms. In computational biology applications, collision detection is a challenging problem since macro-molecules such as proteins can be described as highly articulated chains with up to thousands of DOF. Figure 1 shows the model of an intermediate-size protein and gives an idea of the complexity of the corresponding articulated mechanism. The requirement for specific CD techniques is therefore crucial to circumvent the high quadratic cost of enumerating all non-bonded atom pairs in models with thousands of atoms. In particular, the number of pairs to be considered for self-collision can be substantially reduced by considering the structural constraints imposed by the kinematic chain structure. Only a few works in CD literature [7], [13] addressed the specific problem of testing self-collisions for such complex kinematic chains.

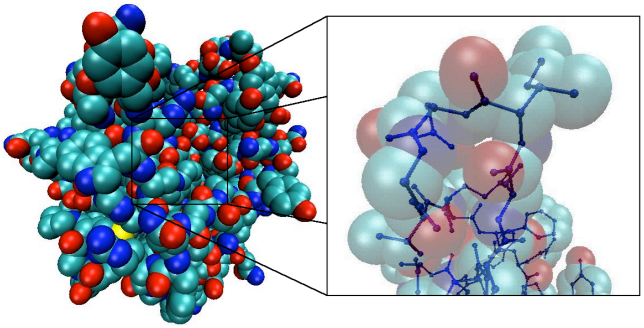

Fig. 1. A protein can be seen as a complex articulated chain.

This paper describes the BioCD algorithm that we developed for efficient collision and distance computations between highly articulated molecular chains, including efficient self-collision detection inside each chain. BioCD is currently integrated into the path-planning based approach investigated in [6] for computing large-amplitude motions of flexible molecules.

BioCD considers a mechanical model of macromolecules. They are described as long serial chains with short side chains along, but also with cycles to represent structural features of

[1] e.g. see http://gamma.cs.unc.edu

the molecule (Section II). After stating our motivation with respect to the close related work (Section III), we describe the approach used in BioCD (Section IV). The bodies of the articulated molecular model are composed of rigid atom groups that correspond to structural elements. The algorithm relies on two-levels of bounding volume hierarchies grouped according to spatial proximity. The first level organizes the rigid groups of the articulated model and the second one organizes the atoms inside each rigid group. Such data-structure is not attached to particular topologies of the kinematic chain. It can be efficiently tested for collision and updated with a moderate cost. Finally, BioCD is also tailored to the particularities imposed by the molecular context on collision detection (Section V). First, atoms are not characterized by an unique geometry; their size varies depending on the type of interaction with the other atom tested for collision. In addition, collisions must not be checked between all atom pairs, but only between pairs of non-bonded atoms that are separated by at least four covalent bonds. Experimental tests show the practical efficiency of BioCD to process thousands of collision tests per second for flexible protein models with hundreds of DOF (Section VI).

## II. Macromolecules as articulated mechanisms

### A. Kinematic structure

Biological macromolecules such as proteins have a complex and flexible structure. They can be represented as a graph in which vertices are the atoms and the edges are bonds. Molecular modeling approaches usually assume for reducing the number of variables that bond lengths and bond angles are constant parameters. Under this assumption, the molecular chain can be seen as an articulated mechanism with revolute joints that correspond to bond torsions.

A protein is composed of one or several long polypeptide chains, folded in a globular manner. The mechanical model of a polypeptide is composed of a set of kinematic chains: the main-chain (or backbone) and the side-chains of the amino acid residues.

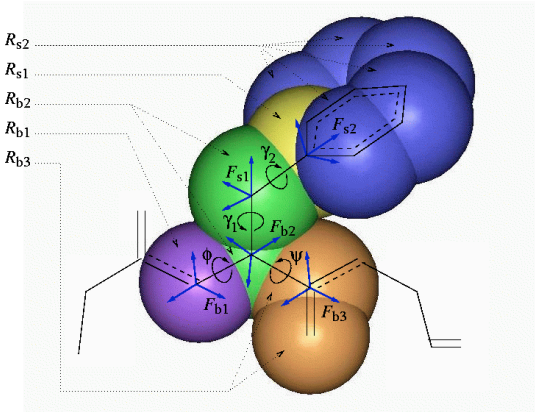


Fig. 2. Mechanical model of a protein residue (phenylalanin). Groups of rigidly bonded atoms form the bodies. The articulations between bodies correspond to bond torsions.

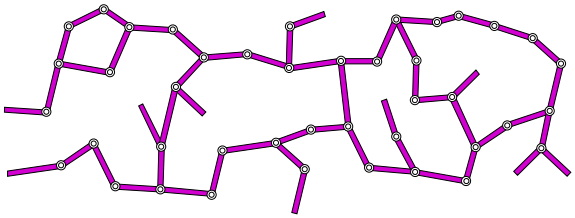

Fig. 3. Multiple closed chain illustrating a protein model.

Figure 2 shows the mechanical model for one flexible amino acid residue of a protein. It is composed of five rigid bodies, classified in: backbone rigid groups $\{R_{b1}, R_{b2}, R_{b3}\}$ and side-chain rigid groups $\{R_{s1}, R_{s2}\}$. The rotation angles between them are $\phi$ and $\psi$ for the backbone, and $\gamma_1$ and $\gamma_2$ for the side-chain. These angles are the dihedral angles classically used in molecular modeling.

There is clearly a linear structure in the topology of a protein given by the sequence of amino acids. However, the chains are affected by loop closure constraints due to structural features induced by molecular interaction such as hydrogen bonds or disulphide bonds. Thus, the mechanical model of a protein is a complex multi-loop with many DOF, similar to the one illustrated by the simple planar example of Figure 3.

### B. Rigid groups of atoms

The bodies of the articulated molecular model are formed by groups of rigidly bonded atoms. These groups may have very different size. Taking advantage of a known secondary structure (see Figure 4), $\alpha$-helices and $\beta$-sheets are often considered as rigid elements by molecular modeling approaches. Rigid groups of atoms can be even larger, for example when they involve all the secondary structure elements in a domain. In contrary, in flexible protein portions such as protein loops, the rigid groups are much smaller and they only concern a few atoms inside a residue, as illustrated in Figure 2.

Following a classical robot kinematics approach, a cartesian coordinate frame can be attached to each rigid body. The relative location of consecutive frames is defined by an homogeneous transformation matrix that is a function of the rotation angle between them. The position of an atom in a rigid group with relation to the corresponding frame is simply defined by a vector. A similar modeling approach was proposed in [19].

### C. Steric model

CD applied to molecular model is aimed to act as a geometric filter that rejects conformations with prohibitively high van der Waals (VDW) energy. The filter needs to be as selective as possible, but not at the cost of rejecting valid conformations. We discuss below how the energetic constraints can be translated into geometric distance constraints between the atom positions.

The VDW interaction between atom $i$ and $j$ depends on their relative distance, $d$, and on an equilibrium distance, $d_0$, determined by the type of the two atoms. The associated force

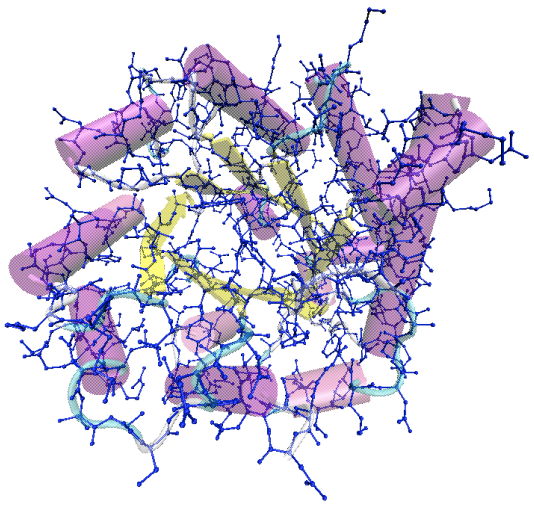

Fig. 4. Secondary structure elements of a protein model.

is slightly attractive at medium distances, null at $d = d_0$ and exponentially repulsive at short distances. Also, there exists a VDW energy limit that cannot be compensated by any other energy component. That energy is reached at a fraction $0 < \rho < 1$ of the equilibrium distance ($\rho \approx 0.8$). Therefore, the collision detector must reject any conformation for which $d < \rho \cdot d_0$ for any atom pair.

Two molecular constraints need to be handled by the collision detector to avoid rejecting valid conformations. First, VDW interactions that only concern non-bonded atoms are not relevant between atoms separated by three or less chemical bonds. Therefore, only pairs of atoms separated by four or more bonds have to be considered for collision. This is one of the particularities that any molecular CD must take into account. The evaluation of the distances between topologically close atoms is in consequence useless, and must be avoided as far as possible instead of computing all interactions and then filtering the relevant one in a postprocessing stage.

Another specific constraint of molecular applications is that the collision distance $\rho \cdot d$ depends on the type of the two interacting atoms. For example, this is necessary to model the presence of hydrogen bonds that shorten the equilibrium distance between specific atom pairs. In summary, the collision criterion can be stated as :

$$d < M_{t(i),t(j)} \text{ and } TopDist(i,j) > 3 \qquad (1)$$

where $M$ is a square symmetric matrix of threshold distances indexed by the atom types and $TopDist(i,j)$ is the chemical topological distance between the atoms $i$ and $j$, i.e., the *minimal* number of chemical bonds that separates them. Note that a variable-geometry criterion such as the matrix condition in (1) is more general than the usual one in CD literature.

## III. Collision detection for molecular chains

This section states the motivation of our technique described in the rest of the paper in relation to the closest related work : techniques developed for the efficent maintenance and self-collision detection of large kinematic chains [7], [13], [14] and the classical Grid algorithm [15] widely used in computational biology for finding interaction pairs in large molecular models.

### A. Related work

ChainTree [13], [14] and the algorithm for deforming necklaces [7] both consider long serial linkages composed of $n$ links and $n-1$ joints. Both techniques also rely on Bounding Volume Hierarchies (BVH) based on the chain topology, i.e. hierarchies built by grouping together *topologically* closed elements of the chain. This kind of hierarchies allows to perform collision detection in $O(n^{4/3})$ for well-behaved chains [2]. The approach in [7] considers problems for which all links move simultaneously in a continuous way. In such situations, the worst case maintenance time for updating the chain is $O(n\ log\ n)$, but in practice it becomes close to linear complexity under smooth deformation of the chain.

ChainTree was designed to speed up Monte-Carlo Simulations (MCS) that apply *controlled changes* to a *few* randomly chosen DOF in each step. It exploits that only a few ($k << n$) DOF's change in each MCS step (and also the linear form of the chain) to identify the parts of the molecule that remain rigid. The update phase takes advantage of this to only process the strictly concerned nodes of the hierarchies, allowing a total update time of $O(k\ log\frac{n}{k})$ that never exceeds $O(n)$. This is also used during the collision detection phase to avoid checking atom pairs not affected by the DOF change. Thus, only new collisions are reported at each step.

In addition to the techniques above, the classical Grid algorithm from computational biology can be used to find interacting atom pairs in linear time by indexing the atoms in a regular grid using a hash table.

### B. Our Contribution

Our aim is to propose an efficient technique adapted to the model and requirements discussed in Section II, and also that operates well on what we dub *sampling-based motion planning* (SBMP) conditions. These conditions can be enunciated as follows:

- Only a pre-selected set of $k$ DOF change in each iteration. This set, obtained from the structural knowledge of the chain (and possibly other biological knowledge), is always the same (or at least the same for many iterations).
- The configuration before each iteration is *arbitrary*, and *the selected DOF change arbitrarily*.
- The number of DOF is limited, but much larger than for MCS conditions (from a few tens to one hundred).

None of the method in related works is satisfactory enough for the requirements listed above. First, we do not want to

[2]The well-behaved assumption means the the chain cannot contain two spheres whose centers are closer than a small fixed distance.

limit us to self-collision of linear chains. We need a more general tool for testing both self-collision and collision between molecular chains without restrictions about their topology. Also, the expected continuity of the deforming necklaces and the incremental collision detection of ChainTree does not fit well with random sampling methods. We need to know if a configuration resulting from a radical change in the selected DOF is in collision, independently of the colliding pairs in common with the previous query. Grid satisfies the constraints above. However, it is unable to exploit the fact that, when limited DOF are changed, most of the distances between atoms remain constant. This ability makes a great difference in practice, and is only well addressed by ChainTree. However, since our SBMP conditions assume that the same DOF change from one iteration to another, the exploitation of the permanence of the rigid groups can be simpler. This may allow to build hierarchies with better properties. Finally, none of the previous methods consider the two particularities of the collision criterion (1) to deal with them efficiently.

## IV. The BioCD Algorithm

### A. Two-level hierarchy

The algorithm relies on a two-level hierarchy organized around the concept of *rigid groups* of atoms, for taking advantage of the SBMP conditions stated above: only a set of $k$ DOF is allowed to change while all the other remain blocked. The preselected DOF may change *occasionally* (when defining a new motion planning problem on the molecular chain), but *many* CD queries will be performed with the same set of selected DOF. While the selected DOF do not change, many atoms in the molecular chain do not undergo any relative displacement with respect to other atoms. The two-level hierarchy allows a simple way to avoid useless tests between such pairs of atoms.

A rigid group is defined as the maximal set of connected atoms in which no internal distance can change. The rigid groups can have very different sizes depending on the selected DOF (see Section II). The smallest rigid groups correspond to the bodies of the articulated residue model. Larger atom groups are created along the rigid backbone segments of the chain, e.g. $\alpha$-helices or $\beta$-sheets of the secondary structure. Also note that nearby rigid groups whose relative positions remain fixed (e.g. $\beta$-sheet pairs) are gathered into one group. Figure 5 illustrates how the atoms of a protein segment form the rigid groups depending on the selected DOF.

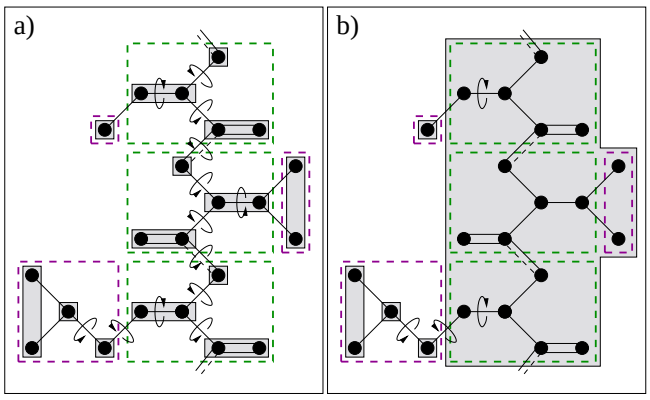


Fig. 5. Representation of a fully articulated protein segment (a) and the same segment with only two articulated side-chains (b). The plain grey boxes contain the rigid groups of atoms. The dashed boxes correspond to the basic atom groups handled by BioCD to check short-range interactions (see explanations in Section V-A).

BioCD identifies the rigid groups of the molecular model and builds a hierarchy for each of them as explained next. These are the *low-level* hierarchies. Each of them arranges the atoms of the rigid group and the root represents the group itself. The *upper-level* hierarchy arranges the roots of the low-level ones. Discarding the atom pairs whose interaction does not change from one iteration to another (i.e., that take place inside a rigid group) is simply performed by not testing the root node representing the rigid group with itself.

The two-level hierarchy is induced by the requirement to have a single node representing the each whole rigid group of atoms. This is the best way to exclude tests between atoms with fixed relative position. It also allows to nearby isolate the parts of the hierarchy that must not be rebuilt (see Section **??**), maximizing its size in a way that a flag in a one-level hierarchy would not be able to do.

### B. Spatially-adapted hierarchies

Both hierarchy levels group bounding volumes according to spatial proximity, in a $k$d-tree-like way [3]. The interest of spatially-adapted hierarchies is twofold. First, they allow fast self-collision detection, $O(n)$ in the worst case (see [14]) and also, they are not bounded to any particular topology. However, they require a worst case $O(n\ log\ n)$ building time. As discussed in Section IV-E, the building time is in practice lower under our SBMP conditions.

The hierarchies are represented by binary trees of Axis Aligned Bounded Boxes (AABBs), chosen because they allow very fast overlap tests, while being more tight bounding volumes than spheres. Their purpose is to organize a set of basic AABBs in such a way that the depth of a node is a good indicator of the spatial proximity of its bounded elements.

For the low-level hierarchies, the basic AABBs (i.e. the leaves) are formed by basic atom groups determined by the topology of the molecule (see Section V-A) to efficiently handle the "separated by more than 3 covalent bonds" rule stated in the collision criterion (1). The basic AABBs of the upper-level hierarchy are simply the AABBs of the rigid atom groups.

Each node in the hierarchies owns a list of all the basic AABBs below it, and it also maintains an AABB bounding all the basic AABBs of its list. The root node contains all the basic AABBs to be organized by the hierarchy.

### C. Lazy hierarchy building

Both levels of hierarchies are built (and updated) using a lazy procedure. Only the roots are computed in a first stage, then the hierarchies are partially (re)-built depending on the need for solving the CD queries.

**Split**($N$)
//Splits node $N$ into two children.

```
1  SplitDim := Max_Dimension(N.AABB)
2  SplitVal :=
     (N.AABB.max[SplitDim] - N.AABB.min[SplitDim])/2
3  LList := {AABB∈ N.listAABB | (N.AABB.max[SplitDim]
               - N.AABB.min[SplitDim])/2 < SplitVal }
4  RList := {AABB∈ N.listAABB | (N.AABB.max[SplitDim]
               - N.AABB.min[SplitDim])/2 ⩾ SplitVal}
5  N.left:= Create_New_Node() // creates left child
6  N.right:= Create_New_Node() // creates right child
7  N.left.listAABB:= LList
8  N.right.listAABB:= RList
9  N.left.AABB:= Bound(LeftList)
10 N.right.AABB:= Bound(RightList)
```

Fig. 6. Node split procedure.

The building procedure first checks which rigid groups have moved since the last collision request (all rigid groups for the initial building). The "static" ones keep their old hierarchy. The hierarchies of the other are destroyed. Then a new root is created for each of them by first updating all the basic AABBs and recomputing their bounding AABB that is assigned to the rigid group root. Afterwards, the upper-level hierarchy is destroyed, and a new root is created, with an associated bounding AABB computed from the roots of the first level.

After this stage, the development of the hierarchies is intertwined with the CD query. The splitting procedure is launched on demand of the CD algorithm (see next section) to develop the hierarchies only in the partitions of the space requiring finer resolution.

The principle of the splitting mechanism is outlined in Figure 6. To split a node, the widest dimension of its AABB is selected jointly with a split value. All the basic boxes whose center component in the splitting dimension is less than the split value are owned by the left child and the remaining ones by the right child. The splitting value may be any one such that the two children own the same number of basic boxes, producing a perfectly balanced tree. We prefer, however, to simply take the middle value of the splitting dimension, because it produces more discriminating hierarchies. Moreover, since the rigid groups are generally compact sets of atoms, the hierarchies with this splitting value remain very reasonably balanced. When one of the boxes occupies more than half the length of the node BB splitting dimension, that box goes alone with the left child and all the other boxes are owned by the right one, thus avoiding infinite recursion when one of the boxes occupies completely the largest dimension of the node.

### *D. Collision detection*

The CD phase follows a standard algorithm to test the collisions between two BVHs. The algorithm first starts with the upper hierarchy of the concerned molecules and tests whether their roots boxes overlap. For self-collision, the root of the molecule is checked against itself. In the absence of overlap the algorithm simply returns that there is no collision. When the roots do overlap, it splits the node with the largest number of basic AABBs (if not split yet) and continues testing recursively the children with the other, smallest node. The algorithm skips the test of a node with itself. When this node is not a leaf, the auto-test is substituted by the three children combination tests (left-left, right-right and left-right). When the node is a leaf, then the auto-test is simply discarded, because it corresponds to the collision of a rigid group with itself, which must be avoided. Finally, a test between two different leaves whose AABBs overlap, launches a call to the collision detection algorithm between the two low-level hierarchies.

The test of two low-level hierarchies is done in a similar recursive way. However, when the algorithm reaches a leaf-leaf pair, a fast test is made to check whether the interactions between the two associated subsets of atoms have been already treated by the functions in charge of computing relevant short range interactions (see Section V-A). If the answer is positive, nothing is done. Else, all the combinations between the atoms of the two subsets are tested for collision.

Simple variants of this basic algorithm allow to integrate other operation modes in BioCD, e.g. to report all the colliding atom pairs, to retrieve all the atoms whose surface is at a distance minor than a constant, or to compute the closest pair of atoms. When a conformation is checked to be in collision, this last mode allows to determine the more interpenetrating pair of atoms, which can be a useful information for the motion planning algorithms.

### *E. Performance Analysis*

We next discuss our choice of spatially-adapted hierarchies and show that the worst case $O(n \log n)$ updating time pointed in [14] can be lower in practice for our SBMP applications.

Let us consider that the $k$ preselected DOF for a kinematic chain of length $n$ create $O(k)$ rigid groups with a size of $O(\frac{n}{k})$. Building the upper hierarchy for the $O(k)$ rigid groups therefore requires $O(k \log k)$ in worst case (but generally less in practice because of the lazy building mechanism). Now concerning the low-level hierarchies, the first query requires to build in worst case all the $k$ hierarchies, each of size $O(\frac{n}{k})$. Then, the initial building takes $O(n\ log\frac{n}{k})$.

It is however important to note that the following updates can be performed much more efficiently in practical situations for which the displaced atoms between two queries are associated to small rigid groups. This is for example the case when the protein model has a rigid secondary structure with large rigid segments involving $O(n\text{-}k)$ residues, connected by few flexible loops made by $O(k)$ residues. In such situations, only the $O(k)$ small hierarchies have to be rebuilt, each having a constant complexity (one residue). Then, the total rebuilding time for all changed hierarchies decreases to $O(k)$.

In such situations, the overall update time of the algorithm then decreases to $O(k + k \log k)$. This relates favorably to the good update time of chain-aligned hierarchies used in ChainTree [14] which requires $O(k\ log\frac{n}{k})$ update time. However, for situations were $k \approx n$ DOF are changed, the

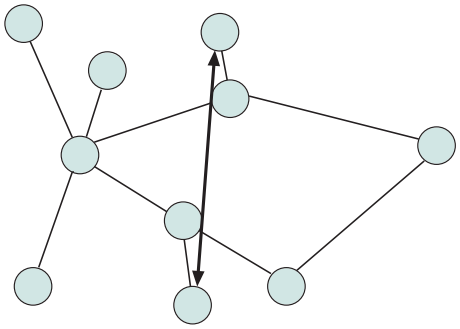

Fig. 7. Circles represent atoms and fine lines between atoms represent covalent bonds. Only the interaction indicated by the bold arrow must be considered for this exemple.

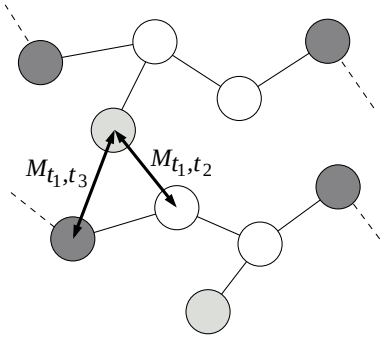


Fig. 8. Collision distance depends on the atom types.

worst case of our updating phase remains $O(n \log n)$. Then, when most DOF of the chain are selected, the update of our two-level spatially-based hierarchy is less efficient than with ChainTree whose update complexity never exceeds $O(n)$. This is however compensated by the efficient $O(n)$ collision detection stage of spatially-adapted hierarchies, instead of $O(n^{4/3})$ with ChainTree.

## V. Molecular CD specificities

This section deals with the two particular conditions for collision on molecular models discussed in Section II-C. These specificities have conditioned certain choices for the algorithmic design of BioCD, in particular the choice for the basic atom groups corresponding to the leaves of the low-level hierarchies.

### A. Short-range interactions

When checking self-collision in a molecular model, pairs of atoms separated by less than four covalent bonds must be disregarded (see Figure 7 for illustration). In the following, we call such pairs of atoms *excluded* pairs. However, it is not obvious how to integrate this rule within algorithms based on hierarchical bounding structures. A simple but inefficient solution is to check all atom pairs, asking afterward if each detected interaction belongs to the very long list of excluded ones. Our solution is much more efficient thanks to a particular choice of the basic elements handled in the BioCD structures (i.e. the leaves).

In a natural way, the basic elements handled by BioCD nearly correspond with the basic components of macro-molecules. In the case of proteins, each amino acid residue $r$ is divided into two basic atom groups. One group, which we refer to as *pseudo-backbone* $B_r$, involves all backbone atoms and the first side-chain atom ($C_\beta$); the rest of the atoms is grouped into the *pseudo-side-chain* $S_r$. There are only two exceptions to this rule for grouping protein atoms: 1) The proline cycle is considered rigid, and thus all the atoms are included in the pseudo-backbone unit. 2) In a disulphide bond, the two cystein pseudo-side-chains form one only unit. The dashed boxes in Figure 5 shows how this partition method is applied on a protein segment.

Using this basic partition of a protein, excluded atom pairs only appear when checking collisions:

- Between the pseudo-backbone group and the pseudo-side-chain group of a same residue ($B_r$ and $S_r$).
- Between the pseudo-backbone groups of two consecutive residues ($B_r$ and $B_{r+1}$).
- Inside a pseudo-backbone or a pseudo-side-chain, only if it is articulated.

Indeed, most atoms pairs in the three cases are excluded pairs. During the inizialization of BioCD, a list of non-excluded atom pairs is created for each couple of connected basic groups $\{B_r,S_r\}$ and $\{B_r,B_{r+1}\}$ belonging to different high-level leaves, and for each articulated group. Collision detection (or distance queries) in such cases is treated by a Short-Range Interaction function (SRI) that symply tests collisions in the corresponding list of non-excluded atom pairs, instead of making a systematic test like in the rest of the cases.

### B. Collision detection between different atom types

For a better accuracy of the geometric filtering of molecular conformations, BioCD checks distances between atom pairs that depend on the atom types. (see Section II-C). Figure 8 shows an example with three different atom types. This condition does not imply a particular difficulty for collision detection, however, it affects the definition of the BBs. The AABBs used by BioCD have to be defined in a different way than in classical collision detection algorithms that treat geometric primitives with constant size.

The AABBs must be large enough to safely exclude the possibility of collision if there is no overlap between them. This condition can be simply guaranteed using the following equations to compute the lower and upper bounds of the AABB associated to each node $N$:

$$N.\text{AABB}.min[d] = \min_i\{X_i[d] - R_{t(i)} |\ i \in N\} \quad (2)$$

$$N.\text{AABB}.max[d] = \max_i\{X_i[d] + R_{t(i)} |\ i \in N\} \quad (3)$$

$X_i[d]$ is the position of the atom $i$ in dimension $d = \{x, y, z\}$. $R_{t(i)}$ is a size associated with the type of atom $i$, and it is calculated as:

$$R_{t(i)} = \max_{t(j)}\{M_{t(i),t(j)}/2\} \quad (4)$$

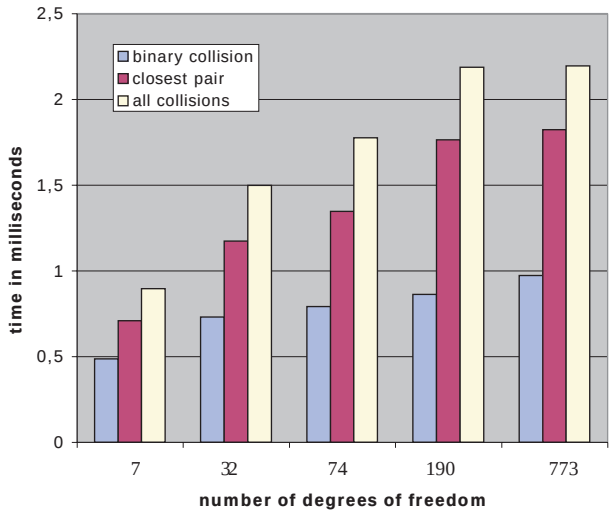


Fig. 9. Performance of BioCD with different number of DOF.

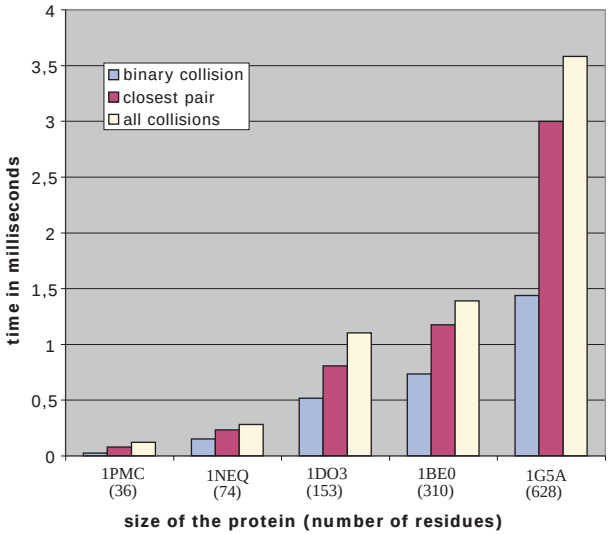


Fig. 10. Performance of BioCD with proteins of different size.

where the $t(j)$ are all the atom types. This is equivalent to cautiously assume, for the AABBs construction, a fixed size for each atom type corresponding to the maximum it can take in any context. Since for an atom type $t(i)$ all the values of $M_{t(i),t(j)}$ are quite similar, such conservative AABBs are sufficient. The exact collision test using the distance values in $M$ is only performed for the atom pairs tested at the leaf-leaf level.

## VI. Experiments

In this section we present experimental results that show the performance of BioCD for solving three types of problems:

- Determine whether a conformation is in collision or not.
- Compute the closest pair of atoms.
- Report the list of all colliding atoms pairs.

The indicated times were obtained on a 2.5 Ghz G5 Apple computer and were averaged over 10.000 randomly sampled conformations. Also remember that $\rho$ is the fraction of the VDW equilibrium radius allowed for penetration. The experiments were carried out with $\rho = 0.8$, a value typically used in applications because it has a chemical significance.

Figure 9 compares the performance of BioCD for testing the self-collision of a medium size protein (1DO3, 153 amino acid residues) with different number of free DOF. Five different sets of preselected DOF were chosen, freeing completely the internal DOF of 1, 5, 10, 35 and 153 residues in each case. For 1 residue this corresponds to 7 effective DOF, and for the 153 residues data to 773 effective DOF. The free residues were always selected at regular intervals along the chain. For example, in the case of 1 free residue, it is chosen as the middle one of the chain, and in the case of 35 free residues, 3 blocked residues are systematically followed by one free residue. The graphic shows that BioCD scales very gently with the increase in the number of DOF. We can see that computing time when all the DOF of the molecule are free is no more than twice the time required for only one free residue. Also note that the times to report all collisions or to get the closest pair of atoms remain very similar and do not require more than twice the time of binary collision detection in the case of the fully articulated test. This is a very good performance specially if we consider that the random conformations were sampled inside the full range of dihedral angles, resulting in a very high number of average collisions per conformation. For such cases, the recursion must test many leaves to get the complete list of colliding pairs (in the order of thousands for the fully articulated test). It is also important to mention that when testing collision-free conformations the performance of the algorithm slightly improves (factor up to 2) compared to the times reported in the figure.

Figure 10 compares the performance for several proteins of increasing size, tested with the same number of DOF. The protein sizes span from small to medium-high: 36 (1PMC), 74 (1NEQ), 153 (1DO3), 310 (1BE0) and 628 (1G5A) amino acid residues. The proportion between the size of any two consecutive proteins in the graphic is approximately 2. Thus, the scale of the abscissae is logarithmic on the size of the protein. The rate between the collision detection time for the first protein and second one is 6, while the rate between the fourth and the last is only 2. Thus, at least for these sizes of proteins, the computation time is almost linear in practice. For larger sizes, the time to find all the colliding pairs (and also the closest pair) is however expected grow faster since the number of collisions to be reported (or tested) grows with the size of the protein.

The comparison of the tests above with ChainTree results shown in [14] seem to confirm a much lower dependence of BioCD computing time with the number of DOF simultaneous changes. Indeed, while a performance slow-down of a factor of 30 is indicated in [14] when the number of DOF increases from a few to one hundred, Figure 9 shows a much lower

performance degradation with BioCD. For only few DOF changes, results in [14] (obtained on a 400MHz Sun Ultra Enterprise with 4GB RAM) seem to indicate a slightly better performance of ChainTree. This should be counterbalanced by the fact that ChainTree tests did not consider the side-chains of the molecular model. Consequently, for a same protein model, BioCD (which deals with side-chains) handles about twice the number of atoms than ChainTree. Moreover, in the experiments above, BioCD does not take advantage of possible rigid elements of the secondary structure. One could expect some additional speed up for such applications with only some flexible loops and side-chains.

## VII. Conclusion

Macromolecules such as proteins are complex mechanical systems and their motion analysis is a highly challenging problem due to their huge number of DOF. MCS techniques classically used in molecular modeling search the conformational space by slightly perturbing a few DOFs randomly chosen at each step. Another promising way to tackle such problems is to use motion planning techniques while considering the most relevant DOF.

In this paper, we described a new BioCD algorithm specially designed for sampling-based motion planners applied to molecular models. It assumes that molecules are composed of a set of rigid elements determined by a limited number of selected DOF. This permits the efficient use of two-level data structure based on spatially-adapted hierarchies. The algorithm is general, in the sense that it does not assume particular topologies of the kinematic chains, while allowing good collision and self-collision detection times. BioCD integrates in an efficient way the particularities imposed by the biomolecular context on collision detection. It is also very versatile and offers several operation modes allowing to get useful information (e.g. distances or closest atoms pairs) for the motion planning algorithms. Our experiments performed with real protein models shows a very good performance for reporting self-collision in large protein models, with only a limited overhead for also computing distances on all the colliding atom pairs.

There remain several possible improvements. In particular there are ways to improve the update time of the hierarchies that must be compared. Although the current implementation of BioCD is tailored to protein models, the approach can be extended to other kinds of macromolecules such as DNA or RNA.

## Acknowledgment

This work was partially supported by the BioMove3D project of the French Bioinformatic program of CNRS, the IST European Project 39250 MOVIE (Motion Planning in Virtual Environments) and the DPI 2004-07358 I+D project funded by the Spanish Ministry of Education.

## References

[1] N.M. Amato, K.A. Dill and G. Song, *Using Motion Planning to Map Protein Folding Landscapes and Analyze Folding Kinetics of Known Native Structures*, Journal of Computational Biology, 10, 149-168, 2003.

[2] M.S. Apaydin, D.L. Brutlag, C. Guestrin, D. Hsu and J.-C. Latombe, *Stochastic Roadmap Simulation: An Efficient Representation and Algorithm for Analyzing Molecular Motion*, Proc. RECOMB, 12-21, 2002.

[3] J.L. Bentley, *Multidimensional Divide and Conquer*, Communications of the ACM, 23(4),214-229, 1980.

[4] G. van der Bergen, *Collision Detection in Interactive 3D Environments*, Elsevier Eds, 2004.

[5] J. Cortés, T. Siméon, D. Guieysse, M. Remaud-Siméon and V. Tran, *Geometric Algorithms for the Conformational Analysis of Long Protein Loops*, Journal of Computational Chemistry, 25, 956-967, 2004.

[6] J. Cortés, T. Siméon, V. Ruiz de Angulo, D. Guieysse, M. Remaud-Siméon and V. Tran, *A Path Planning Approach for Computing Large-Amplitude Motions of Flexible Molecules*, Int. Conf. on Intelligent Systems for Molecular Biology (ISMB 2005), submitted.

[7] L.J. Guibas, A. Nguyen, D. Russel and L. Zhang, *Deforming Necklaces*, Symp. on Comp. Geometry, 33-42, 2002.

[8] P. Jimenez, F. Thomas and C. Torras, *3D Collision Detection. A Survey*, Computers and Graphics, 25(2), 269-285, 2001.

[9] L.E. Kavraki, P. Svestka, J.-C. Latombe and M. OverMars, *Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces*, IEEE Transactions on Robotics and Automation,12(4), 1996.

[10] J. Kuffner and S.M. LaValle, *RRT-Connect: An Efficient Approach to Single-Query Path Planning*, Proc. IEEE Int. Conf. on Robotics and Automation, 2000.

[11] J. Kuffner, K. Nishiwaki, S. Kagami, Y. Kuniyoshi, M. Inaba and H. Inoue, *Self-Collision Detection and Prevention for Humanoid Robots*, Proc. IEEE Int. Conf. on Robotics and Automation, 2002.

[12] M. Lin and D. Manocha, *Collision and Proximity Queries*, Handbook of Discrete and Computational Geometry, 2003.

[13] I. Lotan, F. Schwarzer, D. Halperin and J.-C. Latombe, *Efficient Maintenance and Self-Collision Testing for Kinematic Chains*, Proc. 18th ACM Symp. on Computational Geometry, Barcelona, Spain, June 2002.

[14] I. Lotan, F. Schwarzer, D. Halperin and J.-C. Latombe, *Algorithm and data structures for efficient energy maintenance during Monte Carlo simulation of proteins*, Journal of Computational Biology, 11, 902-932, 2004.

[15] J. Moult and M.N.G. James, *An Algorithm for Determining the Conformation of Polypeptide Segments in Proteins by Systematic Search*, Proteins, 1, 146-163, 1986.

[16] J. Pettré, J.-P. Laumond, T. Siméon, *A 2-Stages Locomotion Planner for Digital Actors*, Proc. ACM SIGGRAPH/Eurographics Symp. on Computer Animation, San Diego, California, 2003.

[17] M.L. Teodoro, G.N. Phillips and L.E. Kavraki, *A Dimensionality Reduction Approach to Modeling Protein Flexibility*, Proc. RECOMB, 299-308, 2002.

[18] K. Yamane, J. Kuffner and J. Hodgins, *Synthezizing Animations of Human Manipulation Tasks*, Proc. SIGGRAPH, 2004.

[19] M. Zhang and L.E. Kavraki, *A New Method for Fast and Accurate Derivation of Molecular Conformations*, Journal of Chemical Information and Computer Sciences, 41, 64-70, 2002.