Editing molecular structures with smoothed articulated-body accelerations

Stephane Redon
INRIA
Univ. Grenoble Alpes, LJK, F-38000 Grenoble, France
stephane.redon@inria.fr

***Abstract*—Articulated bodies are commonly employed to model complex physical systems, including robots, humanoid figures, hair, animals, plants, molecules, nanosystems, etc. As a result, Computer-Aided-Design (CAD) tools frequently allow users to construct, edit and simulate articulated-body systems.**

When an articulated-body system contains a large number of degrees of freedom, however, manually editing its configuration becomes tedious, since it becomes unclear how to effectively map user interfaces with few degrees of freedom (e.g. a mouse or a haptic device) to the space of possible articulated-body motions.

In this paper, we introduce a simple, efficient algorithm to tunably *smooth* the acceleration of an articulated body, and demonstrate how this smoothed acceleration may be used to edit the configuration of complex molecular structures.

Our approach may be combined both with internal and external force fields, so that editing an articulated-body system may be performed while taking into account the system's physics. This helps the user design and analyze complex structures, and produce well-formed configurations and trajectories. We demonstrate our approach on several large-scale structural biology examples containing up to a few thousands of degrees of freedom.

I. Introduction

Articulated bodies are commonly employed to model complex physical systems, including robots, humanoid figures, hair, animals, plants, molecules, nanosystems, etc. As a result, Computer-Aided-Design (CAD) tool frequently allow users to construct, edit and simulate articulated-body systems.

Among physical systems that may be modeled as articulated bodies, perhaps some of the most computationally challenging arise from structural biology and nanoscience, where molecular models may contain up to several thousands of degrees of freedom. Despite significant recent progress, it is still a challenging problem to easily perform complex structural modifications on a large molecular system. This is unfortunate, since it appears that such a capability would have many applications, including protein structure analysis, simulations and minimization setups, analysis of macromolecular motions and transitions, analysis of signal transduction, hypothesis building, corrections of invalid structural models, fitting cryo-electron microscopy maps, etc.

To our knowledge, however, few methods have been proposed to intuitively deal with large molecular systems which may contain several thousands of degrees of freedom. Powerful interactive molecular dynamics systems (*e.g.* [6]) enable the user to act on a simulation and modify structures. However, such modifications often have *local* effects, and it is not clear how large-scale conformational changes may take place, especially on the time scales typically accessible to molecular dynamics simulations. Inverse kinematics may produce large scale structural changes, but typically act on a limited number of joints at any given time, and are not straightforwardly compatible with forward dynamics or quasi-statics (*i.e.* structure minimization) [2]. Finally, although some earlier pioneering methods have allowed multiresolution editing of molecular structures *via* multilevel analysis [1], these methods were only based on the geometry of the molecular structure, and not on an underlying force field, which could lead to invalid conformations (*e.g.* steric clashes) during modification of the model.

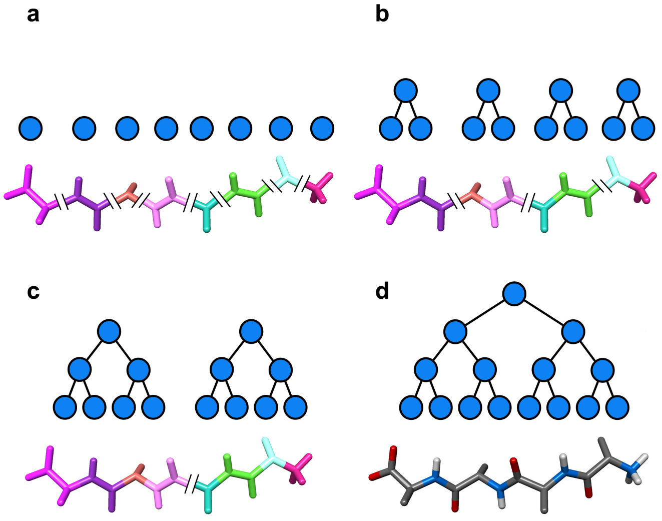

Fig. 1. **Recursive assembly of a tetra-alanine**. The *assembly tree* describes the sequence of assembly operations. **a**: the eight rigid bodies correspond to the leaf nodes of the assembly tree. **b**: pairs of rigid bodies are formed. **c**: two sub-articulated bodies with four rigid bodies each. **d**: the complete tetra-alanine model. This recursive description can handle any acyclic, branched molecule (see Sections II and IV).

Essentially, the difficulty lies in the *mapping* of user interfaces, which typically have a few degrees of freedom only, to complex dynamical systems, which may contain several thousands of degrees of freedom, while retaining some of the underlying quasi-statics or dynamics of the system.

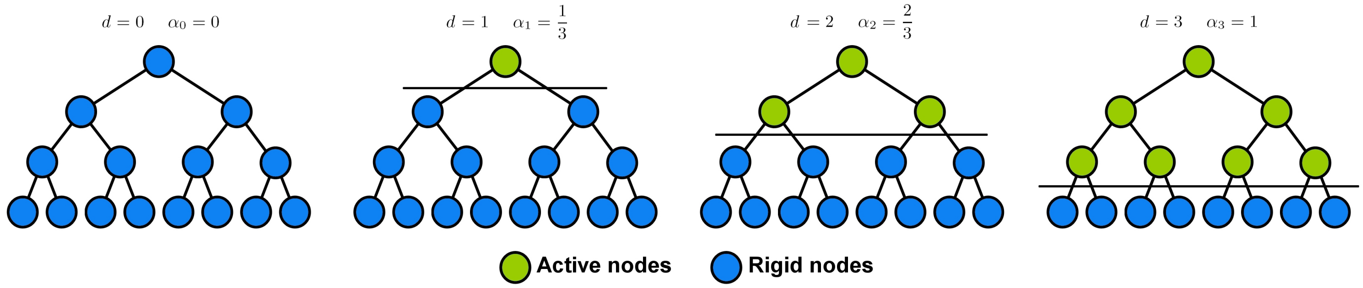
$d = 0 \quad \alpha_0 = 0$
$d = 1 \quad \alpha_1 = \frac{1}{3}$
$d = 2 \quad \alpha_2 = \frac{2}{3}$
$d = 3 \quad \alpha_3 = 1$
Active nodes
Rigid nodes

Fig. 2. **Progressive motion simplification**. For a given assembly tree of depth D (here, $D = 4$), we define a series of D hybrid trees of degree d, $0 \leqslant d \leqslant D - 1$, mixing active and rigid joints. When $d = 0$, all joints are rigid, and the molecular system behaves like a floating rigid body. When $d = D - 1$, all joints are active, and the motion of the molecular system is the original, unconstrained one (see Section III-A).

To address this problem, we introduce a simple but effective multiresolution editing method, which allows a user to interactively modify complex molecular structures. Our approach enables both local and global, large-scale structural modifications. The user may finely tune the effects of applied forces during the interaction to obtain the desired deformations. Furthermore, our approach handles underlying molecular force fields, so that the molecular system being modeled is continuously being minimized. This helps the user design and analyze complex molecular systems and transitions, and produce well-formed configurations and trajectories. We demonstrate our approach on several examples with up to a few thousands of degrees of freedom.

II. Divide-and-Conquer forward dynamics

Our algorithm is based on Featherstone's Divide-and-Conquer Algorithm [4, 5], which computes forward dynamics of articulated bodies. For completeness, we begin by providing a brief overview of this algorithm.

In the DCA, an articulated body is built by recursively assembling sub-articulated bodies. The sequence of assembly operations is described in a binary *assembly tree*, in which leaf nodes represent rigid bodies, internal nodes represent both sub-articulated bodies and the joint connecting them, and the root node represents the complete articulated body. A joint can have any number of degrees of freedom, so that this definition may accommodate both torsion-angle representations as well as more general representations (i.e. including varying bond angles and bond lengths) and groups of molecules [9]. Furthermore, any acyclic, branched articulated structure can be represented this way, so that we can model side-chain mobility as well[1]. Figure 1 demonstrates this approach for a tetra-alanine with eight degrees of freedom (the ϕ and ψ torsion angles).

[4, 5] shows that the dynamics of any sub-articulated body can be described by the following *articulated-body equation*:

$$\mathbf{a} = \mathbf{\Phi f} + \mathbf{b}, \tag{1}$$

where $\mathbf{a}$ is the composite acceleration of the articulated body (a vector which concatenates the bodies accelerations), $\mathbf{\Phi}$ is the composite inverse inertia of the articulated body, $\mathbf{f}$ is a composite kinematic constraint force (which holds the articulated body together), and $\mathbf{b}$ is a composite bias acceleration, due to external forces and torques. In this work, external forces and torques include all inter-atomic forces and torques (*e.g.* van der Waals and electrostatic forces, as well as dihedral torques, see Section IV). However, we assume that the user edits the structure of the molecular system in a "quasi-statics" mode (infinite friction assumption), so that we do not include velocity-dependent (Coriolis) terms in the bias accelerations. Featherstone's algorithm computes the forward dynamics of the articulated body in two passes over the complete assembly tree. During the *main pass*, the inverse inertias and bias accelerations of each node are computed from the bottom up. The coefficients of the leaf nodes (the rigid bodies) are:

$$\mathbf{\Phi} = \mathbf{I}^{-1} \qquad \mathbf{b} = \mathbf{I}^{-1}\mathbf{f}_k, \tag{2}$$

where $\mathbf{I}$ is the 6×6 spatial inertia of the rigid body, and $\mathbf{f}_k$ is the external force applied to the rigid body. Then the articulated-body coefficients of non-leaf nodes are recursively computed from those of their children. Precisely, if a sub-articulated body C is formed by assembling two articulated bodies A and B, then the dependencies between coefficients are as follows:

$$\mathbf{\Phi}^C = \mathbf{\Phi}^C(\mathbf{\Phi}^A, \mathbf{\Phi}^B) \qquad \mathbf{b}^C = \mathbf{b}^C(\mathbf{b}^A, \mathbf{b}^B, \mathbf{Q}^C), \tag{3}$$

where $\mathbf{Q}^C$ is a torque applied to the joint connecting A and B. When all articulated-body coefficients have been computed, a top-down *back-substitution pass* recursively computes the kinematic constraint forces $\mathbf{f}^A$ and $\mathbf{f}^B$ (which hold A and B together) and the acceleration $\ddot{\mathbf{q}}^C$ of the joint connecting A and B based on $\mathbf{f}^C$, $\mathbf{\Phi}^A$ and $\mathbf{\Phi}^B$:

$$\mathbf{f}^A = \mathbf{f}^A(\mathbf{f}^C) \quad \mathbf{f}^B = -\mathbf{f}^A \qquad \ddot{\mathbf{q}}^C = \ddot{\mathbf{q}}^C(\mathbf{\Phi}^A, \mathbf{\Phi}^B, \mathbf{f}^C), \tag{4}$$

starting with $\mathbf{f}^C = \mathbf{0}$ for the root node (since the full articulated body is floating). The six-dimensional spatial acceleration $\mathbf{a}^r$ of the root node is computed during this top-down pass as well, so that the motion of the complete articulated body (molecular system) is entirely known when the top-down pass completes. For convenience, we denote this motion by $(\mathbf{a}^r, \ddot{\mathbf{q}})$, where $\ddot{\mathbf{q}}$ concatenates all joint accelerations.

[1] Note that the resulting assembly tree is not necessarily perfectly balanced, but we attempt to make it as balanced as possible by recursively cutting the kinematic graph of the molecular system in half, until we reach rigid bodies.

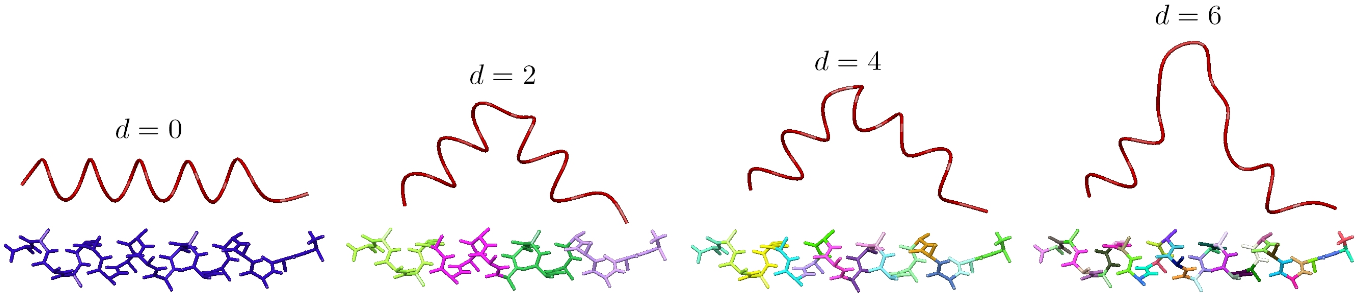

Fig. 3. **Pure hybrid motions of a 20-alanine**. A constant force is applied to the mid-residue of a 20-alanine for one thousand steps. The bottom row shows the amount of rigidification depending on the depth of the hybrid body (one color per rigid body), while the top row shows the corresponding final configuration of the 20-alanine after one thousand steps. Increasing the depth of the hybrid body results in a localized deformation, which unrolls the helix (see Section III-A).

III. Multiresolution editing

A. Hybrid motions

As mentioned above, applying a force on an atom of the molecular system often results in local structural modifications, which makes it difficult to interactively perform large-scale deformations. In order to allow the user to smoothly adjust between local and global structural modifications, we propose to combine *hybrid motions*, *i.e.* motions computed when only part of the joints are allowed to move, and the others are considered rigid.

Let *hybrid body* denote an articulated body in which some joints have been rendered rigid. We have used hybrid bodies in our work on adaptive torsion-angle quasi-statics [9]. In the adaptive algorithm, the set of active joints in a hybrid body can be *any* sub-tree of the assembly tree. In this work, however, our goal is not to determine and focus on the most important joint accelerations (*i.e.* determine the most appropriate sub-tree), since this would result in local structural modifications as well. Instead, we combine several levels of rigidification at each time step to determine the motion of the molecular system. Furthermore, we restrict ourselves to some types of hybrid bodies only.

Let d denote the depth of the assembly tree, and assume the levels of the assembly tree are indexed from 1 (the level which only contains the root node) to D (the level which only contains leaf nodes[2]). We call *hybrid body of degree* d, for $0 \leqslant d \leqslant D-1$, a hybrid body in which the active joints are the joints at levels smaller than or equal to d. When $d = 0$, all joints are rigid, and the molecular system behaves like a floating rigid body. When $d = D-1$, all joints are active, and the motion of the molecular system is the original, unconstrained one. Figure 2 shows the four hybrid trees associated to the tetra-alanine model of Figure 1.

The motion of a hybrid body of degree d can be easily computed. In Featherstone's DCA, a joint motion space is described by a $6 \times k$ matrix $\mathbf{S}$, where k is the number of degrees of freedom of the joint [4, 5]. The matrix $\mathbf{S}$ enters in the computation of the articulated-body coefficients (bottom-up pass) and the joint acceleration and forces (top-down pass). In order to inactivate joints, we can simply set their $\mathbf{S}$ matrix to zero. The bottom-up pass of the DCA is performed as usual, but setting $\mathbf{S} = \mathbf{0}$ in the articulated-body coefficients equations (3). However, the top-down pass can be restricted to the active joints, since the rigid ones cannot move.

Our approach to multiresolution editing relies on a fairly simple observation: when more and more joints become rigid, the effect of applied forces, be they van der Waals, electrostatic, or user-applied forces, tends to propagate throughout the articulated body instead of locally modifying its configuration. Precisely, whereas an active joint absorbs energy by changing its configuration, a rigid joint can only transmit work, so that the action of an applied force can be felt farther away from its point of application.

Figure 3 shows the motion of a 20-alanine depending on the depth of the hybrid body, when a constant force is applied upwards during one thousand steps. As can be expected, deeper hybrid bodies result in localized deformations, while shallower ones tend to smooth the effect of the applied force.

Using pure hybrid motions may be too crude for precise control, however. In order to allow the user to smoothly adjust the effect of an applied force (as well as the effects of the force field used in the quasi-statics minimization), we compute all hybrid motions, and linearly combine them, at each time step.

[2] When the assembly tree is not perfectly balanced, leaf nodes can belong to lower levels as well.

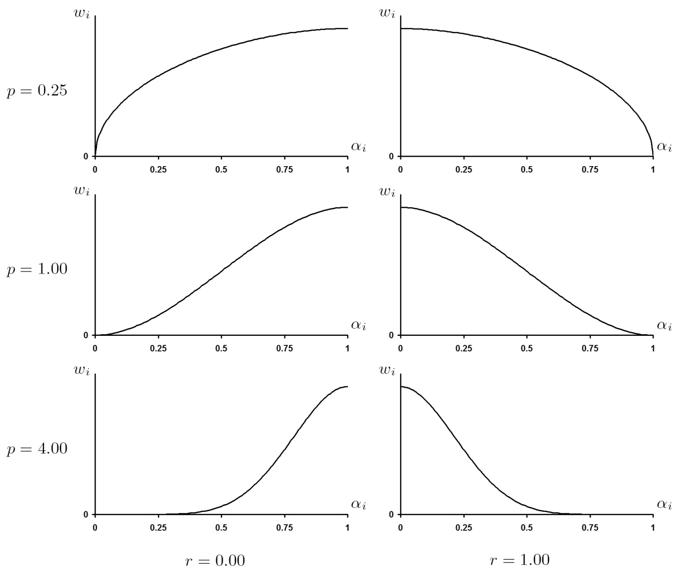

Fig. 4. **Weighing the hybrid motions**. The weight attributed to each hybrid motion depends on two user-defined parameters: the *rigidification factor* r, and the *power coefficient* p (see Section III-B).

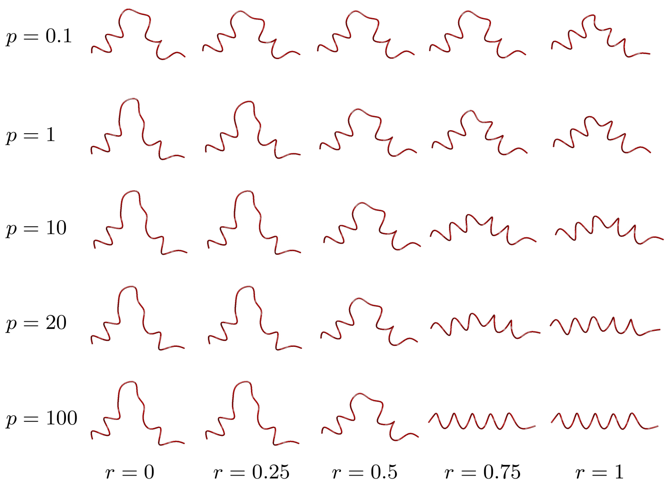

Fig. 5. **Multiresolution editing of a polyalanine**. A constant force is applied to a 20-alanine for one thousand steps. The parameters r and p allow the user to finely tune the effect of a force applied to the structure, and continuously choose between local or global deformations (see Section IV).

B. Multiresolution control

Let $(\mathbf{a}_d^r, \ddot{\mathbf{q}}_d)$ denote the motion[3] of the hybrid body of degree d. Then the combined motion $(\mathbf{a}^r, \ddot{\mathbf{q}})$ is simply:

$$\begin{array}{l} \mathbf{a}^r = \sum_{d=0}^{D-1} w_d \mathbf{a}_d^r \\ \ddot{\mathbf{q}} = \sum_{d=0}^{D-1} w_d \ddot{\mathbf{q}}_d, \end{array} \quad (5)$$

where the w_d, $0 \leqslant d \leqslant D-1$, are weights[4].

In order to provide the user with a simple way to control the effect of an applied force, we compute the weights w_d as follows. For a given depth d, $0 \leqslant d \leqslant D-1$, let $\alpha_d = \frac{d}{D-1}$ denote the *activation ratio* of the hybrid body ($0 \leqslant \alpha_d \leqslant 1$). Then we set

$$w_d = \frac{1}{W}(r + (4r-2)(\alpha_d - 1.5)\alpha_d^2)^p, \quad (6)$$

where r and p are two user-defined parameters, and

$$W = \sum_{i=0}^{D-1} (r + (4r-2)(\alpha_i - 1.5)\alpha_i^2)^p, \quad (7)$$

is a normalization factor.

The parameter r is a *rigidification factor*, which can vary continuously between 0 and 1. When $r = 0$, more weight is given to deeper hybrid bodies (*i.e.* with more active joints), which tends to lead to local deformations. On the opposite, $r = 1$ gives more weight to hybrid bodies with smaller depths, which are more rigid, which leads to larger-scale deformations. The parameter p is a *power coefficient*, used to control the "spread" of the weighing function and, thus, the relative importance of intermediate levels.

Figure 4 plots the weighing function for various values of r and p. Other weighing functions can of course be used, but we have found this one, which provides smoothly varying control with only two parameters, to suit our purpose well (see Section IV).

C. Efficient computation of hybrid motions

All hybrid motions can be efficiently computed at each time step, by taking advantage of the dependencies in the articulated-body coefficients.

Assume the articulated-body coefficients $\mathbf{\Phi}$ and $\mathbf{b}$ have been computed for all nodes of a hybrid tree of degree d. According to the definition of such a hybrid body, all joints whose depth is smaller than or equal to d are active, while the others are rigid.

Assume $d \geqslant 1$. In order to compute the articulated-body coefficients of the hybrid body of degree $d-1$, we have to make the joints at level d rigid. As mentioned before, this is done by setting the corresponding joint motions matrices $\mathbf{S}$ to zero. Because of the coefficients dependencies in the DCA, we then have to update the articulated-body coefficients of all levels $l \leqslant d$. *The coefficients of deeper levels do not have to be updated, however.* This suggests an efficient algorithm to compute all hybrid motions, at each time step, assuming the external forces are known (*i.e.* the underlying force model, including for example van der Waals and electrostatic forces, as well as forces applied by the user). Starting with $d = D-1$

[3] Because part of the joints of a hybrid body are rigid, their corresponding components in $\ddot{\mathbf{q}}_d$ are equal to zero. However, all composite vectors $\ddot{\mathbf{q}}_d$ have the same dimension.

[4] Unlike conventional accelerations, spatial accelerations form a vector space, so that we can linearly combine them [3].

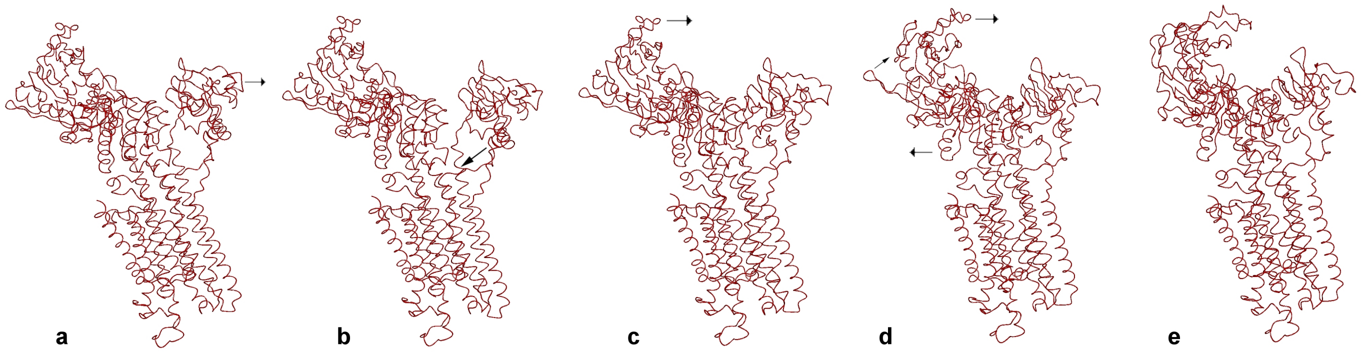

Fig. 7. **Modifying an ATPase model**. Starting from a known structure of a Ca^{2+}-ATPase (PDB code 1SU4), the user attempts to obtain the unbound form of the ATPase (PDB code 1IW0). The arrows indicate the main regions where forces were applied during editing. The multiresolution approach allows the user to displace helices or larger groups of atoms in a mostly rigid way, while the model is being minimized. The ability to tune the parameters during the manipulation allows to adjust for atom groups of varied sizes. The range of parameters used in this example was $0.6 \leqslant r \leqslant 0.9$ and $7 \leqslant p \leqslant 10$. The model contains 9305 atoms and 4103 degrees of freedom. The depth of the assembly tree is 14.

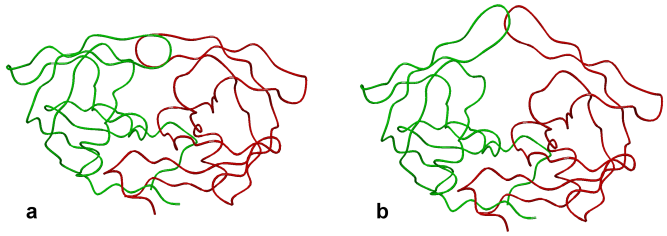

Fig. 6. **Creating an open structure for an HIV protease**. This example shows how, starting from a closed structure of an HIV protease (**a**, pdb code 2AZ8), a user can easily create an open structure with a few mouse clicks (**b**). The model contains two dimers, 1834 atoms, and 816 degrees of freedom. The parameters used in this example are $r = 0.5$ and $p = 10$.

(corresponding to the fully articulated model), the articulated-body coefficients of levels $l \leqslant d+1$ are computed ($d+1$, since the coefficients of the leaf nodes have to be computed as well). The spatial acceleration of the root of the hybrid body and its joint accelerations can then be computed, for all levels $l \leqslant d$, starting from the root. This produces the motion $(\mathbf{a}^r_{d-1}, \ddot{\mathbf{q}}_{d-1})$. The remaining hybrid motions are then iteratively computed, from $d = D-2$ to $d = 0$. For each depth d, only part of the assembly tree is processed: the levels $l \leqslant d+1$. Assuming the assembly tree is balanced, the complexity of the complete algorithm is linear in the number of joints.

IV. Implementation and Results

We have implemented our algorithm in C++, and tested the software on a 1.7GHz laptop computer with 1GB of RAM. The underlying force fields are CHARMM19 [8] and CHARMM22 [7], adapted to our torsion-angle representation [9]. In this section, we demonstrate our approach on several examples[5]. As noted above, we assume the user edits the structure of the molecular system in a “quasi-statics” mode (infinite friction assumption). Our experience with interactive modeling and simulation sessions has indeed shown that it is typically extremely difficult for a user to manipulate virtual structures when inertia is present (and when objects continue to move after having been acted on). We note, however, that the smoothed acceleration could also be computed in the presence of velocity-dependent terms, even though we feel that this would be useful in other types of applications.

A. *Poly-alanine*

In order to demonstrate the influence of the user parameters r and p, we first perform a simple non-interactive test. Figure 5 shows the motion of a 20-alanine when a constant force is applied during one thousand steps, depending on the values of r and p. This example shows that the two parameters effectively allow the user to control the influence of an applied force. Furthermore, in contrast with using pure hybrid motions as in Figure 3, the user can smoothly combine them and finely tune the influence of applied forces.

B. *HIV protease*

Figure 6 shows how a user can use our multiresolution editing method to easily produce an open structure for a HIV protease (Figure 6.b), starting from the closed structure (Figure 6.a — pdb code 2AZ8). The model contains two dimers, 1834 atoms, and 816 degrees of freedom. The parameters used in this example are $r = 0.5$ and $p = 10$. Note that the underlying force model, which includes van der Waals and electrostatic forces, helps the user produce valid, physically-based structures with no steric clashes.

C. *ATPase*

This final example demonstrates large-scale structural deformation of a Ca^{2+}ATPase model (PDB code 1SU4). Figure

[5] Please note that protein side chains are hidden in the renderings for clarity, but they are present in the articulated bodies.

7 shows the main forces applied by the user during editing. Despite the size of the model (9305 atoms and 4103 degrees of freedom, for an assembly tree depth of 14), the multiresolution approach allows the user to displace helices or larger groups of atoms in a mostly rigid way, while the model is being minimized. The ability to tune the parameters during the manipulation allows to adjust for atom groups of varied sizes (the range of parameters used in this example was $0.6 \leqslant r \leqslant 0.9$ and $7 \leqslant p \leqslant 10$). Again, while the smoothing algorithm allows the user to easily perform large modifications to the structure, the underlying force field ensures that no steric clashes are created.

V. Conclusion

We have introduced a novel algorithm to perform multiresolution editing of a molecular system. Our approach combines *hybrid motions*, *i.e.* motions of partially rigidified instances of the molecular system. Our approach allows a user to finely tune the effect of an applied force, from local to more global, large-scale structural deformations. We have tested our method on several systems and shown how the multiresolution algorithm allows a user to easily perform large-scale deformations of complex systems.

In effect, our approach enables tunable mappings between user interfaces with few degrees of freedom, such as a mouse, and a complex dynamical system involving a few thousands of degrees of freedom. This approach is general and can be used with other types of user interfaces, including pressure sensitive devices (pen-based interaction) and haptic interfaces. Furthermore, we note that our algorithm may handle any number of user forces, allowing for multi-finger, multi-hand and multi-user interaction.

We have noted that our approach naturally handles molecular force fields (CHARMM19 and CHARMM22 in our current implementation, although other force fields can easily be included). Editing a structure is thus physically-based: the molecular system continuously attempts to reach the closest energy (local) minimum, and thus avoids steric clashes. We believe that such an integrated approach, combining the ability to perform large-scale deformations with continuous minimization, greatly helps the user model and analyze complex molecular systems.

We believe that smoothing articulated-body accelerations may have many more applications than interactive editing of large molecular structures, though, and we would now like to explore such extensions. For example, producing large-amplitude motions that preserve local structures could be used in energy minimization (to rapidly explore low-energy regions). Moreover, we note that the smoothed articulated-body acceleration produced by our algorithm could be used to smooth the *dynamics* of a molecular system, which might help explore phase space faster.

One limitation of our approach is its reliance on an assembly tree, which introduces a bias in the smoothing of the articulated-body acceleration. For example, the root joint is made rigid in only one rigidification level: when the complete articulated body is rigid. We have not yet attempted to address this, but we believe it may be possible to choose smoothing weights in a different way to compensate this bias.

It is also not clear how the user should choose the smoothing parameters, and whether these parameters would need to be changed for each molecule being deformed (in particular, the parameters where empirically chosen for the demonstrated tests). We believe that the user interface should thus make these parameters readily accessible, such as when the "brush size" is chosen in a painting software.

Acknowledgments

The author thanks the reviewers for their help in clarifying the text, and acknowledges funding from the European Research Council (ERC Grant Agreement n. 307629).

References

[1] Mike Carson. Wavelets and molecular structure. *Journal of Molecular Graphics*, 10(4):273–283, 1996.

[2] S. Crivelli, O. Kreylos, B. Hamann, N. Max, and W. Bethel. Proteinshop: A tool for interactive protein manipulation and steering. *Journal of Computer-Aided Molecular Design*, 2004.

[3] R. Featherstone. The acceleration vector of a rigid body. *The International Journal of Robotics Research*, 2001.

[4] Roy Featherstone. A divide-and-conquer articulated body algorithm for parallel o(log(n)) calculation of rigid body dynamics. part 1: Basic algorithm. *International Journal of Robotics Research 18(9):867-875*, 1999.

[5] Roy Featherstone. A divide-and-conquer articulated body algorithm for parallel o(log(n)) calculation of rigid body dynamics. part 2: Trees, loops, and accuracy. *International Journal of Robotics Research 18(9):876-892*, 1999.

[6] W. Humphrey, A. Dalke, and K. Schulten. Vmd: Visual molecular dynamics. *Journal of Molecular Graphics*, 14(1):33–38, 1996.

[7] A.D.Jr. MacKerell, D. Bashford, M. Bellott, R.L.Jr Dunbrack, J.D. Evanseck, M.J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F.T.K. Lau, C. Mattos, S. Michnick, T. Ngo, D.T. Nguyen, B. Prodhom, W.E. Reiher III, B. Roux, M. Schlenkrich, J.C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiorkiewicz-Kuczera, D. Yin, and M. Karplus. All-atom empirical potential for molecular modeling and dynamics studies of proteins. *J. Phys. Chem.*, B 102:3586–3616, 1998.

[8] E. Neria, S. Fischer, and M. Karplus. Simulation of activation free energies in molecular systems. *J. Chem. Phys.*, 105:19021921, 1996.

[9] R. Rossi, M. Isorce, S. Morin, J. Flocard, K. Arumugam, S. Crouzy, M. Vivaudou, and S. Redon. Adaptive torsion-angle quasi-statics: a general simulation method with applications to protein structure analysis and design. *Bioinformatics*, 23(13): i408–17, 2007.