Combined static-dynamic deformations with haptic rendering

Yasaman Sedaghat

Master of Science

School of Computer Science

McGill University

Montréal, Québec

November 2011

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Science

ACKNOWLEDGEMENTS

I would like to gratefully thank my supervisors, Prof. Paul G. Kry and Prof. W. Robert J. Funnell for their inspiration, guidance and support. I would also like to thank the Computer Graphics lab members for their support throughout my work, especially paper discussions. In particular, I thank Olivier Rémillard and Sheldon Andrews for providing me with some components of the 3D simulation. Finally, I thank all my friends and family for their love and encouragement.

ABSTRACT

We present a real-time, physically based simulation method for animating high-resolution elastic deformations with a focus on haptic interaction. To achieve interactive rates without losing accuracy, the reduced material stiffness matrix is precomputed by removing the equations that correspond to the internal nodes of the system. In addition, we employ linear modal analysis to precompute the natural vibration modes of the system. We introduce a deformation-coupling technique in order to achieve the reduced dynamic behaviour while keeping the high-resolution local deformations. During real-time simulation, the high-spatial-frequency static deformations are coupled with the low-spatial-frequency dynamics, by projecting the reduced coordinate deformations onto an orthogonal basis constructed from natural vibration modes. To explore the implications of the coupling system, we describe different integration techniques to time-step the reduced dynamic solution in addition to evaluating the force feedback. Moreover, we show how we handle multiple contact points for non-sticky materials. To improve the contact-handling procedure, we employ our sliding technique to include friction. We compare our proposed method to the previously existing techniques in terms of run-time complexity and deformation properties using 3D meshes embedded in finite elements.

ABRÉGÉ

Nous présentons une méthode de simulation temps réel conforme aux lois de la physique pour animer des déformations élastiques à haute résolution, tout en portant une attention particulière aux interactions haptiques. Pour obtenir un résultat permettant une interaction temps réel sans perte de précision, la matrice réduite de rigidité du matériau est pré-calculée en excluant les équations correspondant aux noeuds internes du système. De plus, nous avons recours à l'analyse modale linéaire pour pré-calculer les modes de vibration naturelle du système. Nous proposons une technique de couplage des déformations afin d'obtenir le comportement à dynamique réduite recherché tout en préservant les propriétés des déformations locales à haute résolution. Lors de la simulation temps réel, les déformations statiques à haute fréquence spatiale sont couplées à la dynamique spatiale réduite à basse fréquence en projetant les déformations en coordonnées réduites sur une base orthogonale construite à partir des modes de vibration naturelle. Afin d'explorer l'impact du système de couplage, nous décrivons différentes techniques d'intégration pour avancer la solution de dynamique réduite dans le temps tout en évaluant le retour de force haptique. De plus, nous détaillons notre approche pour la gestion de points de contact multiples pour des matériaux non-adhésifs ainsi que notre méthode pour la gestion du glissement. Nous comparons la méthode que nous avançons aux techniques existantes en termes de complexité du temps d'exécution et en termes des propriétés de déformation, et ce en utilisant un maillage 3D intégré à un système à éléments finis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

Physically based simulation techniques have changed the traditional creation of animations significantly. These techniques, which are defined based on physics, robotics and mathematics, have greatly increased the realism of computer-generated animations. Over the past two decades, physically based animation methods have been widely used in simulating virtual characters, fluids and gases, fire and smoke, clothes and hair, and rigid and deformable bodies. The applications range from video games and movies to virtual surgery simulations.

Among all physics-based techniques, simulating deformations plays an important role in both off-line and on-line applications. In particular, elastic deformable bodies are used to simulate soft tissues in surgical training applications. These applications help surgeons to acquire the manual skills required for real surgeries. In addition to observing the deformations, the user can feel the contact forces with the use of haptic devices. As in many of the on-line applications, a key challenge in surgical haptic simulations is the trade off between interactivity and accuracy. The simulation must be fast enough to provide the user with a convincing sense of touch and resistance. Additionally, the deformable models need to resemble the behaviour of the real-word materials.

In this thesis, we propose a novel technique to simulate high-resolution physical deformable bodies in real-time with a focus on interactive haptic simulations. We build a

Figure 1–1: ***Our haptic simulation framework:*** *The 3D hand model is deformed in an interactive simulation using the 3-DOF Novint Falcon haptic device.*

framework consisting of a 3D volumetric deformable model, which is defined by a surface triangle mesh. In this framework the user can interactively deform the model and feel the resistance, using a haptic device. Figure 1–1 demonstrates an example of our framework, where the user is poking a deformable model with the 3 degree-of-freedom (DOF) Novint Falcon haptic device.

To make the haptic interactions realistic and convincing for the user, the simulation of the deformations along with the evaluation of the response forces should be performed at very high rates. In order to achieve these high rates some existing solutions simplify the problem by evolving the configuration of the model in a quasi-static state with precomputation of system responses [10, 17]. While not considering the dynamics of the system, which are

costly, these techniques have the benefit of not producing oscillations near the equilibrium of the object.

Similarly, we precompute the global stiffness matrix. During the off-line stage, we also employ the condensation technique [10] to remove the equations that correspond to the interior nodes from our system. The approach is to reduce the complexity of computing the deformations without giving up the volumetric behaviour of the deformable model.

In addition to the static methods, there exist more advanced techniques that further improve the deformations with decoupled linearized dynamics using modal vibration analyses [29]. The main advantage of using the natural vibration modes is to reduce the complexity of the problem to be solved by using only a small number of precomputed low-spatial-frequency mode shapes (the most dominant modes) to simulate the reduced dynamics of the system [18]. Solving for the complete deformations and forces considering all the modes while satisfying the displacement constraints is too expensive. However, by using these reduced methods we lose local deformations with high spatial frequencies such as sharp edges in the vicinity of the virtual finger when imposing displacements.

We have therefore developed the method of coupling the full-resolution static deformations with the low-spatial-frequency dynamics using orthogonal basis projections, which to the best of our knowledge has not been explored before. Alternative approaches either find completely static solutions in order to deform the object configuration, or solve the dynamics by considering the accelerations and the velocities of the object nodes.

Figure 1–2 illustrates the difference between (a) reduced low-frequency dynamic deformations, and (b) reduced dynamics coupled with full-resolution static deformations. In all

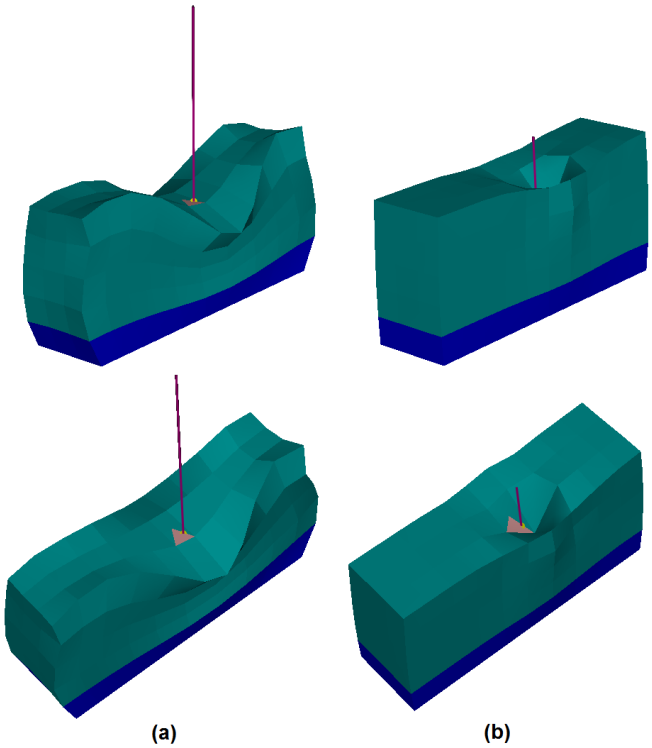

Figure 1–2: *Comparison of (a) reduced low-spatial-frequency dynamic deformations, which show the global deformation, and (b) reduced dynamics coupled with high-spatial-frequency statics, which show the local deformations in the vicinity of the virtual finger. The model is fixed at the bottom, and the pink line segment represents the response force. In both solvers, 30 modes out of 660 modes are picked for computing the low-spatial-frequency dynamics.*

of the cases, the 3D model is fixed at the bottom and undergoes an imposed displacement from the top.

We use different numerical integration methods to time-step the low-frequency dynamics. One of our proposed coupling techniques uses the symplectic Euler method for the purpose of momentum preservation. In the second approach, we solve a constraint-satisfaction

problem using Lagrange multipliers, with the backward Euler method being used to integrate the reduced dynamics. With this approach, we make it possible to include higher-frequency modes in the dynamic solution without losing stability of the simulations.

By using the linear mode shapes as our basis, we assume that the deformations are linear. Alternatively, we can apply our coupling approach to existing methods that consider non-linear large deformations, such as the work of Barbič and James [5]. They use a small number of precomputed non-linear modes to reduce the run-time complexity, but the resulting deformations do not include the high-spatial-frequency deformations. With our proposed method, we make it possible to superimpose the high-spatial-frequency statics on their model without increasing the time complexity.

1.1 Thesis overview

In Chapter 2 we discuss previous work that has been done in the field of deformation modelling, with the focus on interactive techniques. Chapter 3 describes the concept of our proposed method along with the main procedure for finding the deformations and evaluation of the response force as a result of the user interactions. In Chapter 4 we present the components of our contact-handling procedure, as well as the simulation process. Chapter 5 demonstrates the modifications required to integrate the proposed technique with a 3D finite-element model. We present our experimental results in Chapter 6. Lastly, the conclusions and future work are discussed in Chapter 7.

CHAPTER 2
Related work

The first part of this chapter, Section 2.1, describes the main physics-based methods that have been proposed for deformation modelling, followed by Section 2.2 where we discuss the challenges of haptic interaction with deformable models and the previous work that has been done in order to achieve real-time behaviour. Finally, Section 2.3 explores the existing contact-handling approaches to prevent the objects from penetrating one another.

2.1 Simulation of deformable objects

Physically based simulation of deformable bodies has been widely discussed in the computer graphics community for more than two decades. The very first approaches were proposed by Lasseter describing squash and stretch as one of the principles of animation [19] and by Terzopoulos et al. regarding elastically deformable models [30]. Since then, in order to achieve realistic animations, numerous different methods for simulating physical deformable objects have been proposed. Refer to [22] for a thorough survey of physically based deformable models. The applications range from cloth simulation, human and animal character modelling and facial expressions to medical image-analysis problems. In particular, realistic highly detailed deformable objects are used in surgical simulations and medical training applications to model soft tissues.

The simplest way to handle deformations is to use a mass-spring system in which the model is discretized into point masses that are connected via massless springs. In one

of the earliest such works, Waters used a static mass-spring model for facial animation [32]. Mass-spring systems are intuitive, easy to construct and computationally efficient. However these systems are not sufficiently accurate for many purposes. The material properties are only defined by spring stiffness constants and the discrete model is not a close approximation of the actual object. In addition, it is difficult to model incompressible volumetric objects and also thin shells that are resistant to bending. More recently, Wang et al. used a mass-spring model with a rigid core in order to preserve the volume and shape of the deformable model for use in virtual surgery simulations [31]. Conti et al. proposed the idea of filling the model with elastic volumetric spheres for volume conservation while keeping the interactivity rates [12].

More accurate physics-based models are defined based on continuum methods such as the finite-element method [34] or the boundary-element method [8, 13]. The FEM approaches are more natural but more expensive, since the only unknowns in the BEM method are the boundary displacements and forces. The BEM method gives us a small but dense set of equations to solve, whereas the FEM system results in a larger but sparse set of equations. For example, James and Pai solved a boundary-value problem involving linear elastostatic models using the precomputed boundary Green's function responses [17]. Bro-Nielsen and Cotin used linearized FEM with condensation for fast computation of static deformations in surgery simulation [10].

While solving a linear algebraic system is efficient and stable, it is only valid for small deformations. For large rotational deformations it will produce inaccurate forces that lead to inflation artifacts. To avoid these artifacts, Müller et al. proposed the co-rotational approach in which the rotational part of each finite element's deformation is extracted in

order to compute the forces based on the non-rotated frame of reference [21]. Barbič and James presented an efficient simulation of large deformations by evaluating cubic polynomials to determine the dynamics of the reduced coordinates [5]. Aiming for interactive simulation of highly detailed deformable bodies, Nesme et al. combined the traditional FEM with a novel technique for fast simulation of a coarse model which approximates the physical behaviour of a much finer model [23].

2.2 Haptic rendering of deformable objects

Recently the field of haptic rendering and interaction with virtual environments has grown enormously [20], specifically for interactions with deformable models. One of the main applications is surgical training simulations where the deformations and the response forces need to closely resemble the behaviour of real-world materials. These interactive applications need to update the configuration of the model at very fast rates to maintain the force-feedback refresh rate ($\sim$1 kHz). These high rates are required to maintain stability of the haptic controller and to provide the user with convincing tactile feedback. Therefore, general numerical methods such as the traditional finite-element method are not suitable for simulating the interactive models because they are computationally expensive, unless we use a technique to alter the computations or to simplify the model.

One possible way of simplifying the deformations is to consider only the static deformations [10, 17]. While this method ignores the effects of velocities and accelerations in the computation of the deformations, the benefit is that the complexity of computations is reduced significantly.

On the other hand, there exists the dimension-reduction method for simplifying the simulation of dynamic deformations. The reduction is performed by approximating the deformations with a smaller set of degrees of freedom in order to simulate a reasonable behaviour of the deformable model. Linear modal analysis [29] was first used in computer graphics simulations by Pentland and William [26, 15]. In the more recent work of James and Pai [18], the modal vibrations are used to achieve real-time deformation simulation. Additionally, Barbič and James [5] introduced non-linear bases constructed from modal derivatives and an interactive sketching technique, to reduce the dimensionality of the problem. By using these approaches, the complexity of the problem is significantly reduced through considering a small number of modes to represent the deformations. However, we lose the high-spatial-frequency deformations such as sharp edges in the vicinity of the virtual finger when imposing displacements. In contrast, our method combines low-spatial-frequency dynamics with full-resolution statics using orthogonal basis projections in order to preserve the realistic behaviour of the object, while not losing the interactivity of the system.

There have also been several approaches that employ precomputation to compute the fixed properties of the system that are needed during the run-time interaction. For fast computation of static deformations, the condensation method has been used during the precomputation stage [10]. James and Pai [17] precomputed boundary Green's function responses to be used later in the run-time simulation with incremental low-rank updates to guarantee interactivity. Barbič and James [5] precomputed the coefficients of the cubic polynomials for efficient performance of large-deformation simulations.

2.3 Contact handling

Another challenge in haptic rendering of deformable objects is to deal with contacts. Precise contact-handling, which is necessary in surgical simulations, consists of two stages: identifying all the potential collisions between the user manipulator and the deformable object, and finding the exact response of the model to satisfy the imposed displacement constraints. Bridson et al. proposed a robust continuous collision-detection solver with friction for cloth animation [9]. Barbič and James solved the multi-point contact problem by finding the penalty forces based on a nested pointshell and a distance field in order to interactively deal with rigid-deformable contacts [6]. They further improved their system by handling contact between deformable models [7].

One common strategy to handle contacts in the context of haptics is the linear complementarity problem (LCP) formulation. Pauly et al. used the LCP formulation to handle contacts between quasi-rigid objects [25]. Saupin et al. solved an extended form of the LCP formulation to include friction in their efficient contact-handling procedure for medical simulations [28].

Here we handle multi-point contacts using our two coupling approaches. In one of them we solve a Lagrange-multiplier constraint system to satisfy the imposed displacements, and in the other approach the contact response force is found by solving the high-resolution statics. This force then activates both the low-spatial-frequency dynamics and the high-spatial-frequency statics in order to satisfy the user-imposed constraints. Moreover, we enhance our single-point contact-handling model with an implementation of sliding friction.

CHAPTER 3
Problem statement

In most approaches to simulating a physically based deformable object, an external force is applied to the object to generate the deformations. In the case of using a haptic device, however, there is no information available about the applied force, so it is imposed displacements that drive the deformations. In our approach, based on the known displacements imposed by a virtual stylus, we compute the global configuration of the deformable body and also the reaction forces generated at the points of contact. After transmitting the force values to the haptic device, the mechanical actuators would apply appropriate forces to the user's hand.

Our framework consists of a 3D deformable model defined by a high-resolution mesh which is embedded in a coarse grid of finite elements. However, in this chapter we describe the problem in 2D assuming a mass-spring model which is simpler to understand. Chapter 5 addresses the overview of the 3D case along with the modifications that need to be made.

The displacements are assumed to be small enough to allow the assumption of linear elastic behaviour. It is assumed that the object is attached to the ground with enough pinned nodes to prevent the object from translating.

Our simulation process consists of an off-line stage during which the precomputations are performed only once (Section 5.3), and an on-line simulation stage in which the new configuration of the model and the feedback force are computed at each step given the

imposed displacements. The physics of the system is described by full-resolution static deformations, which are discussed in Section 3.1, and low-spatial-frequency dynamics, which is explained in Section 3.2. We discuss the details of the coupling of statics and dynamics in Section 3.3.

3.1 Static deformations

The static behaviour of the system can be described using the inverse Hooke's law (linear approximation):

$$u = Cf \tag{3.1}$$

where C is the compliance matrix and is symmetric, positive definite. For a system with n degrees of freedom, $C \in \Re^{n \times n}$. In this case $n = 2N$, considering N nodes in 2D. The forces that are acting on the degrees of freedom are defined by a column vector $f \in \Re^n$ and $u \in \Re^n$ is a column vector that denotes the displacements of the surface nodes due to the deformations.

The compliance matrix is computed by inverting the stiffness matrix in the preprocessing stage. In order to simulate the deformations in real-time, which is desirable in haptic interactions, the stiffness matrix is precomputed during the off-line stage taking into account the gradients of the forces acting on every nodee in the rest pose. This reduces the computational cost of evaluating the force response significantly as described later in Section 3.1.1 for the case of single-point contact, and in Section 3.1.2 for the case of multi-point contact. Note that the fixed nodes are removed from the stiffness matrix to make it invertible, therefore n is actually the number of non-fixed degrees of freedom.

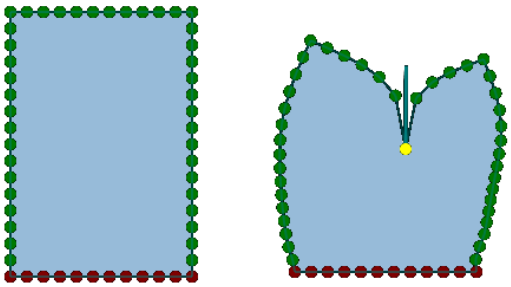

Figure 3–1: ***Imposed displacement on one particular node:*** *The red nodes are pinned and the blue line segment represents the force response.*

3.1.1 Single-point contact problem

In this section, we assume that the object is poked with a virtual stylus only at a single contact point. The case of multi-point contact-handling is addressed in Section 3.1.2.

Consider u_k to be some known displacement imposed on a DOF that would produce deformations. The corresponding f_k is the response force acting on this DOF due to the deformations and is fed back to the haptic device. By assuming zero forces everywhere else, Equation 3.1 can be efficiently solved by inverting a small block of the precomputed compliance matrix to find the unknown force. The displacements of the remaining nodes are computed by multiplying the complete compliance matrix by the force vector which is a zero vector except for the values that correspond to f_k. Figure 3–1 shows a deformable 2D object which undergoes an imposed displacement at one particular node. The red nodes are pinned and the blue line segment represents the force response.

This scenario holds when the contact is made directly on a node. In order to be able to make contacts on the edges between the nodes (springs in the mass-spring system), we

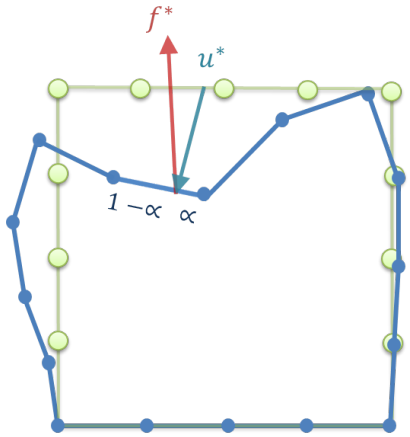

Figure 3–2: ***Contact at an interpolated point within an edge:*** *The red arrow labeled as f_* represents the response force, and the blue arrow labeled as u_* corresponds to the imposed displacement from the rest pose.*

define an interpolation matrix

$$H = \begin{pmatrix} 1-\alpha & 0 & \alpha & 0 \\ 0 & 1-\alpha & 0 & \alpha \end{pmatrix} \tag{3.2}$$

in which α is a number between 0 and 1 used to compute a weighted average of the displacements of the two nodes on either side of the edge. Figure 3–2 shows a deformable 2D object which undergoes an imposed displacement at a particular position within an edge. The red arrow represents the response force and the blue one corresponds to the imposed displacement from the rest pose.

Letting u_* be the imposed displacement of a position, f_* be the force feedback at the same position, u_{12} be the vector of the displacements of the two nodes, and f_{12} be the force, then we have

$$u_* = Hu_{12} \tag{3.3}$$

and

$$f_{12} = H^T f_*. \tag{3.4}$$

Using the inverse Hooke's law we know that

$$u_{12} = C_{12} f_{12} \tag{3.5}$$

where C_{12} is a 4×4 block of the compliance matrix which corresponds to the positions of these two nodes and to the response forces (assuming zero forces on the other nodes). We can rearrange Equation 3.5 in order to find f_* since we are not interested in the actual positions of the two nodes, as long as the interpolation produces the desired displacement. Therefore, by combining Equations 3.3, 3.4 and 3.5 we obtain

$$u_* = H C_{12} H^T f_*. \tag{3.6}$$

This small system of equations is solved using LU decomposition method [16], to find f_*. Then by substituting f_* in Equation 3.4 we can compute f_{12} and use that in the inverse Hooke's law to determine all of the displacements.

Since we have not considered topological changes such as cutting in our simulation, the object interacts with the environment only at its surface nodes, and the behaviour of the interior nodes is not visible. Therefore, during the off-line stage we apply the condensation method to reduce the size of the stiffness matrix by removing the equations that correspond to the interior nodes from the matrix equation as suggested by [10]. The details of performing this method are described in Section 5.3. Using this technique, the deformation computation time is reduced to the time required for finding the deformations of the surface nodes, while taking into account the effects of the interior of the object.

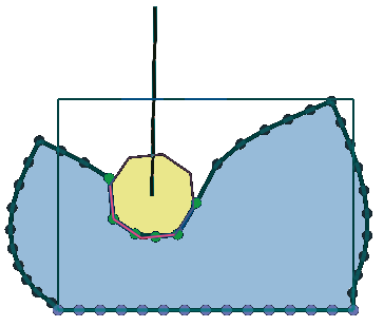

Figure 3–3: ***Types of multi-point contact:*** *The object is deformed from its rest configuration using a virtual stylus (the octagon). The pink edges of the rigid stylus and the green nodes of the object show two types of contacts.*

3.1.2 Multi-point contact problem

In Section 3.1.1 we described solving a single-point contact problem. The proposed model also allows multiple contact points. The virtual stylus whose movement is driven by a haptic device or a mouse is modeled as a discretized rigid object consisting of a set of nodes and edges, and the interaction begins by detecting collisions between either the nodes of the rigid stylus and the boundary of the deformable object, or the edges of the stylus and the surface nodes of the deformable object. These two types of collisions are shown in Figure 3–3.

In the case of a multi-point contact, the imposed displacement for each of the contact points produces an unknown force, and this force would change the configuration of the object. Therefore the multi-contact problem cannot be reduced to solving a number of independent single-point contacts.

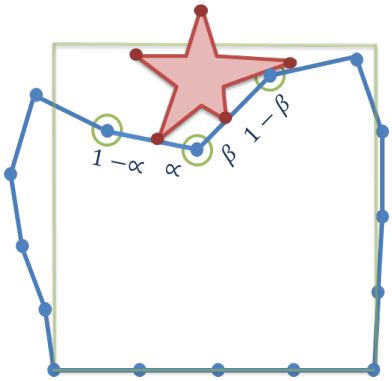

Figure 3–4: *The red polygon representing the rigid stylus is in contact with the deformable object at three points, for which the interpolation values are shown.*

Consider a simple example in which the number of contact points is 3, where two of them involve the interaction of stylus nodes and edges of the deformable object, and the last one involves the interaction of a stylus edge and a node of the object. This means we are enforcing displacements of 6 DOFs in the 2D system. See Figure 3–4 for an illustration of this example.

The interpolation matrix is then defined as follows:

$$H = \begin{pmatrix} 1-\alpha & 0 & \alpha & 0 & 0 & 0 \\ 0 & 1-\alpha & 0 & \alpha & 0 & 0 \\ 0 & 0 & \beta & 0 & 1-\beta & 0 \\ 0 & 0 & 0 & \beta & 0 & 1-\beta \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.7}$$

where each pair of rows corresponds to a single contact, and defines a weighted average of the two node positions on either side of an edge with the weights denoted by α and

β. Note that the last pair of rows corresponds to the contact in which we are imposing displacements directly on an object node. For a general case with M contacts, $H \in \Re^{m \times m}$ where $m = 2M$ (in the 2D case).

To find the unknown forces, the interpolation matrix is used to solve a modified version of Equation 3.6:

$$u_* = HC_m H^T f_* \tag{3.8}$$

where $u_* \in \Re^m$ is a vector of the imposed displacements, and C_m is an $m \times m$ block of the compliance matrix that relates these displacements to the forces represented by $f_* \in \Re^m$. Similar to what we described in Section 3.1.1 for a single-point contact, this system of equations is solved to find the response force. The new configuration of the system is then determined by multiplying the sparse force response by the complete compliance matrix.

Note that in some cases, a node or a DOF of the deformable object can be involved in more than one contact, as shown in Figure 3–5. Another example is when two or more contacts are made on a single edge of the object. The same situation happens if $\beta \to 0$ in our previous example shown in Figure 3–4. This will generate a set of equations in which there are more equations than unknowns. We use QR decomposition method with column pivoting [16] in order to solve this linear least squares (LLS) problem.

The force response that is transferred to the haptic device, and felt by the user as resistance, is defined by the sum of the computed forces of all of the contact points, which is shown with a black line segment in Figure 3–5.

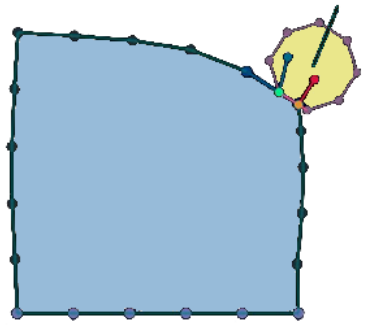

Figure 3–5: ***Shared point of contact:*** *A node (the orange one) is involved in two contacts introducing an LLS problem. The blue and red line segments represent the computed contact forces and the black line segment which starts from the centre of the virtual stylus shows the total force response.*

3.2 Dynamic deformations

The motion of the deformable body is derived from formulation of linear elastodynamics:

$$M\ddot{u} + D\dot{u} + Ku = f. \tag{3.9}$$

The $n \times n$ mass, damping and stiffness matrices are denoted by M, D and K respectively, and f is an externally applied force which acts on the system. The n nodal displacements are noted by u.

Our system of equations can be diagonalized with a change of coordinates [18]:

$$u = Uq \tag{3.10}$$

where $U \in \Re^{n \times n}$ contains the generalized eigenvectors of the stiffness matrix and the mass matrix computed using a generalized eigenvalue decomposition. The columns of the U matrix are the mode shapes of the system which represent the relative motions of the degrees of freedom (n in our case). The eigenvalues represent the natural frequencies

of the system and q is referred to as the modal coordinate vector. Refer to the work of Shabana [29] for additional details on linear modal analyses.

If we substitute Equation 3.10 into 3.9 and pre-multiply by the transpose of the eigenvector matrix, we obtain a simplified version:

$$M_q\ddot{q} + D_q\dot{q} + K_q q = \Gamma. \tag{3.11}$$

The external force Γ represents the force projected into the modal basis:

$$\Gamma = U^T f. \tag{3.12}$$

By performing this change of coordinates, both the stiffness matrix and the mass matrix will be diagonalized and will contain the mass and stiffness values for each of the modes separately. In order to make the damping matrix diagonal as well, we use Rayleigh damping as described in, for example, [11]:

$$D = \alpha M + \beta K \tag{3.13}$$

where α and β are the pre-defined damping ratios that are constant for all modes. Hence the damping matrix reduces to

$$D_q = \mathrm{diag}(\alpha m_i + \beta k_i). \tag{3.14}$$

Therefore the system is decoupled into n equations that can be solved independently. For each mode of the system, we have

$$m_i\ddot{q}_i + d_i\dot{q}_i + k_i q_i = \Gamma_i\ , \qquad i = 1 \cdots n. \tag{3.15}$$

The main advantage of decoupling the system as described is that a small number of low-spatial-frequency vibration modes can be used in order to simulate the reduced dynamics. Note that in our system, the external force f is actually a response to the user manipulation of the object. A possible method for solving this decoupled system is to use a time-domain IIR (infinite impulse response) digital filter convolution as suggested in [18]. This method is useful for under-damped systems, but we are more interested in over-damped systems to more closely resemble the behaviour of soft tissues. In Section 3.3 we explain the concept of combining a reduced dynamic solution with the full-resolution static solution. In order to explore the behaviour of our method for coupling statics and dynamics, we use different time-step integration schemes to find the dynamic solution. These different techniques are described in Section 3.3.1 and Section 3.3.2.

3.3 Coupling of statics and dynamics

In this section we present our two proposed methods for combining the static and dynamic solutions. In the first approach, described in Section 3.3.1, we find the force response based on the high-spatial-frequency statics and use that to time-step the reduced low-spatial-frequency dynamics using the symplectic Euler integration technique. In the second approach, which is explained in Section 3.3.2, we use the Lagrange-multiplier method to find the contact forces as well as the coupled statics and dynamics with a more stable integration technique.

As described in the previous section, with a coordinate transformation using the generalized eigenvectors of M and K, the vibration modes of the system are decoupled. The orthogonality properties of the eigenvectors provides the possibility of combining the static

and dynamic deformations of the system in order to have reduced dynamics along with the high-spatial-frequency statics. The trajectories of the two parts of the system response are independent of each other.

Any state u can be seen as a combination of the low-frequency modal coordinates q_1 and high-frequency coordinates q_2. If we let m be the number of low-frequency mode shapes that are used to define the dynamic deformations, then the remaining $(n - m)$ modes are used when computing the static deformations. Hence

$$\begin{aligned} u &= Uq \\ &= \begin{bmatrix} U_1 & U_2 \end{bmatrix} q \\ &= U_1 q_1 + U_2 q_2 \end{aligned} \tag{3.16}$$

where $U_1 \in \Re^{n \times m}$ and $U_2 \in \Re^{n \times (n-m)}$ are two orthogonal subspaces defined by the mode shapes. Note that in the case of modelling a deformable object that has a non-uniform mass distribution, the eigenvectors are not necessarily orthogonal to each other, so we use the null space [16] of U_1 to get U_2 in order to have an orthogonal basis to perform the coordinate transformation.

3.3.1 Dynamics driven by statics

In our proposed method, by changing the position of a DOF in the simulation, we allow the imposed displacements to directly specify only q_2 and let q_1 be determined by the low-frequency dynamics. Applying a displacement to a point of contact will produce an external force and this force will in turn activate both parts of the system.

Figure 3–6(a) illustrates our stepping method, where the blue bars indicate the state of the system at two consecutive time-steps. At time-step t, as part of the first step drawn in red in the figure, given q_1 computed at the previous step, let u_i^* be a new desired position imposed by the user interaction on a DOF denoted by i. Then we have

$$\Delta u_i^t = u_i^* - (U_1 q_1^t)_i \tag{3.17}$$

in which we compute the difference between the desired position and the position given by the dynamic state. In order to find the response force f_i that is acting on this DOF, we need to use a modified compliance matrix C_2 for which the U_1 basis has been projected out:

$$C_2 = C(U_2 U_2^T) \tag{3.18}$$

where the operator $(U_2 U_2^T)$ maps the compliance matrix into a space orthogonal to the space spanned by U_1 that is used to derive the dynamics deformations.

Using the modified compliance matrix, we solve a small system of equations given by the inverse Hooke's law to find the force as described in Section 3.1. Recall that f^t is a zero vector except for the contact points, which is used to compute the full static displacement vector

$$\Delta u^t = C_2 f^t. \tag{3.19}$$

The displacements vector Δu is entirely described in the U_2 basis.

By using Equation 3.16, the current pose of the system is coupled by summing the static part and the dynamic part. The computed response force f^t evolves the reduced low-frequency dynamics by using the symplectic Euler integration method [4]. This method

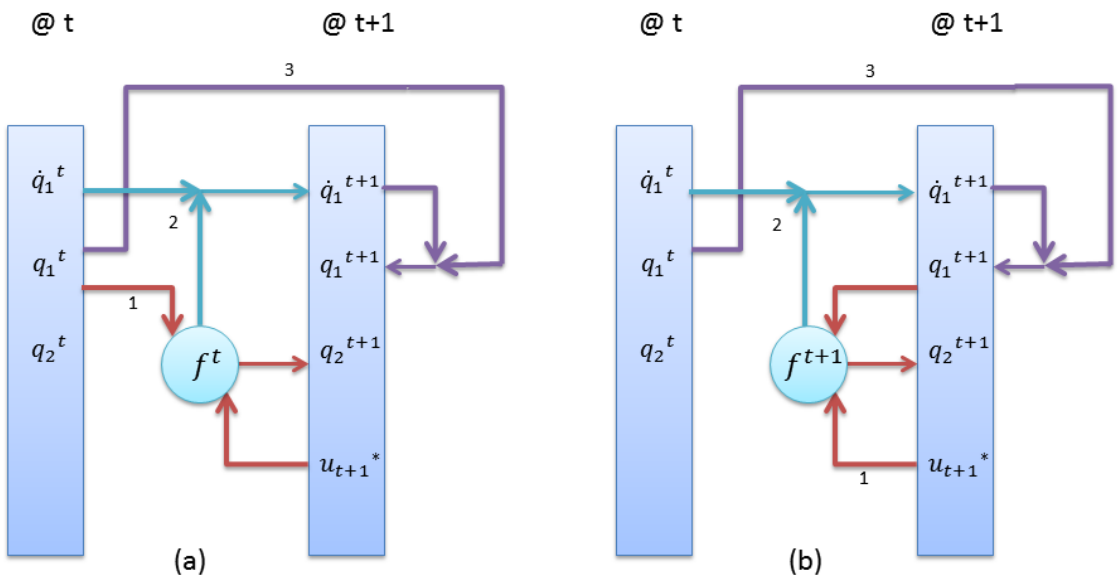

Figure 3–6: ***Our stepping methods for statics and dynamics coupling:*** *Diagram (a) indicates the symplectic Euler integration (Section 3.3.1) and diagram (b) indicates the backward Euler integration (Section 3.3.2). The blue bars indicate the state of the system at each time-step. Step 1 (red) illustrates the evaluation of the response force along with the high-spatial-frequency static deformations. Steps 2 (blue) and 3 (purple) show the integration of the low-spatial-frequency dynamics.*

has the advantage of preserving the energy of the system, but it will lead to instabilities for large time-steps. Since we are performing the integrations over the low-spatial-frequency modes, the required stable time-step is not very small. Applying this method, writing q_1^{t+1} for the modal displacement at the next time-step, q_1^t for the value at the current time and h for the time-step, we have

$$\begin{cases} \dot{q}_1^{t+1} = \dot{q}_1^t + h\ddot{q}_1^t \\ q_1^{t+1} = q_1^t + h\dot{q}_1^{t+1} \end{cases}. \tag{3.20}$$

If we substitute Equation 3.20 into 3.15, for each low-spatial-frequency mode we obtain

$$\dot{q}_i^{t+1} = \frac{\dot{q}_i^t + h(\frac{\Gamma_i^t}{m_i} - \frac{k_i}{m_i} q_i^t)}{1 + \frac{d_i}{m_i} h + \frac{k_i}{m_i} h^2} \tag{3.21}$$

where

$$\Gamma_i^t = U_1^T f^t. \tag{3.22}$$

Equation 3.21 is used to time-step the low-frequency modal displacements to obtain q_1^{t+1} based on f^t which is computed using the statics. This step is shown by the blue and the purple arrows in Figure 3–6(a). Therefore at the next step of the simulation, the new q_1^{t+1} is used in Equation 3.17 in order to continue to satisfy the imposed displacement constraints.

3.3.2 Statics driven by dynamics

Our second method for coupling the high-spatial-frequency static deformations with the low-spatial-frequency dynamics is described by using Lagrange multipliers to handle constraints [27]. Given the imposed displacements, instead of solving the system to find the static deformations first, we compute the dynamics in reduced basis via Lagrange multipliers. This approach has the benefit of being able to involve a larger number of mode shapes in the dynamic part of the deformations, whereas in the previous approach (Section 3.3.1), it is not possible to satisfy the displacement constraints without instabilities if not enough degrees of freedom are left in order to determine the static deformations. We will discuss when these instabilities occur in Section 6.3.

As in Section 3.1, let $u^* \in \Re^c$ be the user-imposed displacements where c is the number of degrees of freedom of the contact points ($c = 2$ in the case of a single contact with a 2D model) and let matrix H be the interpolation matrix for picking the associated DOFs

of the deformable object. Then by substituting Equation 3.16 in Equation 3.3, we obtain

$$HU_1q_1^{t+1} + HU_2q_2^{t+1} = u_{t+1}^*. \tag{3.23}$$

Note that the $H \in \Re^{cxn}$ matrix has the same structure as the interpolation matrix that was defined in Section 3.1, in which columns are filled with zeros in the place of not-in-contact DOFs.

In order to satisfy the constraints, a general energy function $g(u)$ is defined which is an implicit function over the positions and has to be equal to zero for all positions

$$\begin{aligned} g(u) &= Hu - u^* \\ &= 0. \end{aligned} \tag{3.24}$$

To ensure that energy is not added to or subtracted from the system (principle of virtual work), forces need to act in a perpendicular direction to the degrees of motion, so we take the derivative of the implicit function with respect to time. This gives us the valid force directions:

$$\begin{aligned} \dot{g}(u) &= \frac{\partial g}{\partial u}\frac{\partial u}{\partial t} \\ &= G(u)\dot{u}. \end{aligned} \tag{3.25}$$

Note that since the $G \in \Re^{c\times n}$ matrix is the derivative of the implicit function $g(u)$ with respect to u, it is equal to the interpolation matrix H. By defining the constraint force as $G^T\lambda$ to satisfy the position constraints, the static term $U_2q_2^{t+1}$ in Equation 3.23 can be replaced with $C_2(G^T\lambda)$ (inverse Hooke's law), where λ is the vector of the unknown

Lagrange multipliers:

$$HU_1 q_1^{t+1} + HC_2(G^T\lambda) = u_{t+1}^*. \tag{3.26}$$

Formulating the equation of motion while including the constraint force would give us

$$M_q \ddot{q}_1^{t+1} + D_q \dot{q}_1^{t+1} + K_q q_1^{t+1} + U_1^T G^T \lambda = 0. \tag{3.27}$$

Our goal at each step of the simulation is to find the deformations at the next time-step (both statics and dynamics) that satisfy the position constraints along with the constraint forces. Therefore, our system of equations consists of Equation 3.26 and Equation 3.27. As we mentioned in the first part of this section, in order to include more dynamic modes in the system without instabilities due to time-steps that are not small enough, we use the backward Euler method to integrate the dynamics of the system. Therefore, by choosing h as the time-step size, we have

$$\begin{cases} q_1^{t+1} = q_1^t + h\dot{q}_1^{t+1} \\ \dot{q}_1^{t+1} = \dot{q}_1^t + h\ddot{q}_1^{t+1} \end{cases}. \tag{3.28}$$

In order to find the low-frequency accelerations in the reduced basis at the next time-step, as well as the Lagrange multiplier vector, we substitute this integration scheme into Equation 3.26:

$$HU_1 q_1^t + hHU_1 \dot{q}_1^t + h^2 HU_1 \ddot{q}_1^{t+1} + HC_2(G^T\lambda) = u_{t+1}^*. \tag{3.29}$$

Combining Equation 3.27 and Equation 3.29 yields the following system of equations:

$$\begin{bmatrix} M_q + hD_q + h^2 K_q & U_1^T G^T \\ h^2 HU_1 & HC_2 G^T \end{bmatrix} \begin{bmatrix} \ddot{q}_1^{t+1} \\ \lambda \end{bmatrix} = \begin{bmatrix} -D_q \dot{q}_1^t - K_q q_1^t - hK_q \dot{q}_1^t \\ u_{t+1}^* - hHU_1 \dot{q}_1^t - HU_1 q_1^t \end{bmatrix}. \tag{3.30}$$

This system of equations can be solved efficiently by using the Schur complement method [16]. We can rewrite Equation 3.30 as a block matrix system:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \ddot{q}_1^{t+1} \\ \lambda \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}. \tag{3.31}$$

Recall that the upper left block A is diagonal and therefore can be inverted trivially by taking the inverses of the diagonal elements. Therefore we pre-multiply the first row with CA^{-1} and subtract the second row from the first row to get

$$(CA^{-1}B - D)\lambda = CA^{-1}a - b. \tag{3.32}$$

By solving Equation 3.32 the unknown Lagrange multipliers are found. Then we have

$$A\ddot{q}_1^{t+1} + B\lambda = a \tag{3.33}$$

which is solved to compute the low-frequency accelerations. Figure 3–6(b) demonstrates our stepping method. The low-frequency accelerations at the next time-step, combined with the positions and the velocities at the current time-step, give us the position updates in the dynamic reduced basis for all the DOFs (step 2 and step 3 in the diagram). On the other hand, using the vector of Lagrange multipliers along with the modified compliance (C_2) gives us the high-resolution static deformations. This step is part of the first step which is shown in red in Figure 3–6(b). By combining the dynamics with the static deformations (Equation 3.16), we find the positions at the next time-step that will satisfy the imposed displacement constraints. Note that the force feedback at each time-step is indeed the constraint forces ($G^T\lambda$) which would keep the DOFs of the deformable object at the desired configuration.

CHAPTER 4
Contact handling

In Chapter 3 we explained our methods for computing the unknown forces and the new configuration of the system based on the displacements that are imposed by the contacts. In this chapter we will present our contact-handling procedure which is performed by identifying all the potential colliding contact points, described in Section 4.1, and identifying when the contacts are broken and should let go of the object, described in Section 4.2.

4.1 Detecting contact

The collision-detection process is performed at each simulation step to find all the potential contacts that take place between the moving virtual stylus and the deformable object. Note that we have not considered self collisions in our method. For the sake of better understanding our algorithm, consider two phases: motion phase is when the virtual stylus is moving and can collide with the object; the update phase happens when the object is in the process of updating its configuration due to the computed force based on the imposed displacements from the previously detected contacts. The key point of our proposed algorithm is that if we update the configuration of the deformable object in the same simulation step in which the deformations are found, there is a chance of missing a potential collision, if the stylus happens to be in the middle of the update trajectory of the object. We explain how we detect collisions in these two phases in Algorithm 1, in which the overall collision-detection pipeline within the simulation thread is outlined. This algorithm summarizes the

entire procedure of the deformation simulation at each step, which is performed in parallel with the haptic thread. The haptic thread consists of fetching the new position of the haptic device p_h, and sending back the total force feedback f_t which is the sum of the response forces of all the contact points f_c^*.

Algorithm 1 Simulation thread

Input: p_h
Output: f_t

1: $v_s^{t+1} \leftarrow \frac{p_h - p_s^t}{h}$
2: Collision detection
3: **for** Each found contact **do**
4: Compute the weights to build the interpolation matrix H (Section 3.1)
5: Add the new contact to the contact list
6: **end for**
7: $p_s^{t+1} \leftarrow p_h$
8: Update u_c^* of the previously found contacts
9: $x \leftarrow x_0 + (U_1 q_1^t + U_2 q_2^t)$
10: Solve the statics and dynamics to obtain q_1^{t+1} and q_2^{t+1} (Section 3.3)
11: $f_t \leftarrow \sum_c f_c^*$
12: $v^{t+1} \leftarrow \frac{((U_1 q_1^{t+1} + U_2 q_2^{t+1}) + x_0) - x}{h}$
13: **if** Contact list is not empty **then**
14: Perform sliding (Section 4.3)
15: Check if the user is breaking any of the contacts (Section 4.2)
16: **end if**

As described in Algorithm 1, in Step 1 after fetching the new position of the haptic device denoted by p_h, instead of updating the position of the stylus, we set its velocity to the value obtained by dividing the difference of the haptic position and the stylus previous position by the time-step. As a result, the robust collision-detection procedure is able to find all the potential contacts that take place in the current motion phase. The collision detection is

performed within an iterative continuous collision-detection solver [9], with the modification of processing the earliest collision during each iteration; the iterations continue until no new collision is found. Based on the deformable model, the associated nodes for each of the contact points along with the interpolation weights are found. In Step 7 the position of the stylus is updated, and based on that we also update the imposed displacements of the contact points that were detected in the previous steps of the simulation thread.

In general, while the user is in contact with the deformable model, the system is solved in order to evaluate the unknown forces and the deformations, which is explained in Section 3.3. The solution gives us the new state of the system. Note that in Algorithm 1, the rest positions of the object nodes are represented by $x_0 \in \Re^n$. Similarly the current positions of the object nodes are represented by $x \in \Re^n$ and the current velocities are represented by $v^{t+1} \in \Re^n$. As mentioned at the beginning of this section, in order to identify the collisions that may take place in the update phase, we do not update the positions right after the new states of the object nodes (q_1^{t+1} and q_2^{t+1}) are found. Instead, the difference between the new positions and the previous positions is divided by the time-step and the result is assigned to the velocities of the object nodes (Step 12). By considering the velocities, we are able to detect the potential contacts in the next step of the simulation procedure, which will occur in the update phase due to the displacements of the object itself. After the contact identification step, we update the positions of the deformable object nodes with the computed states (q_1^t and q_2^t) in Step 9. Note that we could also take the $\dot{q}_1^{t+1}$ computed in the dynamic solver as the velocities, but it would not be accurate in terms of finding the potential contacts: it would not give us the new state of the deformable object because it would not take into account the static component of the displacement solution.

The last steps of our algorithm consist of checking if any of the contacts are broken and performing sliding friction, which are explained in Sections 4.2 and 4.3.

4.2 Breaking contact

The last step of the simulation thread is to identify whether any of the contacts are broken. The virtual stylus should let go of the object when all of the contacts are broken. As mentioned in Chapter 2, there exist several works using LCP formulations for modelling contacts [25, 14, 28]. In this section, we explain our method for dealing with contact-breaking which handles single-point contacts. This method works for multi-point contacts if the topology changes of the objects are smooth. In future, we plan to apply the complementarity methods in order to handle all cases of multi-point contacts.

For a non-sticky material, the user is only considered to be in contact with the object when the projection of the contact response force onto the surface outward-pointing normal is greater than zero, that is,

$$f \cdot \vec{n} > 0. \tag{4.1}$$

In other words, the user is only allowed to push the object. Figure 4–1 represents a 2D model which is deformed due to a single contact. Note that n defines the surface outward-pointing normal, and f_n shows the response force in the normal direction.

4.3 Sliding

Consider a situation where the virtual stylus is pushing the object while the direction of the imposed displacements is along the surface. This situation is demonstrated in Figure 4–2. Although the user is pushing the object, the response force direction is pointing

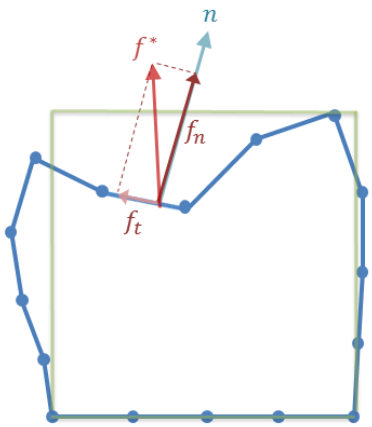

Figure 4–1: *A 2D model which is deformed due to a single contact. Note that n shows the surface outward-pointing normal at the point of contact, f_n shows the response force in the normal direction and f_t shows the tangential component of the force.*

inward leading the contact to break. To avoid these invalid contact-breakings, we have implemented sliding.

So far the deformable object has been assumed to have infinite friction due to the bilateral constraint setup on the contacts. In our sliding method we are able to employ both viscous friction and Coulomb friction.

For applying viscous friction, by changing the position of the actual point of contact on the surface, we decrease the ratio of the magnitude of the tangential force to the magnitude of the normal force. The cancelling of the tangential force is performed at each simulation step and the speed of convergence is chosen by the sliding coefficient ρ_s. Hence, the sliding equation is defined as follows:

$$\delta p = \rho_s |f_t| \tag{4.2}$$

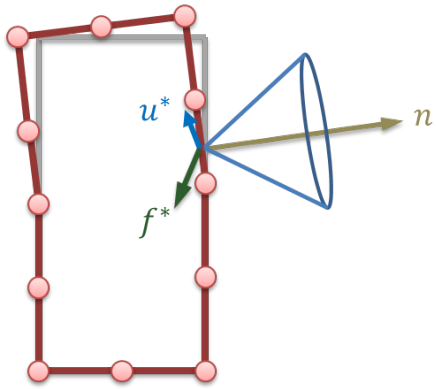

Figure 4–2: *A 2D model is deformed due to a single contact. Note that n defines the surface outward-pointing normal at the point of contact. Sliding iteratively modifies the position of this point to decrease the ratio of the magnitude of the tangential force to the magnitude of the normal force, so that f^* falls inside the friction cone.*

where δp is the displacement of the contact point on the surface and f_t is the tangential force. The direction of sliding is determined based on the relative velocity of the virtual stylus.

The approach to taking Coulomb friction into account is to approximate a friction cone, so that the user feels the static friction only when the response force lies inside this cone. Otherwise sliding will take place.

The friction cone is defined to limit the ratio of the magnitude of the tangential force to the magnitude of the normal force. Its vertex is at the point of contact and its axis is along the surface outward-pointing normal as shown in Figure 4–2. When the ratio of the tangential force to the normal force exceeds the tangent of the cone angle, sliding will occur. We use Equation 4.2 in an iterative fashion to find a new position for the contact point, which brings the force inside the friction cone.

Let μ_c be the tangent of the friction cone angle, and f_n be the normal force along the surface. Then as long as

$$|f_t| \leq \mu_c |f_n| \tag{4.3}$$

sliding will not occur.

CHAPTER 5
3D finite-element model

Our 3D volumetric deformable model is discretized into cubes that approximate the behaviour of the actual model. Each element is defined by eight nodes (locally labelled 0,1,..,7) leading to 24 degrees of freedom as shown in Figure 5–1. In order to better simulate highly detailed geometries, the complex 3D triangular mesh is embedded in a coarse grid of finite elements. Figure 5–2 illustrates a triangular mesh of a sphere which is embedded in a coarse grid of cube elements.

The procedure that we described for our solver in Chapter 3 does not depend on the structure of the deformable model. Clearly, the concept of finding the static and dynamic deformations, as well as the evaluation of the force response, can be combined with any physical model. In this chapter we describe those parts of the whole procedure that are modified in order to achieve the desired behaviour when using a 3D finite-element model. In particular, in Section 5.1 we discuss the derivation of the shape functions used to interpolate the displacements. Section 5.2 describes the collision-detection process within a finite-element-model framework, followed by Section 5.3 in which we describe the preprocessing step in detail.

5.1 Displacements and shape functions

In the classical finite-element method, the nodes of the elements are the only degrees of freedom that we can control. The displacement of a point inside an element is parametrized

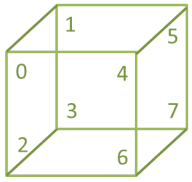

Figure 5–1: *Each cube element is defined by eight nodes which are locally labelled 0,1,..,7.*

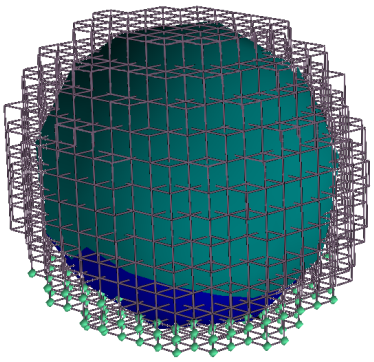

Figure 5–2: *Triangular mesh of a sphere embedded in a coarse grid of finite elements.*

by the shape functions. Shape functions are used to map point displacements to the displacements of the element nodes using a set of weights. Since it is assumed that the elements are homogeneous, we use tri-linear functions to interpolate the displacements. Therefore, for the weights of an arbitrary point p inside a cubic element we have

$$w_0 + w_1 + ... + w_7 = 1. \tag{5.1}$$

The value of w_n is one at node n and zeros at all of the remaining nodes of the element, and a tri-linear function in between. Let $u_p \in \Re^3$ be the displacement of the point, and $u_e \in \Re^{24}$ be the displacements of the corresponding element nodes. Then we have

$$u_p = W u_e = \begin{bmatrix} W_0 & W_1 & ... & W_7 \end{bmatrix} u_e \tag{5.2}$$

where the interpolation matrix consists of the shape functions W_n, each defined as follows:

$$W_n = \begin{bmatrix} w_n & 0 & 0 \\ 0 & w_n & 0 \\ 0 & 0 & w_n \end{bmatrix}. \tag{5.3}$$

5.2 Collisions with embedded objects

When embedding meshes, the collision detection is applied to the embedded objects rather than to the coarse grid of elements. While the user makes contact with a triangular patch in the surface mesh, the imposed displacements are mapped to the nodes of the elements associated with that particular patch. Once this mapping is found, it can be used in the form of an interpolation matrix in the deformation solver (described in Chapter 3) to compute the actual node displacements. Mapping these nodal displacements back to the triangular mesh vertices at each step of the simulation gives us the new configuration of the highly detailed mesh as a response to the user interactions. Refer to Figure 5–3 for a 2D illustration of a mesh triangle embedded in three finite elements where each vertex lies in a different element. The nodes of the elements corresponding to the contact point p are shown in blue.

The displacement of each vertex of a triangular patch is parametrized by using Equation 5.2 where each vertex displacement u_{T_i} is linearly mapped to the associated element's nodes. Similarly, for any point p within the triangle, we find the barycentric coordinates that will map the displacement of the point to the displacements of the triangle vertices.

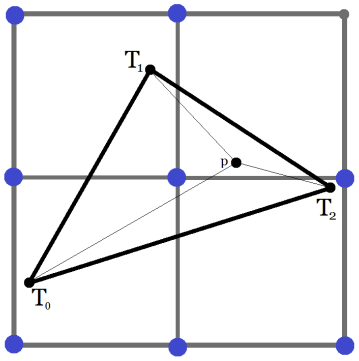

Figure 5–3: ***Collision with an embedded object:*** *A mesh triangle is embedded in a 2D element grid, where each vertex lies in a different element. Note that although the contact point* p *lies in the upper right element, the corresponding nodes in the coarser grid are the other three elements' nodes (shown in blue).*

Hence we have

$$u_p = Bu_T \tag{5.4}$$

where $B \in \Re^{3\times 9}$ contains the barycentric weights and $u_T \in \Re^9$ is a vector of the three vertex displacements. For each vertex we have

$$u_{T_i} = \mathsf{W}_i u_{ei} \ , \qquad i = 0, 1, 2 \tag{5.5}$$

where W_i denotes the interpolation matrix that maps the displacement of vertex i to the displacement of its corresponding element nodes $u_{ei} \in \Re^{24}$ as described earlier in Section 5.1. Thus the combined mapping is defined as follows:

$$u_p = B \begin{bmatrix} \mathsf{W}_0 & 0 & 0 \\ 0 & \mathsf{W}_1 & 0 \\ 0 & 0 & \mathsf{W}_2 \end{bmatrix} \begin{bmatrix} u_{e1} \\ u_{e2} \\ u_{e3} \end{bmatrix} = H u_{et}. \tag{5.6}$$

In summary, this mapping gives us the interpolation matrix H that maps the displacement of the contact point on the surface mesh to the displacement of the controllable DOFs of the corresponding finite elements u_{et}. The remaining steps of the collision handling procedure are similar to what we described in Chapter 4.

5.3 Off-line preprocessing step

This section describes the precomputation step which is performed one time during the off-line stage, before starting to interact with the deformable model. The precomputation step includes computing the global stiffness matrix, applying condensation to extract the behaviour of the surface nodes, generalized eigenvalue decomposition in order to find the natural vibration modes, and modifying the compliance matrix defined in the static basis.

In the classical form of the finite-element formulation, the stiffness of each element is described by the Hessian of the potential energy of the system. The global stiffness matrix is constructed by combining the element stiffness matrices.

The global stiffness matrix is a large sparse matrix which defines the linear behaviour of all the DOFs of the deformable model. Since in our framework changes in topology such as cutting are not allowed, the haptic interaction only happens with the surface of the object. Therefore it is sufficient to simulate the deformations of the surface visible nodes and not the rest. Hence the computation time for calculating the static and dynamic deformations is greatly reduced, becoming the time required to find the deformations only of the surface nodes. In order to preserve the volumetric behaviour of the object, we use the condensation method [10] in our static solution. Let us organize our finite elements so that the surface elements appear first in the list, followed by the internal ones. The size of

the global stiffness matrix is then reduced by taking its Schur complement [16] to remove the internal degrees of freedom from the matrix equations (Guyan reduction). Because our model is assumed to be attached to the ground with enough fixed nodes to prevent it from translating, the fixed nodes are also removed from the original stiffness matrix.

The next step of the precomputation stage is to find the basis to perform the coupling. Our basis is the result of the generalized eigenvalue decomposition taking the sparse stiffness matrix and mass matrix into account. We assume lumped masses at the nodes of the deformable object, which gives us a diagonal mass matrix. Computing the linear basis can be done either by finding the eigenvectors of $M^{-1}K$ or by using the generalized eigenvalue decomposition of M and K. The results are similar except for the scaling of the eigenvectors and eigenvalues.

Since we are only interested in the deformations of the surface nodes, the degrees of freedom that correspond to the interior nodes are removed from the shape modes. Note that the reduced modes still capture the behaviour of the interior nodes due to the fact that the eigenvalue decomposition is performed on the complete stiffness and mass matrix.

Now that the reduced mode shapes have been found, they are decoupled into U_1 and U_2 bases, based on the number of requested low-spatial-frequency modes to be used for computing the dynamic deformations. If the model has a non-uniform mass distribution, the eigenvectors are not necessarily orthogonal to each other. In general, therefore, we use the null space of U_1 to get U_2 in order to have an orthogonal basis to perform the coordinate transformation. The U_1 basis will be used to simplify the equation of motion by diagonalizing the mass, stiffness and damping matrices as described in Section 3.2.

To get the compliance matrix, the dense surface stiffness is inverted once during the pre-processing step. We then map the compliance matrix into the space spanned by the U_2 basis to describe the static deformations. Using Equation 3.18, we get the modified compliance matrix that is used in the static computation phase of our simulation.

CHAPTER 6
Experimental results and discussion

Our simulation framework and the proposed solvers are implemented in Java. We use the CHAI 3D SDK [3], which supports a variety of different haptic devices, to fetch the position of the virtual stylus and to deliver the response force during the simulation. The system has been tested with the 3-DOF Novint Falcon haptic device. For most of the matrix operations, we used the Matrix Toolkit Java (MTJ) package [2] but for the general eigenvalue decomposition we used LAPACK [1].

In this chapter we present the results obtained using the methods that are discussed in Sections 3.3.1 and 3.3.2. In Section 6.1, the characteristics of the deformable models used in our simulation are presented. Section 6.2 describes the results of the off-line stage. Lastly, in order to compare the time complexity of solving such systems, in Section 6.3 we compare our run-time complexity to some similar approaches that have been proposed to find the deformations along with the response force in an interactive simulation.

6.1 Our 3D models

Two 3D finite-element models are used in order to obtain the results. Refer to Table 6–1 for the characteristics of the meshes used, the numbers of elements and the numbers of degrees of freedom. Figure 6–1 illustrates our models.

Model	Mesh vertices	Number of elements	Number of DOFs
Cube	1225	343	1536
Hand	2146	313	1707

Table 6–1: ***The characteristics of the models and the meshes that are used in our implementation:*** *The finite elements are cubes and the embedded meshes are triangular. The last column presents the number of DOFs of the full system (before extracting the surface nodes and fixed nodes).*

Model	m	Reduced DOFs	Precomputation time [s]
Cube	10	1152	122
Hand	15	1368	200

Table 6–2: ***The precomputation time complexities:*** *Column* m *shows the number of low-frequency modes in the dynamic basis. "Reduced DOFs" corresponds to the surface nodes only.*

6.2 Off-line stage

The off-line stage, as described in Section 5.3, involves computing the stiffness matrix; removing the rows and columns that correspond to the fixed nodes (to prevent the system from translating and also to make the stiffness matrix invertible); removing the internal nodes by performing the Guyan reduction method; and inverting the matrix to get the compliance matrix. Then the eigenvalue decomposition is performed to compute the natural vibration modes and also to modify the compliance matrix with the null space of the U_1 basis to be used in the static solver. Table 6–2 illustrates the time needed to execute the off-line stage for each model. It also presents the number of reduced DOFs after removing the internal nodes and the fixed nodes from the system. Note that this number shows the size of the condensed stiffness matrix and the number of the natural vibration modes. The column labeled m shows the number of low-spatial-frequency modes in the dynamic basis.

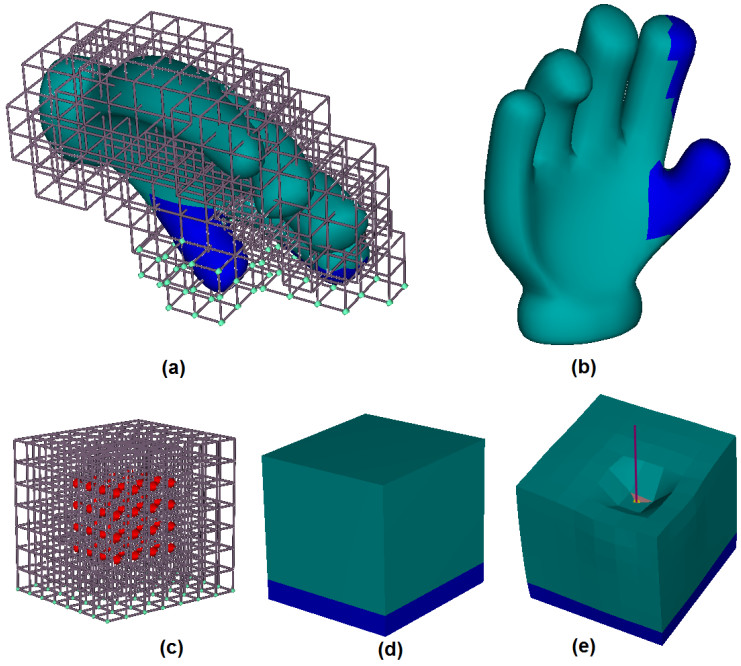

Figure 6–1: ***Our 3D Models:*** *The top row shows the hand model in its rest shape; in (a) it is surrounded by a grid of finite elements. The bottom row shows (c) the elements surrounding the cube model, where the red dots are the internal nodes removed in the condensation, (d) the undeformed cube in its rest shape and (e) the deformed cube undergoing a single contact. In both models the blue part of the mesh and the set of green dots within the elements represent the fixed nodes.*

6.3 On-line simulation

We performed an experiment in order to highlight the run-time performance of the on-line simulation using different approaches. During this experiment, we compared a full static solver, a full linearized dynamic solver considering all of the vibration modes, a reduced dynamic solver which only involves computing a small number of low-frequency vibration modes, and finally our coupled solver using the two described approaches (Section 3.3.1 and Section 3.3.2). The comparison of our proposed methods is discussed in Section 6.4.

Model	m	full statics [ms]	full linearized dynamics [ms]	reduced dynamics [ms]	first method [ms]	second method [ms]
Cube	10	1	147	0.7	1.1	1.2
Hand	15	1.2	228	1.1	1.4	1.9

Table 6–3: ***The run-time computation performance using different solvers per simulation step:*** *Column* m *shows the number of low-frequency modes in dynamic basis. Our first method corresponds to the reduced dynamics driven by full statics and our second method corresponds to the reduced dynamics with full statics, solved with Lagrange multipliers.*

Table 6–3 illustrates the run-time per simulation step ($h = 0.08$ s) for each of the solvers using our 3D models. The reported values are achieved by averaging over 1000 computation steps. During each step, the system is solved in order to find the deformations and the response force based on the user-imposed displacements.

As expected, the run-time of the full linear static solver is insignificant. As described in Section 3.1, evaluation of the force in the static solver only involves inverting a small block of the compliance matrix. In addition to that, we need to multiply the computed force with the full compliance matrix in order to achieve the new configuration of the system (Equation 3.1). Since only a small number of elements in the response force vector are non-zero, the computation time of this sparse multiplication can be reduced by using Selective Matrix Vector Multiplication (SMVM), as suggested in [10].

As shown in Table 6–3, augmenting the linear static deformations with the linearized low-frequency dynamics using our first method (Section 3.3.1) does not affect the performance significantly. It only involves computing a small number of mode displacements using Equation 3.21, by substituting the computed force response projected onto the U_1 basis.

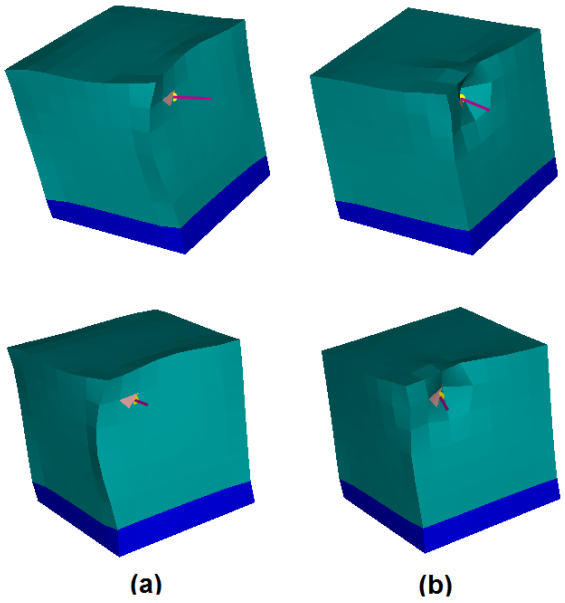

Figure 6–2: ***Reduced dynamic deformations vs reduced dynamics with statics:*** *The cube model is deformed considering only reduced dynamic deformations (a) whereas the model in (b) also involves the high-frequency static deformations; hence the local deformations in the vicinity of the virtual finger. The pink line segment shows the response force. Note that in both solvers 30 modes out of 1152 modes are picked for computing the low-spatial-frequency dynamics.*

Consider a case where we only solve for the linearized dynamics, using m low-spatial-frequency vibration modes. Figure 6–2(a) demonstrates our cube model from two different viewpoints, deformed by employing 30 low-frequency dynamic modes. Compared with (b), the lack of local deformations of high-spatial-frequency modes is evident.

The more modes we add to the dynamic basis, the more complete the deformations. Eventually, in order to capture all the high-frequency deformations, we have to add all the linearized vibration modes to the system of equations. Looking at Table 6–3, it is noticeable that the run-time complexity of the full linearized dynamic solver is much higher

than a reduced one, which makes this method unsuitable for interactive applications. This is because finding the constraint forces along with the deformations using the Lagrange-multiplier approach discussed in Section 3.3.2 (Equation 3.30) results in a large dense vector of unknown accelerations $\ddot{q}_1^{t+1}$, since it involves a complete basis using all of the vibration modes. In our proposed method, on the other hand, we only use a small number of low-spatial-frequency modes for the dynamic solution. In order to present the high-spatial-frequency deformations, we couple the system with full-resolution static deformations. Using either of the proposed approaches (Section 3.3.1 and Section 3.3.2), the unknown accelerations consist of a very small number of variables which can be computed fast. Figure 6–2(b) illustrates the coupled deformations for the cube model.

Figure 6–3 demonstrates our simulation snapshots of the hand model which undergoes several imposed displacements. The coupled deformations are computed using 15 low-spatial-frequency modes for the dynamics out of 1368 modes. The remaining high-spatial-frequency modes are selected for the static deformations.

6.4 Comparison of our proposed methods

Our motivation for solving the system using Lagrange multipliers was the instabilities that we came across using the first approach, where the response force computed from solving the linear statics drives the low-frequency dynamics. Often, the stable time-step that is required to integrate the higher-frequency modes is much smaller than the time-step required for integrating lower-frequency modes. Therefore, as we select a larger number of modes for the dynamic solution, we need to decrease the time-step in order to have a stable system within a symplectic Euler integration scheme. On the other hand, by

not leaving enough degrees of freedom for the static solver, the evaluated force response and the computed deformations are not able to satisfy the constraints. Therefore, in our second approach using the backward Euler scheme in Equation 3.30, the system is solved implicitly to find the static and dynamic deformations.

With a reasonable number of dynamic modes, the difference between the results of the two approaches are insignificant, in terms of the trajectory of the nodes and the response forces. In the cube model, when h is less than 0.5 s and m is less than 30 the difference is not noticeable. But as we increase the number of dynamic modes, these differences grow to the point that when using the first approach, the system either blows up due to large time-steps or simply cannot satisfy the displacement constraints even with very small time-steps, as a result of not having enough DOFs to solve the problem. As an example, if we only select two high-frequency modes for the static deformations, the system is not able to produce satisfactory deformations.

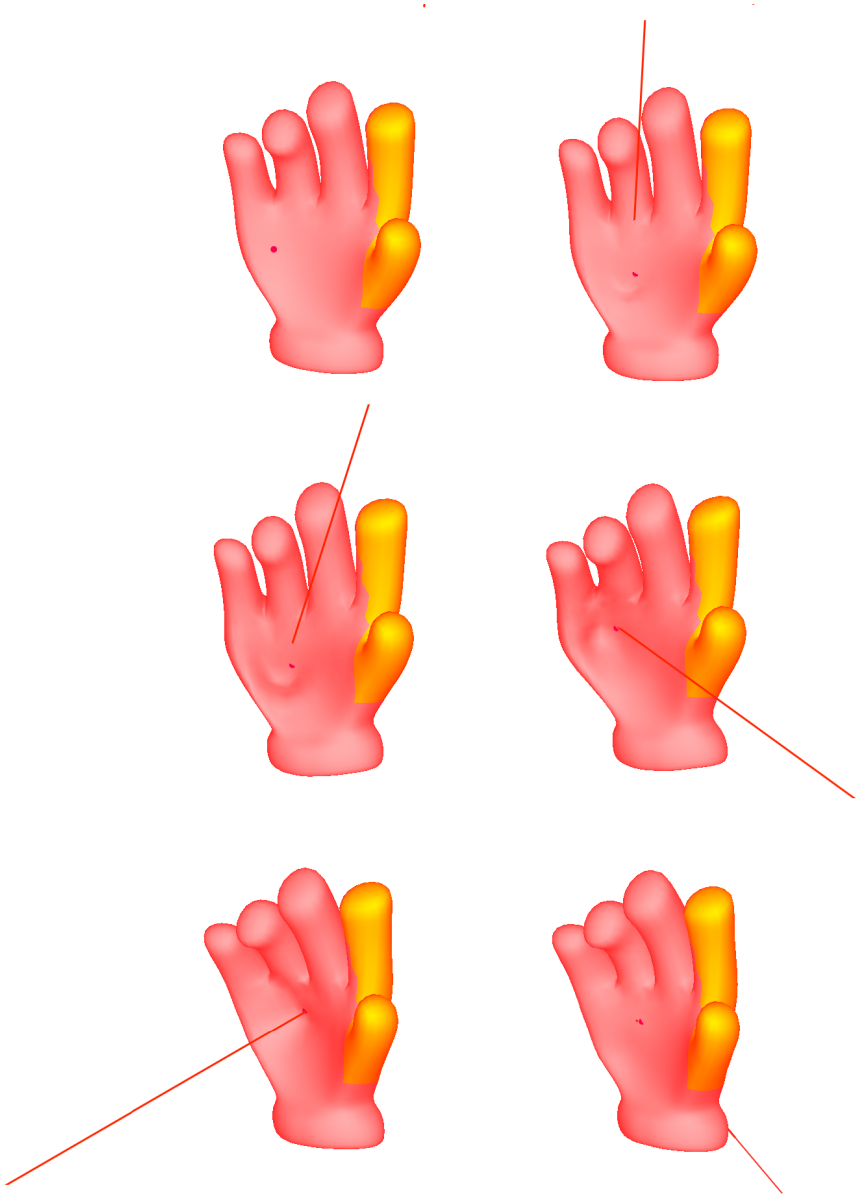

Figure 6–3: ***Simulation snapshots of the coupled deformations:*** *The hand model is deformed by the user interaction. The user comes into contact with the model, imposes some deformations, and lets go of the model (last snapshot). The orange region of the mesh contains the fixed nodes. The red dot shows the position of the virtual stylus and the line segment represents the response force. In the coupling solution 15 low-spatial-frequency modes out of 1368 modes are selected for the dynamics and the remaining modes are left for constructing the high-resolution-static basis.*

CHAPTER 7
Conclusion and future work

In this work, we proposed different methods for coupling of low-spatial-frequency dynamics and high-spatial-frequency statics in order to have an interactive deformable model simulation. Our simulation technique is appropriate for real-time haptic-enabled applications, such as surgical simulations. In the future, we will test our simulation with the high fidelity 7-DOF MPB Freedom 7S device. Using our framework, the surgeon can acquire manual skills from haptic interaction with 3D volumetric models which represent the real tissues. In comparison with previous approaches, the advantage of our coupling method is that, without losing the interactivity, we add the high-spatial-frequency local deformations to the system so the deformations better resemble a principled model of elastic deformation, so that the simulation corresponds to a real-world material.

In the two proposed coupling methods, we use first-order numerical integration techniques to find the discretized trajectories using time-steps. An alternate choice would be a higher-order integration technique to achieve better accuracy. The second-order mid-point method has $O(h^3)$ error instead of $O(h^2)$ and the order-4 Runge-Kutta method has $O(h^4)$ error. However, these methods suffer from instabilities when used with large time-steps. Refer to [33, 4] for a comprehensive description of the accuracy and stability of the different time-stepping methods for solving differential equations.

Another option is the family of Newmark single-step methods [24] for solving dynamic problems. When integrating the system, it uses a portion of the accelerations at the current time-step in addition to the accelerations at the next time-step. With the right choice of parameters, this method is able to select between accuracy and stability depending on the application.

Our coupling technique provides the possibility of using an implicit stable method to integrate the higher-frequency modes, while using a fast explicit method in order to compute the lower-frequency modes. In the future, we would like to employ this strategy to combine our two presented coupling methods to achieve better accuracy while not losing interactivity.

In order to remove the linear-deformation assumption, we plan to allow large non-linear deformations by employing the co-rotational approach [21], in which the rotational part of each finite element's deformation is extracted in order to compute the forces based on the non-rotated frame of reference. Moreover, instead of using linear dynamic modes, we plan to use a non-linear basis such as the combination of the natural modes and the modal derivatives as suggested in [5]. The key idea would continue to be the use of physics-based modes that vary smoothly and slowly across the surface of the deformable object, and let the high-spatial-frequency local deformations be handled with the statics.

References

[1] *LAPACK–Linear Algebra Package*. `http://www.netlib.org/lapack/`.

[2] *MTJ–Matrix Toolkit Java Library*. `http://code.google.com/p/matrix-toolkits-java`.

[3] *CHAI 3D Documentation*, 2009. `http://www.chai3d.org/doc/index.html`.

[4] Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1st edition, 1998.

[5] Jernej Barbič and Doug L. James. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Transactions on Graphics (SIGGRAPH 2005)*, 24(3):982–990, August 2005.

[6] Jernej Barbič and Doug L. James. Time-critical distributed contact for 6-dof haptic rendering of adaptively sampled reduced deformable models. In *2007 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, August 2007.

[7] Jernej Barbič and Doug L. James. Six-dof haptic rendering of contact between geometrically complex reduced deformable models. *IEEE Transactions on Haptics*, 1(1):39–52, 2008.

[8] C. A. Brebbia, J. C. F. Telles, and L. C. Wrobel. *Boundary element techniques : theory and applications in engineering*. Springer-Verlag, New York, 1984.

[9] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics*, 21:594–603, 2002.

[10] Morten Bro-Nielsen and Stephane Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Computer Graphics Forum*, pages 57–66, 1996.

[11] T. K. Caughey. Classical normal modes in damped linear dynamic systems. *Journal of Applied Mechanics*, 27(2):269–271, 1960.

[12] F. Conti, O. Khatib, and C. Baur. Interactive rendering of deformable objects based on a filling sphere modeling approach. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 3, pages 3716 – 3721 vol.3, sept. 2003.

[13] S. L. Crouch and A. M. Starfield. *Boundary Element Methods in Solid Mechanics*. Unwin Hyman Inc., London, 1990.

[14] C. Duriez, F. Dubois, A. Kheddar, and C. Andriot. Realistic haptic rendering of interacting deformable objects in virtual environments. *Visualization and Computer Graphics, IEEE Transactions on*, 12(1):36 –47, jan.-feb. 2006.

[15] Martin Friedmann and Alex Pentland. Distributed physical simulation. In *Third Eurographics Workshop on Animation and Simulation*, pages 1–17, 1992.

[16] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)(3rd Edition)*. The Johns Hopkins University Press, 3rd edition, October 1996.

[17] Doug L. James and Dinesh K. Pai. ArtDefo: accurate real time deformable objects. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 65–72, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[18] Doug L. James and Dinesh K. Pai. DyRT: dynamic response textures for real time deformation simulation with graphics hardware. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 582–585, New York, NY, USA, 2002. ACM.

[19] John Lasseter. Principles of traditional animation applied to 3d computer animation. *SIGGRAPH Comput. Graph.*, 21:35–44, August 1987.

[20] S.D. Laycock and A.M. Day. A survey of haptic rendering techniques. *Computer Graphics Forum*, 26(1):50–65, 2007.

[21] Matthias Müller and Markus Gross. Interactive virtual materials. In *Proceedings of Graphics Interface 2004*, GI '04, pages 239–246, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

[22] Andrew Nealen, Matthias Mller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.

[23] Matthieu Nesme, Paul G. Kry, Lenka Jeřábková, and François Faure. Preserving topology and elasticity for embedded deformable models. *ACM Trans. Graph.*, 28:52:1–52:9, July 2009.

[24] N. M. Newmark. A method of computation for structural dynamics. *Journal of the Engineering Mechanics Division*, 85(7):67–94, 1959.

[25] Mark Pauly, Dinesh K. Pai, and Leonidas J. Guibas. Quasi-rigid objects in contact. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, pages 109–119, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[26] A. Pentland and J. Williams. Good vibrations: modal dynamics for graphics and animation. *SIGGRAPH Comput. Graph.*, 23:207–214, July 1989.

[27] John C. Platt and Alan H. Barr. Constraints methods for flexible models. *SIGGRAPH Comput. Graph.*, 22:279–288, June 1988.

[28] Guillaume Saupin, Christian Duriez, and Stephane Cotin. Contact model for haptic medical simulations. In *Proceedings of the 4th international symposium on Biomedical Simulation*, ISBMS '08, pages 157–165, Berlin, Heidelberg, 2008. Springer-Verlag.

[29] Ahmed A. Shabana. *Theory of Vibration, Volume II: Vibration of discrete and continuous systems*. Number v. 2 in Mechanical engineering series. Springer, 1997.

[30] Demetri Terzopoulost, John Platt, Alan Barr, and Kurt Fleischert. Elastically deformable models. *Computer Graphics*, 21:205–214, 1987.

[31] Yanzhen Wang, Yueshan Xiong, Kai Xu, Ke Tan, and Guangyou Guo. A mass-spring model for surface mesh deformation based on shape matching. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, GRAPHITE '06, pages 375–380, New York, NY, USA, 2006. ACM.

[32] Keith Waters. A muscle model for animation three-dimensional facial expression. *SIGGRAPH Comput. Graph.*, 21:17–24, August 1987.

[33] Andrew Witkin and David Baraff. An introduction to physically based modeling: Differential equation basics. *SIGGRAPH 2001 Course Notes*, 2001.

[34] O. C. Zienkiewicz. *The finite element method*. McGraw-Hill, London, 3d expanded and rev. ed. edition, 1977.