3^{rd} Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2006)
C. Mendoza, I. Navazo (Editors)

Local Constraint Methods for Deformable Objects

Marc Gissler Markus Becker Matthias Teschner

Computer Graphics, University of Freiburg, Germany

Abstract

We present a local scheme to enforce non-conflicting geometric constraints for dynamically deforming objects. The approach employs information on the underlying numerical integration method and we present a unified scheme to illustrate the incorporation of a variety of integration methods. The proposed technique is very efficient in terms of memory and computing complexity. Iterative solvers and stabilization techniques are avoided. The method is not subject to numerical drift or other inaccuracies and all constraints are accurately met at each simulation step. Since the approach does not require any pre-processing, dynamically changing constraints can be handled efficiently. Experiments indicate that thousands of constraints can be processed at interactive rates. Although our approach is restricted to non-conflicting constraints, experiments illustrate the versatility of the method in the context of deformable objects.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism: Animation

Keywords: physically-based modeling, constraints, deformable objects, numerical integration schemes, interactive applications

1. Introduction

Constraint techniques have been investigated for various applications ranging from graphical manipulation [Sut63] to robotics [LWP80]. In computer animation, constraints are applied to keyframe animation [LCG95], collision handling [MW88, KEP05], and resting contact [Bar89]. Constraints are used in dynamic simulations of deformable objects [LF04, MHTG05] and there exist sophisticated constraint solvers for articulated rigid bodies loops [RGL05, WTF06].

Many existing constraint approaches are global methods. These techniques solve for all constraints simultaneously and minimization techniques are employed to handle conflicting constraints. In the context of rigid bodies, global methods are appropriate since conflicting constraints commonly occur due to the small number of degrees of freedom. However, solving the resulting linear systems can be expensive in terms of memory and computing costs.

In contrast to rigid bodies, deformable objects are characterized by a larger number of degrees of freedom. If a deformable object is represented as a mass-point system, more than one non-conflicting constraint can be defined per object by distributing the constraints to different mass points. This easily allows to avoid conflicting constraints without sacrificing the versatility of the applied constraints. E. g., closed loops can be realized by attaching adjacent models to different mass points of an object. Thus, a large class of conflicting rigid-body constraints can be replaced by non-conflicting constraints for deformable mass-point systems.

In contrast to conflicting constraints, non-conflicting constraints can be solved locally and time-consuming iterative schemes are not required. Further, if information on the underlying numerical integration scheme is employed, non-conflicting constraints can be enforced very accurately. This idea has been introduced in [WTF06], where the particular form of the explicit Euler scheme is employed in the stabilization procedure for conflicting constraints.

Our contribution. We present a local approach to enforce non-conflicting geometric constraints for dynamically deforming mass-point systems. Information on the numerical integration method is employed and we present a generic scheme to illustrate the incorporation of a variety of integration methods such as explicit / implicit Euler, semi-implicit Euler-Cromer, midpoint method, Runge-Kutta 2, and Verlet.

The proposed technique is very efficient in terms of memory and computing complexity. While latest approaches to articulated rigid bodies require a simulation time of about one minute for 800 objects and 1000 constraints [WTF06], our approach processes environments of similar complexity at interactive rates. Iterative solvers and stabilization techniques are avoided. The method is not subject to numerical drift or other inaccuracies and all constraints are accurately met at each simulation step. Since the approach does not require any pre-processing, dynamically changing constraints can be handled efficiently.

Although our approach is restricted to simple non-conflicting constraints, experiments illustrate the versatility of the method in the context of deformable mass-point systems. The following constraints have been implemented: point to nail, points to point, point to line, point to triangle, point to tetrahedron, point to triangulated surface.

2. Related Work

There exist a variety of constraint solving techniques. In the context of inverse kinematics, e. g. in order to determine the movement of a robot arm, symbolic methods [LGHB88] or neural networks [FE96] are often used. However, these methods are less efficient for larger numbers of degrees of freedom and they are not able to respond to dynamic changes in the environment. In [MW88], an analytic constraint approach is presented which is based on the results of [Mac36]. In this method, attachment impulses are computed for joint constraints of articulated rigid bodies. This allows for multiple joints between two objects including sliding joints.

Further, there exist two main strategies to solve constraints numerically. The first technique computes forces or impulses using maximal coordinates. In this context, Lagrange multipliers or propagation methods have been proposed . In contrast, the second technique reduces the number of coordinates to represent the system state. The remaining coordinates are so-called reduced coordinates or generalized coordinates. Reduced coordinates are preferred in global approaches, if the number of constraints is large compared to the number of degrees of freedom.

Lagrange multipliers are based on d'Alembert's principal of virtual work. These approaches commonly compute forces that cancel out constraint-braking force components. Therefore, a linear system is solved for coefficients that describe the constraint forces with respect to an a-priori known basis [Bar96]. In [WGW90], Lagrange multipliers are used for constraints on linear deformations. In [GW91], Lagrange multipliers are used for constraints in physically-motivated drawing programs. In [MT92], Lagrange multipliers are used for imposing constraints on articulated objects consisting of rigid and deformable parts. In contrast to reduced coordinate approaches, Lagrange multipliers can handle non-holonomic constraints [Bar96]. In [WK88], a space-time approach is proposed that processes the physics and the kinematic constraints in a unified way. This approach solves a constrained optimization problem which calculates the forces. Recent approaches accelerate the solvers by employing the specific structure of the linear systems. In [Bar96], $\mathcal{O}(n)$ is achieved for sparse acyclic constraint systems of articulated figures. In [LF04], the calculation is accelerated by employing the specific form of the mass matrix.

In general, Lagrange multipliers are based on linear systems. Stabilization techniques such as Baumgarte's method [Bau72] are commonly applied to address drift problems. However, it is difficult to determine appropriate parameters for [Bau72]. Improved stabilization techniques have been proposed in [ACPR95] and [Fau98]. The method presented in [GC94] manipulates positions instead of computing forces. However, [Pla92] and [WTF06] note that the processing of positions and velocities tend to yield non-physical effects.

Alternatively, physically-motivated penalty methods have been proposed that compute constraint forces based on energy functions. In [BB88], energy functions are considered for self-assembling rigid bodies. In [PB88], this method is extended to deformable models. [PB88] is also able to handle multiple constraints with Lagrangian multipliers. In [Pla92], this approach has been extended to a pure global method which can also solve for time-varying and inequality constraints . In [WFB87], energy functions are employed in non-dynamic simulations of deformable models.

In general, energy-based methods can result in stiff equations if there are large forces counteracting a constraint. Further, the solvers can be trapped in local minima which can only be addressed by user interaction as described in [BB88].

In [IC87], reduced coordinates are proposed in the context of computer animation. Here, forward and inverse dynamics are combined for linked figures without closed loops. Therefore, a linear system is considered that relates accelerations and forces. In [IC88], a global extension is provided by combining generalized coordinates with Lagrange multipliers. In contrast to many other approaches, this method can process complex kinematic constraints, i. e. constraints that are functions of multiple degrees of freedom. In [HG97], a recursive constrained propagation algorithm based on reduced coordinates is presented. And in [HSO03], an alternative approach for contact constraints of deformable bodies is presented.

In addition to the discussed constraint solving techniques, there exist a variety of applications for constraints. In [TWK88], intrinsic and extrinsic constraints are used to recover 3D structures of non-rigidly moving objects. Constraints have been used for collision detection and non-penetration conditions [Bar89]. The method is generalized to flexible bodies for both collision handling and resting contact in [BW92]. A good survey of constraints in human-walking techniques is provided in [MFCD99]. And in [LCG95], constraints are used to combine keyframe techniques with physical simulation and collision handling. Further, [Pro95] applies elongation constraints to mass-spring systems to simulate rigid cloth behavior. Thorough surveys of existing constraint solving techniques can be found in [Gle94, Wit97].

Recently, a constraint approach for articulated rigid bodies has been presented in [WTF06]. This approach employs information on the underlying numerical integration scheme. While [WTF06] relies on the forward Euler integration scheme, we show how to incorporate information on a large variety of numerical integration methods into the constraint solving scheme. Further, our constraint approach is applied to deformable objects.

3. A generic representation for numerical integration schemes

In our approach, we consider dynamic mass point systems. Each mass point is characterized by its mass m_i, position $\mathbf{x}_i^t$, velocity $\mathbf{v}_i^t$, and force $\mathbf{F}_i^t$ with t denoting time. Further, we consider schemes that numerically integrate positions and velocities from time t to $t+h$ with h being a small time step. In the following, we focus on schemes that can be represented as

$$\begin{pmatrix} \mathbf{x}_i^{t+h} \\ \mathbf{v}_i^{t+h} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_i \\ \mathbf{B}_i \end{pmatrix} \mathbf{s}_i^t + \begin{pmatrix} c_i\,\mathbf{F}_i^t \\ d_i\,\mathbf{F}_i^t \end{pmatrix} \tag{1}$$

with system matrices $\mathbf{A}_i, \mathbf{B}_i \in \mathbb{R}^{3\times k}$, state vector $\mathbf{s}_i^t \in \mathbb{R}^k$, and $c_i, d_i \in \mathbb{R}$. The state vector $\mathbf{s}_i^t$ commonly represents the current position $\mathbf{x}_i^t$ and the velocity $\mathbf{v}_i^t$ of a mass point. However, some integration schemes require different or additional information from previous time steps in the state vector, e. g. velocity Verlet or Beeman. Therefore, the general form of the state vector is $\mathbf{s}_i^t \in \mathbb{R}^k$.

Now, the generic representation given in (1) is illustrated using the Verlet integration. In this scheme, positions and velocities are updated with

$$\begin{aligned} \mathbf{x}_i^{t+h} &= 2\mathbf{x}_i^t - \mathbf{x}_i^{t-h} + \frac{h^2}{m_i}\mathbf{F}_i^t \\ \mathbf{v}_i^{t+h} &= \frac{1}{2h}\left(\mathbf{x}_i^{t+h} - \mathbf{x}_i^{t-h}\right) \end{aligned} \tag{2}$$

where the velocity update can be rewritten as

$$\mathbf{v}_i^{t+h} = \frac{1}{h}\left(\mathbf{x}_i^t - \mathbf{x}_i^{t-h} + \frac{h^2}{2m_i}\mathbf{F}_i^t\right). \tag{3}$$

Using the state vector

$$\mathbf{s}_i^t = \begin{pmatrix} \mathbf{x}_i^t \\ \mathbf{x}_i^{t-h} \end{pmatrix} \tag{4}$$

we get

$$\begin{aligned} \mathbf{A}_i &= \begin{pmatrix} 2\mathbf{I}_3 & -\mathbf{I}_3 \end{pmatrix} & c_i &= \tfrac{h^2}{m_i} \\ \mathbf{B}_i &= \begin{pmatrix} \tfrac{1}{h}\mathbf{I}_3 & -\tfrac{1}{h}\mathbf{I}_3 \end{pmatrix} & d_i &= \tfrac{h}{2m_i} \end{aligned} \tag{5}$$

which represent the Verlet scheme if substituted into (1). Please note that (4) and (5) do not correspond to the state vector and system matrix of the Verlet scheme that would be used in a system analysis. Instead, the state vector is user-defined and the system matrix is determined accordingly. The left-hand side of (1) is always comprised of $\mathbf{x}_i^{t+h}$ and $\mathbf{v}_i^{t+h}$ since we want to impose constraints on these quantities. Nevertheless, alternative system matrices and state vectors could also be used to derive the constraint forces.

In the following, representations are given for further integration schemes, whereas the state vector is always defined as

$$\mathbf{s}_i^t = \begin{pmatrix} \mathbf{x}_i^t \\ \mathbf{v}_i^t \end{pmatrix}. \tag{6}$$

For the explicit midpoint method we get

$$\begin{aligned} \mathbf{A}_i &= \begin{pmatrix} \mathbf{I}_3 & h\mathbf{I}_3 \end{pmatrix} & c_i &= \tfrac{h^2}{2m_i} \\ \mathbf{B}_i &= \begin{pmatrix} \mathbf{0}_3 & \mathbf{I}_3 \end{pmatrix} & d_i &= \tfrac{h}{m_i}. \end{aligned} \tag{7}$$

For the explicit second-order Runge-Kutta scheme we get

$$\begin{aligned} \mathbf{A}_i &= \begin{pmatrix} \mathbf{I}_3 & h\mathbf{I}_3 \end{pmatrix} & c_i &= \tfrac{h^2}{2m_i} \\ \mathbf{B}_i &= \begin{pmatrix} \mathbf{0}_3 & \tfrac{h^2}{2m_i}\mathbf{J}_F(\mathbf{x}_i) \end{pmatrix} & d_i &= \tfrac{h}{m_i} \end{aligned} \tag{8}$$

with $\mathbf{J}_F(\mathbf{x}_i)$ denoting the Jacobian of $\mathbf{F}_i$ at $\mathbf{x}_i$. For the semi-implicit Euler-Cromer scheme we get

$$\begin{aligned} \mathbf{A}_i &= \begin{pmatrix} \mathbf{I}_3 & h\mathbf{I}_3 \end{pmatrix} & c_i &= \tfrac{h^2}{m_i} \\ \mathbf{B}_i &= \begin{pmatrix} \mathbf{0}_3 & \mathbf{I}_3 \end{pmatrix} & d_i &= \tfrac{h}{m_i} \end{aligned} \tag{9}$$

and for the implicit Euler we get

$$\begin{aligned} \mathbf{A}_i &= \begin{pmatrix} \mathbf{I}_3 & h\mathbf{I}_3 + \tfrac{h^3}{m_i}\mathbf{J}_F(\mathbf{x}_i) \end{pmatrix} & c_i &= \tfrac{h^2}{m_i} \\ \mathbf{B}_i &= \begin{pmatrix} \mathbf{0}_3 & \mathbf{I}_3 + \tfrac{h^2}{m_i}\mathbf{J}_F(\mathbf{x}_i) \end{pmatrix} & d_i &= \tfrac{h}{m_i} \end{aligned} \tag{10}$$

These examples illustrate how to represent a wide range of numerical integration schemes with the generic form given in (1). Although $\mathbf{A}_i$ and $\mathbf{B}_i$ are only approximate representations in some cases, e. g. for implicit or multi-step methods, we will show that these approximations do not compromise the accuracy of the constraints.

4. Constraint forces

This section describes how to derive the constraint forces. In the following, the goal is to compute a constraint force $\tilde{\mathbf{F}}_i^t$ in order to meet a user-defined constraint on a position or velocity which are denoted with $\tilde{\mathbf{x}}_i^{t+h}$ and $\tilde{\mathbf{v}}_i^{t+h}$, respectively.

$$\begin{pmatrix} \tilde{\mathbf{x}}_i^{t+h} \\ \tilde{\mathbf{v}}_i^{t+h} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_i \\ \mathbf{B}_i \end{pmatrix} \mathbf{s}_i^t + \begin{pmatrix} c_i \left(\mathbf{F}_i^t + \tilde{\mathbf{F}}_i^t\right) \\ d_i \left(\mathbf{F}_i^t + \tilde{\mathbf{F}}_i^t\right) \end{pmatrix} \quad (11)$$

By substituting $\mathbf{s}_i^t$, $\mathbf{A}_i$, $\mathbf{B}_i$, c_i , d_i, the constraint forces are adapted according to the actually employed numerical integration scheme. Thereby, the exact enforcement of constraints can be guaranteed.

4.1. Point-to-nail constraint

As an introductory example, we consider a constraint that enforces a given position $\mathbf{x}_{goal}$ for a mass point. From

$$\mathbf{x}_{goal} = \tilde{\mathbf{x}}_i^{t+h} = \mathbf{A}_i \mathbf{s}_i^t + c_i \left(\mathbf{F}_i^t + \tilde{\mathbf{F}}_i^t\right) \quad (12)$$

the constraint force $\tilde{\mathbf{F}}_i^t$ can easily be derived as

$$\begin{aligned} \tilde{\mathbf{F}}_i^t &= \frac{1}{c_i}\left(\mathbf{x}_{goal} - \mathbf{A}_i \mathbf{s}_i^t\right) - \mathbf{F}_i^t && (13) \\ &= \frac{1}{c_i}\left(\mathbf{x}_{goal} - \mathbf{x}_i^{t+h}\right). && (14) \end{aligned}$$

If the constraint is not met, i. e. $\mathbf{x}_i^t \neq \mathbf{x}_{goal}$, the constraint force moves the point exactly to the goal position. In the following integration step, the point is at its goal position, so the computed constraint force cancels the velocity of the point. All subsequent steps compute a constraint force that cancels the current force at the point. Thus, the constraint is enforced under all circumstances.

To illustrate the implementation of this constraint, we consider the Verlet scheme. In this case, the constraint force in (14) is computed based on (4) and (5) as

$$\tilde{\mathbf{F}}_i^t = \frac{m_i}{h^2}\left(\mathbf{x}_{goal} - 2\mathbf{x}_i^t + \mathbf{x}_i^{t-h} - \frac{h^2}{m_i}\mathbf{F}_i^t\right). \quad (15)$$

If this force is used in the integration step as given in (2) we get

$$\tilde{\mathbf{x}}_i^{t+h} = 2\mathbf{x}_i^t - \mathbf{x}_i^{t-h} + \frac{h^2}{m_i}\left(\mathbf{F}_i^t + \tilde{\mathbf{F}}_i^t\right) \quad (16)$$

and thus $\tilde{\mathbf{x}}_i^{t+h} = \mathbf{x}_{goal}$. In order to illustrate that the constraint force depends on the integration scheme we illustrate the midpoint method as a second example. Using the representation of the midpoint scheme as denoted in (6) and (7) we see that

$$\mathbf{x}_i^{t+h} = \mathbf{x}_i^t + h\mathbf{v}_i^t + \frac{h^2}{2m}\mathbf{F}_i^t \quad (17)$$

Therefore, the point-to-nail force in combination with the midpoint method is

$$\tilde{\mathbf{F}}_i^t = \frac{2m}{h^2}\left(\mathbf{x}_{goal} - \mathbf{x}_i^t - h\mathbf{v}_i^t - \frac{h^2}{2m}\mathbf{F}_i^t\right) \quad (18)$$

which is slightly different to the force that is computed for the Verlet scheme. Although (14) provides a general form to compute the point-to-nail force, the force adapts to specific integration schemes.

As mentioned in Sec. 3, the matrices $\mathbf{A}_i$ and $\mathbf{B}_i$ are approximate representations for some integration methods. However, these matrices are not required to compute the point-to-nail force as can be seen in (14). Instead of using the approximative representation, $\mathbf{x}_i^{t+h}$ can be computed with the actually employed integration method. Interestingly, the approximate representations $\mathbf{A}_i$ and $\mathbf{B}_i$ are always used to derive constraint forces, but are not used in the concluding computation rule as will be seen in the following sections.

4.2. Points-to-point constraint

Now, constraint forces are derived that hold n points $\mathbf{x}_i^t$ at a jointed position $\mathbf{x}_{goal} = \tilde{\mathbf{x}}_0^{t+h} = \tilde{\mathbf{x}}_1^{t+h} = \ldots = \tilde{\mathbf{x}}_{n-1}^{t+h}$. In contrast to the point-to-nail constraint, $\mathbf{x}_{goal}$ is neither fix nor user-defined. Instead, the joint position is implicitly considered in the n equations that are employed to compute the n constraint forces $\tilde{\mathbf{F}}_i^t$. The first equation

$$\tilde{\mathbf{F}}_0^t + \tilde{\mathbf{F}}_1^t + \ldots + \tilde{\mathbf{F}}_{n-1}^t = 0 \quad (19)$$

preserves the momentum of the mass point system. And for $\tilde{\mathbf{x}}_0^{t+h} = \tilde{\mathbf{x}}_i^{t+h}$ with $i = 1 \,..\, n-1$ we get additional $n-1$ equations for the constraint forces:

$$\mathbf{A}_0 \mathbf{s}_0^t + c_0 \left(\mathbf{F}_0^t + \tilde{\mathbf{F}}_0^t\right) = \mathbf{A}_i \mathbf{s}_i^t + c_i \left(\mathbf{F}_i^t + \tilde{\mathbf{F}}_i^t\right) \quad (20)$$

The linear system described by (19) and (20) results in the following constraint forces

$$\begin{pmatrix} \tilde{\mathbf{F}}_0^t \\ \tilde{\mathbf{F}}_1^t \\ \vdots \\ \tilde{\mathbf{F}}_{n-1}^t \end{pmatrix} = \frac{1}{u} \begin{pmatrix} e_0 & e_0 & \cdots & e_0 \\ e_1 & -u_1 & & e_1 \\ \vdots & & \ddots & \vdots \\ e_{n-1} & \vdots & e_{n-1} & -u_{n-1} \end{pmatrix} \begin{pmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_{n-1} \end{pmatrix} \quad (21)$$

with $e_i = \frac{c_0}{c_i}$, $\mathbf{b}_i = \frac{1}{c_i}\left(\mathbf{x}_i^{t+h} - \mathbf{x}_0^{t+h}\right)$, $u = \sum_i e_i$ and $u_i = u - e_i$. Thus, each $\tilde{\mathbf{F}}_i^t$ can be computed as

$$\begin{aligned} \tilde{\mathbf{F}}_i^t &= -\mathbf{b}_i + \frac{e_i}{u}\sum_j \mathbf{b}_j \\ &= -\frac{1}{c_i}\left(\mathbf{x}_i^{t+h} - \mathbf{x}_0^{t+h}\right) + \frac{e_i}{u}\sum_j \frac{1}{c_j}\left(\mathbf{x}_j^{t+h} - \mathbf{x}_0^{t+h}\right) && (22) \end{aligned}$$

These forces move the n points exactly to a joint goal position while preserving the momentum of the mass point system. To illustrate the actual implementation, we consider the Verlet integration scheme. In this case, we have

$$c_i = \frac{h^2}{m_i} \qquad e_i = \frac{m_i}{m_0} \qquad u = \frac{1}{m_0}\sum_j m_j \quad (23)$$

and the constraint forces can be computed by substituting (23) into (22) as

$$\tilde{\mathbf{F}}_i^t = -\frac{m_i}{h^2}\left(\mathbf{x}_i^{t+h} - \mathbf{x}_0^{t+h}\right) + \frac{m_i}{\sum_j m_j}\sum_j \frac{m_j}{h^2}\left(\mathbf{x}_j^{t+h} - \mathbf{x}_0^{t+h}\right)$$

$$= \frac{m_i}{h^2}\left(\frac{1}{\sum_j m_j}\sum_j m_j \mathbf{x}_j^{t+h} - \mathbf{x}_i^{t+h}\right) \tag{24}$$

Since the Verlet scheme is employed, the positions $\mathbf{x}_i^{t+h}$ are computed using (2). So, the constraint forces can be basically computed with an additional integration step for all affected points which is very efficient for explicit integration schemes.

By applying the constraint forces, we get the goal position

$$\begin{aligned}\mathbf{x}_{goal} = \tilde{\mathbf{x}}_i^{t+h} &= 2\mathbf{x}_i^t - \mathbf{x}_i^{t-h} + \frac{h^2}{m_i}\left(\mathbf{F}_i^t + \tilde{\mathbf{F}}_i^t\right)\\ &= \frac{1}{\sum_j m_j}\sum_j m_j \mathbf{x}_j^{t+h}\end{aligned} \tag{25}$$

which corresponds to the center of mass of the $\mathbf{x}_j^{t+h}$.

4.3. Constraining a point to a line, a face, and a tetrahedron

Now, we consider a mass point $\mathbf{x}_0^t$ and a corresponding position $\mathbf{y}^t$ on a line with $\mathbf{y}^t = \alpha_1\mathbf{x}_1^t + \alpha_2\mathbf{x}_2^t$, on a triangle with $\mathbf{y}^t = \alpha_1\mathbf{x}_1^t + \alpha_2\mathbf{x}_2^t + \alpha_3\mathbf{x}_3^t$, or in a tetrahedron with $\mathbf{y}^t = \alpha_1\mathbf{x}_1^t + \alpha_2\mathbf{x}_2^t + \alpha_3\mathbf{x}_3^t + \alpha_4\mathbf{x}_4^t$ with $0 \le \alpha_i \le 1$. In general, the corresponding position is given as $\mathbf{y}^t = \sum_i \alpha_i \mathbf{x}_i^t$ whereas the α_i are computed by projecting $\mathbf{x}_0^t$ onto a line or triangle or by computing the barycentric coordinates of $\mathbf{x}_0^t$ within a tetrahedron. It is not required that $\mathbf{x}_0^t$ equals $\mathbf{y}^t$. However, the approach only provides plausible results if $0 \le \alpha_i \le 1$. Now, we are looking for constraint forces such that

$$\tilde{\mathbf{x}}_0^{t+h} = \tilde{\mathbf{y}}^{t+h} = \sum_i \alpha_i \tilde{\mathbf{x}}_i^{t+h} \tag{26}$$

Therefore, we compute a constraint force $\tilde{\mathbf{F}}_0^t$ for the point $\mathbf{x}_0^t$ and apply the forces $-\alpha_i\tilde{\mathbf{F}}_i^t$ to the points $\mathbf{x}_i^t$. Since $\sum_i \alpha_i = 1$, the momentum is preserved. Further, a two-way coupling is realized by this scheme. Substituting (11) into (26)

$$\mathbf{A}_0\mathbf{s}_0^t + c_0\left(\mathbf{F}_0^t + \tilde{\mathbf{F}}_0^t\right) = \sum_i \alpha_i\left(\mathbf{A}_i\mathbf{s}_i^t + c_i\left(\mathbf{F}_i^t - \alpha_i\tilde{\mathbf{F}}_i^t\right)\right) \tag{27}$$

we get the constraint force

$$\begin{aligned}\tilde{\mathbf{F}}_0^t &= \frac{1}{c_0 + \sum c_i\alpha_i^2}\left(\sum_i \alpha_i\left(\mathbf{A}_i\mathbf{s}_i^t + c_i\mathbf{F}_i^t\right) - \mathbf{A}_0\mathbf{s}_0^t - c_0\mathbf{F}_0^t\right)\\ &= \frac{1}{c_0 + \sum c_i\alpha_i^2}\left(\sum_i \alpha_i\mathbf{x}_i^{t+h} - \mathbf{x}_0^{t+h}\right)\end{aligned} \tag{28}$$

Again, the potentially approximate parts of the generic representation of the numerical integration schemes (1) are avoided in the calculation rule.

4.4. Point-to-surface constraint

Since we can attach a point to a point, a line, and a face, it is straightforward to implement a constraint that holds a moving mass point on a triangulated surface. As described in Sec. 4.3, a two-way coupling of the connected components is realized. To determine the corresponding point, line, or face on a triangulated surface, we employ the closest point transform presented in [Mau00].

4.5. Self-assembling

The presented constraints can also be used for the self-assembling of objects. In order to avoid stability problems in the case of large initial distances between object parts, the constraint forces can be applied gradually. In our implementation, the user can specify the number of time steps n until the actual constraint force is applied. For the time steps $t + jh$ with $1 \le j \le n$, a gradually growing constraint force is computed as $\frac{j}{n}\tilde{\mathbf{F}}_i^t$ with $\tilde{\mathbf{F}}_i^t$ being the actual constraint force. Nevertheless, the constraint solving technique is not iterative.

5. Results

A variety of test scenarios has been implemented to illustrate the efficiency and the versatility of the proposed constraint technique. Therefore, we have combined the constraint approach with a simple and efficient deformable modeling approach for tetrahedral meshes [THMG04]. In all experiments, the simulated tetrahedral mesh is geometrically coupled with a triangulated surface mesh. The explicit Verlet scheme is used and all tests have been performed on an Intel Pentium 4 PC, 3.4 GHz.

Basically, all presented constraints require the computation of one additional integration step per point that is involved in a constraint. This additional integration step is performed to compute $\mathbf{x}_i^{t+h}$ which is used to determine $\tilde{\mathbf{F}}_i^t$. So, the performance of the proposed constraint technique mainly depends on the number of points that are involved in a constraint and on the efficiency of the integration scheme. In our experiments, we have employed the explicit Verlet scheme which results in approximately 1000 points that can be processed in one millisecond. Thus, about 1000 point-to-nail constraints or 500 points-to-point constraints with two points per constraint or 250 points-to-triangle constraints with four points per constraint or 200 points-to-tetrahedron constraints with five points per constraint can be computed in one millisecond. On the other hand, 1000 points involved in one points-to-point constraint require the same amount of time.

Fig. 1 illustrates the test scenario that we have used for performance measurements. The top level row of the cubes is fixed with point-to-nail constraints and all cubes are interconnected by points-to-point constraints with two points per constraint. The number of cubes and constraints can be varied and Fig. 2 shows the measurements for up to 9990 cubes with 49950 tetrahedrons and 39924 constraints. Up to 25000 constraints are processed within 50*ms* and the entire dynamics of 2990 cubes with 14950 tetrahedrons and 11924 constraints are computed in 46.15*ms* including force computation, constraint computation and numerical integration.

Fig. 3 depicts an application for point-to-surface constraints. In this case, the computation of 14 points-to-point and 2 point-to-surface takes 1.54*ms* which is comparatively expensive due to the required closest point transform. Nevertheless, the scenario can be simulated at interactive rates including visualization. Fig. 4 illustrates simulation sequences with dynamically changing constraints. Constraints can be activated or deactivated during a simulation. The set of constraints can be changed interactively. Again, these simulations can be computed and visualized at interactive rates.

6. Conclusion

We have presented local constraint methods for deformable objects. Information on the underlying numerical integration scheme is employed to efficiently and accurately solve the constraints. A generic scheme has been presented to illustrate the incorporation of a variety of integration methods. Iterative solvers and stabilization techniques are avoided. Our method is not subject to numerical drift or other inaccuracies. Since the approach does not require any pre-processing, dynamically changing constraints can be handled efficiently.

Although our approach is restricted to local constraints, the experiments illustrate the versatility of the method in the context of deformable mass-point systems. All illustrated simulations have been computed at interactive rates. Ongoing research focuses on constraint applications such as collision handling, fracture, and cutting which might be of interest for interactive animations in games or surgical simulators. Further, we intend to investigate nonlinear effects that might occur for integration schemes that are approximately represented with the presented generic scheme.

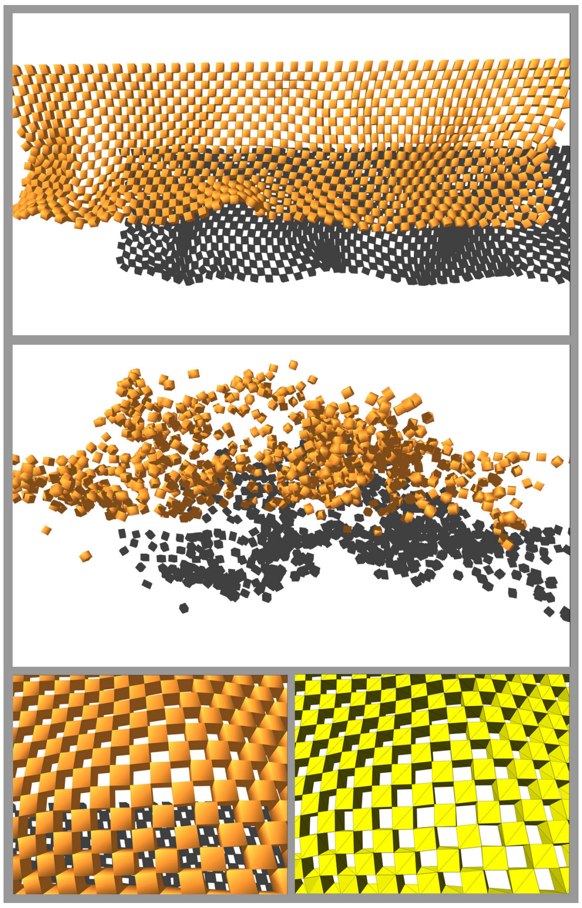
Figure 1: *Cubes with points-to-point and point-to-nail constraints. All constraints are active in the first image. In the second image, all constraints are deactivated. Further, the underlying tetrahedral structure for the dynamic simulation is shown. 3950 tetrahedrons, 160 point-to-nail and 2964 points-to-point constraints are used in this sequence. The forces $\mathbf{F}_i^t$ based on [THMG04]are computed in 3.42ms, constraint forces $\tilde{\mathbf{F}}_i^t$ take 5.38ms, and the Verlet integration takes 3.2ms. So, one timestep can be simulated in 12ms. For the performance measurements shown in Fig. 2 the number of cubes and constraints is varied.*

References

[ACPR95] ASCHER U., CHIN H., PETZOLD L., REICH S.: Stabilization of constrained mechanical systems with DAEs and invariant manifolds. *J. Mech. Struct. & Mach 23* (1995), 135–157. 26

[Bar89] BARAFF D.: Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Proc. of ACM SIGGRAPH '89* (1989), pp. 223–232. 25, 26

[Bar96] BARAFF D.: Linear-time dynamics using lagrange multipliers. In *Proc. of ACM SIGGRAPH '96* (1996), pp. 137–146. 26

[Bau72] BAUMGARTE J.: Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering 1* (1972), 1–16. 26

[BB88] BARZEL R., BARR A. H.: A modeling system based on dynamic constraints. In *Proc. of ACM SIGGRAPH '88* (1988), pp. 179–188. 26

[BW92] BARAFF D., WITKIN A.: Dynamic simulation of non-penetrating flexible bodies. In *Proc. of ACM SIGGRAPH '92* (1992), pp. 303–308. 26

[Fau98] FAURE F.: Interactive solid animation using linearized displacement constraints. In *Proc. of EG Workshop on Computer Animation and Simulation (EGCAS)* (1998), pp. 61–72. 26

[FE96] FERREIRA A., ENGEL P.: Positioning a Robot Arm: An Adaptive Neural Approach. In *Int. Workshop on Neural Networks for Identification, Control, Robotics,*

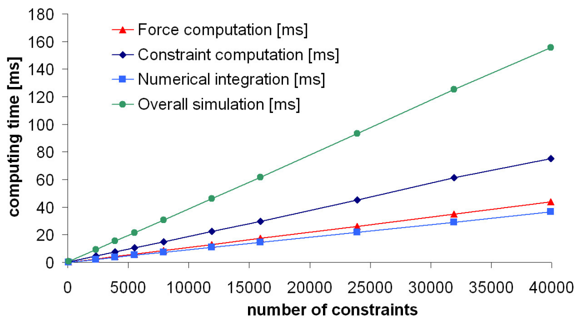

Figure 2: *This graph illustrates the performance measurements using the test scenario shown in Fig. 1. The number of objects and constraints is varied and measurements for up to 39924 constraints are shown which corresponds to 9990 cubes with 49950 tetrahedrons. Up to 25000 constraint forces* $\tilde{\mathbf{F}}_i^t$ *are computed at interactive rates. Further, the entire dynamics of 2990 cubes with 14950 tetrahedrons and 11924 constraints is computed in* 46.15*ms.*

and Signal/Image Processing (NICROSP '96) (1996), pp. 440–448. 26

[GC94] GASCUEL J.-D., CANI M.-P.: Displacement constraints for interactive modeling and animation of articulated structures. *The Visual Computer 10*, 4 (1994), 191–204. 26

[Gle94] GLEICHER M.: *A Differential Approach to Graphical Manipulation*. PhD thesis, Carnegie Mellon University, 1994. 27

[GW91] GLEICHER M., WITKIN A.: Differential Manipulation. In *Proc. of Graphics Interface* (1991), pp. 61–67. 26

[HG97] HUMMEL A., GIROD B.: Fast dynamic simulation of flexible and rigid bodies with kinematic constraints. In *Proc. of ACM VRST '97* (1997), pp. 125–132. 26

[HSO03] HAUSER K. K., SHEN C., O'BRIEN J. F.: Interactive deformation using modal analysis with constraints. In *Proc. of Graphics Interface* (2003), pp. 247–256. 26

[IC87] ISAACS P. M., COHEN M. F.: Controlling dynamic simulation with kinematic constraints. In *Proc. of ACM SIGGRAPH '87* (1987), pp. 215–224. 26

[IC88] ISAACS P. M., COHEN M. F.: Mixed methods for complex kinematic constraints in dynamic figure animation. *The Visual Computer 4*, 6 (1988), 296–305. 26

[KEP05] KAUFMAN D. M., EDMUNDS T., PAI D. K.: Fast Frictional Dynamics for Rigid Bodies. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH '05) 24*, 3 (2005), 946–956. 25

[LCG95] LAMOURET A., CANI M.-P., GASCUEL J.-D.: Combining physically-based simulation of colliding objects with trajectory control. *Journal of Visualization and Computer Animation (JVCA)* (1995). 25, 27

[LF04] LENOIR J., FONTENEAU S.: Mixing deformable and rigid-body mechanics simulation. In *Proc. of Computer Graphics International* (2004), pp. 327–334. 25, 26

[LGHB88] LUIS G. HERRERA-BENDEZU EDUARDO MU J. T. C.: Symbolic computation of robot manipulator kinematics. In *Proc. of IEEE Conference on Robotics and Automation* (1988), pp. 993–998. 26

[LWP80] LUH J. Y. S., WALKER M. W., , PAUL R. P. C.: On-line computational scheme for mechanical manipulators. *ASME Journal of Dynamic Systems, Measurement, and Control 102* (June 1980), 69–76. 25

[Mac36] MACMILLAN W. D.: *Dynamics of Rigid Bodies*. Dover Publications, Inc, 1936. 26

[Mau00] MAUCH S.: A fast algorithm for computing the closest point and distance function. *Technical Report, CalTech, unpublished* (2000). 29

[MFCD99] MULTON F., FRANCE L., CANI M.-P., DEBUNNE G.: Computer animation of human walking: a survey. *Journal of Visualization and Computer Animation (JVCA) 10* (1999), 39–54. 26

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless Deformations Based on Shape Matching. *ACM Transactions on Graphics 24*, 3 (2005), 471–478. 25

[MT92] METAXAS D., TERZOPOULOS D.: Dynamic deformation of solid primitives with constraints. In *Proc. of ACM SIGGRAPH '92* (1992), pp. 309–312. 26

[MW88] MOORE M., WILHELMS J.: Collision detection and response for computer animation. 289–298. 25, 26

[PB88] PLATT J. C., BARR A. H.: Constraints methods for flexible models. In *Proc. of SIGGRAPH '88* (1988), pp. 279–288. 26

[Pla92] PLATT J.: A generalization of dynamic constraints. *CGVIP: Graphical Models and Image Processing 54*, 6 (1992), 516–525. 26

[Pro95] PROVOT X.: Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Proc. of Graphics Interface '95* (1995), pp. 147–154. 27

[RGL05] REDON S., GALOPPO N., LIN M. C.: Adaptive dynamics of articulated bodies. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH '05) 24*, 3 (2005), 936–945. 25

[Sut63] SUTHERLAND I.: *Sketchpad: A ManMachine Graphical Communication System*. PhD thesis, Massachusetts Institute of Technology, 1963. 25

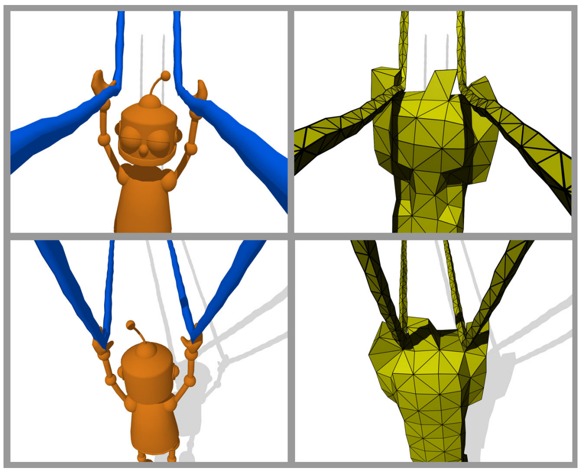

Figure 3: *Point-to-surface constraints are used to attach two points of the tetrahedral mesh of the robot to the rope models. This enables the robot to slide along the ropes. 1339 tetrahedrons, 14 points-to-point, and 2 point-to-surface constraints are used. The forces* $\mathbf{F}_i^t$ *are computed in* 0.81*ms, constraint forces* $\tilde{\mathbf{F}}_i^t$ *take* 1.54*ms, and Verlet takes* 1.17*ms.*

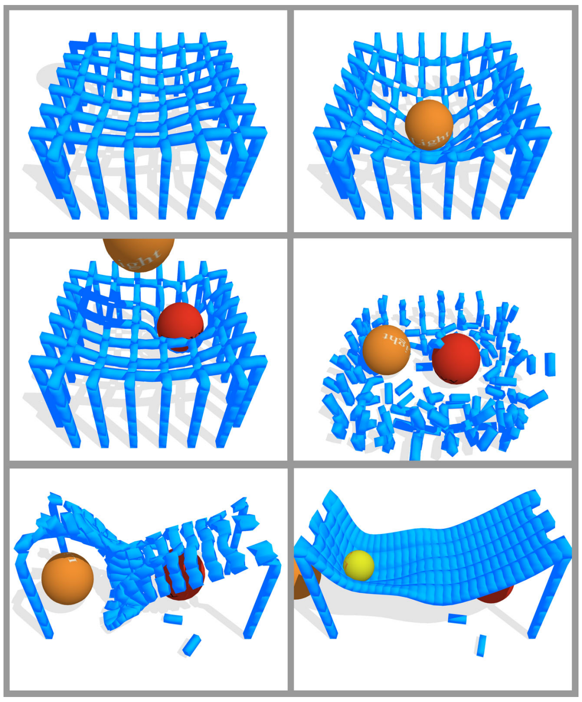

Figure 4: *Dynamically changing constraints can be handled. The constraints of the net are deactivated and new constraints for the bridge are activated. 2760 tetrahedrons and 1584 (net) or 1320 (bridge) constraints are used. The forces* $\mathbf{F}_i^t$ *are computed in* 2.16*ms, constraint forces* $\tilde{\mathbf{F}}_i^t$ *take* 1.6*ms (net) and* 2.38*ms (bridge), and Verlet takes* 1.13*ms.*

[THMG04] TESCHNER M., HEIDELBERGER B., MÜLLER M., GROSS M. H.: A versatile and robust model for geometrically complex deformable solids. In *Proc. of Computer Graphics International* (2004), pp. 312–319. 29, 30

[TWK88] TERZOPOULOS D., WITKIN A., KASS M.: Constraints on deformable models: Recovering 3D shape and nonrigid motion. *Artificial Intelligence 36*, 1 (1988), 91–123. 26

[WFB87] WITKIN A., FLEISCHER K., BARR A.: Energy constraints on parameterized models. In *Proc. of SIGGRAPH '87* (1987), pp. 225–232. 26

[WGW90] WITKIN A., GLEICHER M., WELCH W.: Interactive dynamics. In *Proc. of Symposium on Interactive 3D Graphics SI3D '90* (1990), pp. 11–21. 26

[Wit97] WITKIN A.: An introduction to physically based modeling: Constrained dynamics. *ACM SIGGRAPH 97 Tutorial Notes* (1997). 27

[WK88] WITKIN A., KASS M.: Spacetime constraints. In *Proc. of SIGGRAPH '88* (1988), pp. 159–168. 26

[WTF06] WEINSTEIN R., TERAN J., FEDKIW R.: Dynamic Simulation of Articulated Rigid Bodies with Contact and Collision. *IEEE TVCG 12*, 3 (2006), 365–374. to appear. 25, 26, 27