EUROGRAPHICS '96 / J. Rossignac and F. Sillion (Guest Editors), Blackwell Publishers

Volume 15, (1996), Number 3

Physically-Based Deformations Constrained in Displacements and Volume

E. Promayon[1], P. Baconnier[1], C. Puech[2]

[1]TIMC, IMAG-CNRS UMR 5525, Faculté de Médecine, Institut Albert Bonniot, 38706 La Tronche cedex and [2] iMAGIS*, GRAVIR-IMAG, BP53,38041 Grenoble cedex 09 (France)
e-mail: Ernmanuel.Promayon @imag.fr

Abstract:

This paper presents a method of constraining physically-based deformable objects. In this method, an object can be defined locally in terms of kinetic and dynamic (mass, position, speed), and physical parameters (compressibility, elasticity, motor functioning) . Several problems are solved: constant volume deformation, displacement constraints (fixed or moving required positions), and real object modelling. An object is described by a set of mass points on its contour. The evolution algorithm runs in two phases dealing successively with forces and constraints (which are presented as reaction forces).
The main contribution of the method is the control of object volumes during evolution. We define a function that explicitly gives the inside volume of an object in order to use it as a constraint. Thus, the volume can be kept exactly constant during deformation without using an iterative process, in opposition to lagrangian approaches. Some results are illustrated by examples at the end of the paper.

Keywords: Dynamic simulation, constant volume, physically-based modelling, constraint, deformable models, shape memory, elasticity.

1 Introduction

1.1 Motivation

Our simulation attempts to realistically simulate respiratory movements in order to be an educational and a medical tool. Our aim is to develop an anatomical and functional model of the evolutions of the human trunk structures during respiration. The most critical part in the respiration simulation is the abdomen and diaphragm set since these parts are active deformable bodies. The abdomen is an incompressible deformable object which has a rest position (due to its surrounding muscles and to the pelvis and spine skeleton). It can be compared to a balloon full of water surrounded by an elastic membrane.
Such a simulation should run on segmented data from medical imaging systems and computations should be fast enough. In our study, we want to simulate a complex deformable 3D object from: anatomical data, functional data (passive and active, normal or pathological mechanical properties), and particular situations or scenarii (like body position or motion). In a computer model, we can express these as topological parameters and constraints on the objects of the simulation. Thus, we need an adaptative dynamic model that can be over-constrained (for example, set position constraints and/or constant volume constraints). We describe here a physically-based model that manages multiple constraints and takes into account our particular needs.

1.2 Related work

Physically-based models are well suited to simulate natural motion and flexible elastic objects ([Ter87] [Ter89]). Given a small amount of description data, complex motions can be generated using dynamics laws. Because of the lack of direct control in the object positionning or trajectory, the need for constraining the objects of the scene rapidly appears.

---

* iMAGlS is a joint project of CNRS, INRIA, INPG and UJF

In [Pla88], three methods of constraining flexible models are presented: reaction constraints, penalty methods and augmented lagrangian methods. Reaction constraints combine the projection method and dynamics. It is the only method that exactly fulfills the constraints. No extra differential equation is required. The authors explore constrained minimization methods (like the use of lagrangian multiplier) and adapt them to the flexible object simulation problem. Such a simulation is described in terms of differential equation systems (like in [Ter87]). Augmented lagrangian adds a term to the differential equation systems. A constraint is exactly met when this term is null. The procedure consists in progressively finding a minimum.
Three main approaches can also be found to constrain physically-based articulated rigid bodies. In [Bar88], rigid bodies are composed of simple objects (like rods) connected together by geometric constraints. Reaction forces are computed by solving differential equations. [Van91] presents another iterative approach: the distinction is made (like in [Gas94]) between constraint forces and external forces. [Van91] solves a set of equations where constraints are taken into account by iteratively calculating the yielding constraint forces. [Gas94] avoids an explicit computation of these forces by iteratively tuning the solid positions.
Our approach consists in discoupling normal "finite" forces and constraint forces, combined with the reaction constraint approach. However, neither object energy nor differential equation systems are computed. Object properties and constraints are only defined by forces. These forces are applied to each mass point that composes the object. We avoid the use of time-consuming procedures (differential equations). We extend the reaction constraint notion presented in [Pla88] to meet global and multiple constraints. All constraints that can be defined as a region in the position or velocity space, and that are differentiable, can be used in our method. This includes constant volume constraints, fixed or moving required position constraints, and pre-defined trajectory or velocity constraints.
Related work includes two somewhat different approaches: on one hand the spring-mass network approach which can be constrained by adding local controlers [Jim93] [Del93], on the other hand a geometrically designed model (not physically-based and using Free-Form Deformations, FFD) which iteratively minimizes an error term to deal with the constant volume constraint [Aub95]. In [Rap95], the volume of a FFD using Bézier solids is kept constant during interactive deformations. The algorithm iteratively minimizes the error on constraints (constant volume and continuity of Bézier primitives) by using the lagrangian multiplier method. The method presented in this paper can be applied in FFD modelling to solve the constraint problem in only one iteration.
Section *2* of this paper develops our choice for the object model and the forces that make it move. In section 3, we explain with more details our resolution algorithm and how the constant volume constraints can be exactly satisfied.

2 Model description

We describe our model in three parts: the first section deals with geometrical design, the second section introduces dynamics in the model (masses, forces and simulation), the third section presents some force examples (muscle contraction and elastic recall).

2.1 Geometrical description

Each object of the scene is represented by points that discretize the object contour. This allows for direct control of shapes of objects. The main drawback may come from complex scenes. Despite this, the simplification coming from the point discretization overcomes time-consuming problems.
Each vertex knows its neighbours. The surface of the object is defined by a triangular mesh. Thus, image segmentation algorithms and reconstruction algorithms (see [Aya96], [Lac94]) can directly be used in a first step as object modeler. We can also use 3D point acquisition systems [Fer94]. The use of triangles in order to define our object external surface will largely simplify our constant volume constraint computations (see section 3.4). The method can also be applied with objects defined by polygonal faces.

2.2 Dynamic

Physical dynamic descriptions are based on two basic notions: material points and forces that induce movement. We describe here how these two notions are included in our model and present a first simulation algorithm.

2.2.1 Material points

A mass is assigned to each contour point, and dynamic forces will generate its movements. A given object is described by three vectors: a position vector (in euclidian space): $X = (x_1, \cdots, x_n, y_1, \cdots, y_n, z_1, \cdots, z_n)^t$, a velocity vector: $\dot{X} = \frac{\partial X}{\partial t} = (\dot{x}_1, \cdots, \dot{x}_n, \dot{y}_1, \cdots, \dot{y}_n, \dot{z}_1, \cdots, \dot{z}_n)^t$, and a mass vector: $M = (m_1, \cdots, m_n)^t$. The size of X and $\dot{X}$ is 3n (where n is the number of points of the object). The size of the mass vector is n.

2.2.2 Forces

The evolution of an object in the scene depends on the forces that act on its mass points. Two types of forces are modelled: external forces (like gravity field) and internal forces. Internal forces allow to describe actual properties such as elasticity, motor functioning, and compressibility. The forces presented in this section are finite forces, their directions and intensities are known. External and internal forces can be classified in three categories:

- *forcefields:* the force vector $\vec{F}_{field}$ is known on each object point (e.g. gravitation field) $\vec{F} = k_{field} \cdot \vec{F}_{field}$;
- *locally applied forces:* a particular force $\vec{F}_{local}$ (e.g. user manipulation) $\vec{F} = k_{local} \cdot \vec{F}_{local}$ is exerted on a
- *force point, induced by attractive positions:* in this case, we only know the position P_{ideal} where point P should be located. This position is fixed or moving. This kind of force is used to create point displacements that minimize a distance (e.g. muscle contraction force). If P is the current point position in euclidean space and P_{ideal} its attractive position, the force $\vec{F} = -k_{attract} \cdot (P - P_{ideal})$ should be applied to point P. Thus, we create an elastic force so that P is attracted by P_{ideal}.

All forces act on mass points of the objects. The weight k_i of each force parametrizes the model.

2.2.3 Simulation

The evolution of the objects is determined by integrating through time the forces applied to each mass point. If no constraints are applied to mass points, the evolution algorithm is classic. At each time step, we compute the sum of forces and we use the Euler integration scheme for equations of motion.

2.3 Forces envolved in our simulation

The simulation uses the following forces:

- **gravitational forces:** on each point $\vec{F}_g = k_g \vec{g}$ (where $\vec{g}$ is the physical acceleration constant).
- **muscular force:** in the diaphragm and the intercostal muscles, see section 2.3.1
- **shape memory force:** to render elastic behaviours, see section 2.3.2.

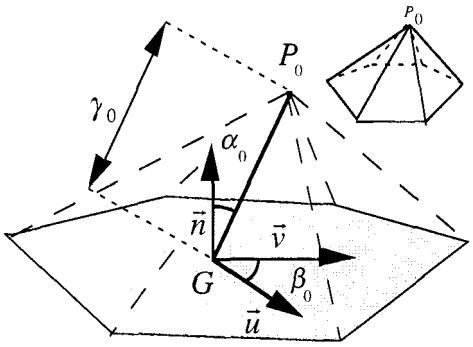
Section 2.3.3 presents how attractive positions can help to control animation.

2.3.1 Muscular force

Our aim is not to develop a complete model of the muscle (as in [Che92]), but to simulate muscular behaviour with our objects. Contraction is simply the diminution of the distance between a mass point and some of its neighbours. The contraction direction determines which neighbours are used. In order to model the muscular force, these neighbours are considered as attractive points. For each used neighbour point, we apply a force $\vec{f}_i = -k_{muscle} \cdot (P - P_{neigh(i)})$ to the mass point P. $P_{neigh(i)}$ is the i^{th} neighbour of P used in the contraction. The weighting parameter k_{muscle} defines the tissue contraction. During the evolution of the object, the use of this force stiffness k_{muscle} can increase or decrease muscle action.

2.3.2 Shape memory: elastic recall

The elastic properties of an object can be defined as its ability to come back in its original shape once deformed. The shape is locally defined by using the position configuration of a given point P and its neighbours (see figure 1). This configuration is used to define a local coordinate system. At each point, three values characterize this local coordinate system: α_0, β_0 and γ_0. We define a plane "close to" the neighbours of P. This plane is called "the approximate neighbours plane".



- $\vec{n}$ is the unit vector normal to the approximate neighbours plane,
- $\vec{v}$ is a unit vector normal to $\vec{n}$ (therefore it is in the approximate neighbours plane),
- G is the isobarycenter of the neighbours of point P.

α_0, β_0 and γ_0 are determined once, at rest shape: α_0 is the angle between $\vec{n}$ and $\overrightarrow{G_0P_0}$, β_0 the angle between $\vec{u} = \vec{n} \times \overrightarrow{G_0P_0}$ and a vector $\vec{v}$, and $\gamma_0 = \|G_0P_0\|$.

Figure 1: the local shape coordinate system

If the surface is exactly planar then $\alpha_0 = \pi / 2$. If $G_0 = P_0$, a special flag is set. This local coordinate system is different from the classical spherical coordinate system because it is defined by the relative neighbour positions (computing the inertia matrix of neighbour points is not required). During the simulation, the local coordinate system can be used to determine $P_{shape\ mem}$. This point is the ideal position for P in order to fit the rest state. If $P = P_{shape\ mem}$, the object is not locally deformed.

We can always find this ideal position by computing : $\vec{n}$ (deducing $\vec{v}$), G from neighbouring positions, α_0, β_0 and γ_0. This gives the shape memory force: $\vec{F}_{elastic} = -k_{elastic} \cdot \left(P - P_{shape\ mem}\right)$; P is attracted by $P_{shape\ mem}$. This model does not use a deformation energy like in [Ter87]. The elastic behaviour is locally modelled by a force on each point. This force tends to put the point at its rest configuration. It verifies the elasticity theory: it is null for a rigid displacement (in this case $P = P_{shape\ mem}$). The observed convergence is better with our elastic shape memory than with springs linking each neighbouring point. The parameter $k_{elastic}$ gives us an easy mean to model local properties of an object.

2.3.3 Animation control with attractive forces

The force generated by an attractive position gives a mean to handle animation control (path following or key-frame guiding). A similar idea can be found in [Lam95]. At each time step, an ideal position can be found on the path. A spring force is applied between this ideal position and the mass point P.

3. Constraints as reaction forces

As explained before, the model presented in section 2 has to be constrained to achieve our simulation goal. This section explains how constraints can be added to our model.

Constraints can be applied to our deformable object by two means:

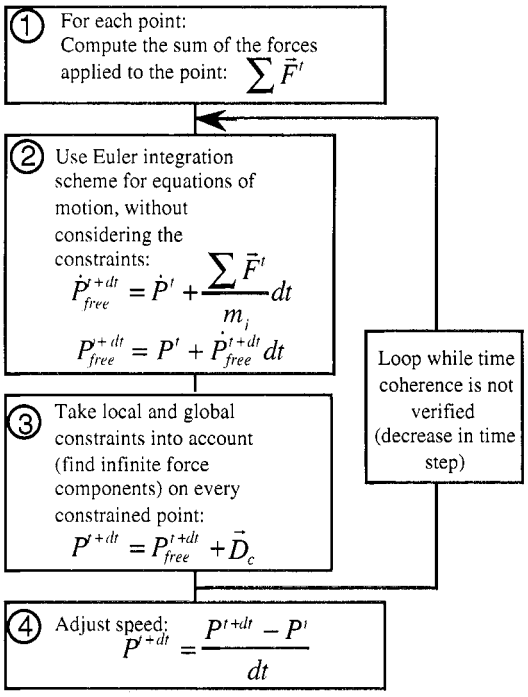
- *local constraint* (applied on a point) :
  - fixed or moving required positions : a point is nailed in the euclidean space or has to rigidly follow a path,
  - enforced regions : a point has to stay in a given region of the space, but can freely move inside.
- *global constraint:* a set of mass points of an object has to verify a given constraint. As we will see later, it is the case for the constant volume constraint.

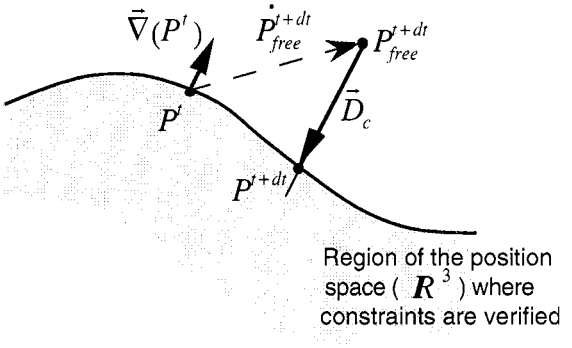
So far, we have shown how to generate finite forces on mass points, and how to compute the resulting movement. Let us now examine how we can manage constraints. Section 3.1 gives a resolution algorithm to solve constraints. Section 3.2 explains how the algorithm steps can be performed. Section 3.3 focuses on multiple constraint handling. The last section (section 3.4) deals with a particularly interesting constraint: the constant volume constraint.

3.1 Resolution algorithm

Constraints can be considered as non-quantified force components. For example, nailing a point at a position in space can be considered as an elastic spring force between a geometric space point and a material point

with a stiffness that avoids any movement. Even if a constraint can conceptually be considered as a reaction force (see [Pla88]), the only thing to do is to compute the movement of mass points in order to verify the constraints (see [Gas94]). In [Gas94][Van91]objects are articulated rigid bodies: their movements have to be directly or indirectly decomposed in small displacements that iteratively verify the constraints. As our objects are deformable, a solution can be directly found in only one step.
For each time step, the resulting processing algorithm can be decomposed as show in figure 2.





P_{free}^{t+dt} is an intermediate position that does not appear on the screen. It is the position computed without taking the constraints into account.

Finding a displacement vector $\vec{D}_c$ corresponding to a constraint is equivalent to find the point that minimizes the distance between P_{free}^{t+dt} and the constraint region (in position or velocity space). See also section 3.2.

Figure 2: projection on constraint method and its resolution algorithm.

The time coherence in the loop is verified when $\left\| P^t P_{free}^{t+dt} \right\| \geq \left\| P^t P^{t+dt} \right\|$. Thus, dealing with constraints does not move the object too far from its previous position. If time coherence is not verified, the time step is decreased. We explain in section 3.2 how the vector $\vec{D}_c$ can be found.

3.2 Finding $\vec{D}_c$, the projection vector

The vector $\vec{D}_c$ can be seen as a projection vector. Thus, our method belongs to the projection on constraint function method. Every constraint that represents a region in the parameter space (position or velocity space) can be computed like this. The constraint is exactly satisfied when we can find the nearest point from P_{free}^{t+dt} in the constraint region. This is done by projecting P_{free}^{t+dt} on the constraint region along the gradient vector (see figure 2). In opposition to the lagrangian method where an approximate solution is found with a given error after some iterations, our algorithm finds the exact solution of the constraint problem in one pass. The only restriction is that the constraint function has to be differentiable.

Given some constraints, we must define the corresponding region in the position (or velocity) space. This space is a $3n_c$ space (where n_c is the number of constrained points). In the remainder of this part, we will consider a constraint applied in the position space on n_c points (a constraint applied to the velocity space can similarly be derived). Let Y be the position vector (which size is $3n_c$). Y holds all the points subject to

a given constraint. Let Y^t and Y^{t+dt} be respectively the position vectors at time t and $t+dt$. Y^{t+dt}_{free} is the intermediate vector computed in the second phase of the algorithm (see figure 2).

Two cases appear:

- *the region is a point* Y_c*:* we simply have $\vec{D}_c = \overrightarrow{Y^{t+dt}_{free} Y_c}$ e.g. fixed position or assigned path constraints,
- *the region is analytically defined:* e.g. enforced geometric region or constant volume constraints.

In this latter case, the region gradient is used. The gradient $\vec{\nabla}^{t+dt}$ in Y^t (or in Y^{t+dt}_{free} if known) is the normal vector of the constraint region frontier. Knowing the constraint function $\Phi_c(Y)$ in the position space, we can find the gradient vector in each point of the position space: $\vec{\nabla}(Y) = \frac{\partial \Phi_c(Y)}{\partial Y}$ ($\Phi_c(Y)$ has to be differentiable).

Hence, we have: $\vec{\nabla}^{t+dt} = \frac{\partial \Phi_c(Y^{t+dt})}{\partial Y^{t+dt}}$. The resulting equation system at time t is:

$$\begin{cases} Y^{t+dt} = Y^{t+dt}_{free} + \vec{D}_c = Y^{t+dt}_{free} + \varphi \cdot \vec{\nabla}^{t+dt} \\ \Phi_c(Y^{t+dt}) = 0 \end{cases}.$$

This system has $3n_c + 1$ equations and $3n_c + 1$ unknowns (Y^{t+dt} and φ).
In section *3.4*, this idea is explained and illustrated with the example of the constant volume constraint.

3.3 Multiple constraints

If a mass point P is locally and globally constrained, the processing order is:

- find the position that verifies the local constraints,
- take point P into account in the global constraint resolution, but do not project it on the constraint function (set its corresponding components in the gradient vector to zero).

If points are over-constrained (locally or globally) priority beween the constraints are defined. The algorithm first takes higher priority constraints into account, and then iteratively finds the solution that verifies at best the lower priority constraints (and match the higher priority constraints). If priority levels are the same, a barycenter is used between positions that verify the constraints. **A** mixed gradient descent method can also be used to minimize the error on all the constraints of a same priority level.

3.4 Constant volume

A particular constraint is very useful: the constant volume constraint. It allows to model incompressible objects (e.g. the abdomen), To apply our algorithm, we first have to find the constraint function $\Phi_c(X)$, where X is the incompressible object position vector. The constraint equation is : $\Phi_c(X) = V(X) - V^0 = 0$, where $V(X)$ is the volume defined by the mass points of the incompressible object, and V^0 the initial volume. In our model, these mass points define the external contour of the object. Thus, $V(X)$ is the object discretized volume. If $\Phi_c(X) = 0$ is verified at any time step, the volume is kept constant: the object is exactly incompressible. Another global constraint can be handled in the same way : the surface constant constraint. Like the volume defined by a set of points, the surface is a function of the positions vector X. The surface gradient vector can easily be deduced and the same method can be applied.
We now successively present the volume computation, the volume conservation method, and a generalization of our method to any other model.

3.4.1 Volume computation

From the divergence formula we find: $V(X) = \frac{1}{3}\sum_k \text{Area}(F_k) \cdot \left(\overrightarrow{OG_k} \cdot \vec{n}_k \right)$, where F_k is the k^{th} face defining the object surface, $\vec{n}_k$ a unit vector normal to F_k, and G_k its isobarycenter. As explained in section 2.1, the

object surface is composed of triangle faces (F_k is a triangle). If P_k^0, P_k^1 and P_k^2 are the three mass points that define F_k, the area is computed as: $\text{Area}(F_k) = \frac{1}{2}\left\|\vec{N}_k\right\| = \frac{1}{2}\left\|\overrightarrow{P_k^0P_k^1}\times\overrightarrow{P_k^0P_k^2}\right\|$, with $\vec{n} = \frac{\vec{N}_k}{\left\|\vec{N}_k\right\|}$.

Finally $V(X) = \frac{1}{6}\sum_k\left(\overrightarrow{OG}_k \cdot \vec{N}_k\right)$.

For each face k, we compute its contributing volume V_k, using the position vector (define in section **2.2):** $X = (x_1, \cdots, x_n, y_1, \cdots, y_n, z_1, \cdots, z_n)^t$.

As $\vec{N}_k = \overrightarrow{P_k^0P_k^1}\times\overrightarrow{P_k^0P_k^2} = \left(X^tA_k^xX \quad X^tA_k^yX \quad X^tA_k^zX\right)^t$ (sparse matrices A_k^x, A_k^y, A_k^z are the matrices form of the vector product, i.e. $x_{u\times v} = u^tA^xv$). V_k can be expressed as:

$$V_k = \overrightarrow{OG}_k \cdot \vec{N}_k = x_{G_k}\cdot X^tA_k^xX + y_{G_k}\cdot X^tA_k^yX + z_{G_k}\cdot X^tA_k^zX = X^t\left(\Gamma_k^xX^tA_k^x + \Gamma_k^yX^tA_k^y + \Gamma_k^zX^tA_k^z\right)X,$$

where $\Gamma_k^x, \Gamma_k^y, \Gamma_k^z$ are the vectors that yield respectively $x_{G_k}, y_{G_k}, z_{G_k}$ (the coordinates of G_k).

As there is no topological change during the simulation, $\Gamma_k^x, A_k^x, \Gamma_k^y, A_k^y, \Gamma_k^z, A_k^z$ are constant sparse matrices and vectors. Let $\Psi_k(X) = \Gamma_k^xX^tA_k^x + \Gamma_k^yX^tA_k^y + \Gamma_k^zX^tA_k^z$, then the volume can be expressed as:

$$V(X) = \frac{1}{6}\sum_k V_k = \frac{1}{6}\sum_k X^t\Psi_k(X)X = \frac{1}{6}X^t\left(\sum_k\Psi_k(X)\right)X.$$

From the object position vector we derive the inside volume. This means that we can compute the region in the position space where the constraint function $\Phi_c(X) = 0$ is satisfied.

3.4.2 Volume conservation

In order to use the constraint projection method, we need to know $\vec{\nabla}_V(X) = \frac{\partial V(X)}{\partial X}$, the gradient vector of the volume function. From the gradient theorem, we finally find: $\vec{\nabla}_V(X) = 3\left(\sum_k\Psi_k(X)\right)X$. Now, in order to keep the volume constant, the algorithm described in the previous section is used (see figure 3):

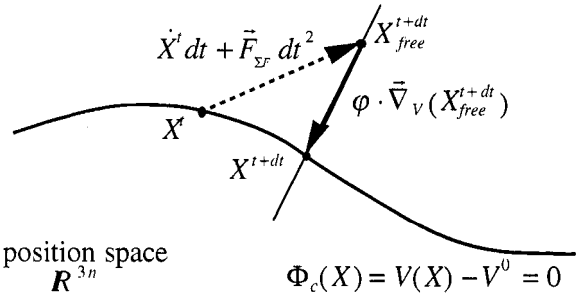


Figure 3: projection on constant volume constraint

- Compute the sum of forces. A vector $\vec{F}_{\Sigma F}$ is used to hold all these sums divided by the corresponding mass of each point.
- Loop (while time coherence is not verified)
  Compute the new position, without constraints,
  Take constant volume constraint into account,
  i.e. solve the system:

$$\begin{cases} X^{t+dt} = X^{t+dt}_{free} + \varphi\cdot\vec{\nabla}_V\left(X^{t+dt}_{free}\right) \\ V(X^{t+dt}) = V^0 \end{cases}$$

- Adjust speed depending on real displacement.

The solution of this system gives the new position that exactly keeps the volume constant (in one iteration). As the matrices and vectors $\Gamma_k^x, A_k^x, \Gamma_k^y, A_k^y, \Gamma_k^z, A_k^z$ are sparse, our algorithm stores only the non-null values. This allows us to optimize computing time by solving a third order equation and directly find the value of φ.

3.4.3 Generalization:

We can also demonstrate that this constraint resolution has a physically-based interpretation. Actually, the projection vector $\vec{\nabla}_V$ is equal (on each vertex) to a vector normal to the object surface. Hence the projection vector $\varphi\cdot\vec{\nabla}_V$ can be compared to a pressure force. As pressure on an object membrane is linked to the object volume, our constraint resolution has a physical validity. Notice that, as $\vec{\nabla}_V(X) = \frac{\partial V(X)}{\partial X}$ (the gradient

vector of the volume function), $\vec{\nabla}_V$ determines the direction of the maximum volume variation: intuitively, it is also the direction defined by the vectors normal to the surface. In fact, our constraint resolution is strictly equivalent to a kind of scaling function applied to the object. Each point is moved along a vector normal to the surface in order to match V^0.
Thus, the presented algorithm would still work with any other model, even with non physically-based modelling (as in FFD, [Rap95]). In order to keep constant the volume of an object, three steps are necessary:
1. compute the normal to the surface on each point,
2. find the correct value of φ, using the normal vectors instead of the volume gradient vectors,
3. move each point along its normal vector multiply by φ.

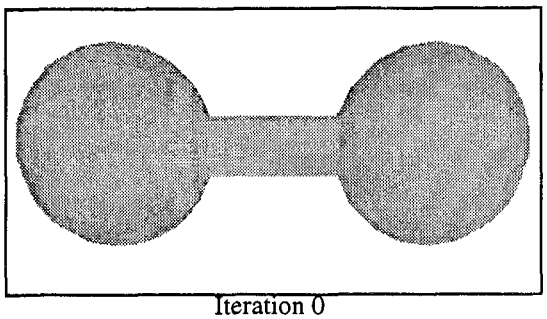
Moreover, we can use our formula to give a compressibility property to the objects. Using point $X_i = X_{free}^{t+dt} + \varphi \cdot \vec{\nabla}_V \left(X_{free}^{t+dt} \right)$ as an ideal point, we can introduce a new force: $\vec{F}_{compressibility} = -k\left(X_{free}^{t+dt} - X_i \right)$.
This force should be no longer processed as a constraint but as a normal (finite) force. Notice that we can apply our constant volume constraint only to few points of an object. This allows for definition of local properties.

4 Conclusion

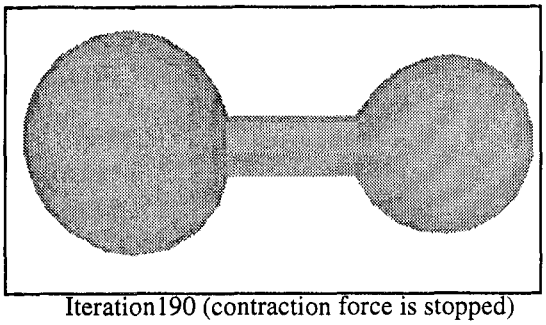
The proposed physically-based model can easily handle natural object properties: elasticity, motor functioning and compressibility (see examples in figures 4 to 6). A simple efficient algorithm taking multiple constraints into account has been described. Local and global constraints can be used. A particular constraint that is very useful for natural modelling can be handled easily: the constant volume constraint.
Future works will include modelling of rigid parts on objects. This could be done considering a set of mass points, computing a global force balance on them, and deducing translation and rotation torques. In this way, the rigid parts of human trunk (spine, pelvis and rib cage) could be modelled. The human respiration system will also use volume control on an object to model the lungs.

References

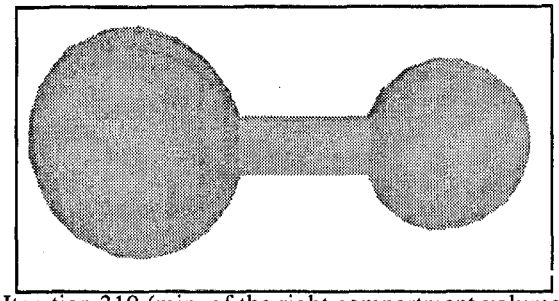
[Aub95] F. Aubert, D. Bechmann, Déformation d'objets a volume constant, Actes des 3ème journées AFIG, 22-24 Nov., 1995.
[Aya96] N. Ayache, P. Cinquin, I. Cohen, L. Cohen, F. Leitner, 0. Monga, Segmentation of complex medical objects: a challenge and a requirement for computer assisted surgery planning and performing, *Computer Integrated Surgery*, MIT Press, 1996.
[Bar88] R. Barzel, A. Barr, A modelling system based on dynamic constraints, *ACM Siggraph'88*, 22(4):179-188, August 1988.
[Che92] D.T. Chen, D. Zeltzer, Pump it up: computer animation of a biomechanically based model of muscle using the finite element method, *ACM* ***Siggraph'92,26(2):89-98,*** July 1992.
[Del93] *Y* Delnondedieu, A. Luciani, C. Cadoz, Physical elementary component for modeling the tensory-motricity: the primary muscle. *4th Eurographics Animation and Simulation Workshop,* Sept. 93.
[Fer94] G.Ferrigno, P. Carnivalli, A. Aliverti, F. Molteni, G. Beulcke, A. Pedotti, Three-dimensional optical analysis of chest wall motion, *Journal of Applied Physiology,* 77(3):1224-1231, 1994.
[Gas94] M.P. Gascuel, J.D. Gascuel, Displacement constraints for interactive modeling and animation of articulated structures, *The Visual Computer,* 10(4):191-204, March 1994.
[Jim93] S. Jimenez, A. Luciani, 0. Raoult, Physical simulation of land vehicules with obstacle avoidance and various terrain interactions. *The Journal of Visualisation and Computer Animation,* 4:79-84, 1993.
[Lac94] J.O. Lachaud, E. Bainville, A discrete adaptive model following topological modifications of volume applied to three-dimensional image segmentation, *4th DGCI Grenoble conference act,* 19-21 Sept., 1994.
[Lam95] A. Lamouret, M.P. Gascuel, J.D. Gascuel, Combining physically-based simulation of colliding objects with trajectory control, *The journal of Visualization and Computer Animation,* 5:1-20, 1995.
[Pla88] J.C. Platt, A.H. Barr, Constraint methods for flexible models, *ACM Siggraph'88,* 22(4):279-288, August 1988.
[Rap95] A. Rappoport, A. Sheffer, M. Bercovier, Volume-preserving free-form solids, *3rd ACM Siggraph synposium on solid modeling (Solid Modeling '95),* Salt Lake City, pp361-372, May 1995.
[Ter87] D. Terzopoulos, J. Platt, A. Barr, K. Fleischer, Elastically deformable models, *ACM Siggraph'87,* 21(4):205-213, July 1987
[Ter89] D. Terzopoulos, A. Barr, D. Zeltzer, A. Witkin, J. Blinn, Physically-based modeling: past, present and future, *ACM Siggraph'89 panel proceedings,* August 1989.
[Van91] C. Van Overveld, An iterative approach to dynamic simulation of 3-D rigid-body motions for real-time interactive computer animation, *The Visual Computer,* 7:29-38, 1991.



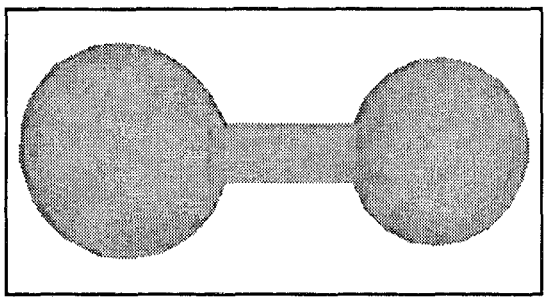
Iteration 0



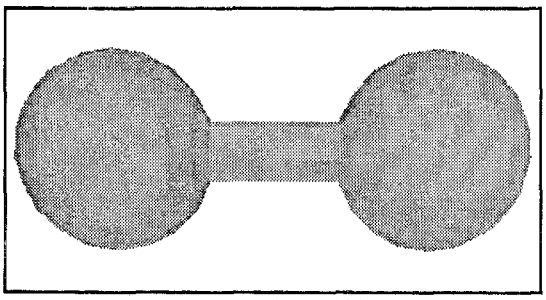
Iteration190 (contraction force is stopped)



Iteration 310 (min. of the right compartment volume)



Iteration 600



Iteration 1400

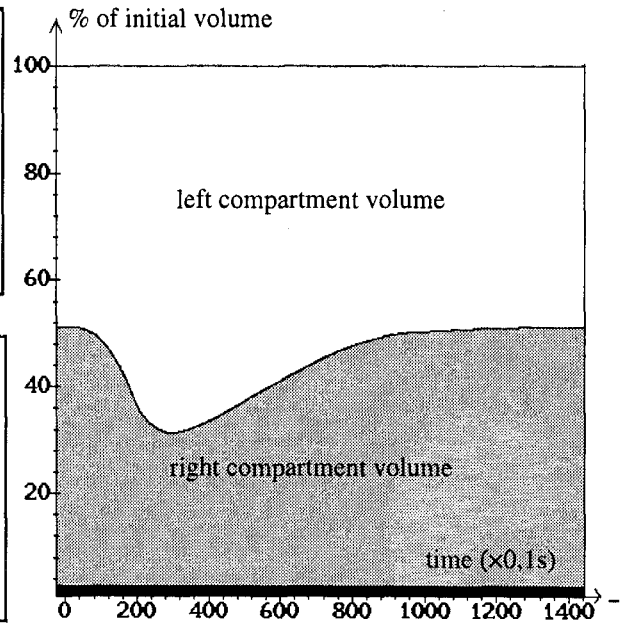
Figure 4: Two compartmented ball

The volume is kept constant during the simulation. We put a muscular contraction force on the right end-ball. The middle compartment points are fixed. As the right end-ball contracts, the left end-ball volume increases proportionally. When the muscle contraction force is stopped (at iteration 190), the volume variation progressively slows down, and the object finally return to its rest shape (due to the shape memory force).

The graph bellow shows the evolution of the three compartment volumes (the middle compartment volume is constant and represented in black).

The total volume is kept constant with at least a 10^{-12} accuracy.

Average computation time is 0.2s by iteration (dt = 0.1s) on a DEC Alpha 3000 workstation. The object presented here has 254 points and 504 facets.



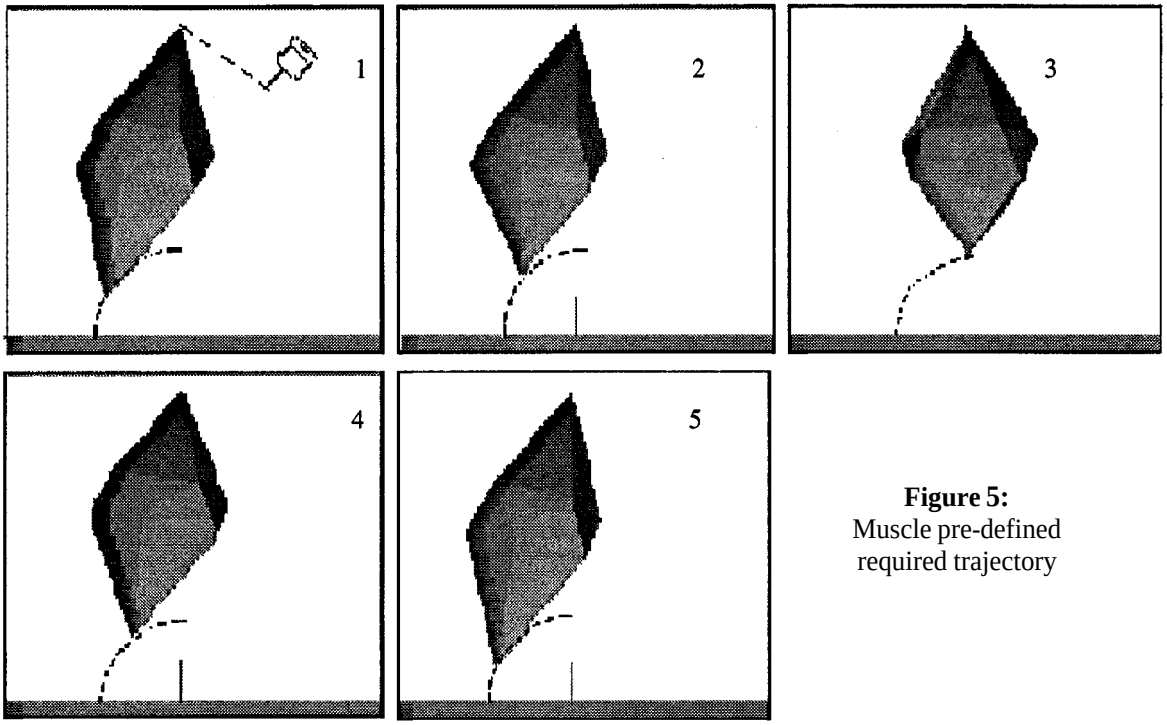
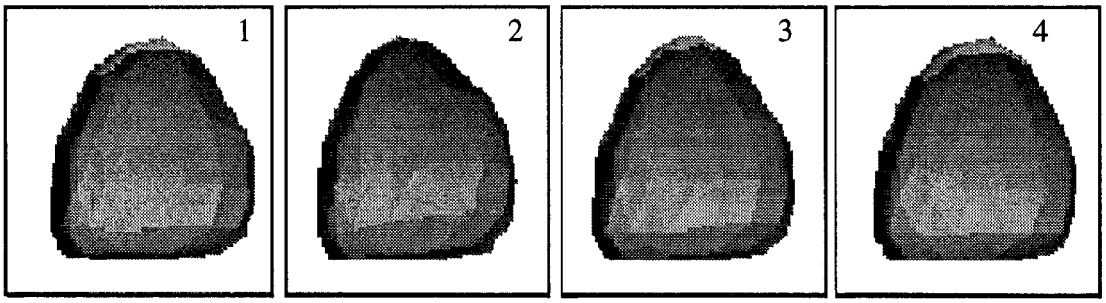


Figure 5:
Muscle pre-defined required trajectory

Two local constraints are applied: the top point of the object is nailed in space and the bottom point has to stay on a circular trajectory. The other points can contract towards their neighbours. The muscle contraction starts at its rest shape (image 1). During the contraction the bottom point moves along a required circular trajectory (image 2 and 3). When the contraction stops, the object returns to its rest shape (image 4 and 5). The volume is kept constant with at least a 10^{-12} accuracy.



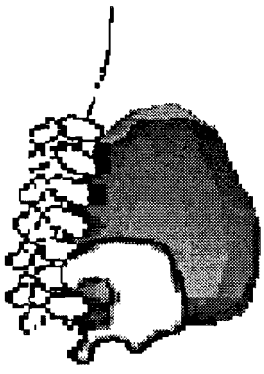


Figure 6: The abdomen-diaphragmset simulation

The top points define the muscular part (modelling the diaphragm), and the back points are nailed (modelling the spine and the pelvis, on the left of the image). As muscle points contract (image 1 and 2), the abdomen slightly moves up (increasing the rib cage volume). When contraction decreases the object comes back to its rest shape (image 3 and 4). Here again, the volume is kept constant with at least a 10^{-12} accuracy.