Robot : Dynamic modeling system for robotics applications

Ammar Joukhadar

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

*Robot*Φ : Dynamic modeling system for robotics applications

Ammar Joukhadar

N 2543

Mai 95

PROGRAMME 4

Robotique,

image

et vision

ISSN 0249-6399

INRIA
RHÔNE-ALPES

*Robot*Φ : Dynamic modeling system for robotics applications

Ammar Joukhadar*

Programme 4 — Robotique, image et vision
Projet SHARP

Rapport de recherche n° 2543 — Mai 95 — 38 pages

Abstract: We describe models and algorithms designed to produce efficient and physically consistent dynamic simulations in the context of Intervention Robotics (i.e., for Robotics tasks involving strong physical interaction constraints and not necessarily rigid objects). These models and algorithms have been implemented within the *Robot*Φ system which can be potentially configured for a large variety of intervention-style tasks such as dextrous manipulations with a robot hand, manipulation of non-rigid objects, tele-programming the motions of an all-terrain vehicle, and some robot assisted surgery tasks (e.g. positioning of an artificial ligament in knee surgery). The approach uses a novel physically based modeling technique to produce dynamic simulations which are both efficient and consistent according to the laws of the Physics. The main advantages over previous works in Robotics and Computer Graphics fields are twofold: the development of a unique framework for simultaneously processing motions, deformations, and physical interactions; and the incorporation of appropriate models and algorithms for obtaining efficient processing times while insuring consistent physical behaviors. These models are based upon three basic principles : the energy-based adaptive time step principle, the inertia-based adaptive discretization principle, and the dynamic surface/surface interaction laws.

Key-words: Physics, dynamics, modeling, motion, deformation, interaction, robotics, adaptive discretization, adaptive time step

(Résumé : tsvp)

*Email : Ammar.Joukhadar@imag.fr

Unité de recherche INRIA Rhône-Alpes
46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1 (France)
Téléphone : (33) 76 57 47 77 – Télécopie : (33) 76 57 47 54

*Robot*Φ : Un système de modélisation physique pour des applications robotiques

Résumé : Plusieurs tâches robotique exigent la prise en compte d'une interaction complexe entre le robot et son environnement. Cette interaction peut modifier largement le comportement du robot (et par conséquent la probabilité du succès de la tache). C'est le cas quand on manipule un objet rigide ou déformable avec une main articulée, ou quand on planifie le mouvement d'un véhicule sur un terrain accidenté.

Les modèles géométriques classiques développés dans les domaines de CAO-Robotique et planification de mouvement ne sont évidement pas adaptés pour étudier ces interactions. En fait, le but des modèles géométriques est de représenter la forme et les propriétés spatiales d'un objet, tandis que les interactions entre les objets dépendent de propriétés physiques comme la masse, la distribution de la masse, la rigidité/élasticité de l'objet, la viscosité de l'environnement, la force de la collision.

Ce rapport présente le système de modélisation physique *Robot*Φ qui permet de représenter simultanément le mouvement, les déformations et les interactions d'un objet avec l'environnement.

Deux approches adaptatif sont développées afin de rendre le système plus efficase :

La discrétisation adaptative: Il permet de trouver automatiquement la représentation minimale d'un objet qui respecte sa forme géométrique, sa matrice d'inertie et sa capacité de déformation.

Le pas de temps adaptatif: Il permet de éviter la divergence numérique , d'estimer l'erreur commise et d'améliorer la vitesse d'exécution d'un facteur variable qui peut atteindre des valeurs très importantes comme 10000.

Mots-clé : Physique, dynamique, modélisation, mouvement, interaction, robotique, discrétisation adaptive, pas de temps adaptatif

Contents

List of Figures

1 Introduction

Powerful sensing, planning, and dynamic simulation tools are the essential components of Intervention Robotics applications such as the maintenance and decommissioning of equipment in hostile environments (nuclear, space, deep sea ...), the inspection of damage in contaminated areas after a nuclear or a chemical accident, or the exploration of new sites (e.g. space exploration). Tasks, in these situations, are extremely complex and involve contact interactions and forces that may strongly influence the behaviour of the robot (and consequently drastically reduce the chances of success of the task). Such situations occur, for instance, when manipulating a rigid or a deformable object using a dextrous hand [2, 1], when moving an all-terrain vehicle on a hilly terrain [5], or when searching for appropriate fixture points for an artificial ligament in order to insure a satisfactory functioning of the knee in robot assisted surgery [15]. On one hand, it is seldom easy for a human operator to guess what will be the robot behaviour when the complexities of robot kinematics, environmental collisions and contact, motions and deformations of interacting objects are combined with the task constraints. On the other hand, it is obviously out of the state of art to include all these considerations in a planner. This is why powerful dynamic simulation tools are required.

Classical geometric models developed in the field of CAD-Robotics, and of motion planning are obviously not adapted to the processing of such interactions. Indeed, the purpose of geometrical models is to represent spatial and shape properties of objects, while complex contact interactions and object behaviours mainly depend on physical properties such as mass, mass distribution, rigidity/elasticity factors, viscosity, collision forces. Dealing with this problem requires making use of appropriate models which have the capability to represent motions, deformations, and object interactions in unified framework consistent with the laws of the Dynamics. This is why we have developed a system (the *Robot*Φ system [3]) which exhibits such characteristics. This system uses a novel physically based modelling technique to produce dynamic simulations which are both efficient and consistent with the laws of the Physics. The main advances over previous work in the Robotics and Computer Graphics fields are twofold: the development of unified framework for simultaneously processing motions, deformations, and physical interactions; and the incorporation of appropriate models and algorithms for obtaining efficient processing time while insuring consistent physical behaviours. The next four sections respectively describe the problem to be solved, the main characteristics of the *Robot*ϕ system, the three basic principles which have been developed to improve the efficiency and the consistency of the system (the *energy-based adaptive time step* principle, and the *inertia-based adaptive discretization* principle), and some experimental results.

2 Related works

Usually, robotics systems combine geometric models and partial physical models[1] in order to study contacts [18, 6, 7], to simulate collision effects [17], to avoid obstacles [19], or to study stability properties[20]. These approaches are generally efficient when solving the considered sub-problems, but constructing a complete dynamic simulation system, requires the combination of three partial physical models : motion, deformation, and contact interaction. Combining existing efficient partial models is not always possible, and even when it is possible it may give a non efficient model. For instance, the efficient motion model "rigid body dynamic" and the deformation model "finite element" [9] can not be automatically combined because they use paradoxical conditions (rigid and deformable). Also, the "finite element" is not an efficient model, because it needs a squared time $O(n^2)$ to calculate deformations.

Several physically based models have been developed in the field of Computer Graphics in order to produce behavioural animation [10, 8, 16, 14]. Among these models, the "Spring based approaches", have been widely used [10, 14]. This model was initially developed in the mechanics domain to study the traction [12] and flexion(Timoshenko-beam model[11]. Figure1) behaviour of objects.

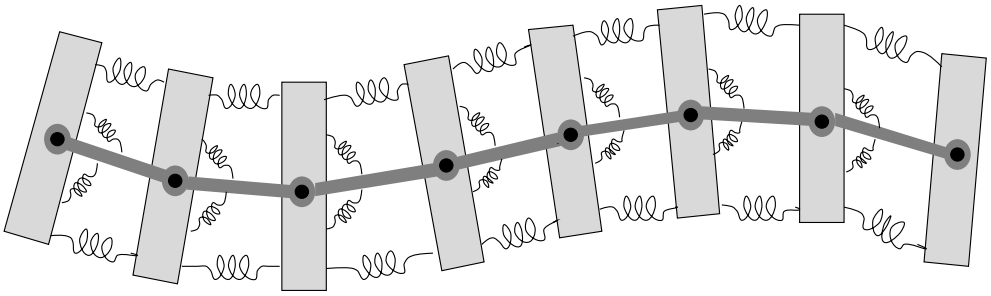

Figure 1: Timoshenko-beam model.

We have already shown in [5] how this type of model could be applied with some appropriate improvements inrobotics applications involving strong physical interaction constraints; we have also shown in [3] that some numerical problems have to be solved when dealing with robotics applications. Indeed, robotics applications differ from graphics applications by the fact that graphic systems try to generate *"looks right" animated pictures* (even if some basic principles of Physics are violated), while robotics systems have to control *"real" mechanical systems* (and consequently, the dynamics laws govering it, have to be satisfied at all time). The reason for some inconsistent behaviours exhibited by the particle based systems comes from the large number of particles of

[1]Partial physical model represents the motion, the deformation, or the interaction but not all the three aspects simultaneously

the models and the associated numerical and algorithmic problems (time and space discretization, error accumulations). Such problems generate numerical divergences which may lead the system to violate some basic principles of Physics (for instance, the mechanical energy of an object increases without any physical reason). This is why appropriate models and algorithms have to be integrated in these approaches in order to be able to consider robotics applications.

The *Robot*Φ system, which uses a "particle model", has been devised in order to meet these requirements [3]. In this paper, we will focus on the modelling and algorithmic problems we have recently solved for making the system more efficient and physically consistent.We will show how we have drastically reduced the required computing time, and how the consistency properties can be guaranteed by controlling the incurred errors and the numerical convergence of the differential equations solver:

- Energy-based adaptive time step : this approach allows us to estimate the incurred error, to minimise the computational time and to avoid numerical divergence.

- Inertia-based adaptive discretization: this approach allows us to minimise the number of particles needed to represent an object. With few number of particles the modeled object becomes hollow, a dynamic surface/surface interaction was developed in order to solve this problem, and to be coherent with existing CAD models, and to be able to make use of fast collision checking methods (as proposed in [6, 7]).

3 Some notions about particle mechanics

In order to understand the *Robot*Φ dynamic modeling system, some notions about the particle mechanics have to be introduced:

- The motion of a point mass m is characterized by the general law of dynamics:

$$\vec{F} = m\,\vec{\gamma} \tag{1}$$

  where, $\vec{F}$ is the sum of the external forces applied on this mass, and γ its acceleration. As the acceleration is the derivative of the velocity $\vec{\gamma} = \frac{\delta V}{\delta t}$, and the velocity is the derivative of the position $\vec{V} = \frac{\delta P}{\delta t}$ one can write:

$$\vec{V} = \vec{\gamma}.t + \vec{V_0} = \frac{\vec{F}}{m}t + \vec{V_0}$$

$$\vec{P} = \frac{1}{2}\vec{\gamma}.t^2 + \vec{V_0}.t + P_0 = \frac{\vec{F}}{2m}.t^2 + \vec{V_0}.t + P_0$$

- The inertial center of an object behaves as a point mass having the same mass of the object and subjecting to the sum of the forces applied on this object.

- For two interacting objects, the force applied on one is equal in magnitude and opposite in direction the force applied on the other, because of the action/reaction principle. So, the sum of the internal forces applied on an isolated physical system is always null. A physical system can be an object or a set of objects.

- The trajectory of the inertial center of a physical system is not affected by the internal forces.

- An object is free if the sum of the external forces applied on it is zero. The velocity of the inertial center of a free object is constant (no energy dissipation), but the vibration of this object has to stop because of the interaction between its particles (which is an internal force).

- The kinetic energy E_k of a mass M having a velocity V is $E_k = \frac{1}{2}MV^2$.

- The variation of the potential energy E_p of a particle P having a position $\vec{P}$ and subjecting to a force $\vec{F}$ is equal to the work of this force $\Delta E_p = -\int_P^{P+\Delta P} \vec{F} \times \delta\vec{P}$.

- The mechanical energy E_m of a particle P, is the sum of its potential energy and its kinetic energy, $E_m = E_k + E_p$. The mechanical energy of a free system is constant $\frac{\delta E_m}{\delta t} = 0$.

4 The dynamic modeling system *Robot*Φ

The *Robot*Φdynamic system is based on the damper/spring model [12, 14, 10], or particle model. In this approach an object is basically modeled as a conglomerate of particles connected by appropriate spring/damper connectors which play the role of giving the object a certain form. In the sequel, we show how such an approach has been adapted to solve some robotic problems.

4.1 Basic constructions

A physical representation of an object is given by a set of particles(Figure 2). These particles have no geometrical dimension but they have masses m_s which verify some physical properties of the real object such as the total mass, the inertia center, and the inertia matrix.

They are connected by three types of spring/damper connectors in order to fix all their degrees of freedom:

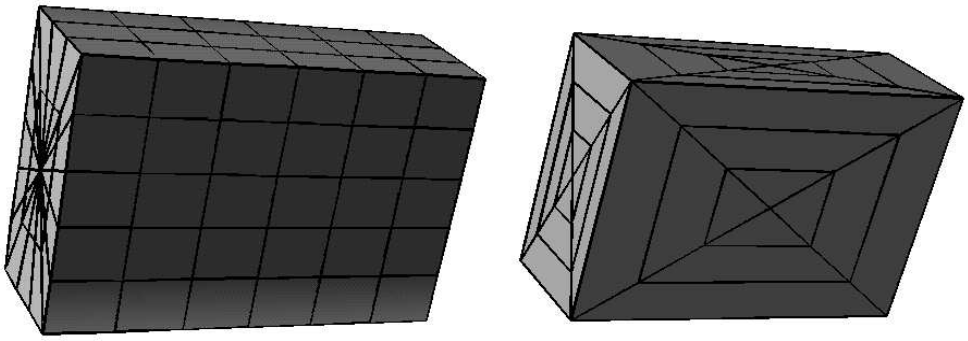

Figure 2: A physical model of an object can be obtained be discretizing it into a number of particles (the intersection points) related by certain connectors.

The ***linear spring/damper connector*** (LS) connects two particles (a and b) in order to fix the distance between them (figure 3.a). The force generated by this connector is expressed by the following equation:

$$\vec{F_a} = (-\lambda \Delta p - \mu \dot{p})\vec{k_a}, \tag{2}$$

$$\dot{p} = (\dot{\vec{a}} - \dot{\vec{b}}).\vec{k_a}$$

where λ is the stiffness of the spring, Δp the variation of its length (it depends on its free length p_0), μ the damping factor of the damper, $\dot{p}$ the relative speed of the particle a compared to b, and $\vec{k_a}$ is the unit vector along the two particles a and b. The term $-\mu\dot{p}$ is a damper used to prevent the oscillation of the system near the equilibrium position of the pair of particles. Such a connector constrains one degree of freedom for each particle. Consequently it is not adapted to model a more complex structures such as revolute and prismatic joints[2].

The ***torsion spring/damper connector*** (TS) has been developed to associate an angular constraint to a set of three particles (a, b and c). In TS, the forces which are generated onto the non-central particles can be defined by the following expression:

$$\vec{F_b} = (-\lambda \Delta \theta - \mu \dot{\theta})\vec{k_b}$$

$$\dot{\theta} = \frac{1}{sin\theta |\vec{ca}||\vec{cb}|}\left(\frac{|\vec{ca}|}{|\vec{cb}|} cos\theta(\vec{cb}.\dot{\vec{cb}}) + \frac{|\vec{cb}|}{|\vec{ca}|} cos\theta(\vec{ca}.\dot{\vec{ca}}) - \vec{ca}.\dot{\vec{cb}} - \vec{cb}.\dot{\vec{ca}}\right)$$

[2]Such properties may be represented by appropriately combining several LS connectors (see for instance [14]), but this necessitates the introduction of additional fictitious point masses and connectors in the model. A consequence of this approach is to artificially modify the mass distribution and to increase the complexity of the model.

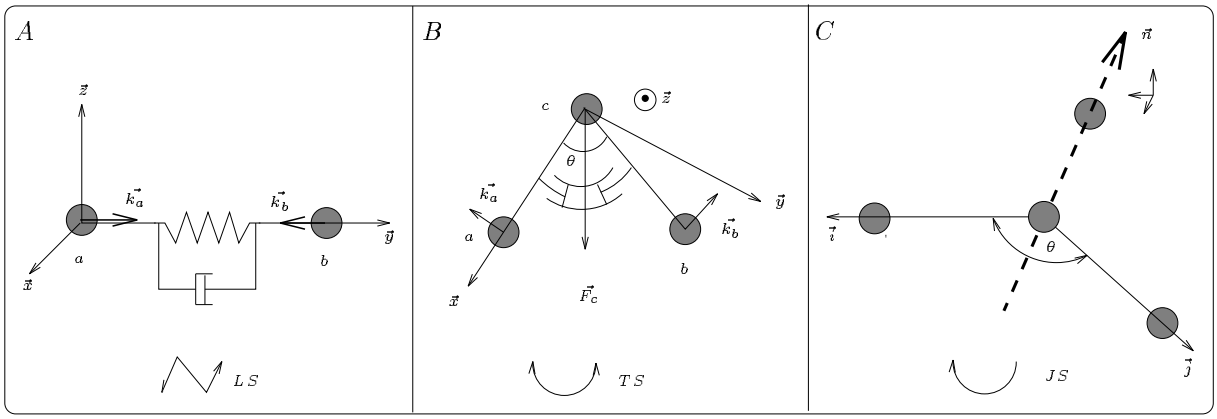

Figure 3: LS, TS and JS connectors.

where $\Delta\theta$ is the variation of the angle formed by the three particles (it depends on its initial angular value θ_0), $\dot{\theta}$ the angular speed, $\vec{ca} = \vec{a} - \vec{b}$, $\dot{\vec{ca}} = \dot{\vec{a}} - \dot{\vec{b}}$ is the speed of the vector $\vec{ca}$, and $\vec{k_b}$ is the unit vector normal to the direction defined by the involved pair of particles, figure 3.b illustrates. Because of the action/reaction principle, the force $\vec{F_c}$ applied to the central particle is equal to the negative sum of $\vec{F_b}$ and $\vec{F_a}$. This connector constrains three degrees of freedom (one degree for each connected particle). But when the equilibrium angle θ_0 of this connector is equal to 180°, this connector constrains two degree of freedom for each particle. This property is very useful in representing stiff bars or prismatic joints (fig. 4). In the general case this connector is very useful in representing the morphological modifications of an object. The disadvantage of this connector is that the two non-central particles are symmetric, so there is no difference between the angle θ and the angle $\theta + \pi$, which may cause a problem when a revolute joint moves more than π.

The ***joint spring/damper connector*** (JS) is developed in order to avoid the symmetry of the TS relation and consequently to be able to represent a revolute joint (fig. 4). The JS connector (fig. 3.a) is similar to a TS connector provided by a fourth particle which allows the definition of a positive rotation direction. The forces which are generated onto the non central particles can be defined by the expression:

$$\vec{F_a} = (-\lambda(\theta - \theta_0) - \mu\dot{\theta})\vec{k_a}$$

$$\dot{\theta} = \frac{1}{sin\theta|\vec{ca}||\vec{cb}|}(\frac{|\vec{ca}|}{|\vec{cb}|}cos\theta(\vec{cb}.\dot{\vec{cb}}) + \frac{|\vec{cb}|}{|\vec{ca}|}cos\theta(\vec{ca}.\dot{\vec{ca}}) - \vec{ca}.\dot{\vec{cb}} - \vec{cb}.\dot{\vec{ca}})$$

where,

$$\theta = \begin{cases} acos(2\pi - \vec{i} \times \vec{j}) & if\ (\vec{i} \wedge \vec{j}) \times \vec{n} < 0 \\ acos(\vec{i} \times \vec{j}) & else \end{cases} \qquad \vec{k_a} = \vec{n} \wedge \vec{j}$$

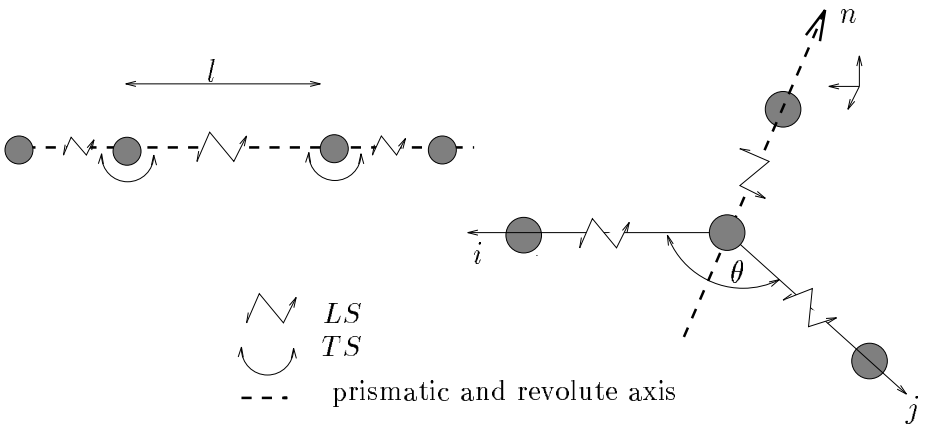

Figure 4: A prismatic joint and a revolute joint.

The physical behaviour of a connector depends on the values of λ and μ. Figure 5 shows the effect of λ and μ on the relative position of two connected particles.

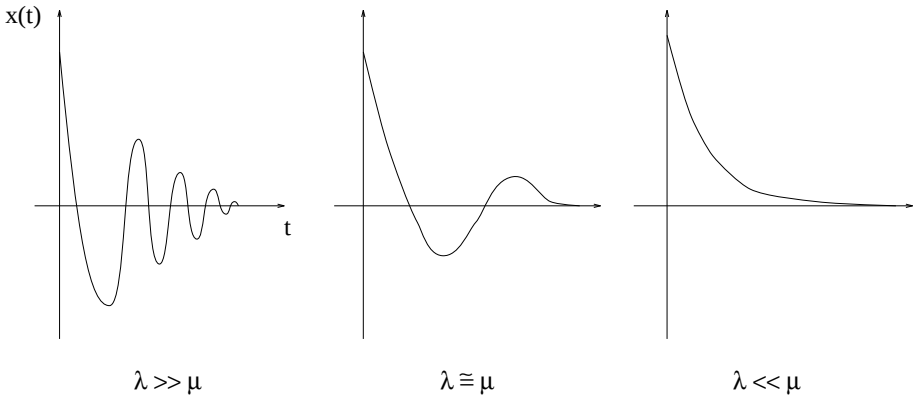

Figure 5: The effect of λ and μ on the motion of a particle.

4.2 Object modeling

4.2.1 Description

To model an object, we decompose it into a set of particles, and we connect these particles appropriately in order to obtain a complete structure (i.e. all the

degrees of freedom of the particles are constrained). Figure 8 shows a possible representation for convex polygon.

The obtained object has no geometrical dimension, to give it this dimension one has to relate the neighbourhood particles by implicit surfaces (figure 6). The role of these surfaces is not only to give the object the geometrical dimension but they play a very important role when processing the interaction between two objects (§ 4.4).

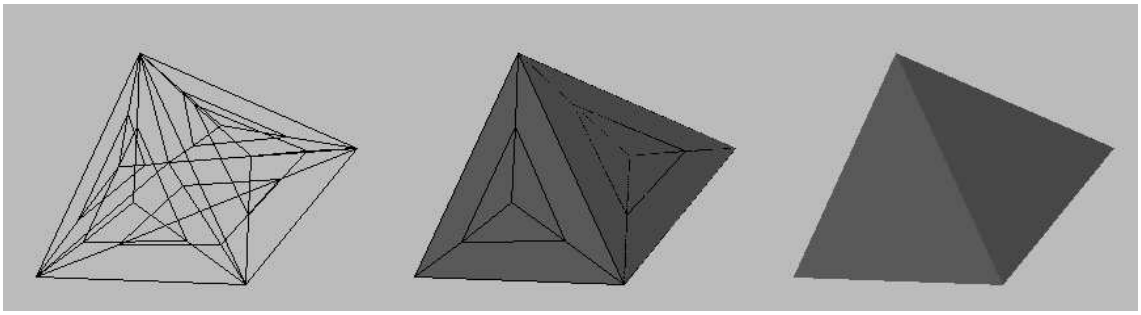

Figure 6: This pyramid is initially represented by a small number of particles. It becomes hollow. One relates these particles by facets to give this object a third dimension. For the visualisation we do not need to see the edges and the vertexes.

An efficient automatic method to construct the physical model of an object has to respect the following constraints:

- The mass of each particle has to be determined such that the modeled object has the same inertia-matrix and the same inertia center as the real one.
- Position of these particles has to be chosen so that the geometrical form of the modeled object coincide with the geometrical form of the modeled object after adding surfaces between these particles.
- The positions of the particles have to be parallel to the principal axis of the object because this minimises the incurred error in behaviour.
- Deformable objects (this is the case when λ is small) need a large number of particles, but rigid ones (λ is large) need relatively few particles.

4.2.2 Object discretization

It is clear that uniform discretization does not satisfy the above-mentioned criteria, because it needs a large number of particles to respect the geometrical form of an object and its inertia center (figure 7).

So an adaptive discretization which minimises the number of particles and respects the symmetry of the object, its inertia matrix, and its inertia center is proposed. One distinguishes three cases:

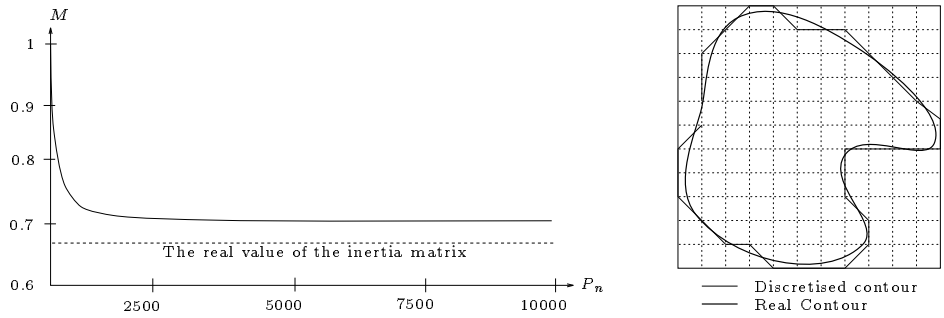

Figure 7: Regular discretization needs a large number of particles to give a good approximation of the real geometrical model. Even when the object has a simple geometrical form, one needs a large number of particles to respect the inertia matrix value. The above figure shows the variation of the inertia matrix of the discretized object as function of the particle number.

Polygon discretization: To discretize a facet one begins by discretizing its contour. If this facet has four edges, then we project two edges on the two others. If this facet has more than four edges, one extrapolates the lines relating each contour point with the inertial center of the facet (figure 8).

Polyhedral primitives: To discretize a polyhedral object, one begins by discretizing its facets, then one extrapolates the lines relating each surface point with the inertia center of the object. This is the same principle of the Russian doll (figure 9).

Sweeping primitives: In this case the object is defined by two facets. First, one discretizes each facet, then one projects one of these facets on the other by adding a set of intermediate facets (figure 10).

This method of discretization respects the symmetrical properties of the object, which is important in order to have a correct behaviour. Also, it allows us to minimise the number of particles needed to represent an object. When the object is rigid, one needs a very small number of particles to represent it, and one comes near the complexity of a geometrical model. When the object is deformable, one needs a finer discretization to insure that the inertia matrix is preserved during the deformation (§ 4.2.3).

4.2.3 Inertia-matrix conservation

Conserving[3] the inertia matrix of the object consists in finding the mass value m_i of each particle i in order to have the same inertia matrix of the initial object. These masses can be obtained in solving the following equation system

[3]This work has been performed during the *DEA* stage of Manuel Da Vinha, and it has been supervised by Ammar Joukhadar and Christian Laugier. This *DEA* report will be published in details in June 1995.

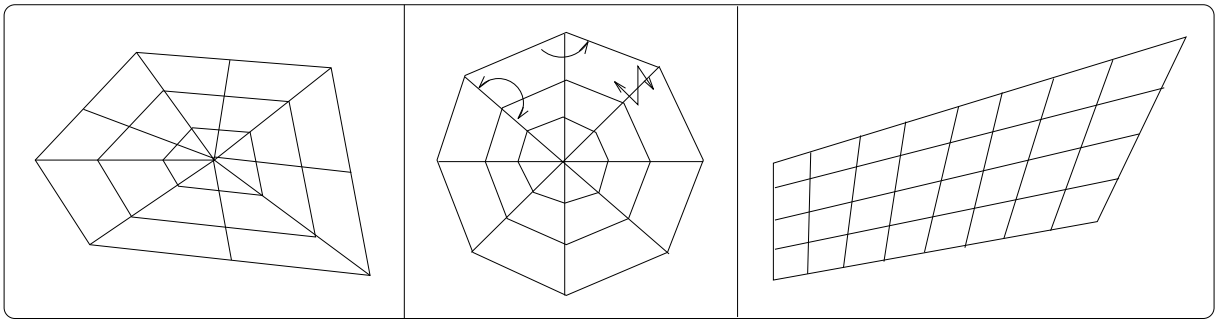

Figure 8: Discretization of a convex polygon

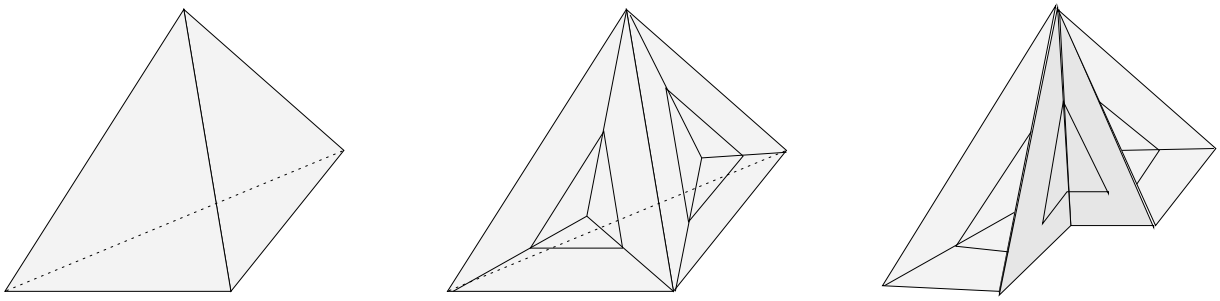

Figure 9: Discretization of a convex polyhedron

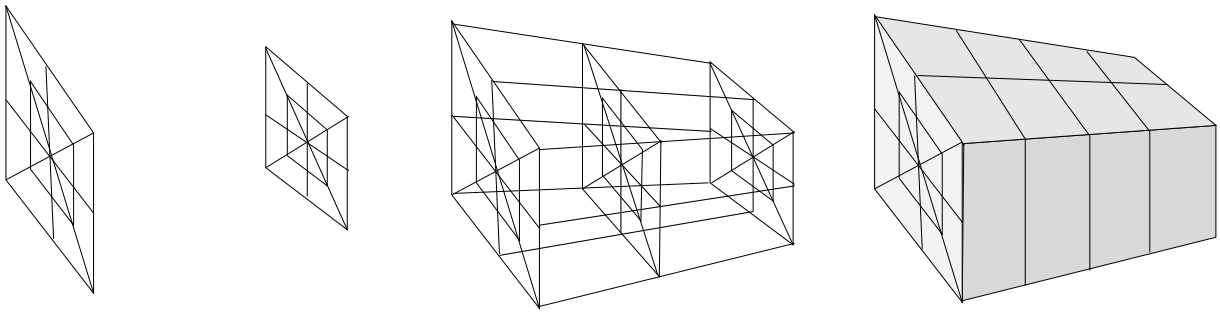

Figure 10: Discretization of a palliating object

(at the left the inertia matrix of the real object and at the right the inertia matrix of the discretized object):

$$\begin{pmatrix} A & -F & -E \\ -F & B & -D \\ -E & -D & C \end{pmatrix} = \begin{pmatrix} \sum ({y_i}^2 + {z_i}^2) * m_i & -\sum x * y * m_i & -\sum x * z * m_i \\ -\sum x * y * m_i & \sum ({x_i}^2 + {z_i}^2) * m_i & -\sum y * z * m_i \\ -\sum x * z * m_i & -\sum y * z * m_i & \sum ({x_i}^2 + {y_i}^2) * m_i \end{pmatrix}$$

$$\sum_{i=1}^{n} m_i * (P_i - P_c) = 0$$

where P_c is the position of the inertial center of m_i.

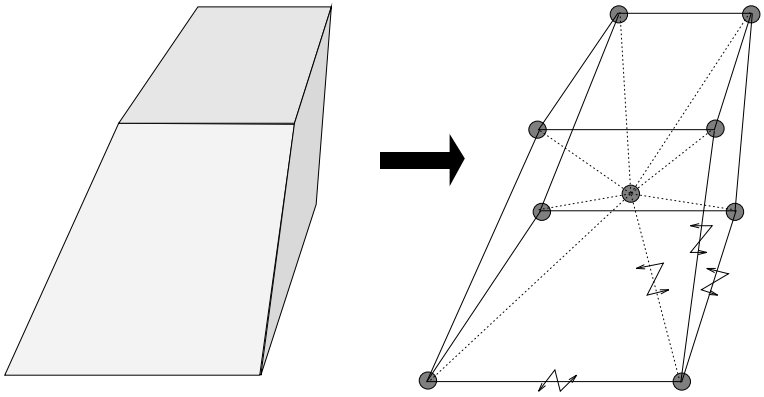

Figure 11: The adaptive discretization divide the initial object into a set of sub-objects, each one contains 8 particles.

It is obvious that this equation system has an infinite number of solutions, and when the object deforms, a given solution can be no more correct. The above-mentioned discretization approach divides the object into a set of sub-object. Each sub-object constitutes 8 particles (figure 11). Within these solutions, one chooses the one which conserve the inertia matrix of each sub-object. Solving the above-mentioned system for these 8 particles and their inertia center can give sometimes negative values of m_i. In order to avoid that, one divides each sub-object into a set of tetrahedra. Solving the above system for a tetrahedron gives an unique solution, because the inertial center of an tetrahedron is the same as its geometrical center. So one can replace a tetrahedron by 5 particles (figure 12) whose masses are:

$$m_i = \begin{cases} \frac{1}{20} M & i \text{ is a particle which is put on a vertex} \\ \frac{4}{5} * M & i \text{ is a particle which is put on the inertia center} \end{cases}$$

So to respect the inertia matrix of a hexahydron we can divide it into tetrahedra; we have developed several methods to do it:

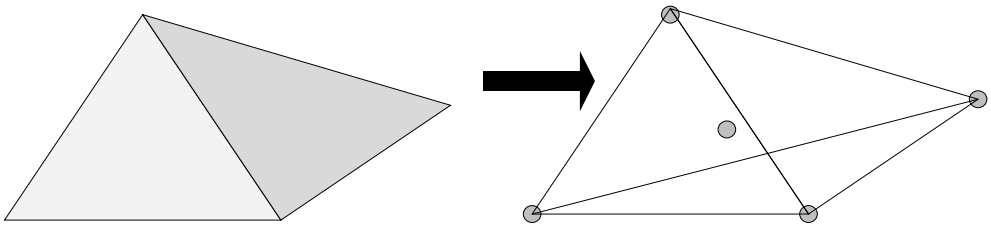

Figure 12:

Symmetric exact method: In this case, one begins by adding a particle in the inertial center of the hexahydron and a particle in the inertial center of each facet. Then each facet can be divided into 4 triangles, each triangle forms a tetrahedron with the inertial center of the hexahydron. One replaces each tetrahedron by 5 particls to obtain a symmetric and exact representation of the hexahedron. In this case, one needs 39 particles to represent a hexahedron.

Non-symmetric exact method: In this case, one divides the hexahydron into 5 tetrahedra and replaces each one by 5 particles. One needs, in this case, 13 particles.

Symmetric approximative method: In this case, one behaves first as in the first method by dividing the hexahydron into 24 tetrahydra, then one eliminates the inertia center of each tetrahedron by dividing its mass between the vertex of this polyhedron, and one eliminates the inertia center of each facet in dividing its mass between the vertex of this facet. The hyxahedron now is represented by 9 particles, and has an approximative inertia matrix value. To ameliorate this precision one can give a ratio of the vertex masses to the inertia center. This ratio can be determined recursively. For 10000 tests, the avarege relative precision was 0.0003. In this case, One represents the hexahydron by 9 particles.

The proposed discretization method has two adaptive aspects:

The number of sub-objects: The difference between the inertia matrix of the real object and the inertia matrix of the modeled one, changes during the deformation of the object. 10000 tests has shown that if one particle from a sub object changes its position of 10% relatively to the sub-object dimension, the inertia matrix relative error increases of 0.1%. One can measure this error and choose the precision of discretization as a function of this error.

The position of sub-objects: The deformation of a sub-object depends on its position in the object, for example, the center of a table deforms more than their edges for the same forces. Also the side facets of this table do

not hold any things and their is no forces on theme. In this case a good choice of the position of the sub-object is near its upper face (figure 13.

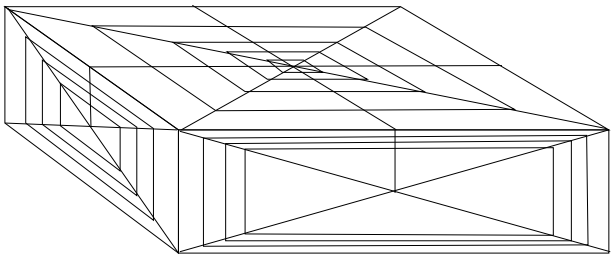

Figure 13: An adaptive discretization of a table: The upper face of the table is discretized into a large number of sub-objects because it deforms. Also these sub-objects are concentrated in the middle which is the weakest point. The side faces of the table are discretized into a fewer sub-objects and these sub-objects are concentrated at the upper face.

4.2.4 Adding connectors

At this stage the object is replaced by a set of point masses which respect its inertia matrix and we have relate these point masses by some connectors in order to obtain a rigide object.

Linear connectors LS have to be added between every two neighbourhood particles (figure 11). These connectors can fix all degrees of freedom and they are sufficient when the object is non-deformable. But LS connectors do not have the same reaction against external forces and when the object is deformable we can not have a realistic deformation using only linear connectors (figure 14). So we add a torsion connector TS between each three collinear particles and a joint connector JS between each three non-collinear successive particles (figure 8). These connectors can correct the reaction of the linear ones and we can obtaine a realistic behaviour (figure 14).

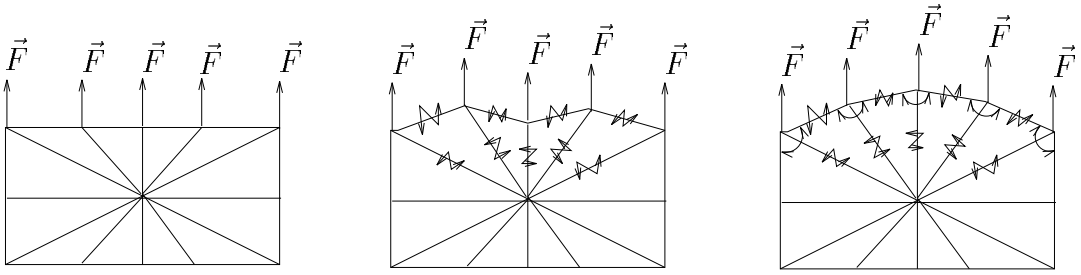

Figure 14: Linear connectors LS have different reactions against external forces. To obtain a realistic deformation we have to add torsion TS and joint JS connectors.

4.3 Motion and deformation

The motion and the deformation of a physical object are the result of the motion of its elementary particles. The position and the velocity of these particles are given by applying (in parallel) the general law of dynamics $\vec{F} = m\,\vec{\gamma}$ on each one, where $\vec{F}$ is the sum of the forces (internal and external), m the mass of the considered particle, and $\vec{\gamma}$ its acceleration. The external force $\vec{F}$ comes from the internal connectors (LS, TS, and JS), from the collision force between objects, and from the control force imposed by the user on the system. Applying the general law of dynamics gives a differential equation system which the general form is:

$$\{m_i \ddot{\vec{P}}_t^i = \vec{F_{ext}}_t^i + f(\vec{P}_t^j, \dot{\vec{P}}_t^j, \vec{P}_t^i, \dot{\vec{P}}_t^i) \setminus \text{i,j are particles}\}$$

where m_i and $\vec{P}_t^i$ are respectively the mass and the position at time t of the particle P_i, $F_{ext}{}_t^i$ is the sum of the external forces applied by the user on P_i, $\dot{\vec{P}}$ is the derivative of the position function $\vec{P}$, and f is a function which givese collision and connector forces. It is obvious that f is neither continuous nor linear.

4.3.1 Solving the system

This system has to be solved numerically because it is virtually impossible to solve it analytically:

- f is a non-continuous and non-linear function.
- User put usually the dynamic system in a control loop, thus it is impossible to predict their control forces $\vec{F_{ext}}_t^i$.

Some methods for solving a differential equation system: Numerical methods are based on one of the following notions[4]:

Finite difference: This approach expresses the derivatives of a function U_t in terms of the function values. Houbolt gives the following implicit[4] solution which is stable irrespective of the value of τ.

$$\dot{U} = \frac{1}{6\tau}(11U_{t+\tau} - 18U_t + 9U_{t-\tau} - 2U_{t-2\tau})$$

$$\ddot{U} = \frac{1}{\Delta t^2}(2U_{t+\tau} - 5U_t + 4U_{t-\tau} - U_{t-2\tau})$$

If we replace the values of $\dot{U}$ and $\ddot{U}$ in the differential equation, we obtain another non linear equation system in which U_t,$U_{t+\tau}$, $\dot{U}_t$, and $\dot{U}_{t+\tau}$ are unknown.

[4] If $P_{t+\tau} = f(P_x)$ *and* $x <= t$ the approach is explicit (incremental), else it is implicit.

Limited development: This approach expresses a function U_t in terms of its derivatives:

$$U_{t+\tau} = \sum_{0}^{n} \frac{\tau^n}{n!} U_t^{(n)} + O(\frac{\tau^{n+1}}{(n+1)!} U_t^{(n+1)})$$

When using this approach to solve the general dynamic law equation $\vec{F} = m\vec{\gamma}$, the data are the external forces which are equivalent to the second derivative, so we have to develop until the second order. Newmark and Wilson have approximated the third derivative of the function $\dot{\vec{\gamma}}$ by $\frac{\gamma_{t+\tau}-\gamma_t}{\tau}$ and they have obtained the following implicit approach which is unconditionally stable if $\alpha \geq \frac{1}{2}; \beta \geq \frac{1}{2}(\alpha + \frac{1}{2})^2$:

$$\vec{V}_{t+\tau} = \vec{V}_t + \tau\,((1-\alpha)\,\vec{\gamma}_t + \alpha\,\vec{\gamma}_{t+\tau})$$

$$\vec{P}_{t+\tau} = \vec{P}_t + \tau\,\vec{V}_t + \frac{\tau^2}{2}((1-\beta)\,\vec{\gamma}_t + \beta\,\vec{\gamma}_{t+\tau})$$

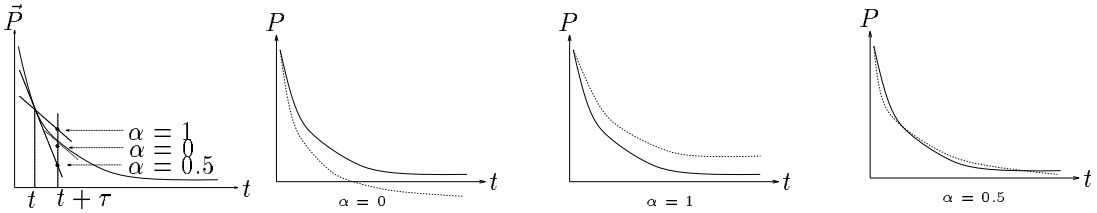

Figure 15: The effect of α on the solution stability: when $\alpha = 0$ the error is always negative, when $\alpha = 1$, the error is alway positive, and when $\alpha = 0.5$, the error changes its sign and the solution is always stable

It is obvious that implicit methods can not be used to solve our differential equation system, because of two principal reasons:

- Our dynamic system is an interactive system which allows users to manipulate objects. The intentions of the user are not known in advance, consequently one can not express the system state at $t + \tau$ in terms of its state at t, which is very important when using an implicit method.
- Even when the user does not intervene, the collision forces are discontinuous functions for which there is no mathematical formulation.

So explicit methods are unique solutions to our differential equation system. Using such a method, the dynamic system state depends only on its past. Consequently, the equations which define our differential system becomes independent, the next position of a given particle $\vec{P_{t+\tau}}$ depends only on its position at t and on the external forces at $t-\tau$. So we have one equation for each particle and the execution time becomes linear.

Our choice was to use the limited development method because it gives a better result comparing to the explicit finite difference method[5]. In approximating the value of the third derivative $\dddot{\vec{\gamma}}$ by $\frac{\gamma_t - \gamma_{t-\tau}}{\tau}$ one obtains:

$$\vec{V}_{t+\tau} = \vec{V}_t + \tau\,((1+\alpha)\,\gamma_t - \alpha\,\gamma_{t-\tau}) \tag{3}$$

$$\vec{P}_{t+\tau} = \vec{P}_t + \tau\,\vec{V}_t + \frac{\tau^2}{2}((1+\beta)\,\gamma_t - \beta\,\gamma_{t-\tau}) \tag{4}$$

The disadvantages of an explicit method applied on a dynamic formulation are:

- It is very sensitive to the chosen time step: if the time step is too large, then the incurred error will produce an inconsistent behaviour as the numerical divergence (figure 16); if the time step is too small, then the execution time is prohibitive.
- There is no way to determine the time step which insure stability, because it depends on many factors like the collision rigidity, the collision nature, the deformability, etc. So we have to repeat the test many times to experimentally determine this time step.
- Even if the time step, which insures stability, is experimentally determined, one can not estimate the incurred error.

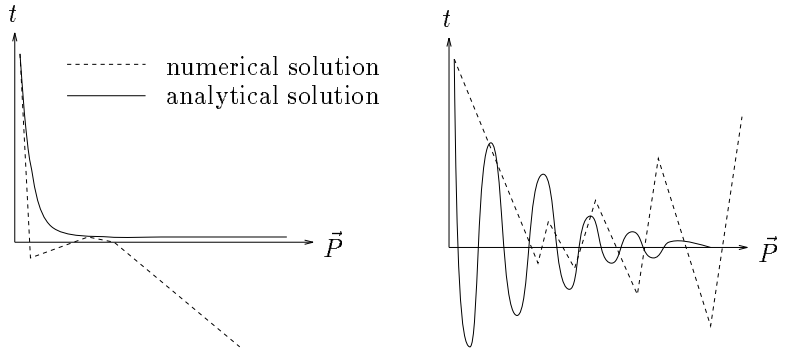

Figure 16: Without a suitable time step the numerical solution can diverge

[5]We have compared the two methods in applying them on a minimal physical object using Maple, which is a computer algebra system, with powerful symbolic, numeric and graphics routines

Adaptive time step and error estimation: Given the function derivatives, the limited development approach gives the following solution:

$$\vec{V}_{t+\tau} = \vec{V}_t + \tau\ \gamma_t + O(\frac{\tau^2}{2}\dot{\gamma}_t) \tag{5}$$

$$\vec{P}_{t+\tau} = \vec{P}_t + \tau\ \vec{V}_t + \frac{\tau^2}{2}\ \gamma_t + O(\frac{\tau^3}{6}\dot{\gamma}_t) \tag{6}$$

One notes that the incurred errors in the values of $\vec{V}_{t+\tau}$ and $\vec{P}_{t+\tau}$ depend on two factors:

The time step τ: when τ decreases, the incurred error decreases.

The value of $\dot{\gamma}$: if the value of $\dot{\gamma}$ decreases, the incurred error decreases too.

It is clear that we can not control the value of $\dot{\gamma}$, the only solution to decrease the error value is to decrease the time step value. Consequently, decreasing the incurred error implies an increase in the computational time, which is not practical.

An optimal solution consists in using an adaptive time step τ (figure 17) which the value, increases when the value of $\dot{\gamma}$ decreases, and decreases otherwise in order to keep a constant value of the error and to go as fast as possible.

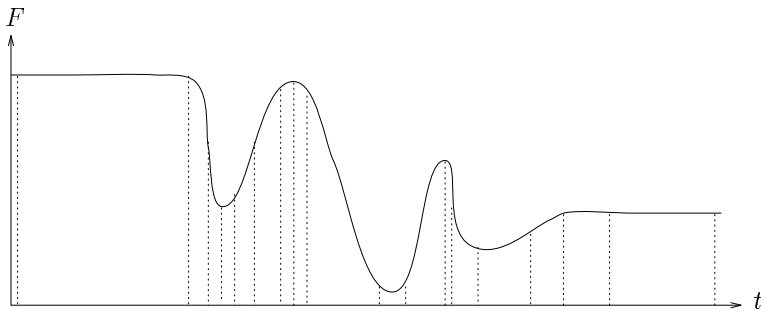

Figure 17: A good solution to avoid divergence and to accelerate the computational time is to use an adaptive time step τ which increases when the function has a small frequency and decreases else.

Normally the value of $\dot{\gamma}$ is not known, so we need another constraint to detect and estimate errors. Equation 5 represents the position $\vec{P}$ and the velocity $\vec{V}$ of a physical object. Thus the obtained solution has to verify the energy conservation principle: The mechanical energy E_m, which is the sum of the potential energy E_p and the kinetic energy E_k, is constant $\Delta E_m = 0$. It depends on $\vec{P}$ and $\vec{V}$:

$$\Delta E_m = E_p + E_k = -\int_P^{P+\Delta P} \vec{F}.\delta\vec{P} + \frac{1}{2}m\Delta V^2$$

This constraint $\Delta E_m = 0$ can practically be expressed by $|\Delta E_m| < \epsilon$. one then detects the value of $|\Delta E_m|$ at each time step and chooses the greatest time step which satisfies $|\Delta E_m| < \epsilon$.

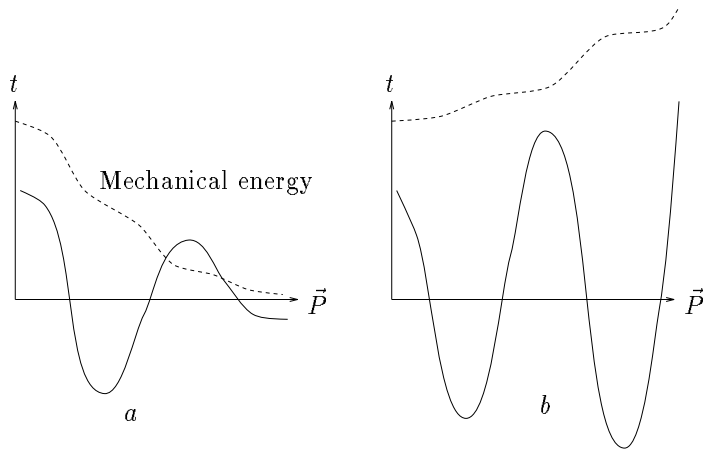

Figure 18: When the mechanical energy decrease the solution becomes stable (a), else it diverges (b).

In fact, the stability of the solution can be verified using the constraint $\Delta E_m < 0$, but the correctness of this solution needs to verify the constraint $|\Delta E_m| < \epsilon$.

The algorithm which determines the greatest value of the time step which makes the error less than ϵ is the following:

Let $\vec{P_t}$ be the actual position of the object.
Let $\vec{V_t}$ be its velocity.
Let E be its total energy.
LOOP
  Calculate the force F_t acting on the object.
  Calculate $\vec{P_{t+\tau}}$ and $\vec{V_{t+\tau}}$ using (eq.5).
  Calculate ΔE.
  IF $|\Delta E| > \epsilon$ THEN
    LOOP
      $\tau = \frac{\tau}{2}$
      Calculate $\vec{P_{t+\tau}}$ and $\vec{V_{t+\tau}}$ using (eq.5).
      Calculate ΔE
    UNTIL $|\Delta E| < \epsilon$
  ELSE
    $\tau = \frac{3}{2}\tau$
UNTIL ∞

The figure 20 shows the evolution of τ when a ball falls down on the earth.

Let T_a be the required time to achieve an iteration using an adaptive time step, I_a the number of performed iterations to achieve the simulation using an adaptive time step, T_c the required time to achieve an iteration using a constant time step, I_c the number of performed iterations to achieve the simulation using a constant time step, τ the current time step, τ_g the time step which verifies the desired constraint, and $I_{lookfor}$ the iteration number needed to reach τ_g. The characteristics of this algorithm are:

- As one divides τ by 2 at each iteration, $I_{lookfor}$ has a very small value:

$$I_{lookfor} = \log_2 \tau - \log_2 \tau_g$$

- Because of the continuity of the motion, τ_g and τ are very close, so $I_{lookfor} = 1$ when $\tau_g < \tau$ and $I_{lookfor} = 0$ when $\tau < \tau_g$:

$$T_a = 1.5 * T_c$$

- The gain in computational time during t second is proportional to the relationship between the greatest frequency of the function $\vec{P_t}$ and the average frequency of this function.

$$Gain_t = \frac{I_a}{I_c} = \frac{\text{The greatest frequency}}{\text{The average frequency}} > 1$$

  We have experimentally found that the ratio between the greatest time step and the average time step obtained with this method is generally very significant (for instance, it can reach a large value such as a ratio of 10000 for a stiff collision).

- We do not use the same factor when increasing (one multiplies by 1.5) and when decreasing (one divides by 2) the time step in order to not get always the same values and to converge to the ideal value if it exists.

- To estimate the velocity error at each iteration, one can assume that in the worst case, the energy error ϵ_e will be transformed totally to kinetic energy and one can write:

$$\Delta E_k = \frac{1}{2} m \Delta V^2 \leftrightarrow \varepsilon_v = \sqrt{\frac{2\varepsilon_e}{m}}$$

- To estimate the position error ϵ_p one can write:

$$P = V * t \leftrightarrow \varepsilon_p = \varepsilon_v * \tau$$

- Experiments show that the average incurred error in position during x seconds is proportional to the used ϵ (figure 19).

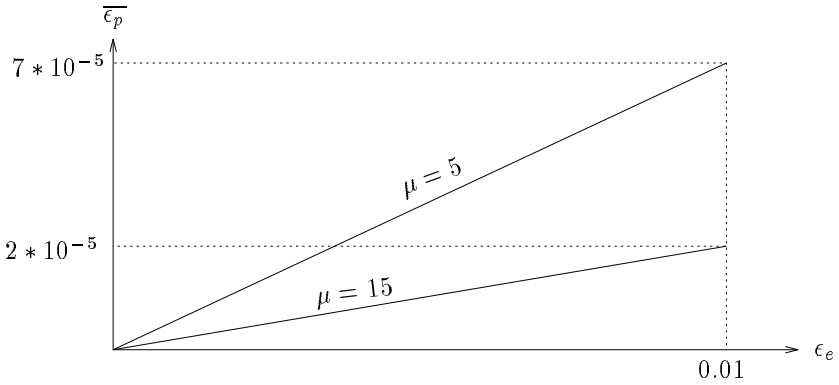

Figure 19: The average value of the position error ϵ_p as a function of the energy error ϵ_e during 10 second and for two different values of μ.

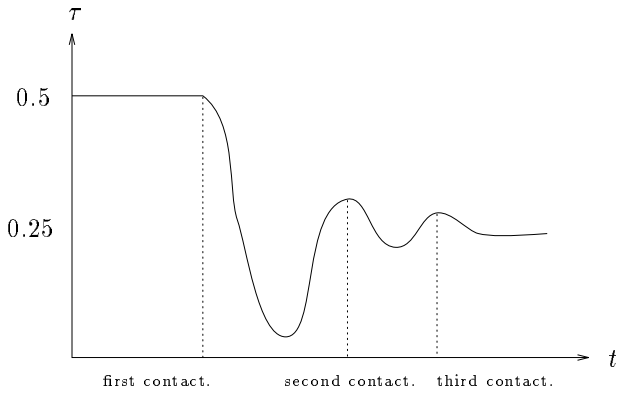

Figure 20: The evolution of τ when a ball falls down on the earth. Note that τ decreases when the ball hits the earth and increases when it leaves the earth. Observe also that the decrement value depend on the force of the collision which is greater when the ball is faster.

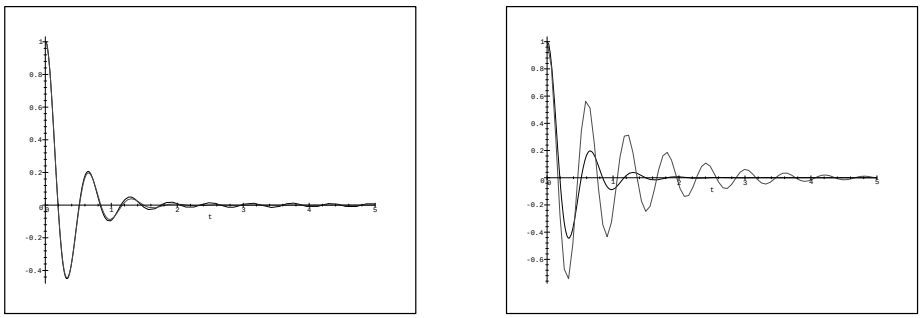

Figure 21: The variation of the relative position of two particles as a function of the time. At the left one uses an adaptive time step, the incurred error is 0.00006. At the right one uses a constant time step which is equal to the average time step obtained by the adaptive approach, the incurred error is 0.003. The used parameters are $\lambda = 100$, $\mu = 5$, $\tau_m = 0.065$, $\epsilon_e = 0.01$.

4.3.2 The adaptive time step and the real time processing

For real time processing we needs to update the environment state at a constant interval of time I, so the gain is limited to $Gain_I$. In practice, the gain is very large when the function contains a very heigh frequency (stiff collision, for example) and small when it is not the case. So the adaptive approach behaves as a filter which makes the execution time quasi- constant during the different intervals of time.

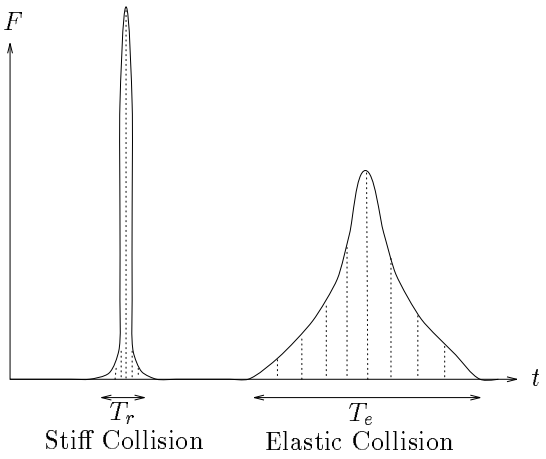

Figure 22: The stiff collision is a phenomenon which needs a very small time step τ_r to be processed, it remains a very small period T_r. The elastic collision remains a larger period of time T_e but it needs a larger time step to be processed τ_e. The computational time needed to process each phenomenon is almost the same because $\frac{T_r}{\tau_r} \simeq \frac{T_e}{\tau_e}$

The real time constraint needs that one is limited by the smallest computational time, one will not loose very much because the computational time is almost the same during these intervals. The efficiency of this filter is proportional to the width of the time interval (figure 23).

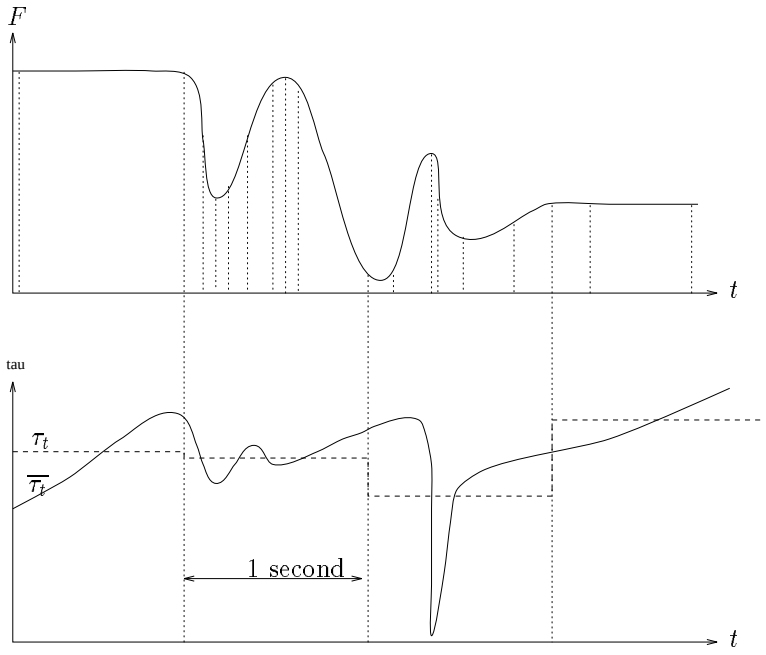

Figure 23: F is a function of the time. τ_t is the time step needed at each iteration to insure stability. $\overline{\tau_t}$ is the average time step during each interval of time. Note that this average changes slowly.

4.4 Interactions

When objects move in a real environment they may interact with each other. The *Robot*Φ system distinguishes between three types of interactions: collision, friction and external viscosity.

Collisions: When two objects collide with each other, they locally deform (figure 24). Consequently, they store potential energy, which will be transformed into kinetic energy when the objects return to their initial forms. In practice, this deformation is detected by measuring the inter-penetration between the two objects.

When two points from two objects collide, the interpenetration is simply equal to the distance d between them, so the collision force can be expressed as:

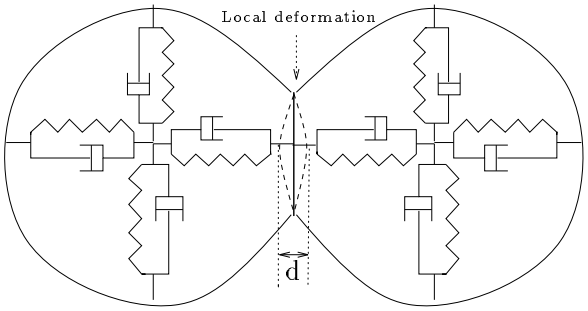

Figure 24: The collision force is the result of a local deformation at the contact area. It is proportional to d.

$$\vec{F_c} = \begin{cases} (-\lambda d - \mu \dot{d})\vec{k} & \text{If } d < 0 \\ \vec{0} & \text{Else} \end{cases} \tag{7}$$

where λ is the rigidity factor of the collision, μ is a damping factor (which represents the dissipation of the energy), d the distance between these two points, and $\vec{k}$ the unit vector along the two points. The distance d is considered negative if each point penetrates the other object.

Practically, it is very expensive to discretize the object surface into a large number of points in order to evaluate the collision force which is the case in [14].

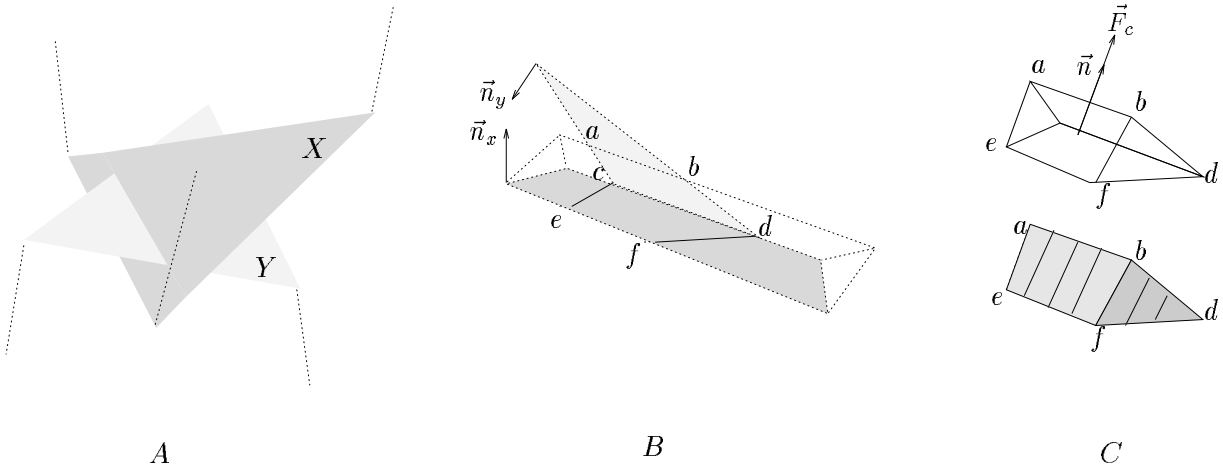

Figure 25: The surface/surface collision.

Our solution is to decompose the object surface into triangular facets. The collision between two facets is considered as an infinite number of point/point

collision. To calculate this collision force, when two triangles X and Y intersect (figure 25.A), one proceeds as follow:

- Finding the part of Y which is inside the object containig X and the part of X which is inside the object containig Y (figure 25.B).
- Projecting each triangle on the other in the direction of $\vec{n}$, where $\vec{n}$ is the average of the normals on each facet.
- The collision force is proportional to the sum of the distances between one point on X and the correspondant point on Y. So the value of the collision force is proportional to the volume of the shape (a, b, c, d, e, f) determined by these two projections $F_c = -\lambda * V_{(a,b,c,d,e,f)}$ (figure 25.C). This force has to be applied in the inertia center of this shape P_c.
- Only particles have masses and can be subjected to external forces, so the collision force has to be distributed between the closest particles which are the vertexes of the triangle in contact (P_1, P_2, P_3). One has to apply a force $\vec{F_i}$ on each particle P_i that the value and the moment of $\vec{F_c}$ remain respected:

$$\vec{F_c} = \vec{F_1} + \vec{F_2} + \vec{F_3}$$
$$\vec{F_c}\vec{P_c} = \vec{F_1}\vec{P_1} + \vec{F_2}\vec{P_2} + \vec{F_3}\vec{P_3}$$

  When a particle i is subjected to a collision force it can lose energy. This energy dissipation can be represented by a force $\vec{F_d}$ which is proportional to its velecity:

$$\vec{F_d^i} = -\mu \vec{V}^i_{relative}$$

  where μ is the energy dissipation factor, and $\vec{V}^i_{relative}$ is the relative velocity of the particle i comparing to the other object.

Friction: When two objects collide, they are subjected to a friction force. Friction is basically a statistical notion. It is the result of a large number of elementary collisions on surfaces in contact (figure 26).

In practice, to represent this force, it should be necessary to decompose the object into a large number of particles, but obviously it is not a realistic solution. In practice, the magnitude of the friction force $|\vec{F_f}|$ can be considered as proportional to the magnitude of the normal force $|\vec{F_n}|$ (Coulomb's law). Then, this force can be defined as follows [6] :

$$\vec{F_f} = \begin{cases} -c\,|\vec{F_n}|\frac{\vec{V}}{|\vec{V}|} & \text{If } \vec{V} \neq \vec{0} \quad (sliding) \\ -s\,|\vec{F_n}|\frac{\vec{F_t}}{|\vec{F_t}|} & \text{If } |\vec{F_t}| > s|\,\vec{F_n}| \quad (intermediate) \\ -\,\vec{F_t} & \text{Else} \quad (sticking) \end{cases} \tag{8}$$

[6] Other approach consist in representing the static friction by a spring connecting the two colliding objects[13].

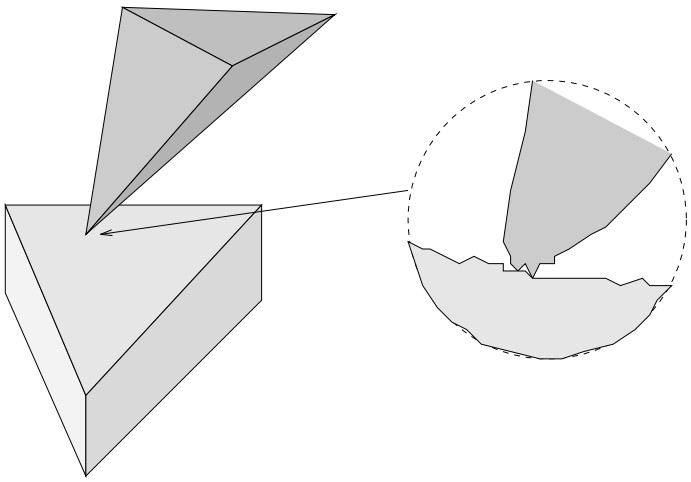

Figure 26: The friction force is the result of a large number of micro collisions at the contact point.

where c is the kinetic friction parameter, s the static friction parameter ($s > c$), $\vec{V}$ the relative speed of the colliding particles, and $\vec{F}_t$ the tangential force applied by one particle on the other.

The resulting force applied by one particle i on another particle j is given by: $\vec{F}_{result} = \sum F_i - \sum F_j$.

If p is a point on the triangle abc then:

$$\vec{P}_p = \alpha\vec{P}_a + \beta\vec{P}_b + \gamma\vec{P}_c$$

abc are subjected to a projective transformation, so α, β, and γ are constants. If one derives, one obtains:

$$\vec{V}_p = \alpha\vec{V}_a + \beta\vec{V}_b + \gamma\vec{V}_c$$

Viscosity: When an object is moving, it interacts with the environment (air, water, vacuum). This collision has an important effect on the motion of the robot and on the interaction between the robot and the manipulated object. For example, the rotation of a falling object is the result of its collision with the air. Representing this collision force (called viscosity), requires to associate a physical representation of the environment. But a representation of fluid involving a very fine decomposition is not applicable in practice. This is why we make use of a macroscopic representation of the resulting phenomena by considering that the generated force $\vec{F}_v$ is proportional to the speed of the particle[7]: $\vec{F}_v = -k\vec{V}$, where k is the viscosity factor and $\vec{V}$ the velocity. For an

[7]This approximation is true only for an object moving in the air with a small velocity. Which is the case for actual robots (for the liquids this force has the form $\vec{F}_v = -k\ \vec{V}^2$)

infinitesimal surface δs : $\vec{F}_v^s = -k\vec{V}_s$. For the complete surface S

$$\vec{F}_v = -k \int_S \vec{V}_s \delta s = -k \int_S \frac{\delta \vec{P}_s}{\delta t} \delta s = -k \frac{\delta}{\delta t} \int_S \vec{P}_s \delta s$$

$$\vec{F}_v = -k\ S \frac{\delta \vec{P}_{Inertia-Center}}{\delta t} = -k\ S\ \vec{V}_{Inertia-Center}$$

Obviously, only those particles which are colliding with the oncoming fluid are subjected to this force. The viscosity force is maximum when $\vec{V} \times \vec{n} = |\vec{V}|$ and it is zero when $\vec{V} \times \vec{n} < 0$, where $\vec{n}$ is the external normal vector to the object surface at the particle location, So,

$$\vec{F}_v = \begin{cases} -k\ S\ (\vec{V}_{Inertia-Center} \times \vec{n}).\frac{\vec{V}}{|\vec{V}|} & \text{if } \vec{V}_{Inertia-Center} \times \vec{n} > 0 \\ 0 & \text{else} \end{cases} \quad (9)$$

When the considered surface is a triangle, which is our case, the $\vec{V}_{Inertia-Center}$ is:

$$\vec{V}_{Inertia-Center} = \frac{\sum_{\text{i is a vertex}} \vec{V}_i}{3}$$

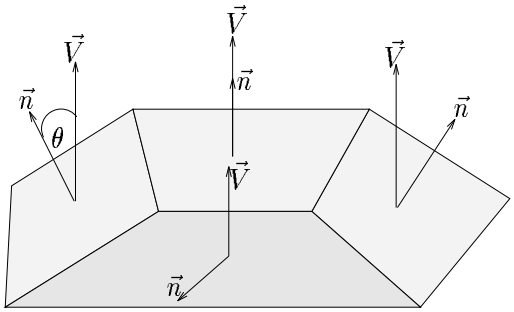

Figure 27: The viscous force applied on a facet depends on the area of this facet and on the angle between the normal on this facet and its velocity.

5 Solving Robotic tasks

The purpose of the *Robot*Φ system is to make it possible to create a *virtual world* in which objects and robots have a physical behaviour consistent with the real world. This possibility is obviously very useful to program or to plan robot tasks involving complex contact interactions which can not be processed using classical geometric models. This section shows how the concept of the physical world can be used to solve some of the main problems of robotic programming.

5.1 Motion generation

One of the main aspects of a physical object is its ability to be manipulated by an external force. This property can be used to plan the motion of a robot. Two cases have to be considered: mobile robots and manipulators.

Mobile robot: A straightforward way to bring a mobile robot to a new desired position is to attract it by a "physical attractor". This attractor is a virtual duplication of the robot itself. It is located at the new desired configuration. The actual robot is connected to the virtual one by means of an appropriate set of spring/damper connectors (figure 28.*a*). The choice of the type and the location of these connectors depends on the tasks to be achieved. If the new configuration is completely defined (position and orientation) we have to connect (by means of LSs connectors) at least four non coplanar particles of the real robot with their corresponding one on the virtual one. TS connectors are useful to constraint the trajectory to be followed by the robot (straight lines or arcs of circle).

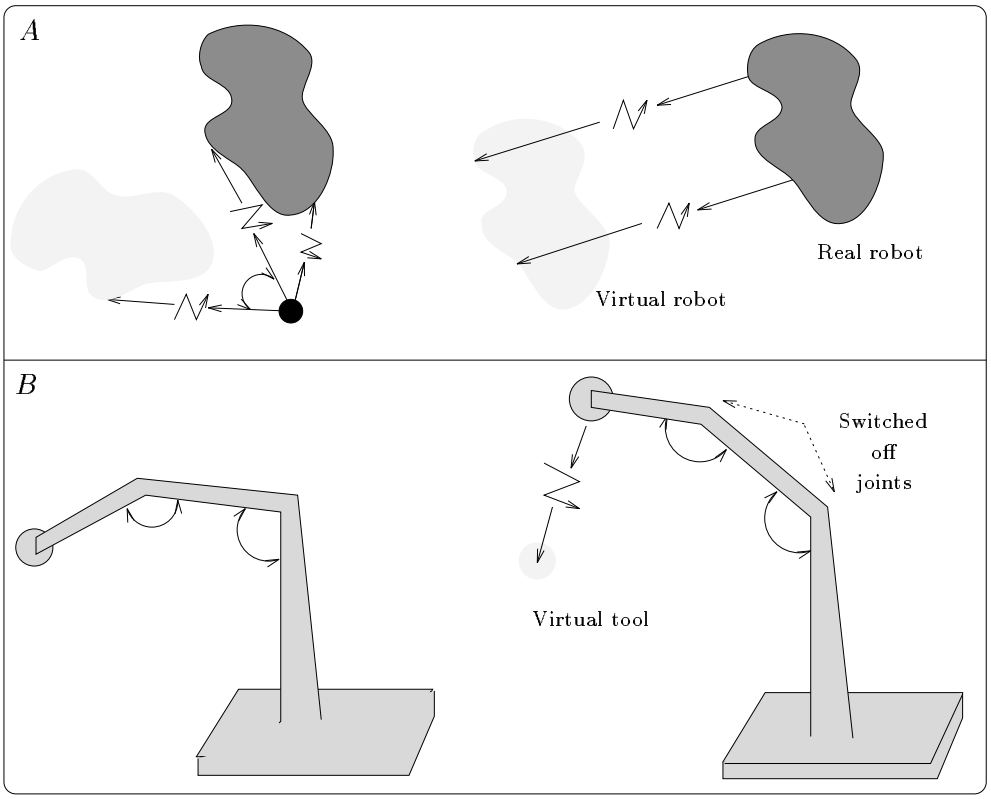

Figure 28: The physical path planning.

Manipulator: A physical way to move a manipulator to a desired configuration is to temporarily switch off their joints while attracting its tool to a virtual one located at the desired position. As the switched off joints do not resist the external forces, they will follow the tool according to their mechanical characteristics (figure 28.*b*). A switched off joint (or passive joint) is characterised by $\lambda = 0 \; and \; \mu \simeq \infty$. At each time step, the joints values are sent to the real robot, which will generate the same motion.

5.2 Obstacles avoidance

Physical objects are capable of interacting with each other. When an object collides with an obstacle, the collision force (§ 4.4) will prevent it from continuing to move in the same direction. So the object will slip (when applying this algorithm, one does not consider the friction force between the object and the obstacle) along the surface of the obstacle (figure 29.a). A simple way to avoid the collision with the obstacle when physical contacts are not allowed, is to expand the obstacles by a given security distance. Local minima are characterized by a null speed of the moving object. Avoiding local minima is possible by adding progressively virtual obstacles in the local minimum (the local minimum will be progressively filled). At the next attempt the object will slide along the surface of the local new virtual obstacle (figure 29.b). This method is efficient when the robot has no constraint on its motion.

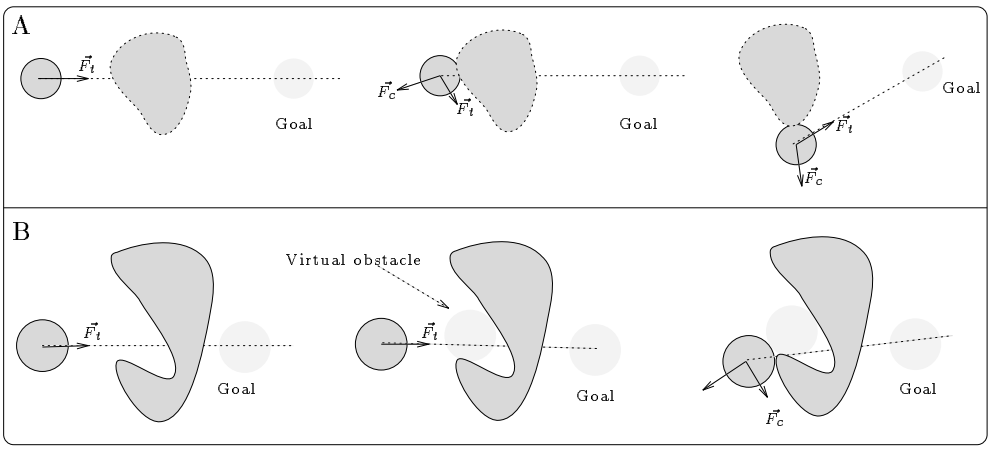

Figure 29: Obstacle avoidance.

5.3 Stabilisation

Any object can automatically search its equilibrium state (i.e the state corresponding to a minimum potential energy). This state is characterised by a null

external force. An object is free if the sum of the external forces applied on it is null. The speed of the inertial center of a free object is constant. If the initial speed of a free object is null, the only possible motion of this object is to rotate around its inertia centre. In a constrained environment this motion will stop because of the collision with other objects. This property is useful to ensure the stability of a grasp executed by an articulated hand. In this case the object is initially stable, the environment is limited by the hand, and the external forces applied by the hand on the object are null if the finger-tips are attracted to each other by means of LS connectors: the sum of the forces generated on the extremities of a LS connector is always null (figure 30).

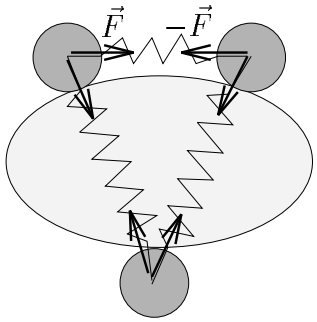

Figure 30: Adding LS connectors between the finger-tips guarantee that the sum of the external forces is null.

The stability of a power grasp can also be obtained using physical model: this stability is the result of a large number of contacts between the hand and the object as shown in the figure 31.

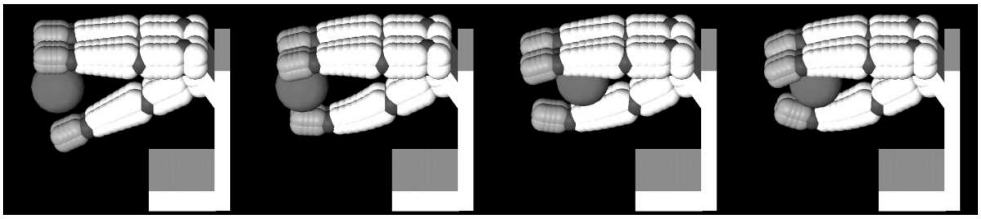

Figure 31: Power grasp simulation.

6 Experimental simulation

The system *Robot*Φ has been implemented using the $3D$ visualisation system *Geomview* [8]. The three basic mechanical behaviours of a stiff bar (traction, flexion, torsion) has been initially tested. Then we have simulated more complex phenomena. Figure 32 shows the difference between the behaviour of a wheel (which turns) and the behaviour of a cube (which topples over the earth) when they are submitted to the gravity force. With a small friction force the wheel slids only, with a large friction force the wheel turns only, else it slides and turns at the same time. Figure 33 shows a vehicle which automatically finds its configuration when crossing a hilly terrain. Grasp stability has been simulated using a physical model of our Salisbury hand. A very stiff collision ($lambda = 2000$) between a pyramid and a table has been simulated (figure 33) the needed sampling frequency varies between 300 and 600 Hertz.

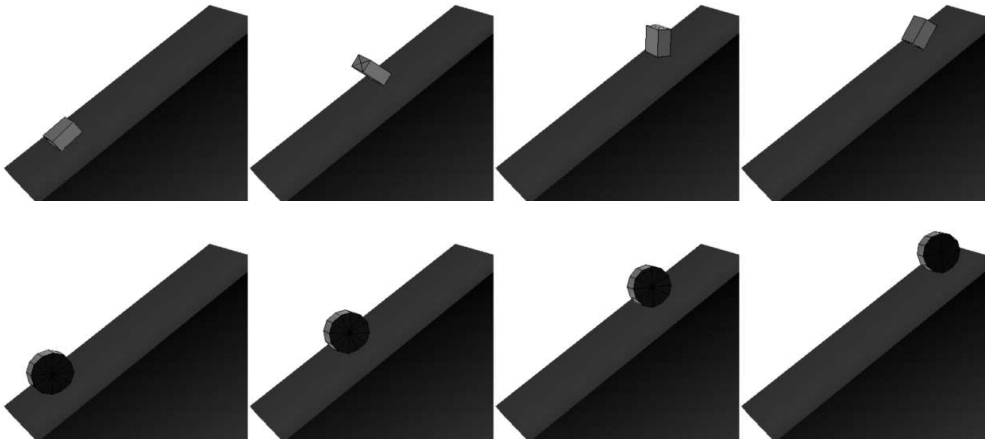

Figure 32: The difference between the behaviour of a wheel which turns and the behaviour of a cube which topples over the earth

The correctness of the mathematical approach used to find the trajectory of an object has been tested using MAPLE. For the cases which can be solved analytically, we have compared the adaptive time step approach ATS with the constant time step approach CTS (figure 21). Using ATS, the numerical solution converges always and is more accurate than the solution obtained using CTS, the incured error in position is proportional to the incured error in mechanical energy. The computation time needed to do one iteration using ATS is about 1.5 times the computation time using CTS, but the gain in iteration number is always greater than 1 (for a rigid collision this gain can

[8] Geomview, is an interactive program for viewing and manipulating geometric objects. It is available for free via anonymous ftp on the Internet from host 'geom.umn.edu' (IP address 128.101.25.35).

reach 100000). The execution time depends on the task to perform. For certain examples one can reach a real time execution, for others it is not the case. The simulation of the motion of the Salisbury hand (figure 31) during 50 seconds needed 50 seconds to perform using ATS (it needs 250s using CTS).

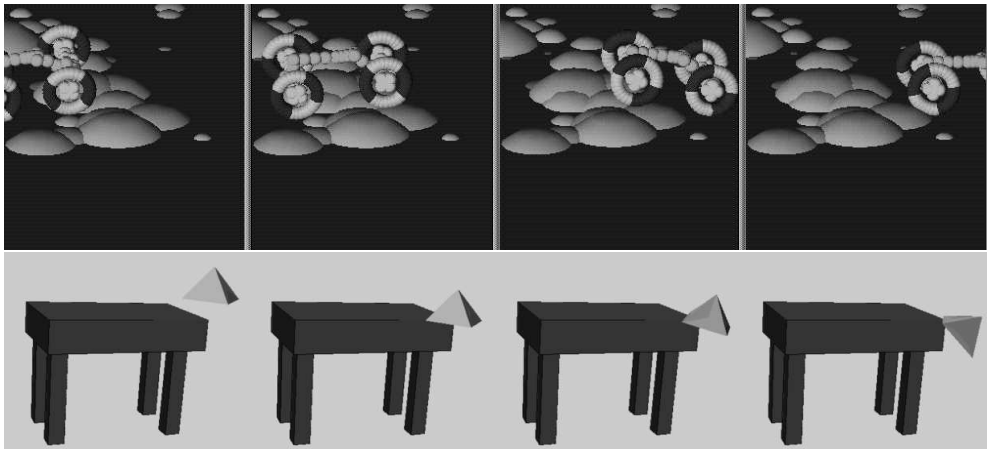

Figure 33: Some simulations executed using *Robot*Φ.

Current work [9] tries to study the effect of the chosen position of an artificial ligament on the knee behaviour[10].

7 Summary and Conclusions

In this paper we have describe the *Robot*Φ system. This system allows us to model robots and their environments in order to simulate their motions, their deformations, and their interaction with the environment. This system allows us to semi-automatically create a physical representation of an object from its geometrical model, this physical representation respects the inertia matrix and the inertia center of the real object and it optimises the needed particle number. It uses an energy-based adaptive time step in order to avoid the numerical divergence, to estimate incurred errors and to reduce the computation time. The motion complexity at each time step $O(n)$, where n is the number of particles in the environment.

Current work integrates some efficient algorithms to detect contacts between objects in order to accelerate the execution time. The truncation error of the computer is also an important factor, we try to study this error in order to add recompense factors.

[9] In cooperation with the TIMC laboratory at Grenoble/FRANCE

[10] We have get some data using an animal knee, we have construct a physical model of the ligament and we tries to do the same thing for the bones to compleat the model.

Acknowledgements

The work presented in this paper has been partly supported by the CNES (Centre National des Etudes Spatiales) through the RISP national project, and by the Rhône-Alpes Region through the IMAG/INRIA Robotics project SHARP.

Thanks

I would like to thank Christian Laugier, the supervisor of my PhD, for his remarks, his encouragement, and for all what he did in order to accelerate the development of this work, and Emanuel Da Vinha for his participation in some part of this work during his *DEA* stage. Also I would like to thank Ambarish Goswami and Alistair Mclean for re-reading this report. Finally many thanks for our team *SHARP* and our laboratory *LIFIA* for their nice work environment.

References

[1] A.Joukhadar, C.Bard, and C.Laugier. Combining geometric and physical models , the case of a dextrous hand. *IEEE/International Conference on Intelligent Robots and Systems "IROS"*, Septembre 1994.

[2] A.Joukhadar, C.Bard, and C.Laugier. Planning dextrous operations using physical models. *IEEE/ICRA*, May 1994.

[3] A.Joukhadar and C.Laugier. Dynamic modeling of rigid and deformable objects for robotic task: motions, deformations, and collisions. *International conference ORIA*, Decembre 1994.

[4] Marc Atteia and Michel Pradel. *Elemement d'analyse numerique.* CEPADUES-EDITIONS.

[5] C.Laugier, C.Bard, M.Cherif, and A.Joukhadar. Solving complex motion planning problems by combining geometric and physical models: the case of a rover and of a dextrous hand. *Workshop on the Algortihmic Fondations of Robotics (WAFR)*, February 1994.

[6] Ming C.Lin and John F.Canny. A fast algorithm for incremental distance calculation. *IEEE/ICRA*, April 1991.

[7] Ming C.Lin, Dinesh Manocha, and John Canny. Fast collision detection between geometric models. Technical Report TR93-004, January 1993.

[8] M. Gascuel, A. Verroust, and C. Puech. A modelling system for complex deformable bodies suited to animation and collision processing. *Visualisation and computer animation*, 1991.

[9] D. Gay and J. Gambelin. Une approche simple du calcul des structures par la méthode des éléments finis. *Hermès*, 1989.

[10] J.-P. Gourret. *Modélisation d'images fixes et animées.* MASSON, 1994.

[11] Stephane H.Carandall, Dean C.Karnopp, Edward F.Krtz, and David C.Pridmore-Brown. *Dynamics of mechanical and electronical systems.* Krieger publishing company, Malabar, Florida, 1968.

[12] J.Dorlot, J.Batlon, and J.Masounve. *DES MATRIAUX.* Ecole polytechnique de Montereal, 1986.

[13] S. Jimenez and A. Luciani. Animation of interacting objects with collision and prolonged contacts. *Modeling in computer graphic*, 1993.

[14] A. Luciani and al. An unified view of multiple behaviour, flexibility, plasticity and fractures: balls, bubbles and agglomerates. In *IFIP WG 5.10 on Modeling in Computer Graphics*, pages 55–74. Springer Verlag, 1991.

[15] Sandra Martelli. Investigations on acl biomechanics: Towards computer assisted functional reconsruction. Technical Report N.215.26, July 1994.

[16] D. Metaxas and D. Terzopoulos. Dynamic deformation of solid primitives with constraints. *Computer Graphics, volume 21, Number 2*, july 1992.

[17] Brian Mirtich and John F.Canny. Impulse-based, real time dynamic simulation. *Workshop on the Algortihmic Fondations of Robotics (WAFR)*, February 1994.

[18] P. Sikka, H. Zhang, and S. Sutphen. Tactile servo: Control of touch-driven robot motion. *Third international symposium on experimental robotics*, octobre 1993.

[19] C. Thibout, P. Even, and R. Fournier. Virtual reality for teleoperated robot control. *International conference ORIA*, Decembre 1994.

[20] D. Williams and O. Khatib. The virtual linkage: A model for internal forces in multy-grasp manipulation. *IEEE*, 1993.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399