Intelligent Autonomous Systems 10
W. Burgard et al. (Eds.)
IOS Press, 2008
© 2008 The authors and IOS Press. All rights reserved.
doi:10.3233/978-1-58603-887-8-374

Design and Implementation of Humanoid Programming System Powered by Deformable Objects Simulation

Ryohei Ueda, Takashi Ogura, Kei Okada and Masayuki Inaba [b]
ueda@jsk.t.u-tokyo.ac.jp
[b] Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

Abstract. This paper describes the design and implementation of a humanoid robot system that is capable of simulating deformable objects such as a cloth or a towel. Two key issues of the program system design are discussed: 1) adding dynamics to selected objects at the selected time. 2) the simple API for mesh creation of deformable objects. This system helps users to develop deformable object manipulation and recognition. Finally, a cloth recognition and motion generation carrying a mat using the developed system is presented.

Keywords. Humanoids, Deformable Objects Simulation, Robot Simulator, Recognition of Cloth

1. Introduction

Humanoid robots are expected to assist human activities in daily life. Therefore humanoid robots need to deal with various kinds of objects. Especially, it is important for humanoid robots to be able to deal with deformable objects like cloth.

In order to simulate flexible objects in a humanoid robot simulator, the function of deformable objects simulation is needed. However the existing humanoid robot simulators, such as OpenHRP[1], FAST[2], Robotics Studio[3], do not incorporate soft body simulation. The existing humanoid robot simulation systems need to consider how to embed deformable objects simulation into them.

When humanoid robots recognize deformable objects such as a cloth, it is difficult to recognize deformable objects with visual processing using the only real robot's view images because of its deformation and occlusion. It may be efficient to apply flexible models for recognition of deformable objects. Kita et al. shows the effectiveness of a deformable model for estimating the states of clothing[4]. Felzenszwalb adopted deformable template models to detect deformable shapes[5].

In grasping field, Howard et al. used 3D physically-based model for grasping soft body objects by two cooperative manipulators[6].

In this paper, the design and implementation of a deformable objects simulation embedded into the existing humanoid robots programming environment. The simulator is applied to a humanoid robot for recognition and handling of deformable objects.

2. Humanoid Robot Simulator Embedded into a Robot Brain

The robot simulator presented in this paper is embedded into a robot brain and applied for recognition and motion generation.

A robot simulation should enable the user to realize an efficient software development. A virtual robot model is used to test new algorithms before using them for experiments on the real robot. The following requirements have to be fulfilled:

I The robot models in the simulator can be controlled with the same interface as the real robot.
II The sensors of the robot can be emulated in the simulator.

Robot simulators are also expected to be embedded into a robot brain. They are used to predict a situation or to plan a specific task. In order to meet these requirements, a robot simulator additionally needs to have the following functions:

1. Addition of an object to the virtual world at any time.
2. Adding or removing dynamics to an object at any time.

1. is required when a robot finds a new object. The object is added to the virtual world of the robot brain.

2. is needed to reduce the computational load. When a robot execute a task in a complex environment, such as a room, the robot needs to deal with many objects. However, if the dynamics of every object are simulated the computational load will be enormous. It is therefore important to compute only the dynamics of the task objects the robot deals with.

Ogura et al. developed EusDyna[7], which meets the requirements described above. In this paper the incorporation of a deformable objects simulation into EusDyna is discussed.

Fig.1 illustrates the system configuration of EusDyna. EusDyna is implemented in EusLisp[8], which is a robot system description language. EusLisp is a Lisp implementation capable of dealing with three dimensional shapes and multithreading.

The humanoid robots in EusDyna have the same interface to the real robots and the software resources developed in EusLisp can be used in EusDyna. Thus it is easy to embed a simulation engine into a robot brain.

EusLisp is capable to deal with three dimensional shapes. The robot and environment models are represented by non-dynamic three dimensional objects. EusDyna adds dynamics to these objects using an external simulation engine. EusDyna is also capable of switching the external simulation engine between ODE[9],PhysX[10],and Math Engine of Vortex.

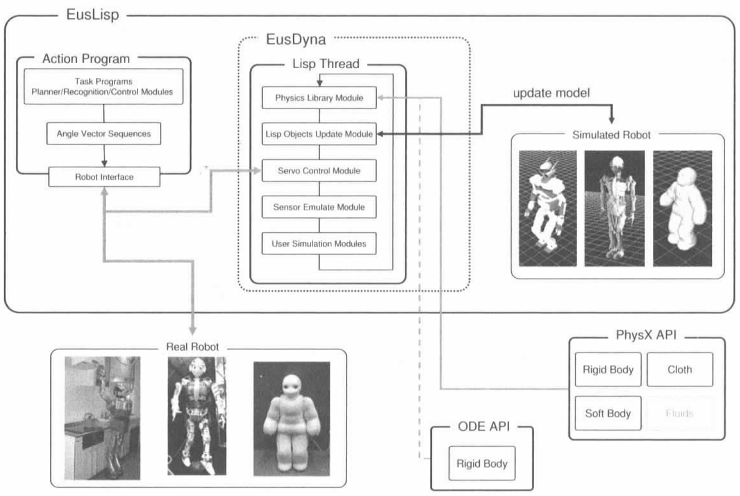

Figure 1. Configuration of Humanoid Robot Programming System and EusDyna

3. Implementation of a Simulation Environment Incorporating Deformable Objects

3.1. Mass-Spring System Model of Deformable Objects in PhysX

It is effective to use the simulator for action generation. An application example is when the robot recognizes objects of the real world and this information is sent to the simulation where a virtual robot performs actions on the objects and learns how to handle them. This virtual learning process allows the robot brain to acquire a knowledge of its environment and can be used by the real robot encounting these same objects in the real world. If this learning process shall be performed online the simulation speed is important.

We use the game dynamics engine "PhysX", because PhysX is capable to simulate the behavior of rigid bodies in a fast time by PPU (Physics Processing Unit) and is equipped with a deformable objects simulation that can treat with 2-D flexible objects (Cloth model), 3-D flexible objects (Soft body model) and liquids (Fluids model). In this paper, soft body model and cloth model are focused on.

In PhysX, the modeling method of Position Based Dynamics[11] is used for the deformable objects model.

Fig.2 shows the components of the Soft Body and Cloth model of PhysX. A tetrahedra element is used for the Soft body model. A tetrahedron mesh is used to represent the different shapes of 3-D deformable object. The Cloth model component is a triangle.

3.2. Adding Dynamics at a Selected Time

EusDyna is capable of adding dynamics to a selected object at any time as described above. This means that the dynamic calculation of every object has to be started first.

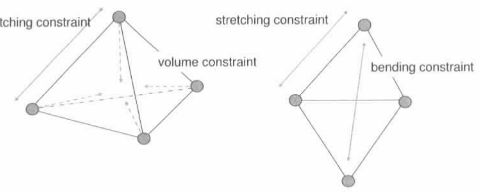

Figure 2. Tetrahedra : Component of Soft Body and Triangle : Component of Cloth

In the case of a deformable object its deformation is only computed when it is made to be simulation target beforehand. Otherwise, it corresponds to a rigid body and it conserves its initial shape. Fig.3 shows a cloth model before and after applying the dynamic calculation. By treating deformable objects in this way the software development efficiency increases because the position and orientation of the deformable objects can be defined before the dynamics calculation is started. There is another advantage in embedding into a robot brain. When a robot recognizes deformable objects, it is difficult to determine whether rigid or soft a motionless deformable object is. It is not until a robot can find deformation that a deformable object is distorted. Therefore it may be efficient to recognize a motionless flexible object as a rigid body first and recognize it as a deformable object when it is deformed. In this recognition way, treatment of deformable objects in EusDyna described above is functional.

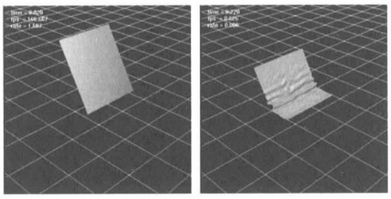

Figure 3. Left : Before Adding DynamicsRight : After Adding Dynamics

3.2.1. API for Mesh Creation of Deformable Objects

When using a deformable object model the mesh creation is a difficult problem. Furthermore, the mesh of a soft body is one of the major elements which determines its behavior. Taking into account the facility of programming and being embedded into a robot brain a simple API for mesh creation is required. Fig.4 shows the constitution of a deformable cube which is a primitive shape of deformable objects of the API of EusDyna. The shape of a soft cube is defined by the edge length in x, y and z axis and the mesh resolution in x, y, and z axis. The cube is a three dimensional deformable object and consists of a tetrahedron mesh. It is assembled of vowels and each of the vowels is composed of five tetrahedron. Fig.5 shows the images of a deformable cube simulation using this system. Fig.6 is the source code of Fig.5 simulation.

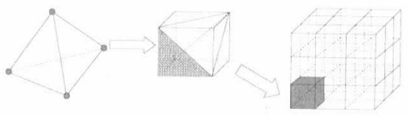
Figure 4. Constitution of Soft Body Cube

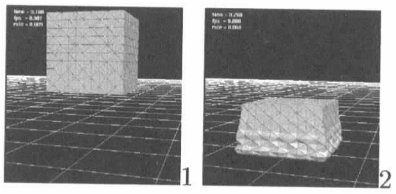

Figure 5. A Deformable Cube in EusDyna

```
(d-init) ;;initialize dynamics
(setq *sc* (instance soft-body
    :init :cube (list 200 200 200 10 10 10)))
        ;;create soft-body cube
(setf (get *sc* :face-color) :gray)
(send *sc* :locate #f(0 0 300))
(d-make *sc*) ;;add dynamics to *sc*
(send *sc* :draw-tetra)
    ;;select the mode of drawing
(objects (list *sc*)) ;;draw *sc*
(d-start)   ;;start simulation
```

Figure 6. The Source Code of Fig.5

4. Experiments

4.1. Recognition of Deformable Objects

4.1.1. Estimating the state of a cloth

In this section a cloth recognition method using this system is presented. The problem situation is the following:

1. Determining where a cloth is picked up.
2. The robot is given only two images(Fig.7).
3. The environment information such as the height of the desk is known.

In order to solve this problem the EusDyna system is used. The method is described below.

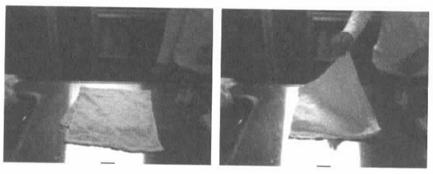
Figure 7. Picking Up a Cloth

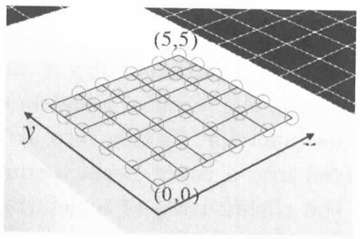

Figure 8. the Grid of a Cloth model

First, the cloth model is created. The parameters of the cloth shape are obtained by the left of Fig.7, and the height of the desk.

Second, the simulated cloth model is picked up. The candidates of the picking-up point are 6 × 6 grid points of the cloth model(Fig.8). Each of the cloth grid

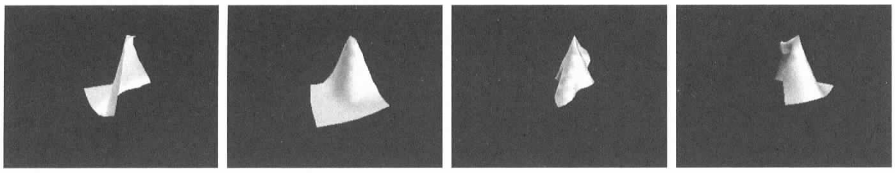

Figure 9. The Simulated Cloth Images

points are moved in the simulator towards the position of the hand, which is obtained by stereo vision.

Third, the simulated cloth model is compared with the cloth in the real world. The image of the cloth in the real world, the right of Fig.7, is obtained from the view of the real robot. The images of the cloth model in the virtual world are created from the robot model view placed in the virtual environment.The similarities between the simulated view images and the real view image are calculated.

The similarity is calculated using the cloth contour. The definition of the similarity is eq.(1).

$$Similarity = \frac{And(SimulatedImge, RealImage)}{PixelNum} + \frac{And(Not(SimulatedImage), Not(RealImage))}{PixelNum} \tag{1}$$

As a result, the most similar image of the simulated cloth is the second one of Fig.9. The image is picked up at $(5,0)$ in the grid coordinates of Fig.8.

4.1.2. Replaying the Cloth Folding Motion

In the next experiment, we control a cloth model in the simulator according to the behavior of the cloth in the real environment (Fig.10).

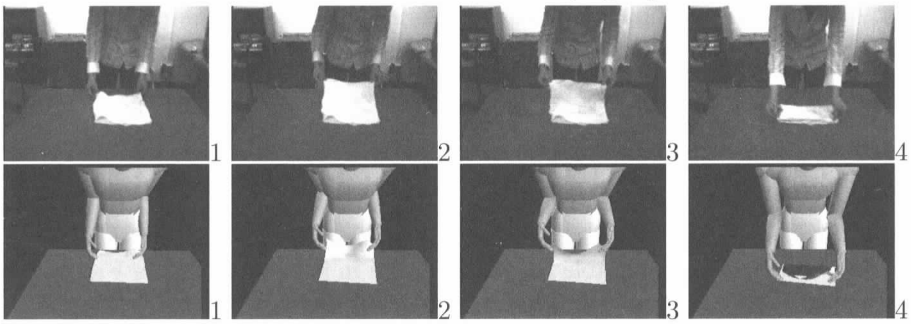

Figure 10. The Real and Simulated Images of Cloth Folding

In this experiment, the cloth model is created from the robot's view of a cloth before being folded. When the cloth is folded, the cloth model in the simulator is moved according to the position of the hands(Fig.11), which is obtained using stereo view of the real robot. When a robot recognizes the flexible object which deforms largely, it is difficult to recognize it based on the visual processing using the only real robot's view image. This experiment shows the effectiveness of recognition using a deformable objects simulation.

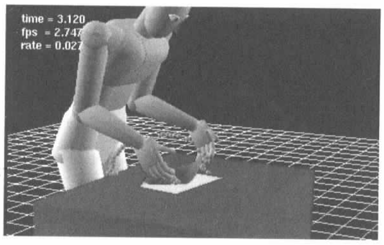

Figure 11. The Path of the Man's Hands

In addition, the action for a playback of the motion may be planned using this system.

4.2. Motion Generation of Carrying a Mat

In the next experiment, we present the method to acquire a stable motion for carrying a mat using the system developed. In full search, the trial programs are executed after the initial condition is defined based on each of the parameters.

The motion parameters are the distance of the robot's hands in x and y direction from the center of gravity of a mat(Fig.12). *ArmDistance* is changed in 10 steps and each of $x_{DeformabelObjects}$ and $y_{DeformabelObjects}$ is varied in 3 steps.

The value function is eq.(3). The less the value of F is, the more stable the motion is. In addition, the threshold in order to detect whether the mat is fallen down(Fig.13).

$$dF = \left(\frac{\sum_{v \in V} v}{\sum_{v \in V}} - P_{\text{humanoid}} - D_0 \right) \tag{2}$$

$$F = \sum_{\text{for all simulation step}} dF dt_{sim} \tag{3}$$

Where V are the vertices of the deformable objects, P_{humanoid} is the position of the robot and D_0 is the initial distance between the robot and the mat.

As a result, the most stable motion is shown in the left images of Fig.14. The right images of Fig.14 are one of the unsuccessful motions which is valued less.

5. Conclusion

In this paper, a system configuration consisting of a deformable objects simulation was presented. The simulation is embedded in a humanoid robot programming environment. The applicability of the method was shown at the examples of cloth recognition and mat carrying motion generation.

The key feature of the system is:

1. Adding dynamics to the selected objects at any time

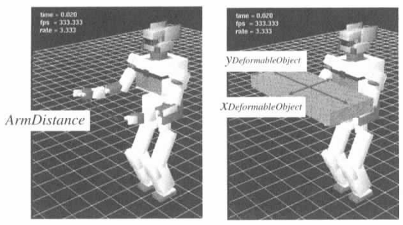

Figure 12. Simulation Parameters

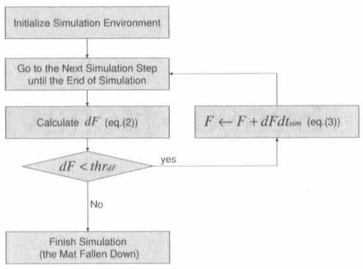
Initialize Simulation Environment
Go to the Next Simulation Step until the End of Simulation
Calculate dF (eq.(2))
F ← F + dFdt_sim (eq.(3))
dF < thr_dF
yes
No
Finish Simulation (the Mat Fallen Down)

Figure 13. The Flow Chart of a Trial in Simulation

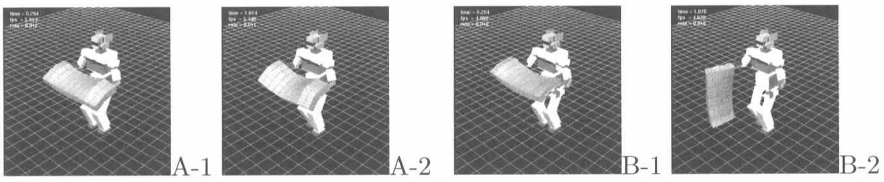

Figure 14. The Temporal Images of Carrying a Mat. A-1 and A-2 are the Most Stable Motion. B-1 and B-2 are One of the Unsuccessful Motion.

2. Simple API for mesh creation of deformable objects

For future research, it is planned to use this system for teaching motions such as cloth folding.

References

[1] Fumio Kanehiro, Kiyoshi Fujiwara, Shuuji Kajita, Kazuhito Yokoi, Kenji Kaneko, Hirohisa Hirukawa, Yoshihiko Nakamura, and Katsu Yamane. Open architecture humanoid robotics platform. In ICRA, pages 24–30. IEEE, 2002.

[2] Kei Okada, Yasuyuki Kino, Fumio Kanehiro, Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue. Rapid development system for humanoid vision-based behaviors with real-virtual common interface. In Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS'02), pages 2515–2520, 9 2002.

[3] Microsoft. Robotics Studio. http://www.microsoft.com/.

[4] Y. Kita and N. Kita. A model-driven method of estimating the state of clothes for manipulating it. pages 63–69, 2002.

[5] P.F. Felzenszwalb. Representation and detection of deformable shapes. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 27(2):208–220, Feb. 2005.

[6] A.M. Howard and G.A. Bekey. Recursive learning for deformable object manipulation. Advanced Robotics, 1997. ICAR '97. Proceedings., 8th International Conference on, pages 939–944, 7-9 Jul 1997.

[7] T. Ogura, K. Okada, and M. Inaba. Realization of Dynamics Simulator Embedded Robot Brai for Humanoid Robots. In IEEE International Conference on Robotics and Automation (ICRA2007), pages 2175–2180, 2007.

[8] Toshihiro Matsui and Masayuki Inaba. Euslisp: An object-based implementation of lisp. Journal of Information Processing, 13(3):327–338, 1990.

[9] Open Dynamics Engine ODE. http://ode.org/.

[10] AGEIA. PhysX. http://www.ageia.com/.

[11] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. J. Vis. Comun. Image Represent., 18(2):109–118, 2007.