2010 11th Int. Conf. Control, Automation, Robotics and Vision
Singapore, 5-8th December 2010

Modelling and Simulation of a Multi-fingered Robotic Hand for Grasping Tasks

J. A. Corrales, C. A. Jara and F. Torres
Physics, Systems Engineering and Signal Theory Department
University of Alicante
03690 San Vicente del Raspeig (Alicante), Spain
{jcorrales; carlos.jara; fernando.torres}ua.es

***Abstract*— This paper develops the kinematic, dynamic and contact models of a three-fingered robotic hand (BarrettHand) in order to obtain a complete description of the system which is required for manipulation tasks. These models do not only take into account the mechanical coupling and the breakaway mechanism of the under-actuated robotic hand but they also obtain the force transmission from the hand to objects, which are represented as triangle meshes. The developed models have been implemented on a software simulator based on the Easy Java Simulations platform. Several experiments have been performed in order to verify the accuracy of the proposed models with regard to the real physic system.**

***Keywords*—robotic hand, robotic manipulation, kinematics, dynamics, VRML, tessellation, contact, breakaway.**

I. INTRODUCTION

The development of powerful computational platforms for simulating the behavior of robotics systems constitutes a fundamental tool for designers, users, and researchers of this field. These simulation platforms are very important for robotic manipulation because they enable the computation of the robot's trajectories which are required for grasping objects in the environment. It is worth mentioning the following software platforms for robotic manipulation:

- Easy-Rob3D [1]: simulation software of robot arms specifically designed for industrial applications. This program uses OpenGL as 3D graphics support.
- Qilex Simulator [2]: open-source software tool of a robotic system which consists of a Stäubli *RX90* robot arm with robotic hand of 4 MA-I fingers at the end-effector. This simulator implements the kinematics of the system and allows users to import 3D objects in the virtual environment.
- GraspIt [3]: software application exclusively created for the study of robot grasping. This simulator allows users to insert a wide range of robotic hands (BarrettHand, DLR, Robonaut and Rutgers), robot arms and mobile robots. This software platform implements collision detection among the 3D objects and computes the contact points.

Multi-fingered robotic hands are becoming more and more widespread in robotic manipulation applications because they not only enable the robot to grasp objects of different shapes, but they can also perform manipulation tasks which require dexterity [4]. These robotic systems are usually based on under-actuated schemes due to their compactness and their simplicity in control [5]. These special features should be taken into account by simulation platforms which model the behavior of these systems.

The joints of under-actuated fingers are generally coupled by transmission systems (such as tendons and gear trains) so that the number of required actuators is inferior to the number of degrees of freedom of the finger. In order to reduce the loss of dexterity of the fingers, many under-actuated fingers implement decoupling mechanisms (based on springs or clutches) which change the linkage relation between the joints when an object blocks one of the phalanxes. Thereby, under-actuated fingers are able to envelope the objects to be grasped and adapt to their shape.

One example of a widely used robotic hand with under-actuated fingers is the BarrettHand [6]. This hand is composed of three articulated fingers and a palm which integrates four servomotors and the control electronics. The two phalanxes of each finger are driven by one of the servomotors while the bases of fingers 1 and 2 rotate around the palm in a spread movement which is generated by the fourth servomotor. When the inner phalanx contacts an object with a specific force, a breakaway system based on a clutch decouples the outer phalanx from the inner phalanx so that it can perform an enveloping grasp of the object.

Several previous papers [7-8] have only represented the normal linkage between the phalanxes without considering this breakaway system. Other research [3] has modeled this breakaway system but without considering the force/torque which activates it. The present paper develops a more complete description of this under-actuated hand, describing its kinematic and dynamic modelling and taking into account the torque threshold for the breakaway mechanism. In addition, a contact model for the interaction between the hand and objects has been developed. This model is based on the representation of the object to be grasped as a triangle mesh. Thereby, the simulation platform can determine at each step if the hand comes into contact with an object by applying an efficient collision detection algorithm and then calculates how forces are transmitted to that object through the contact points.

978-1-4244-7815-6/10/$26.00 ©2010 IEEE

The authors have used Easy Java Simulations (EJS) [9] for the development of the simulation platform which implements these models (kinematics, dynamics, object and contact) of the robotic hand. EJS is an open-source software platform based on Java and Java3D which represents a powerful tool for easily developing simulations with a higher graphical interface capacity. This software provides a series of advantages over the robotic software platforms mentioned above:

- EJS is easy to use and it does not require special programming skills for the final user.
- EJS is platform independent and it generates high quality graphical simulations.
- EJS creates all the necessary files for the execution of the final application as an applet or as a stand-alone Java application.

In addition, the authors have an extensive experience in the development of robotic applications using this software (see Fig. 1). The readers can experiment with a lot of simulations developed with EJS in the following web page [10].

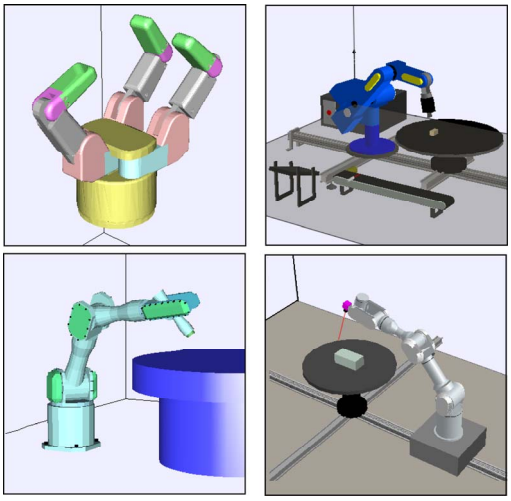

Figure 1. Several robotic applications developed with EJS.

The remainder of this paper is organized in six sections. Section 2 describes the modeling of the robotic hand, including the direct and inverse kinematic models, the inverse dynamic model and the breakaway detection procedure. Section 3 explains how objects are imported into the software platform from VRML files and how their surfaces are represented as triangle meshes by applying a tessellation algorithm. Section 4 presents the contact model which uses these triangle meshes to verify if the hand touches any object of the environment and then computes the contact forces. Section 5 shows the simulation results obtained. Finally, the last section enumerates the conclusions of this paper and future work.

II. MODELLING OF THE ROBOTIC HAND

A. Direct Kinematic Model

First of all, a coordinate system is assigned to each link of each finger k of the hand according to the standard Denavit-Hartenberg convention [11]. Fig. 2 depicts all the frames assigned to each finger.

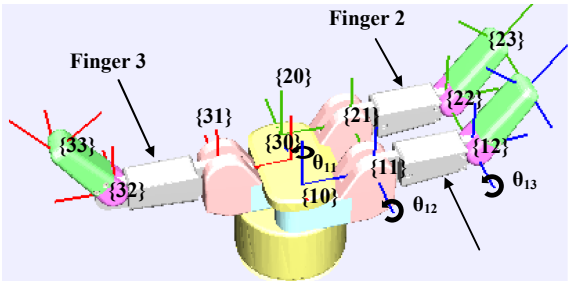

Figure 2. Coordinate frames assigned to each finger.

Next, a set of Denavit-Hartenberg parameters are defined for each coordinate system (see Table 1). The values for the dimension constants of the links are shown in Table 2.

TABLE I. DENAVIT-HARTENBERG PARAMETERS FOR EACH FINGER K.

Frame	a_{ki}	α_{ki}	d_{ki}	θ_{ki}
{k1}	A_1	90º	0 mm	$r \cdot q_{k1}$ [a]
{k2}	A_2	0º	0 mm	$q_{k2} + \Phi_2$
{k3}	A_3	0º	0 mm	$q_{k3} + \Phi_3$

a. r = [1, 1, 0] with k= [1, 2, 3]

TABLE II. DIMENSIONS OF THE LINKS OF THE BARRETTHAND.

Parameters	Values
A_1, A_2, A_3	0.050, 0.070, 0.056 m
Φ_2, Φ_3	2.46º, 50º

Finally, the joint angles are related with the encoder counts of the motors by the following transmission equations:

$$\theta_{11} = -2/35 \cdot q_{M4} \quad (1)$$

$$\theta_{21} = 2/35 \cdot q_{M4} \quad (2)$$

$$\theta_{k2} = 1/125 \cdot q_{Mk} \quad (3)$$

$$\theta_{k3} = 1/375 \cdot q_{Mk} \quad (4)$$

Where $q_{Mk} \in [0, 17500]$ with $k = [1,2,3]$ is the number of encoder counts of the motor of finger k. $q_{M4} \in [0, 3150]$ is the number of encoder counts of the spread motor.

B. Inverse Kinematic Model

If the fingertip of finger k is located at the position with coordinates ${}^{k0}p = ({}^{k0}p_x, {}^{k0}p_y, {}^{k0}p_z)$, the corresponding joint values $(\theta_{k1}, \theta_{k2}, \theta_{k3})$ can be computed by considering the geometric relations between the links.

In particular, the angle θ_{k1} for fingers 1 and 2 is obtained with the following expression:

$$\theta_{k1} = \arctan({}^{k0}p_y / {}^{k0}p_x) \qquad \text{for } k = 1,2. \quad (5)$$

In the case of finger 3, θ_{31} is always zero because it does not perform spread movements. After the computation of θ_{k1}, angles θ_{k2} and θ_{k3} can be calculated by taking into account that the two phalanxes of the finger are contained in a plane kII, as shown in Fig. 3.

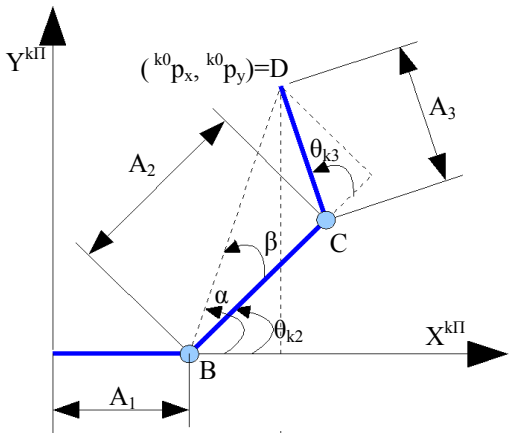

Figure 3. Geometric relations of the links of a finger.

Thereby, by applying the law of cosines to the BCD triangle in Fig. 2, the following expression can be derived for the computation of the angle θ_{k3}:

$$\theta_{k3} = \arccos\left(\frac{\left(\sqrt{{}^{k0}p_x^2 + {}^{k0}p_y^2} - A_1\right)^2 + {}^{k0}p_z^2 - A_2^2 - A_3^2}{2A_2A_3}\right) \quad (6)$$

Finally, the angle θ_{k2} is obtained from the difference between the angles α and β in Fig. 3:

$$\theta_{k2} = \alpha - \beta = \arctan\left(\frac{{}^{k0}p_z}{\sqrt{{}^{k0}p_x^2 + {}^{k0}p_y^2} - A_1}\right) - \arctan\left(\frac{A_3\sin(\theta_{k3})}{A_2 + A_3\cos(\theta_{k3})}\right) \quad (7)$$

C. Inverse Dynamic Model

For each finger k, its inverse dynamic model can be represented by the following equation:

$$\tau_k = D_k(\Theta_k)\ddot{\Theta}_k + H_k\left(\Theta_k, \dot{\Theta}_k\right) + C_k(\Theta_k) \quad (8)$$

Where Θ_k is a $n_k \times 1$ vector with the joint values, τ_k is a $n_k \times 1$ vector with the joint torques, D_k is the $n_k \times n_k$ inertia matrix, H_k is the $n_k \times 1$ Coriolis-centrifugal vector and C_k is the $n_k \times 1$ gravitational vector. n_k is the number of joints of each finger k ($n_1 = n_2 = 3$ and $n_3 = 2$).

The Lagrange-Euler algorithm [12] has been implemented in order to obtain the analytical representation of the matrices in (8), which have been implemented in the simulation platform. The input parameters for this algorithm are the following features of each link j of each finger k: its mass m_j, its inertia tensor I_j computed at the frame $\{kj\}$ and the coordinates $p_j = (p_{jx}, p_{jy}, p_{jz})$ of its centre of mass with respect to frame $\{kj\}$. The real values of these parameters for the BarrettHand are shown in Table 3. The gravity acceleration vector $g=(g_x, g_y, g_z)$ at the base frame $\{k0\}$ of each finger is also required as input parameter.

TABLE III. MASS PARAMETERS OF THE LINKS OF THE BARRETTHAND.

Parameters	Values
m_1, m_2, m_3	0.16252678, 0.05795741, 0.03421781 Kg
I_{1xx}, I_{1xy}, I_{1xz}	0.0000658, 0.0000572, -0.0000011 Kg·m^2
I_{1yy} , I_{1yz} , I_{1zz}	0.0001310, -0.0000013, 0.0001808 Kg·m^2
I_{2xx} I_{2xy} I_{2xz}	0.0000047, -0.0000020, 0.0000030 Kg·m^2
I_{2yy} ,I_{2yz} , I_{2zz}	0.0001208, -0.0000000, 0.0001218 Kg·m^2
I_{3xx}, I_{3xy}, I_{3xz}	0.00000298, -0.00000135, -0.00000032 Kg·m^2
I_{3yy}, I_{3yz}, I_{3zz}	0.00005836, -0.00000001, 0.00005732 Kg·m^2
p_{1x}, p_{1y}, p_{1z}	-0.01798334, -0.01569925, 0.00086502 m
p_{2x}, p_{2y}, p_{2z}	-0.03827982, 0.00107042, -0.00058628 m
p_{3x}, p_{3y}, p_{3z}	-0.03629026, 0.00182016, 0.00020041 m

D. Breakaway Mechanism Model

As stated in the introduction section, breakaway systems enable under-actuated fingers to perform enveloping grasps. In the case of the BarrettHand, this mechanism is implemented by a threaded shaft, a pair of Belleville spring washers and a spur gear [6]. In normal operation, the spur gear transmits the torque from the axis of the motor to the axis of the inner phalanx joint through the threaded shaft. However, if the inner phalanx contacts an object while the finger is closing and this object exerts a torque over the joint axis higher than a predefined threshold (τ_{break}); the spur gear is disengaged from the threaded shaft by the Belleville spring washers. Then, the motor torque is only transmitted to the outer phalanx while the inner phalanx remains mechanically jammed. The spur gear and the threaded shaft are reengaged when the finger is opening and passes through the joint position where the breakaway happened.

Previous simulation systems which take into account this breakaway mechanism [3] assume that it is activated as soon as the inner phalanx of a finger touches an object. Nevertheless, in order to develop a more precise model of the robotic hand, the torque threshold which activates mechanically the breakaway should be considered.

If an object contacts the inner phalanx at a point ${}^{k0}p_{c_i}$ and applies a force ${}^{k0}f_{c_i}$ and a torque ${}^{k0}n_{c_i}$ on it, then the torque τ_{k2} which is transmitted to axis of joint k2 can be calculated by the following equation:

$$\tau_{k2} = \left({}^{k0}\hat{Z}_{k1} \times \left({}^{k0}p_{c_i} - {}^{k0}p_{k1}\right)\right)^T {}^{k0}f_{c_i} + {}^{k0}\hat{Z}_{k1}^T {}^{k0}n_{c_i} \quad (9)$$

Where ${}^{k0}\hat{Z}_{k1}$ are the coordinates of the Z axis of frame $\{k1\}$ with respect to frame $\{k0\}$. This vector corresponds to the third column of the transformation matrix ${}^{k0}T_{k1}$, which is obtained by the direct kinematic model. ${}^{k0}p_{k1}$ are the coordinates of the origin of frame $\{k1\}$ expressed in frame $\{k0\}$. This vector corresponds to the fourth column of the transformation matrix ${}^{k0}T_{k1}$.

Therefore, the breakaway mechanism is activated when finger k is closing and the inner phalanx touches an object which generates a torque $\tau_{k2} > \tau_{break}$. The breakaway mechanism is deactivated when finger k is opening and joint $k3$ goes through the position q_{Mk_break} where the breakaway was activated.

Finally, this breakaway detection model should be considered in the kinematic and dynamic models. In the case of the kinematic model, when the breakaway is activated, the angles θ_{k3} and θ_{k2} are decoupled and the relation in (4) is not verified. In particular, θ_{k2} remains fixed while the breakaway is activated and θ_{k3} moves freely. In addition, the transmission relation between the motor of the finger and the third joint is changed from (4) to the following expression:

$$\theta_{k3} = 4/375 \cdot q_{Mk} - 3/375 \cdot q_{Mk\_break} \quad (10)$$

In the case of the dynamic model, not only the decoupling between θ_{k3} and θ_{k2} should be considered, but also the fact that $\dot{\theta}_{k2}$ is null. When the breakaway is deactivated, the models present their normal behaviors which were defined in the previous sections.

III. Modelling of the Objects

A. VRML Modelling

VRML (Virtual Reality Modeling Language) [13] is a file format that allows the creation of objects and interactive three-dimensional worlds. This format can represent any 3D object which can be modeled with a polygon mesh. It is possible to use Java3D like a viewer of VRML files. For that purpose, some VRML loaders developed for Java3D can be used. EJS uses open-source loaders to import VRML files in its 3D environment. Fig. 1 shows several robotics applications developed with EJS using VRML files.

The VRML format file describes a 3D scene by a hierarchical tree structure of objects (*nodes*). Every node represents some functionality of the scene such as a shape, a transformation, an appearance, a viewpoint, etc. As mentioned, some of them are shape nodes which represent real 3D objects, and others are grouping nodes (*groups*) used for holding child nodes.

The authors have developed a tessellation algorithm that is specifically optimized for VRML files with the objective of collision detection and computation of contact points, which are the essential part of the interaction modeling in grasping tasks. This computational algorithm is described in the next subsection.

B. Tessellation Algorithm

The VRML loader for Java3D builds a 3D object from a URL or file path called *BranchGroup* (BG). The resulting BG can then be placed in any Java 3D application's scene graph. The tessellation algorithm gets the BG, extracts its geometry and tessellates it by the following pseudo-code algorithm:

```
Procedure Tessellation VRML

#Extraction of the geometry
Input BG branch = VRML.load("file");
Input Shape3D shape;
Input Vector v;
Input int[] nElements, int level = 0;
Input Node node, element;
v.addElementAt(branch, level);
While (level>=0)
  element = v.getElement(level);
  If iElements[level]<element.numChildren()
    node = element.getChild(iElements[level]);
    iElements[level]++;
    If node is a Shape3D
      Geometry g = node.getGeomtry();
      shape.addGeometry(g);
    Else
      level++;
      v.addElementAt(nodel,level);
    End If
  Else
    iElements[level] = 0;
    level--;
  End IF
End While

#Tessellation process
Input Geometry geometry;
Input int points;
npoints = shape.getVextexCounts();
geometry = TriangleArray(npoints);
Output geometry;

EndProcedure
```

This algorithm gets all the geometry nodes, i.e. *Shape3D* nodes, of the VRML file and converts them to Java 3D shape objects. Afterwards, the tessellation process obtains their points and tessellates them with Java 3D methods. Fig. 4 shows the application of this tessellation algorithm to the robotic fingers and to an object sphere.

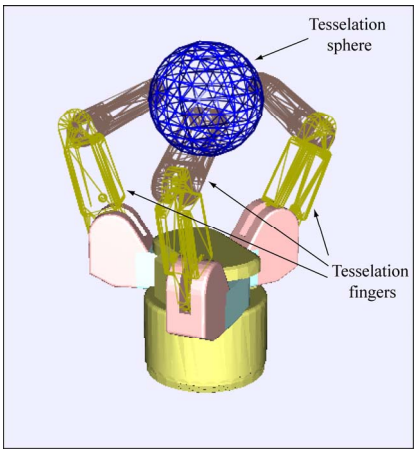

Figure 4. Tessellation of the surfaces of the hand's fingers and a sphere.

IV. Modelling of the Contact

The tessellation algorithm described in the previous section represents the objects to be grasped and the hand as meshes of triangles which are computed only once at the beginning of the simulation process. Afterwards, they are used by the simulation algorithm as an approximation of the objects' surfaces in order to compute the contact parameters. In particular, the simulation algorithm implements a contact model based on three main steps: collision detection, contact surface generation and contact force determination.

The simulator verifies at each simulation step if any component of the hand comes into contact with any object by applying a collision detection algorithm. This algorithm uses a hierarchy of DOPs (Discrete Oriented Polytopes) [14] in order

to reduce the number of intersection tests. Firstly, the collision detection algorithm verifies if the two root level bounding volumes collide. If they do not collide, the collision detection ends and no more tests are required. Otherwise, each of these bounding volumes is divided into two child bounding volumes and the collision detection is performed again between them. This subdivision process is repeated only between the colliding bounding volumes until the process reaches the last level of the hierarchy, which contains the triangles from the tessellation algorithm. Finally, the collision detection algorithm computes the intersection between each pair of triangles contained in the colliding bounding volumes [15].

The collision detection algorithm returns a group of intersection lines, including the indexes of the corresponding intersected triangles and their end points' coordinates. The contact surface generation algorithm orders these lines so that it obtains an intersection patch which identifies the contact surface. Next, the normal vector ${}^{k0}d_{c_i}$ of this patch is computed by summing all the normal vectors of the contained triangles, weighted by their areas. In addition, the center point ${}^{k0}p_{c_i}$ of this patch is computed as the center of the bounding box which covers the patch. With the normal vector ${}^{k0}d_{c_i}$ and the center point ${}^{k0}p_{c_i}$, the contact patch is simplified into a punctual contact. This process is repeated for each intersection patch and a group of collision points and normal vectors are obtained to characterize the contact between the hand and the object.

The last step of the contact model calculates the contact force that it is transmitted from the hand to the object through each contact point. If the hand touches the object on contact point c_i through link kj, the contact force applied over the object on that point can be computed from the torques on the kinematic chain of finger k by the following expression:

$$ {}^{k0}f_{c_i} = \sum_{m=0}^{j} \left[{}^{k0}p_{c_i} - {}^{k0}p_{km} \right]_{\times}^{-1} \tau_{km} \, {}^{k0}d_{c_i} \qquad (11) $$

Where $\left[{}^{k0}p_{c_i} - {}^{k0}p_{km} \right]_{\times}^{-1}$ is the inverse of the cross product matrix corresponding to the vector between the origin of frame $\{km\}$ and the contact point c_i. The joint torque τ_{km} is obtained from the inverse dynamics model of section 2.C. Finally, the computed contact force ${}^{k0}f_{c_i}$ is used by the simulation algorithm in order to determine if the breakaway mechanism is activated according to the algorithm described in section 2.D.

V. Simulation Results

The interface of the simulator developed is shown in Fig. 5. All the developed models of the BarrettHand described in the previous sections (kinematics, dynamics, object and contact) have been implemented in this software application. In the figure, it can be seen the 3D virtual environment and the temporal evolution of the main robotic variables. In the upper part of the application there are some control buttons which represent the most important options of the simulator: *Kinematics,* to move the robotic hand using forward (joint values) and inverse kinematics (Cartesian values); *Path Planning*, to execute several types of trajectories specifying the end point; *Dynamics*, to compute all the contact forces and joint torques of the virtual BarrettHand; and *Objects*, to import 3D models in the virtual environment using VRML files.

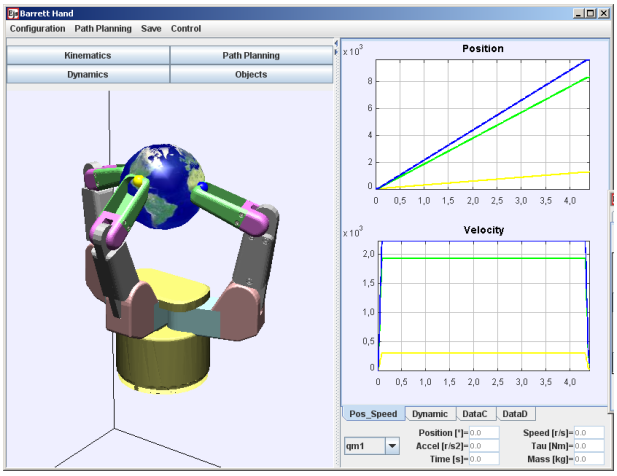

Figure 5. Interface developed for the BarrettHand simulation

This section describes the simulation of the breakaway mechanism in order to verify the correct behavior of the developed models. In addition, a comparison with the real movement of the BarrettHand is done (Fig. 6). A trapezoidal trajectory of the motors' velocities has been used as input trajectory for the models. This trajectory is composed of three phases: constant acceleration, constant maximum velocity and constant desacceleration. To compute the contact forces, a 3D cube has been imported into the virtual environment (Fig. 6).

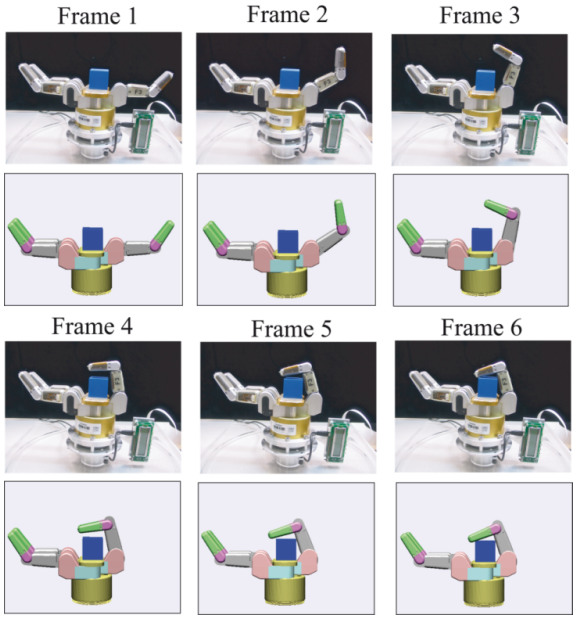

Figure 6. Comparison between the real and the simulated trajectories of the BarrettHand during a grasping task with breakaway activation.

In this experiment, finger 3 is closed by moving the finger motor in the range [0, 17500 counts]. This trajectory involves a

140º rotation over joint 32 (inner phalanx) and a 98º rotation over joint 33 (outer phalanx). When the object contacts the inner phalanx, a torque is generated over joint 32 (see Fig. 7 and Fig. 8). This torque increases linearly until it reaches the threshold τ_{break}= 1.26 N·m (equivalent to a force of 18 N), when the breakaway is activated. Then, joint 32 is blocked and the transmission relation of joint 33 changes according to (10). Fig. 7 shows the temporal evolution of the angle, velocity, acceleration and torque of joints 2 and 3 (j2, j3) and how they change when the breakaway is activated at t= 1.8 s (Fig. 8). As it shown in Fig. 7, there is a discontinuity in the velocity of joint 3 caused by the contact of the inner phalanx with the object. This discontinuity is treated as a constraint in the simulation system which sets the velocity to zero in case of contact.

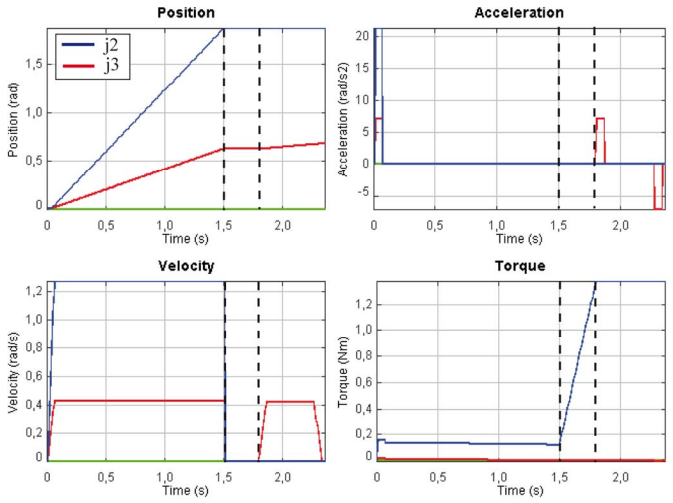

Figure 7. Temporal evolution of the robotic variables in the grasping task.

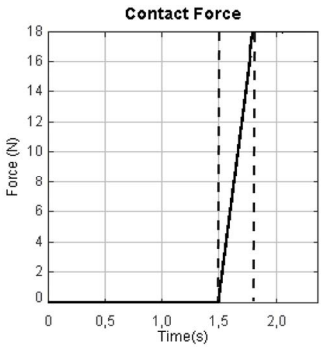

Figure 8. Contact force over the inner phalanx in the grasping task.

VI. CONCLUSIONS

This paper presents a complete model of a multi-fingered under-actuated robotic hand for grasping tasks. This model contains four main components: kinematics, dynamics, object representation and contact force transmission. The kinematics and dynamics models use the Denavit-Hartenberg formulation and the Lagrange-Euler algorithm adapted to two typical particularities of many under-actuated hands: the coupling between the finger joints and the breakaway mechanism. The object representation approximates the object surface by a triangle mesh obtained by a tessellation algorithm from a standard VRML file. The contact model determinates the forces that are applied to the object by the robotic hand. Finally, a complete software simulation has been implemented in order to verify the accuracy of the developed models.

As future works, authors will improve the contact model by computing the normal force from all the triangles which compose the contact area instead of approximating the area into one contact point. Friction forces should also be added because now only a frictionless model is implemented. In addition, the simulator will be proved with more complex real manipulation tasks.

ACKNOWLEDGMENT

This work is supported by the Spanish Ministries of Education, Science and Innovation through the research project DPI2008-02647 ('Intelligent Manipulation through Haptic Perception and Visual Servoing by Using an Articulated Structure situated over a Robotic Manipulator') and the grant AP2005-1458.

REFERENCES

[1] Easy Rob Simulation System, "Easy-Rob 3D", in: www.easy-rob.de/ , 2005.

[2] R. Prado, and R. Suárez, "Modelado y control de una mano mecánica," in Proceedings of 1ª Jornada de Recerca en Automática. Visió i Robótica, 2004, pp. 299-306.

[3] A. T. Miller, and P. K. Allen, "GraspIt! A versatile simulator for robotic grasping," IEEE Robot. Autom. Mag, vol. 11 (4), 2004, pp. 110-122.

[4] C. Melchiorri, and M. Kaneko, "Robot hands," in Handbook of Robotics, B. Siciliano and O. Khatib, Eds. Berlin: Springer, 2008, pp. 345-360.

[5] L. Birglen, T. Laliberté, and C. Gosselin, "Underactuated robotic hands", in Springer Tracts in Advanced Robotics, B. Siciliano, O. Khatib and F. Groen, Eds. Berlin: Springer, 2008.

[6] W. Townsend, "The BarrettHand grasper – programmably flexible part handling and assembly," Ind. Rob., vol. 27 (3), 2000, pp. 181-188.

[7] A. Peer, S. Einenkel, and M. Buss, "Multi-fingered telemanipulation – mapping of a human hand to a three finger gripper," in Proceedings of the 17th IEEE International Symposium on Robot and Human Interactive Communication, 2008, pp. 465-470.

[8] J. Zamora-Esquivel, and E. Bayro-Corrochano, "Geometric tehcniques for visually guided grasping," in Proceedings of the IEEE International Conference on Robotics and Automation, 2008, pp. 1312-1317.

[9] F. Esquembre, "Easy Java Simulations: a software tool to create scientific simulations in Java,", Comput. Phys. Commun., vol. 156, 2004, pp 199-204.

[10] C. A. Jara, "EjsRL Library", in: www.aurova.ua.es/rcv/ , 2010.

[11] J: Denavit, and R. S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices,". ASME J. Appl. Mech., vol. 22,1955, pp. 215-221.

[12] K. S. Fu, R. C. Gonzalez, and C. S. Lee, Robotics: Control, Sensing, Vision and Intelligence, McGraw-Hill, New York, USA, 1987.

[13] R. Carey, and G. Bell, The Annotated VRML 2.0 Reference Manual, Addisson-Wesley Professional, Essex, UK, 1997.

[14] G. Zachmann, "Collision detection by dynamically aligned DOP-trees," Proceedings of the Virtual Reality Annual International Symposium (VRAIS), 1998, pp. 90-97.

[15] T. Moller, "A fast triangle-triangle intersection test," J. Grap. Tools, vol. 2 (2), 1997, pp. 25-30.