Advances in Robotics & Automation
ISSN: 2168-9695

Mattar, Adv Robot Autom 2013, 2:2
DOI: 10.4172/2168-9695.1000109

Research Article Open Access

e_GRASP: Robotic Hand Modeling and Simulation Environment

Ebrahim Mattar*

College of Engineering, University of Bahrain, Kingdom of Bahrain

Abstract

Modeling and simulation of robotics hands are significant topics that have been looked into by many robotics Specialists and programming experts. This is due to a demand to build a friendly platform for analyzing proposed hand design and movements, earlier to hand physical construction. For meeting such demands, a dexterous robotic hand software simulator was synthesized. The developed code is dexterity characterized robotic hand modeling and simulation software environment. The simulator was developed for robotics hands research purposes. This manuscript is presenting a brief documentation of such a modeling and simulating environment for simulating dynamic movements of multi-finger robotics hand. The environment is named as the *e*_GRASP. To make use of other supporting environments, *e*_GRASP is a Mat lab based simulator, with a quite large number of linked functionalities and routines that helps in simulating hand movements in a defined 3D space. *e*_GRASP was built after a number of years of experience while dealing with robot hands, hence it is a comprehensive Mat lab based Toolbox that makes use of other Mat lab defined Toolboxes. *e*_GRASP can also be interfaced to real-time hand control, with an ability to be linked with even higher levels of hierarchy. This includes Mat lab AI Tools, optimization, as considered useful toolboxes for dexterous hands for grasping and manipulation.

Keywords: Dexterous robotic manipulation; Modeling; Real-time simulation; Control; Matlab

Introduction

Literature and related studies

There are number of efforts to build robotics system modeling and simulation environments using Mat lab. For example, in reference [1], a Mat lab Toolbox for the iRobot Create (MTIC) was reported. It was mentioned that, "The toolbox replaces the native low level numerical commands, with a set of high level, intuitive, Mat lab functions (aka "wrappers")" [1]. In terms of modeling complexity, furthermore, Shaoqiang et al. [2] have mentioned that, "Biped robots are often treated as inverted pendulums for its simple structure. But modeling of robot and other complex machines is a time-consuming procedure. A new method of modeling and simulation of robot based on SimMechanics is proposed", Shaoqiang et al. [2]. Over number of years, Corke [3], has been working towards a MATLAB based robotic arm simulation. Hence, Corke [3], has introduced the well-known book for MATLAB environment with associated libraries. This is known as "Robotics Toolbox for Matlab", Corke [3]. The environment was a very useful tool for modeling robotic arms. It was also an easy methodology and coding for modeling and simulation different robotics arm structures. Jambak, et al. [4], mentioned the importance and paramount of Robot Simulation Software nowadays. This is to increase the accuracy and efficiency of industrial robot. They adopted a project using "virtual reality interface design methodology and utilizes MATLAB/Simulink and V-Realm Builder as the tools". They also mentioned that, "a robot model has been developed and a Robot Simulation Software life cycle has been implemented", Jambak et al. [4].

Furthermore, in reference to Olivier [5], Cyberbotics, it was reported that, Webots TM has a number of essential features intended to make such simulation tool both easy to use and powerful:

- Models and simulates any mobile robot, including wheeled, legged and flying robots.

- Includes a complete library of sensors and actuators.

- Lets you program the robots in (*C*, *C++* and *Java*), or from third party software through TCP/IP.

- Transfers controllers to real mobile robots, including Aibo®, Lego®, Mindstorms®, Khepera®, Koala®, and Hemisson®.

- Uses the ODE (*Open Dynamics Engine*) library for accurate physics simulation.

- Creates AVI or MPEG simulation movies for web and public presentations.

- Includes many examples with controller source code and models of commercially available robots.

- Lets you simulate multi-agent systems, with global and local communication facilities", Olivier [5].

Gourdeau [6] has indicated that, "Using an object-oriented programming approach, ROBOOP, a robotic manipulator simulation package which is both platform and vendor independent, compares favorably against a package requiring similar coding effort", Gourdeau [6]. In fact, he also indicated that, the performance tests show that with ROBOOP, the routine (with a class), inverse dynamics of a 6-DOF robot was made faster, can be computed and simulated in less than (5 ms) with a Pentium (100 MHz) computer.

Ramasamy and Arshad, [7] have both developed a robotic hand simulator. They indicated that "This robotic hand simulation is divided into three main parts. The main objective is to design a three dimensional graphic of a robotic hand and its movement animation

***Corresponding author:** Ebrahim Mattar, College of Engineering, University of Bahrain 32038, Kingdom of Bahrain, E-mail: ebmattar@ieee.org

Received August 21, 2013; **Accepted** October 14, 2013; **Published** October 16, 2013

Citation: Mattar E (2013) e_GRASP: Robotic Hand Modeling and Simulation Environment. Adv Robot Autom 2: 109. doi: 10.4172/2168-9695.1000109

that imitates the movement of a human hand". They used the graphic design as a foundation to find the kinematics and dynamic properties of the robotic hand.

Furthermore Miller and Allen [8] have both presented the GRASPIT! In reality, GRASPIT! is a dexterous robotic hand simulation code. It was reported that the work focus of the grasp analysis, has been on force-closure grasps, which are useful for pick-and-place type tasks. Miller and Allen in [8], have also reported that, "This work discusses the different types of world elements and the general robot definition, and presented the robot library".

Miller et al [9]. have furthermore developed a simulation platform, where they focused on (Design/methodology/approach), as they are unlike other simulation systems. However, the simulation system was built specifically to analyze grasps by robotics hand. Miller et al. [9] also reported that, "It can import a wide variety of robot designs by using standard descriptions of the kinematics and link geometries. Various components support the analysis of grasps, visualization of results, dynamic simulation of grasping tasks, and grasp planning".

Jagdish et al. [10], have proposed and presented in their chapter, a fast as well as automatic hand gesture detection and recognition system. It was worked towards a reorganization of a hand gesture, hence an appropriate command is to be sent to the robot hand. Jagdish et al. [10], also mentioned that, once robot receives a command, it does a pre-defined work and keeps doing until a new command arrives.

Corrales et al. [11], have developed a kinematic, dynamic and contact models of a three-fingered robotic hand. The hand name was (BarrettHand), as this to obtain a complete description of the system which is required for manipulation tasks and simulation. Corrales et al. [11] have also stated that, "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 3 accuracy of the proposed models with regard to the real physic system".

Gourret et al. [12], have addressed the problem of simulating deformations between objects and the hand of a synthetic character during a hand grasping progression. Hence, a numerical method based on finite element theory was developed and used to allow considering into account active forces of the each finger on the object and the reactive forces of the object on the fingers. Magnus [13], stated that, "before the implementation of the controller was made on the real hand it was tested and development on a simulation created in MATLAB/simulink with help from a graphic physics engine called GraspIt!".

In this context, Magnus [13], have also looked into movements of the hand finger, and found how this is affected by the force from a leaf spring and a tendon that bends the finger. They also exposed the hand fingers to contact forces. They used these results and all these components to create models that are used in the simulation, hence to make the finger perform accurately.

Tarmizi et al. [14], as justified by increasing demands robotics hands both modeling and simulation, and due to the increasingly gaining importance amongst researchers for industrial and medical applications. Tarmizi et al. also stated that, "a multifinger robot hand with five fingers is modeled and simulated for grasping task. This was done using a CAD tool known as SOLID WORKS, and an analytical tool known as SimMechanics of MATLAB". In fact, obtained simulation results, indicated that the improved performance of grasping functions for robotics hands.

Ramasamy and Arshad [7], used animation techniques, hence, this to facilities designs and three dimensional graphic of an robotic hand, and its movement animation that imitates movements of the human hand. Such graphic designs, are hence, used as a foundation to find the kinematics and dynamic properties of the robotic hand. The end result is a robotic hand simulation that comes with analyses of the kinematics and dynamic properties".

Boughdiri et al. [15], have mentioned that there are numerous simulation results that shown, the derived dynamic model can predict the motion of the multi-fingered hand in free motion without holding any object. Additionally they have developed dynamic model that used and led to decoupling dynamic characteristics, by which the control of different parts of the system are simulated. They have reported a model-based computed torque technique, as for tracking control of the multi-fingered hand.

Ohol and Kajale [16] have looked into a number of issues related to robotic hand simulation. This includes; required task for the robots, as this becoming more complicated. In addition, handling of objects with various properties, e.g. material, size, mass, and physical interaction between the finger and an object. Ohol and Kajale [16] have stated in their research that, "Design procedure, solid modeling, force analysis and simulation have been discussed for further dynamic analysis towards confirmation of the viability".

Chan and Yun-Hui [17], have further looked into the issue of a dynamic simulator, that can helps and facilitates developments and applications of a multi-fingered robot hand. They indicated that, the existing dynamic simulators cannot effectively simulate dexterous manipulation of a multi-fingered robot hand. This is due to the lack of capability to cope with frequent changes in contact constraints and grasping configurations as well as impulsive collision occurring during manipulation. Chan and Yun-Hui, [17] also mentioned that, "We propose a unified framework to model free motions, collisions, and different contact motions including sticking, rolling, and sliding". Hence they proposed a innovative transition model for handling transitions between these contact motions. Finally, a 3D dynamic simulator has been developed to simulate dexterous robotic manipulation tasks, while involving combination of different contacts. That was based on a unified dynamic model. Simulation results indicated and confirmed the validity of the dynamic model and the simulator efficiency.

Main article contribution

In an effort to synthesis a dexterous multi-finger robotic hand MODELING and SIMULATION environment, this research framework was thoroughly focused towards the utilization and employment of Matlab-based platform coding, for a kinematics and dynamic simulation of an (n) number of fingers hand, with (m) number of joint within each finger dexterous robot hand.

The coding software is referred to as the "*e*_Grasp". *e*_Grasp was coded using Matlab functions and routine, with some associated Matlab libraries. The main motivation for selecting Matlab as the programming environment, this due to the availability of extended libraries and the ability to link Matlab with other external libraries. The simulator was built to be even supportive for complicated model-based control algorithms for hand-object movements. Examples of which include the Computed Torque approach, and adaptive control. The simulator was also linked with MATLAB AI tools, as an attempt to make use of available AI techniques for implementing intelligent hand manipulation.

Manuscript organization

Within this manuscript, we shall be introducing *e*_Grasp simulator, where the main features of such modeling and simulation space are presented. We shall also show how this environment is linked to other internal Matlab main Toolboxes and external C++ libraries. Particularly, this manuscript is presenting a long term research framework, which is actually focused for dexterous robotics manipulation that have gained experience and achieved over a number of years, where it was totally dedicated towards building a robotic hand modeling and simulation environment. The manuscript was divided into six sections. Section 3 gives a brief introduction and literatures related to this research theme. In Section 4 we present he simulator integrated structure and blocks. Robotics hand model building, as in relation to hands kinematics and dynamics, are therefore introduced in Section 5. Section 6 gives the simulator HIERARCHCY and DATA_STRUCTURE. This focuses on coding features for dexterous hand simulation. In Section 7 we show few *e*_Grasp simulation results. Finally, in Section 8, we provide conclusions and draw few concluding remakes.

Simulator Integrated Structure and Blocks

Multi finger robotics hands do represent complicated dynamic and interrelated closed chain kinematics system. Therefore, modeling and simulation of such systems, always represent challenging tasks. This is illustrated in Figure 1. Here we show the top level building blocks of the *e*_Grasp simulator hierarchy. It composes of five building units. The first is the top level (which includes the hand motion planning and higher USER commands), within such environment task definition is developed. In this top unit an appropriate supervisory software has been developed to facilitate the communication with the various functions of the hand. Typically the user can state the needed motion parameters, like velocities and positions. The second level of hierarchy, is the controllers array box; motion and force control hardware aspects are mainly located within this unit. The third level of hierarchy is dedicated towards the mechanical hand and the drive box used for actuation. Finally, the last level of hierarchy is dedicated towards the level hand motion sensing and instrumentations. Within Section 5, the theme of hand design will be further expanded; hence, kinematic and dynamic modeling matters are therefore discussed in further details.

The e_GRASP has developed its awareness, as a result of the continuous demands to look into the most challenges effort, which is modeling and simulating dexterous robotic hands. Most computer controlled systems used for robotics control, have distributed simulators, which are dedicated to specific tasks. For a six degrees of freedom arm, six simulators they are usually dedicated for the production of digitally controlled motion, hence parallel motion of all the DOF is achieved. All of these DOF are then managed by a single upper simulator. Multi-simulators are unusual distributed simulators which may be utilized to share the computational load for the same task, for providing redundancy in computation, or for sharing the multi axis controllability load in a system.

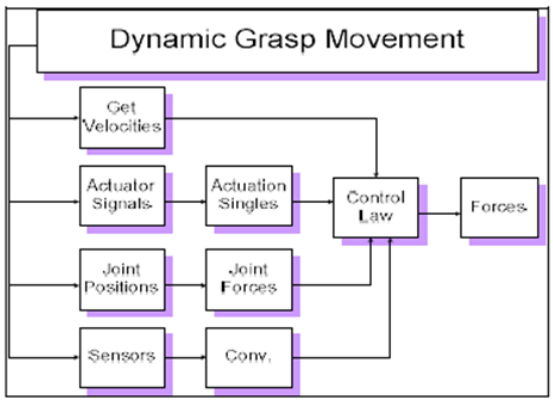

Figure 1: Hand simulator hierarchy and DATA_STRUCTURE, made it an easy procedure for interfacing different hand blocks.

For multi-fingered robot hands, a few dedicated simulation structures have been used in practice. For multi-fingered robot hands, and due to the large number of actuators to be controlled by a supervisory software, the design adopted here comprises of (n) linked motion simulators, simulating the digital control, for hand tendon displacement control. Furthermore, the simulator is to consider control of hand distal joints, as each distal joint in the hand is also digitally force controlled.

Due to the large number of actuators and the need to have a user friendly software interface, a TOP SUPERVISORY code has been written for the purpose of hand programming and coordinating the various units over the hand entire system. Smooth fingertip Cartesian motion is an essential capability for robot hands in general, therefore for achieving an (n) path Cartesian point of fingertip motion, the motion is required to be planned, such that, fingertips pass near to defined via points without stopping. The *e*_Grasp hand simulator should be featuring the below listed features:

(i) Model robotic hand kinematics and dynamics.

(ii) Simulate closed hand-object chain kinematics.

(iii) Simulate constrained hand-object dynamics.

(iv) Perform joint-space inverse kinematics

(v) Plotting and graphing functionalities.

(vi) Linking the simulator to others MATALB toolboxes.

(vii) Optimization, and optimal force distribution.

(viii) Linking to external tools, and others C++ libraries.

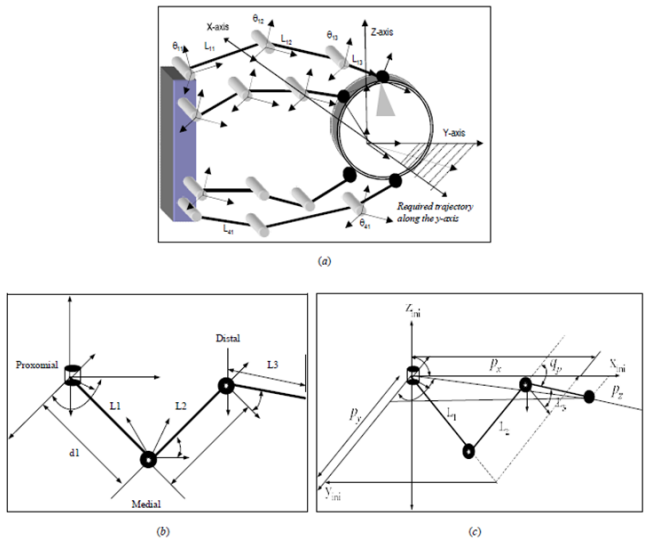

Figure 2: (a) We need to simulate a typical hand-object motion in 3D.That was made an easy task with e_Grasp, (b) A typical finger Kinematics and (c) Single finger geometry and related kinematics.

Building Simulator Models

Simulator library building: kinematics models

Hand Thumb Finger Model: In reference to Figure 2, and Table 1, $\left({}^{0}H_{3}\right)$ is formed by matrix multiplication of all the individual link matrices. While forming such a product, sub-results can be derived. This will be useful while solving hand inverse kinematics. By multiplying for simulation purposes, and for a simple situation of three joints in a finger, forward kinematics are expressed in terms of forward transformation matrix $\left({}^{0}H_{3}\right)$. Here the $\left({}^{0}H_{3}\right)$ entries have been calculated by:

$$R_1=\begin{Bmatrix}+(C_1C_{23})\\-(C_1C_{23})\\S_1\\C1(l_3C_{23}+l_2C_2+l_1)\end{Bmatrix}\qquad R_2=\begin{Bmatrix}+(S_1C_{23})\\-(S_1S_{23})\\-C_1\\S_1(l_3C_{23}+l_2C_2+l_1)\end{Bmatrix}\qquad(1)$$

$$R_3=\begin{Bmatrix}+S_{23}\\+C_{23}\\0\\(l_3C_{23}+l_2C_2)\end{Bmatrix}\qquad R_4=\begin{Bmatrix}0\\0\\0\\1\end{Bmatrix}$$

$$\left({}^{0}H_{3}\right)=\begin{pmatrix}C_1C_{23}&-C_1C_{23}&S_1&C_1(l_3C_{23}+l_2C_2+l_1)\\S_1C_{23}&-S_1S_{23}&-C_1&S_1(l_3C_{23}+l_2c_2+l_1)\\S_{23}&C_{23}&0&(l_3C_{23}+l_2c_2)\\0&0&0&1\end{pmatrix}\qquad(2)$$

While describing fingers kinematic, it is also essential to describe their kinematics with respect to fixed Cartesian frames in the hand palm. These frames are essential elements for a grasped object motion. To define the fingertips of the four fingers to a fixed Cartesian frame, a permanent palm frame has been placed at the base of the first finger sub-system. This is illustrated in Figure 2.

Other hand digits: Each of the fingers is related to the hand palm reference coordinate (attached to the root of the first finger) by transformations which are designated by *d*2, *d*3 and *d*4 for second, third, and fourth fingers respectively. Due to the position of the second finger in the hand palm, the fingertip location with respect to the hand palm is defined by the kinematics equation, as obtained by multiplying the finger transformation matrix $\left({}^{0}H_{3}\right)$ by the transformation matrix which represents the displacement of that finger from the origin of the palm coordinate frame. If displacement of the second finger is defined by $(-d_{x2}\ \ 0\ \ -d_2)$, the location various hand fingers with respect to a fixed hand frame are:

2nd finger:

$${}^{0}H_{3}=\begin{pmatrix}C_1C_{23}&-C_1C_{23}&S_1&(P_x-d_{x2})\\S_1C_{23}&-S_1S_{23}&-C_1&(P_y)\\S_{23}&C_{23}&0&(-d_2+P_z)\\0&0&0&1\end{pmatrix}\qquad(3)$$

3rd finger:

$${}^{0}H_{3}=\begin{pmatrix}C_1C_{23}&-C_1C_{23}&S_1&(P_x-d_{x2})\\S_1C_{23}&-S_1S_{23}&-C_1&(P_y)\\S_{23}&C_{23}&0&(P_z)\\0&0&0&1\end{pmatrix}\qquad(4)$$

4th finger:

$$H_f=\begin{pmatrix}(0.76C_2S_{23}+0.64S_{23})&(0.76C_1S_{23}+0.64C_{23})&0.76S_1&(0.7P_x+0.64P_z-d_4)\\S_1C_{23}&-S_1S_{23}&-C_1&(P_y)\\(-0.64C_2S_{23}+0.76S_{23})&(0.64C_2S_{23}+0.76S_{23})&-0.64S_1&(-0.64P_x+0.76P_z)\\0&0&0&1\end{pmatrix}\qquad(5)$$

Eq (2)-(5) do constitute the kinematics model of a (n) digits fingers hand. They also specify the computation of the position and orientation of frame 3 in reference to frame (0, *initial*), for each finger. These are the basic kinematics equations for simulating hand kinematics.

Simulator library building: (inverse kinematics)

In order to describe a finger position, reverse problem must be considered. Given a finger posture in terms of 3D components (P_x, P_y, P_z), what are the corresponding joint coordinates? In Section (5.1), the issue of computing position and orientation of the fingertip frame $\left({}^{0}H_{3}\right)$ was considered relative to a frame fixed at the hand palm. Here we are summing a given a set of joint angles of the finger. In this section, inverse kinematics is presented. Expressed another way, the inverse kinematics problem is given a frame or a homogeneous $\left({}^{0}H_{3}\right)$ matrix in the reachable subset, solve for the corresponding values of the joint variables that would result in similar $\left({}^{0}H_{3}\right)$ matrix numerical values. In contrast to forward kinematics, inverse kinematics problem does not have a general analytical solution. Given the desired position and orientation of the fingertip relative to the finger root, how we compute a set of joint angles which will achieve this desired result within a constrained kinematic system? The Cartesians upon which to base a decision vary, but a reasonable choice would be the closest solution. A finger will be considered kinematically solvable, if joint variables can be determined by an algorithm which allows one to determine all the sets of joint variables associated with a given position and orientation. We shall restrict the attention to closed form solution. In this context closed form means a solution method based on analytic expressions or on the solution of a polynomial. Within closed form solutions, two methods of obtaining a solution are distinguished. FIRST APPROACH: In

Finger revolute joints	Finger 1st joint proximal	Finger 2nd joint medial	Finger 3rd joint distal	---	---	Finger joint nA (distal)
θ_i	θ_1	θ_2	θ_3	---	---	θ_n
α_i	90°	0°	0°	---	---	0°
a_i	l_1	l_2	l_3	---	---	
d_i	0	l_1	0	---	---	0°
Motion range	$\begin{pmatrix}-45^\circ\\\leftrightarrow\\+45^\circ\end{pmatrix}$	$\begin{pmatrix}-45^\circ\\\leftrightarrow\\+45^\circ\end{pmatrix}$	$\begin{pmatrix}-45^\circ\\\leftrightarrow\\+45^\circ\end{pmatrix}$	---	---	$\begin{pmatrix}-45^\circ\\\leftrightarrow\\+45^\circ\end{pmatrix}$

Table 1: Hand Kinematics and interrelated parameters.

reference to eq (2), and frame assignment given in Figure 2, we desire a solution for joint space vector $\Theta^{T}_{i}=(\theta_{1i}\ \theta_{2i}\ \theta_{3i})$ given a Cartesian numeric coordinate of a given fingertip posture. Equate ${}^{0}A_{3}$ with numeric matrix, gives:

$$\begin{pmatrix} \chi_{11} & \chi_{12} & \chi_{13} & \chi_{14} \\ \chi_{21} & \chi_{22} & \chi_{23} & \chi_{24} \\ \chi_{31} & \chi_{32} & \chi_{33} & \chi_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \eta_x & o_x & a_x & P_x \\ \eta_y & o_y & a_y & P_y \\ \eta_z & o_z & a_z & Pz \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (6)$$

This result in:

$$P_x=\left(C_1\left(l_3C_{23}+l_2C_2+l_1\right)\right)\ \ and\ \ Py=\left(S_1\left(l_3C_{23}+l_2C_2+l_1\right)\right)$$

$$\left(P_y/P_x\right)=\left(S_1/C_1\right) \quad (7)$$

$$\Theta_1=\tan^{-1}\left(P_y/P_x\right) \quad (8)$$

Likewise using $P_z=\left(l_3S_{23}+l_2C_2\right)$ divide by $\cos(\theta_2)$ This results in:

$$\left(P_z/C_2\right)=\left(L_3S_{23}+l_2S_2/C_2\right)\ \ and\ \ \tan\left(\theta_2\right)=\left(P_z-l_3S_{23}/l_2C_2\right) \quad (9)$$

Searching for a suitable value of $\cos(\theta_2)$ in Equ (9):

$$P_x=\left(l_3C_1O_z+C_1l_2C_2+C_1l_1\right)$$

$$l_2C_2=\left(\left(P_x/C_1\right)-l_3O_z-l_1\right)$$

$$\theta_2=\tan^{-1}\left(\left(P_z-l_3n_z\right)/\left(P_x/C_1\right)-l_3O_z-l_1\right) \quad (10)$$

$$\theta_3=\tan^{-1}\left(n_z/O_z\right)-\theta_2 \quad (11)$$

The initial solution is that which corresponds to a known orientation vectors. SECOND APPROACH: A geometric approach for solving the inverse kinematic problem is a technique of decomposing the spatial geometry of the finger into several plane geometry problems. Joint space vectors is solved for, using tools of plane geometry. Figure 2b also displays the kinematics and geometric configuration of one finger. Applying rules of geometry to one finger shape, this results in the following equations describing the joint space vector in terms of the given (P_x) (P_y) (P_z) fingertip vector. From Figure 2c, and for the triangle as due to finger geometry:

$$\tan\left(\theta_1\right)=\left(P_y/P_x\right)\ \ and\ \ \left(\theta_1\right)=\tan^{-1}\left(P_y/P_x\right) \quad (12)$$

Furthermore, from the figure:

$$\left(P_q=\mathrm{P_r}-l_1\right)\ \ and\ \ P_t=\sqrt{\left(P_z^2+P_q^2\right)}$$

$$\theta_z=\tan^{-1}\left(P_z/P_q\right)\qquad W_a=C^{-1}\left(l_3^2-P_t^2-l_2^2/-2P_tl_2\right)$$

Having found θ_z, then $\left(\theta_z-W_a\right):\theta_3=180^{\circ}-\cos^{-1}\left(\left(P_t^2-l_2^2-l_3^2\right)/2l_2l_3\right)$ (13)

Motion Finger. Using the geometric model of the finger for the simulator is an easy and quick approach to obtaining the joint space vector. One drawback of the geometric method is that: at a definite finger posture the geometric configuration of the finger fails to give a solution. Refer to Figure 1 to observe that certain configurations of the fingertip where the solution is not well defined, hence multiple solutions may appear. However as θ gets large enough, the fingertip location P_t goes to a position with respect to (0) and T_r angles where the cosine rule used in eq (13) is not applicable. To achieve realistic finger solutions, hand simulator source code has been designed in such a way as, to get the inverse kinematic for the finger model using the geometric approach. Subsequently, once joint space vector are evaluated for, orientation vectors are evaluated, as they constitute columns of the well-known finger Jacobian matrix of J_θ We shall use two known methods in literature to find the inverse kinematics geometric and algebraic approaches of solutions fail to give definite solution.

Simulator library building: building hand Jacobian

The Jacobian matrix plays important role in hand simulation. Differential changes in finger end locations, are caused by as results of joint differential changes. The number of rows in a Jacobian is determined by the number of degrees of freedom in Cartesian space required by the task which is, in turn, determined by the degrees of mobility of a task. In fact, the JACOBIAN is a multi-dimensional form of the derivative of a function of several variables. Since the Jacobian is a differential formula of a matrix, hence it is possible to obtain elements of that matrix using the CHAIN FORMULA as far as the functions of independent variables. For the finger Jacobian J_θ there are a number of techniques which have been developed for the Jacobian calculation by: Waldron Algorithm, Renand Algorithm, Paul's Algorithm, [18,19]. Location of a fingertip is characterized by relative positioning of frame f_{i+1} attached to the fingertips, with respect to a frame f_0. In addition, a fingertip velocity is always a vector in the tangent space of the frame space, which is related by the Jacobian matrix J_θ. Such framings are illustrated in details in Figure 3.

Simulator library building: jacobian singularities avoidance

Due to the fact the Jacobian is a position dependent matrix, in particular finger configurations, the matrix becomes not full of rank. When this occurs the Jacobian rows and column vectors are linearly dependent, thus do not span the $\in\Re^{(3\times1)}$ vector space of X. Therefore, there exists at least one direction in which the fingertip cannot be moved no matter how the joint velocities θ_1, through θ_3 are chosen. Here we shall introduce the derivation of a finger Jacobian matrix in terms of D-H matrices. In Eq (5), the Jacobian is defined with respect to the reference frame, frame (0) at the finger root. However, for our purpose it has been assumed that the points of contact are stationary on the grasped object and another algorithm can be implemented based upon the kinematic model of the finger. Such algorithm is based on the method shown by Paul [19]. From the time when all joints are revolute, the three columns of the matrix are given by:

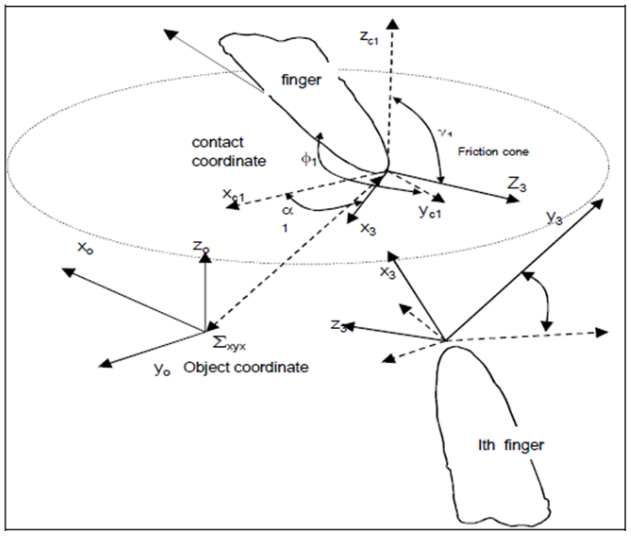

Figure 3: Simulator deals with diverse classes of fingertip contact models. The default type, is the (frictional point of contact).

$$\left.\begin{aligned} A^3 d_i &= \left(\left(-n_x P_y\right) i \left(-o_x P_y + o_y P_x\right) j \left(-a_x P_y + a_y P_x\right) k\right) \\ A^3 \delta_i &= \left(n_z i \;\; o_z j \;\; a_z k\right) \end{aligned}\right\}$$

While using this algorithm, it is necessary to evaluate the orientation vectors of a finger posture, which are assumed to be known by using the geometric finger model and the forward transformation matrix. Paul's algorithm can be used to calculate the Jacobian of an (n) degrees of freedom arm, where each degree of freedom corresponds to one joint. Thus, each column in the Jacobian is a vector which describes the differential motion of a joint. A revolute joint has a differential rotation around the axis of rotation Z axis:

$$\left.\begin{aligned} \left(\partial P_x / \partial \theta_n\right) &= J_{1n} = \left({}^{n-1}n_y \; {}^{n-1}P_x - {}^{n-1}n_x \; {}^{n-1}P_y\right) \\ \left(\partial P_y / \partial \theta_n\right) &= J_{2n} = \left({}^{n-1}o_y \; {}^{n-1}P_x - {}^{n-1}o_x \; {}^{n-1}P_y\right) \\ \left(\partial P_z / \partial \theta_n\right) &= J_{3n} = \left({}^{n-1}a_y \; {}^{n-1}P_x - {}^{n-1}a_x \; {}^{n-1}P_y\right) \\ \left(\partial \delta_x / \partial \theta_n\right) &= J_{4n} = {}^{n-1}n_z \;\; \left(\partial \delta_y / \partial \theta_n\right) = J_{5n} = {}^{n-1}o_z \;\; \left(\partial \delta_z / \partial \theta_n\right) = J_{6n} = {}^{n-1}a_z \end{aligned}\right\} \tag{15}$$

To relate a fingertip position to a set of joint angles of the fingers, one may make use of the forward solution and obtain the fingertip matrix $\left({}^0H_3\right)$. Paul [19] has presented a routine that relates differential changes in joint variables, to the differential changes in the tip position and orientation. This is summarized here as:

$${}^0H_3 = \begin{pmatrix} C_1C_{23} & -C_1C_{23} & S_1 & C_1\left(l_3C_{23} + l_2C_2 + l_1\right) \\ S_1C_{23} & -S_1S_{23} & -C_1 & S_1\left(l_3C_{23} + l_2c_2 + l_1\right) \\ S_{23} & C_{23} & 0 & \left(l_3S_{23} + l_2S_2\right) \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{16}$$

$${}^1H_3 = \begin{pmatrix} C_{23} & -S_{23} & 0 & \left(l_3C_{23} + l_2C_2\right) \\ C_{23} & -C_{23} & 0 & \left(l_3C_{23} + l_2c_2\right) \\ S_{23} & C_{23} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{17}$$

$${}^2H_3 = \begin{pmatrix} C_3 & -S_3 & 0 & \left(l_3C_3\right) \\ S_3 & -C_3 & 0 & \left(l_3S_3\right) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{18}$$

Further application of the algorithm, and using the relation defined by eq (17) gives: (1st) column of J_θ

$$\left.\begin{aligned} \partial x_1 &= \left(\psi S_1C_1C_{23} - S_1C_{23}\right) \\ \partial y_1 &= \left(\psi C_1S_1S_{23} + S_1C_1S_{23}\right) \\ \partial z_1 &= \left(-C_1^2\psi\right) - S_1^2\psi \\ \partial x_1 = 0 \quad \partial y_1 &= 0 \quad \partial z_1 = 0 \end{aligned}\right\} \tag{19}$$

$$\left.\begin{aligned} \partial x_1 &= S_{23} \\ \partial y_1 &= C_{23} \\ \partial z_1 &= 0 \end{aligned}\right\} \tag{20}$$

(2nd) column of J_θ

$$\left.\begin{aligned} \partial x_2 &= \left(S_{23}\left(l_3C_{23} + l_2C_2\right)\right) - C_{23}\left(l_3S_{23} + l_2S_2\right) \\ \partial y_2 &= \left(C_{23}\left(l_3C_{23} + l_2C_2\right)\right) - S_{23}\left(l_3S_{23} + l_2S_2\right) \\ \partial z_2 &= 0 \\ \partial z_2 = 0 \quad \partial y_2 &= 0 \quad \partial z_2 = 0 \end{aligned}\right\}$$

Leading to (2nd) column of J_θ to be $\left(\partial x_2 = l_2S_3 \quad \partial y_2 = l_3 + l_2C \quad \partial z_2 = 0\right)$ (21)

Third Column:

$$\left.\begin{aligned} \partial x_3 &= \left(\left(l_3C_3S_3 - l_3S_3C_3\right)\right) \quad \partial y_3 = l_3 \quad \partial z_3 = 0 \\ \delta x_3 &= 0 \;\; \delta y_3 = 0 \quad \delta z_3 = 1 \end{aligned}\right\} \tag{22}$$

Hence J_Θ is finally written as:

$$J_\Theta = \left(u_1 \; u_2 \; u_3\right), \quad u_1{}^T = \left(0 \;\; 0 \;\; -\psi\right), \quad u_2{}^T = \left(\left(l_2S_3\right) \;\; \left(l_3 + l_2C_3\right) \;\; 0\right), \quad u_3{}^T = \left(0 \;\; l_3 \;\; 0\right) \tag{23}$$

Eq (23) has been verified using Matlab software, and using velocity cross product (VCP) technique used by Fu et al. [20]. Since each finger has a 3-DOF mechanism responsible for producing translational differentials motion at the fingertip, the Jacobian matrix can be rewritten in terms of differentials motion of the fingertip. The methodology taken here is to divide the simulation environment into (n) main blocks of simulators. Respectively, individual blocks have been dedicated for the simulation and control of the (m) joints of one finger. This is shown in Figure 4.

Simulator Hierarchicy

Top level: simulator data structure

Configuration of hand simulating software was based on an

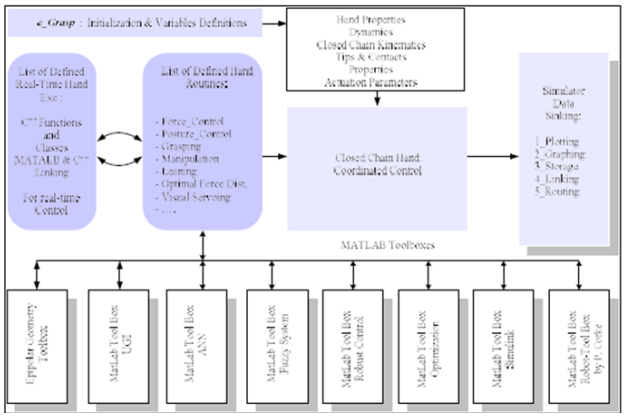

Figure 4: e_GRASP, a MULTI-FINGER robot hand comprehensive modeling and simulator MATLAB based environment.

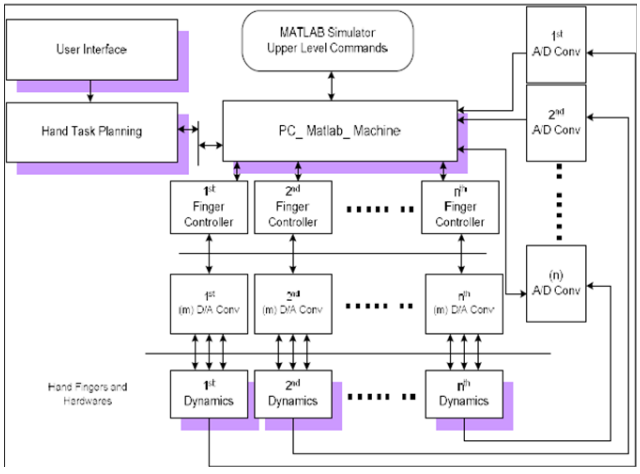

Figure 5: e_Grasp simulator cascaded levels. The simulator deals with both top level control laws, in addition to the lower level finger joints laws.

assumption, of using discrete time joint-space controllers. This is clearly illustrated in Figure 5. That was also based on using a model of MOTION CONTROLLER. There a number of available real-tome DSP controllers. Example of which is the national semiconductor product, (LM628) motion processor. For simulation purposes, there are up to ($n \times m$) motion processors that have been employed for the overall hand control. Individual single processors were dedicated control an individual DOF in the hand. Overall MatLab supervisory coding and software designing for the entire hand simulation, is to be run on high speed machine. However, it was also achieved while using an up-to-date high speed Laptops. High speed machine is needed, as this requires communicating with ($n \times m$) separated controllers blocks via ($n \times m$) processors hardware. From software viewpoint, simulating coordinated hand control, takes place at different computational levels. The hand hierarchical nature leads inherently to a bottom-up supervisory simulation design. Top level of concurrency, looks at entire hand simulation.

*e*_GRASP hand data structure and classes

At start, we shall define a finger data structure to contain a finger name. The "." operator, used in the case of Structure. Field tells mat lab to access the field named field in the structure.

```
% Creating Hand Data Structure And Classes..
Finger_Data.First='Finger_1';
Finger_1Data.Hand='Joint_1';
NameData.Last='Joint_m';
% Creating a HandData structure with a name field. HandData.
Finger=FingerData;
% Initializing rest of the hand structure …
Finger_1.Status=Finger_1_Data';
Finger_1Data.x_pos=10;
Finger_1Data.y_pos=40;
Finger_1Data.z_pos=-60;
Finger_1Data.q_1=-60°;
Finger_1Data.q_m=30°;
HandData.Fingers=linspace(60,30,45,…);
% View contents of the data_structure FingerData FingerData.
Name FingerData.deatils
% Operating on Elements of the Structure.
FingersData.X_Y_Z(3)=0;
FingerData.X_Y_Z  FingertData.X.First='First_Finger';  Finger_
Data.First
% Creating arrays of structures for each hand finger.
num_fingers=4; % depends on hand number of fingers
for i=1:num_Fingers
ClassData(i)=FingerData;
end
ClassData
ClassData(2)
```

Joints space and fingers simulation: non closed chain dynamics simulation

For simulating whole hand dynamics, either in open or closed loops, it is therefore important to have a model of the individual joint dynamics. This consists of a model of a single discrete time motion closed controller. A good model is an (n) bit controller, connected with a D/A converter, driving current amplification. The digital closed loop system also requires a simulation for measuring the individual joint displacement and position. In order to create an entire closed loop simulator model of the individual joints, and for describing the whole closed chain system, the following sub-sections will rather focus on the dynamics of a single joint controller, as this will be leading to a finger simulation, hence, leading to entire hand simulation.

Discrete time simulation of a single axis joint

Digital simulation of the lower level of individual joints, is discussed and simplified. That is because the control input for ith joint depends only on locally measured variables, not on the variables of the other joints. Moreover, computations are easy and do not involve solving the complicated derived nonlinear hand inverse dynamics. Contrary to this, for manipulation and exertion of forces points of view, an additional joint torque control has to be added, this is for an exertion of grasping forces throughout object motion. From literature, a typical PID controllers expressed by:

$$u(t) = k\left(e(t) + \frac{1}{\tau_i}\int_0^t e(t)\,dt + \tau_d\left(\frac{de(t)}{dt}\right)\right) \tag{24}$$

In eq (24), $u(t)$ is input to the joint controller, and $e(t)$ is the controller output. Digitizing eq (24), resulting in:

$$U(z) = \left(K_p + \frac{K_{ip}}{(1-z^{-1})} + Kd\left(1-z^{-1}\right)\right)E(z) \tag{25}$$

A simplified dynamical model of the joint with the electric actuator may be written as:

$$\left(J_a + J_m + n_i^2 J_L\right)\ddot{\theta}_i + \left(\frac{\left(B_a + B_m + k_b k_i\right)}{R}\right)\dot{\theta}_i = \left(\frac{k_i}{Rv_i}\right) - n_i d_i \tag{26}$$

$$\tau_a(s) = \left(J_T S^2 + B_T S\right)\theta(S)$$

In eq (26), the $\left(J_a + J_m + n_i^2 J_L\right)$ terms are effective inertia and effective damping coefficients of actuators shafts. This also includes the joint inertia. The joint position trajectory is calculated by a trajectory planning routine. This is passed to the digital control as number of pulses.

A typical joint space closed loop transfer function, relating an input to the joint, to its output, (block diagram shown in Figure 6, this is expressed by:

$$\left(\frac{\theta_m(S)}{\theta_d(S)}\right) = \left(\frac{\left(nk_\theta k_{PID} k_i R J_T\right)}{\left(S^2 + \left(\frac{RB_T + k_i k_b}{RJ_T}\right)S + \left(\frac{nk_\theta k_{PID} k_i}{RJ_T}\right)\right)}\right) \tag{28}$$

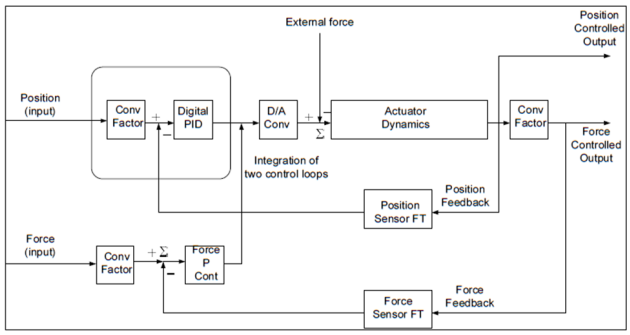

Figure 6: e_GRASP detailed discrete-time controller synthesis for a single joint closed loop control.

$$\frac{\theta_m(z)}{\theta_d(z)} = \left(\frac{z^2}{(z-\alpha_1)(z-\alpha_2)} \right) \tag{29}$$

Individual joint space controllers synthesis, was achieved using and supported by Matlab. Matlab environment, with its control toolbox synthesis and facilities, provided an aid to the controller testing and simulation. Typical design tools do include, Nyquist diagram, Modulus Frequency response, and time step response. For achieving a discreet time simulation within Matlab, the finger joints were being controlled digitally rather than in analog approach. Moving from continuous time domain to discrete time domain, Within (Z) domain, sampling time has an important role in determining the system stability. We start to do that by initially modeling the D/A converter in terms of sampling rate, $\left(\left(1-e^{-TS}\right) \Big/ s \right)$, hence the corresponding (Z) transform of the (D/A), including actuator, is evaluated. In addition, the computed discrete-time poles of eq (29) are, as well, function of the variable factor in the system, the scalar value of the proportional gain (k_p).

Simulation of joints-space controllers

Furthermore, in Figure 6 we show the bottom low-level controller block diagram, and the associated parameters. This is to avoid a joint dynamic behavior that is not desirable, as due to the amount of joint motion disturbances. Therefore we need to overcome any undesirable effects. The use of adjusted parameters control law have the effect of making the response behavior enhanced. The location of adjusting the full PID control parameters can be seen in enhancing fingers motion. As this will be rather simulated later; the choice of low proportional gain in addition to the integral and derivative has the effect of decreasing the corresponding (rise time) as compared with the result for a purely high proportional gain.

N Configuration hand motion: (creation of arrays, pointers, and records)

The simulator has a large data structure. This is to comprehend the complicated fingers motion. System points of contact motions are generated by specifications of (r) direction of motion. From an object to be moved within the work-space of the hand, principle axes of motions are defined in addition to Cartesian fingertips paths. If fingertips paths are specified as (H, K, L, U,.....), for the (n) fingers, then for (n) segments path, the Cartesian segment (localities in space), are passed and download to the entire hand controller coding. This is based on the use of such data structure:

$$\left.\begin{array}{ll} configuration & A=(1\ 2\ 3\ 4) \\ First\ finger & H=(H_1\ H_2\ H_3\ H_4) \\ Second\ finger & K=(K_1\ K_2\ K_3\ K_4) \\ Third\ finger & L=(L_1\ L_2\ L_3\ L_4) \\ Fourth\ finger & U=(U_1\ U_2\ U_3\ U_4) \end{array}\right\} \tag{30}$$

As indicated earlier, every finger segment vector consists of an ARRAY (Data_Type). This is for storage of finger joints positions and velocities. In accordance to the specified path coordinates, hand controller creates array of Matlab based records. At the ending phase of the (RECORDS) and (POINTERS) creation, the hand controller holds ($n \times n$) of records containing the Cartesian motion information of (n) segment paths for the individual fingers.

Simulation of joint force control

For simulating a joint torque dynamics and control, this requires a continuous measurement of the joint torque via some typical sensors. It is an essential for a joint torque sensor to be enclosed within a closed loop system. Furthermore, this is to be incorporated within the position control loop. While discussing the simulating aspects of a joint force control, the mechanism of transmission at joints do play an important role in the performance of the servo system itself. The problem of closing a joint force control feedback loop is the non-ideal transmission and drive system, i.e. presence of STICTION, COGGING, COULOMBIC friction, together with higher order resonance modes in the system. Furthermore, equivalent rotational stiffness do play an essential role in force control simulation. It relates the change in joint torque with changes in joint displacement and has to be computed for a rotational system. For a specific geometry of joint pulley and cantilever beam, a joint space rotational stiffness is mathematically expressed:

$$k_\theta = \gamma\left(c(\alpha) - s(\alpha)s(\beta)t(\alpha+\beta)\right)\chi \tag{31}$$

$$\gamma = \left(1 \Big/ \left(8s(\beta)s(\alpha+\beta)\right)\right) \tag{32}$$

and χ is a parameter representing the force sensor stiffness. For an ideal actuator characteristic, the system force control transfer function is given by:

$$\tau_a = \left(J_T\left(\frac{d^2\theta}{dt^2}\right) + B_T\left(\frac{d\theta}{dt}\right)\right) \tag{33}$$

Hence, for simulation of the force proportional control system, a typical control law is considered:

$$\tau_a = k_{ps}\Delta\tau \tag{34}$$

In eq (34), (k_{ps}) is the force loop controller gain, $\Delta\tau = (\tau_r(t) - \tau_f(t))$, where (τ_r) is the desired behavior, whereas the term (τ_f) is the finger actual joint torque. Furthermore, Eq (34) represents an ideal mechanism for controlling a joint torque. While examining a real typical physical construction of a force sensor, joint transmission system, and grasped object compliance, do add complexity to force control system. There is also variation in the dynamics of the finger due to its configuration dependence, however such changes are masked by the actuator dynamics. A grasped object compliance plays an important role in the response of the force controller. There exists the following linkage between (τ_f) and $(\Delta\theta)$, the angular displacement of the driving pulley. This is expressed by:

$$\tau_f = (\delta\Delta\theta) \tag{35}$$

In eq (35), δ is the compound rotational stiffness of the finger mechanical structure, as expressed in terms of: (i) sensor rotational stiffness, (ii) the grasp rigidity characteristics, (iii) and stiffness of the tendon. This is expressed by:

$$\left(1/\delta\right) = \left(\left(1/\upsilon\right) + \left(1/\nu_p\right) + \left(1/\eta\right)\right) \tag{36}$$

While considering eq (34) and eq (35), a relation between τ_f and τ_r is expressed by:

$$\left(\frac{\tau_f(s)}{\tau_r(s)}\right) = \left(\frac{(\delta k_p)}{J_a s^2 + B_a s + \delta k_i k_p}\right) \tag{37}$$

Eq (37) gives the torque response of the closed loop system is controllable by υ and k_p parameters. The simulation behavior of the designed force servo system, does not remain uniform even for constant gains k_p. This is due to the fact, that δ involves some varying quantities η and ν_p. For instance increases from a very stiff environment while it decreases for a compliant object surface. Additionally ν_p, tendon

stiffness, might change if a greater stretching force is applied to its ends. The behavior could be further simplified by taking into account that the robot hand has been categorized to be grasping a rigid object, hence η is having a very high value. v_p is quite a high quantity for the routed tendon due to the short length of the tendon and the use of high stiffness tendon characteristics. An equivalent rotational stiffness υ, by itself is a function of the physical construction of the sensor and its location in the finger. Having considered such assumptions, Eq (36) is totally dominated by an equivalent rotational stiffness. Here υ is a function of the sensor physical parameters. Finally, for real time simulation of the force closed loop system (feedback control), we transform the dynamics from time domain to frequency domain, i.e. relying on Laplace transform of the entire force closed loop control transfer function. This is expressed as:

Finally, for real time simulation of the force closed loop system (feedback control), we transform the dynamics from time domain to frequency domain, i.e. relying on Laplace transform of the entire force closed loop control transfer function. This is expressed as:

$$\left(\frac{F_0(S)}{F_r(S)}\right)=\left(\frac{\left(nk_{c\Theta}P_ik_i \middle/ RJ_T\right)}{S^2+\left(RB_T+k_ik_b/RJ_T\right)S+\left(nk_{\Theta}k_{c\Theta}P_ik_pk_i\right)\left(RJ_T\right)}\right) \quad (38)$$

In eq (38), p_i, is a conversion ratio. It relates the a conversion from a desired fingertip force, into a volts within the loop. The expressed relation of Eq (38), does indicate the changing parameters that do affect the force closed loop dynamics. Therefore, the built simulator environment is to take care of such parameters while simulating a typical a joint closed loop system.

Closed chain hand-object simulation: advanced control laws (computed torque method)

One of the great benefits of such a simulation instrument, is an ability to simulate robotics hands, under advanced control laws. Typical example is the Cartesian computed torque control law. In addition, other advanced control laws is the sliding mode control. In reality, such control methodologies are complicated, and do require massive computational requirements. However, they were made easy while using *e*_Grasp simulator. More results will be shown over the next Section. In addition, adaptive or even ANN based nonlinear control can also be simulated through the *e*_Grasp simulator. For (*m*) joints (in a finger), with (*n*) fingers robotic hand, the individual fingers are modeled by the time varying equation of motion:

$$A_i(\theta)\ddot{\theta}+B_i\left(\theta,\dot{\theta}\right)\dot{\theta}+C_i(\theta)\theta=\left(\tau_i+\sigma_i\right) \quad (39)$$

In eq (39), (θ) is a trajectory of finger joint in rad. $\left(\dot{\theta}\right)$ is joint speed trajectory in (rad/sec). $\left(\ddot{\theta}\right)$ is joint acceleration in (*rad/sec*2). $A_i(\theta), B_i\left(\theta,\dot{\theta}\right)$ *and* $c_i(\theta)$ are the hand concatenated dynamics. Additionally, (σ_i) is equivalent of externally added forces. Hand wrenches and forces are found by:

$$f_{hand}=-\left(G^{+1}M_b+\lambda\eta\right) \quad (40)$$

M_b the grasp dynamics, and G^{+1} *1* is the hand grip transform inverse. For a balanced motion, the resulting forces and moments of the entire closed chain dynamics is equal to zero. Hence, equating hand dynamics with object dynamics, this gives:

$$A_h{J_h}^{-1}\left(\dot{G}^T\dot{u}-J_h\ddot{q}_h\right)+B_h\dot{q}_h+C_h=\tau_h+{J_h}^T\dot{G}^{+1}\left(A_o\ddot{u}+B_o\dot{u}+C_o\right)-{J_h}^T\lambda\eta \quad (41)$$

Furthermore, in eq (40), f_{hand} is a set fingertips forces. Calculating fingertip forces using pseudo inverse of grip transform *G*. In Equ (41), $\lambda\eta$ is part of the force solution, and λ is an adjusting vector. Furthermore, the vector ${u^c}_d$ is the position and orientation of the grasped object. For non-slipping contacts, there is no change at contact points, i.e. $\left(\partial\Gamma_h \middle/ \partial t\right)=0$, though there might be rotational change at each point of contact. eq (41) also represents a typical and the entire hand-object dynamics. Hand fingertips forces, f_{hand} are playing major roles in such balancing equation. Defining a Cartesian based posture error (*e*) of an object in 3D as $\in\Re^{6\times1}$, as the error between a defined posture $\left({u^c}_a\right)$, and the real object posture $\left({u^c}_a\right)$ as $\left(e\cong{u^c}_a-{u^c}_a\right)$. Object-hand closed chain system is described in terms of hand joint-space torques $\left(\tau_h\right)$ joint torques and Euler dynamics. In Cartesian space, and for the three terms Cartesian based controller, this is expressed by:

$$\tau_h=\left(A_h{J^{-1}}_hX_h+T_{ex}\right) \quad (42)$$

$$T_{ex}=\left(B_k+C_k\right)+{J^T}_h\left(F_{cd}-Z_i\int_0^x\left(\phi_{cd}-\eta\lambda\right)\right),\quad X_h=\left(G^{+1}\Theta_a-J_h\Theta_k\right)\rightarrow\Re^{n\times1} \quad (43)$$

In eq (42), $X_h\in\Re^{12\times1}$ and expressed mathematically as in Equ (43). In addition, A_k, B_k *and* C_k are the entire hand augmented dynamics. F_{cd} is a commanded set of forces, $(\eta\lambda)$ is an adjustable term, J_h is hand Jacobain matrix. Each finger maps its joints torque to the object via the entire hand gasp *G* which is formulated as $G=\left(G_1\vdots G_2\vdots\rightarrow\vdots G_n\right)$, as grab sub-matrices $G_i\in\Re^{6\times3}$, for $i=\left(1\ 2\ \cdots n\right)$ are defined in terms of contact location by Eq (42):

$$G_i=\left(I_{3\times3} \middle/ \gamma_i\right) \quad (44)$$

In eq (44), (γ_i) are sub-matrices for contact configuration. They are performing a skew-matrix of position contact (γ_i) over the grasped object surface. In addition, hand fingertip force distribution depends completely on dually heavily computed matrices. The first is G^{+1} witch is an irregular matrix. The second is the hand Jacobian inverse matrix as the $\left({J^{-1}}_h\right)\cdot\left({J^{-1}}_h\right)$ is a large matrix, and it is a concatenated matrix, compromising all the fingers Jacobians, as $\left(J_h=diga\left(J_1\ J_2\ \cdots J_n\right)\right)$. Finally, eq (43) expresses a typical Cartesian based hand controller using PID, that will be simulated through the *e*_Grasp.

Result and Analysis

In particular, *e*_Grasp has been used successfully as a tool for motion and manipulation analysis of robotic multifinger hands. In this respect, for running this environment, this needs a Matlab environment. The code has been also tested lately on MATLAB Version 7.8.0.347 (R2009a). The simulator has been tested over a number of times. For validating the *e*_Grasp simulator potential, we shall present within this section few simulation results as related to a control for moving while grasping a grasp by a robot hand. The chosen hand is of four fingers, where each finger is having three rotational joints, i.e. $\left(n=4,m=3\right)$. The simulation environment gives the user an ability to select the hand configuration, dynamic and kinematics parameters, nature of fingertip contact, control sampling rate, law of controller, simulation time, in addition to others simulation parameters. There are large number of results to be shown, as a result of running the simulator, however, we shall show only few graphics results. In this respect, in Figure 7a, we

show the simulator graphical interface with other plotting graphics and the m-editor in Matlab. Furthermore, in Figure 7b we show part of simulator interfacing with other Matlab related toolboxes. In this respect, here we show the Matlab optimization toolbox being interfaced with e_Grasp simulator to compute optimal forces and torques needed by hand fingers to make an object motion.

For a demonstration, a 3D grasp movement is shown in Figure 8. A grasp object of a known dimension and weight was manipulated by movements of fingertips, while applying a suitable set of fingertips forces. The grasp was moved in periodic sinusoidal movement. Results shown that, simulating a 3D grasp movements was made an easy task while using the *e*_Grasp simulator. Dynamics of a grasped object can

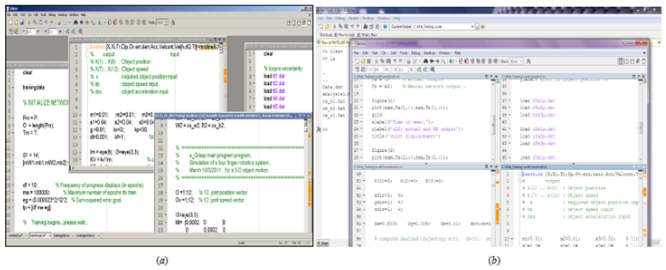

Figure 7: (a) The simulator: A screen shot for e_GRASP connection, (b) Part of simulator interfacing with Mat lab related tool boxes. Here we show the Mat Lab optimization toolbox to compute optimal forces and torques needed by hand fingers to achieve an object motion.

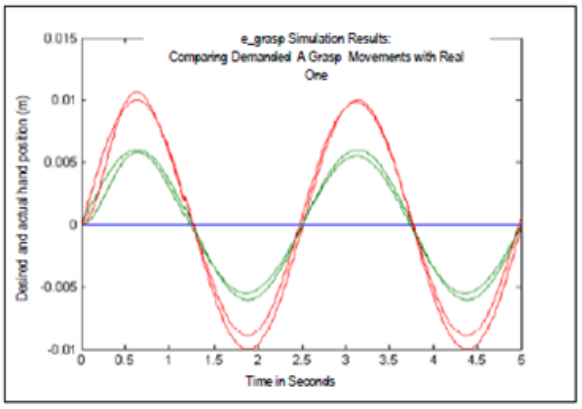

Figure 8: Creating a 3-D grasp movement is an easy task with e_GRASP.

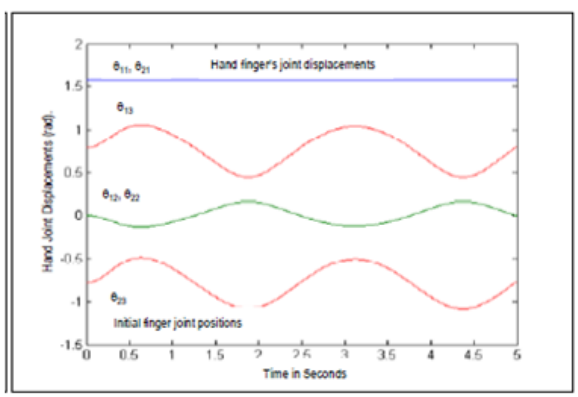

Figure 9: Due to hand motion a 1st finger closed loop simulation.

be changed while alternating both Cartesian controller parameters and joints parameters. This leads to make a grasp motion stable, oscillatory, faster, or slower. Furthermore, Figure 9 displays a finger joint-space closed loop displacement performance. The simulator has taken into account fingertip rotations during a course of a grasp motion. Furthermore, in Figure 10 we show a simulation of hand torques, while grasping an object in 3-D space. The figure shows how torques are computed in response to motion. The simulator was able to transform a grasp from an initial posture to another over a minimum time. For such a simulation, the simulation environment offers a full adjustment of the three terms PID controller parameters, hence they have been selected for minimum overshoot. A grasped object movements can be made much unstable or even with less overshoot over its movement path in 3D. In Figure 11 we show a typical simulator capabilities. The figure is showing the computed error while comparing artificial neural network output with actual joint-space output. The 2nd finger joint-space motion simulations are therefore realistic in terms of displacement and motion. Joint space torques are very realistic in terms of producing an adequate amount of torques for fingertip movements. In Figure 12, we also display an important ability of the *e*_Grasp, which the ability of linking its output to other Matlab toolboxes. Here we show a link between *e*_Grasp with Matlab fuzzy toolbox. The figure shows an adaptation of fuzzy membership functions for optimal forces

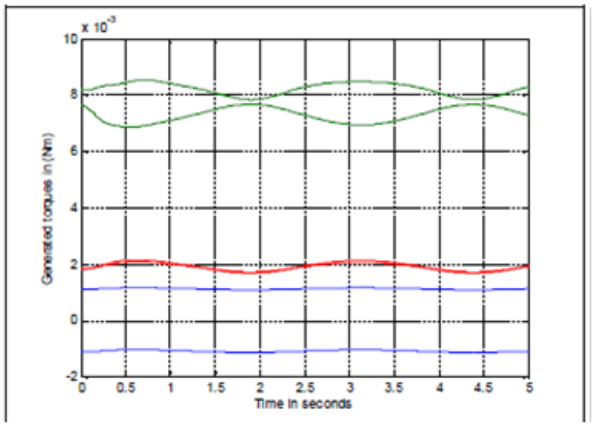

Figure 10: e-Grasp simulation of hand torques. Simulation includes plotting of individual torques while grasping in 3-D space.

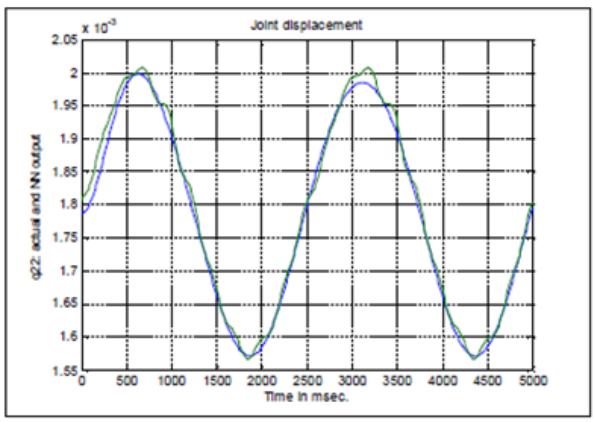

Figure 11: e-Grasp simulation capabilities: Plotting of individual joint error, while comparing ANN controlled output with actual joint output.

learning via fuzzy system. In addition, the package is also able to be linked with much advanced Matlab functionalities, which is the ANN tools. In addition to the basic functions, the simulation package can also provide more analysis of hand movement. For example, Figure 13 shows the ability to plot training patterns for hand ANN training. Such patterns were also used for learning and understanding the hand for optimization of fingertips contact forces.

Conclusions

Software simulation of an (n) DOF articulated robotic hand system is not an obvious task. Specifically, such simulators are truly needed for testing purposes, and for viewing hand performance before physical implementation. This manuscript has focused on a developed simulation software for supervising and controlling a fully described (kinematically and dynamically) (n) fingers robotic hand. The simulation environment was achieved via Matlab with a link to other available Matlab toolboxes. The developed simulator also has the ability to be linked internally to Matlab toolboxes, in addition, to be externally linked to other libraries, like (C^{++}), for a possible real-time hand control. The simulator even has the ability for linking high level commands to the low-level digital motion processor. The simulator has proved to be an effective way to look and view kinematics and dynamics models of robotic hand. These models are found to be essential elements for hand object dynamic simulation. Next stage, is to take the simulator further, with graphical interfaces and functionalities.

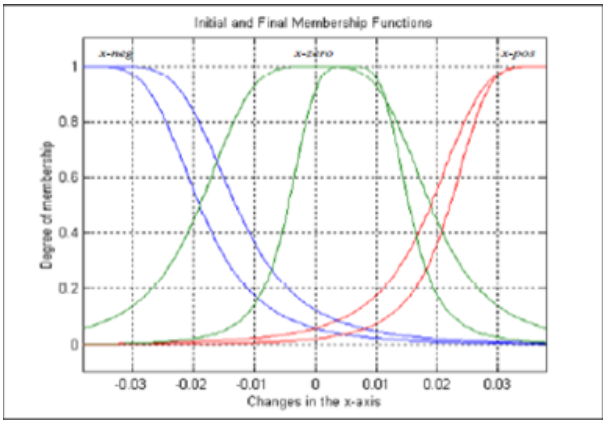

Figure 12: e_GRASP simulator capabilities: Hand learning simulator linkage to MATLAB/FUZZY Toolbox. The FUZZY Toolbox was used for learning of hand motion parameter.

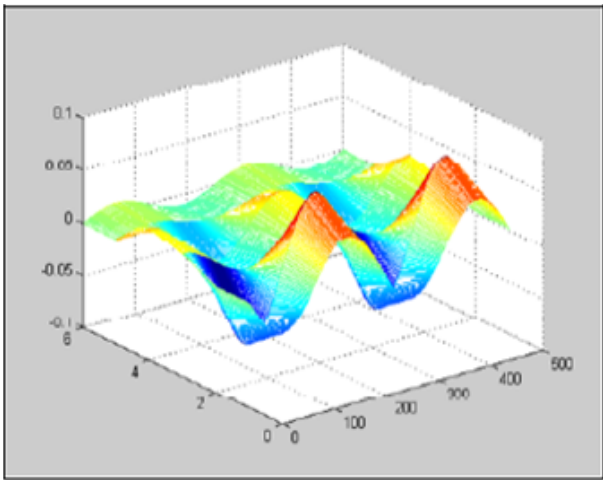

Figure 13: e_GRASP simulator capabilities: Typical integration of MATLAB/ANN Toolbox hand motion.

References

1. Matlab Toolbox for the iRobot Create (MTIC) (2011) Version 2.0.
2. Shaoqiang Y, Zhong L, Xingshan L (2008) Modeling and simulation of robot based on Matlab/SimMechanics. Control Conference, CCC 2008, 27th Chinese, Kunming.
3. Peter Corke (2008) Robotics Toolbox for Matlab.
4. Jambak MI, Haron H, Nasien D (2008) Development of Robot Simulation Software for Five Joints Mitsubishi RV 2AJ Robot Using MATLAB/Simulink and V-Realm Builder. 5th International Conference on Computer Graphics, Imaging and Visualisation, Penang.
5. Olivier M (2004) Cyberbotics Ltd WebotsTM: Professional Mobile Robot Simulation. Inernational Journal of Advanced Robotic Systems.
6. Gourdeau R (1997) Object-oriented programming for robotic manipulator simulation. Robotics and Automation Magazine, IEEE 4: 21-29.
7. Ramasamy S, Arshad R (2000) Robotic hand simulation with kinematics and dynamic analysis. TENCON 2000 3: 178-183.
8. Miller T, Allen K (2004) Graspit! A versatile simulator for robotic grasping. Robotics & Automation Magazine, IEEE 11: 110-122.
9. Miller M, Allen P, Santos V, Valero-Cuevas F (2005) From robotic hands to human hands: a visualization and simulation engine for grasping research. Industrial Robot: An International Journal 32: 55-63.
10. Jagdish R, Radhey S, Rajsekhar A, Bhanu P (2012) Real-Time Robotic Hand Control Using Hand Gestures", Robotic Systems – Applications, Control and Programming. Book. Edited by Dr. Ashish Dutta, ISBN 978-953-307-941-947.
11. Corrales A, Jara A, Torres F (2010) Modelling and simulation of a multi-fingered robotic hand for grasping tasks.11th International Conference on. Control Automation Robotics & Vision (ICARCV), Singapore.
12. Gourret J, Thalmann N, Thalmann D (1989) Simulation of object and human skin formations in a grasping task. 16th annual conference on Computer graphics and interactive techniques.
13. Magnus B (2008) Controlling a Robot Hand in Simulation and Reality. Degree Project Department of Management and Engineering LIU-IEI-TEK-A--08/00336—SE, (2008).
14. Tarmizi W, Adly A, Amirfaiz W, Elamvazuthi I, Begam M (2010) Modeling and simulation of a multi-fingered robot hand. International Conference on Intelligent and Advanced Systems (ICIAS), Kuala Lumpur, Malaysia.
15. Boughdiri R, Bezine H, Sirdi M, Naamane A, Alimi M (2011) Dynamic modeling of a multi-fingered robot hand in free motion. 8th International Multi-Conference on Systems, Signals and Devices (SSD) Sousse.
16. Ohol S, Kajale S (2008) Simulation of Multifinger Robotic Gripper for Dynamic Analysis of Dexterous Grasping. Proceedings of the World Congress on Engineering and Computer Science,San Francisco.
17. Chan C , Yun-Hui L (1999) Simulating dextrous manipulation of a multi-fingered robot hand based on a unified dynamic model . IEEE International Conference on Robotics and Automation Detroit, MI.
18. Orin E, Chao H, Olson W, Schrader W (1985) Pipeline/Parallel Algorithms for the Jacobian and Inverse Dynamics Computations. Proceedings of the IEEE International Conference on Robotics and Automation.
19. Paul RP (1981) Robot Manipulators: Mathematics, Programming, And Control. MIT Press, Cambridge, USA.
20. Fu S, Gonzalez C, Lee S (1987) Robotics Control, Sensing, Vision, and Intelligence. McGraw-Hill International Editions, Singapore.