A Mobile Wireless and Web Based Analysis Tool for Robot Design and Dynamic Control Simulation from Task Points Description

*Tarek M. Sobh, Bei Wang and Sarosh H. Patel**
Department of Computer Science and Engineering
** Department of Electrical Engineering*
University of Bridgeport
Bridgeport, CT 06601, U.S.A.
{beiwang, saroshp}@brideport.edu

Abstract

In this work, we propose a mobile wireless-based, as well as a web-based solution for robot design and dynamic control simulation based on given task point descriptions. The software combines and utilizes the computational power of both the Mathematica and Matlab packages. Given the location and velocity of each task point, our approach formulates the complete design of a 3 DOF robot model by computing its optimal dynamic parameters such as link length, mass and inertia. Furthermore, our package suggests the optimal control parameters (Kp, Kv) for the dynamic control simulation. Our proposal discusses the implemental possibility of making sophisticated scientific computational service available for wireless devices.

Keywords: Robot design, modeling, manipulability, web-based, WAP, wireless computational service, trajectory generation, optimization, control loop simulation, robot control, dynamic control simulation

1 Introduction

A task point's description includes a set of desired positions of an end-effector in the physical space and velocities at these points at a particular instant of time. The problem being considered in this work is to obtain the optimal robot design and dynamic control strategy in such a way that the task can be carried out with maximum manipulability and minimum error in reaching the desired positions and velocities.

Current robotic researches based on task descriptions have focused on the trajectory generation [1, 4] and kinematic synthesis [2] without taking dynamic parameters into consideration. Some research emphasizes the design and numerical optimization of manipulability and cost function [2]. Our robot design and simulation software not only uses the kinematic parameters of task points in designing the manipulability function, but also utilizes the dynamic description of each task point (velocity, etc). Therefore, it offers the user a complete dynamic robot model instead of a simple kinematic description.

WAP (Wireless Application Protocol), as designed to deliver Internet data over wireless networks, brings a new perspective for wireless networks, which is mostly used for voice communication. This work develops wireless applications using JSP and Servlet. It introduces the possibility of designing more sophisticated scientific computations available for wireless devices. Furthermore, comparisons are drawn between scientific computations through the World Wide Web and wireless networks, in terms of convenience and efficiency.

Our software has undergone three major development stages. First, it is a windows application, which runs under the Matlab environment with a Matlab GUI, involving the computational power of both Matlab and Mathematica. It requires local installation of both Mathematica and Matlab on the client machine. Second, it is brought to the World Wide Web for convenience and efficiency by the power of JSP and Servlet. Local installations of the complicated mathematical packages are no longer needed. Clients could access the website with an Internet browser and the server provides computation services. Third and last, our software is made available for wireless clients with cell phone and PDA through WAP technology. This paper focuses on the theory and implementation for the web-based and wireless-based software.

2 Theory

2.1 Manipulability

A robot configuration can be said to be best suited for a particular task, if the manipulability or the dexterity of the manipulator at the set of task points is high. In the past few years numerous approaches have been proposed for calculating the manipulability. The manipulability of a manipulator at a particular point can be defined as the "*the ability of the manipulator to accelerate in all directions from that point*".

Yoshikawa [3] suggested one such measure based on the volume of the manipulability ellipsoid as derived from manipulator kinematic properties i.e., the Jacobian. Yoshikawa's manipulability measure is based on kinematic

data and it calculates the manipulability based on how far is the point from a singularity and thus, be able to exert forces and move uniformly in all directions.

$$\eta_{yoshi} = \sqrt{\det\left|J(q)J^T(q)\right|} \qquad \text{--- (1)}$$

Where $J(q)$ is the velocity Jacobian at that point and q is the joint variable vector.

2.2 The Cost Function

Deciding on the best possible geometric model for a manipulator is a very difficult problem in kinematics as well as mechanics. The equations describing the kinematic behavior of the links are nonlinear and have many variables in the order of thousands [2]. In some cases there might not be any closed solution, whereas in other cases there might be more than one closed solution [2]. One of the best ways (though not the fastest) to solve such a problem is using the theory of optimization. Using optimization, all the possible options are evaluated using a *Cost Function* [2] or *Objective Function* [9]. Depending on whether the Cost Function has to be maximized or minimized the best possible solution is chosen. Multiple factors influencing the modeling technique can be incorporated into the Cost Function.

The criteria used to form the cost function are:

1.Manipulability

2.Accuracy

3.Distance from the point.

The cost function [2] is given by

$$F(K, q1, q2, ...qm) = \xi * L + \sum_{i}^{m} \frac{1}{w_i^2 + b} + \varepsilon * D_i^2 \qquad \text{--- (2)}$$

Where K is the DH parameter of the robot

$q1, q2.....qm$ are the joint vectors of the task points

ξ is the dumping factor

w_i is the manipulability

ε is the weight factor

The optimization parameters are all the Denavit-Hartenberg parameters other than the joint variables.

2.3 Optimizing the Cost Function

The cost function is a minimizing function. Minimizing the function provides the optimal values for the DH table. The function is optimized using the steepest descent algorithm [2], which finds the minima by searching in the direction opposite, to the gradient. If the range of the function does not contain negative values the function always converges, except in a few rare cases when the gradient disappears [2].

2.4 Calculation of Dynamic Parameters

The Robot Design module calculates the manipulator DH table. But the dynamic parameters like Mass, Center of Gravity and Inertia are required for the closed loop control simulation of the robot. Our software simulation package calculates these on the following assumptions:

1.The manipulator links are solid and cylindrical in shape.

2.All links have uniform density (uniform mass distribution).

3.All the links are made of the same material.

4.There are a finite number of actuators and sensors with known specifications that can be used in the design.

The user is requested to enter the link radii. The link radii are necessary for calculating the dynamic parameters of the manipulator. The parameters are calculated as follows:

1.Mass - The user is requested to specify the link radius. Knowing the volume and density the mass of each link can be easily calculated.

$$m_i = \pi r_i^2 a_i d \qquad \text{--- (3)}$$

where m_i is the mass of the ith link,

r_i is the radius of the ith link,

a_i is the length of the ith link,

d is the density of the link material

2.Center of Gravity – The center of gravity is calculated geometrically with respect to the link coordinate frame.

3.Inertia – The Inertias of the links has to be calculated about the respective Center of Gravities. Since the links are considered to be cylindrical, the Inertia about the axis of a cylinder is given by:

$$I_1 = \frac{1}{2} m_i r_i^2 \qquad \text{--- (4)}$$

Using the perpendicular axis theorem the Inertia along the other two axes is given by:

$$I_2 = I_3 = \frac{1}{4} m_i r_i^2 \qquad \text{--- (5)}$$

where m_i is a Mass of the ith link

r_i is the Radius of the ith link.

2.5 Trajectory Generation

The main purpose for trajectory generation is to produce a timed path, which can be tracked by a manipulator. All the task points are linked to generate a trajectory from the initial to the final point on a common time scale. Our implementation uses a seven-degree

polynomial to generate the trajectory. The control loop is implemented over to support this trajectory.

2.6 The PD Control Loop

Most of the control algorithms use proportional derivative (PD) control. In software a local PD control loop [8] is applied to each link independently. It is advantageous to use a PD control loop for the following reasons [6]:

1. It is simple to implement.
2. Since it involves few calculations, it is ideal for real time control provided that the choice of Kp and Kv is appropriate to give optimum control.
3. The behavior of the system can be controlled by changing the feedback gains.

The torque to be applied to the manipulator in order to attain the desired position is calculated using Forward Dynamics [7]. The feedback loop in the case of a computer simulation of the control loop encodes Inverse Dynamics, where as in the case of real time control, the sensors provide the feedback. Figure 1 below displays a block diagram commonly found in robot prototyping research [5].

Where Kp is the proportional gain, Kv is the derivative gain, e is the error in position and e' is the error in velocity.

At every cycle in the Control Loop the manipulator state is compared with the desired position. This gives the Position Error at that instant. Similarly, the *Velocity Error* is also calculated, by comparing the instantaneous velocity of each of the links with the desired velocity at that instant. The torque to be applied at that instant is given by:

Torque = (Kp * *Position Error*) + (Kv * *Velocity Error*)

2.7 Optimization of Kp and Kv

The proposed software optimizes the values of the loop gains (Kp and Kv) by trying all possible values within a given range. The user can also specify this range. The criteria for deciding the optimum values for Kp and Kv is that the Sum of the Square of Errors about the desired trajectory should less than a specified threshold. The user can also run the simulation for particular value of Kp and Kv.

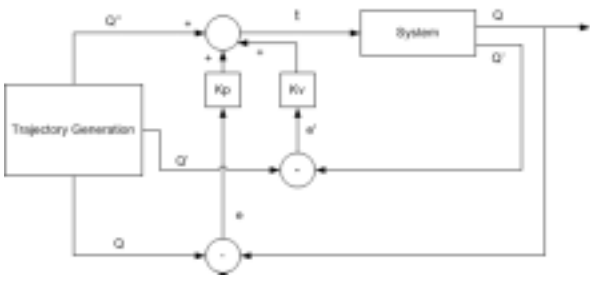

Figure 1 Block diagram of the robot PD control loop

3 The Software Package

Our software package includes five major modules: a dynamic WAP service, a web interface, a kinematics design module, a dynamic design module and a dynamic control simulation module. It is intended that a user would specify the coordinates of each task point, the velocity of the end-effector at each point and the specific time to reach each point. Through kinematic and dynamic computations, our implementation desires a complete optimal robot model with dynamic parameters. It further performs the dynamic control simulation of the obtained robot model and suggests the best control parameters, such as Kp and Kv.

3.1 Dynamic WAP Service

WAP applications are normally developed in WML (Wireless Mark up Language), which is adapted to the limitations of wireless devices. In our application, dynamic WAP documents are generated by JavaServer Pages and Servlets. Our WAP service is hosted under the same directory on the web server with Internet service for regular web browsers.

3.1.1 WAP

Our software is initially designed for a desktop running on reliable networks with, in our case, a speed of 100 Mpbs. Wireless devices such as cellular phones and PDAs, however, have constrained computing environments (small memory, limited CPU power and small displays). Further, wireless networks have less bandwidth and more latency compared to wired computer networks [10]. The WAP platform reuses existing Internet protocols and introduces new technologies to meet the requirements of wireless devices. It standardizes the manner in which wireless devices access Internet data and services [10, 15].

3.1.2 WAP Architecture Overview

Figure 2 WAP Architecture

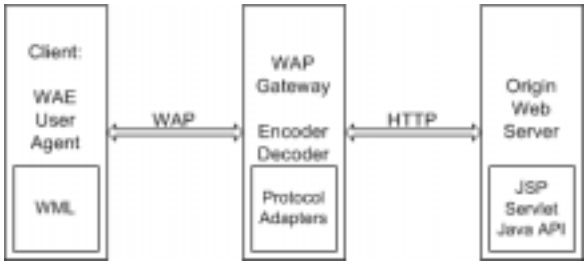

Figure 3 WAP Programming Model

WAP is used for wireless devices, and Java technologies reside at the network terminal and provide computing services. The WAP standard defines two essential elements: an end-to-end application protocol and an application environment based on a browser [10, 15]. The application protocol is a communication protocol stack that is embedded in each WAP-enabled wireless device [10, 15]. The server side implements the other end of the protocol, which is capable of communicating with any WAP client. This is known as a WAP gateway and routes requests from the client to a Web server [10, 15]. Figure 2 illustrates an example structure of a WAP network [10].

3.1.3 WAP Programming Model

As illustrated in Figure 3, the WAP programming model is similar to the WWW programming model, which allows developers to use existing tools (i.e. XML tools) and take advantage of a familiar programming model and proven architecture [18].

The WAE (Wireless Application Environment) architecture enables all contents and services to be hosted on the standard Web origin servers (used by standard web service described in the next section), which can incorporate proven technologies such as JSP and Java Servlet. All content is located using standard URLs [10, 18]. WAE enhances WWW standards and reflects the device and network constraints that are found in mobile terminals. It provides support for low bandwidth and high latency networks. [10, 18]

3.1.4 WAP Limitations

As wireless devices have small screens, limited computing power and small bandwidth, graphics are not easily accessible [11, 14]. As in our work, the web interface works perfectly with graphs and shows the users visual diagrams for the optimization results, such as changes of velocities in terms of desired and simulated results according to a time scale. On the other hand, our WAP service is limited in not being able to deliver images as part of the control simulation result. Instead, it offers a complete list of the dynamic parameters, such as the best Kp and Kv values.

3.1.5 WAP Service Limitations

Our work is an experiment of WAP technology in terms of wireless computing. We focus more on the deliverability instead of the WAP browsing web interface design. More user-friendly and graphic enabled interfaces will certainly enhance the user experience [12, 13]. We used JSP and Java Servlet as the implementing tools; other possible technologies such as Java RMI can be utilized as well [17].

3.2 The Web Interface

The web interface is developed using Java Server Page Technology and Servlet. It is the central control module that communicates with the Mathematica and Matlab applications running on the server. The basic aim is to interact with the user and invoke the respective modules. It uses a modified version of JLink [19] to communicate with Mathematica and a modified version of JMatlink [20] to communicate with Matlab. It controls the Mathematica kernel from JSP pages and Matlab from JMatservlet (part of JMatLink).

The web interface communicates with the kinematics design module and allows the user to enter task point variables, such as coordinates, velocities and time scale. It also interacts with the dynamic design module and a dynamic control simulation module based on user specified radii for each link and range of dynamic simulation variables (Kp, Kv). It provides the web users with a complete design of the kinematic and dynamic modules, and also the optimized PD control variables. The web implementation converts the detailed control simulation results, such as the desired path vs. obtained path curves, into jpeg images so that they can be easily viewed on the web page.

3.3 Kinematic Design Module

The kinematic design module generates the best kinematics robot configuration with the maximum manipulability at user-specified task points. With modifications based on the kinematics synthesis package [2] builds on top of *Robotica* package (v.3.60, Copyright 1993 Board of Trustees, University of Illinois); this module applies numerical optimization in constructing a kinematics robot model. The user (a robot designer) enters a set of task points into this module and obtains a robot configuration in the form of a DH table, describing the optimal kinematics properties of the three-link robot.

The main Mathematica (V. 4.1, Wolfram Research Inc. 2002) procedure that triggers the optimization event is defined as:

*DesignRobot [task*_points, *configuration, precision, size_dump, file_name]*

Where task_points is a matrix with xyz coordinates of task points; configuration is a string of ‘R’s and ‘P’s describing prismatic or rotational joints. For example, “RRR” stands for an articulated manipulator. Our module tries all possible joints combinations for a 3-link robot, achieving the best configuration. Precision and size_dump handle the precision and dimensions of the robot. At last, file_name specifies the location in which the robot configuration file is stored [2].

Table 1 Structure of the dyn matrix

1	Alpha	Link twist angle
2	A	Link length
3	T	Link rotation angle
4	D	Link offset distance
5	Sigma	Joint Type, 0 for revolute, none-zero for prismatic
6	Mass	Mass of the link
7	Rx	Link COG with respect to the link coordinate frame
8	Ry	
9	Rz	
10	Ixx	Elements of link inertia tensor about the link COG
11	Iyy	
12	Izz	
13	Ixy	
14	Iyz	
15	Ixz	
16	Jm	Amateur inertia
17	G	Reduction gear ratio. Joint seed / link speed
18	B	Viscous friction, motor referred
19	Tc+	Coulomb friction (positive rotation), motor referred
20	Tc-	Coulomb friction (negative rotation), motor referred

3.4 Dynamic Design Module

While the Kinematic model is only concerned with the kinematic properties of the robot, the dynamic module involves user input as task point's velocities and operation time. Running in the MATLAB (V. 6.1, MathWorks Inc. 2001) environment, the input to this module is the file, which is generated by Mathematica. This module reads the robot configuration file generated by the kinematic design module as shown in figure 11.

Using the DH parameters and the formulae discussed in the previous section, this module generates the Dynamic parameter matrix 'dyn'. The dynamic matrix is a (n x 20) matrix where n is the number of degrees of freedom of the manipulator. This matrix defines all the dynamic parameters of the manipulator. The structure of the dyn matrix is shown in table 1 [12]:

In order to generate the dynamic parameters the user is asked to input the radii of the links. Based on the radii, the mass of the links, and successively the center of gravity (CoG) and Inertia are calculated. Finally, a Robot object is created in MATLAB using the dynamic parameters calculated; this robot has all the dynamic properties as desired.

3.5 Dynamic Control Simulation Module

Programmed in the MATLAB environment, this module deals with the control simulation of the dynamic robot model. The user is asked to enter the co-ordinates of points with respect to a time frame and the velocities at

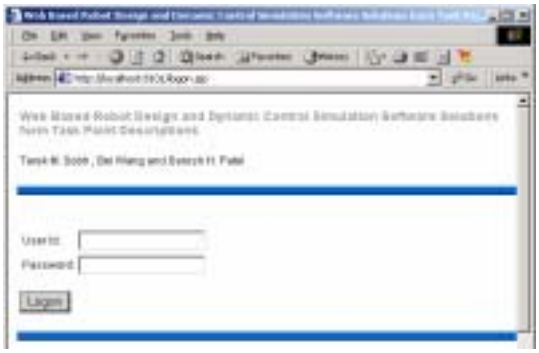

Figure 4 User login screen

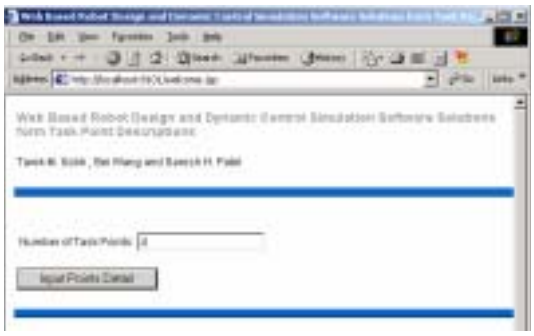

Figure 5 User specifies number of task points

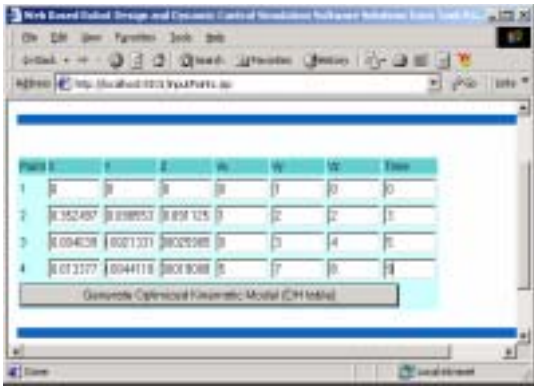

Figure 6 User specifies the coordinates and velocities of each task points with respect to a time scale

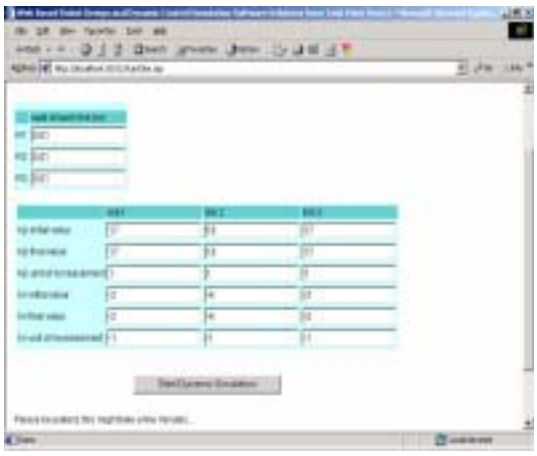

Figure 7 User specifies link radii for dynamic model generation, and Kp, Kv initialization for dynamic PD control simulation

those points. The points specified by the user are linked using a seventh order polynomial in order to generate a trajectory. The PD control loop simulation is run using this generated trajectory.

The user has a choice of either specifying the values of Kp and Kv or else specifying a range of values for Kp and Kv and the step increment. In the second case the software finds the optimum values for both Kp and Kv. The same is the case with the update frequency, where the user can either specify a value or range with the increment, in which case the software decides the optimum value.

4 Results

4.1 Web Interface

This section includes results from a complete sample program run on the web interface. First, the user logs on to our Jrun Server using the username “guest” and password “demo”, as shown in figure 4. Second, the user specifies a number of task points (figure 5). Then the user specifies the coordinates and velocity of each task point with respect to a time scale (figure 6). Figure 7 and figure 8 are part of the dynamic model generation and dynamic control simulation, showing the best Kinematic and Dynamic model (DH table and Dynamic table), as well as the optimal values of Kp and Kv for each link. Figure 9 to figure 16 are the results of the dynamic simulation, displaying the desired path and obtained path for each link.

4.2 WAP Interface

This section includes results from a sample program run on the WAP interface of the wireless device simulator. We use the Openwave SDK 6.2 [16] with its Openwave Mobile Browser Simulator in our prototyping WAP application development. The user input requirements are almost the same as the WWW interface.

First, the user specifies a number of task points (figure 17). Then the user specifies the coordinates and velocity of each task point with respect to a time scale (figure 18, 19). Note that since the wireless device (cell phone simulator interface) has a small display, the user has to scroll down to input each data. Figure 20 shows portion of kinematic parameters generated by Mathematica and Robotica on the server. Figure 21, 22 and 23 are part of the dynamic model generation and dynamic control simulation.

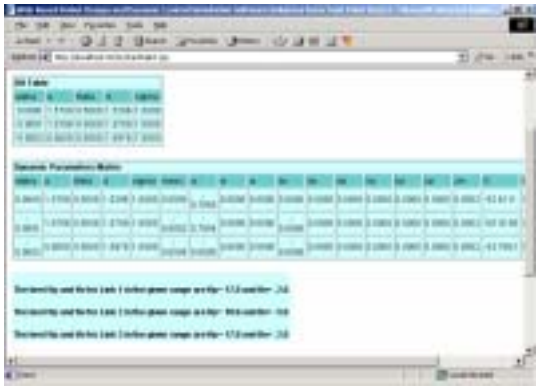

Figure 8 DH table, Dynamic Parameter Matrix and optimal Kp, Kv values for each link

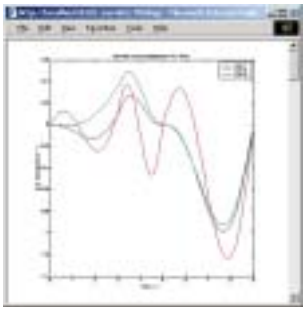

Figure 9 Desired Trajectory for link 1, 2, 3

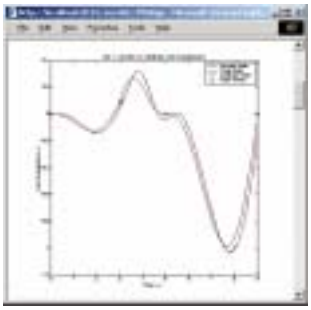

Figure 10 Desired Vs. obtained link displacement for link 1

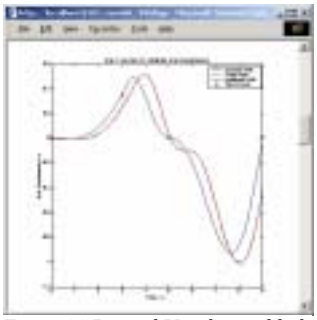

Figure 11 Desired Vs. obtained link displacement for link 2

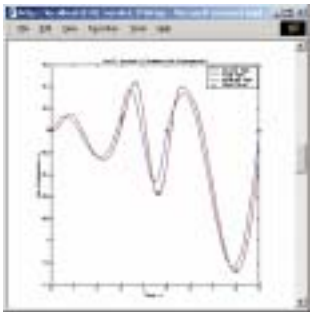

Figure 12 Desired Vs. obtained link displacement for link 3

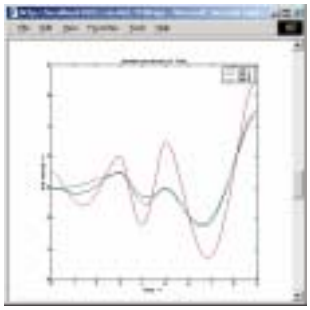

Figure 13 Desired velocity trajectory for link 1, 2 and 3

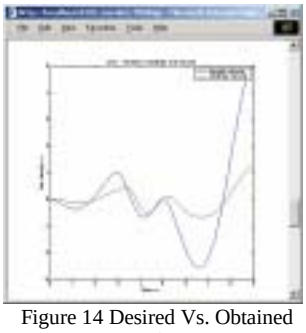

Figure 14 Desired Vs. Obtained velocity for link 1

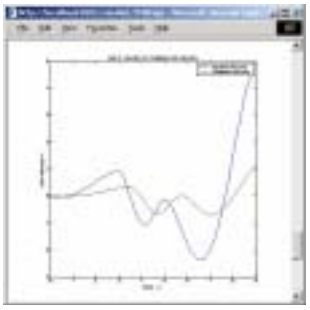

Figure 15 Desired Vs. Obtained velocity for link 2

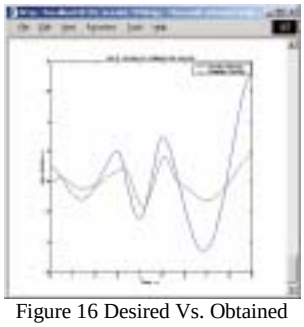

Figure 16 Desired Vs. Obtained velocity for link 3

Figure 17 User input, number of task points

Figure 18 Input task points details, as X, Y, Z

Figure 19 Scroll down to input Vx, Vy, Vz and time scale

```
A Robotica input data file

DOF=3
The Denavit-Hartenberg parameters:
joint1=prismatic
a1=-0.0001248231466785685
alpha1=1.5706151093255496
d1=d1
theta1=1.6850173772401147
joint2=prismatic
…
```

Figure 20 Portion of kinematic parameters generated by Mathematica and Robotica on the server

Figure 21 Input dynamic simulation parameters, such as R1, R2 and R3

Figure 22 Input Kp initial values for each link

Figure 23 Input Kp final values for each link

Figure 24 Input Kv unit of increase for each link

Figure 25 Part of dynamic simulation result

Figure 26 Part of dynamic simulation result

5 Conclusion

In this paper we have presented a mobile wireless as well as web-enabled software utility for the complete design and optimum dynamic control of a manipulator from a task point description. The software generates the basic configuration of a manipulator based on user specified task points, in order to attain the greatest manipulability in the workspace. This software also provides the optimum values of Kp, Kv for optimum dynamic control. We experiment with wireless WAP technology to deliver sophisticated scientific computation capabilities.

6 Future Development

The potential future enhancements to our work include:

1. Building better objective functions, by including more design criteria and assigning weighing coefficients to each according to their importance.
2. Customizable objective functions, by which the user can design a manipulator by defining his/her own objective function.
3. Implementation of advanced and better trajectory generation algorithms.
4. Faster algorithms for calculation of inverse kinematics.
5. Designing a numerical solution package for inverse kinematics for a few common robot models.
6. Implementation of Proportional Integral Derivative (PID) control in addition to Proportional Derivative (PD) control, in order to further minimize the error.
7. Building a graphical user interface for wireless devices
8. Improving the connectivity and speed of WAP service for computationally intense applications

7 References

[1] Proceedings: Lloyd J., Hayward V. “A Discrete Algorithm for Fixed-path Trajectory Generation at Kinematic Singularities”, *IEEE Int. Conf. on Robotics and Automation*, Minneapolis (1996)

[2] Proceedings: Sobh T. and Toundykov D. “Kinematic Synthesis of Robotic Manipulators from Task Descriptions”. To appear in *IEEE magazine on Robotics and Automation, summer* (2003).

[3] Journal: Yoshikawa T. “Manipulability of Robot Mechanisms”. *International Journal of Robotics Research*, vol.4, pp.3--9 (1985)

[4] Proceedings: Pires E., Machado J. and Oliveira P. "An Evolutionary Approach to Robot Structure and Trajectory Optimization", *10th International Conference on Advanced Robotics,* pg. 333-338, Budapest, Hungary, August (2001)

[5] Journal: Sobh, T., Dekhil, M., Henderson T., and Sabbavarapu A. "Prototyping a Three Link Robot Manipulator", *International Journal of Robotics and Automation*, Vol. 14, No. 2 (1999)

[6] Report: Dekhil, M., Sobh T., Henderson T., Sabbavarpu A. and Mecklenburg R. "Robot manipulator prototyping *(Complete design review)*", University of Utah (1994)

[7] Books: Spong M. and Vidyasagar. "Robot Dynamics and Control", Wiley, New York (1989)

[8] Journal: Grigorian M., Sobh T. "Design-Simulation-Optimization Package for a Generic 6-DOF Manipulator with a Spherical Wrist", to appear in *Journal of system Analysis, Modeling and Simulation,* April (2003).

[9] Journal: Pamanes J., Montes P., Cuan E., Rodríguez F. "Optimal Placement and Synthesis of 3R Manipulator", *International Symposium of Robotics and Automation*, Monterrey, Mexico (2000)

[10] Report: Mahmoud Q. "WAP for Java Developers", URL: http://www.javaworld.com/javaworld/jw-06-2000/jw-0602-wap.html, *Java World* (2000).

[11] Proceedings: Ericsson, T., Chincholle, D., Goldstein, M. "Both the Cellular Phone and the Service Impact WAP Usability", *Joint Proc. of IHM 2001 and HCI 2001*, Springer Verlag, Berlin (2001).

[12] Proceedings: Chittaro, L., Dal Cin, P. "Evaluating Interface Design Choices on WAP Phones: Single-choice List Selection and Navigation among Cards", Proc. Mobile HCI 2001:3*rd International Workshop on Human Computer Interaction with Mobile Devices*, IHM-HCI (2001) 7-13

[13] Proceedings: Schmidt, A., Schroder H., Frick O. "WAP – Designing for small user interfaces", *Proc. CHI2000 Conference Human Factors in Computing Systems*, Abstracts Volume, ACM Press, New York (2000) 187-188

[14] Proceedings: Buchanan, G., Jones, M., Thimbleby, H., Farrant, S., Pazzani, M. "Improving Mobile Internet Usability", *Proc. 10th International WWW Conference*, ACM Press, New York (2001), 673-680

[15] Report: WAP Forum, "Wireless Application Environment Overview", URL: http://www.wapforum.org, February 3, 1999.

[16] Openwave Inc., *Openwave SDK version 6.2*, URL: http://developer.openwave.com/, February, 2003

[17] Proceedings: Wall T., Cahill V. "Mobile RMI: Supporting Remote Access to Java Server Objects on Mobile Hosts" Proceedings of the 3rd International Symposium of Distributed Objects and Applications (DOA'01), Dublin, Ireland, 2001

[18] Journal: L. Deri. "Beyond the Web: Mobile WAP-based Management", *Journal of Network and Systems Management, Special Issue on Web-Based Management*, (in press) 2000.

[19] Wolfram Research Inc., *J/Link version 2.0.1,* URL: http://www.wolfram.com/solutions/mathlink/jlink/, February, 2003

[20] Stefan Müller, *JMatLink version 1.00,* URL: http://www.held-mueller.de/JMatLink/, February,2003

Biographies

Tarek M. Sobh, Ph.D., P.E. Professor Tarek M. Sobh received the Ph.D. and M.S. degrees in Computer and Information Science from the School of Engineering, University of Pennsylvania in 1991 and 1989, respectively, and the B.Sc. in Engineering degree with honors in Computer Science and Automatic Control from the Faculty of Engineering, Alexandria University, in 1988. He is currently the Dean of the School of Engineering at the University of Bridgeport, Connecticut; the Founding Director of the Interdisciplinary Robotics, Intelligent Sensing, and Control (RISC) laboratory and a Professor of Computer Science, Computer Engineering, Mechanical Engineering and Electrical Engineering.

He was the Interim Chairman of Computer Science and Computer Engineering and the Director of External Engineering Programs at the University of Bridgeport. He was an Associate Professor of Computer Science and Computer Engineering at the University of Bridgeport from 1995 -- 1999, a Research Assistant Professor of Computer Science at the Department of Computer Science, University of Utah from 1992 -- 1995, and a Research Fellow at the General Robotics and Active Sensory Perception (GRASP) Laboratory of the University of Pennsylvania from 1989 -- 1991. He was the Chairman of the Discrete Event and Hybrid Systems Technical Committee of the IEEE Robotics and Automation Society from 1992- 1999, and the Chairman of the Prototyping Technical Committee of the IEEE Robotics and Automation Society from 1999-2001. His background is in the fields of computer science and engineering, control

theory, robotics, automation, manufacturing, AI, computer vision and signal processing.

Dr. Sobh current research interests include reverse engineering and industrial inspection, CAD/CAM, active sensing/imaging under uncertainty, robots and electromechanical systems prototyping, sensor-based distributed control schemes, unifying tolerances across sensing, design, and manufacturing, hybrid and discrete event control, modeling, and applications, and mobile robotic manipulation. He has published over 100 journal and conference papers, and book chapters in these and other areas. Dr. Sobh edited or co-edited issues of several international research Journals in these areas. He has been on the program committees of several international conferences and has chaired and organized several conferences, sessions, workshops, and tracks in Robotics, Automation, and Sensing meetings and has made many presentations, invited talks, invited lectures and colloquia, seminars, and panel participations, at research meetings, University departments, research centers, and companies.

Dr. Sobh is active in consulting and providing service to many industrial organizations and companies. He has consulted for many companies in the U.S., Switzerland, India, Malaysia, Dubai, and Egypt, to support projects in robotics, automation, manufacturing, sensing, numerical analysis, and control. He has also worked at Philips Laboratories in New York, and a number of companies in Egypt.

Dr. Sobh has been awarded many grants to pursue his work in robotics, automation, manufacturing, and sensing. Dr. Sobh is a Licensed Professional Electrical Engineer (P.E.), a Certified Manufacturing Engineer (CMfgE) by the Society of Manufacturing Engineers, a Certified Professional Manager (C.M.) by the Institute of Certified Professional Managers at James Madison University, a Certified Reliability Engineer (C.R.E.) by the American Society for Quality Control, a member of Tau Beta Pi (The Engineering Honor Society), Sigma Xi (The Scientific Research Society), Phi Beta Delta (The International Honor Society), and Upsilon Pi Epsilon (The Computing Honor Society). Dr. Sobh was the recipient of the Best Research Award by the World Automation Congress in 1998.

Dr. Sobh is a member or senior member of several professional organizations including; ACM, IEEE, IEEE Computer Society, IEEE Robotics and Automation Society, IEEE Computer Society Technical Committee on Pattern Analysis and Machine Intelligence (PAMI), the International Society for Optical Engineering (SPIE), the National Society of Professional Engineers (NSPE), the New York Academy of Sciences, the American Society of Engineering Education (ASEE), the American Society of Quality (ASQ), the American Association for the Advancement of Science (AAAS), Society of Manufacturing Engineers (SME), and a founding member of the Society for Industrial Computing.

Bei Wang Bei Wang is a senior at University of Bridgeport, pursuing a B.S in computer science and a B.S. in mathematics. She is a member of IEEE, ACM, Phi Kappa Phi (The Honor Society) and Upsilon Pi Epsilon (The Computing Honor Society). She is also the president of Upsilon Pi Epsilon university chapter and Vice-president of Phi Kappa Phi university chapter.

Bei Wang is a 2002 Upsilon Pi Epsilon Microsoft Scholarship Award recipient and 2002 Sigma Xi Grant-in-Aid of Research Recipient. She also receives Full tuition Academic Scholarship from University of Bridgeport. She is nominated for inclusion in the 2003 edition of WHO'S WHO AMONG STUDENTS IN AMERICAN UNIVERSITIES AND COLLEGES.

Bei Wang's major research interests are Artificial Intelligence/Robotics, Computational Biology and Algorithm Research.

Sarosh H. Patel Sarosh H. Patel received his B.E. degree in Electrical and Electronics Engineering from the Faculty of Engineering Osmania University, India in 2002 and is currently pursuing his M.S. in Electrical Engineering from the school of Engineering, University of Bridgeport. He is a member of ISTE. He had been involved in teaching Mathematics and Programming from 1998 -- 2001. His interests include programming simulations, network security and ethical hacking.