ENSC 488
Introduction to Robotics
Fall 2015, Simon Fraser University
Class Group Project Report

Instructor	Dr. Shahram Payandeh
Submission Date	16th December 2015

Submitted By:

Benjamin Ng	301180288
Ra'na Chegani	301282847
Sohail Sangha	301186636

Contents

Introduction

As statistics shows in Canada the population of people over 65 years old is increase throw the time. By 2011 %14.4 of the population were over 65 years old and it expected to grow over time. It has been predicted that by the year 2021, %18.5 of the population of Canada will be over 65 years old [1]. This growing population require services that should be provided for them to have a comfortable life. A majority of elderly people are not able to take care of themselves, and need constant care and attention. Nursing homes can provide this services for them, Most of the elderly people prefer to live in their own home, and as a result they will need constant attention from another person.

In recent years by impressive progresses in robotic field, designing robotic systems that can assist elderly people in home environment, has become an active research field.

Nowadays in Italy, Spain and Sweden, home of some elderly people have equipped to the sensors that can track their activity and health. Besides Mobile telepresence robots, a wheeled videoconferencing system that can be piloted remotely, can let doctors and relatives check on elderly people (figure1). A robotic seal has been used in some nursing homes as a companion for lonely residents (figure2) [2].

Figure 1

Figure 2

To make a long story short, from the discussion above it can be concluded, using robots in elderly care can make their life much easier.

In this project, the goal is to design a simple robotic system that can move around the elderly living environment. The robotic system has a camera on the end of the last link, which enables person to control robot remotely.

Requirements

The designed robot should have the following requirements:

a. It should have a moving platform that enables it to move around the elderly people's living environment
b. Have an articulating arm (Similar to PUMA 560) which has a camera system attached to the end of the last link enables it to place in different locations, like on the table, under the sofa, …
c. It is possible to control the combination of the moving platform and articulating arm remotely

Designed robot should be simulated in open GL. Simulated robot should have the following requirements:

a. It can move around easily
b. The designed robot should have a home configuration, which put it in a location that it does not clutter with living environment
c. It can be manufactured by use of standard components in the market

After running the program user should be able to enter different locations in the space and the camera should be able to reach them by calculating the appropriate value for every angel, enter joint angels and camera should be able to position its joint to the desired angels, camera should move to the desired location by following a specified trajectory, this way the movement of robot will be animated, and it should monitor the balance of the robot, assuming that the camera have some weight.

Robotic System Design

The system design of the robot largely influences household support for the elderly. In such case, the robot is designed to complement several design aspects in order to provide sufficient support. The robot must not only be able to move freely in 3D motion, but also lightweight in the event the robot collapses and require assistance, and rigid enough to survive through punishment.

The concept of this robot can be easily mistaken for humanoid structured robots. Although they provide a sense of comfort that support to an elderly usually comes in the form of another person, through investigation, they are currently highly inefficient as senior support mechanisms. For instance, consider Honda's humanoid robot, ASIMO, world's most advanced humanoid robot for domestic use.

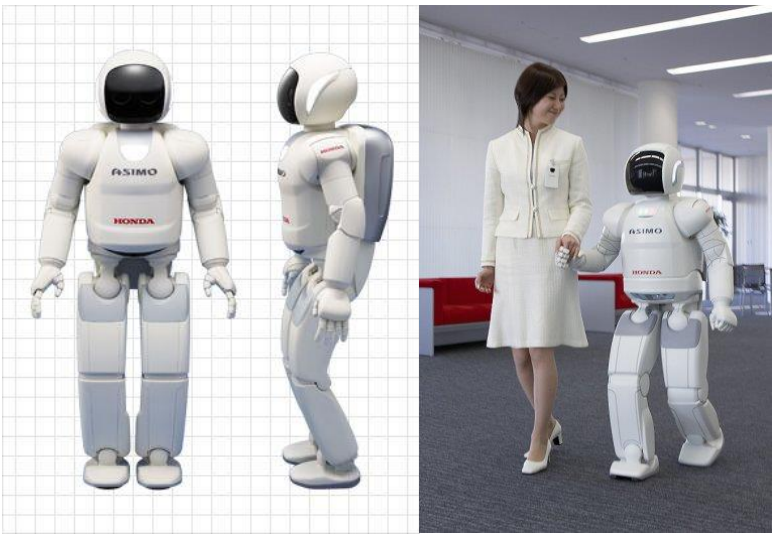

Figure 3: ASIMO robot designed from Honda in the form of a humanoid

ASIMO is designed as a human companion, and can easily provide assistance in everyday life. However, as an assistance mechanism for the elderly, ASIMO cannot perform heavy lifting and doesn't have proper foundation to assist a fallen elderly, while sustaining its own balance. ASIMO has an operating time of only 1 hour and grasping force of 0.5 kg/hand, which is highly inefficient for our scenario. However, one of ASIMO's arms provide up to 7 degrees of freedom, which is most likely the most useful aspect of the robot.

Arms in general has been a large influence with industrial and commercial robots. The focus now aims towards robots for practicality instead of visual appeal. Robots with high efficiency in the workplace are undoubtedly the Articulated Robots excellent for assembly and packaging applications.

SCARA is called a Selective Compliant Assembly Robot Arm, designed for assembly related tasks, and is well known for speed and high precision. The robot is an excellent runner-up, however it fails to meet several requirements. SCARA has the mobility similar, but not better than a human arm. Although the human arm is efficient for everyday use, the concept of the human arm is a base case for all industrial robot arm design. SCARA is restricted in horizontal sweeping and vertical movement, and does not allow rotation along any other axis other than vertical.

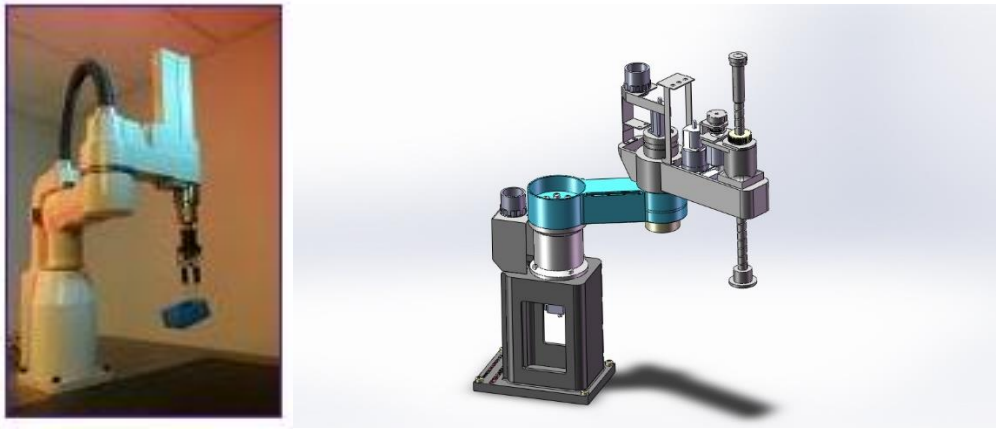

Figure 4: SCARA robotic arm

The structure of SCARA is a chain of rigid linkages connected through parallel revolute joints. These articulated joints have vertical axis which limits all joints in the vertical direction. An excellent robot of high precision and very specific scenarios such as chip manufacturing and building electronics, but unsuitable for elderly support.

The final, and selected system design, is the PUMA robotic arm. As its name suggests, the PUMA (Programmable Universal Machine for Assembly) is a robotic arm designed to suit any user in any field. The robot allows users to program the machine to perform specific actions and reach to certain scenarios.

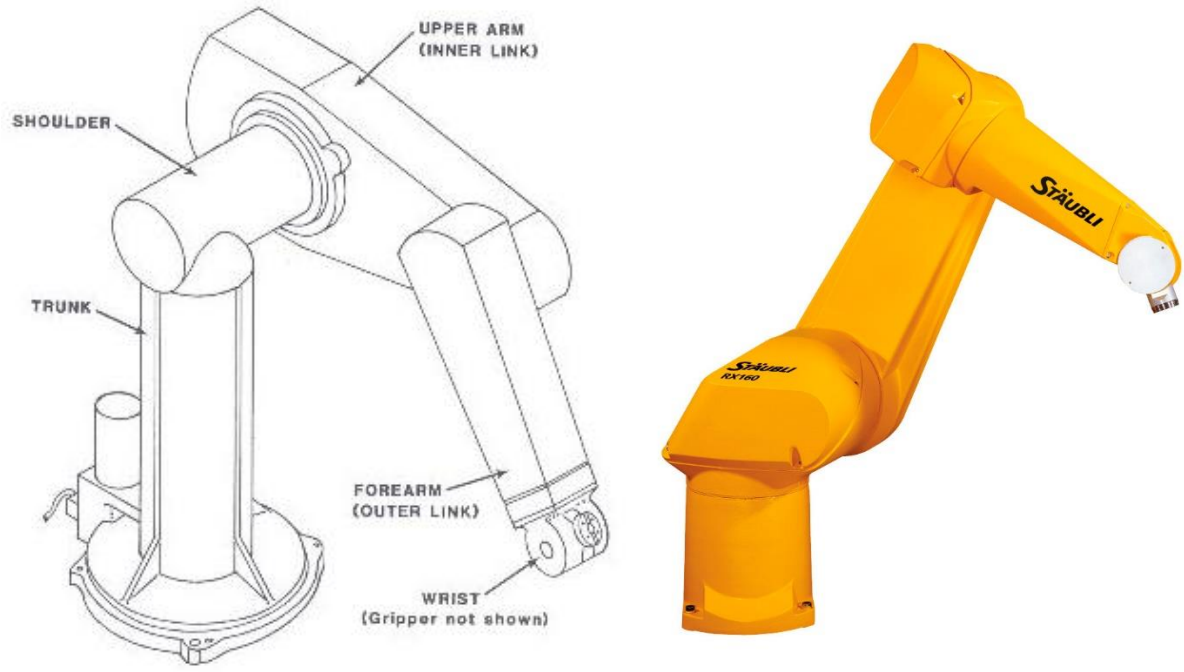

Figure 5: PUMA Type robot arm

Although the PUMA type robot is not perfect, its flaws are negligible. Because the robot is designed for an all-purpose use, the amount of precision and speed is much less compared to the SCARA robot. Its structure consists of a chain of rotary joints with various axes of rotation. This allows it to move freely in space and perform all tasks. Not all joints on the PUMA are parallel, as in figure 3, the second joint is orthogonal, allowing the robot to have 6 degrees of freedom. As such, this robot configuration will become the basis of this project.

Once the main basis has been set, the attention is now directed on the motors for the arms. The robot must be able to perform some form of lifting and be able to move without problems. The power consumption must not be extraordinary as the robot must last through an entire days work to accompany the elderly user. Table 1 lists the practicality and efficiency under various criteria.

Characteristics	**DC Motor**	**Torque Motor**	**Stepper Motor**	**Induction Motor**	**AC Synchronous Motor**
Power Capacity	Low Average	Average	Low	High	High
Speed Controllability	Good	Good	Good	Average	Average
Speed Regulation	Average/Good	Average/Good	Average	Good	Excellent
Linearity	Average/Good	Average	Poor	Poor	Poor
Operating Bandwidth	Good	Good	Low	Good as long as we have frequency control	Good as long as we have frequency control
Starting Torque	Good	Good	Good	Average	N/A
Power Supply Requirements	DC	DC	DC and AC	AC	AC
Commutation Requirements	Split-ring and brushes	None	None	None	Slip-ring and brushes
Power Dissipation	Average	Average	Average/High	Low	Low

Table 1: Characteristics of various motor types

For mobility, this robot has multiple aspects to choose from. As such, one of the considerations for mobility was wheels. The robot will be able to move in any direction in the x and y direction, however, wheels are not great of climbing stairs, rugged terrain, and mud. Even though the user many not come into contact with any of these terrain, the robot in order to stay near or provide support will definitely come into contact with such terrain.

Figure 6: wheels

The second consideration was legs. Legs have been a huge mobility factor for humans and it provides the ability to move up and down stairs, however, balance is a huge issue. A lot of power will be needed to activate sensors and actuators for the robot to have correct balance. This will definitely prove to be an issue for the arms if there isn't enough power going through.

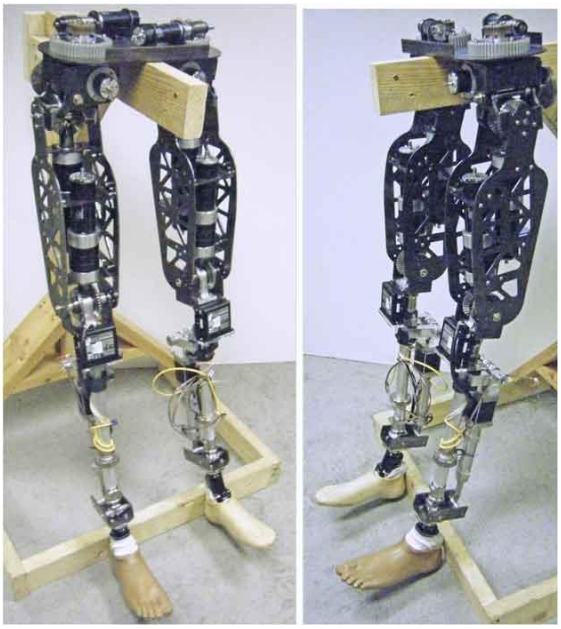

Figure 7: Robotic legs

The last and selected design was the continuous track, which uses multiple wheels, excellent for all terrain purposes, and because of the elongated track of wheels, hill climb will not be an issue. However, this design does limit the speed of the robot, but considering the target user is not mobile efficient and very fast, this design will deliver accordingly.

Figure 8: Continuous track

The concept of this design not only compliment simple parts replacement, but simplifies the testing process significantly. The robot's theoretical and systematic structure has been designed from scratch, however through countless research, it has been decided that the robot will use pre-existing hardware from the market.

This provides four excellent points:

1. The absence of hardware design time and financial investment in research and development in new material structure would lower the overall cost of the robot, to allow the elderly to afford the robot. Not only is the investment going towards the material cost; labour and machinery cost is also factored into play.
2. The confidence in purchasing pre-built parts with great pedigree will entitle high quality.
3. As a start-up company; the leisure of time is non-existent. The demand for the part exists, therefore many larger and well-known companies will occupy this market before anyone else can enter
4. Using existing parts allow users and distributors the simplicity for repairs and parts replacements. Many of the parts can be sourced under the parts manufacture warranty, hence reliving the small start-up from hardware responsibility

Forward Kinematics

Forward kinematics in robotics deals with the orientation and location of the manipulator with respect to a desired frame. In traditional form, this information is calculated with the help of DH parameters. The DH parameters encapsulate the physical form of the links of the robot, using which transformation matrices, relating frames attached to the links, can be calculated. Using these transformation matrices, the robot can find the location and orientation of a frame or manipulator end-point with respect to the environment or tool. The robotic system designed in the project was a PUMA type system. Hence the first three degrees of freedom, leading from the base of the robot, are responsible for the end point

location, while the wrist has three more degrees of freedom which allow it to orient itself at any desired angle in 3D. The forward kinematics algorithm has been resolved for PUMA type robots and extensively available in literature. The main need for forward kinematics for simulation purpose is to compute the location of various frames for visualization. Given that OpenGL has an internal capability to compute transformation and rotation matrices, the simulation was effectively produced by simply rotating the frames inside the OpenGL system itself thus reducing the redundancy of having to calculate and assign frame to various components. This was largely achieved due to the nesting behavior of OpenGL objects, where the rotation and transformation applied to an object can be carried to its children, and by simply applying a relative rotation, the effect of joint rotation in the robot can be achieved. More information on the nesting strategy is available under the software architecture.

Inverse Kinematics

In robotics, inverse kinematics is responsible for finding the required robot joint angle given a specific orientation and location of the manipulator frame. While resolving forward kinematics with software strategies is quite trivial, such cannot be done for inverse kinematics. As mentioned earlier in previous section, PUMA type robots effectively have a location and rotation component. Based on this approach, inverse kinematic solutions are available throughout literature. The following solution has been taken from “Introduction to Robotics by John J. Craig”.

Given:

$$P = [p_x, p_y, p_z], is\ the\ location\ of\ manipulator\ frame$$

$$\theta_1 = Atan2(p_y, p_x) - Atan2(d_3, \pm\sqrt{p_x^2 + p_y^2 - d_3^2}$$

$$K = \frac{p_x^2 + p_y^2 + p_z^2 - a_2^2 - a_3^2 - d_3^2 - d_4^2}{2a_2}$$

$$\theta_3 = atan2(a_3, d_4) - atan2(K, \pm\sqrt{a_3^2 + d_4^2 - K^2}$$

$$\theta_{23} = atan2[(-a_3 - a_2c_3)p_z - (c_1p_x + s_1p_y)(d_4 - a_2s_3), (a_2s_3 - d_4)p_z - (a_3 + a_2c_3)(c_1p_x + s_1p_y)]$$

$$\theta_2 = \theta_{23} - \theta_3$$

Given the rotation matrix R with elements labeled as $r_{row,column}$

$$\theta_4 = atan2(-r_{13}s_1 + r_{23}c_1, -r_{13}c_1c_{23} - r_{23}s_1c_{23} + r_{33}s_{23})$$

$$s_5 = -(r_{13}(c_1c_{23}c_4 + s_1s_4) + r_{23}(s_1c_{23}c_4 - c_1s_4) - r_{33}(s_{23}c_4))$$

$$c_5 = r_{13}(-c_1s_{23}) + r_{23}(-s_1s_{23}) + r_{33}(-c_{23})$$

$$\theta_5 = atan2(s_5, c_5)$$

$$s_6 = -r_{11}(c_1c_{23}s_4 - s_1c_4) - r_{21}(s_1c_{23}s_4 + c_1c_4) + r_{31}(s_{23}s_4)$$

$$c_6 = r_{11}[(c_1c_{23}c_4 + s_1s_4)c_5 - c_1s_{23}s_5] + r_{21}[(s_1c_{23}c_4 - c_1s_4)c_5 - s_1s_{23}s_5] - r_{31}(s_{23}c_4c_5 + c_{23}s_5)$$

$$\theta_6 = atan2(s_6, c_6)$$

The above equations stand true for the puma type robot given in the orientation as follows:

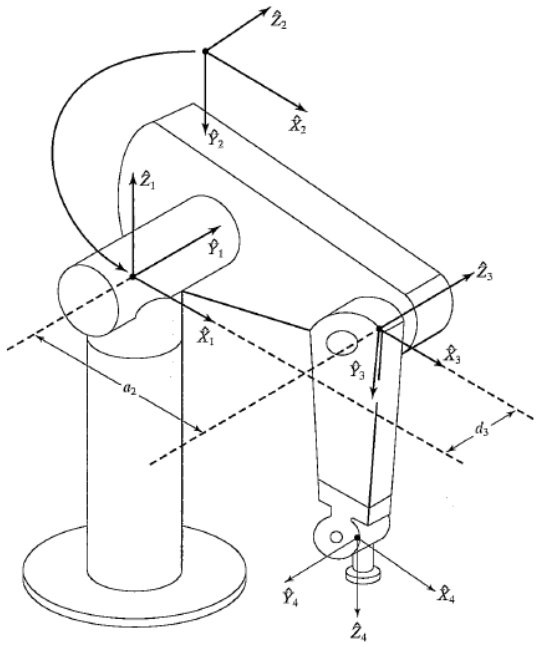

Figure 9: Some frame assignments of PUMA robot, taken from "Introduction to Robotics 3rd edition by John J. Craig"

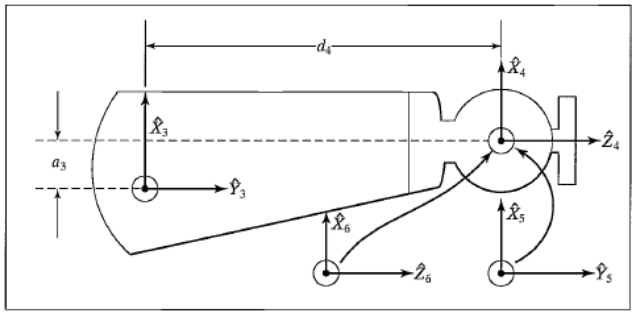

Figure 10: Spherical wrist frame assignments of PUMA robot, taken from "Introduction to Robotics 3rd edition by John J. Craig"

The inverse solution assumes that the robot frames and home configuration is assigned in a particular way. This became a problem when the inverse kinematics equations were used for the robot of this project. Due to the frame or home configurations the angles computed for $\theta_2 and\ \theta_3$ were not correct. The option was either to redraw the complete robot or to compute the angles using another approach. An alternate strategy was opted to compute the angles geometrically; this was facilitated by the simplistic geometrical design of the robot. The mathematical equations are as follows:

θ_1, is computed similar to PUMA type robot

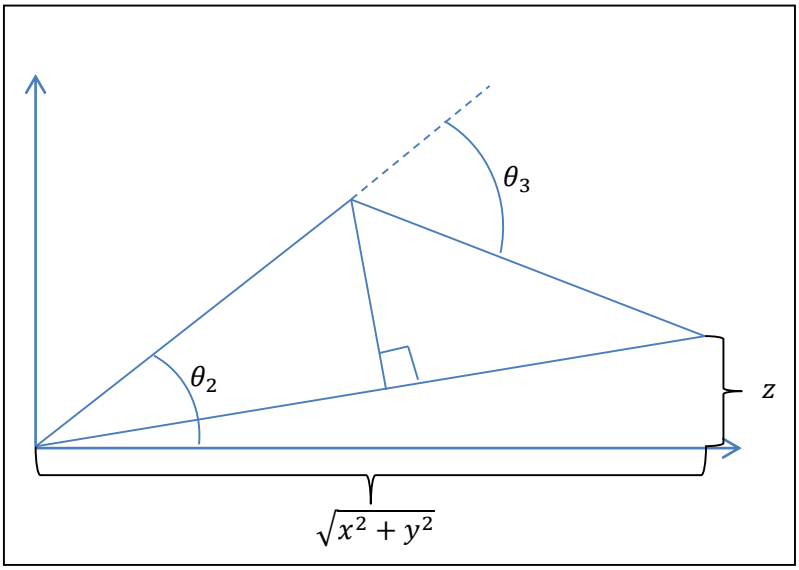

Figure 11: geometric solution for θ_2 and θ_3

As seen in the figure above, the solution for $\theta_2 and\ \theta_3$ is reduced to a 2D system. Further simplicity is offered by the fact that the length of both arms is the same (l), leading to the following equations:

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$\alpha = \text{acos}\left(\frac{r}{2l}\right)$$

$$\beta = atan2(z, \sqrt{x^2 + y^2})$$

$$\theta_2 = \alpha + \beta$$

$$\theta_3 = 2\alpha$$

The above equations yielded the desired joint angles and are computationally cheaper than the inverse equations for PUMA robot from literature. Since the transformation of the robot from frame {4} onwards was exactly the same as PUMA type robot, the inverse kinematics equation given above for $\theta_4\ to\ \theta_5$ were used.

Platform Kinematics

Compared to robot kinematics the platform kinematics was trivial. This was owing to the "two-wheel side by side" design, and further rules imposed on the robot for movements. All the movements to be done by the robot were contained in a rotate and move pattern. The wheels would either rotate anti or together at same speed, thus never resulting in arcing trajectory. For each new point to be moved to, the wheel base would rotate by the required angle, and then the robot would move forward in a straight line.

Given:

$$B = [b_x, b_y], is\ the\ current\ location\ of\ base\ on\ floor$$

$$G = [g_x, g_y], is\ the\ goal\ location\ of\ the\ base$$

$$\theta_b is\ the\ current\ direction\ of\ the\ base$$

$$R_w\ is\ the\ radius\ of\ the\ wheel$$

$$C_l\ is\ the\ base\ chassis\ width$$

then

$$\theta_r = atan2((g_y - b_y, g_x - b_x))$$

$$\theta_m = \theta_r - \theta_b, is\ the\ angle\ to\ move\ base\ by$$

$$\theta_w = \frac{\theta_m C_l}{2R_w}, is\ the\ angle\ to\ move\ each\ wheel\ by$$

Similarly for forward movements by length (l)

$$\theta_w = \frac{l}{R_w}$$

Trajectory Generation

For a desired movement of a robotic system, the programmer often wants a set path which the robot should follow to move between two points. The same applies to the joints of the robot, where the robot moves within the joint space with a desired path. Most often the need is to fluidly move the robot from one configuration to another, thus it reflects as smooth transition of various joints of the robot. A method of obtaining fluid motion is to drive the robot joints from initial to final positions along a cubic spline with velocity set to zero at the end points. Each joint of the robot moves along a cubic spline which is based on the following equations:

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

$$a_0 = \theta_0$$

$$a_1 = 0$$

$$a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0)$$

$$a_3 = -\frac{2}{t_f^3}(\theta_f - \theta_0)$$

Where:

$$\theta_0, \theta_f \; are \; the \; intial \; and \; final \; joint \; angles \; respectively$$

$$t_f is \; the \; time \; taken \; to \; move \; from \; intial \; to \; final \; angle$$

All the above calculations can be done on joint space by assuming $\theta = [\theta_1, \theta_2, \dots, \theta_n]^T$

Camera Balance Checking

Camera balance checking was done in order to comment on the stability of the robot. The camera is assumed to have a certain weight and the further away it is from the base, the further would be the C.G. of the robot from the center of the base, directly impacting its stability. The traditional method to calculate the location of the camera would had been to use the forward kinematics transformation matrices to find the location of camera in the base frame of robot. As mentioned previously, although such methods are available in literature, to ease development process, features inherent to OpenGL were used. While drawing the camera object, OpenGL can be used to compute the transformation matrices yielded by the parent objects. The transformation matrix can be easily used to compute the required locations.

Software Architecture

The robot simulation software is built on top of OpenGL library, with additional features being provided by GLUI. OpenGL is a cross platform application programming interface which allows programmers to interact with graphical processing units. The library consists of many fundamental graphical features such as capability to impose transformations, draw fundamental graphical objects, manage object project from 3D to screen etc. It is assumed that the reader is somewhat familiar with the OpenGL package. As any standard OpenGL based application, the software performs some standard initialization procedures such as for the models, GLUI, display settings, following which the software enters the main OpenGL suspension loop from whereon all the actions are performed through the callbacks evoked either through GLUI or input devices such as keyboard or mouse. Majority of the work discussed here would fall under the callback scheme and code structure developed on-top or in addition to OpenGL for this project.

Call Back Scheme

The software is managed through a callback scheme through GLUI. The GLUI contains the user interface which the lets the user pick between different control strategies for the robot. The GLUI uses “live variables” only for the world view, while all other options are actively evaluated by the software to practice control strategy choices between robot joint angles or tool frame, and the option of animating the path trajectory of the robot. A descriptive flowchart of the scheme is given in figure 12.

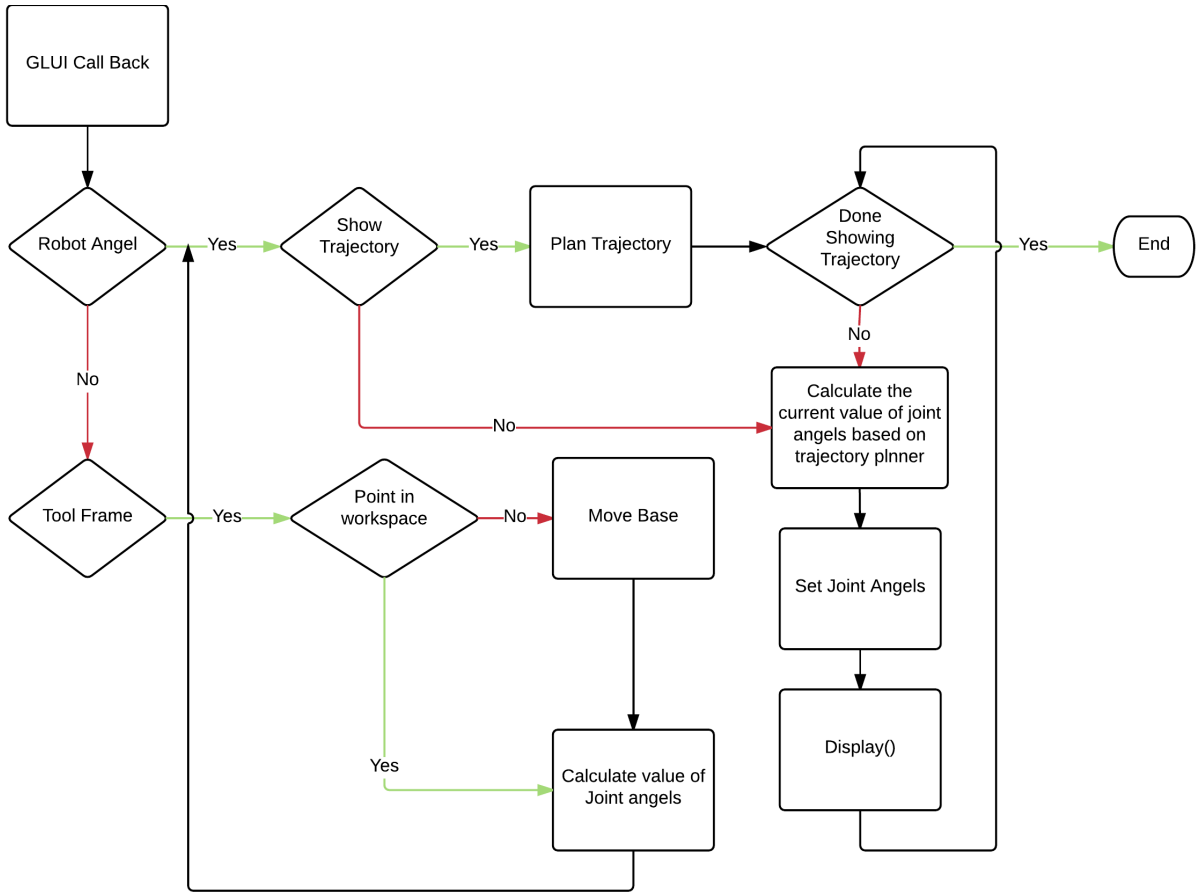

Figure 12: Base software architecture flowchart for robot and environment simulation

If the user chooses not to animate the robot, the final angle are directly set, OpenGL architecture evoked, and the new orientation of the robot displayed. In case animation is chosen, the callback works as a blocking function with a set timer. The callback plans the trajectories to reach the final joint space location in the set time. In case the user chooses tool frame, inverse kinematics is performed to find the required joint angles which are fed into the existing code mechanism of joint control. As such, the software only contains a singular robot instance, which is managed by a "robot controller" through the joint planning interface. The user can chose to facilitate the joint planner by adding inverse kinematics on top, thus effectively creating a top to bottom single channel software with removable parts.

Main Classes

1. Heirs

Heirs is the bottom most class responsible for displaying the composite objects such as robot and articulated human body. The class is built on top of OpenGL framework which allows for "inheritance of frame transformations". The heir class allows the transformation of a parent be imposed on a child, thus the child need only be translated and rotated with respect to the parent with no information required regarding the transformation of the parent with respect to world or another object. The Heir class is

based on a tree system, where each element can be a parent or a child, parents can have multiple children but a child cannot have multiple parents, thus warranting the fact that closed loop chains would not be drawn.

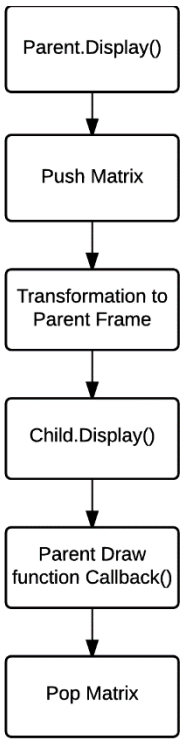

Figure 13: Heir::display() function flowchart

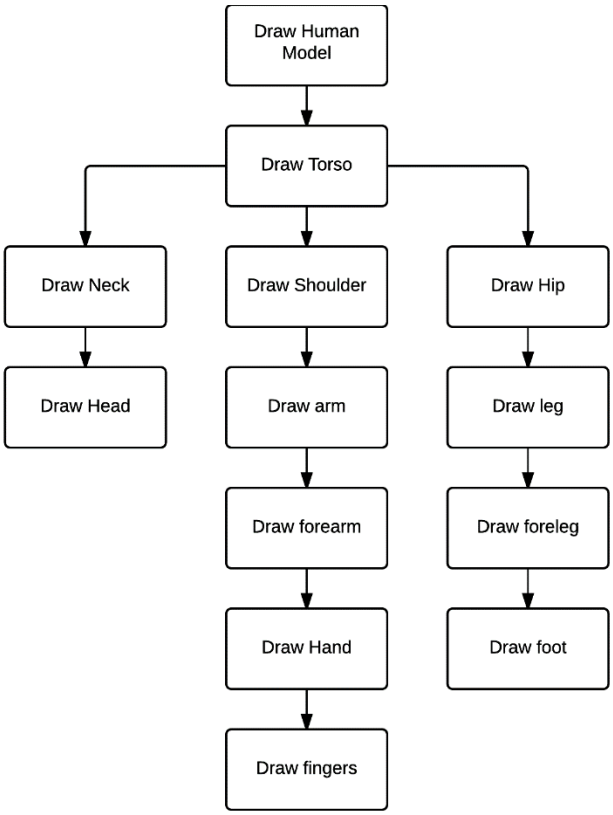

Figure 14: heir class instances and their relations for articulated human model

2. Human Model

The human model is a fully articulated assembly of multiple Heirs resembling a human body. The class itself is built on top of children classes capable of displaying arms and legs, as the model contains repeated limbs for ease of development.

3. Robot Model

The robot model is similar in construction to the human body. The model itself does not contain much information other than the capability to set various joint angles. Majority of the work is done in the “robot controller” part of the software.

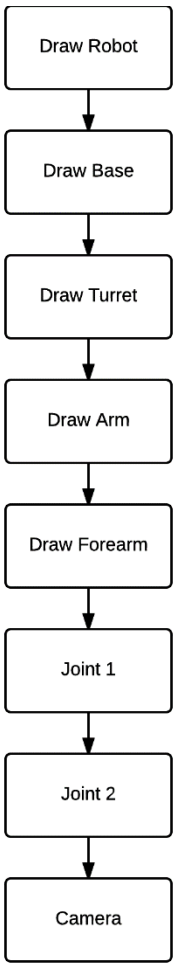

Figure 15: Heir class instances chain for robot model

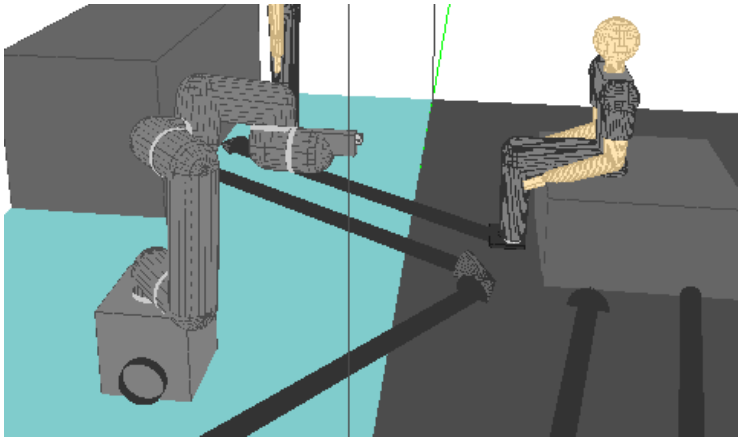

Figure 15: Robot and human model in software

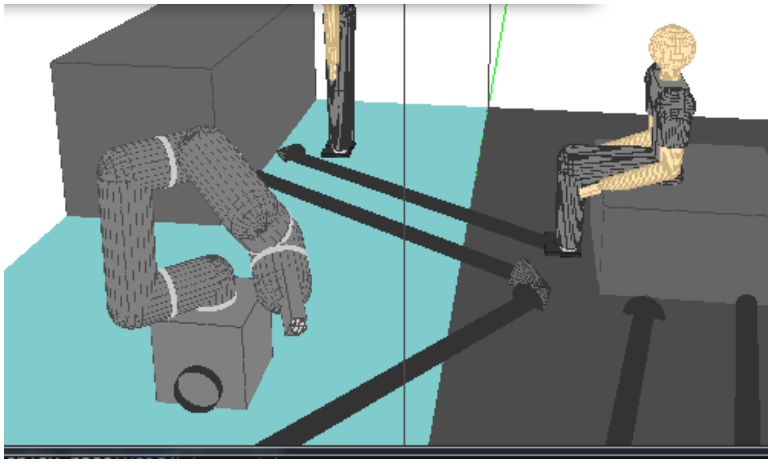

Figure 16: Another view at robot model

4. Robot Controller

The code is a collection of function which act as the front end to the robot model, while the drawing of the robot can be done independent and effectively evoked by any section of the code, the base location, orientation, joint angle of the robot are only done through the robot controller which keeps internal static variable to not only keep track of robot related information but also uses it to simulate movements, calculate workspace boundary violations, perform inverse kinematics etc.

Results and requirements met

The robot has been designed and simulated as explained in pervious parts. A simple scheme of an elderly person and his living environment is simulated as well. The robot is placed in the elderly living environment. It can move around easily and reach different points.

By running the program, user can choose the tool frame information to specify the desired location for the camera, by specifying a single point as the place where the camera should be, two situations can happen:

a. The point is in a robot's work space, so it can reach it by setting the value of its angels equal to the values that has been calculated by inverse kinematic.
b. The point is out of robot's work space, first robot rotates its base to face the point, and moves in a straight line to reach near the point, then it set its angels equal to the calculated joint angels from invert kinematic and place the camera in the desired location

Another option is setting the value of every joint angel, by setting the value of the angels the position of each joint can be calculated by forward kinematic, and joints will place in the calculated locations.

The home configuration of the robot has been defined, by choosing the home configuration the angels of every joint will be set by predefined values.

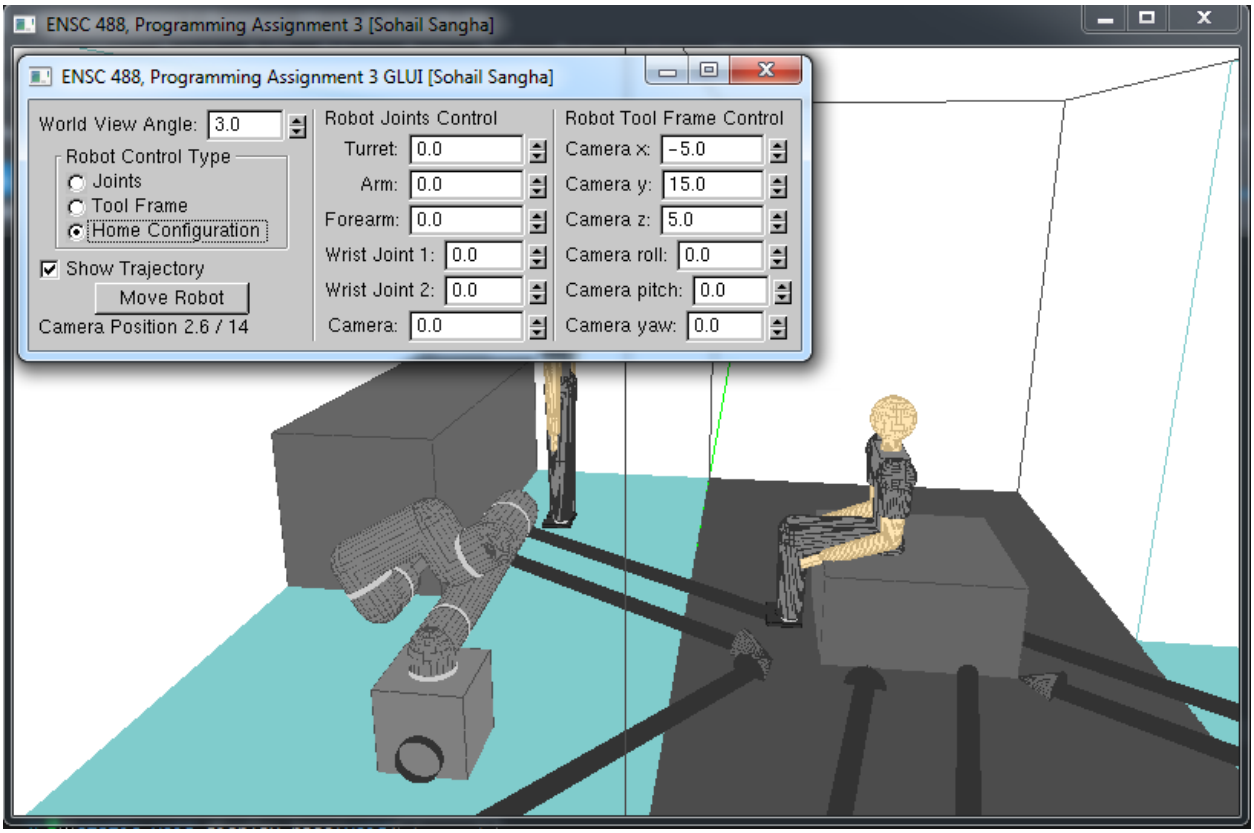

Figure 17: The world view and GLUI control window

By choosing trajectory option, the robot will move from the initial point to the goal point by a cubic trajectory, in this way it will animate the movement of the arm. It calculates the value of the mid-points and move through them as explained earlier.

The camera distance from the base point is calculated, and divided by the length of the arm, which can provide the balance information.

By comparing the explained results by the requirement that have been specified in second section, it can be seen that all of the requirements are met:

a. It can move around easily
b. User can use forward kinematics, set the value of the angels and robot will shape arms in that value, or use the inverse kinematic, and specify the location that the camera should be, and it will place the camera to the desired location
c. The moving between two points is defined by a cubic trajectory , so the moving will be animated, by not choosing the trajectory option, the moving will not be animated and the camera just disappear from the initial point and appear in the final point.
d. The robot has a home configuration
e. By assuming that the tip of the last link is the place of the camera, by dividing the distance of the last links tip from the base from by the length of the robotic arm, the camera balance can be monitored. As long as the value is less than 0.5 it is stable, but when it becomes more than 0.5 it can become unstable.

Limitations and Bugs

Although majority of the requirements set by the project definition were met, given that a robotic system is a very complex mechatronics device, not all of it can be included in the simulation. Some of the aspects which can make the simulated system better can be listed as follows:

1. Calculation of collision detection: currently any of the 3d objects rendered do not detect if they collide with either, meaning a robot can seamlessly walk through a human model. This also speaks for the collision detection for the robot with itself.
2. Angle checks: similar to the problem above, the robot does not perform self-collision checks, thus can contort to any shape.
3. World view: the world view is currently limited to only a point, while the user changes the angle. More complex strategy will be adopted to monitor the robot in various configurations and interactions.
4. Camera location: the camera system location was attempted to be calculated using the OpenGL transformation matrix but it produced problematic results. Currently the geometric solution to the location of intersection point of spherical axis is taken as the camera location.
5. Base Trajectory: a simple approach of rotate and move has been taken, but to move under a more complex trajectory i.e. in arcs more complex simulation strategy will need to be adopted.
6. World view spinner: the spinner is reading in radians whereas all other inputs are in degrees.

User Manual

By running the program two windows open, one of them show the robot in the elderly living environment, and the other one is the graphic user interface (GUI). User can set input values in GUI and see the result in the other window.

Figure 18 shows the GUI window. Each part of the GUI will be explained separately:

1. Word view angel: This value set the observer view point angel, in other words it specifies the angel which observer is looking at the environment with it.
2. Robot control type: There are three types that you can control the robot:
    a. Joints: after choosing this type, you are able to set the joint angel values in robot joints control, part 6. By setting the value for the joint angels, the robotic arm will move to get the desired angels.
    b. Tool frame: if you choose tool frame, you are able to use robot tool frame control, part 7. In this part you can specify the location of the camera in 3D space, and you can set the orientation of it in by camera roll, pitch, and yaw. Then you have to press Move Robot, it is shown in figure 3 by number 4, to move the robot to the desired place.
    c. Home configuration: when you select home configuration, it set the joint angel values equal to predefined values.
3. Show Trajectory: whenever this option is selected, robot will have animated movement, in other words you can see the arm moving from one location to another one.
4. Move Robot: as it has been explained, in tool frame control type, you have to press this button to move your robot to the desired place.
5. Camera Position: it shows the ratio of the camera distance over the arm length. It can be helpful to find out whether camera is in a stable location, or it may fall down. If it is less than 0.5 it is in a stable position, and as it becomes greater than 0.5 the probability of becoming more unstable, will become greater.

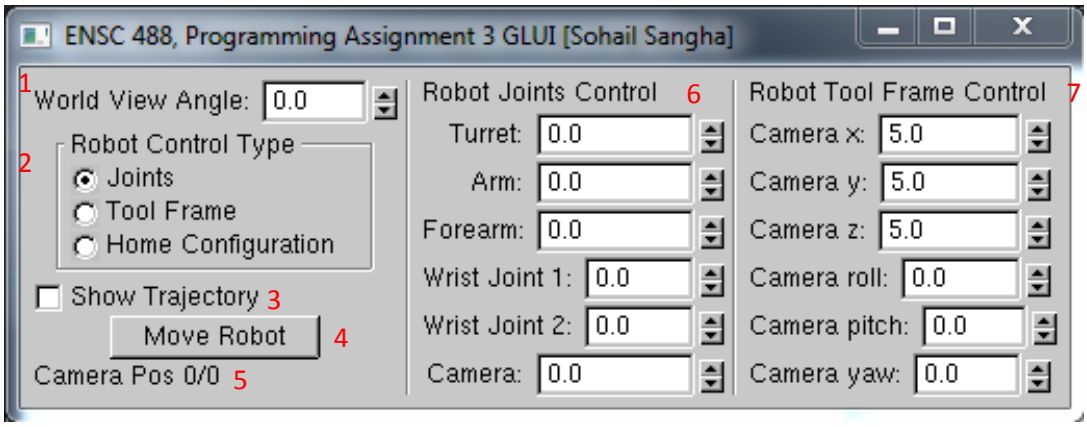

Figure 18: close-up of GLUI control window