Vitual Robot Arm Control Model

D.N.D. Kottege and D.U.J. Sonnadara

Department of Physics, University of Colombo, Colombo 3

ABSTRACT

A Six-axis Virtual Robot arm (SVR) was designed with adjustable kinematic parameters to mimic a 6-axis articulate robotic manipulator with revolute joints having 6 degrees of freedom. The screen based interface was complemented by a custom designed external controller which was connected through the parallel port of the PC. The interface and the main mathematical engine that deals with rigid body transformations has been implemented with Borland Delphi 6, while the external controller was based on the 10-bit A/D channels of the PIC18F458 microcontroller, programmed using the MPASM assembler by Microchip Inc.

The SVR can be manipulated using direct kinematics to change the spatial orientation of virtual objects in 3 dimensions. Picking and placing of virtual objects can be done by using the virtual proximity sensors and virtual touch sensors incorporated in to the jaw design of the SVR. Furthermore, it can be trained to perform a sequence of movements repeatedly, to simulate the function of a real industrial robotic manipulator.

1. Introduction

Modelling and simulation is an important aspect in robotic manipulators as it is in many other fields. It provides a test bed for experimenting with and learning existing and new concepts related to the subject. A virtual robotic manipulator provides a cost-effective and flexible solution compared to a mechanical counterpart. The desire to exploit the possibilities of designing and controlling a robotic manipulator of this nature, led to the concept of the Six-axis Virtual Robot arm and its controller. The Six-axis Virtual Robot arm will be referred to as the SVR while the custom designed external controller will be referred to as the SVRExt throughout this text.

The SVR is a computer program which can be used to simulate the functions of a real robotic manipulator in terms of design parameters, movement and control. It has been designed with adjustable kinematic parameters to mimic a 6-axis articulate robotic manipulator with revolute joints having 6 degrees of freedom.

2. Design and Methodology

In order to determine the kinematic parameters, the D-H representation (Schilling, 1990) and a link coordinate diagram was used. It must be noted that the link coordinate diagram is not unique for a given robotic manipulator. Depending on the initial orientation of the links, the D-H representation will assign different joint angles. The following link coordinate diagram illustrates such a case with the associated kinematics parameters for the SVR.

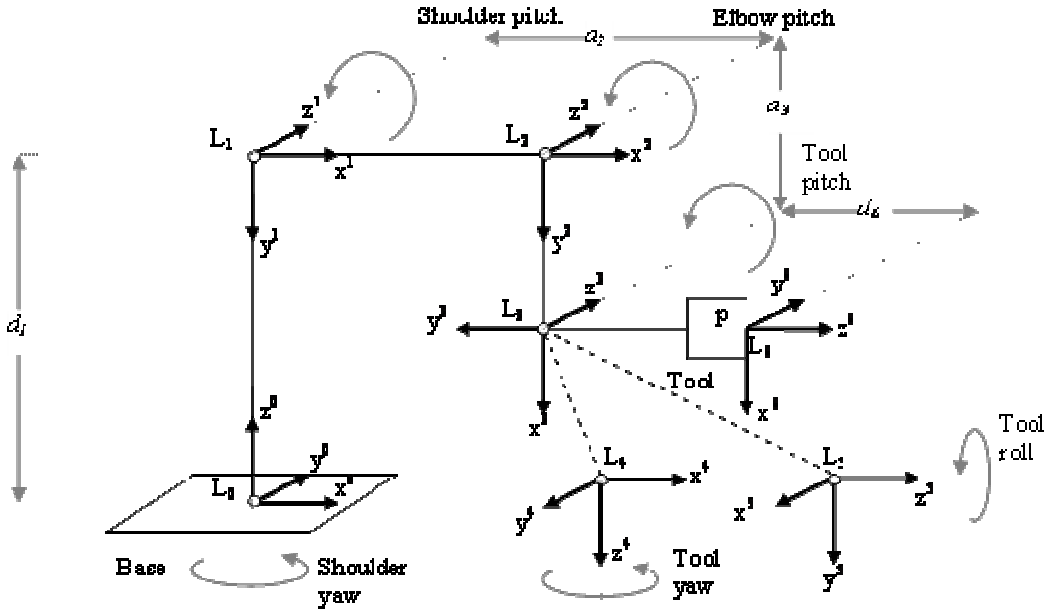

Figure 1: Link coordinate diagram for the SVR with horizontal tool

In the above diagram, the dotted lines connecting the origins of coordinate frames L_3 with L_4 and L_3 with L_5 indicates that the origins of these frames coincide. They have been drawn separately to make the diagram more clear. Table 1 lists the kinematic parameters for each of the 6 joints according to the link coordinate diagram shown in figure 1.

Table 1: Kinematic parameters for the SVR with horizontal tool

Joint No. k	θ_k	d_k	a_k	α_k
1	0	d_1	0	$-\pi/2$
2	0	0	a_2	0
3	$\pi/2$	0	a_3	0
4	$-\pi/2$	0	0	$-\pi/2$
5	$\pi/2$	0	0	$\pi/2$
6	$-\pi/2$	d_6	0	0

This configuration will be referred to as the *soft home position* of the SVR. As it can be seen, three of the four kinematic parameters would always remain constant for a given robotic manipulator. Using this as a basis, just by changing the joint vector consisting of six joint variables, we can manoeuvre the SVR as desired. Furthermore, by changing some of the other kinamatic parameters (design parameters), the SVR can be easily reconfigured without affecting the underlying mathematical model governing the coordinate axis transformations.

The homogeneous transformation matrices necessary for the rotation and translation operations are constructed dynamically at runtime. The 4×4 matrix T is calculated for each rotation and translation operation. The *fillMatrix()* procedure is used to accomplish this task. The parameters *thetaOX, thetaOY, thetaOZ* specifies the rotation angles in

degrees about the x, y and z axes respectively, while *tpx, tpy, tpz* specifies the translation along the x, y and z axes respectively.

Once the 4×4 transformation matrices are constructed using the *fillMatrix()* procedure, they are either post-multiplied or pre-multiplied with each other to form 4×4 composite homogeneous coordinate transformation matrix. Then each 4×1 column vector containing the homogeneous coordinate points that make up the SVR, stored in *base* with structure *pointArray*, is post-multiplied with the relevant 4×4 composite homogeneous transformation matrix stored in *T* with structure *txArray* to yield the new transformed coordinates in the 4×1 column vector *pdash*, again having the same structure *pointArray*.

With all the coordinate transformation matrices and rigid body transformation routines in place, moving the links is a matter of changing the joint vector accordingly. The joint angles q_1 to q_6 are represented by the variables *q[1]* to *q[6]* with a range of -180^o to 180^o. Additional variable *phi* is used to represent the angle of the jaw with 0^o for a fully closed jaw and 90^o for a fully opened jaw.

For every change in the joint vector, the sequence of operations mentioned previously is performed and the resulting points are repeatedly plotted on the computer screen to depict the movement of the links. The main transformation sequence is implemented within the *click* event procedure of the button *btnDrawArm* and in turn it calls the *drawArmSegment()* procedure to handle the plotting of pixels on the screen. The order in which the links are transformed is extremely important. This is due to the fact that when a link is transformed about or along an axis, the coordinate frame attached to its distal end changes as well. The next link connected to this distal end has to be transformed with respect to the changed coordinate frame. The position and orientation of each distal end of the links, i.e. the position and orientation of the coordinate frame attached to each joint need to be taken in to account. This is accomplished by pre-multiplying the transformation matrix of *link k* by the transformation matrix of *link k-1*. Figure 2 below shows the positioning of the links of the SVR for a joint vector of $q = (45^o, 30^o, -75^o, 0^o, 45^o, 0^o)$ with the jaw closed.

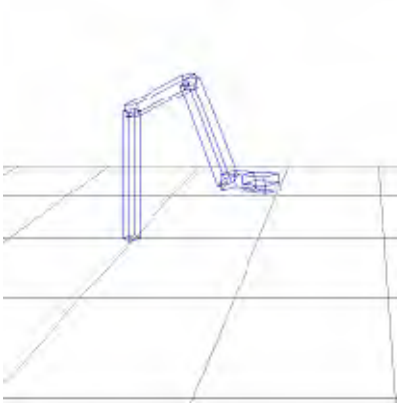

Figure 2: The SVR with a joint vector of $q = (45^o, 30^o, -75^o, 0^o, 45^o, 0^o)$

The screen based interface and the main mathematical engine that deals with rigid body transformations has been implemented with Borland Delphi 6. It is complemented by a

custom designed external controller based on the 10-bit A/D channels of the PIC18F458 microcontroller, programmed using the MPASM assembler, which is connected through the parallel port of the PC. This brings the joint vector manipulating function outside the PC, giving a better feel of controlling the robot arm rather than using the mouse to change scroll bars. Furthermore, it gives the opportunity to change more than one joint variable at a time as opposed to changing one at a time using the mouse and the scroll bars. It must be noted that this particular microcontroller was used in this application, merely for its use in providing A/D channels and not for its other microcontroller capabilities. Full details of its features and capabilities can be obtained from the product data sheet by Microchip Technology Inc (Microchip, 2002). In this application bit 5-3 of PORTB was used to provide the channel address. Contents of ADRESL were directed to PORTC and contents of ADRESH were directed to bit 1-0 of PORTB. PORTA provided the eight analog input channels. Bit 7 of PORTD was used to indicate the current state of the A/D module ($BUSY$ / $\overline{READY}$). The MPASM assembly language coding used to implement this functionality is given below:

MPASM Assembler coding for PIC18F458

```
LIST P=PIC18F458, F=INHX32    ;Directive to define processor and file format
#include <P18F458.INC>        ;Processor specific variable definitions
;Set configuration word to switch off Watchdog timer and Low voltage programming
       __CONFIG       _CONFIG2H, _WDT_OFF_2H & _WDTPS_128_2H
       __CONFIG       _CONFIG4L, _LVP_OFF_4L
              CBLOCK  0x0000
              delay1                ;Variable to implement delay loop
                      tempreg               ;Temporary register variable
              ENDC
              ORG     0x0000
              goto    START
START
              clrf    PORTA           ;Clear PORTA bits
              clrf    PORTB           ;Clear PORTB bits
              clrf    PORTC           ;Clear PORTC bits
              clrf    PORTD           ;Clear PORTD bits
              clrf    TRISD           ;Set PORTD as output
              clrf    TRISC           ;Set PORTC as output
              setf    TRISA           ;Set PORTA as input
              movlw   0xF8
              movwf   TRISB           ;Set PORTB bit0-3,7 as output & bit4-6 as input
MAIN
              movlw   0x80
              movwf   ADCON1          ;Select PORTA as analog inputs + right justify
result
              movf    PORTB,W         ;Read PORTB into W
              andlw   0x38            ;AND W with 00111000, masking bit3-5 of PORTB
              movwf   tempreg
              movlw   0x81            ;Set bit7 and bit 0
              addwf   tempreg
              movff   tempreg,ADCON0 ;Configure A/D control bits and turn on ADC
              bsf     PORTD,7         ;Indicate ADC busy
              movlw   0x10
              call    DELAY           ;Wait for Taq
              bsf     ADCON0,2        ;Start the A/D conversion
LOOP1
              btfsc   ADCON0,2        ;Poll for GO/DONE bit to clear
              goto    LOOP1
              movf    ADRESH,W        ;Put Result high in PORTB
              movwf   PORTB
              movf    ADRESL,W        ;Put Result low in PORTC
              movwf   PORTC
              bcf     ADCON0,0        ;Temporarily turn off ADC
              bcf     PORTD,7         ;Indicate ADC ready
              movlw   0x0A
              call    DELAY           ;wait for Tad
              goto    MAIN            ;goto start of conversion
```

```
DELAY
                movwf   delay1
DEL
                decfsz  delay1,1
                goto    DEL
                return
END
```

The SVRExt is constructed in two boards. The main board contains the microcontroller while the controller board has the potentiometers that act as controls to change the joint vector. Controller boards with different ergonomic configurations can be plugged in to the main board via the 10-pin CON 1 header to provide different handling facilities. The functional blocks of the main board is given in figure 3.

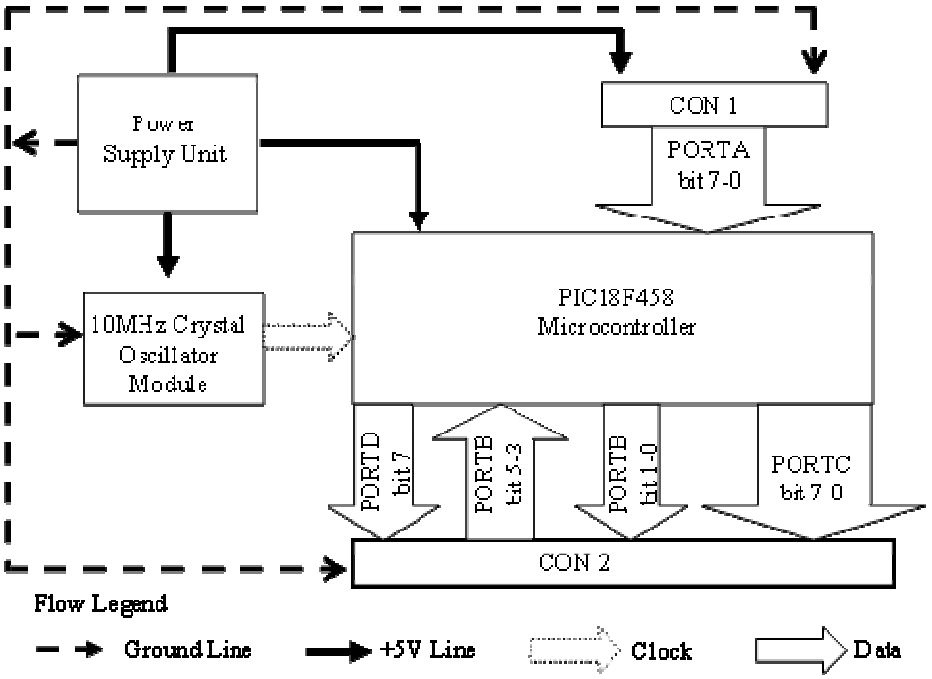

Figure 3: Functional blocks of the SVRExt main board

The SVR can be manipulated using direct kinematics to change the spatial orientation of virtual objects in 3 dimensions. Picking and placing of virtual objects can be done by using the virtual proximity sensors and virtual touch sensors incorporated in to the jaw design of the SVR. Once the object is within reach, the SVR's joint variables (joint vector) can be changed accordingly to manoeuvre the jaw to the vicinity of the object. With the jaw open, an indicator on the interface will inform whether or not the centre of the object (centre of gravity) is within the jaw volume of the SVR, mimicking the function of a proximity sensor. Grabbing the object is only possible when the objects centre of gravity is within the jaw volume. It will also be indicated by the object colour changing from red to green.

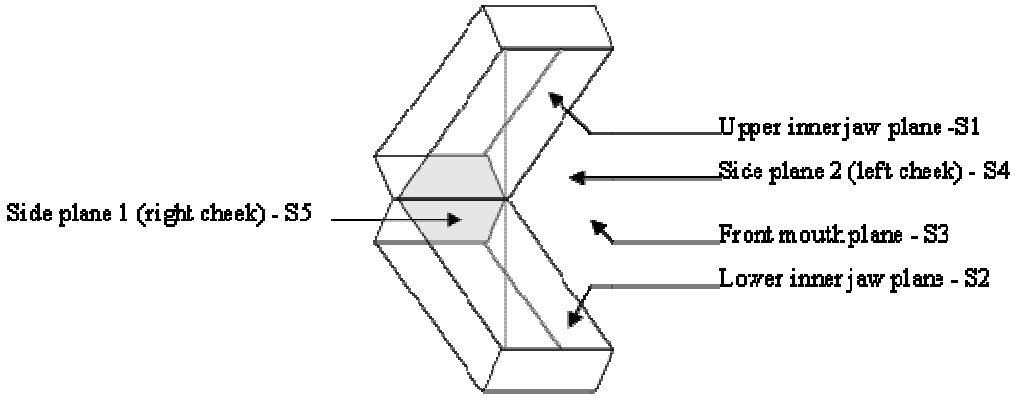

Figure 4: Five planes enclosing the jaw volume

Figure 4 illustrates the various planes involved in making up the jaw volume. When the object is within the SVR's jaw volume it can be grabbed by closing the jaw on the object until the inner jaw planes (S1 or S2) touch the object. An indicator will inform when the object has been touched, mimicking the function of a touch sensor. As soon as the object has been touched the Boolean property *chkGrabbed.checked* will be set to true and the grab operation will take place. Firstly, point *(objx0, objy0, objz0)* substituted in each of the plane equations and the polarity of the function is investigated to perform the proximity sensing function. Secondly, each of the eight vertices of the object cube is substituted in plane equations for S1 and S2 to check whether they belong to the plane. This performs the touch sensing function.

In a physical robotic manipulator, if the jaw continues to close even after the object has been touched by the inner jaw surfaces, it would result in crushing the object causing damage to the object and in some cases, to the jaw as well. The SVR uses the procedure *scbJawChange()* to trap the current action of the jaw (opening / closing) and to prevent the jaw from crushing the object. In order to pick up the virtual object with the SVR, the joint vector needs to be changed until the object centre is within the jaw volume with the jaw opened. Then the jaw needs to be closed upon the object until the inner jaw surfaces touch the object. When the object is touched the grab operation will take place. The object position displayed on the Object panel, the tool position displayed on the Tool position panel, the background grid as well as the virtual proximity sensing and virtual touch sensing indicators described earlier can be used as guides to properly manoeuvre the manipulator arm in to the proper position. Any one of the three available options - internal control through the Joint vector tool panel, external control through a joystick, external control through the SVRExt - can be used to alter the joint vector and open/close the jaw as desired.

Once the grab operation has taken place and the object is lodged between the jaws, further closing of the jaw is prevented by the crush prevention routines mentioned earlier. Now the object can be moved around in the SVR world by once again changing the joint vector as desired. When the object needs to be placed, it can be done by simply opening the jaw until the inner jaw surfaces dose not touch the object, which would release the object from the jaw. With the object placed in the desired orientation and position in space, the SVR can continue to move about as desired. If the object needs to be picked up again, the steps mentioned earlier should be repeated. A sequence of

frames showing a pick up and place operation performed by the SVR is given in figure 5.

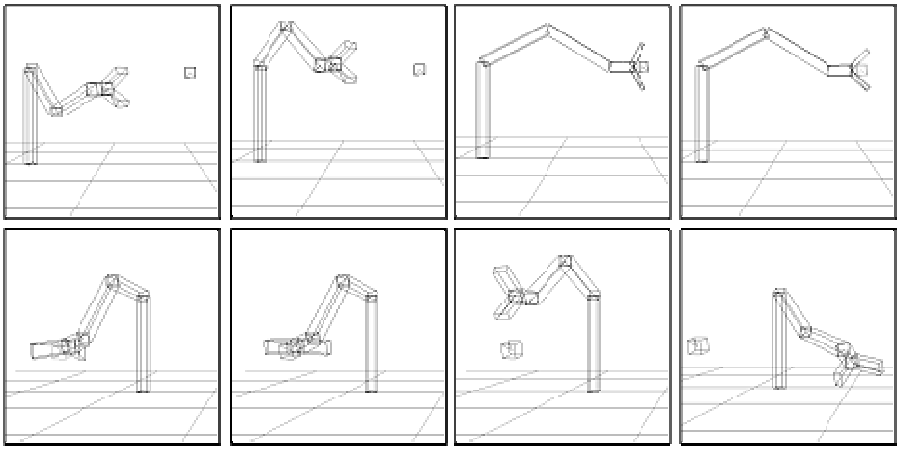

Figure 5: A sequence of eight frames showing a pick up and place operation performed by the SVR

The procedure mentioned earlier in picking up and placing the object can be recorded as a sequence of positions of the SVR with different joint vectors and different jaw positions. The recorded sequence can include, for example, picking up an object, placing it in a specific location in space with a specific orientation, moving away from the object, again moving towards the object and picking it up, change the object's orientation and place it again in the same position or a different position in space etc.. An industrial robotic manipulator may follow a similar sequence in handling parts in an assembly line. Once recorded, this sequence can be played back for a set number of repetitions using the Train positions tool panel. This functionality can be used to simulate the repetitive operations performed by a real robot arm.

3. Comparisons

The SVR was not intended to be a commercial software package but rather a tool for the kinematics analysis and simulate control of a 6-axis articulated robotic manipulator. However, there are commercially available software packages which contain some of the features and functionalities of the SVR. With regard to the 3D modelling capabilities, the commercially available software packages exceed the standards of the SVR by far. Two such packages are *RoboWorks* by Newtonium Software (Newtonium, 2000) and *Yobotics* by Yobotics Inc (Yobotics, 2002). Both these packages are capable of high quality 3D modeling of arbitrary configurations of mechanical manipulators while the SVR is limited to kinematically similar configurations of a 6-axis manipulator.

4. Conclusions

The SVR developed in this work can be used as an educational tool for introducing the basic principles of manipulator design. This becomes a viable option with the pricing schemes for already available software packages with similar capabilities as those

mentioned earlier. However, there are many improvements and additions that can be made to the design of the SVR and its controller to make them more versatile.

One intriguing possibility that emerges from the SVR's object handling capability is its incorporation in to a system with a real manipulator to conduct remote sample handling. Applications of this nature vary from remote space exploration, radioactive sample handling in a shielded laboratory, remote surgery to bomb diffusion. An improved external controller which can be used with the SVR to enhance its object handling capability can be designed by replacing the presently used array of linear variable resistors with a more ergonomic 6 DoF (Degrees of Freedom) controller with rotary type potentiometers.

References

1. Microchip Technology Incorporated. (2002). *PIC18FXX8 Data Sheet: High performance, 28/40 pin enhanced FLASH microcontrollers with CAN*. [DS*41159B.PDF*]. (Arizona)
2. Newtonium (2000). *RobotWorks help file*.
3. Schilling, R. J., (1990). *Fundamentals of robotics: Analysis and control*, Prentice Hall (New Jersey)
4. Yobotics Incorporated (2002). *Yobotics! Simulation and construction set: Users guide.*