Manipulation of Virtual Robotic Arm Using 2D Pointing Device

Masatoshi Sato[1] Hidetoshi Nonaka[1] Johannes Mikulasch[2]
Takeshi Yoshikawa[1] Masanori Sugimoto[1]

[1] Graduate School of Information Science and Technology, Hokkaido University
[2] Technische Universität München

Abstract: In order to operate a multi-joint robotic arm with high degrees of freedom, exclusive controllers with steep learning curves are often required. We believe it is more intuitive to control each joint of the robot on a computer screen using a common computer mouse. In comparison to other input devices, like touch screens or 3D input devices, the mouse can be superior in terms of precision and accuracy. In this paper, we implement a graphical control program showing a 3DCG model of a robotic arm. The joints of this model can be moved by click and drag movements of the mouse. An actual robotic arm is synchronized accordingly to exactly mimic the manipulations of the virtual model.

1 Introduction

In recent years, three-dimensional computer graphics (3DCG) changed from a technology only used in specialized laboratories and companies to a technology that everyone can use.

Today's 3DCG applications are used on personal computers, game consoles, and mobile devices. In those applications it is often required to manipulate an object in 3D space. A typical 3D manipulation task is the free movement of an object in space and has six degrees of freedom (DOF): three to translate an object along the x-, y-, and z-axis, and three to rotate it around those axes.

To accomplish this task, special three-dimensional input devices with 6 DOF like the SpaceNavigator 3D Mouse[1] have been developed, while the research for using a standard computer mouse to do so has been neglected.

In our opinion it is important to research interaction methods to perform 3D tasks with a mouse. The mouse has been a standard input device for decades. People are used to it and can perform 2D tasks, like navigating Graphical User Interfaces (GUI), playing games or even drawing a picture with low effort and high precision. Moreover, computer mice are widely available at an affordable price compared to 3D input devices.

The handling of a robotic arm is an example application where an intuitive interaction method is needed. Typically, robotic arms are directly operated through controllers. Those controllers have one slider or joystick for every joint motor to be controlled. Because robotic arms often consist of five and more joints, it is difficult to master those controllers. An intuitive GUI simplifies the controlling.

In this paper, we implement a prototype interaction method for intuitive and precise teleoperation of a robotic arm using a mouse. The user controls the robot through a virtual model on his computer screen. By clicking and dragging on one of the joints he adjusts its angle. The actual robot is connected to the computer and moves simultaneously. Lastly, we confirm the performance of our method and discuss further ideas.

[1] http://www.3dconnexion.jp/

2 Related Work

We introduce the former research on the manipulation of 3DCG by 2D pointing devices. There are several methods that have been studied to manipulate objects in 3D space using virtual handles. To rotate an object, Chen et al. [1] proposed "virtual sphere" and Shoemake et al. [2] proposed "ARCBALL". Both use a superimposed sphere, on which the user can click and drag to pitch, yaw, and roll the object. To rotate and translate an object, Conner et al. [3] proposed "3D Widgets" and Herndon et al. [4] proposed "Virtual Shadow". However, those methods are not suitable to control a robotic arm regarding all degrees-of-freedom. They only allow to manipulate a single object, whereas a robotic arm requires many objects, its joints, to be controlled.

Operation by selecting and controlling the joints directly is more intuitive than indirect operation by controllers. Hashimoto et al. [5] proposed that users manipulate each part of the robot by directly touching it on a view of the world as seen by a third-person view camera(TouchMe). It was reported that some subjects requested a stylus pen for more precise manipulation. We suggest the mouse is more suitable in terms of the following points:

- Pursuing precision and accuracy
- Eliminating the disadvantage of covering the screen

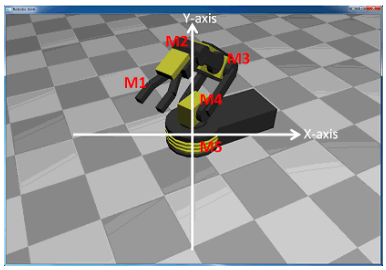

Figure 1: The axes of the screen

Figure 2: The axes of the virtual object

3 Proposed Interface

In the following, we propose our interface to manipulate a robotic arm. A mouse is used to control a virtual robotic arm in a GUI program.

In the application the user can perform three tasks:

- Moving the robot's joints
- Rotating the world
- Zooming in and out

The user moves each joint by clicking on the joint and dragging with the left mouse button. He rotates the coordinate system by clicking and dragging somewhere on the screen (but not on the joint) and moves the viewpoint position by dragging with the right mouse button.

Figure 1 shows the axes of the screen and the position of each joint. Figure 2 shows the axes of the virtual object. The axes of the screen do not change even if the virtual object is rotated, the axis of the virtual object always corresponds to the virtual object. We name the robotic arm's joints M1-M5 from top to bottom.

3.1 Moving the robot's joints

A joint is selected by clicking, and moved by dragging. The joints move differently, as shown in Figure 3.

First, we divide the virtual object into joints and rigid objects. Next, we identify the joint at the position where the user clicked on by using z-buffer. The z-buffer is a memory area storing the depth information of the drawn objects. The z-buffer can return the top-most joint even if multiple joints overlap.

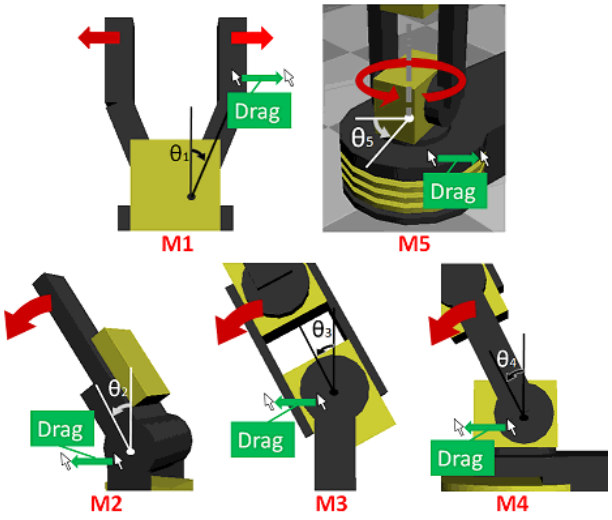

Figure 3: Movement of each joint

Movement of M1 opens and closes the gripper. M1 opens by positive movement along the screen's x-axis and closes by negative movement along the x-axis. M2-M5 rotate by movement along the x-axis. The purpose of joints M2-M4 is to move the arm up and down, while M5 rotates the arm around its base. Depending on the orientation of the operation, the angle $\theta_n (n = 1, \ldots, 5)$ of each joint is increased or decreased by the constant angle α (α is a parameter that controls the angle of each joint) regardless of the position and orientation of the viewpoint.

$$\theta_n \leftarrow \theta_n \pm \alpha \tag{1}$$

However, θ_n should not move over the joint's limits.

3.2 Rotating the world

If the user selects an area outside of a joint and drags the mouse, the world rotates around the x- and y-axis. It only rotates in two degrees-of-freedom, because rotation around the z-axis is not a natural viewpoint movement in the real world. The virtual robotic arm's x-axis is always oriented along the horizontal direction and its y-axis along the upward direction.

The negative rotation around the x-axis is achieved by the positive movement of the screen's y-axis, while the positive rotation around the y-axis is achieved by the positive movement of the screen's x-axis.

Hereinafter, we explain how to determine the angle. The user drags the mouse from point P to point Q on the screen. When the mouse cursor is on the point Q, the corresponding 2D position vector is $\boldsymbol{u} = \overrightarrow{\mathrm{PQ}} = (u_x, u_y)^t$. The angle θ_x around the x-axis and the angle θ_y around the y-axis are given by Equations 2 and 3. However, we limit θ_x to $(0 < \theta_x < 90)$ because we neither want to look under the robot nor turn the world upside down.

$$\theta_x \leftarrow \theta_x - \beta u_y \tag{2}$$

$$\theta_y \leftarrow \theta_y + \beta u_x \quad (3)$$

The parameter β controls the angle of the virtual object.

The rotation-matrices around the x-axis and y-axis are R_x, R_y. The state matrix M of the world is then calculated by Equation 4.

$$M = R_x R_y M_{ini} \quad (4)$$

With M_{ini} being the initial state matrix.

3.3 Zooming in and out

We also want to move the viewer towards the robotic arm and away from it, to create a zooming effect. The user can zoom the scene by right-clicking the mouse and drag along the screen's x-axis.

When the user drags in the positive direction of the x-axis on the screen, the view matrix's z-translation increases. Therefore, the robot model becomes larger. When the user drags in the negative direction of the x-axis on the screen, the view matrix's z-translation decreases. Therefore, the robot model becomes smaller.

The amount of change in the z-coordinate is determined by the speed of dragging, so that the user can zoom in and out faster.

Now we explain how to determine the viewpoint's z-coordinate. The user drags from point P to point Q on the screen. The value of the z-coordinate t_z is then given by Equation 5.

$$t_z \leftarrow t_z + \gamma u_x \quad (5)$$

With γ being a function proportional to u_x. In the result, t_z increases when movement of the cursor gets faster.

The translation-matrix of the z-axis is T_z. The view matrix is then calculated by Equation 6.

$$M = T_z M_{ini} \quad (6)$$

4 Prototype System

We developed a prototype system in which the user controls a robotic arm using our proposed interface. Figure 4 shows a structure chart of our system. Mouse movements control the Virtual Robotic Model on the PC, which is connected to the Real Robotic Arm via an Arduino.

4.1 Robotic Arm

We use ISUPET's "Gripper Arm Robot 40-320C"[1], as shown in Figure 5. It is a wired remote controlled robotic arm with five joints and five motors. The robot has grippers to grab and release something and is able to lift and lower his arm. We use an Arduino Uno[2] to control the robotic arm, as shown in Figure 6. We equipped the Arduino with five motor drivers to control the motors of the robot.

[1]http://www.isupet.co.jp/
[2]http://www.arduino.cc/

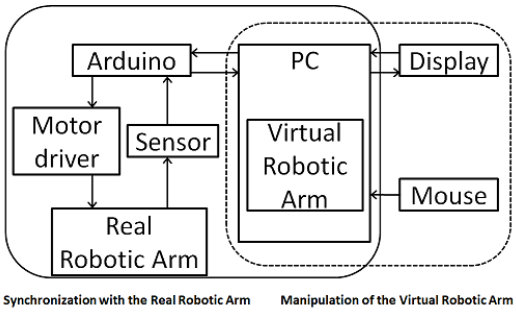

Figure 4: Structure chart

Figure 5: Gripper Arm Robot 40-320C with an exclusive conventional controller

4.2 Application

We developed this application in Microsoft Visual C++ using OpenGL. The environment contains an AMD Athlon(tm) 64 X2 Dual-Core Processor, 1GB memory and the Windows 7 operating system.

First, we created a virtual model of the robotic arm in OpenGL, as shown in Figure 7. The shape and coloring were modeled the real robot.

The parameters for moving the robot's joints and rotating the world are $\alpha = 3, \beta = 1/3$, respectively. The viewpoint's z-coordinate t_z is calculated by Equation 7.

$$t_z \leftarrow \begin{cases} t_z + u_x/20 & \text{if } |u_x| \leq 10 \\ t_z + u_x/10 & \text{if } 10 < |u_x| \leq 30 \\ t_z + u_x/5 & otherwise \end{cases} \quad (7)$$

5 Current Status

5.1 Manipulation of the Virtual Robotic Arm

We confirmed that the three manipulations were successfully accomplished using a mouse. The first manipulation is the manipulation of each joint. The second manipulation is the rotation of the coordinate system. The third manipulation is the scaling of the object by the movement of the viewpoint position. The controlling of each joint is not yet intuitive. De-

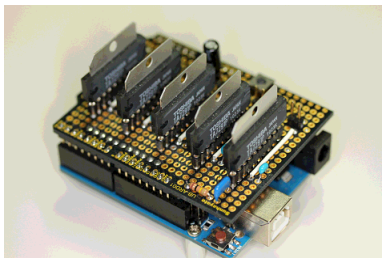

Figure 6: Motor drivers on an Arduino Uno

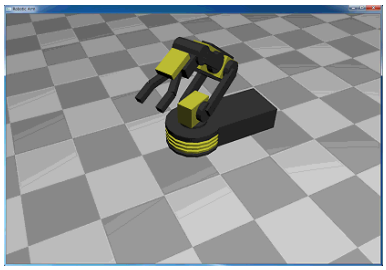

Figure 7: Reproduced 3D model

pending on the position of the viewpoint, the joint moved in a counter-intuitive direction.

5.2 Synchronization with the Real Robotic Arm

We confirmed the synchronization of the actual and virtual robotic arms. When a user manipulates each joint of the virtual model, the corresponding joint of the actual robotic arm moves accordingly.

However, the precision of the rotation angle and the movement speed are still improvable.

6 Conclusion

6.1 Summary

We proposed and implemented an interface to manipulate a robotic arm using a 2D pointing device. Furthermore, we proposed and implemented a system to control the movement of the virtual 3D object and to synchronize the movement of the actual robotic arm. The proposed interface is composed of 2D pointing "click" and "drag" operations. We confirmed that the three manipulations were accomplished using a mouse. The first manipulation is the manipulation of each joint. The second manipulation is the rotation of the coordinate system. The third manipulation is the scaling of the object by the movement of the viewpoint position.

The user manipulates 3DCG by using only familiar 2D pointing devices. Because the selection of each joint is applied directly, an intuitive interface could be accomplished.

Finally, we synchronized our virtual robot model with a real robot.

6.2 Future Work

There are three challenges for future research.

The first challenge is the realization of a more intuitive manipulation. We moved in the opposite direction of the manipulation depending on the direction of the viewpoint. In order to achieve more intuitive operations, transition of the viewpoint should be considered. We need a natural interface viewed from any viewpoint.

Furthermore, we want to use Inverse Kinematics for faster and more intuitive manipulation. By dragging any part of the hand, the other joints rotate appropriately.

The second challenge is to validate the mouse against other input devices, regarding intuitivity and accuracy. For this reason, we have to compare the mouse with different input devices, like 3D mouse and touch interface, using reliable test methods.

The third challenge is an improvement of synchronization with the actual robotic arm. First of all, it is necessary to adjust the movement of the virtual object, and then sent a signal to the actual object. Moreover, it is necessary to perform a comparison test with methods using conventional controllers.

References

[1] Chen, M., Mountford, S. J., and Sellen, A.: A study in Interactive 3-D Rotation Using 2-D Control Devices, *In Proceedings of Computer Graphics*, 22 (4), pp. 121-129 (1988).

[2] Shoemake, K.: ARCBALL: A User Interface for Specifying Three-Dimensional Orientation Using a Mouse, *In Proceedings of Graphics Interface '92*, pp. 151-156 (1992).

[3] B.D. Conner, S.S. Snibbe, K.P. Herndon, D.C. Robbins, R.C. Zeleznik, and A. van Dam.: Threedimensional widgets, *In Proceedings of the 1992 symposium on Interactive 3D graphics*, pp. 183-188 (1992).

[4] K.P. Herndon, R.C. Zeleznik, D.C. Robbins, D.B. Conner, S.S. Snibbe, and A. van Dam.: Interactive shadows, *In Proceedings of the 5th annual ACM symposium on User interface software and technology*, pp. 1-6 (1992).

[5] Hashimoto, S., Ishida, A., Inami, M., Igarashi, T.: TouchMe: An Augmented Reality Based Remote Robot Manipulation, *The 21st International Conference on Artificial Reality and Telexistence*, Proceedings of ICAT2011.