Project Report
Kinect Controlled Robotic Arm
Group 10

Aayush Singhal	Sivakanth Gopi	Sriram Bhargav	Aakash N S
09005041	09005043	09005067	09005069

April 19, 2012

under the guidance of
Prof. Kavi Arya

Embedded and Real-Time Systems Laboratory
Department of Computer Science and Engineering
Indian Institute of Technology
Bombay

Contents

1 Introduction **3**

2 Definitions **4**

3 Hardware Architecture **5**

4 Software Architecture **6**

5 Work Division **7**

6 Innovations, Challenges and Solutions **7**

6.1 Innovations and Challenges 7

6.2 Problems and Solutions 7

6.3 Two Circle Algorithm 8

7 Testing Strategy **9**

7.1 Modular Testing 9

7.2 Combined Testing 9

8 Code Reusability **9**

9 Requirements **10**

9.1 Hardware 10

9.2 Software 10

10 Usage Instructions **10**

10.1 Installation 10

10.2 Usage 10

11 Links **11**

12 References **11**

Abstract

The project aims at developing an simple and intuitive Natural User Interface (NUI) for controlling a robotic arm, using a human arm. All the user needs to do is stand in front of a sensor and move his arm, and the robotic arm tries to imitate the actions and movements of the human arm. As a result, fluid movements which cannot be performed using a controller, are now possible to execute.

The Microsoft Kinect sensor is used to capture skeletal data of the user which is processed and converted into a set of angles which is then sent to the robotic arm over a serial port correction. The algorithm used to generate the angles tries to reach the same position in space as that of the human hand with respect to the human body, relative to the base of the robotic arm. This makes the control fairly intuitive even for first time users.

1 Introduction

Controlling a robotic arm with a traditional controller can be very difficult. It is non-intuitive, complex, and requires training to control the arm properly. It is difficult to control various degrees of freedom simultaneously, and thus the arm cannot be used to perform fluid movements that a human arm is capable of. Therefore, its full power cannot be realized using a traditional controller.

There is a need for a Natural User Interface (NUI) using which a human arm can directly control a robotic arm. Such an interface would make the control the intuitive and easy. No special training would be required to control the arm and even first time users would find it easy to adjust to the controls. The arm can be deployed in remote or hostile places to perform actions and movements that would otherwise be impossible to perform. Robotic arms of various shapes and sizes, suited to various domains, can be controlled easily. The interface, therefore, would open up a plethora of applications.

2 Definitions

Term	Definition
Robotic Arm	A programmable mechanical arm wih similar functions to a human arm
Arm	Unless specified otherwise, refers to the robotic arm
Kinect	A motion sensing input device developed by Microsoft
User	The person controlling the robotic arm
Depth Data	The data produced by the depth image camera in Kinect. Each frame in the stream contains the distance, in mm, to the nearest object at each x and y coordinate in the camera's field of view.
Skeletal Data	The data provided by Kinect about the configuration of a skeleton, as a set of positions that compose the sleleton. The skeleton represents the user's position and posture.
VGA	Video Graphics Accelerator
SDK	Software Development Kit
IDE	Integrated Development Environment
API	Application Programming Interface
GUI	Graphical User Interface
DirectX	A collection of API for handling multimedia related tasks, on Microsoft platforms
Microsoft Visual Studio	IDE for C# programming for the robotic arm and Microsoft Kinect

3 Hardware Architecture

- The Microsoft Kinect Sensor is used to catpure depth data. The sensor is connected to a controller (running Windows 7, or Windows Embedded Standard 7, with the Microsoft Kinect SDK v1 and .NET Framework 4.0 or higher installed), where the depth data is processed to obtain skeletal tarcking data of the user.

Figure 1: Microsoft Kinect

- The Dexter ER2 Heavy Duty Robotic Arm is used as the target arm. It has six degrees of freedom as shown in Figure 2. It has a maximum payload limit of 50 grams. The arm is controlled in real time by sending data from the controller, containing angles of ratations and rotation velocities of the axes, over a serial port.

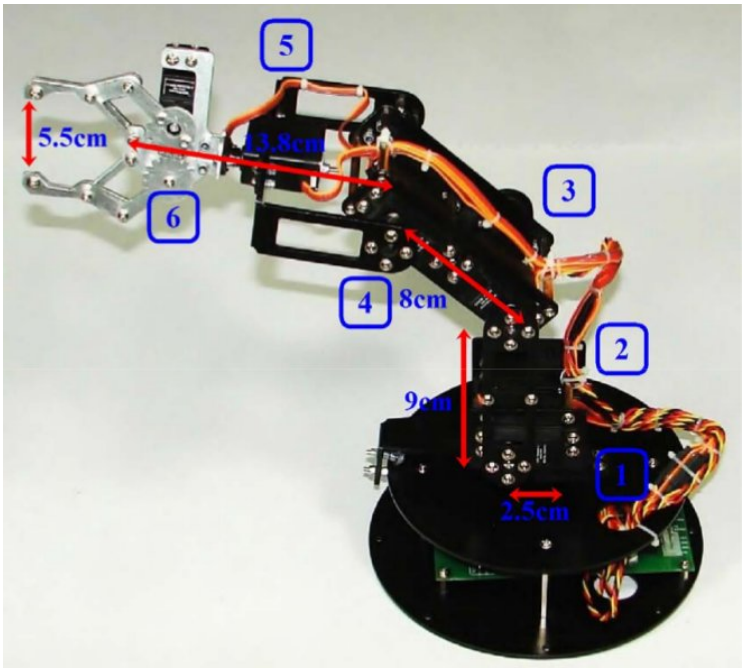

Figure 2: Dexter ER2 Heavy Duty Robotic Arm

Axis Movement
Axis 1: Base rotation
Axis 2: Shoulder rotation*
Axis 3: Elbow rotation
Axis 4: Wrist pitch
Axis 5: Wrist roll

Figure 3: Axis Mappings of Dexter

4 Software Architecture

The application consists of two modules : Main module, and

1. *API for Robotic Arm Control* : This module provides functions for controlling the Robotic Arm in real time over serial port. The .dll file can be used to control the robotic arm from any C# project. Documentation for the library has been provided.

2. *Main Module* : This module uses the above module to control the robotic arm. It provides a GUI for the application. Wrappers for accessing skeletal data obtained from Kinect and using the Two Circle Algorithm have been provided. These can be used from any C# project. Documentation for the module has been provided.

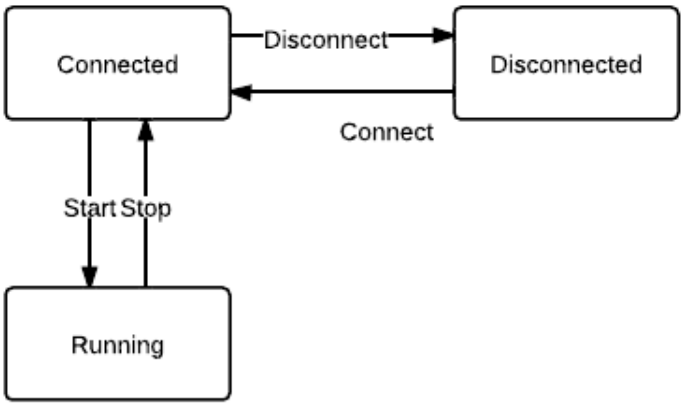

Figure 4: FSM for the Controller

5 Work Division

Work/Critical Tasks	People Responsible
Module for Real Time Serial Port Communication with the arm	Aakash, Gopi, Bhargav
Module for Handling Sekletal Data and Performing Calculations	Aayush, Gopi, Bhargav
Design and Coding of the Two-Circle Algorithm	Gopi, Aayush, Aakash
Graphical User Interface (GUI) for the Application	Bhargav, Aayush, Gopi
Documentation	All

6 Innovations, Challenges and Solutions

6.1 Innovations and Challenges

- Developing an API for controlling the robotic arm.
- Designing an intuitive control mechanism for the user.
- Developing an algorithm for calculating the angles as a continuous function of the positoion of the human arm.
- Implementing averaging features to remove deviations in the skeletal data obtained from Kinect.

6.2 Problems and Solutions

- API/Hardware manual for robotic arm was not available.
  - decompiled the robotic arm control GUI, read and understood the code and developed an C# API/library which future projects can use.
- Kinect cannot capture finger and wrist movements
  - use the left arm to intuitively control the wrist roll and pinch motors
- Kinect data contains many disturbances that lead to the arm deviating unnecessarily
  - used the median of the last few values to remove extremes. Number of samples used shouldn't be too less or too high, 10 seemed a good value after testing
- Human arm and the robotic arm are structurally very different
  - developed the Two Circle Algorithm algorithm for mapping human arm positions to the positions of the robotic arm. This was the most difficult part.

6.3 Two Circle Algorithm

We aim to keep the hand of the robotic arm in the same position relative to its base as that of the human hand relative to the body. The angles at each axis of the robotic arm are generated by the algorithm.

The key challenges faced while developing the algorithm were:

- The angles found by the algorithm should continuously vary with the position of the human hand, to ensure that the movement of the robotic arm is smooth.
- The algorithm should find a solution whenever it exists.
- The robotic arm is fixed to its base, which is in a horizontal plane, whereas the human arm is connected to the body, which is in a vertical plane. Therefore, a direct shoulder to hand mapping is counter-intuitive.

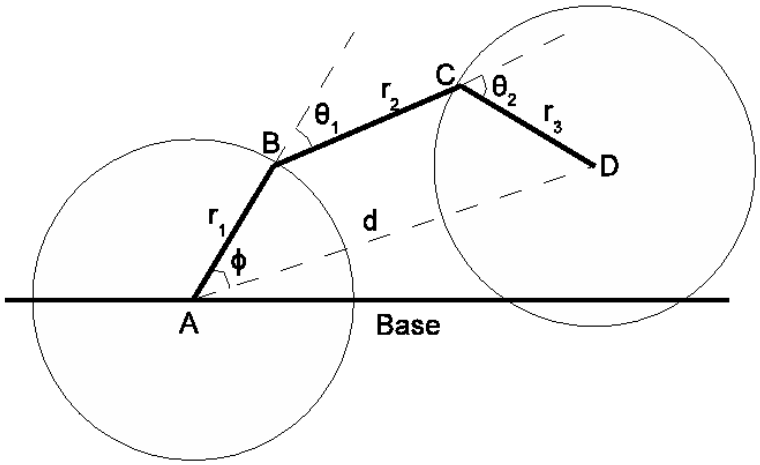

Figure 5: Two Circle Algorithm

Following is a breif description of the algorithm :

- Rotate the base angle so that the robotic arm lies in the same vertical plane as the point we intend to reach.
- In this plane the arm looks as shown in Figure.
- We should find ϕ, θ_1, θ_2 under the constraints are $AB = r_1, BC = r_2, CD = r_3, -\frac{\pi}{2} \leq \phi, \theta_1, \theta_2 \leq \frac{\pi}{2}, AD = d$.
- Observe that minimum value of BC occurs when $\theta_1 = \theta_2 = 0$ and maximum value of BC occurs when $\theta_1 = \theta_2 = \frac{\pi}{2}$(when BC is the direct common tangent for the two circles).

- So there will always exist a solution where $\theta_1 = \theta_2 = \theta$ and $0 \leq \phi, \theta \leq \frac{\pi}{2}$.
- Moreover BC will be a monotonic and continuous function of θ, so we can do a binary search for the solution where $BC = r_2$.
- If $r_2 > maxBC$ then $\theta = \frac{\pi}{2}$ and if $r_2 < minBC$ then $\theta = 0$.

7 Testing Strategy

7.1 Modular Testing

- Testing the API/library for real time serial port communication with the arm.
- Microsoft Visual Studio Professional 2010
- Kinect for Windows SDK v1
- .NET Framework 4.0 or higher
- FT232 USB to Serial Converter (provided with the source)
- Testing the skeletal data obtained using Kinect.
- Testing the calculated angles and lengths required for the Two Circle Algorithm.
- Testing the output of the Two Circle Algorithm.

7.2 Combined Testing

- Testing the intuitiveness of the control by asking subjects to try to pick and place objects using the arm.
- Testing the GUI for application, which is controls and connection various parameters.
- Testing the effect of modifying various parameters of the algorithm and selecting the ones that give the best results.

8 Code Reusability

- Modules independent of each other, hence can be changed without changing others.
- Developed an API/library for robotic arm control which future projects can use.
- The algorithm for finding the arm angles also has wrappers which can be reused.

9 Requirements

9.1 Hardware

- Microsoft Kinect Sensor
- Dexter ER2 Heavy Duty Robotic Arm

9.2 Software

- Microsoft Visual Studio Professional 2010
- Kinect for Windows SDK v1
- .NET Framework 4.0 or higher
- FT232 USB to Serial Converter (provided with the source)

10 Usage Instructions

10.1 Installation

The application is developed to run on the Windows platform only. Installation guide screencast is provided. The following dependencies need to be installed :

1. Kinect for Windows SDK v1
2. .NET Framework 4.0 or higher
3. FT232 USB to Serial Converter (provided with the source)

Additionally, Microsoft Visual Studio Professional 2010 is required to modify/debug the source.

10.2 Usage

1. Connect the Kinect sensor to a power source and to the computer.
2. Connect the Robotic Arm to a power source and to the computer. Press the Reset button on arm to reset the arm.
3. Run the executable 'KinectControlledRoboticArm.exe'.
4. Select the COM port to which the arm is connected from the drop-down menu.
5. Click on the Connect button.

6. Ask a friend to stand in front of the Kinect and make sure his skeletal frame is detected. This can be seen in the video feed from Kinect.
7. Change the elevation of the Kinect, if required, to make sure his arms are in the field of view of the Kinect even when his arms are completely stretched in any direction.
8. Click on the Start button.
9. The robotic arm can now be controlled using your right arm.
10. The gripper and wrist roll can be controlled using the left arm.
11. Click on the Stop button to stop controlling the arm.
12. Click on the Disconnect button to disconnect the arm from the computer.

11 Links

- Project Home Page
  `https://github.com/aayushssinghal/Kinect_Controlled_Robotic_Arm_group10_cs308_2012`
- Hardware Setup Guide
  `http://www.youtube.com/watch?v=o0-Mywjv-kc`
- Software Setup and Usage Screencast
  `http://www.youtube.com/watch?v=G3VrGSS2STc`

12 References

- Kinect for Windows SDK v1 : Resources and Documentation, Microsoft Corporation
  `http://www.microsoft.com/en-us/kinectforwindows/develop/resources.aspx`
- Kinect SDK v1 Quick Starts, Channel 9, MSDN
  `http://channel9.msdn.com/Series/KinectQuickstart`
- Dexter ER2 Robotic Arm User Manual, Nex Robotics, Version 1.00, April 2011
  `http://www.nex-robotics.com/products/robotic-arms-and-grippers/dexter-er-2-heavy-duty-html`
- AVR AtMega2560 Programming Manual, Atmel, May 2011
  `http://www.atmel.com/Images/doc2549.pdf`