An integrated system design interface for operating 8-DoF robotic arm

Vishal Gattani1* and Madhav Rao2

1Research Associate at SARL, IIIT-Bangalore, (vishal.gattani@iiitb.ac.in) * Corresponding author
2Faculty affiliated to SARL, IIIT-Bangalore, (mr@iiitb.ac.in)

Abstract: This study examines the system integration of a game engine with robotics middleware to drive an 8 degree of freedom (DoF) robotic upper limb to generate human-like motion for telerobotic applications. The developed architecture encompasses a pipeline execution design using Blender Game Engine (BGE) including the acquisition of real human movements via the Microsoft Kinect V2, interfaced with a modeled virtual arm, and replication of similar arm movements on the physical robotic arm. In particular, this study emphasizes the integration of a human “pilot” with ways to drive such a robotic arm through simulation and later, into a finished system. Additionally, using motion capture technology, a human upper limb action was recorded and applied onto the robot arm using the proposed architecture flow. Also, we showcase the robotic arm’s actions which include reaching, picking, holding, and dropping an object. This paper presents a simple and intuitive kinematic modeling and 3D simulation process, which is validated using 8-DoF articulated robot to demonstrate methods for animation, and simulation using the designed interface.

Keywords: Robot Simulation, Robot Arm, Motion Capture, Game Engine, Visual Servoing.

1. INTRODUCTION

The development of mechatronics has promoted the rapid improvement of human-robot interaction (HRI) leading many researchers and robot engineers to implement upper limb humanoid systems with prosthetic hands [1]. The collaborative robot systems, as opposed to autonomous systems, work within the controlled human environment that requires regular human intervention [2].

While the human arm is highly dexterous and a complex musculoskeletal structure, it is capable of performing movements including reaching, grasping and manipulation and it is considered an essential component for human interactive robots in a collaborative engagement. The design and development of a biomimetic arm depends on complex parameters like the number of degrees of freedom (DoF), working envelope, and working space that the arm covers, kinematics, payload, speed and acceleration, accuracy, and repeatability [3]. For a human operator to maximize the potential of an advanced robotic arm, the implementation of various innovative control strategies remains a key challenge in the field of humanoid robotics.

Modeling and simulation in general offers a testbed for experimenting and analyzing the behavior of the robotic system. The availability of several game engines such as Unity and Blender Game Engine (BGE) enable virtual simulations to estimate the working of the manipulator in a virtual scenario and provide tools to generate human-like behavior for robotic systems [4]. Therefore, the implemented system integrates the open source computer animation program Blender. In [5], the BGE is used as both, a visualization tool and as a virtual lab (simulator), for the purpose of markerless human motion capturing in the first place but gradually more and more as a full-fledged component in a robot simulation and programming environment.

Markerless human motion capture and depth camera tracking has become popular due to inexpensive entertainment devices such as Microsoft’s Kinect sensor, whose development kit provides skeletal tracking algorithms with the camera at a low cost compared to the marker-based motion capture which has the additional overhead of preparation time. Designed to track the front side of the user [6], the Kinect’s algorithm [7, 8] reduces the depth image to a representation of a stick skeleton estimating a total of 25 joints (Figure 3) making real-time skeletal tracking very accessible, non-invasive and it is shown to have a high correlation with the marker-based motion capture systems [9]. However, it is susceptible to noise and suffers the same occlusion disadvantage as marker-based capture systems to a greater degree. [10] examined the accuracy of Kinect depth data for static objects, showing that the average error of the depth measurement ranges from a few mm up to about 4 cm at the maximum range of 5 m and suggested that the data should be acquired within 1–3 m distance to the sensor for mapping applications. [11] proposed a method to reproduce arm movements in an upper-body humanoid using a Kinect-based MoCap system. [12] uses the Kinect sensor to recognize different body gestures and generate visual HRI interface without the calculation of inverse kinematics (IK) to make the robot arm follow the posture of the human arm.

The development of a computer puppetry system adopts retargeting [13, 14] mechanism, where in the captured movements of a user are mapped to a target character in real-time [15]. The challenges of mapping a motion from the user to the target character becomes more difficult when the target character and user are of differ-

1This paper is supported by the E-Health Research Center (EHRC) at IIIT-Bangalore, India.
ent size and proportion [16–18], since all aspects of the original motion cannot be preserved. Therefore, the arm model was defined to match the anthropometric measures of the human subject and the arm design from [19], avoiding the retargeting problem (i.e., compensate for body differences).

While there are various commercial robotics simulators already available with the most popular being Player/Stage [20] and Gazebo [21], these projects are currently limited in terms of system simulation of robots in a realistic 3D environment. To overcome this, [22] developed a new robotics simulator based on Blender and Python for high reusability under all kinds of robotics research. Moreover, Blender allows for modelling, rigging, animation, simulation, rendering and compositing of 3D meshes. While Unity was also capable of making custom objects though the object editor, it is not well optimised for creating complex meshes and has a much narrower range of accepted file types when importing non-Unity models.

BGE together with Python scripting language is useful in many aspects of simulation and visualization of cognitive autonomous agents. It can be used as a framework which allows to develop and experiment with building blocks of the cognitive agent memory. Many advantages of using BGE include fast prototyping, realistic physics, and flexibility to model virtual environments. Hence, Blender was chosen as the simulator for this study.

The paper presents the implementation of a telerobotic system driven by BGE and allows users to intuitively control a 3D-printed robotic arm to interact with the surroundings. Particularly, the study incorporates the Inverse Kinematics solver, which permits the use of IK armatures in the Game Engine, useful for simulating robotic arms and humans. The forward and inverse kinematic control methods are showcased for testing and experimentation purposes. The integrated system design includes the usage of Microsoft Kinect v2 to enable easy markerless motion capture to the BGE interface, and replicate the same action on the 8-DoF physical robot. The overall integrated system design is significant to robotics researchers who are at the intersection of cognitive science and human motor control.

One of the main interests of creating new simulator is making it reusable by researchers and engineers, the system design reflected the need for simplicity and intuitiveness while keeping the technical complexity easy to use and understand.

2. SYSTEM DESIGN

In this paper, we propose an architecture designed to meet the following primary goals: (i) ability to generate real-time control outputs for teleoperation control, (ii) ability to reproduce motion capture simulations. The system design, as shown in Figure 1, consists of: (1) BGE interface - for Forward and Inverse Kinematics control, (2) motion capture using Microsoft Kinect V2, and (3) the biomimetic robot arm constituting shoulder, elbow, wrist, and gripper as an interactive platform. BGE is used to create the virtual environment that simulates the real robot and to generate a communication channel between the computer and the robot including motion tracking with the integration of Microsoft Kinect.

The graphical design of the kinematic structure of the human arm, particularly the shoulder-elbow-wrist mechanism, and the human skeleton is modeled in BGE through rigging. The virtual arm was designed with identical bone structure mapping the biomimetic robotic arm, and to enable the transfer of the captured joint angles directly to the physical robotic arm for HRI implementation. The graphically modeled arm deforms based on the transformation of the bones during motion and the magnitude of deformation is controlled using weight painting method, allowing user to decide the level of deformation for animation purposes. The different bones control the deformations on different sections of the arm by introducing differences, represented by the gradient in Figure 2, between weights of the vertices that form the arm object.

BGE’s Python environment is used as an interface for the implementation and execution of forward and inverse kinematic control mechanisms. Also, the interface is designed to capture human motion by employing the Kinect sensor for skeleton tracking, and to apply the captured motion onto the 8-DoF biomimetic robot arm.

The Kinect V2 enables tracking of 25 joints of human body and this data is relayed to BGE using NI-Mate software which is a cross-sensor real-time motion capture suite. These joints are rendered in the BGE as points, referred to as empties shown in the Figure 3 (a). The joint coordinates and orientations in the empties format are mapped to a bone structure created inside BGE interface to further analyze any captured or recorded motion, as shown in the Figure 3 (b). This data is used for reconstruction of the human skeleton allowing real time feedback via tracking and modeling of human motion as shown in the Figure 3 (c). To preserve significant qualities (i.e., arm length) of the animated motion such from motion capture to the virtual arm model in BGE interface, the motion was subjected to joint angles and orientations only instead of their coordinates.

The NI-Mate tool takes real-time motion capture data from an OpenNI compliant device, such as the Microsoft Kinect V2, and configures to a industry standard Open Sound Control (OSC) protocol. OSC is a protocol for networking sound synthesizers, computers, and other multimedia devices and it works by specifying an address and its content. For human motion capture, the address and the contents are the skeleton joint’s name, and position and orientation values respectively. The skeletal tracking information as viewed by the Kinect sensor is communicated by NI-Mate and mapped to the BGE interface model for animation and control of the robotic arm. A representation of the human arm with the connected bones, represented in BGE is shown in the Figure 4. The quality of results were assessed by comparing the motion
The 8-DoF robotic arm comprises of 8 Dynamixel motors as shown in the Figure 5, with each motor responsible for one DoF and their role and range of motion is tabulated in the Table 1. The robotic arm design is more for studying reaching than manipulation [19]. The shoulder-to-elbow complex of the robotic arm is operated by the MX series as they offer more load carrying and current driving capability. The forearm, wrist, and gripper joints which do not require as much power are operated by AX series motors. Therefore, the robotic arm’s weight distribution gradually decreases towards the distal end of gripper. The individual Dynamixel motors allow tuning of an internal Proportional-Integral-Derivative (PID) controller to establish desired parameters in the form of motor speed and maximum torque. The motors provide feedback on the following parameters: goal position set by the interface, present position reached by the motor, instantaneous speed, load applied, voltage levels, temperature of the motor, and current consumed by the motor. All the motors are daisy chained to reduce wiring in the system and are driven by the Arbotix-M micro-controller, which is used as an interpreter of the signals generated for the robot through serial communication. An Arduino micro-controller is configured through I2C communication with the Arbotix micro-controller, to handle the feedback data from the assembled motors.

<table>
<thead>
<tr>
<th>Model</th>
<th>ID</th>
<th>DoF</th>
<th>Range of Motion</th>
</tr>
</thead>
<tbody>
<tr>
<td>MX-106 AT</td>
<td>1</td>
<td>Shoulder Pitch</td>
<td>180°</td>
</tr>
<tr>
<td>MX-64 AT</td>
<td>2</td>
<td>Shoulder Roll</td>
<td>90°</td>
</tr>
<tr>
<td>MX-64 AT</td>
<td>3</td>
<td>Arm Yaw</td>
<td>180°</td>
</tr>
<tr>
<td>MX-64 AT</td>
<td>4</td>
<td>Elbow Pitch</td>
<td>100°</td>
</tr>
<tr>
<td>AX-18 A</td>
<td>5</td>
<td>Forearm Yaw</td>
<td>300°</td>
</tr>
<tr>
<td>AX-18 A</td>
<td>6</td>
<td>Wrist Pitch</td>
<td>124.8</td>
</tr>
<tr>
<td>AX-18 A</td>
<td>7</td>
<td>Wrist Roll</td>
<td>101.95°</td>
</tr>
<tr>
<td>AX-18 A</td>
<td>8</td>
<td>Gripper</td>
<td>74.707°</td>
</tr>
</tbody>
</table>
3. EXPERIMENTS AND RESULTS

The joints’ data is communicated serially to the 8-DoF robotic arm through the BGE environment and the feedback data from the motors are used to validate the proposed kinematic control methods. Two kinematic control schemes: forward kinematics (FK), and inverse kinematics (IK) [23] were employed to drive the robot arm independently, and it is investigated for telerobotic application.

An experiment to apply FK to the robotic arm was performed to sequentially orient the elbow, wrist, and gripper closing in a sequence as shown in the Figure 6 (a,b,c,d), suggesting that the action was successfully replicated via inspection.

An alternate control method involving inverse kinematics (IK) is implemented using BGE’s Jacobian IK solver to compute the pose of the virtual arm. The IK Target bone is manipulated directly to reach the desired position and the bending of the arm at the elbow is controlled by positioning the Pole Target bone. The Pole Target and the IK Target bone can be seen in Figure 4 (a). The IK chain consists of 4 bones from \textit{arma} to \textit{armd} as shown in 4 (b).

Unlike the efforts taken in FK to achieve a desired end effector orientation, the IK target bone was translated to apply transformations to target’s parent bone and its predecessors in series, to follow the end effector. The interpolation was performed linearly and in a Bezier fashion where inserted frames comply with a Bezier curve. The quantitative comparison between FK and IK using the motor feedback on elbow joint is shown in Figure 7. The labels A and B shown in the Figure 7 (a,b) correspond to the elbow actions shown in Figure 6 (a,b). The range of rotation applied on BGE virtual arm shown is similar to that of the robotic arm’s elbow joint as shown in Figure 7 suggesting that there is one to one correspondence between the input from BGE interface to the biomimetic robot arm.

An experiment to explore the telerobotic operation abilities using the BGE interface was performed. The virtual arm was controlled to drive the robot arm to perform the following steps: (1) reaching to the position of the cup with open pose of the gripper and then grasping it, (2) changing its orientation, (3) carrying it, and (4) dropping it at another location. These applications were validated by comparing the motor feedback with the joint orientation data transmitted by BGE. The driven inputs and attained positions were found to be in agreement.
Fig. 9. Telerobotic operation showing picking, holding, carrying, and dropping a cup using IK control method.

Fig. 10. Motion Tracking viewed in BGE using Kinect.

Fig. 11. Image showing screenshots of pre-recorded motion captured positions in the form of skeletal tracking using Kinect sensor, followed by mapping to the BGE interface model, and applying to the biomimetic robotic arm.

The Kinect V2 sensor is integrated to the BGE interface for recording and reproducing the arm motion on the robotic arm. The motion was captured at an operational distance of 3 m away from it, with the user facing the Kinect to avoid occlusion during skeletal tracking. Using NI-Mate, the position and orientation of the joints is communicated to BGE in OSC format where it is decoded and then applied onto a rendition of a human skeleton as seen in Figure 10. The upper-limb motion is recorded by inserting keyframes in BGE following which, the implemented script transmits joint orientation data to the micro-controller. Figure 11 shows the stack of recorded motion rendered on a human skeleton, applied motion onto the modeled virtual arm, and corresponding robotic arm action. The three step process starting from recording the motion to applying onto the robotic arm was validated successfully in the Figure 11 (a,b,c,d) which showcases the sequence of reaching, grasping and dropping of a water bottle being replicated by the robotic arm. Blender also enables the user to smoothen the keyframes before transmission using its graph editor features.

4. LIMITATIONS

While the systems motion is jerky in nature, it is primarily due to the step input response of the system. The servos, while having a resolution of 0.293°(AX) and 0.088°(MX), are actuated for every degree change during the motion to avoid the servos from shutting down due to buffer overflow. This is attributed to the hardware limitations of the Arbotix microcontroller.

The jerky motion can be smoothened over by fine tuning the PID controller. Moreover, the system can be improved by dynamic adjustment of the PID gains in accordance to load conditions which induce time-varying delays between control.

The current robotic arm is more suitable for studying reaching than manipulation. The baud rate specifications for the MX and AX series dynamixels are $8,000 \text{ bps-4.5 Mbps}$ and $7843 \text{ bps-1 Mbps}$ respectively. However, the experiments were conducted at a baud rate of 1 Mbps. Therefore, there is scope for improvement in the performance of the system.

5. CONCLUSION

The integrated BGE system interface can be equipped with control strategies, offered adequate accuracy and can be improved to achieve real-time performances, reproduced human-like arm movements by the robotic arm. The novelty lies in the designed open-source architecture flow to drive the robot arm either via motion capture using virtual reality technology or through the game engine. While there can be performance improvements, the architecture flow successfully executed telerobotic operation when the virtual arm was applied to FK and IK control strategies independently. The integration of the Kinect sensor to capture and apply recorded motion shows the architectures flexibility with external hardware. The interface has shown a capability to track the major motions of the upper limb and is intuitively rendered to the robotic arm with delay from the Kinect’s hardware limitations. This study is considered significant in the human robot interactions where the robot can eventually learn and respond to the human actions in the future.

REFERENCES


