Abstract—In this paper we explore the motion control of a robotic puppet through motion capture data. A motion mapping technique is investigated to map the human motion into marionette motion, and from that calculate the rotation of the servo motors to achieve desired marionette motions. A software was developed to capture human motions utilizing a bend-twist sensor system, and use the motion data to control the robotic marionette by either offline control method or online control method (real-time). We also propose the idea of integrating human into the system to close the control loop. Experimental results show that our motion mapping enables puppet to follow the actor motions with good correspondence.

I. INTRODUCTION

Motion capture has become a popular topic in robotics, animation, computer games, virtual reality and so on. On the other hand, application of humanoid robots in the entertainment field is a fast growing industry. We are motivated by the idea of reality games and performance using entertainment robots, in which for example players use their motion to control humanoids to involve in games. This project explores the systematic motion control of a robotic marionette through motion capture data.

Motion capture systems are measurement devices, which are associated with recording of human motion. Various motion capture techniques have been developed for entertainment and biomedicine applications. There are several techniques for recording motion, such as pattern recognition, image processing, magnetic motion capture, mechanical motion capture, hybrid motion capture, and markerless motion capture.

Hybrid motion capture systems are offspring of two or more of capture technologies. ShapeWrap of Measurand is such a system, which is a combination of an exoskeleton system with fiber-optic sensors and magnetic sensors. The system is composed of 4 bend-twist sensor strips and 3 inertial gyros. The 4 sensor strips are positioned along the two arm and the two legs to measure the translation as well as orientation of the limbs’ links. The three inertial sensors are attached to the head, thorax and pelvis to record the orientation of these points. The sensor data make it possible to reproduce the motions of the user for analysis.

In this project we use the hybrid motion capture system make a robotic marionette mimic the performer motion.

II. RELATED WORKS

A few studies of control humanoid robots using different kinds of human motion capture devices have been presented. A computerized controlled life-size marionette, whose mobility is twelve mechanical degrees of freedom, was introduced by Hoffmann [1]. The marionette is controlled by a mechatronic rucksack. Hemami, et al. [2], proposed a strategy for stability of a marionette under a system of unidirectional muscle-like actuators. The strategy provides positive force and positive input to the actuators which are analogous to functions of the firing rate of natural muscles. Yamane et al. used a Vicon vision system to capture human motion and control the hands of a marionette to perform some dances [3]. A feed-forward controller was applied to prevent swinging problem of the puppet hands.

Using a motion capture system with markers attached on the actor’s body, motion data were recorded to control a 32 DOF humanoid robot to dance a Chinese kungfu, Yang et al. [4]. A framework applicable to the problem of imitating an input motion was proposed with a Fujisu HOAP2 humanoid by Chalodhorn et al. [5]. The framework consists of dimension reduction algorithm for efficient and compact state representation, and learning-based predictive control to produce motions optimizing over expected sensory signals. Pollard et al. [6] used motion tracking data of a human actor to control a Sarcos humanoid robot. In this work, several techniques for limiting human motion of upper-body motions were discussed.

Different from the works reviewed previously, we propose a kinematic mapping technique to map the human motion into marionette motion. The mapping allows us to calculate string space variables to control the marionette to strictly follow actor motions.

III. ROBOTIC MARIONETTE SYSTEM

Robotic marionette system is a physical agent which has enough mechanical complexity and behavior diversity for performing various motions. It includes a skillfully designed 16-motor control puppeteer platform (Fig. 1) and a handmade 32DOF puppet. In the robotic puppeteer system, servo motor-operated strings and mechanisms replace conventional manual tools to provide the necessary freedom of movement for the marionettes. Refer to [8] for more detail.
IV. HUMAN MOTION CAPTURE

The model used to capture the human motion is illustrated in the Fig. 3. Each arm is modeled by 3 links, including upper arm, forearm, and hand. The arms are attached to a rigid-body thorax by 3DOF shoulder joints. Each leg is modeled by 3 links, which are upper leg, lower leg, and foot. Similarly to the arms, the legs are attached to the pelvis, a rigid body. The orientations of the thorax and the pelvis are provided by inertial sensors attached to the rigid bodies.

Fiber-optic sensor tapes can provide bend and twist information of the tapes at any moment [9]. Four sensor tapes are used for two arms and two legs. Each tape is clamped at several predefined points along performer’s limb. The bend-twist data of the tapes make it possible to calculate clamping point positions. Then using inverse kinematics, the bone orientations, joint positions and joint angles are calculated based on the measured bone lengths. These data together with the bone model in Fig. 2 allow us to regenerate the actor motion. The mobility of the model is calculated by Gruebler’s formula: 

\[ F = 6N - \sum_{i=3}^{5} i \cdot j_i \]  

Where, \( N \) is the number of movable bones; \( i \) is the class of the joint, \( j_i \) is the number of joints of class \( i \).

Seen from Fig. 3, the model has 16 movable bones; 11 joints of 3\textsuperscript{rd} class (with 3 DOF); 2 joints of 4\textsuperscript{th} class (with 2 DOF); 2 joints of 5\textsuperscript{th} class (with 1 DOF). Therefore,

\[ F = 6 \times 16 - 3 \times 11 - 4 \times 2 - 5 \times 2 = 45 \text{ (DOF)} \]

V. MOTION MAPPING

This section proposes a technique that maps the captured human motions into desired marionette motion, and then determines the motor rotation to achieve the desired motions. Fig. 3 illustrates the proposed mapping procedure.

**Motion Mapping Procedure**

\[
\left( \begin{array}{c}
\text{Human Motion Capture Data} \\
\text{Adapting} \\
\text{Scale Mapping} \\
\text{Inverse Kinematics Mapping} \\
\text{Forward Kinematics Mapping}
\end{array} \right) \rightarrow \left( q_1^h, q_2^h, \ldots, q_m^h \right)^T \\
\downarrow \rightarrow \left( q_1^s, q_2^s, \ldots, q_n^s \right)^T \\
\downarrow \rightarrow \left( \phi_1^s, \phi_2^s, \ldots, \phi_p^s \right)^T
\]

Where, \( \left( q_1^h, q_2^h, \ldots, q_m^h \right)^T \) is human motion space, indicating the human gesture as a function of time. \( \left( q_1^m, q_2^m, \ldots, q_n^m \right)^T \) and \( \left( q_1^s, q_2^s, \ldots, q_n^s \right)^T \) are desired and actual marionette motion space, indicating the puppet gesture as a function of time. \( \left( q_1^m, q_2^m, \ldots, q_p^m \right)^T \) is string space, indicating how much motors have to pull.

A. Adapting

At first we adapt captured human motion to a feature vector of human motion to eliminate redundant DOFs. This feature vector defines the human motion space variables. Because the robotic marionette system can not perform, for instance, pelvis and thorax motion, those data are eliminated in adapting process to reduce the dimension of the space. Thus human motion space consists of the 3D positions of head, shoulders, wrists, elbows, knees, ankles.

B. Scaling mapping

The human motion space variables are then scaled down to the size of the marionette. The following state-space expression is the core of this scale mapping process.

\[
\{ q^n \}_{32x1} = [T_{mh}]_{32x39} \{ q^h \}_{39x1} \quad (3)
\]

Here, \([T_{mh}]_{32x39}\) is scale matrix, including mapping factors to scale the human motion to desired marionette motion. The factors were experimentally evaluated by the ratios of the measured bone lengths of the marionette and those of the

---

**Fig. 1 Puppeteer platform**

![Fig. 1 Puppeteer platform](image)

**Fig. 2 Human link model**

![Fig. 2 Human link model](image)

**Fig. 3 Motion mapping procedure**

![Fig. 3 Motion mapping procedure](image)
actor (the one who wears the sensor strips). Therefore since the bone lengths of different people are different, the scale mapping matrix must be evaluated again with different actors. This process ends up with the desired marionette motion space variables.

C. Inverse kinematic mapping

Inverse kinematic mapping calculates how much the motors have to rotate to achieve the desired marionette motions from the scale mapping. This section derives the inverse kinematics mapping for the control system.

\[
\begin{align*}
\mathbf{C}_0' - \mathbf{C}_1' &= \mathbf{O}_0' \mathbf{C}_0 - \mathbf{O}_1' \mathbf{C}_1 \\
\mathbf{C}_0' - \mathbf{C}_1' &= \sqrt{(x_{k_0} - x_c)^2 + (y_{k_0} - y_c)^2 + (z_{k_0} - z_c)^2} \\
&- \sqrt{(x_{k_1} - x_c)^2 + (y_{k_1} - y_c)^2 + (z_{k_1} - z_c)^2}
\end{align*}
\]

Thus the inverse kinematics equation for the joint \(i^{th}\) is

\[
q_{i}^0 - q_{i}^1 = \frac{360}{2\pi R} \left\{ (x_{k_i} - x_{c})^2 + (y_{k_i} - y_{c})^2 + (z_{k_i} - z_{c})^2 \\
- (x_{k_0} - x_{c})^2 + (y_{k_0} - y_{c})^2 + (z_{k_0} - z_{c})^2 \right\}
\]

The expression determines the servo motor rotating angle to move a joint from one position to another position.

It should be noted that \(|\bar{x}|\) is the magnitude of \(\bar{x}\)

For motors which control the marionette joints:

Fig. 4 shows a geometrical front view sketch of a certain part of the marionette (in this case it is a left leg), in which \(C\) is the contact point of the string and the pulley, \(K\) is the interested joint, the world coordinate system is attached at the pelvis of the marionette to be compatible of the human model of the motion capture system. From the sketch, by some vector calculus we have the following vector relations

\[
\begin{align*}
&[\mathbf{C}_0' - \mathbf{C}_1'] = [\mathbf{O}_0' \mathbf{C}_0 - \mathbf{O}_1' \mathbf{C}_1], \\
&[\mathbf{C}_0' - \mathbf{C}_1'] = \sqrt{(x_{k_0} - x_c)^2 + (y_{k_0} - y_c)^2 + (z_{k_0} - z_c)^2} \\
&- \sqrt{(x_{k_1} - x_c)^2 + (y_{k_1} - y_c)^2 + (z_{k_1} - z_c)^2}
\end{align*}
\]

Thus the inverse kinematics equation for the joint \(i^{th}\) is

\[
q_{i}^0 - q_{i}^1 = \frac{360}{2\pi R} \left\{ (x_{k_i} - x_{c})^2 + (y_{k_i} - y_{c})^2 + (z_{k_i} - z_{c})^2 \\
- (x_{k_0} - x_{c})^2 + (y_{k_0} - y_{c})^2 + (z_{k_0} - z_{c})^2 \right\}
\]

The above formula calculates how much the swinging servo motors have to rotate to make the elbow to move horizontally from the position 0 to the position 1.

Thus the \textbf{inverse kinematics mapping} can be represented as

\[
\begin{align*}
\mathbf{s}_{16d} &= \{ f(\{m\}_{32d}) \}_{16d}
\end{align*}
\]

Here \(f(\{m\}_{32d})\) is so-called inverse kinematics mapping function which maps the marionette space variables into string space variables. The functions were determined based on the above geometric analysis.

VI. PROGRAMMING ISSUES

A. Hardware architecture

Fig. 6 illustrates the fundamental hardware interfacing architecture of the whole system from the human capture device to the robotic marionette.

The brain of the capture system is a Data Concentrator. This concentrator receives light percentage lost data from the sensor tapes and the orientation sensors. The information is then processed to translate into more understandable data, and fed to the host computer through a LAN cable. The connection via RS232 will transfer the information about the configuration of the individual bend-twist sensor tapes and orientation sensors. Four PhidgetServo motor controller cards are connected to the host computer through a 4-port USB hub to control the servo motors driving the puppet. The controller cards handle the motion of sixteen servo motors using external power supply from AC/DC adapters.
B. Control of the marionette using human motion data

The gesture control of the marionette using human motion data are classified into two categories: offline and online.

For offline control, the robotic marionette is not controlled directly from the human motion capture system. The human motion capture (refer to the flowchart in Fig. 7) and the marionette motion control (refer to Section V.) are two separate processes. They are related through intermediate data files. The motions of the actor are recorded in terms of trajectories of the interested points, such as head, shoulders, elbows, wrists, knees, ankles. Those trajectories are stored into intermediate data files, including files containing trajectory information of arm joints, leg joints, and other joints. The program considers the marionette system as a set of several objects, which are left and right arms, left and right legs. Each object will load the respective data file of trajectories, feed it in to data mapping process, and then follow the desired trajectories.

In online control, the marionette motion control process is embedded into the human motion capture procedure to construct an online process. So the robotic marionette is controlled directly from the human motion capture system. Since the whole online process is controlled by a timer, after an amount of time (sampling time), the timer resumes the activated threads to collect the capture data and make the marionette follow the human motions. By this way online control can avoid the synchronization problem of the object motion, which is a disadvantage of the offline control. The precision of individual object’s position is assured, while sequences of the actor’s motions are strictly followed by the marionette as long as the motion speed of the actor does not violate the speed constraint of the motor.

VII. EXPERIMENTAL RESULTS AND DISCUSSION

A. Offline control

These experiments test the ability of the human capture system to control the robotic marionette system through intermediate files. First, the actor’s motions are captured using the bend-twist fiber-optics sensor system. The motion information includes trajectories of the interested points, such as head, shoulders, elbows, wrists, knees, ankles. The motion data of individual limbs are stored into data files. The program then loads those data files, and controls the marionette to follow the trajectories generated by the files. The motions of the actor and the marionette were recorded during the experiments. Fig. 8 shows the sample results.

Though the experimental results of offline control seem to show that the marionette follows with good correspondence the actor’s motions in terms of performance, there are still some drawbacks of offline control. If we assess the quality of each individual object’s motions, such as arms’, legs motions, the limbs follow the respective motions of the
performer quite accurate. However, if we consider the motions of the entire body, it may not follow human movement well. The limbs’ motions are difficult to be synchronized. Time management is a critical issue in real-time programming and servo control. A sequence of the actor’s motions may not be strictly followed by the marionette due to the inability of motor speed control. The performance of the marionette really depends on the number of running processes in the operating system. In other words, the quality of the marionette’s motions is getting better with less processes working in the operation system.

B. Online control

The actor directly controls the motion of the robotic marionette system by motion capture data. When the motion capture data are available, the code segments that control the marionette are activated while the system is still acquiring the actor’s motion. Therefore data are transferred from the capture system to the marionette system in real-time. Fig. 10 illustrates the sample results for online control experiments.

In online control, the two processes, motion capture and marionette motion control, are integrated into one process. The data can flow from the sensor system to the robotic marionette in real-time without intermediate data files. This approach can minimize the synchronization problem appeared in the offline control. As shown in the Fig. 10, the motions of the marionette and the performer showed good correspondence. Moreover sequences of the actor’s motions are strictly followed by the marionette as long as the actor motion speed does not violate the motor speed constraint.

However, it can be observed that there are differences at some moments between the performer’s and the marionette’s gestures. This is caused by the constraints of marionette’s motion in its work volume. For example, there are some positions that human hands may reach, but the puppet hands can not. Initially, the design of the swing mechanism is to provide the horizontal motion of the puppet limbs. But in practice, the marionette limbs can move horizontally only in a quite limited range. This may also result in the mismatch of the actor’s and the marionette’s postures. Moreover even it’s hard to see from the Fig. 8 and Fig. 10, the swinging problem has negative effect on the quality of the marionette’s motions. This is a critical issue not only of the robotic marionette system, but also of string puppets in general.

On the other hand, the marionette could not catch very fast motions of the performer, for instance, when the actor lifts his arm very fast, the marionette also moves his arm up, but at a slower speed, and jerky motion. This problem may occur because of the limitation in servo motor speed and the processing rate of the computer. Better servo motors and a more powerful computer may solve the problem.

VIII. CONCLUSION

In summary, this paper investigates a motion mapping technique to map human motion into marionette motion, and from that calculates the rotation of the servo motors to achieve desired marionette motions. A software was developed to capture human motions using a bend-twist sensor system, and use the motion data to control the robotic marionette by offline and online control method (real-time).

Viewed from the idea of a reality game, in our system we let the user close the loop. As illustrated in Fig. 9, the player will use his senses (eyes in most cases) to observe the marionette motions. This feedback information will be compared with the input goal of the game in his brain. The brain will estimate how his body should move in order to achieve the goal. The control loop therefore continues. This is the main cause that makes computer game so exciting and addictive to people. The idea of leaving loop-closing job to the player actually integrates human being into the control systems. On the other hand, the idea also brings robots nearer to the sociability.

In the longer term, we are developing a reality game using various entertainment robots controlled by human motion.
We also plan to make the robotic marionette more sociable by research on human-behavior recognition which may allows the marionette to react against different situations while interacting with human.

ACKNOWLEDGEMENT

This project is partially sponsored by Singapore's Agency for Science and Technology, and Research (ASTAR), SERC Grant 0521180050.

REFERENCES