Imitating Human Movement with Teleoperated Robotic Head

Priyanshu Agarwal[1], Samer Al Moubayed[2], Alexander Alspach[3], Joohyung Kim[3], Elizabeth J. Carter[3], Jill Fain Lehman[3], Katsu Yamane[3]

***Abstract*— Effective teleoperation requires real-time control of a remote robotic system. In this work, we develop a controller for realizing smooth and accurate motion of a robotic head with application to a teleoperation system for the Furhat robot head [1], which we call *TeleFurhat*. The controller uses the head motion of an operator measured by a Microsoft Kinect 2 sensor as reference and applies a processing framework to condition and render the motion on the robot head. The processing framework includes a pre-filter based on a moving average filter, a neural network-based model for improving the accuracy of the raw pose measurements of Kinect, and a constrained-state Kalman filter that uses a minimum jerk model to smooth motion trajectories and limit the magnitude of changes in position, velocity, and acceleration. Our results demonstrate that the robot can reproduce the human head motion in real time with a latency of approximately 100 to 170 ms while operating within its physical limits. Furthermore, viewers prefer our new method over rendering the raw pose data from Kinect.**

I. Introduction

Real-time imitation of human head movement is of interest for various applications such as telepresence, teleoperation, and imitation learning. For telepresence, the existing solutions only transmit a video of the person. However, studies have shown that physical embodiment in the form of a humanlike robot leads to more natural and humanlike conversations [1]–[3]. The idea of the TeleFurhat system is that instead of simply transmitting a video of the person, the Furhat robot will be present at the remote end to render the gestures of the person, including his or her head motion. Such a system could result in much more effective and engaging conversations and improve upon current entertainment systems that require real-time interaction with remote persons. Current interfaces for these systems are tedious and cumbersome because they require an operator to use remote controls and joysticks [4], [5]. With our TeleFurhat system, the remote system could be teleoperated using human body motion instead, resulting in more intuitive control.

Imitation of human head movement by a robotic head has been attempted previously. Demiris and colleagues [6] presented a system for deferred imitation of human head movement using optical flow for head pose estimation. However, their pose estimation algorithm suffered from several limitations including lack of translation information as they assumed that optical flow always represents rotation. This resulted in inaccurate rotational pose estimates. In addition, no quantitative results were presented on the performance of the system. Hurring and Murray [7] proposed a system for teleoperating a robot head while the user wears a head-mounted display. However, their pose-tracking algorithm required explicit paper markers to be mounted on the display.

Some prior work has focused on use of the full body pose for imitation learning by augmenting the human user with explicit color patches for real-time tracking [8] or with a motion capture system for offline [9] or real-time tracking [10], [11]. However, such systems are often affected by ambient light and placement of markers, and they require tedious calibration of the system for accurate and repeatable performance, limiting the use of the system to laboratory settings. More recently, Microsoft Kinect has been employed to imitate human arm motions with an anthropomorphic dual arm robot [12]. However, its online trajectory generation algorithm causes significant delay, which adversely affects the system performance in teleoperation because additional delay will be added due to communication over the network. In addition, the authors show that the velocity and acceleration estimates obtained by numerical differentiation are extremely noisy; however, their system does not condition the signal to filter out the noise, which could lead to jerky and unnatural motion.

The major challenges in the implementation of the TeleFurhat system for teleoperation are (i) how to map the human motion to a robot with different dynamics and degrees of freedom (DOFs) (i.e., motion retargetting), (ii) how to perform accurate human head pose estimation, (iii) how to limit the motion for the robot's safety while retaining the essence of the human motion, and (iv) how to ensure low overall latency in the system to enable natural conversation.

Motion retargeting has been employed previously to map an actor's trajectory onto an animated character [13]. Prior work has also focused on rendering human motion capture data on humanoid and non-humanoid characters with significantly different morphologies [14]. The problem is more challenging for teleoperation as it requires mapping the user's input on the robot across a significant distance in real time [15]. Dragan and Srinivasa [16] proposed a method for customizing and learning the retargeting function from online demonstration of the desired pose by the user. In this work, our focus is on the second and third challenge.

[1] Priyanshu Agarwal is with University of Texas at Austin, USA. This work was done when he was at Disney Research. priyanshu.agarwal@utmail.utexas.edu

[2] Samer Al Moubayed is with Furhat Robotics and Royal Institute of Technology, Sweden. This work was done when he was at Disney Research. samer.-nd@disneyresearch.com

[3] Alexander Alspach, Joohyung Kim, Elizabeth Carter, Jill Lehman, and Katsu Yamane are with Disney Research, USA. {alex.alspach, joohyung.kim, jill.lehman, kyamane}@disneyresearch.com

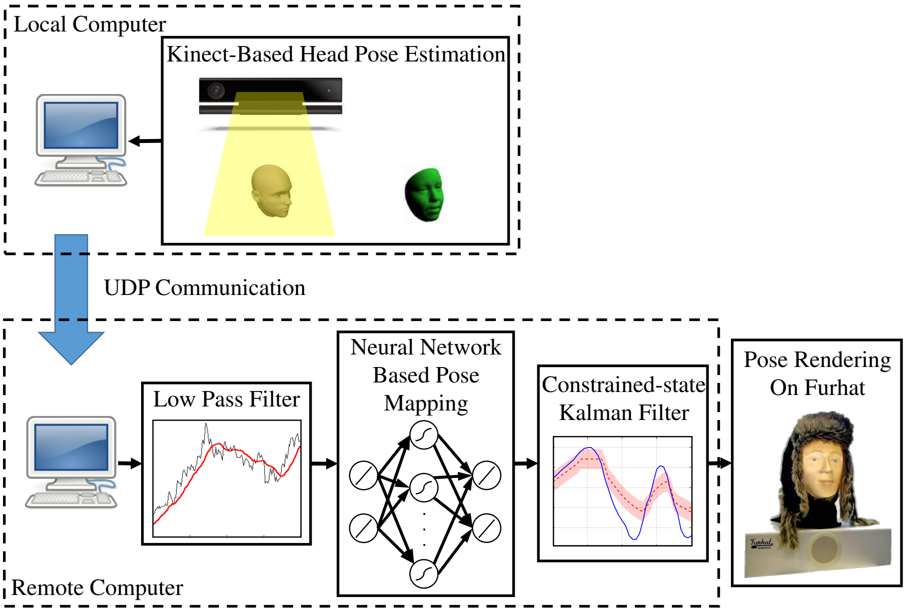


Fig. 1. Overview of the TeleFurhat system developed for teleoperation using the Furhat robot.

In this work, our goal is to build a system for teleoperating the Furhat robot head using human head motion in real time while ensuring the safety of the robot. To this end, we develop the processing framework described below.

II. Overview

The TeleFurhat system consists of a local computer where the user's head pose is estimated and a remote computer to which the Furhat robot is connected (Fig. 1). The head pose estimation is performed by the high-definition face tracking system from Microsoft Kinect 2 sensor API 2.0 [17] at approximately 30 Hz. The estimated head angles (pitch and yaw) are communicated over the network to the remote computer via user datagram protocol (UDP). At the remote computer, the pose measurements are processed to appropriately condition the signal, which is then rendered on the Furhat robot. Our processing framework consists of pre-filtering using a low pass filter, a neural network-based model for mapping the filtered pose measurements to the desired pose, and a constrained Kalman filter to reduce noise and limit the magnitude of the position, velocity and acceleration of the signal using a minimum jerk model.

Furhat is a humanlike, back-projected robotic head that displays facial expression using computer animation as well as pitch and yaw motions of the head in real time [1]. It does not support the roll motion and also has smaller range of motion (28° in pitch and 68° in yaw) than the human head (130° in pitch and 155° in yaw [18]). It allows for natural, humanlike human-agent interaction due to the three-dimensional physical embodiment through its humanlike head and face shape. Furhat also allows for flexible rendering of humanlike facial motion (e.g., eye movements, blinks, eyebrow movements, and lip movements) due to its animated face, enabling it to convey subtle facial expressions and emotions that are limited on robots with hardware-based facial animation. In this work, we omit the animated facial expressions in order to focus on head motion.

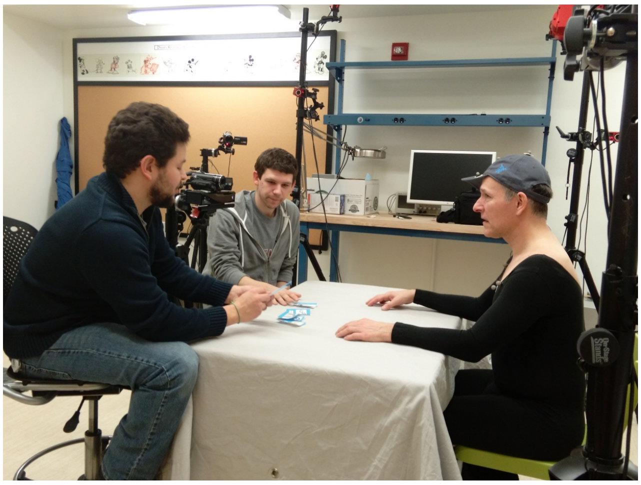


Fig. 2. The setup used for the synchronized Kinect and OptiTrack based head motion capture. The actor interacted with two subjects while performing the tasks to ensure that his gestures are natural.

III. Pose Filtering and Mapping

In this section, we discuss the processing framework implemented for the TeleFurhat system.

A. Motion Capture Data Analysis

First, we collected motion capture data to analyze and model the natural human head motion and establish the

TABLE I

SUMMARY OF THE VARIOUS TASKS PERFORMED BY AN ACTOR TO GENERATE POSES THAT COVER THE POSE VARIATION OBSERVED DURING HUMAN-HUMAN VERBAL OR NON-VERBAL COMMUNICATION. THE TOTAL DATA CORRESPONDS TO OVER 8 MINUTES OF PERFORMANCE AT A SAMPLING RATE OF 30 HZ.

Task	Sample Size
Storytelling	7128
Desert Survival Game	3247
Career Advice to Right Listener	767
Career Advice to Left Listener	951
Expressions	3178

ground truth for benchmarking the performance of the developed algorithms. A professional actor performed several tasks to generate the pose variation observed during human-human verbal and non-verbal communication (Table I, Fig. 2). We recorded synchronized head pose data with an Optitrack motion capture system [19] at 100 Hz, a Microsoft Kinect 2 sensor at 30 Hz, and two HD cameras. The motion capture data sequences were then manually processed to remove any outliers in the pose data and resampled to obtain a synchronized sequence of pose measurements from the two sources. The pitch and yaw angles refer to the angles corresponding to the vertical and horizontal motion of the head, respectively.

B. Pre-filtering Kinect Data

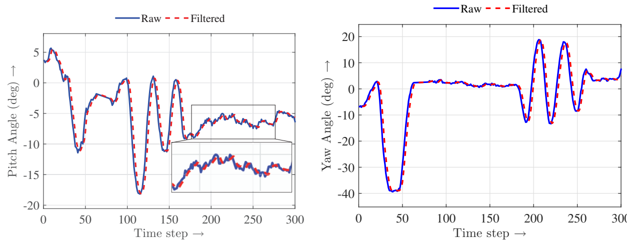
Raw head pose data measured by Kinect typically contain significant noise. To filter out this high-frequency noise, a moving average low-pass filter is designed using filter residual analysis.

First, filter residual analysis is performed to determine the optimal cutoff frequency for the filter. Specifically, it evaluates how the root mean square (RMS) residual (i.e., the discrepancy between the raw signal and the filtered signal) changes with the change in the cutoff frequency of the filter [20]. Results show that the residual drops to 0.1 degrees for both pitch and yaw angles with a cutoff frequency of 5 Hz.

Then, a moving average filter is designed with cutoff frequency $f_{co} = 5$ (Hz). The optimal window size N for the moving average filter with sampling frequency $f_s = 30$ (Hz) is given by (1).

$$N = \frac{\sqrt{0.44294^2 + \left(\frac{f_{co}}{f_s}\right)^2}}{\frac{f_{co}}{f_s}} \approx 3 \tag{1}$$

While filtering removes the high-frequency noise from raw pose data from Kinect, it also introduces a delay of approximately 2 time steps (∼66 ms) as shown in Fig. 3.



(a) (b)

Fig. 3. The moving average filtering results for the actor head angles captured using the Kinect.

C. Neural Network Based Pose Modeling

The Kinect-based measurement system introduces systematic error in the head pose data. The error is both a function of the head pose and the higher order derivatives of the pose trajectory. In contrast, the marker-based OptiTrack motion capture system is more reliable and is less prone to error because it estimates the pose by tracking specialized markers placed at strategic locations on the body using multiple high frame rate cameras. In order to correct the error introduced by Kinect, we learn a model between the Kinect- and OptiTrack-based measurements (Fig. 5). The model could also be learned on any other desired pose data to replace the gestures that could not be effectively rendered on the robot in real time. This could be done by generating custom motion on the robot head with the help of an animator and learning the mapping between the Kinect-based pose estimates and the animated motion. This could also help generate gestures that look more natural on the robot rather than simply rendering the human head pose on the robot accurately.

Neural network regression [21] and Gaussian process regression [22] are two popular techniques for modeling data mapping. Neural network is a parametric model that is relatively computationally inexpensive to evaluate once the network is trained. Gaussian process regression, on the other hand, is a non-parametric Bayesian modeling technique that has much higher computational requirements for inference (exact inference scales with $O(n^3)$, where n is the size of the training sample), especially when the sample size is greater than a few thousand [23], [24]. Because our application requires real-time execution of the framework, we use neural network models. There are also other parametric modeling techniques such as Gaussian mixture regression, locally weighted projection regression etc. However, studies have shown comparable results from these techniques on some problems [25], [26].

The developed neural network model takes a window of the time series for head pitch (θ_{pk}) and yaw (θ_{yk}) data from the Kinect sensor and outputs the corrected current pitch (θ_{pr}) and yaw (θ_{yr}) estimates to render on the robot (Fig. 4). We use linear transfer functions for the input and output layers and hyperbolic tangent sigmoid transfer functions for the perceptrons in the hidden layer. We use hyperbolic

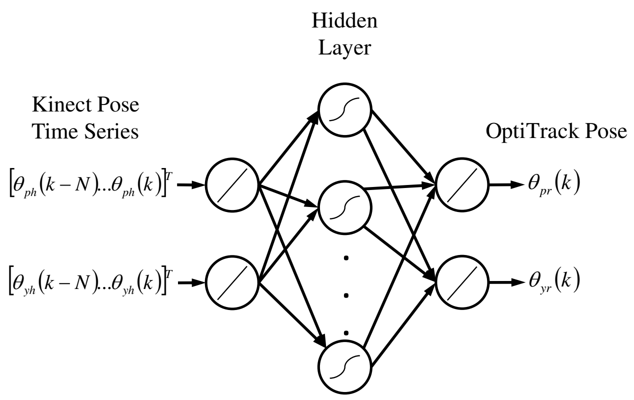


Fig. 4. The neural network model used for learning the pose mapping from the Kinect data to the respective desired robot pose.

TABLE II
THE SAMPLES USED FOR TRAINING, VALIDATION, AND TESTING THE NEURAL NETWORK MODEL LEARNED FOR POSE MAPPING.

	Training	Validation	Testing
Sample Size	10690	2290	2290
Percentage	70%	15%	15%

tangent sigmoid as the transfer function because some studies have shown that networks with these functions are more generalizable [27] and perform better than the networks with other types of transfer functions [28], [29]. We divide the collected pose data into training, validation, and test sets to quantify the performance of the network (Table II). The pitch and yaw values from OptiTrack are also saturated at the joint angle limits of the robot so that the network can learn to limit the output to be within the bounds of the robot's joints. We train the network with different numbers of perceptrons in the hidden layer and past window sizes to select the optimum network for the task. Each network is trained 50 times with random initialization and the best performing network is selected.

D. Post-Filtering

The output from the trained neural network does not guarantee the smoothness of the pose trajectory or ensure that the trajectory has angular position, velocity and acceleration that are safe for rendering on the robot given the limitations on hardware capability. We address this issue by introducing a constrained-state Kalman filter with a minimum jerk model into our framework as a post-filter.

1) Minimum Jerk Model: Coordinated voluntary human arm movement studies have shown that humans follow a minimized jerk trajectory [30], which is also the smoothest possible trajectory [31]. We use the minimum jerk model in the head pose space with added white Gaussian noise at the acceleration level (2) as the underlying model for filtering the pose estimates.

$$\begin{aligned}
\begin{bmatrix}\boldsymbol{\theta}_k\\ \boldsymbol{\omega}_k\\ \boldsymbol{\alpha}_k\end{bmatrix} &= \begin{bmatrix}\mathbf{I}_2 & \Delta T\mathbf{I}_2 & \frac{\Delta T^2}{2}\mathbf{I}_2\\ \mathbf{0}_2 & \mathbf{I}_2 & \Delta T\mathbf{I}_2\\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{I}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{\theta}_{k-1}\\ \boldsymbol{\omega}_{k-1}\\ \boldsymbol{\alpha}_{k-1}\end{bmatrix} + \mathbf{w}_k\\
\boldsymbol{\theta}_k &= \begin{bmatrix}\mathbf{I}_2 & \mathbf{0}_2 & \mathbf{0}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{\theta}_k\\ \boldsymbol{\omega}_k\\ \boldsymbol{\alpha}_k\end{bmatrix} + \mathbf{v}_k\\
\mathbf{w}_k &\sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}\mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2\\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2\\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{Q}\end{bmatrix}\right), \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R})
\end{aligned} \tag{2}$$

where $\boldsymbol{\theta}_k = \begin{bmatrix}\theta_{p,k} & \theta_{y,k}\end{bmatrix}^T$, $\boldsymbol{\omega}_k = \begin{bmatrix}\omega_{p,k} & \omega_{y,k}\end{bmatrix}^T$, $\boldsymbol{\alpha}_k = \begin{bmatrix}\alpha_{p,k} & \alpha_{y,k}\end{bmatrix}^T$. $\boldsymbol{I}_n$ and $\mathbf{0}_n$ represent an identity and zero matrix, respectively, of size $n \times n$. ΔT represents the size of the time step. $\mathbf{w}_k$ and $\mathbf{v}_k$ represent the noise in the process model and the observation model at time step k, respectively. $\boldsymbol{Q}$ and $\boldsymbol{R}$ represent the covariance matrix of the process noise for the angular acceleration and observation noise for the angular position, respectively. $\mathcal{N}(\mu, \Sigma)$ represents a Gaussian distribution with mean μ and covariance Σ.

2) Constrained-state Kalman Filtering: The original Kalman filter, represented by (3), does not constrain the system states in the filter output.

$$\begin{aligned}
\boldsymbol{K}_k &= \boldsymbol{A}\boldsymbol{\Sigma}_{k-1}\boldsymbol{C}^T\left(\boldsymbol{C}\boldsymbol{\Sigma}_{k-1}\boldsymbol{C}^T + \boldsymbol{R}\right)^{-1}\\
\hat{\boldsymbol{x}}_k &= \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{K}_k\left(\boldsymbol{y}_k - \boldsymbol{C}\hat{\boldsymbol{x}}_{k-1}\right)\\
\boldsymbol{\Sigma}_k &= \left(\boldsymbol{A}\boldsymbol{\Sigma}_{k-1} - \boldsymbol{K}_k\boldsymbol{C}\boldsymbol{\Sigma}_{k-1}\right)\boldsymbol{A}^T + \boldsymbol{Q}
\end{aligned} \tag{3}$$

where $\boldsymbol{K}_k$ is the Kalman gain; $\boldsymbol{A}$ and $\boldsymbol{C}$ are the system and output matrices, respectively, as defined in (2); Σ_k is the state covariance matrix at time step k; $\boldsymbol{y}_k$ is the observed pose estimate as obtained from the neural network model at time step k; and $\hat{\boldsymbol{x}}_k$ is the system state estimate at time step k.

A constrained-state Kalman filter overcomes this limitation and projects the unconstrained estimates of the Kalman filter on the constraint surface [32], [33]. It solves for the system state estimates by minimizing the state variance while subjecting them to the state constraints (4).

$$\begin{aligned}
\tilde{\boldsymbol{x}}_k &= \arg\min_{\boldsymbol{x}} \left(\boldsymbol{x} - \hat{\boldsymbol{x}}_k\right)^T \boldsymbol{\Sigma}_k^{-1}\left(\boldsymbol{x} - \hat{\boldsymbol{x}}_k\right)\\
\text{subject to: } & \boldsymbol{D_1 x} \le \boldsymbol{d_1}\\
& \boldsymbol{D_2 x} \ge \boldsymbol{d_2}
\end{aligned} \tag{4}$$

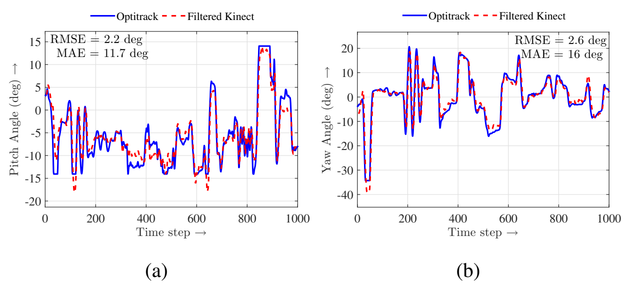


Fig. 5. A comparison of the pose estimates obtained from the OptiTrack motion capture system with the filtered Kinect-based estimates for (a) pitch angle comparison and (b) yaw angle.

The closed form solution (5) to the optimization problem can be obtained using the Augmented Lagrangian method [34].

$$\tilde{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k - \boldsymbol{\Sigma}_k \hat{\boldsymbol{D}}^T \left(\hat{\boldsymbol{D}} \boldsymbol{\Sigma}_k \hat{\boldsymbol{D}}^T \right)^{-1} \left(\hat{\boldsymbol{D}} \hat{\boldsymbol{x}}_k - \hat{\boldsymbol{d}} \right) \qquad (5)$$

where $\hat{\boldsymbol{D}} = [\boldsymbol{D_1} \ \ -\boldsymbol{D_2}]^T$ and $\hat{\boldsymbol{d}} = [\boldsymbol{d_1} \ \ -\boldsymbol{d_2}]^T$. The matrices D_1 and D_2 are configured such that the angular position, velocity and acceleration of the two robot joints are bounded by the vectors d_1 and d_2. The filter ensures that the estimated states are always within their respective upper and lower bounds and thus, guarantee that the robot operates within its physical limits while retaining the original human head motion as much as possible.

IV. Experiments and Results

To evaluate the performance of the TeleFurhat system, we present the results from the neural network training, the performance of the trained network on the robot using the collected motion capture data, and the results from real-time execution of the TeleFurhat system. Both the remote and the local computer were present on the same local area network for the experiments carried out in this work. Additionally, we perform a perceptual study to obtain viewer impressions of our system.

A. Neural Network Training Performance

1) Neural Network RMS Error Mapping: We first determine the optimum number of perceptrons in the hidden layer and the size of the past frame window. As shown in Fig. 6(a), the RMS error (between the network output and reference) for the pitch angle decreases as both the hidden layer size and the past window size increase over the training dataset. However, the error increases beyond hidden layer size of 10 and past window size of 20 over the test dataset (Fig. 6(b)). In contrast, yaw angle error decreases with the increase in hidden layer size and past window size over both the training and test datasets (Fig. 6(c) and (d)). This result shows that the learned networks are relatively less generalizable for the pitch motion compared to the yaw motion. This may be because the human head motion is much more dynamic for the pitch angle motion as compared to the yaw angle motion, which increases its variability. To achieve an optimum compromise for the performance of both the pitch and yaw angles, we choose a network with a hidden layer size of 20 and a past window size of 30, which ensures that the error is under 1.5 degree for the pitch angle and under 1.3 degree for the yaw angle in both the training and test datasets. The model-fitting statistics in Table III show that the network has reduced both the mean squared error (MSE) and the maximum absolute error (MAE) as compared to the error evaluated between the Kinect data and the OptiTrack data. Furthermore, the standard deviation of the error distribution with the neural network model is smaller than without the model for both the pitch and the yaw angles, demonstrating that there are more samples with lower error when the model is used.

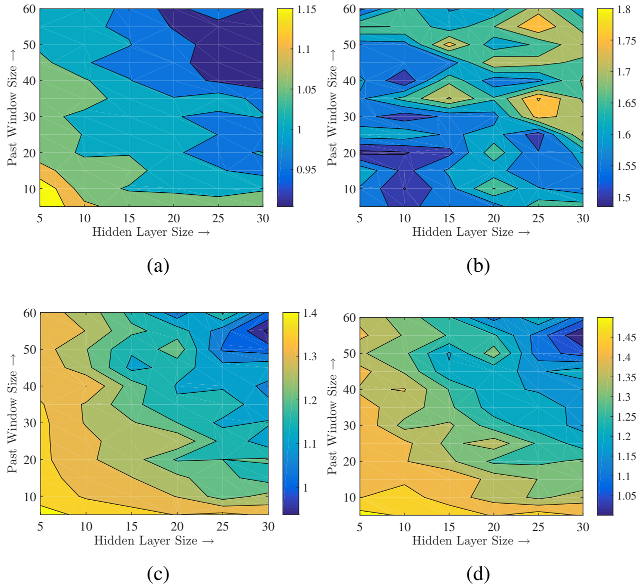


Fig. 6. Root mean square (RMS) error plots for different numbers of neurons in the hidden layer and past window sizes for the learned neural network-based output. The error is evaluated by comparing the neural network output with the corresponding motion capture data. (a) The pitch angle prediction error for the training data, (b) the pitch angle prediction error for the test data, (c) the yaw angle prediction error for the training data and (d) the yaw angle prediction error for the test data.

TABLE III

Model-fitting statistics for the selected network with a hidden layer size of 20 and a past window size of 30. MSE, MAE and SD represent mean squared error, maximum absolute error and standard deviation, respectively.

Dataset	Pitch			Yaw		
	MSE	MAE	SD	MSE	MAE	SD
Raw Overall*	1.616	8.827	2.487	2.564	15.366	2.432
Training	1.003	7.288	1.242†	1.200	6.147	1.445†
Test	1.581	6.854		1.272	10.414	

∗ statistics without any neural network model
† statistics over all data

2) Learned Neural Network Output: A comparison of the output from the learned neural network model with the reference data shows that the network output is similar to the reference data (Fig. 7). In general, the performance is better for the training dataset than the test dataset.

B. Offline System Performance Test

To test the performance of the neural network model on the robot head, we control the robot head using the Kinect-based pose measurements from one of the captured sequences as input to our framework. The robot head is also tracked simultaneously using the OptiTrack motion capture system to verify its performance.

Fig. 8 shows the results from the offline performance test on the robot with the human head angles from the OptiTrack

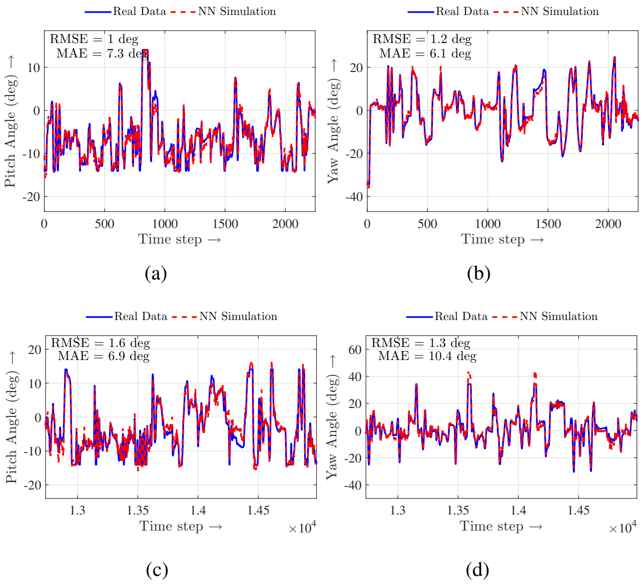


Fig. 7. A comparison of the measured data with the output obtained from the learned neural network. (a) The pitch angle comparison for the training data, (b) the yaw angle comparison for the training data, (c) the pitch angle comparison for the test data, and (d) the yaw angle comparison for the test data.

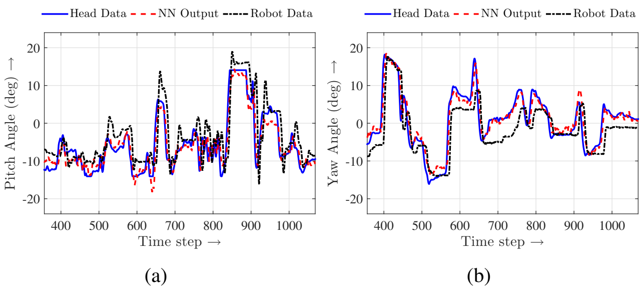


Fig. 8. Comparison of the measured actor head pose (head data) with the learned neural network output (NN output) and the measured robot head pose obtained by rendering the filtered network output on the robot (robot data) with a network having 20 neurons in the hidden layer and a past window size of 30: (a) pitch angle, (b) yaw angle. The post-filter used for these results is the original Kalman filter without any bounds on the velocity and acceleration of the system.

system, the output from the learned neural network, and the actual robot head angles. Snapshots from this experiment are shown in Fig. 10. Some deviation is occasionally observed during fast motion due to the dynamics of the robot joint that has not been modeled by the neural network. The reconstruction of the pose estimates on the robot head shows that the human head pose matches the robot head pose most of the time (refer to the supplemental video for details). Some deviation in pose observed in Fig. 10 images 23 and 24 is due to the fact that there is no roll DOF on the robot. Also, the pose deviation observed in Fig. 10 image 10 is due to the limit on the yaw angle of the robot.

C. Real-time System Performance Test

To test the real-time performance of the system, a user controlled the robot head in real time using our framework.

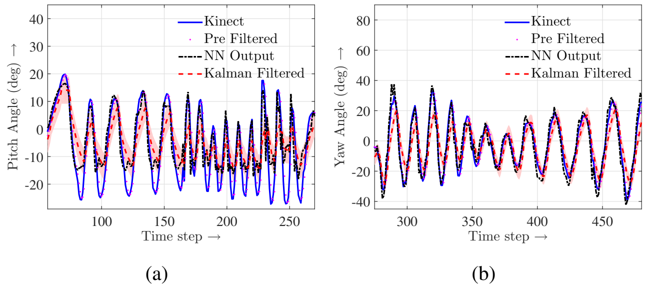


Fig. 9. Output pose at various stages of the framework as obtained during real-time execution of the TeleFurhat system. The shaded area in the plot shows the one sigma bound of the estimates obtained using the constrained-state Kalman filter.

The data is logged after every block in the framework to verify its contribution to the overall performance of the system.

Our results show that the system is able to obtain smooth, safe pose trajectories for the robot using the Kinect data as input (Fig. 9). The moving average filter effectively removes the high-frequency noise in the data. The learned neural network then corrects for any error in the filtered pose and limits the pose to be within the bounds of the robot. Finally, the constrained Kalman Filter limits the position, velocity and acceleration within their allowable limits while preserving the nature of the motion.

During the experiments, we observed that the timing of the motion is more important than the accuracy of the amplitude of the motion for the gestures to look natural. When the head motion has a high frequency, the pose magnitude is limited by the Kalman filter to ensure that the system does not exceed the allowable velocity and acceleration bounds while maintaining the timing of the motion. This leads to effective and safe rendering of the motion on the robot in real time (Fig. 11 and supplemental video). A delay of 100 to 170 ms is observed between the human head pose and the robot head pose. A major portion of the delay is introduced due to the communication over the network ($\sim$34 to 104 ms). This delay due to communication is also present in the baseline method.

D. Perceptual Studies

To compare our new framework for controlling head motion to the use of raw pose measurements from the Kinect sensor, we performed two online comparison studies using Amazon Mechanical Turk. This study was approved by our Institutional Review Board, and participants provided informed consent and were reimbursed for their time.

1) Participants: Using Mechanical Turk and SurveyGizmo.com, 203 participants successfully performed the first study and 205 completed the second study. All participants had to be 18 years of age or older and have self-reported normal or corrected-to-normal hearing and vision. Additionally, participants had to have an approval rating above 95% and more than 50 previous assignments on Mechanical Turk in order to enroll in the research.

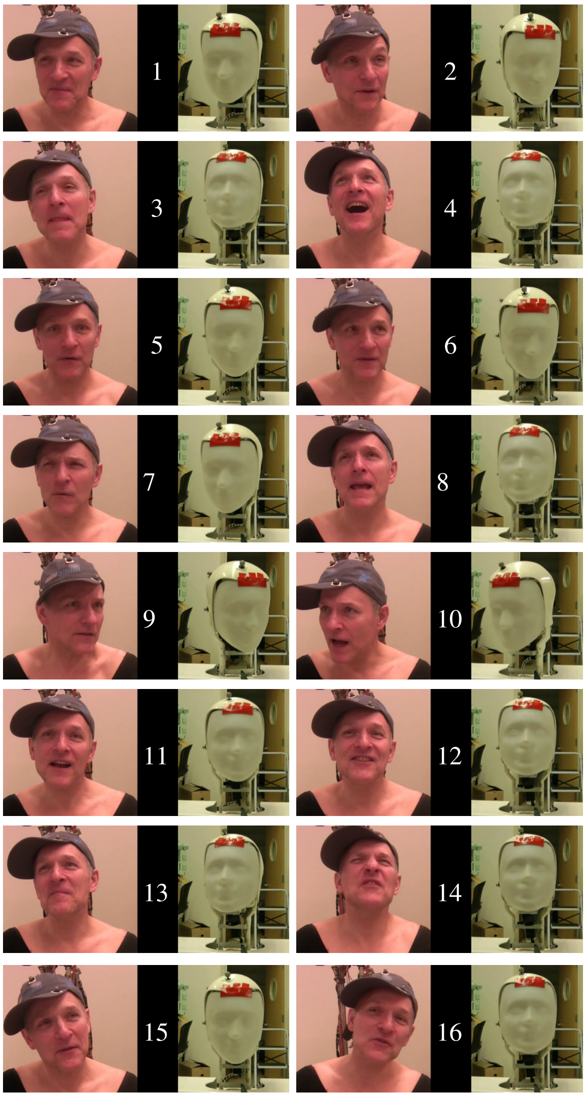


Fig. 10. Results from the offline rendering of the pose using the learned pose mapping. The motion capture system is used to measure the pose of the actor and the robot to compare the two.

2) *Stimulus Generation:* Using 3.78 minutes of the actor's tracked motion data from the recorded storytelling task, we created two types of head motion for TeleFurhat. In the first type, the motion was created using the raw pose measurements of the Kinect sensor as a baseline. For the second type, we used our method for modifying the position commands sent to the robot controller.

3) *Study 1 Method:* We used SurveyGizmo, an online service, to display the stimuli for the research. The input video of the actor was presented at the center, and the two output videos were presented on each side of the video of the actor. All three videos were time-aligned to present simultaneous movements. Different versions were created to counterbalance the display by switching the left/right positions of the output videos; half of the participants saw each order. Each participant watched the videos for at least 3.5 minutes before submitting an answer to the question,

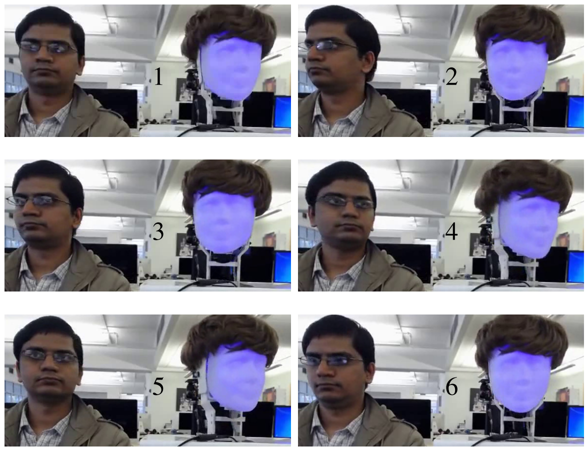


Fig. 11. Results from the real-time estimation of the human head pose and rendering of the pose on the Furhat robot head using the proposed framework. A delay of approximately 100 to 170 ms is observed between the human head pose and the robot head pose. The slight differences in pose is due to the observed delay in the system.

"Which robot head motion is closer to the human head motion?"

4) *Study 1 Results:* The two methods did not significantly differ for perceived accuracy of head motion (113 our method vs. 90 baseline method; $\chi^2 = 2.606$, $p = .1065$, where χ^2 and p are the test statistics from the Pearson's chi-squared test)

5) *Study 2 Method:* Before completing the questionnaire, participants were shown a 18-second clip of the actor alone in order to provide an idea of what types of head movements could be expected. Then, they proceeded to the main page, which contained a 3.78-minute video only showing the two videos of the robot heads side by side. Again, participants had to play the video for at least 3.5 minutes before submitting their responses. At the bottom of the page were four questions: "Which robot looks more alive?", "Which robot looks more humanlike?", "Which robot looks more intelligent?", and "Which robot looks more friendly?"

6) *Study 2 Results:* Our method was rated as significantly more alive (135 vs. 70; $\chi^2 = 20.61$, $p < 0.0001$) and friendly (122 vs. 83, $\chi^2 = 7.42$, $p = 0.0065$) than the baseline method. There were no significant differences for humanlike (112 vs. 93; $\chi^2 = 1.76$, $p = 0.18$) or intelligent (98 vs. 107, $\chi^2 = 0.40$, $p = 0.52$).

7) *Discussion:* Our method was preferred to the previous method for questions of how alive and friendly the robot appeared. The baseline method was never preferred. The two methods did not differ for closeness to human head motion, humanlikeness, and perceived intelligence. Overall, these findings suggest that our method improves upon the use of raw pose data from the standpoint of a viewer.

V. Conclusion

We developed a system for teleoperation using the Furhat robot, which we call TeleFurhat. Our experiments showed that the robot head is able to effectively imitate the human head motion in real time while ensuring that the robot operates within its physical limits. In the future, we also plan to use the uncertainty estimates from the Kalman filter to evaluate the pose input to the robot. We observed that some very high-frequency motion (e.g., head nodding) is not accurately captured even by the marker-based motion capture system in real time. In subsequent research, we plan to ask an animator to create an animation on the robot that matches a measured human head motion, and learn models between the Kinect-based pose estimates and the animated motion. This process will help account for creative decisions often made by artists to overcome the hardware limitations such as lack of a roll joint.

Acknowledgments

The authors thank Ken Bolden and Josh Gyory for participating in the various human motion data capture tasks and Justin Macey in the Motion Capture Lab at Carnegie Mellon University for assisting in capturing the human motion data.

References

[1] S. Al Moubayed, J. Beskow, G. Skantze, and B. Granström, "Furhat: a back-projected human-like robot head for multiparty human-machine interaction," in *Cognitive Behavioural Systems*. Springer, 2012, pp. 114–130.

[2] D. Todorović, "Geometrical basis of perception of gaze direction," *Vision research*, vol. 46, no. 21, pp. 3549–3562, 2006.

[3] S. Al Moubayed and G. Skantze, "Turn-taking control using gaze in multiparty human-computer dialogue: effects of 2d and 3d displays." in *AVSP*, 2011, pp. 99–102.

[4] J. Y. Chen, E. C. Haas, and M. J. Barnes, "Human performance issues and user interface design for teleoperated robots," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, no. 6, pp. 1231–1245, 2007.

[5] S. Nishio, H. Ishiguro, and N. Hagita, *Humanoid Robots, New Developments*. INTECH Open Access Publisher Vienna, 2007.

[6] J. Demiris, S. Rougeaux, G. Hayes, L. Berthouze, and Y. Kuniyoshi, "Deferred imitation of human head movements by an active stereo vision head," in *IEEE International Workshop on Robot and Human Communication*, 1997, pp. 88–93.

[7] J. J. Heuring and D. W. Murray, "Modeling and copying human head movements," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 6, pp. 1095–1108, 1999.

[8] M. Riley, A. Ude, K. Wade, and C. G. Atkeson, "Enabling real-time full-body imitation: a natural way of transferring human movement to humanoids," in *IEEE International Conference on Robotics and Automation*, vol. 2, 2003, pp. 2368–2374.

[9] A. Shon, K. Grochow, A. Hertzmann, and R. P. Rao, "Learning shared latent structure for image synthesis and robotic imitation," in *Advances in Neural Information Processing Systems*, 2005, pp. 1233–1240.

[10] A. P. Shon, J. J. Storz, and R. P. Rao, "Towards a real-time bayesian imitation system for a humanoid robot," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 2847–2852.

[11] J. Koenemann, F. Burget, and M. Bennewitz, "Real-time imitation of human whole-body motions by humanoids," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 2806–2812.

[12] R. C. Luo, B.-H. Shih, and T.-W. Lin, "Real time human motion imitation of anthropomorphic dual arm robot based on cartesian impedance control," in *IEEE International Symposium on Robotic and Sensors Environments*, 2013, pp. 25–30.

[13] M. Gleicher, "Retargetting motion to new characters," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM, 1998, pp. 33–42.

[14] K. Yamane, Y. Ariki, and J. Hodgins, "Animating non-humanoid characters with human motion data," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2010, pp. 169–178.

[15] J. Kofman, X. Wu, T. J. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *IEEE Transactions on Industrial Electronics*, vol. 52, no. 5, pp. 1206–1219, 2005.

[16] A. D. Dragan and S. S. Srinivasa, "Online customization of teleoperation interfaces," in *RO-MAN*, 2012, pp. 919–924.

[17] "Microsoft kinect 2 sensor api," https://msdn.microsoft.com/en-us/library/dn785525.aspx, accessed: 2015-03-15.

[18] V. F. Ferrario, C. Sforza, G. Serrao, G. Grassi, and E. Mossi, "Active range of motion of the head and cervical spine: a three-dimensional investigation in healthy young adults," *Journal of orthopaedic research*, vol. 20, no. 1, pp. 122–129, 2002.

[19] "Optitrack motion capture system," http://www.optitrack.com, accessed: 2016-02-29.

[20] D. A. Winter, *Biomechanics and motor control of human movement*. John Wiley & Sons, 2009.

[21] D. F. Specht, "A general regression neural network," *IEEE Transactions on Neural Networks*, vol. 2, no. 6, pp. 568–576, 1991.

[22] C. E. Rasmussen, "Gaussian processes for machine learning," 2006.

[23] F. Tancret, H. Bhadeshia, and D. MacKay, "Comparison of artificial neural networks with gaussian processes to model the yield strength of nickel-base superalloys," *ISIJ international*, vol. 39, no. 10, pp. 1020–1026, 1999.

[24] C. Fyfe, T. D. Wang, and S. J. Chuang, "Comparing gaussian processes and artificial neural networks for forecasting," in *9th Joint International Conference on Information Sciences*. Atlantis Press, 2006.

[25] J. Willmore, R. Price, and W. Roberts, "Comparing gaussian mixture and neural network modelling approaches to automatic language identification of speech." in *Aust. Int. Conf. Speech Sci. & Tech.*, 2000, pp. 74–77.

[26] F. A. Anifowose, "A comparative study of gaussian mixture model and radial basis function for voice recognition," *International Journal of Advanced Computer Science and Applications*, vol. 1, no. 3, pp. 1–9, 2012.

[27] D. Wettschereck and T. Dietterich, "Improving the performance of radial basis function networks by learning center locations," in *NIPS*, vol. 4, 1991, pp. 1133–1140.

[28] B. Karlik and A. V. Olgac, "Performance analysis of various activation functions in generalized mlp architectures of neural networks," *Int. J. Artificial Intell. Expert Syst*, vol. 1, no. 4, pp. 111–122, 2011.

[29] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural computation*, vol. 1, no. 2, pp. 281–294, 1989.

[30] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *The Journal of Neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[31] R. Shadmehr, *The computational neurobiology of reaching and pointing: a foundation for motor learning*. MIT press, 2005.

[32] D. Simon and T. L. Chia, "Kalman filtering with state equality constraints," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 1, pp. 128–136, 2002.

[33] B. O. Teixeira, J. Chandrasekar, L. A. Tôrres, L. A. Aguirre, and D. S. Bernstein, "State estimation for linear and non-linear equality-constrained systems," *International Journal of Control*, vol. 82, no. 5, pp. 918–936, 2009.

[34] D. Simon, "Kalman filtering with state constraints: a survey of linear and nonlinear algorithms," *IET Control Theory & Applications*, vol. 4, no. 8, pp. 1303–1318, 2010.