Minimum Jerk Based Prediction of User Actions for a Ball Catching Task

Mattias Bratt, Christian Smith, and Henrik I Christensen

Centre for Autonomous Systems

Royal Institute of Technology

Stockholm, Sweden

Email: {bratt,ccs,hic}@kth.se

Abstract—The present paper examines minimum jerk models for human kinematics as a tool to predict user input in teleoperation with significant dynamics. Predictions of user input can be a powerful tool to bridge time-delays and to trigger autonomous sub-sequences. In this paper an example implementation is presented, along with the results of a pilot experiment in which a virtual reality simulation of a teleoperated ball-catching scenario is used to test the predictive power of the model. The results show that delays up to 100 ms can potentially be bridged with this approach.

Index Terms—prediction, minimum jerk, teleoperation

I. INTRODUCTION

Teleoperation in robotics has been studied for more than half a century. Applications vary from direct remote control as seen for example in hot cells for handling of nuclear material, to task specification of space vehicles on remote planets. Traditionally, delays in command and control systems have been handled by predictive methods [1] or through increased autonomy - i.e. higher level control. The degree of autonomy increases with time delays. As part of the design of autonomy a challenge is the selection of control primitives. The ultimate challenge in design of tele-operation control is for systems with significant dynamics. The type of systems considered here, is one in which there is a need for closed loop control of the system in reaction to sensory feedback. For such a system control becomes a major challenge where the time delay (T_d) is a significant portion of the overall process time (T_p). To study these kinds of systems we have chosen to study tele-operated catching. In a regular laboratory or living room a thrown ball has a flying time of 600-950 ms, and if we were to consider teleoperated catching over the internet from one continent to another the time-delay could easily be 200 ms which is on the order of 20-35% of the overall flying time. How does one select a control strategy for such systems?

If a person were to catch the ball directly then s/he would perform a ballistic arm movement [2] [3] followed by some correction. The strategy could be thought of as “throw the arm to be close to the correct position and use the added time to obtain an improved estimate, which is then executed as a small correction”. Given such a strategy how can early motion by the user be estimated with sufficient accuracy to allow it be to used for early motion by the end-effector.

If we consider a process as outlined above then it can be divided into 5 phases:

- 1) The time delay (T_d) during which no action/data are available
- 2) The reaction time (T_r) which is the time it takes for the user to initiate a response to data/perceptual data
- 3) The estimation period (T_e) which is the time needed for estimate what the user is trying to achieve and use this as a basis for control generation
- 4) The fly phase (T_f) - a largely feed-forward action is generated while a refined estimate is computed.
- 5) The correction phase (T_c) - during which the improved estimate is used to generate corrective action commands to the robot.

Obviously the entire time used must be smaller than the entire process time (T_p). In the present paper we study the estimation of commands issued by the user under significant time pressure. The questions addressed in the present paper are:

- What are good estimation models for identification of the trajectory specified by the user?
- How does one identify any corrective commands?
- How can these commands be integrated into the control system?
- What kind of performance can be achieved with a real system?

The paper is organized as follows: Section II describes the teleoperation task, Section III presents the minimum jerk models, Sections IV and V describe the implementation and experimental setup, and the results are given in Section VI.

II. TASK DESCRIPTION

In order to study teleoperation of a process where the time-delay is a large fraction of the process cycle time, ballcatching has been chosen as an example task. The task for the operator is to guide the robot's end effector so that it intercepts the ballistic trajectory of the ball. The operator has to perform this without any direct observations of the ball or the manipulator, with all information being relayed through a user interface.

Given the specifications of the available experimental setup, the task can be specified more exactly as catching a ball with a diameter of 7 cm that has been thrown a distance of 4 meters for a flight time of approximately 900 ms. The ball is caught in a cardboard cylinder with a diameter of 14 cm, which means that the spatial precision of the catch must

be within 7 cm for the center of gravity of the ball to fall within the cylinder. A detailed description of the manipulator and the catching task can be found in [4] and of the control setup and internet time delay handling in [5]. Details of the user interface are described in Section V.

As an illustration of the difficulties that arise with delays, the setup has been teleoperated with the user being located in Pontedera, Italy, and the manipulator being in Stockholm, Sweden. The average round-trip delay when using a public internet connection was measured to 50 ms. In that test, one single ball was caught, possibly the first successful trans-continental teleoperated ball-catching experiment of its kind. However, an examination of the subsequent misses showed that in most cases a correct intercept position was reached, but too late to be in time to catch the ball. Thus it is of interest to find a technique to bridge the time-delay problem.

The current paper aims to find a means to augment the user input in order to successfully perform the ballcatching task even in the presence of these significant delays.

III. MODELING

A tried approach to bridging time delays is by including a simulation of the system in the user interface, as described in e.g. [6] or [5]. Simulating the dynamics of the robotic manipulator or the ballistics of the ball is dependent on the complexity and quality of available models.

In the present setup, models of the manipulator and ball path are well-known and simple enough to simulate within small error margins. This only solves one part of the time-delay problem. The operator will have access to a good simulation of real-time information, but there will still be delays as the user reacts to this information, and as the system relays user input to the remote manipulator. In the present paper, first steps towards including a simulation of the operator's input in the system are taken. This requires a model of the user's input that relates previous and present inputs to future ones.

A well-known model for explaining the kinematics of visually guided human reaching motions is the *minimum jerk* (MJ) model. It was first proposed by Hogan for single-joint motions in [7], and later extended to include multi-joint planar motion in [8]. Hogan and Flash observed that the trajectory of voluntary arm motions, when described in extra-corporeal cartesian space, follow certain constraints. It seems that the trajectories can be predicted by using a model in which the square sum of the third derivative of position, *jerk*, integrated over time was minimized. I.e, given a starting point, an end point and a time to move between the two, the trajectory that minimizes the jerk on this interval is the MJ trajectory.

It was shown that all MJ trajectories share the property that the 6th derivative is zero for the duration of the motion, and that they thus can be described as 5th degree polynomials, as in Equation 1.

$$\mathbf{x}(t) = a_1 t^5 + a_2 t^4 + a_3 t^3 + a_4 t^2 + a_5 t + a_6 \quad (1)$$

If we also add the start and end points of the motion, $\mathbf{x}(t_0)$ and $\mathbf{x}(t_1)$, and state the position, velocity, and acceleration at these points, we get the following constraints on Equation 1.

$$\begin{aligned} \mathbf{x}(t_0) &= \mathbf{x}_0, & \mathbf{x}(t_1) &= \mathbf{x}_1 \\ \dot{\mathbf{x}}(t_0) &= \dot{\mathbf{x}}_0, & \dot{\mathbf{x}}(t_1) &= \dot{\mathbf{x}}_1 \\ \ddot{\mathbf{x}}(t_0) &= \ddot{\mathbf{x}}_0, & \ddot{\mathbf{x}}(t_1) &= \ddot{\mathbf{x}}_1 \end{aligned}$$

The above constraints will give us 6 equations, and we get a well-defined system to find the 6 parameters $a_1 \dots a_6$. Thus, there is only one possible MJ trajectory for a given start and end, and it can be found by solving a simple system of linear equations.

The trajectories described by the MJ model are limited to one single motion. What happens if a more complex motion is desired or if the target of the motion is changed in mid-motion can be described by superpositioning several MJ trajectories. If the added MJ trajectory has an initial position, velocity, and acceleration of zero, this will still result in a continuous motion where the 6th derivative is zero, so the jerk is still minimized. This is described thoroughly in [9] and [10]. The interesting point is that with this description, even when the target position is changed, the original MJ motion is not removed from the trajectory, but kept alongside the new motion. According to [11], a new submovement can be generated as often as once every 100 ms.

There are other models — such as the minimum torque change model — that can accurately describe a wider class of motions, but for motions that are not perturbed by outer forces or include significant changes of posture, MJ will fit observations equally good, according to [2]. The MJ model is chosen since it is significantly simpler.

Observations in [3] and [12] on the motions that humans make when freely catching a thrown ball indicate that they start by moving towards the expected point of impact with a distinct MJ-type reaching motion, and later add smaller corrective MJ-type motions to accurately catch the ball.

If the assumption is made that the first rough reaching motion is accurate enough so that the corrective motions have no significant impact on the overall trajectory, it is possible to approximate the entire motion with one MJ trajectory. An example of such a motion recorded from experiment with the MJ superimposed is shown in Fig. 1.

If, on the other hand, the first rough reaching motion is not accurate enough and the subsequent corrective motions are large when compared to the first, an approximation of the entire motion with one MJ trajectory will not result in a good fit, but rather several MJ trajectories are needed, see Fig. 2.

IV. IMPLEMENTATION

In the present paper, the MJ trajectory model described in Section III will be used to predict future user inputs. If the endpoint of a user-commanded motion can be known beforehand, it should be possible to bridge the time delays present in the system.

Fig. 1. Y component of measured hand trajectory with MJ trajectory fitted. In this case the hand trajectory contains only one major MJ component.

Fig. 2. Y component of measured hand trajectory with MJ trajectory fitted. In this case the hand trajectory contains two major MJ components. The two MJ components are shown superimposed to an approximation of original data.

A. Trajectory Estimation

To predict future motions, it is necessary to know when the motion starts and stops. Given that the goal is predicting the outcome of a motion, it is plausible to register the starting point of the motion after it starts, but the point in time at which the motion ends has to be found before it can be observed. Using Equation 1, we see that if the start of a motion is known, but not the end, there are two unknowns — the time and position at which the motion ends. Thus, in theory, if two points on the trajectory are known, the system of equations can be solved to find these two unknowns and thereby specify the entire motion exactly. Since this involves solving a 5th degree polynomial, the solution will be very sensitive to error, especially in the time domain. However, if the start and end times are known, extrapolations that fit well with observed data are possible.

Fig. 3. 4th degree MJ velocity profile fitted to Kalman filtered velocity data. The solid line is the data used for fitting, the dotted line is the (unused) remainder of the measured velocity, and the dashed line represents the fit MJ velocity profile.

A robust way to detect the start time t_0 and the end time t_1 is proposed in the following. Since the motion itself is expected to follow a 5th degree polynomial curve, the velocity profile is expected to follow a 4th degree polynomial curve that starts and ends at zero value, with zero first derivative. Such a curve is symmetric around the apex. Empiric analysis of recorded motion data shows that by using an extended Kalman filter, the point of maximum velocity $t_{v_{max}}$ is easy to detect. Using a least square approach, the 4th degree velocity profile can be fit around this peak, fitting only to data before the peak, see Fig 3. The zeros of the polynomial found with this approach are used as candidates for the start and end of the motion, and used when fitting the motion data to a MJ trajectory.

The weakness in this approach is that half of the motion has to be observed before a prediction can be made. In order to facilitate earlier predictions, the EKF observer can be used to predict when $t_{v_{max}}$ will be reached. This involves estimating higher derivatives, and is prone to high uncertainties in the presence of observation noise. The implementation used in the experiments described in Section V therefore uses a $t_{v_{max}}$ predicted from the EKF until $t_{v_{max}}$ is reached, when the algorithm switches to using the observed value.

When t_0 and t_1 are known, along with the start position x_0 of the motion, Equation 1 can be rewritten as in 2.

$$\begin{aligned} \mathbf{x}(t) = & \mathbf{x}(t_0) + \\ & a_1 \left(t^5 - \frac{5}{2}(t_1 + t_0)t^4 + \frac{5}{3}(t_1^2 + 4t_1 t_0 + t_0^2)t^3 - \right. \\ & \left. 5(t_1 t_0^2 - t_1^2 t_0)t^2 + 5t_0^2 t_1^2 t - \right. \\ & \left. \frac{1}{6}t_0^5 + \frac{5}{6}t_0^4 t_1 - \frac{5}{3}t_0^3 t_1^2 \right) \end{aligned} \quad (2)$$

This equation only contains one unknown, a_1 . By fitting to the latest measured data points with a least squares fitter, a fairly robust trajectory prediction is possible.

B. User Intent Prediction

Given that the point in space that the user will command the robot to move to is known beforehand, it should also be possible to some extent to predict what the user is trying to accomplish. This requires that some model of possible tasks that the user can perform is available.

In the present setup, this problem is simplified to distinguishing between two possible tasks:

- 1 Catching a ball.
- 2 Moving the robot without catching a ball.

The second task can be any possible task that is not task 1. If it is probable that the user is trying to perform task 1, the system can aid the user by switching to an autonomous ball-catching mode with a high probability of success. If it is more probable that the user is trying to perform task 2, moving the robot without catching any ball, the system can aid the user by moving the robot to the probable motion goal without delays.

In the present implementation, it is examined if the user intent can be extracted from the predicted hand motion. The predicted hand motion trajectory is compared to the predicted trajectory of a thrown ball. If the distance between the predicted trajectories is below a certain threshold, the user input is interpreted as an attempt at ball catching. The distance can be scaled differently with respect to the main axis of motion, and the threshold and scaling can differ with the expected time to impact.

In order to test the system's ability to discriminate between a catch attempt and another motion, we use the catching of another ball with a similar, but not identical, trajectory as the 'non-catching' task 2. If this difference can be successfully detected, it should also be possible to discriminate between catching tasks and other, more dissimilar tasks.

V. EXPERIMENTAL SETUP

To test the utility of using minimum jerk trajectories to predict human motion as described above, an experiment was performed. The test equipment was the same as that used for teleoperating the catching robot, but no command signals were sent to the robot. Ball position data was not collected in real-time, but instead prerecorded data from real throws was retrieved from file by the stereo vision system, so that the same set of throws could be repeated.

The subject was presented with the virtual reality interface also used for teleoperation. This shows the current position of the robot, and during a throw, the position and predicted trajectory of the ball. The subject's task was to catch the ball by directing the virtual robot to a point on the ball trajectory before the ball had passed.

To make movements as natural as possible, we chose not to feed back any information about robot inertia or simulated actual robot position to the user, even though a dynamic simulation of the robot was done in the user interface computer. The user therefore saw and felt a massless manipulator, that instantly followed given commands. The assumption of motion consisting of superimposed minimum jerk trajectories

Fig. 4. User interface hardware configuration, with a CRT stereo display, an Omega haptic device, and a loudspeaker.

would probably have been invalidated by dynamic feedback forces unpredictable to the user had such forces been used. When teleoperating a real robot however, we plan to display the actual position of the robot graphically, together with the current commanded position.

A. User Interface

The hardware user interface (shown in Figure 4) used for the experiment consisted of

- a CRT monitor stereo display. Shutter glasses (from an inexpensive eDimensional 3D Vision System kit) connected to a professional graphics board were used for displaying stereo on a standard 21-inch CRT monitor.
- a 3D force reflectance joystick. An Omega unit from Force Dimension served as input channel for user hand motion. This haptic device uses parallel linkage, which provides a large workspace, as well as high stiffness and force output.
- a loudspeaker for audio feedback.

In the user interface visualization is performed using Open Inventor 6 from Mercury Computer Systems, supporting OpenGL quad buffered stereo with asymmetric frustum perspective mapping. Force feedback is implemented using the low level API provided by Force Dimension, but in the present experiment it was used only for setting up virtual boundaries around the workspace of the robot, and for compensating the gravity forces on the device itself. A screen dump from the user interface is in Figure 5.

The user interface receives recorded estimated positions of the ball from the stereo vision system of the physical robot setup. It also gets the complete state of the vision system Kalman filter to be able to estimate future ball positions. When controlling the real robot this can bridge communication delays, but the user interface also uses the predictive power over the much longer time interval of a complete throw to show the user the projected trajectory of the ball.

Fig. 5. The user's view in the ball-catching experiment.

The predicted ball trajectory is drawn as a red line through the ball, and the uncertainty inherent in the future Kalman states is displayed as a semi-transparent shell at a distance of one standard deviation from the predicted trajectory. The result is that the uncertainty of the ball trajectory drawn is visible as a funnel that has its narrow end at the current ball position, and widens as the uncertainty increases the further into the future the prediction extends. A predicted ball trajectory and its uncertainty representation are both visible in Figure 5.

Just before each throw in the experiment, the ball was shown with an alternate color at the start position of the throw. When the throw started, the word 'go' was output to the speaker to alert the user. Another sound was played when the ball was caught. Catches were detected by measuring if the center of the ball passed within 7 cm of the center of the bottom of the cardboard cylinder. No other collision detection was performed, since it is not relevant to the task.

All data about user actions, as well as received ball state, is logged to file at the user interface computer. The log files can then be exported and used for data analysis with tools such as Matlab, but also serves as input to the user interface software in replay mode.

B. Procedure

The experiment was performed with one well practiced subject, who after several hundred practice throws had a catch rate of almost 100%.

To minimize changes in the configuration of the arm controlling the input device, no external support at e.g. the elbow was used. Using a support could result in the subject controlling the virtual robot with mainly wrist motions, yielding large changes in arm configuration over the workspace. This in turn would correspond to a large change in posture, and would decrease the accuracy of the minimum jerk assumption (see Section III and [2]).

The input workspace volume is limited by the haptic device to approx $16 \times 16 \times 12$ cm. The balls thrown all pass

Fig. 6. Proportion of predicted hand end positions outside 6 cm radius from actual hand catch position. Results are shown for selected hand trajectories, all hand trajectories, and for the case of using current hand position instead of a predicted endpoint (there is no visible difference when the latter curve is plotted for all hand trajectories or just the selected ones). For a fair comparison the curve representing the complete set of hand trajectories has been modified as to not include nine trajectories that are completely contained within the 6 cm radius.

through a 60×60 cm window in front of the robot, which maps to a 12×12 cm window in input space.

The data from the experiment consists of 105 throws in five series of 21 throws each. Twice during the experiment the subject failed to catch the ball. Each of these throws were immediately discarded and replaced by an extra throw, since missed throws do not offer a reference point where the subject was aiming. Furthermore, some throws that did not generate interesting data were excluded from parts of the analysis. For example, for some throws by chance the operator was in the correct final position before the throw began. For these throws, a successful catch was performed without moving the robot at all, and the prediction of the motion endpoint could trivially be found within a very high absolute accuracy. Also, for some throws, the operator made a very poor initial motion and the subsequent corrective motions were of equal or larger magnitude than the first, as exemplified in Fig 2. It is not expected that these motions can be accurately predicted with the approach in the current implementation. There were approximately 12 throws where the initial position was too close, and 24 throws where the operator's initial guess was significantly off. This left 69 throws to be considered in the main part of the analysis. The performance on all recorded motions is also presented, for reference.

VI. RESULTS AND DISCUSSION

The data collected in the experiment was analyzed offline, and the endpoint of the hand motion was continually estimated as time approached the catching instant. This prediction was then compared to the hand catch position to test the quality of the prediction. The results of this analysis are shown by the plots in Figures 6 and 7.

Fig. 7. Median deviation of the predicted endpoints for selected hand trajectories catches from actual hand catch position.

The plots demonstrate that for most hand trajectories in the selected set, good estimates of the catch position become available during an interval from approximately 250 to 200 ms before impact. At around 200 ms the time advantage before the same level of convergence is reached without prediction is approximately 70 ms.

The threshold distance of 6 cm from the catch position that was used to generate Figure 6 was chosen by examining several similar plots. Increasing the threshold reasonably beyond 6 cm gives little improvement in terms of earlier convergence, whereas decreasing it causes a substantially larger delay. A sphere with radius 6 cm has a volume equal to 2.3% of the smallest possible convex volume containing all the hand catch positions.

The prediction time gain for reaching within 6 cm of the goal can also be studied for individual throws (not visible in the plots). It is roughly normally distributed, and better than 100 ms for 15% of the throws, better than 70 ms for 50%, and worse than 30 ms for 15% .

When used in a teleoperation system, a relevant metric of the current prediction approach is its ability to discriminate between attempts to catch a flying ball and other hand motions. This will be most difficult when the user is performing a quick hand motion at a point in time consistent with trying to catch the ball, but for a different target. Therefore, we cross-tested each hand trajectory not only against the ball trajectory used when it was recorded, but also against all others. In this case, as an online classifier will not have access to the hand catch position, the proximity to the ball trajectory was used instead. The ball trajectory is well estimated by the vision system by the time the user has reacted.

In Figure 8, the classification results are shown for four different times relative to impact. Also shown are the threshold distances from the predicted endpoint at that time to the nearest point on the ball trajectory. As expected there is a trade-off between getting few false positives (classifying a hand trajectory associated with a different throw as a

(a) Selected hand trajectories.

(b) All hand trajectories.

Fig. 8. Classification performance. On the x axis is the number of misclassifications of hand trajectories as catch attempts when compared to an unrelated ball trajectory, divided by the maximum possible number of such misclassifications. The y axis shows the number of catch attempts not classified as such when compared to their associated ball trajectory. Thick curves represent performance at constant times before impact, and thin ones do the same for constant threshold distances. See text.

catch attempt) or few false negatives (not recognizing a hand trajectory as a catch attempt for its own throw).

Depending on the relative importance of these two parameters, the optimum threshold distance seems to be between 8 cm and 12 cm, where there are about 10% false positives and 20% false negatives for the most relevant time of 200 ms and the selected set of hand trajectories. For 300 ms the performance is much worse in agreement with Figure 6, and 100 ms is simply too little to have time to do anything useful. Also note that at 0 ms (impact), with perfect knowledge of the hand catch position, perfect classification is still not achieved. This is because hand position is not exactly on the ball trajectory even for a catch, and several ball trajectories pass very near each other.

VII. CONCLUSIONS

The present paper has shown how the minimum jerk kinematic model of unconstrained human motion can be used to predict user input in a teleoperation scenario. After the first half of a hand motion has been observed, an endpoint prediction good enough for determining the intended target can be made.

For the ball catching task described, the single largest delay in the system is the user's reaction time of typically 300–400 ms before any motion occurs. To shorten this delay, the current implementation of the ball catching system, which includes predictive display of the ball and robot to overcome communication delays, could be set to show the user the ball a little ahead of real time. This would force the user to react earlier, which might be required for catching the ball by the dynamics of the robot.

ACKNOWLEDGMENT

The research presented in this paper is funded by the 6th EU Framework Program, FP6-IST-001917, project name Neurobotics.

REFERENCES

- [1] K. J. Åström and B. Wittenmark, *Adaptive Control*. Addison-Wesley, 1995, second edition.
- [2] M. Kawato, "Trajectory formation in arm movements: Minimization principles and procedures," in *Advances in Motor Learning and Control*, ser. Human Kinetics, H. N. Zelaznik, Ed., 1996, pp. 225–259.
- [3] A. Hauck, M. Sorg, and T. Schenk, "What can be learned from human reach-to-grasp movements for the design of robotic hand-eye systems?" in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 4, Detroit, Michigan, USA, 1999, pp. 2521–2526.
- [4] C. Smith and H. I. Christensen, "Using COTS to construct a high performance robot arm," in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*. Rome, IT: IEEE, April 2007.
- [5] M. Bratt, C. Smith, and H. I. Christensen, "Design of a control strategy for teleoperation of a platform with significant dynamics," in *IROS-06*. Beijing, China: IEEE/RSJ, oct 2006.
- [6] O. J. M. Smith, "A controller to overcome dead time," *ISA Journal*, vol. 6, pp. 28–33, Feb 1959.
- [7] N. Hogan, "An organizing principle for a class of voluntary movements," *Journal of Neuroscience*, vol. 4, pp. 2745–2754, 1985.
- [8] 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, July 1985.
- [9] T. Flash and E. Henis, "Arm trajectory modification during reaching towards visual targets," *Journal of Cognitive Neuroscience*, vol. 3, pp. 220–230, 1991.
- [10] T. Gat-Falik and T. Flash, "The superposition strategy for arm trajectory modification in robotic manipulators," *IEEE Transactions on Systems, Man, And Cybernetics—Part B: Cybernetics*, vol. 29, no. 1, pp. 83–95, Feb 1999.
- [11] T. E. Milner, "A model for the generation of movements requiring endpoint precision," *Neuroscience*, vol. 49, no. 2, pp. 487–496, 1992.
- [12] N. Krüger, "Entwicklung einer humanoiden ball-fang-strategie für ein roboter-hand-arm-system," Master's thesis, Technischen Universität München, 2006.