Teleop Command/Imitation Learning/Regression (in progress)

Rahul Peddi and Nicola Bezzo

***Abstract*—**

I. Introduction

II. Related Work

III. Problem Formulation

In this work, we are interested in finding a policy that enables autonomous UAV navigation over a user-defined trajectory.

Formally, the problem we investigate in this work can be stated as:

♠[1] **Problem 1:** ***Demonstration-based autonomous control generation***: A UAV has the objective to navigate over a certain trajectory. Trajectories are defined by a series of goal locations and times at which the goals are reached. Given m flight demonstrations, find a policy to

1) process and analyze the data in the training sets.
2) identify boundaries for user trajectories such that $d_{\min} \leq d_u \leq d_{\max}$ and $t_{\min} \leq t_u \leq t_{\max}$, where d_u and t_u denote user-set targets, or possibly waypoints within a trajectory, for the UAV.
3) generate autonomous commands, $\mathbf{x}_a$.
4) correct autonomous commands during run-time $\mathbf{x}_c$, subject to a constraint that UAV should stay within a certain distance of the user-set trajectory:

$$||\mathbf{p}(t) - \mathbf{p}_r(t)|| \leq \delta, \ \forall t \in [0, T] \tag{1}$$

where $\mathbf{p}_r(t)$ is the reference position at time t, where T represents an overall time horizon for the trajectory, and δ is a threshold for allowable deviation.

IV. System Dynamics

The type of UAV in question is modeled using a 12^{th} order state vector. ♠[2]

V. Approach

In our approach, we leverage the data from multiple demonstrations to build a regression model that enables identification of an appropriate integral, g_u ♠[3], and average velocity, $\bar{x}_u$ given a user-set distance, d_u, and time, t_u. With the appropriate integral and average velocity, the demonstrated command string with the closest integral, g_i, to the fitted integral, g_u, is resized to match the user-set time, t_u and fitted average velocity, $\bar{x}_u$, to obtain the autonomous command string $\mathbf{x}_a$. This command is sent to the physical UAV for implementation, where the on-board computer ♠[4] collects data regarding the error between actual position, $\mathbf{p}(t)$, and the expected position, $\mathbf{p}_r(t)$ is determined based on the trajectory. This error is reduced by using a proportional controller on the input $\mathbf{x}_a$ to obtain the adjusted input commands, $\mathbf{x}_c$. ♠[5]

Block Diagram

A. Regression Based Training and Evaluation

In order to build the appropriate policy for autonomous command generation, we perform offline training on data collected over m human-piloted trials. Because the goal indicated in our problem statement is to be able to track a certain trajectory, the inputs of the training phase are $\{d_i, t_i, \mathbf{x_i}\}$ ♠[6], where $i = 1, \ldots, m$, d_i is the distance travelled in each trial, t_i is the length of the trial, and $\mathbf{x_i}$ is the string of teleoperation commands given the by the pilot. In addition to the inputs given by the pilot, we are also interested in two entities ♠[7]

- The integral of the string of teleoperation commands, $g_i = \int_0^{t_i} \mathbf{x_i}$
- The steady-state teleoperation command, $\bar{x}_i \in \mathbf{x_i}$

The steady-state command and integral are necessary portions of the analysis because of the directly proportional relationship between teleop commands and system velocity ♠[8]. The steady-state command, in this case, provides information about the pilot's average in-flight velocity, and is obtained by taking the mean of all entries in $\mathbf{x_i}$. The integral, meanwhile, describes the area under the string of commands. Because the teleop commands are proportional to velocity, this area value gives important information about the distance traveled, as position is the integral of velocity over time.

The offline training data is applied to a thin-plate spline surface fit to describe the relationship between training inputs and integral and average velocity ♠[9]. The general form of a

Rahul Peddi and Nicola Bezzo are with the Department of Systems and Information Engineering and the Charles L. Brown Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22904, USA. Email: {rp3cy, nb6be}@virginia.edu

[1]NB: problem is still not good. What is the problem that you are trying to solve??? It's not processing and analyzing data!

[2]NB: what's the point of this section? What are you trying to demonstrate here?

[3]NB: why? You cannot state this without first explaining what is this integral

[4]NB: ?? on board computer collect data about position?

[5]NB: Rewrite

[6]NB: explain how and what you are assuming here. A reader will be confused on how and what you are doing. The reader was nto present during experiments

[7]NB: mention first why the integral and the ss teleop

[8]NB: why?

[9]NB: why?

thin-plate spline equation is

$$f(x,y) = a_1 + a_x x + a_y y + \sum_{i=1}^{m} w_i U(||(x_i,y_i) - (x,y)||) \quad (2)$$

where $a_1, a_x,$ and a_y are scalar coefficients, w_i is a coefficient that corresponds to each specific trial, subject to the following condition:

$$\sum_{i=1}^{m} w_i = \sum_{i=1}^{m} w_i x_i = \sum_{i=1}^{m} w_i y_i = 0 \quad (3)$$

and the function U is of the form

$$U(r) = r^2 \log r \quad (4)$$

Given the corresponding z_i for each (x_i, y_i) pair, we are able to solve the following linear system to obtain the coefficients $w_i, \ldots, w_m$ and a_1, a_x, a_y,

$$\begin{bmatrix} K & P \\ P^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{z} \\ \mathbf{o} \end{bmatrix} \quad (5)$$

where $K_{ij} = U(||(x_i, y_i) - (x_j, y_j)||)$, $P_i* = (1, x_i, y_i)$, $\mathbf{0} \in \mathbb{R}^{3\times3}$ is a matrix of zeros, $\mathbf{o} \in \mathbb{R}^{3\times1}$ is a column vector of zeros, $\mathbf{w} \in \mathbb{R}^{m\times1}$ and $\mathbf{z} \in \mathbb{R}^{m\times1}$ are formed from w_i and z_i, respectively, and $\mathbf{a}$ is the column vector with elements a_1, a_x, a_y.

Given the general framework for performing the thin-plate spline, we develop two separate relationships for our specific application: $g_i = f(d_i, t_i)$ and $\bar{x}_i = h(d_i, t_i)$ ♠[10] With the functions we have obtained, we are able to find an estimated integral and steady-state command, g_u and $\bar{x}_u$, for any given desired distance and time,

$$g_u = f(d_u, t_u) \quad (6)$$

$$\bar{x}_u = h(d_u, t_u) \quad (7)$$

where d_u and t_u are the user-set desired distance and time, respectively.

While a thin-plate spline is continuous, and a result can be obtained with any combination of distance and time as an input pair, the accuracy of the results of any pair (d_u, t_u) can suffer as the distance between evaluation points and training points increases ♠[11]. In order to quantify this, we leverage the standard error of the estimate, which is a statistic used to measure the accuracy of predictions given a certain type of regression with known values:

$$\sigma_{est} \approx \frac{s}{\sqrt{m}} \quad (8)$$

where s is the sample standard deviation of all of the points in the training set and m is the number of training samples. In (8), an increased sample standard deviation results in ♠[12]. The standard error, σ_{est}, is then used to set the bounds for each training data point, for example:

$$\beta_{\min} = \hat{\beta} - \sigma_{est}$$
$$\beta_{\max} = \hat{\beta} + \sigma_{est} \quad (9)$$

where $\beta_{\min}$ and $\beta_{\max}$ are the lower and upper bounds for parameter $\hat{\beta}$.

In our case, there are two parameters we are primarily concerned about for prediction; distance and time. As a result, we perform this calculation twice over, treating each of the two parameters as statistically independent, to obtain a generic interval for each point in the training set, denoted $[d_{i\,\min}, d_{i\,\max}]$ and $[t_{i\,\min}, t_{i\,\max}]$, where $i = 1, \ldots, m$. Because we are ultimately interested in using the distance and time values together as input pairs, we then obtain a maximum Euclidean distance from each of the training data points using:

$$\Delta_i = \sqrt{(\sigma_{d_{est}})^2 + (\sigma_{t_{est}})^2} \quad (10)$$

Using the maximum distance for each point Δ_i, we are able the generate intervals around each point, where the data is within the standard error of the estimate ♠[13].

B. Autonomous Behaviour Generation

In order for the system to autonomously reach a user-set goal, g_u and $\bar{x}_u$ are obtained using equations (6) and (7), and the offline training samples are leveraged to generate a new string of commands.

From the training set of m samples, we select the trial that has the closest integral, g_i, to the estimated integral g_u. This is done by forming an error vector, $\mathbf{e} \in \mathbb{R}^{1\times m}$, where each element is defined by ♠[14]

$$e_i = |a_i - a_u|, \; i \in \{1, \ldots, m\} \quad (11)$$

The lowest error is then found and is paired with the appropriate pre-trained sample, $\mathbf{x}^*$,

$$\mathbf{x}^* = \mathbf{x}_i \in \mathbf{x} | e_i = \min_e(\mathbf{e}) \quad (12)$$

This optimal pre-trained sample is then adjusted to reflect the user-set time, t_u. This is done by performing bicubic interpolation to resize the vector $\mathbf{x}^*$. Bicubic interpolation is the chosen method for resizing, as it performs better than nearest-neighbor and bilinear interpolation methods, while only marginally increasing computational complexity. The general form of a bicubic interpolation equation is

$$p(x) = \sum_{i=0}^{3} a_i x^i \quad (13)$$

where x is an entry in vector $\mathbf{x}^*$ and a represents the coefficients of the function at each point. Bicubic interpolation takes the weighted sum of the four nearest neighbors

[10]NB: how did you get these relations?
[11]NB: why?
[12]NB: ???
[13]NB: really confusing and not sure about correctness and why this works
[14]NB: what is ai and au?

of each entry in the command vector in order to identify the intermediate points between each value in $\mathbf{x}^*$. After resizing, we obtain the time adjusted input vector $\mathbf{x}'$.

The next step is to adjust the input vector such that the system reaches the user-set goal d_u. This is done by leveraging the average velocity information, that is, $\bar{x}_u$. Because distance is a function of average velocity and time, the scale time-adjusted vector $\mathbf{x}'$ is scaled such that its mean is equivalent to $\bar{x}_u$. We then obtain

$$\mathbf{x}_a = \mathbf{x}' \left(\frac{\bar{x}_u}{\bar{x}'} \right) \tag{14}$$

The input $\mathbf{x}_a$ is then sent to the UAV to reach the goals set by the users.

♠[15]

C. Online Adaptation of Generated Commands

The commands generated in the previous section are generated and sent to the UAV prior to any testing or evaluation. Therefore, we propose a method to correct for any error that may occur during evaluation.

D. Trajectory Decomposition

VI. Experiments

VII. Conclusions

VIII. Acknowledgement

References

[1] G. O. Young, Synthetic structure of industrial plastics (Book style with paper title and editor), in Plastics, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 1564.

[2] W.-K. Chen, Linear Networks and Systems (Book style). Belmont, CA: Wadsworth, 1993, pp. 123135.

[3] H. Poor, An Introduction to Signal Detection and Estimation. New York: Springer-Verlag, 1985, ch. 4.

[15] NB: this section needs to be rewritten...it's confusing