Shared Control With Efficient Subgoal Identification and Adjustment for Human–Robot Collaborative Tasks

Zongyao Jin and Prabhakar R. Pagilla

***Abstract*— In this article, we describe a novel method to facilitate the process of subgoal identification from a robotics task demonstrated by a human operator. The method defines a unified metric to quantify the human operator's commands, which can be employed to effectively extract subgoal distributions and identify subgoals. We then address the problem of online subgoal adjustment for shared control (SC) applications, which is initiated by the human operator and is necessary to sustain performance in a dynamically changing environment. The adjustment is facilitated by designing a hyperrectangle around the subgoal whose volume is obtained via a hyperbolic slope transition function (HSTF) based on the distance between subgoals. The adjustment actions within the hyperrectangle are encoded by a skill-weighted action integral. A practical implementation of the automatic control input is also provided where proper scaling for input-blending is considered. The developed SC method is corroborated on a scaled hydraulic excavator platform with human operators. Experimental results are presented and discussed.**

***Index Terms*— Blended shared control (BSC), collaborative robotics, human performance augmentation, human-centered automation, robotics in construction.**

I. Introduction

HUMAN–MACHINE shared control (SC) for improving the performance of machine operators is an emerging field where the goal is to combine the benefits of robust situation awareness of human operators with the efficiency and precision of automatic control. Effective modeling of a given task as a set of subgoals using data obtained from a skilled human operator's demonstration is an essential building block for applying SC. With meaningful subgoals, one can provide automatic control assistance to the human operator by predicting the operator's intent of seeking different subgoals and blending human and automatic control inputs with proper blending weights. The appropriate fusion of human and automatic control inputs with the framework of SC [1], [2] has been shown to result in improving the operators' performance [3]–[5], situation awareness, and safety [6], [7].

Blended SC (BSC) typically involves three key components, namely, task learning, intent prediction, and input blending. That is, given a human–machine collaboration task, the first step toward BSC is to model the task from a human operator's demonstration into a collection of subgoals where each subgoal represents a machine state. At each state, there are actions taken by the human and the automatic control in order to transition into different subgoals based on the operator intent. A correct sequence of such subgoal transitions is expected to result in task completion. Once the task is modeled with subgoals, it is then possible to predict, from the real-time actions of the human operator, which transitions are intended and with what probabilities. With such predicted information and the fact that automatic control can provide assistance to transition from one subgoal to another, BSC can significantly improve the performance of the human operator during task execution.

Policy models are commonly employed for learning a robotics task from the demonstrations of a human operator [8] to fully automate the execution of the task using the robot. However, for SC applications where the human is integral to the task execution, researchers often choose subgoal models [5], [9], [10]. The advantage of dividing the task into subgoals is that, when a human operator is in-the-loop executing the task, one can predict the human operator's subgoal-seeking intent and provide automatic control assistance based on the prediction [4], [11]. The automatic control assistance is often provided in the form of BSC, where the human input is blended with the automatic control input generated based on the predicted human intent. BSC has been applied in various robotics applications, such as power wheelchair, rehabilitation and surgery robots, robotic manipulator operations, and airplane and boat navigation [9], [12]–[25]. In [4], subgoals are employed with BSC to assist human operators in navigating and finishing excavator earth-moving tasks for performance improvement; the concept of termination sets based on subgoals was proposed, and within those termination sets, automatic control yields its authority completely to the human operators. One important reason for creating termination sets is to compensate specifically for robotic tasks where the target object may change shape or location during task execution, and therefore, this aspect must be addressed by, if not by adjusting the subgoals, yielding the control authority to the human operator at the cost of providing less assistance.

The developments in this article build on the authors' prior work [5], [16], and the significant contributions of this work over prior work are as follows.

1) A novel approach for identifying subgoals by quantifying the operator's command pattern and applying

Manuscript received January 9, 2021; accepted February 15, 2021. Date of publication March 29, 2021; date of current version December 15, 2021. Manuscript received in final form March 5, 2021. This work was supported by the NSF National Robotics Initiative (NRI) under Grant 1527828. Recommended by Associate Editor Y. Pan. *(Corresponding author: Prabhakar R. Pagilla.)*

Zongyao Jin is with Flexiv Robotics Ltd., Santa Clara, CA 95054 USA (e-mail: zongyaojin@outlook.com).

Prabhakar R. Pagilla is with the Department of Mechanical Engineering, Texas A&M University, College Station, TX 77840 USA (e-mail: ppagilla@tamu.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCST.2021.3064801.

Digital Object Identifier 10.1109/TCST.2021.3064801

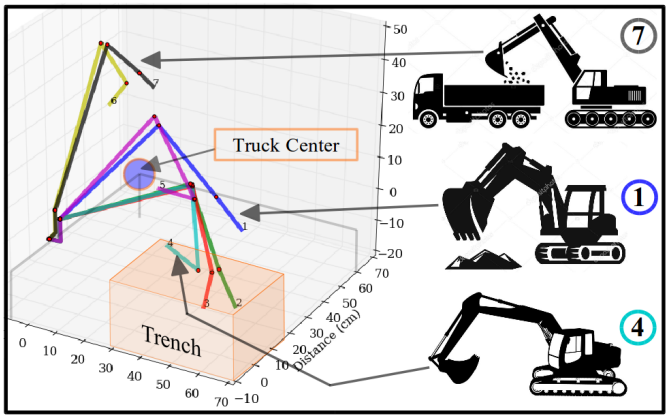

Fig. 1. Illustration of subgoals in joint space.

statistical inference is provided. The quantification is achieved by introducing the concept of motion command variance (MCV), which provides a unified metric for representing the complexity of the human command patterns. MCV can be used to observe the evolution of command pattern complexity and detect command changes leading to subgoal distributions.

2) A method for subgoal adjustment by dynamically capturing adjustment information from operator actions and encode such capabilities to BSC is developed. The concept of adjustment encoding hyperrectangle AEHR) to collect information for subgoal adjustment is provided. Then, a method for encoding this adjustment information by taking the skill weighted action integral (SWAI) of the operator input when the robot state is within the AEHR is provided.
3) A practical implementation of automatic control assistance with a dynamic sigmoid controller (DSC) for input blending is provided. The DSC addresses the scaling problem for input blending and uses the operator's effective magnitude (OEM) to change the overall controller gain according to the magnitude of the portion of the operator's input for seeking the predicted subgoal.
4) Finally, results from comprehensive testing of the approach on a scaled hydraulic excavator platform with the blending of a human operator and automatic control inputs are described for typical trenching and truck loading tasks in earth-moving applications.

The rest of this article is organized as follows. The need for employing a unified metric to identify task subgoals and adjusting subgoals in SC applications is described in Section II. Section III introduces the concept of MCV, the procedure, and its implementation for extracting subgoal distributions. A method for blending control inputs with DSC is discussed in Section IV. Section V describes the proposed subgoal adjustment technique. Experimental results are discussed in Section VI. Concluding remarks and potential future work are provided in Section VII.

II. Subgoals and Problem Motivation

In the context of human-operated robot tasks, subgoals can be key points in the robot joint space, which corresponds to different robot configurations or points in the Euclidean workspace that corresponds to different end-effector positions. To illustrate the concept of subgoals, consider the excavator shown in Fig. 1, performing trenching and truck loading tasks. During trenching and truck loading tasks, the excavator bucket is first positioned on the top of the trench, the bucket is inserted into the trench and moved to draw a load of dirt by curling the bucket and moving the forearm toward the excavator body at the same time, and then, the bucket is lifted from the trench and moved to the top of the desired spot and released. If the trenching area (environment) does not change, these crucial task steps (or way-points) can be modeled by subgoals via learning from demonstration under nominal conditions. However, as the task is repeated many times to move dirt, the trench typically gets deeper, and the excavator has to move the bucket further down in order to pick up a full load of the bucket. The same situation applies to other excavator operations, such as moving dirt from a pile (earth moving) where dirt from a loose pile is moved and the pile geometry and size keeps changing. In Fig. 1, seven nominal subgoals (excavator configurations in wire frame) are shown; by following such subgoals in a correct sequence, the operator can successfully complete the task. Each subgoal, in this case, is a point in the joint space but can be instantiated by a robot configuration via forward kinematics, shown as poses 1, 4, and 7 on the right.

A. Unified Metric for Subgoal Identification

Subgoals can be detected by monitoring the robot states or operator input (via input devices such as a joystick) from demonstrations of the task. Significant changes in robot states or operator input typically indicate achieving one subgoal and initiation of seeking the next subgoal. To capture and quantify these changes, primitive-based methods [4], [5], [9] consider thresholds to categorize the operator input or joint actuator velocities. For example, when the operator input crosses a given threshold, it is considered that a noticeable pattern change is detected, i.e., a subgoal is achieved and the operator is switching to another subgoal by initiating a different input pattern. The associated robot states and the corresponding configuration are recorded as a potential subgoal. Typically, many potential subgoals will be detected from a demonstration and would form subgoal distributions. Statistical inference tools may then be employed to cluster these distributions and identify subgoals from the clusters. These primitive-based methods are shown to be effective with the advantage of computational simplicity and suitable for many SC applications. However, they are generally sensitive to noise, which could cause the measured signal to cross the threshold. A unified metric as opposed to thresholds can efficiently quantify command changes that indicate subgoal transitions.

B. Necessity for Adjusting Subgoals

Subgoals for a given task can change significantly over time after many repeated cycles of the task. For example, consider again the excavator trenching task. Fig. 2 illustrates a gradual

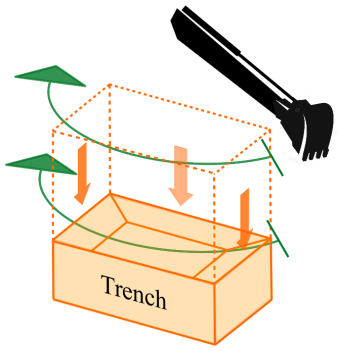

Fig. 2. Gradual but cumulatively significant changes in trench geometry.

but cumulatively significant change of the trenching area and depth. These changes typically can be adapted by making small adjustments to the existing nominal subgoals during each task cycle while having the operation patterns remain largely the same. These subgoal adjustments are essential for improving the efficiency of BSC. However, existing BSC methods consider only nominal subgoals without making any adjustments resulting from task operation and operator command changes over time. Since nominal subgoals are assumed all the time, there is the gradual deterioration of automatic control assistance. Human operators, with their unique sensory abilities, environmental awareness, and domain knowledge, can generally make decisions and take actions to adapt to the subgoal changes. We can obtain meaningful interpretations of such decisions or actions in real time via human operator intent, which can be encoded into dynamic subgoal adjustments that can lead to improving BSC performance.

III. Extracting Subgoal Distributions With Motion Command Variance

In many industrial applications where the robot is operated by a human, the operator typically controls the velocity of each DOF independently, i.e., the operator inputs are mapped to velocities of different robot actuators. In some other applications, operator inputs get mapped via robot kinematics to motions/velocities of a robot (or its end-effector) in the Cartesian space. Since the operator commands initiate motions, we refer to those as motion commands. To quantify the changes in operational motion commands, we develop a unified metric called the MCV whose formulation and construction are described in the following.

The motion command initiated by the human operator for an m-DOF robot at a sample time t can be described by a vector of the form

$$\boldsymbol{o}(t) = \left[o_{1t}, o_{2t}, \ldots, o_{mt}\right]^T \tag{1}$$

where $o_{it}(t),\ i \in 1, 2, \ldots, m$ denote the operator input, or the velocity it gets mapped to, for each robot joint actuator or motion axis. For a time interval starting from t and containing n samples of motion commands with a sampling period of k, the motion commands initiated by the operator can be compactly described by the matrix

$$\boldsymbol{O}(t) = \left[\boldsymbol{O}_1, \boldsymbol{O}_2, \ldots, \boldsymbol{O}_n\right]^T \tag{2}$$

where $\boldsymbol{O}_i = \boldsymbol{o}(t + ik - k)^T,\ i \in 1, 2, \ldots, n$. Let $[O_1, O_2, \ldots, O_m]^T$ denote a random vector representing the operator command input; then, $\boldsymbol{o}(t+ik-k)^T,\ i \in 1, 2, \ldots, n$ are its realizations. The covariance matrix of the motion commands captured in such a time interval is then given by

$$\boldsymbol{\Sigma}(t) = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1m} \\ \Sigma_{21} & \Sigma_{22} & \cdots & \Sigma_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m1} & \Sigma_{m2} & \cdots & \Sigma_{\mathrm{mm}} \end{bmatrix} \tag{3}$$

where

$$\Sigma_{ij} = E[(O_i - \mu_i)(O_j - \mu_j)] \tag{4}$$

E denotes the expectation operator, $\mu_i = (1/n)\sum_t o_{it}$, and $\mu_j = (1/n)\sum_t o_{jt}$. The covariance matrix of motion commands not only encodes data correlation but also contains information about subtask switching (indication of occurrence of subgoals) and can be interpreted as follows. During a given time interval, if the operator focuses on one subtask, the commands should be similar; if the operator has just finished one subtask and switched to another, the input commands to the robot during this time interval should be observably different and revealed as having large variances. To identify the subgoals for a given task using the covariance matrix, we further define the diagonal matrix

$$\boldsymbol{\Sigma}_d(t) = \mathrm{diag}\{\Sigma_{11}, \Sigma_{22}, \ldots, \Sigma_{\mathrm{mm}}\} \tag{5}$$

and use the following metric to quantify such information which we refer to as the MCV:

$$\mathscr{M}(t) = \boldsymbol{w}^T \boldsymbol{\Sigma}_{\boldsymbol{d}}(\boldsymbol{t})\boldsymbol{w} \tag{6}$$

where

$$\boldsymbol{w} = [w_1, w_2, \ldots, w_m]^T \tag{7}$$

and $w_1, w_2, \ldots, w_m \in \mathbb{R}^+$ are scalar weighting parameters. The MCV metric $\mathscr{M}(t) \in \mathbb{R}^+$ is then the weighted summation of the principal axes of the hyperellipsoid formed by samples of operator's input command each considered as a point in an m-dimensional space. If the operator initiated subtask switching with very different commands during the time interval, geometrically, these points in the m-dimensional space should be rather spread out. If one uses a multidimensional Gaussian distribution to model these points, the hyperellipsoid enclosing these points must be stretched along one or more of its principal axes, which corresponds to a large value for $\mathscr{M}(t)$ [26], [27]. A 2-D illustration is provided in Fig. 3 with a Gaussian distribution of $\mathscr{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu} = [0, 0]^T$ and $\boldsymbol{\Sigma}$ having 1 and 0.9 as its diagonal and off-diagonal elements, respectively. One observes that large differences in the operator command inputs in a given time interval can be quantified as stretching of the hyperellipsoid along its principal axes.

The weighted summation of lengths of the principal axes is used to handle situations where a designer would like to selectively emphasize the motion of one or more specific robot DOF, which will cause a more significant change of robot configuration compared to other robot DOF. Note that, although multiplication of the principal axes is directly related

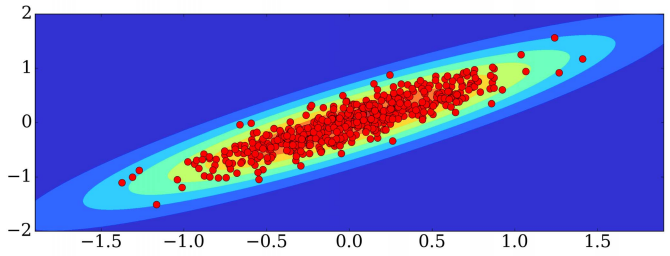

Fig. 3. 2-D illustration of encoding input commands (each red dot represents a sampled 2-DOF command input) using a multidimensional Gaussian and quantifying significant differences by observing the principal axes of the hyperellipsoid.

to the volume of the hyperellipsoid, it does not provide the refined differentiation to extract subgoals because, if one principal component is close to zero, the overall result will be close to zero.

We then introduce a design parameter $\gamma \in [0, 1]$ referred to as the MCV ratio; this design parameter facilitates capturing of the set of desired robot states from the demonstration data set denoted by χ, which satisfies the following:

$$\{x(t) \in \chi \mid \mathscr{M}(t) > \gamma \cdot \max_{\tau \in \Gamma} \mathscr{M}(\tau)\} \tag{8}$$

where $x(t)$ denotes the robot states at any sample time t, $\mathscr{M}(t)$ is the associated MCV, and Γ is the set of all sampled times for the entire demonstration. The set of robot states in (8), thus, represents the subgoal distributions, i.e., the MCV at those time instances is large, indicating that the operator's command changed significantly as a result of subtask-switching operations during the associated time intervals. Once the subgoal distributions are obtained, one can apply the Bayesian nonparametric clustering (BNPC) method, such as in [5], to identify the subgoals.

The implementation of extracting distributions of subgoals via MCV is given in Algorithm 1, where χ is the demonstration data set with ν data points. Each data point has an m-dimensional robot joint information associated with a time stamp. We assume that the total number of samples, denoted by $K \in \mathbb{N}^+$, collected during the entire demonstration is much larger than the number of samples $n \in \mathbb{N}^+$ that are utilized to calculate the MCV for a given time interval, namely, $n \ll K$, which implies that the operator's command pattern changes for subtask-switching occur in a much shorter time interval relative to the time duration of the entire task demonstration. Note that practical implementation requires appropriate scaling of the measurement data from different joints because different types of robot joints (prismatic and revolute) or body motions (translation and rotation) may have measurement values that belong to different ranges. We can normalize each input or robot state to the range of [−1,1] by applying the following transformation:

$$f(y_t) = \frac{2(y_t - y_{\min})}{y_{\max} - y_{\min}} - 1 \tag{9}$$

where y_t denotes the input or robot state at a sample time t, and $y_{\max}$ and $y_{\min}$ denote the maximum and minimum values, respectively, of $y_k, \forall k \in \Gamma$.

Algorithm 1 Extracting Distributions of Subgoals via MCV

Input: Data set $\chi \in \mathbb{R}^{(m+1)\times\nu}$, MCV ratio $\gamma \in [0, 1]$, number of samples $n \in \mathbb{N}^+$ for the MCV time interval

Output: Subgoal distributions in the form of a set of data points

1: **for** i in $1, 2, \cdots, K-n$ **do**
2: Collect n samples starting from index i in time sequence each in the form of (1)
3: Construct matrix in (2) with the n collected samples
4: Compute MCV value using (3) and (6) and attach the result to each data point
5: **end for**
6: Store and output all the data points with MCV values satisfying (8)

IV. Blending Human and Automatic Control Inputs

In the setting of hydraulic excavators and many other construction robots, the human operator typically commands the velocity of each excavator actuator independently via a joystick input. With the intent prediction based on subgoals, the automatic control assistance to the human for seeking the subgoal generates an input to each actuator to track the desired position. More specifically, for the hydraulic excavator, the automatic control generates input for each actuator, corresponding to the velocity of a hydraulic cylinder, by calculating the error between its current and desired position. Thus, with the predicted subgoal and the prediction probability, the BSC input for each actuator is given by

$$u_b = (1 - p^*)u_h + p^* u_a \tag{10}$$

where $p^* \in [0, 1]$ is the blending parameter, which is selected based on the prediction confidence of seeking a subgoal [5], and u_h and u_a denote the operator input and automatic control input, respectively. For the automatic control part of the blended input, we can consider any type of established controller design with the desired performance that is relevant to the particular robot.

From an implementation viewpoint, we have to consider two additional aspects: first, normalizing the scale of the automatic control and human inputs to facilitate appropriate blending, and second, relating the speed at which the operator is trying to seek a subgoal and the overall gain of the controller. A large operator input magnitude toward a subgoal would indicate that the operator wants to reach the subgoal faster; one should appropriately increase the controller gain to enhance the assistance, and vice versa. We propose the concepts of DSC and OEM to address these two aspects.

The DSC is given by

$$u_a = \frac{2\iota}{1 + e^{-\zeta\epsilon}} - \iota \tag{11}$$

where ϵ is the error between the current measured position and the desired position based on the predicted subgoal, the parameter ι is related to the limits of control saturation, and the parameter ζ controls the controller gain and is related

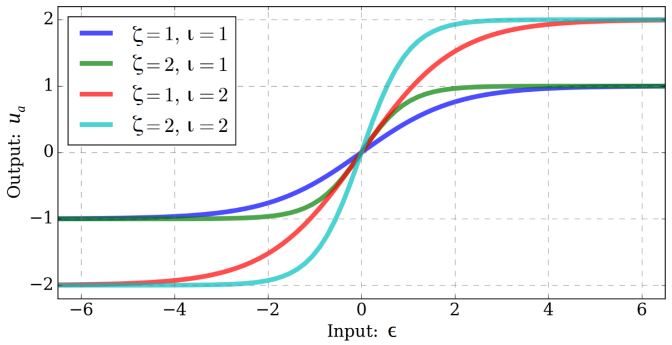

Fig. 4. Parameter selection versus DSC.

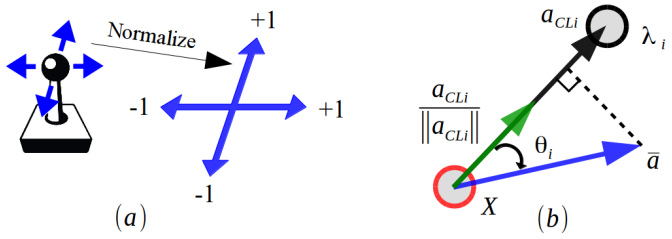

Fig. 5. Normalized joystick axes and visualization of effective magnitude. (a) Normalized joystick input range. (b) Effective magnitude projection.

to operator's input magnitude. Fig. 4 shows some examples of the mapping of DSC with different parameter selections. With the upper and lower saturation limits imposed to the input from automatic control, it is then possible to scale the operator's input accordingly such that u_h and u_a in (10) are in the same scale before they are blended to provide overall input.

To relate the magnitude of the operator's input to the controller gain for tracking the predicted subgoal, we propose the concept of OEM. It is defined as the magnitude of the portion of the operator's input effectively acting along the same direction as the closed-loop action for tracking the predicted subgoal. We employ the following method based on vector projection to quantify such OEM. Let ω be the effective magnitude, which is given by

$$\omega = \overline{a} \cdot \frac{a_{CLi}}{||a_{CLi}||} = ||\overline{a}|| \cos\theta_i \tag{12}$$

where θ_i denotes the dynamic angle between operator's action vector and the closed-loop action vector a_{CLi} for tracking the predicted subgoal λ_i, and $\overline{a}$ is the normalized operator's action vector with each of its element normalized along the corresponding human input device axis (joystick for example). To illustrate this, Fig. 5(a) provides the axiswise normalization of the operator's action input, where each joystick axis is normalized to take a value in the interval $[-1, 1]$. Fig. 5(b) provides a 2-D illustration for calculating the OEM.

With such a formulation, it is then possible to relate the controller gain parameter ζ in (11) to the operator's input in a manner that the effective magnitude is proportional to the controller gain. We propose that, in (11), the parameter ζ is

$$\zeta = \nu\omega \tag{13}$$

where $\nu \in \mathbb{R}^+$ is a design parameter that scales the proportionality between the effective magnitude ω and the controller gain parameter ζ. Larger values for ν in (13) result in faster response. One can assign a smaller value for ν to obtain a controller gain that is more sensitive to the effective magnitude of the operator's input; for example, assistance in surgical robots where the systems have fast dynamics and operational precision is critical. For the excavator application, a larger ν value is chosen because the hydraulic dynamics of the excavator are slow. Also, notice that the operator's input along different axes gets normalized to the range of $[-1, 1]$ to facilitate the formulation of effective magnitude. It is then suggested, in practice, to set ι such that both inputs are in the range of $[-1, 1]$ for blending. The blended input then gets mapped to the required range of the physical actuation signals, for example, PWM duty cycles or voltage levels, by a proper affine transformation.

V. Dynamic Subgoal Adjustment

Subgoal adjustment is necessary in order to adapt to the gradual but cumulatively significant change of environment and target objects. The adjustments, based on our observations of excavator operators during task execution, are realized by making small modifications to the initial/nominal subgoals. Such observations coincide with the results from extensive research in psychology [28], i.e., complex human behavior is learned through the modification of simpler behaviors, especially when the environment change is gradual. For example, operators in excavator trenching usually make small adjustments to the subgoals to stay productive, such as removing more dirt. It is reasonable to assume that such knowledge is reflected in the operator's actions taken for each robot joint near the subgoal, which needs to be adjusted. In the following, we will describe how to adjust the subgoals based on operator actions by utilizing the notions of a hyperrectangle, which embeds the subgoal with proper volume, and SWAI, which accounts for the amount of adjustment needed.

A. Adjustment Encoding Hyperrectangle (AEHR)

In order to detect the operator's subgoal adjustment information, with the assumption that such behavior appears near the target subgoal, we define an AEHR, denoted by Δ. It is the generalization of a rectangle for higher dimensions and is dynamically created and centered at the predicted target subgoal. The AEHR also dynamically changes its center and length of the edges if the prediction updates the target subgoal. The AEHR either lies within the m-dimensional robot joint space, denoted by Ω, or overlaps with it. This process is illustrated in Fig. 6 with a 2-D visualization of the AEHR in the robot joint space Ω. At time t, as shown in Fig. 6(a), the prediction result is λ_j according to operator's action vector $a(t)$; hence, the AEHR is created and centered at λ_j. However, at time $t + t_\varepsilon$, as shown in Fig. 6(b), the prediction updates the target to λ_n based on the new action vector $a(t + t_\varepsilon)$; thus, the AEHR changes its center and edge length accordingly. The AEHR, thus, defines a dynamic vicinity around the target subgoal whose size is relevant to the position of the last visited subgoal and the predicted target subgoal. The size of AEHR is a function of the proximity of the last visited subgoal to the target subgoal, which is described next.

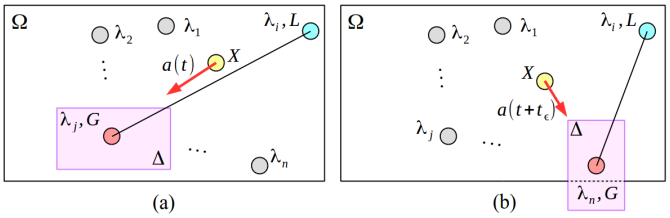

Fig. 6. Joint space Ω and AEHR Δ in 2-D. AEHR is (a) entirely and (b) partially included in robot joint space.

B. Hyperbolic Slope Transition Function (HSTF)

We will describe a method for finding the edge lengths of the AEHR through the HSTF, as illustrated in Fig. 7. The normalized distance (denoted by d_r) between subgoals along the rth coordinate of the m-dimensional joint space Ω is given by

$$d_r = \frac{|\lambda_{ir} - \lambda_{jr}|}{\max_{p,q} |s_p - s_q|} \tag{14}$$

where λ_{ir} and λ_{jr} denote the distance of subgoals i and j along the rth coordinate of the m-dimensional joint space, and s_p and s_q denote two arbitrary points along the same coordinate.

The edge lengths of Δ, denoted by δ_r, $r \in 1, 2, \ldots, m$, are given by the following HSTF:

$$\delta_r = \xi d_r + \frac{1 + \tanh[\kappa(\xi d_r - \mu)]}{2}(\mu - \xi d_r) \tag{15}$$

where $\kappa \in \mathbb{R}^+$ and $\xi \in (0, 1)$ are two design parameters. Note that the HSTF is designed such that: 1) when the subgoals are close to each other, we have $\delta_r = d_r$ and 2) when they are far apart, we have $\delta_r = \mu$, where μ is a constant number in the interval $(0, 1)$; the design parameter κ controls by how much (15) resembles the combination of these two situations.

An example illustration of HSTF with parameter values $\mu = 0.2, \kappa = 20$, and $\xi = 0.95$ is shown in Fig. 7(a), and two examples of the relationship between subgoals and edge length of Δ for a one degree of freedom revolute joint are provided in Fig. 7(b).

Remark 1: The particular HSTF given in (15) for the AEHR edge lengths is chosen for the following reasons. If the edge lengths are kept constant, then is an issue when the last visited subgoal and the predicted subgoal are close to each other; that is, the AEHR may very well encompass both of the subgoals, which effectively shuts down the prediction updates. If the edge lengths are some constant fractions of the distance between the two subgoals, another problem arises when the two subgoals are very far from each other: the size of the AEHR would be very large, which reduces the active region of the prediction algorithm. We would like the size of the AEHR to be large enough to accumulate and encode sufficient adjustment information from the operator and small enough such that the prediction is updated for active assistance as much as possible.

C. Skill Weighted Action Integral (SWAI)

When the robot is not in the AEHR, we predict operator's intent and update the target subgoal using methods described

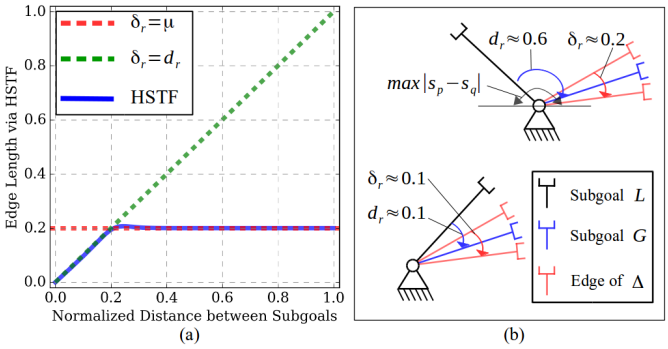

Fig. 7. (a) HSTF and (b) examples.

in [5]. Once the robot enters the AEHR, we stop the prediction algorithm and encode operator adjustment actions. We define a method for interpreting and encoding such operator adjustment actions via an SWAI defined by

$$\boldsymbol{\rho} = \beta \int_{t_0}^{t_f} \boldsymbol{a}(t) \mathbb{1}(\Omega \cap \Delta | X \neq G)\, dt \tag{16}$$

where t_0 and t_f, respectively, denote times at which the operator enters and exits the AEHR, β is a positive design scaling parameter (controls how much the designer wants to scale the integration results based on the knowledge of the skill level of the operator), $\rho \in \mathbb{R}^m$ is the vector denoting the adjustment for the target subgoal, $\boldsymbol{a}(t) \in \mathbb{R}^m$ is the action vector, and the indicator function $\mathbb{1}(\Omega \cap \Delta | X \neq G)$ is given by

$$\mathbb{1}(\Omega \cap \Delta | X \neq G) = \begin{cases} 1, & X \in (\Omega \cap \Delta) \text{ given } X \neq G \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

The indicator function provides a mechanism for SWAI to encode subgoal adjustment information only when the robot is within the intersection of Ω and Δ, and it has not reached the target subgoal. Fig. 8 shows an illustration of such conditions. Fig. 8(a) illustrates this concept by showing that, in Δ, the robot trajectory τ is perturbed by operator's subgoal adjustment actions $a(t)$, $a(t + t_\varepsilon)$. The closed-loop subgoal (G) tracking actions are denoted by $a_{CL}(t), a_{CL}(t + t_\varepsilon)$, and $a_{CL}(t + 2t_\varepsilon)$, and the positions of the robot are denoted by $X(t)$, $X(t+t_\varepsilon)$, and $X(t+2t_\varepsilon)$. Notice that, at the time $t+2t_\varepsilon$, the operator may be satisfied with the adjustment and might not take any additional action; thus, $a(t + 2t_\varepsilon) = 0$ is not shown in the figure. Fig. 8(b) illustrates another scenario where the SWAI finishes encoding adjustment information since the robot has left the AEHR without reaching the target subgoal.

Once the robot either reaches the subgoal or leaves the AEHR, as shown in Fig. 8, we update the last visited subgoal L to the previous prediction result G, which is the subgoal λ_k in the subgoal set λ. Then, adjustment to this subgoal is made in the m-dimensional robot joint space by

$$\lambda_k^+ = \lambda_k^- + \boldsymbol{\rho} \tag{18}$$

where λ_k^+ denotes the adjusted position and λ_k^- denotes its previous/nominal position.

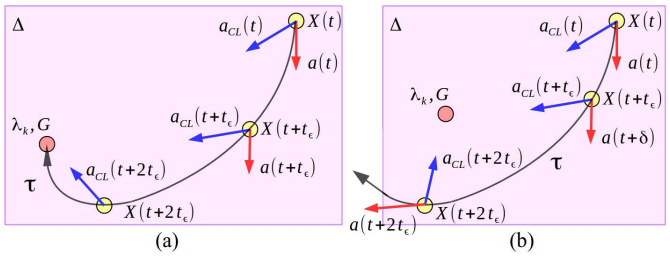

Fig. 8. Skill weighted action integration in the target subgoal vicinity. (a) Robot eventually reaches the subgoal. (b) Robot goes through the subgoal vicinity without visiting the subgoal.

Algorithm 2 BSC With Subgoal Adjustment

Input: Current position $X \in \mathbb{R}^m$ and velocity $V \in \mathbb{R}^m$, last visited subgoal $L \in \mathbb{R}^m$, all learned subgoals $\lambda \in \mathbb{R}^{m\times n}$, skill weighting $\beta \in \mathbb{R}^+$

Output: Adjusted $\lambda_k \in \mathbb{R}^m$ (if any adjustment detected)

Initialization Robot at L, create variable **S** indicating whether the robot is in AEHR Δ and set to False

while robot is not shutdown **do**
  if S is False **then**
    Predict target subgoal G via algorithm in [5]
    Dynamically blend control inputs by (10)
    Create Δ according to G and set edge lengths by (15)
    Set **S** to True when robot gets in to Δ
  else
    Statically blend control inputs
    Encode adjustment actions, if any, by (16)
    Set **S** to False and adjust λ_k by (18) if robot leaves Δ or G is reached
  end if
end while

A real-time implementation for BSC with subgoal adjustment, as discussed in this section, is provided in Algorithm 2.

Remark 2: An important observation is that only actions taken by the operator are encoded by the SWAI, which represents the operator's subgoal adjustment information. It is hard to extract such information from sensing only the robot states in a BSC scheme because the operator's action always gets blended with the automatic control input.

Remark 3: One additional aspect is that, once the robot enters the AEHR Δ, we stop the prediction updates and keep the last predicted subgoal as the target subgoal. Furthermore, we fix the blending parameter in (10) to a constant value such that the human operator's blending weight is always larger than that of the automatic control input for tracking the target subgoal. We also stop updating the OEM in (12) and fix it to its last computed value. Therefore, the SC assistance is still providing active assistance, but the operator can always overwrite the assistance to make arbitrary adjustments.

VI. Experimentation and Results

To test the approach and algorithms, a trenching and truck loading environment was set up with a 1/12th-scale hydraulic excavator platform, as shown in Fig. 9. Details of the platform can be found in [5].

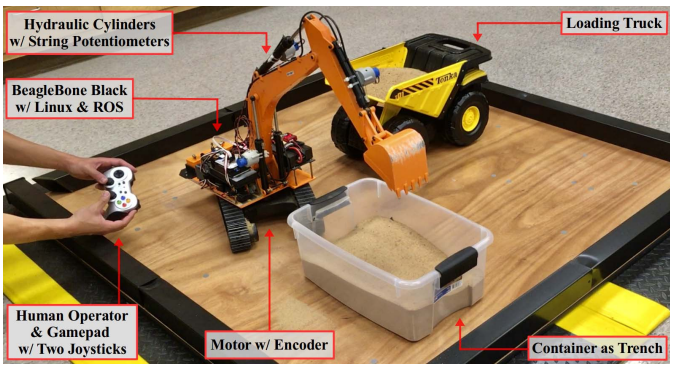

Fig. 9. Truck loading environment with a scaled hydraulic excavator.

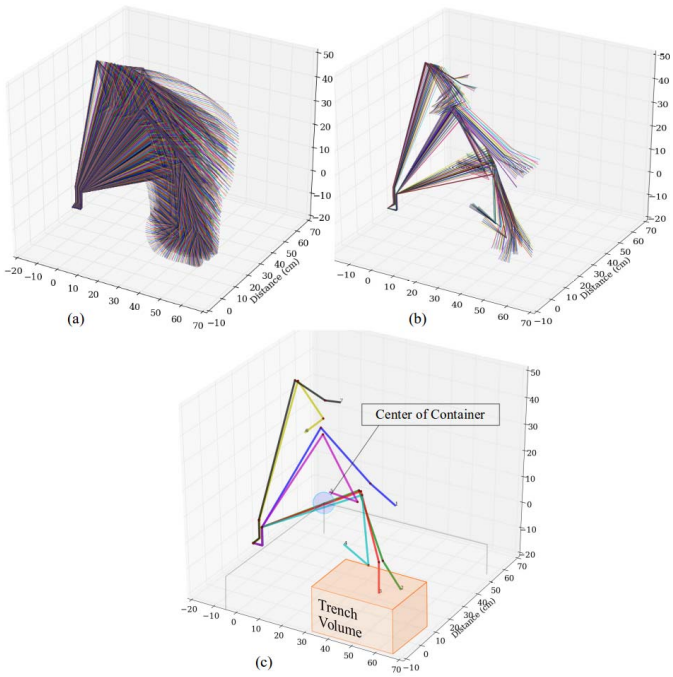

Fig. 10. Subgoal identification results. (a) All recorded robot configurations from demonstration. (b) Distribution of subgoals extracted by the MCV method. (c) Identified subgoals.

A. Subgoal Identification Experiments

To corroborate the proposed subgoal identification approach, a skilled operator (SO) was invited to perform the trenching and truck loading tasks for three cycles as a task demonstration for the learning algorithm. In each task cycle, the operator moves the excavator arm to the top of the trench, followed by a scooping and lifting motion to pick up a bucket of sand; then, the operator moves the bucket to the top of the container and dumps the sand into it. The operator interfaces with each of the four actuators independently via two double-axis joysticks commonly found in commercial excavators [5]. One can observe from the illustration that completing such a task by coordinating the four actuators independently and simultaneously is rather challenging for human operators with no prior experience. The data from the SO demonstration are processed by the MCV and BNPC to identify the subgoals for the task.

The extracted subgoal distributions and identified subgoals from the demonstration are shown in Fig. 10, where

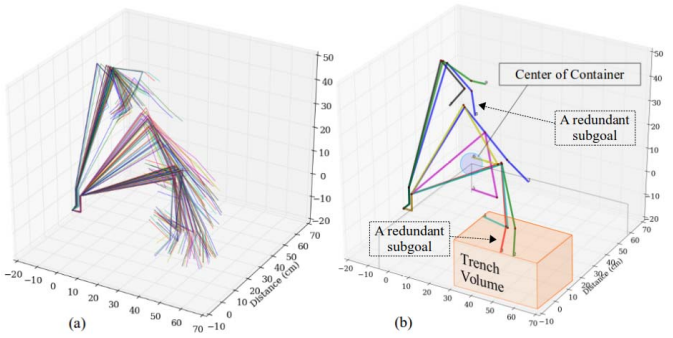

Fig. 11. Subgoal identification results via primitives-based methods. (a) Distribution of subgoals extracted via the primitives-based threshold method. (b) Identified subgoals.

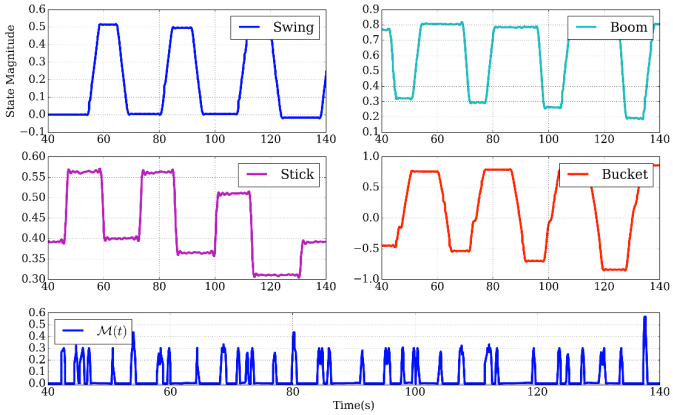

Fig. 12. Excavator states and MCV values.

each wire frame with four line segments represents a configuration of the excavator at a specific sample time. Fig. 10(a) provides the configuration trajectory swept by the excavator during the entire demonstration, Fig. 10(b) provides the subgoal distributions (each as a cluster of configurations) after the data set is processed via the MCV method, and the subgoal identification results in the form of subgoals (each realized by a robot configuration) are provided in Fig. 10(c). To illustrate the advantage of the proposed MCV method on noise-sensitivity issues, we used the threshold-based method given in [5] and [9] with the same demonstration data set, and the extracted subgoal distributions and identified subgoals are provided in Fig. 11(a) and (b), respectively. One can observe that the subgoal distributions, i.e., the clusters of robot configurations, are not as tight as those resulting from the MCV method. Such sparse distributions, in contrast to Fig. 10, led to two redundant subgoals, as shown in Fig. 11(b), which are not representative of any subtask-switching behavior of the operator. In particular, the redundant subgoal at the bottom part of Fig. 11(b) indicates that there should be a change in the operational pattern in the middle of curling the bucket and stick for picking up the dirt; the redundant subgoal on the top part of Fig. 11(b) indicates that there should be a change in the operational pattern in the middle of uncurling the bucket for dumping the dirt. These two additional subgoals are not meaningful because the motions corresponding to the two additional subgoals are relatively consistent and do not require the operator to switch to a different operational pattern to complete them during the process. Such discrepancies would also introduce statistical difficulties for BNPC inference in that: 1) it would take more iterations for the data to convert into clusters using iterative methods (such as the Gibbs sampler) and 2) furthermore, the clustering process is more sensitive to the concentration parameter. The evolution of the excavator states and MCV during the entire demonstration is provided in Fig. 12. One can observe that the operator efficiently controls the excavator's four degrees-of-freedom (bucket, stick, boom, and swing as used in standard excavator control terms) to perform the trenching task. Sudden changes in any of the excavator states typically reflect the fact that the operator has just reached a subgoal configuration and is trying to move the excavator to a different subgoal configuration by initiating a noticeably different command pattern. The goal of the MCV is to capture these changes from observing motions of multiple degrees-of-freedom of the system; the peaks in the MCV values reflect those time instances where the operator changes the command pattern indicating the completion of one subgoal and moving on to the next.

B. BSC With Subgoal Adjustment Experiments

To corroborate the effectiveness of the subgoal adjustment methods, a different series of human–machine interaction experiments was designed based on the same excavator trenching and truck loading tasks with ten novice operators (NOs) and one SO. The experiments were divided into the following stages: a preparation stage (PSt), a manual stage (MSt), a demonstration stage (DSt), a BSC stage (BSC-St), and a BSC with subgoal adjustment stage (BSC/SA-St).

In PSt, the basics of excavator operation and the trenching and truck loading tasks are explained to the NOs by providing them with an excavator operational instruction guide (which is always available to NOs) and have the NOs familiarize themselves with the excavator operations. After the notification from NO of getting ready for the task, we move to the MSt and ask the NO to start executing the task for three cycles without SC assistance. The process is timed, and the amount of sand loaded into the truck is weighed. We then transition into the DSt where we have the SO demonstrate the task, record the excavator states, and run the subgoal identification algorithm to learn the subgoals from the SO's demonstration. Note that this demonstration is separate from the one illustrated in Section VI-A; however, the procedure is the same. The identified subgoals are also slightly different since different demonstrations generate different data sets. With the learned subgoals, we start the BSC-St and activate the BSC algorithm with intent prediction to provide active assistance. The NO's are asked to execute the same task for nine cycles. The task execution times and weight data are collected and analyzed for each of the three cycles. Finally, in the BSC/SA-St, we repeat the process of the BSC-St but provide active assistance with the proposed subgoal adjustment BSC algorithm. The data are collected and analyzed in the same manner as in BSC-St.

Fig. 13 shows the operator performance comparison for each of the three task cycles. We can observe that, for the first three cycles when there is no significant shape change of the

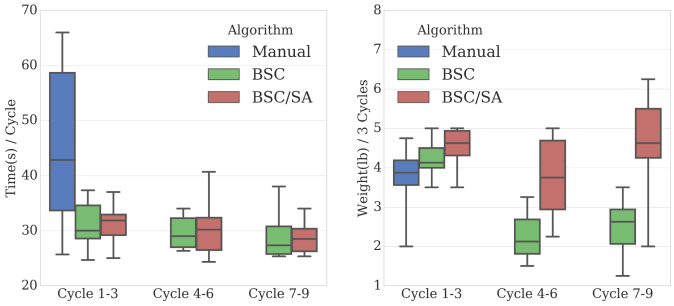

Fig. 13. Performance comparison for each three out of nine task cycles.

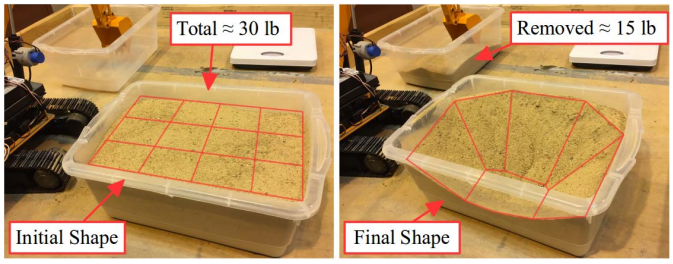

Fig. 14. Real shape change of the target object after nine task cycles.

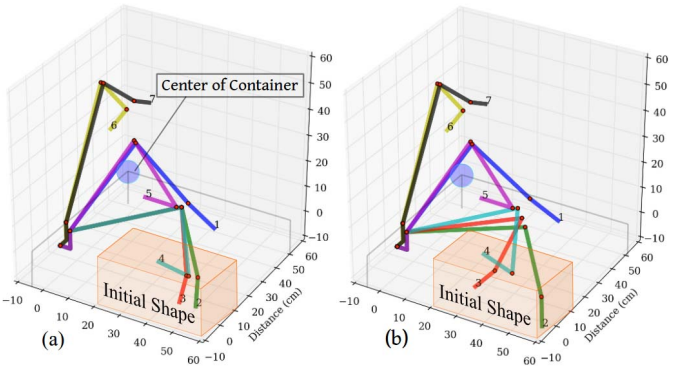

Fig. 15. Target initial shape and subgoals versus adjusted subgoals. (a) Initial subgoals. (b) Adjusted subgoals.

target object, the two BSC algorithms are able to improve the operator performance from the manual operation without a significant quality difference. The mean cycle time and its 95% confidence interval, improved from 45.53 ± 10.61 s (MSt) to 31.10 ± 3.06 s (BSC-St) and 31.13 ± 2.44 s (BSC/SA-St), and the sand moved for three cycles improved from 3.78 ± 0.55 lb (MSt) to 4.25 ± 0.36 lb (BSC-St) and 4.55 ± 0.34 lb (BSC/SA-St). For the last two sets of three cycle operation, we observed that the time improvement from the two BSC schemes, with respect to manual operation, stays very similar to the first three cycles. However, the performance in terms of the amount of sand moved for three cycles is different for the two BSC schemes. For the intent prediction BSC, the mean amount of sand moved and its 95% confidence interval go down from 4.25 ± 0.36 lb (cycles 1–3) to 2.28 ± 0.42 lb (cycles 4–6) and 2.53 ± 0.48 lb (cycles 7–9) because of target object shape change versus nominal subgoals. However, for the subgoal adjustment BSC, the mean weight performance and its 95% confidence interval change from 4.55 ± 0.34 lb (cycles 1–3) to 3.75 ± 0.72 lb (cycles 4–6) and 4.65 ± 0.85 lb (cycles 7–9). The subgoal adjustment BSC maintains its assistance quality almost throughout in the presence of gradual but cumulatively significant shape change of the target object, as shown in Fig. 14. In addition, we observed that, although the weight per three cycles data from the BSC/SA-St show sustainable performance improvement, the data spread is relatively wide. One possible reason could be the fact that, in our implementation, we fixed the value of the skill level parameter of SWAI since it is hard to evaluate one's skill level and operation style in our case. However, for industrial applications, that problem can be solved by designing a standardized test to rate the skill level of operators and relating that to the design parameter. Note that there is no perceptible performance difference in terms of cycle time between BSC with and without subgoal adjustment. This is expected as the gross motion for the excavator is similar for both cases, and the subgoal adjustments lead to only small changes in bucket position/orientation for each cycle such that dirt is picked up by the bucket. Of course, one could argue that the bucket is lighter without subgoal adjustment than with subgoal attachment, but the response time of the hydraulic excavator when the bucket is full is very similar to that when it is empty. The advantage that we gain from subgoal adjustment is that slight changes in each cycle leads to better coverage of the trench as it gets deeper and wider from the removal of dirt, thus resulting in more dirt being picked up by the bucket in each cycle.

To illustrate the effectiveness of the subgoal adjustment algorithm from other perspectives, we present in Fig. 15 a visualization of subgoal adjustments made by the SO in a few cycles. In the visualization, each subgoal is realized by converting its position in robot joint space to its configuration space of the excavator through forward kinematics. Fig. 15(a) shows the original identified subgoals and the initial target shape, and the initial target shape is shown again along with the adjusted subgoals in Fig. 15(b) to emphasize the adjustment. One can observe from these subfigures that the subgoals shown in green and red have changed significantly, while there is little change in other subgoals. The subgoals shown in green and red reflect the excavator configurations of inserting the bucket into the trench and curling the excavator forearm and bucket for picking up the dirt. In particular, the subgoal shown in green in Fig. 15(b) indicates that the bucket will be inserted much deeper than that of Fig. 15(a), and the subgoal in red indicates that the excavator will curl its forearm much more than in Fig. 15(a) from the insertion configuration to the picking-up-dirt configuration. These are necessary and important adjustments made to the nominal subgoals to deal with the changes in the nominal condition. Without these adjustments, the SC input will be generated based on the nominal subgoals and, thus, cannot help provide effective assistance in the sense that the nominal subgoals will barely reach and cover the trench.

VII. Conclusion

In this article, we have developed novel methods for subgoal identification and adjustment for SC tasks in this article. With a focus on task modeling from demonstration, we have

introduced the notion of MCV, which quantifies the operator command complexity with a unified metric. The results indicate a clear advantage of the MCV method over primitive-based methods. The evolution of the MCV metric and the robot states provide an intuitive indication of the relationship between robot states, operator command characteristics, and potential subgoals. As a mechanism to detect sudden changes in a data set containing high-dimensional sequential temporal data, the MCV metric can also be employed in other related areas. To facilitate subgoal adjustment based on human operator intent, we introduced new concepts of AEHR, SWAI, and HSTF. To aid input blending in practice, we proposed a DSC together with the concept of OEM, which considers the scaling problem in control blending and relates the operator's input and the gain of the automatic controller. A novel human–machine interaction setup with a scaled excavator system and a series of experiments were employed for testing. The methods are evaluated via experiments conducted on a scaled hydraulic excavator with trenching and truck loading tasks. Experimental results showed that MCV is capable of effectively identifying the subgoal distributions, thus providing meaningful subgoals for the task. In the presence of gradual but cumulatively significant shape change of the target object, the subgoal adjustment BSC outperforms the BSC without subgoal adjustment.

REFERENCES

[1] D. A. Abbink *et al.*, "A topology of shared control systems—Finding common ground in diversity," *IEEE Trans. Human-Mach. Syst.*, vol. 48, no. 5, pp. 509–525, Oct. 2018.

[2] P. Trautman, "Assistive planning in complex, dynamic environments: A probabilistic approach," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2015, pp. 3072–3078.

[3] M. K. O'Malley, A. Gupta, M. Gen, and Y. Li, "Shared control in haptic systems for performance enhancement and training," *J. Dyn. Syst., Meas., Control*, vol. 128, no. 1, pp. 75–85, Mar. 2006.

[4] M. Allain, S. Konduri, H. Maske, P. R. Pagilla, and G. Chowdhary, "Blended shared control of a hydraulic excavator," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 14928–14933, Jul. 2017.

[5] Z. Jin, P. R. Pagilla, H. Maske, and G. Chowdhary, "Task learning, intent prediction, and adaptive blended shared control with application to excavators," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 1, pp. 18–28, Jan. 2021.

[6] T. Carlson, R. Leeb, R. Chavarriaga, and J. D. R. Millan, "Online modulation of the level of assistance in shared control systems," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2012, pp. 3339–3344.

[7] Z. Jin and P. R. Pagilla, "Operator intent prediction with subgoal transition probability learning for shared control applications," in *Proc. IEEE Int. Conf. Human-Machine Syst. (ICHMS)*, Sep. 2020, pp. 1–6.

[8] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.

[9] H. Maske, K. Emily, G. Chowdhary, and C. Abramson, "Learning task-based instructional policy for excavator-like machines," in *Proc. IEEE Int. Conf. Robot. Autom.*, Jun. 2018, pp. 1962–1969.

[10] B. Michini, T. J. Walsh, A.-A. Agha-Mohammadi, and J. P. How, "Bayesian nonparametric reward learning from demonstration," *IEEE Trans. Robot.*, vol. 31, no. 2, pp. 369–386, Apr. 2015.

[11] Z. Jin, P. R. Pagilla, H. Maske, and G. Chowdhary, "Methods for blended shared control of hydraulic excavators with learning and prediction," in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2018, pp. 1973–1978.

[12] T. Carlson and Y. Demiris, "Collaborative control for a robotic wheelchair: Evaluation of performance, attention, and workload," *IEEE Trans. Syst., Man, Cybern., B (Cybern.)*, vol. 42, no. 3, pp. 876–888, Jun. 2012.

[13] J. Philips *et al.*, "Adaptive shared control of a brain-actuated simulated wheelchair," in *Proc. IEEE 10th Int. Conf. Rehabil. Robot.*, Jun. 2007, pp. 408–414.

[14] T. K. Stephens, N. J. Kong, R. L. Dockter, J. J. O'Neill, R. M. Sweet, and T. M. Kowalewski, "Blended shared control utilizing online identification," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 13, no. 6, pp. 769–776, Jun. 2018.

[15] J. C. F. de Winter, R. Happee, M. H. Martens, and N. A. Stanton, "Effects of adaptive cruise control and highly automated driving on workload and situation awareness: A review of the empirical evidence," *Transp. Res. F: Traffic Psychol. Behav.*, vol. 27, pp. 196–217, Nov. 2014.

[16] Z. Jin, P. R. Pagilla, H. Maske, and G. Chowdhary, "Blended shared control with subgoal adjustment," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2018, pp. 2724–2729.

[17] M. J. A. Zeestraten, I. Havoutis, and S. Calinon, "Programming by demonstration for shared control with an application in teleoperation," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1848–1855, Jul. 2018.

[18] A. Hansson and M. Servin, "Semi-autonomous shared control of large-scale manipulator arms," *Control Eng. Pract.*, vol. 18, no. 9, pp. 1069–1076, Sep. 2010.

[19] Z. Jin and P. R. Pagilla, "Collaborative operation of robotic manipulators with human intent prediction and shared control," in *Proc. IEEE Int. Conf. Human-Machine Syst. (ICHMS)*, Sep. 2020, pp. 1–6.

[20] H. Yu, S. Huang, G. Chen, and Y. Pan, "Human–robot interaction control of rehabilitation robots with series elastic actuators," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1089–1100, Oct. 2015.

[21] D. A. Abbink, M. Mulder, and E. R. Boer, "Haptic shared control: Smoothly shifting control authority?" *Cognition, Technol. Work*, vol. 14, no. 1, pp. 19–28, Mar. 2012.

[22] J. C. F. de Winter, M. Mulder, M. M. van Paassen, D. A. Abbink, and P. A. Wieringa, "A two-dimensional weighting function for a driver assistance system," *IEEE Trans. Syst., Man, Cybern., B (Cybern.)*, vol. 38, no. 1, pp. 189–195, Feb. 2008.

[23] V. Honing, T. L. Gibo, R. J. Kuiper, and D. A. Abbink, "Training with haptic shared control to learn a slow dynamic system," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2014, pp. 3126–3131.

[24] Z. Jin and P. R. Pagilla, "Blended shared control in collaborative robotics," in *Manufacturing in the Era of 4th Industrial Revolution*. Hackensack, NJ, USA: World Scientific, 2021, ch. 6, pp. 153–186.

[25] A. Enes and W. Book, "Blended shared control of Zermelo's navigation problem," in *Proc. Amer. Control Conf.*, Jun. 2010, pp. 4307–4312.

[26] D. Paindaveine, "A canonical definition of shape," *Statist. Probab. Lett.*, vol. 78, no. 14, pp. 2240–2247, 2008.

[27] L. Wasserman and L. Wasserman, *All of Statistics: A Concise Course in Statistical Inference* (Springer Texts in Statistics). New York, NY, USA: Springer, 2004.

[28] R. Miltenberger, *Behavior Modification: Principles and Procedures*. Boston, MA, USA: Cengage Learning, 2015.

Zongyao Jin received the B.S. degree in mechanical engineering from Harbin Engineering University, Harbin, China, in 2009, the M.S. degree in aerospace engineering from the China Academy of Space Technology, Beijing, China, in 2012, and the Ph.D. degree in mechanical engineering from Texas A&M University, College Station, TX, USA, in 2020.

He currently works at Flexiv Robotics Ltd., Santa Clara, CA, USA, as a Robotics Controls Engineer. His research interests include robotics, controls, industrial automation, and human–robot collaboration.

Dr. Jin was a recipient of the James J. Cain Outstanding Graduate Student Award and the Student Teaching Excellence Award from Texas A&M University.

Prabhakar R. Pagilla received the Ph.D. degree in mechanical engineering from the University of California at Berkeley, Berkeley, CA, USA, in 1996.

He is currently an Associate Department Head and the James J. Cain Professor II of Mechanical Engineering with Texas A&M University, College Station, TX, USA. His research interests include modeling and control-related problems in robotics/mechatronics, roll-to-roll manufacturing systems, connected and autonomous vehicles, and large-scale nonlinear dynamic systems.