2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Inverse Reinforcement Learning Framework for Transferring Task Sequencing Policies from Humans to Robots in Manufacturing Applications

Omey M. Manyar[1], Zachary McNulty[1], Stefanos Nikolaidis[2], and Satyandra K. Gupta[1].

***Abstract*— In this work, we present an inverse reinforcement learning approach for solving the problem of task sequencing for robots in complex manufacturing processes. Our proposed framework is adaptable to variations in process and can perform sequencing for entirely new parts. We prescribe an approach to capture feature interactions in a demonstration dataset based on a metric that computes feature interaction coverage. We then actively learn the expert's policy by keeping the expert in the loop. Our training and testing results reveal that our model can successfully learn the expert's policy. We demonstrate the performance of our method on a real-world manufacturing application where we transfer the policy for task sequencing to a manipulator. Our experiments show that the robot can perform these tasks to produce human-competitive performance. Code and video can be found at: https://sites.google.com/usc.edu/irlfortasksequencing**

I. Introduction

The industry is currently facing a severe shortage of skilled individuals who can perform complex processes [1]. There is a tremendous interest in deploying robots to automate these tasks for producing high-quality parts. Examples of such complex processes are surface finishing, composite sheet layup, spraying of surfaces, etc. (Refer Fig. 1). For these processes, experts break down a complex operation into a set of subtasks. In each subtask, experts perform intricate motions and complete the given process by sequentially executing these subtasks. This work aims to learn the human expert's policy for performing these processes from their demonstrations. This learned policy can serve as a skill-transfer medium between humans and robots. More importantly, our learned policy can enable the robot to execute the process on a brand new part without requiring a new human demonstration.

We approach this problem by interviewing several domain experts. In our interactions with these experts, we discovered that the process outcome is heavily influenced by the sequence in which the subtasks are performed. This work focuses on modeling the expert's policy for sequencing the subtasks to achieve desirable outcomes in the process.

One could think of several approaches to capture the expert's policy to sequence these tasks, e.g., using physics-based simulation for planning defect-free sequences. However, simulating defects in a process such as composite prepreg layup that involves manipulating a sheet can be complex [2]–[4]. Additionally, heuristic-based approaches to capture this policy can be cumbersome and may be hard to generalize. Inverse Reinforcement Learning (IRL) has proved quite effective in solving several task sequencing problems in robotics [5]–[7]. Hence we adopt an IRL-based approach where we try to learn the expert's policy for sequencing these tasks with a robot.

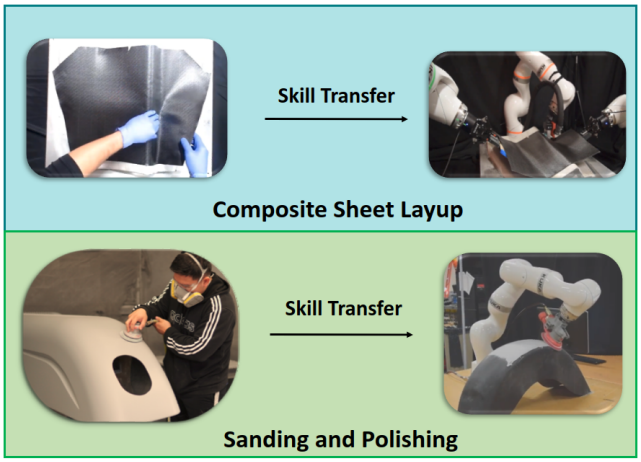

Fig. 1: Example processes that require human-to-robot skill transfer.

Our entire framework for learning the expert's policy is summarized in Fig. 2. We have a demonstration stage where we collect the expert's task sequencing and motion data on different tools. Tools here are defined as the entity on which the process is executed to manufacture the desired part. The tools used in these processes are usually divided into local regions (Refer to Demonstration Stage in Fig. 2). A task is defined as the action of processing an individual region.

The experts can verbalize their process knowledge and give us explicit information that helps us select features for our learning problem. Moreover, experts suspected that for these processes, several features could interact to influence their policy. The processes described in this work are high-mix low volume, so capturing these feature interactions becomes essential. Thus, we have an aspect that computes whether enough interactions are captured in the demonstration dataset.

After obtaining high-quality demonstration data, our next step is to learn the policy, which refers to the expert's preferences for executing the subtasks. While multiple sequences may lead to desirable performance on a given tool, experts may have specific preferences due to external factors such as the design of the work cell, ease of operation, tool and robot placement, and other considerations. To address this issue, we distinguish between two types of preferences:

[1]Realization of Robotic Systems Lab, University of Southern California, Los Angeles, CA, USA. [2]Department of Computer Science, University of Southern California, Los Angeles, CA, USA, Address all correspondence to guptask@usc.edu

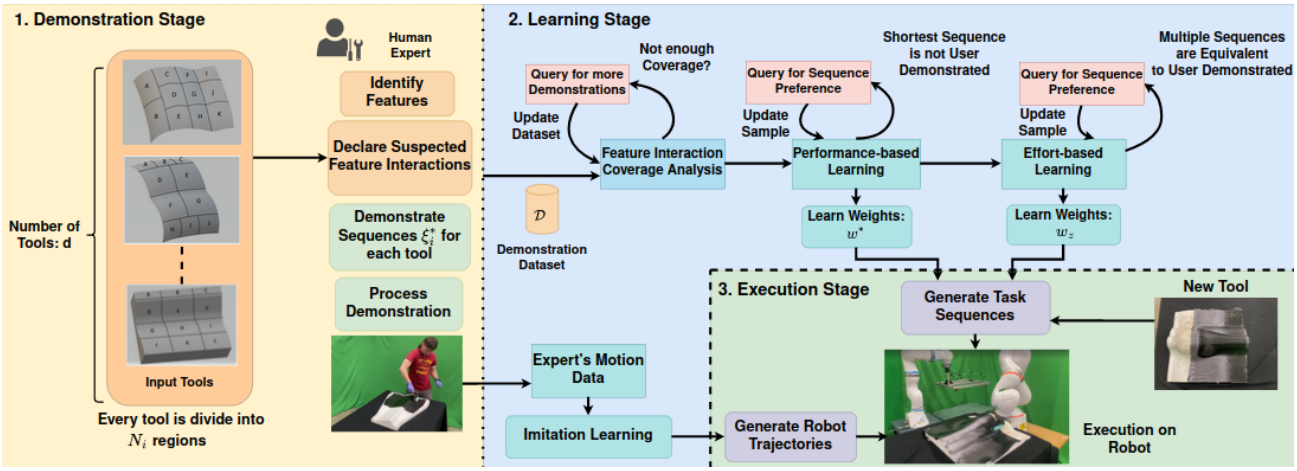

Fig. 2: Overview of the proposed framework for learning task sequencing policy.

"performance-based" and "effort-based," and learn them independently. This approach enables us to generalize the policy across various tools and operational settings more effectively. In this manner, we propose an IRL framework to learn the expert's policy for task sequencing and transfer it to a robot. The key contributions of our work are (1) Feature Interaction coverage determination for demonstrations, (2) an active IRL methodology to learn expert's independent preferences, and (3) robot demonstrations on an actual industrial problem.

II. Related Work

IRL Overview: The field of IRL has been mainly categorized under the umbrella of imitation learning, with the objective of learning an expert's policy. Earlier work in IRL focused on imitating an expert's policy on already demonstrated data or simple tasks [8]. There are four main IRL approaches: Max-margin IRL, Max-entropy IRL, Bayesian IRL, and Regression/Classification IRL [9]. Max-entropy-based methods [10] propose a solution to select weights in the presence of incomplete information. In contrast, Bayesian-IRL methods [11] compute a probability distribution over reward functions based on informative priors. Most of the work in IRL assumes a linear reward function, except for [12], which focuses on sub-optimal stochastic demonstrations. In this work, our problem requires us to strictly comply with the expert in the absence of any noise. Besides finding a reward function for the expert, we also need to penalize sequences that result in poor process performance. Therefore, max-entropy and Bayesian IRL-based methods are not suitable. Max-margin methods proposed by [13], [14] introduce the concept of scaling sequences based on their proximity to the expert's demonstration. Therefore, max-margin-based methods provide the appropriate learning scheme for our problem. However, previous work on max-margin-based methods did not address scenarios with varying levels of preferences, feature interactions, and limited demonstrations. Recent work proposed in [15]–[17] introduces the active learning element to learn reward functions from preferences. However, these methods do not take into account interacting features.

IRL in Manufacturing: IRL for sequencing has been studied for assembly tasks in [18]–[20] but the focus is on assisting the expert rather than transferring policy to the robot. Several works also focus on learning rewards for insertion tasks [21], [22]. In [23], authors mention about using IRL for sequencing for machining tasks but do not account for transferring policies to a robot. Prior work in learning from demonstration has explored the problem of solving surface finishing operations [24]–[27] but they do not address the region sequencing problem.

III. Problem Formulation

We formulate the problem described in Section I as a task sequencing problem, where the robot has to perform a set of subtasks to complete the process.

Assumptions: Our assumptions for the task sequencing problem are as follows:

- Demonstrations are optimal and have no noise as they are performed by experts.
- Features and Feature Interaction information can be captured from expert's description of the process.
- Only pairwise feature interactions are present.

State Preliminaries: Section I describes processes that feature a tool, as shown in Fig. 2. The tool is divided into N_i regions, $\forall i \in 1,..,d$, where d is the number of demonstration tools. Regions are indexed alphabetically based on their appearance row-wise from left to right. A task for the agent is defined as processing a region on the tool, so the agent's objective is to compute a desirable sequence for performing these N_i tasks.. We denote the human expert by $\mathcal{H}$ whose performance and effort-based preference we are trying to model. The autonomous agent that learns from $\mathcal{H}$ will be represented by $\mathcal{A}$. The state of $\mathcal{A}$ gets denoted as $s \in \mathcal{S}_i, \mathcal{S}_i \mapsto \mathbb{R}^{N_i}, \forall i \in \{1,..,d\}$, and we represent s as a set of tasks that $\mathcal{A}$ completes. The action that $\mathcal{A}$ takes at a state s is denoted as $a \in A$, where A is a set of available actions at state s. An action a signifies the task that $\mathcal{A}$ chooses to perform. In our case, A gets defined by the set of tasks that are yet to be executed by the agent. The

state transitions $s \rightarrow s'$ due to action a are assumed to be completely deterministic.

Transition Cost: In this work, we will refer to reward as a negation of cost. Hence, our formulation will refer to the IRL problem of maximizing reward as minimizing cost. We define the policy of the agent as to follow a minimum cost sequence for performing the N_i tasks. In a standard inverse reinforcement learning problem, the cost for transitioning to a state gets defined as a linear function of weights w and an array of feature values ϕ that is dependent only on the current state of the agent [28]. The nature of the sequencing problem that we study in this work is such that the feature array is a function of the state s and action a. Additionally, as per $\mathcal{H}$, there might be certain feature pairs in the feature array that might interact with each other.

Therefore, we define the cost of transition for $\mathcal{A}$ by Eq. 1, where $\phi(s,a) = \{\phi_1, \phi_2, ..., \phi_n\}$ is an array of feature values that are dependent on state transitions. The set of interacting features for a subset of feature values from $\phi(s,a)$ is denoted by $\phi_{int}(s,a) = \{(\phi_i \cdot \phi_j)_1, .., (\phi_k \cdot \phi_l)_m\}, \forall i,j,k,l \leq n,\ i \neq j, k \neq l$. The total number of feature interactions suspected by the expert $\mathcal{H}$ is m. In our formulation, non-linearity is introduced by $\phi_{int}(s,a)$. We will denote our unified weight array as $w = \{w^\frown w_{int}\}$, that is a concatenated array of the linear feature weights w and the interacting feature weights w_{int}.

$$c(s,a) = w^T\phi(s,a) + w_{int}^T\phi_{int}(s,a) \tag{1}$$
$$w, \phi \in \mathbb{R}^n, \forall \phi : s \in \mathcal{S}, a \in A$$
$$w_{int},\ \phi_{int}(s,a) \mapsto \mathbb{R}^m$$

Learning Performance-based Preferences: The human expert $\mathcal{H}$ demonstrates a desired sequence ξ^* on a sample tool. The agent $\mathcal{A}$ has access to d demonstrations, each on different tools. We define the demonstration dataset as $\mathcal{D} = \{\delta_1, \delta_2, ..., \delta_d\}$, where $\delta_i = \{\xi_i^*, \mathcal{F}_i, N_i\}$, $\mathcal{F}_i$ is the feature information for i^{th} demonstration tool from which ϕ is computed. Furthermore, the expert $\mathcal{H}$ also gives information about the interacting feature set ϕ_{int} for the demonstration dataset $\mathcal{D}$. Now as described earlier, the agent has to solve the following problem:

$$C(\xi_i^*) \leq C(\xi_i^j), \forall \xi_i^* \in \mathcal{D}, \forall \xi_i^j \in \mathcal{P}_i \mapsto \mathbb{R}^p \tag{2}$$

where,
$$C(\xi_i) = w^T\Phi(\xi_i)$$
$$\Phi(\xi_i) : \{\sum_{k=1}^{N_i}(\phi(s_k,a_k)\ ^\frown \sum_{k=1}^{N_i}\phi_{int}(s_k,a_k)\}$$

In Eq. 2, $C(\xi_i)$ is the total cost incurred by the agent for following a sequence ξ_i. We define $\mathcal{P}_i$ as the set of all possible sequences for the i^{th} tool. $\Phi(\xi_i)$ is an array of sum of individual feature values and interactions for the state transitions in sequence ξ_i. Thus, we find a weight array w^* by solving Eq. 2 such that the expert demonstrated sequence ξ_i^* is the lowest cost sequence for all the corresponding tools in $\mathcal{D}$.

Learning Effort-based Preferences: The feature values used for the problems discussed in Section I are geometric features of the tool. Thus, for a given demonstration in $\mathcal{D}$, there might be several sequences with equivalent feature arrays due to the symmetry of the tool. These equivalent sequences (ξ_i^e) will have identical cost to the user demonstrated sequence (ξ_i^*). We formulate this feature equivalence property by Eq. 3.

$$\Phi(\xi_i^*) \equiv \Phi(\xi_i^e); \xi_i^e \in \Omega_i, \forall i \in \{1, ..., d\} \tag{3}$$

where, Ω_i is the set of sequences for i^{th} demo that exhibit the equivalence property in Eq. 3. However, as discussed in Section I, the expert $\mathcal{H}$ might prefer specific sequences in Ω_i due to their effort-based preferences. $\mathcal{H}$ might choose sequences that start at a specific task and end at a specific task to improve their ease of operation. In this case, $\mathcal{H}$ is queried to compare their preference for the equivalent sequences in relation to corresponding ξ_i^*'s in $\mathcal{D}$. The objective is to learn a preference penalty function η such that the preferred sequences ξ_i^{pref} have the lowest penalty. Hence we can formulate the effort-based preference learning problem as follows:

$$\eta(\xi_i^{pref}) < \eta(\xi_i^j),\ \forall i \in \{1, ..., d\}, \forall \xi_i^j \in \psi_i, \forall \xi_i^{pref} \in \psi_i^* \tag{4}$$

where, $\eta(\xi) = w_z^T\ z(\xi)$; and $z(\xi)$ are effort-based preference features for an arbitrary sequence ξ. We denote, $\psi_i^* \subseteq \Omega_i$ as the set of preferred sequences, and $\psi_i \subseteq \Omega_i$ as the set of unpreferred sequences $\ s.t.\ \psi_i^* \cap \psi_i = \emptyset$. It is important to note that adding $z(\xi)$ in learning the weights w in Eq. 2 would not be appropriate as $z(\xi)$ will change based on external factors mentioned in Section I. In situations when the expert has no preference between the demonstrated sequence and any other equivalent sequence ξ^e, we do not need to learn the preference penalty $\eta(\cdot)$. We can set the value of $\eta(\xi) = 0$, for any arbitrary sequence ξ.

Feature Interaction Coverage in Demonstrations: In order to learn the expert's preferences, the demonstrations need to be informative with respect to feature interactions. Assuming access to enough demonstrations that can capture all levels of feature values and feature interactions for the transitions in ξ^* is impractical. We propose feature interaction coverage as a metric that assesses whether the dataset $\mathcal{D}$ enables learning of these interactions. This metric can then be used to query the expert for more informative demonstrations if needed. We represent this metric as $\rho(\cdot)$ and define it as follows:

$$\rho(\mathcal{D}) \propto \kappa(\{\phi, \phi_{int}\}) \tag{5}$$

where, the function $\kappa(\cdot)$ computes the overall extent of coverage of ϕ and ϕ_{int}. Thus, using $\rho(\cdot)$, we should be able to query $\mathcal{H}$ to ask for specific demonstrations that improve feature coverage.

IV. Method

In the previous section, the formulation that we proposed engenders three main problems: (1) Learning $\mathcal{H}$'s performance-based preference, (2) Learning effort-based preferences, and (3) Feature interaction coverage. The entire framework is depicted in Fig.2. We build upon the structured max-margin approach outlined in [13] to solve the proposed problem. Section. II gives the reasoning for selecting margin-based

IRL methods for our approach. Before we commence learning performance and effort-based preferences, we evaluate the extent of feature coverage in the demonstration data. Once we have enough coverage of feature values, we proceed with learning. Subsequently, we define a graph-based state space representation for each demo tool in $\mathcal{D}$. Then we introduce our iterative max-margin approach with an active learning element for diversely sampling the training data and a cost function designed to solve the problem in Eq. 2.

A. Estimating Feature Interaction Coverage

To solve the feature interaction coverage problem we employ a 2-factor factorial design of experiments technique to capture feature values and feature interactions on three levels: high, medium, and low. For each interaction level, we count the instances such that the remaining feature values are uniformly distributed. We use the standard Chi-squared test for determining whether the features are uniformly distributed with an α value of 0.05 [29]. Once we ensure uniform distribution, we compute a confidence metric of $\mathcal{D}$ for capturing all expected interactions according to Eq. 6.

$$\rho(\mathcal{D}) = \frac{\text{no of interactions captured}}{\text{total number of interactions possible}} \tag{6}$$

Based on the interactions that are weakly captured, we query the expert $\mathcal{H}$ to provide demos with a specific feature interaction value. This helps in improving the $\rho(\cdot)$ value of the dataset as shown in Section VI.

B. Graph-based State Sequence Representation

After obtaining $\mathcal{D}$ with sufficient feature value coverage, we represent the state space ($\mathcal{S}_i$) for every demonstration in $\mathcal{D}$ in the form of a separate graph $\mathcal{G}_i, \forall i \in \{1, ..., d\}$. Where each node/vertex *v* in the graph is a state s, and each edge *e* is the cost incurred to transition between states $s \rightarrow s'$. Therefore, each demonstration $\delta_i \in \mathcal{D}$ has a corresponding $\mathcal{G}_i$.

In $\mathcal{G}_i$, we use a one-hot vector of size N_i to encode each node, where the completed tasks are represented as 1's and incomplete tasks are represented as 0's. We define s_{start} as a state node for which none of the tasks are completed by $\mathcal{A}$ and s_{end} is the state node when $\mathcal{A}$ has completed all the tasks for a given tool. Naturally, s_{start} becomes a vector of 0's of size N_i and s_{end} becomes a vector of 1's of size N_i. Thus, an arbitrary sequence ξ_i for $\mathcal{G}_i$ becomes the path traversed from s_{start} to s_{end}. As discussed, our objective becomes to learn w^*, such that ξ_i^* will be the shortest cost path from s_{start} to s_{end} for all the corresponding $\mathcal{G}_i$'s. For a demonstration with N tasks, there are a total of $N!$ possible sequences, which necessitates a specialized approach for learning w^*.

C. Loss function for performance-based preferences

The Quadratic formulation, as described in [13], is the most commonly used method to formulate the max-margin problem. Such a formulation with constraints becomes inefficient to optimize when we have a large number of sequences with cost values significantly higher than the lowest cost sequence [30]. After computing the cost of a random sample of sequences, we found that 55% of sequences still had a cost value 70 times higher than the lowest cost sequence. Therefore, we resort to an online cost function where the constraints are absorbed into the objective function itself. Our unconstrained loss function is as follows:

$$\mathcal{L}(w) = \frac{1}{d}\sum_{i=1}^{d}\Big[\ \frac{\alpha_i}{p_i}\sum_{j=1}^{p_i}(w^T\Phi_i^* - w^T\Phi_i^j)\ + \beta_i(w^T\Phi_i^* - \min_{\Phi_i^j}(w^T\Phi_i^j))\ +\ \gamma_i\ (w^T\Phi_i^* - w^T\Phi_i^s)\ \Big] + \lambda||w||^2 \tag{7}$$

In Eq. 7, $\alpha_i, \beta_i, \gamma_i$, are hyperparameters and λ is regularization term. Φ_i^* represents the feature array for the user demonstrated sequence ξ_i^* in $\mathcal{D}$. To train the model, we generate an initial sample of sequences $x_i \in \mathcal{X} \mapsto \mathbb{R}^{p_i}$ of size p_i for i^{th} demo, as per Section IV-D. For the initial sample x_i, $\min_{\Phi_i^j}(w^T\Phi_i^j)\forall j \in \{1, .., p_i\}$ returns the cost value for the lowest cost sequence in x_i. The feature array for the overall minimum cost sequence ξ_i^s of the corresponding $\mathcal{G}_i$ is denoted by Φ_i^s. We minimize this loss using a generalized version of gradient descent based on subgradients [31]. Section VI gives an intuition of every term in Eq. 7.

D. Learning performance-based preference

The state space of the proposed problem experiences rapid complexity growth due to the problem's combinatorial nature. For a given $\mathcal{G}_i$ there are overall $N_i!$ possible sequences. Hence, solving this problem with one-shot optimization can be inefficient. To address this challenge, we adopt a solution that begins with a nominal sample size p_i drawn from the entire population of sequences for the i^{th} demonstration. However, random sampling may result in a poor representative sample, particularly when many sequences have equivalent feature values (see Section III). To generate a diverse initial sample, we use a similarity metric based on cosine similarity between feature values of a given sample ξ and expert-demonstrated sequence ξ^*. We then iteratively update this sample set with sequences that have costs below a certain threshold compared to ξ^* as described in Algorithm 1.

Algorithm 1 takes the demonstration dataset $\mathcal{D}$ as input. Using $\mathcal{F}_i$ we compute the feature array $\phi(s, a)$ and initialize the corresponding graphs $\mathcal{G}_i$ for each demonstrations in $\mathcal{D}$ with *w*. We generate an initial sample data $\mathcal{X}$ by performing cosine similarity-based diversity sampling. This sample helps max-margin to appropriately scale the cost of the sequences that are truly bad compared to the good ones. Since our initial sample size is a small representation of the entire sequence population, we perform learning in an iterative manner. After every iteration, we compute the first $\mathcal{K}_i$ shortest cost sequences and query $\mathcal{H}$ to compare if they are similar to ξ_i^*. We then update the sample space $\mathcal{X}$ with non-similar sequences for all $\delta_i \in \mathcal{D}$. Using such an iterative update and starting with a diverse sample helped us converge faster [32].

E. Learning Effort-based Preference

Algorithm. 2 depicts how we learn a penalty function based on Eq. 4. This penalty function is trained based on positional feature values of the starting and ending regions in

Algorithm 1: Performance-based Preference Learner

Input: $\mathcal{D}$
Initialize:
w: Random Initialization
$\phi(s,a)$: Compute $\phi(s,a)$ $\forall \delta_i \in \mathcal{D}$
$\mathcal{G}_i$: Initialize $\mathcal{G}_i = (v, e)$ with w; $\forall \delta_i \in \mathcal{D}$
$\alpha_i, \beta_i, \gamma_i$: Hyperparameters
p_i: Initial sample size for i^{th} demo in $\mathcal{D}$
$\mathcal{K}_i$: Dataset update parameter for i^{th} demo in $\mathcal{D}$
ζ: Learning Rate
T: Total Number of Iterations
$t = 1$: Current Iteration Count
GenerateSample:
$\mathcal{X} : x_1, x_2, ..., x_d; x_i \mapsto \mathbb{R}^{p_i}, \forall i \in \{1, .., d\}$

while *(not converged)* **do**
  while $t < T$ **do**
    Compute($C(\xi_i^*), \forall i \in \{1, .., d\}$)
    Compute($C(\xi_i^j), \forall i \in \{1, .., d\}, \forall j \in \{1, .., p_i\}$)
    Compute ($C(\xi_i^s), \forall i \in \{1, .., d\}$)
    Compute Loss: $\mathcal{L}(w)$ as per Eq. 7
    Compute Subgradient g of $\mathcal{L}(w)$
    $w \leftarrow w - \zeta g$
    reinitialize $\mathcal{G}_i$, $\forall i \in \{1, .., d\}$ with updated w
    $t \leftarrow t + 1$
  if *(*$C(\xi_i^*) == C(\xi_i^s), \forall i \in \{1, .., d\}$*)* **then**
    converged = True
  else
    Compute $\mathcal{K}_i$ shortest cost sequences in $\mathcal{G}_i$, $\forall i \in \{1, .., d\}$
    Query $\mathcal{H}$ to check for similarity between the $\mathcal{K}_i$ sequences and demonstrated sequence
    update $\mathcal{X}$ with non-similar shortest cost sequences
    t = 1
return w

Algorithm 2: Effort-based Preference Learner

Input: $\mathcal{D}$
$z(\xi_i)$: Positional features for a sequence $\forall \delta_i \in \mathcal{D}$
$\mathcal{Q} : \{\Omega_1, \Omega_2, ..., \Omega_d\}$.
Initialize:
w_z: Random Initialization
ψ^* = []: Set of preferred sequences,
ψ = []: Set of unpreferred sequences,

for Ω_i *in* $(\mathcal{Q})$ **do**
  ψ^*.append($z(\xi_i^*)$)
  for ξ_j *in* (Ω_i) **do**
    preference = evaluate$\mathcal{H}$preference(ξ_i^*, ξ_j)
    if *preference* **then**
      ψ^*.append($z(\xi_j)$)
    else
      ψ.append($z(\xi_j)$)
if *(len(ψ) == 0)* **then**
  $w_z = 0$
else
  Learn w_z using max-margin approach with $\mathcal{L}(w_z)$ as the loss function (Refer Eq. 8)

a sequence ξ. A simple quadratic loss function gave us the desired results as follows:

$$\mathcal{L}(w_z) = \frac{\lambda_z}{2}||w_z||^2 \tag{8}$$
$$s.t. \forall i, j \ \max_{\xi_i \in \psi^*} \eta(\xi_i) + \epsilon < \min_{\xi_j \in \psi} \eta(\xi_j)$$

where, ψ^* is a set of sequences that are preferred by $\mathcal{H}$ and ψ is a set of sequences that are not preferred by $\mathcal{H}$. ϵ is a slack variable. When the expert equally prefers all equivalent sequences i.e. $\psi = \emptyset$, we set $\eta = 0$ for all sequences.

V. Data Collection

We evaluate our method on the composite prepreg layup process. Previous work [33] showcases how this process is performed in a sequential manner, where the task of the agent is to conform the sheet region-by-region on top of the tool. Our dataset comprises of two categories: 1. Real Dataset ($\mathcal{D}_{real}$) and 2. Synthetic Dataset ($\mathcal{D}_{syn}$). $\mathcal{D}_{syn}$ is a simplified version of $\mathcal{D}_{real}$. We use $\mathcal{D}_{syn}$ for model evaluation and ablation studies, whereas $\mathcal{D}_{real}$ is used for training our model and performing physical robot experiments. $\mathcal{D}_{real}$ consists of industrial tools used in the layup process (Refer Fig. 3). Overall we have 10 tools each in $\mathcal{D}_{real}$ and $\mathcal{D}_{syn}$. We use a clustering algorithm based on feature attributes such as curvature, relative height, and aspect ratio of the tool [34] to transform the given tool's CAD into local regions. We then record the expert's desired sequence on all the tools in $\mathcal{D}_{real}$. In this case, we use a total of 10 feature values that are selected after consulting with the process expert for learning performance-based preferences[1]. The process expert also provides information on pairwise feature interactions. In our case the expert suspected interaction between the relative height and curvature of a region, as well as region orientation and curvature.

We validate the effectiveness of our model in generating high-quality parts by conducting physical robot experiments on two tools selected from $\mathcal{D}_{real}$, as illustrated in Fig. 3. During the execution of the layup task on these tools, we record the expert's motion and force data. To capture the force data, we design a custom handheld tool with an embedded force sensor. The motion of this tool is tracked via an OptiTrack motion capture system with an accuracy of $\pm 0.1mm$.

VI. Results

We perform ablation studies for our loss function on $\mathcal{D}_{syn}$ by setting a known weight value w_{syn}. The weight array w_{syn} represents a hypothetical human preference. We then evaluate the shortest cost sequence and try to recover w_{syn}. We use a 6:4 training to testing split for $\mathcal{D}_{syn}$ and $\mathcal{D}_{real}$. For $\mathcal{D}_{real}$, we train our model and perform robot demonstration on a testing tool.

[1]note: Due to space constraints we will not described every feature in detail. Refer to our website: https://sites.google.com/usc.edu/irlfortasksequencing

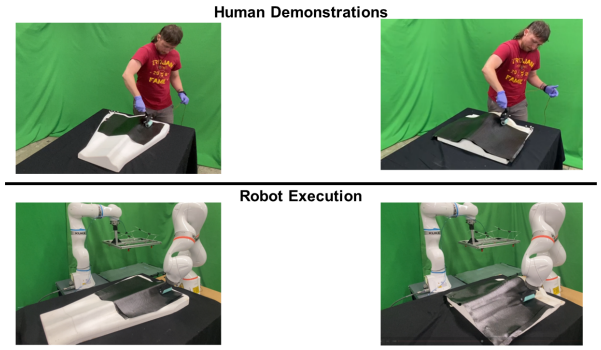

Fig. 3: Two tools are selected from $\mathcal{D}_{real}$for collecting human motion data and then the robot performs the same process. The tool on the left is used for training and the tool on the right is used for testing.

Performance-based Preference Learning Results ($\mathcal{D}_{syn}$): Fig. 4 illustrates how every term in our loss function in Eq. 7 has an impact on performance. The average term with α hyperparameter helps in penalizing bad sequences by ensuring a good spread of cost values for sample sequences. The min term for the training sample with β hyperparameter helps in effectively penalizing sequences that are close to demonstrated sequence when we update the initial sample. Lastly, the shortest-path sequence term with γ hyperparameter helps us achieve convergence in a smaller number of updates.

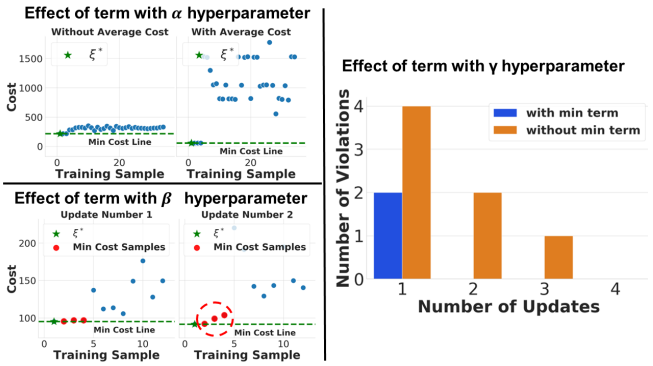

Fig. 4: In the absence of the α term, other sequences were not appropriately scaled despite ξ^* being the lowest cost. The β term scaled the minimum cost data points properly after iterative updates, as seen from the shift in cost of red dots in Update Number 1 and 2. The γ term required more update steps for convergence. The y-axis violations represent the number of training demonstrations where the desired sequence was not the lowest cost.

Effort-based Preference Learning Results ($\mathcal{D}_{syn}$): Fig. 5 shows that initially, the cost for all equivalent sequences for an example tool is the same for a learned weight array w^*. Eventually, the learned η levied an appropriate penalty for the sequences that were less preferred by the expert. In our implementation, we use the normalized coordinates of tool's individual region centroids as the positional features for learning. In the example in Fig. 5, the expert had a starting preference at a region to the right of the tool.

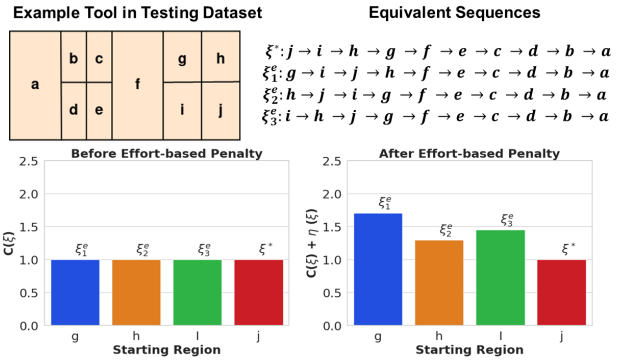

Fig. 5: Note that $C(\xi)$ is normalized. Also we scale the costs such that η for ξ^* is 0.

Feature Interaction Coverage Results ($\mathcal{D}_{syn}$): In order to evaluate feature interaction coverage, we perform a study where we select different sets of tools from $\mathcal{D}_{syn}$ and assess their influence on ρ.

Set of Tools	{1,2,3,4,5,6,7,8,9,10}	{1,6,7,8,9,10}	{2,3,4,6,7}	{1,2,3,4,5}
Value of ρ	1.0	0.9	0.5	0.4

TABLE I: Set of Tools means a subset of the 10 tools from $\mathcal{D}_{syn}$. First column is for all the tools in $\mathcal{D}_{syn}$. We can see as we vary the dataset, ρ value changes indicating varying feature interaction coverage. For more info on tools: website

Robot trials ($\mathcal{D}_{real}$): We train the model on 6 tools from $\mathcal{D}_{real}$ and test them on 4 tools. We used one of the tools on which we recorded expert's motion and force data for training and the other one for testing. We trained on a simpler tool and evaluated our model on a complex tool. We executed the robot with motions learned from the human motion data for the sequence generated for the testing tool. To test the effect on part quality, we scanned the human laid-up part and robot laid-up part and computed the error between the two. The error was 0.7 mm ($\pm$ 0.026mm) which is acceptable for the mentioned process [35]. The demonstration data had a ρ value of 1.0, depicting that feature interactions are captured at all levels. Our model evaluated well on both the training and testing datasets, with the demonstrated sequence consistently yielding the lowest cost for all tools in $\mathcal{D}_{real}$.

VII. Conclusion

We showed that our framework can learn the expert's policy for task sequencing in a complex manufacturing process. Our results demonstrate that our cost function builds robustness in learning the model weights, and that the feature interaction coverage metric can aid in evaluating the quality of demonstration data. By learning weights that prioritize the expert's sequence, our method was able to achieve the lowest cost for all demonstration tools in both real and synthetic data. Furthermore, we have shown the benefits of decoupling performance and effort-based preferences.

Acknowledgement: This work is supported in part by National Science Foundation Grant #1925084. The opinions expressed are those of the authors and do not necessarily reflect the opinions of the sponsors.

REFERENCES

[1] P. Wellner, V. Reyes, H. Ashton, and C. Moutray. (2021) Creating pathways for tomorrow's workforce today. beyond reskilling in manufacturing. [Online]. Available: https://www2.deloitte.com/us/en/insights/industry/manufacturing/manufacturing-industry-diversity.html

[2] O. M. Manyar, J. Cheng, R. Levine, V. Krishnan, J. Barbic, and S. K. Gupta, "A synthetic image assisted deep learning framework for detecting defects during composite sheet layup," in *42nd Computers and Information in Engineering Conference (CIE)*, ser. International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 08 2022.

[3] R. K. Malhan, R. Jomy Joseph, A. V. Shembekar, A. M. Kabir, P. M. Bhatt, and S. K. Gupta, "Online grasp plan refinement for reducing defects during robotic layup of composite prepreg sheets," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 11 500–11 507.

[4] Y.-W. Chen, R. J. Joseph, A. Kanyuck, S. Khan, R. K. Malhan, O. M. Manyar, Z. McNulty, B. Wang, J. Barbič, and S. K. Gupta, "A Digital Twin for Automated Layup of Prepreg Composite Sheets," *Journal of Manufacturing Science and Engineering*, vol. 144, no. 4, 09 2021, 041010. [Online]. Available: https://doi.org/10.1115/1.4052132

[5] S. Krishnan, A. Garg, R. Liaw, B. Thananjeyan, L. Miller, F. T. Pokorny, and K. Goldberg, "Swirl: A sequential windowed inverse reinforcement learning algorithm for robot tasks with delayed rewards," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 126–145, 2019. [Online]. Available: https://doi.org/10.1177/0278364918784350

[6] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889008001772

[7] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual review of control, robotics, and autonomous systems*, vol. 3, pp. 297–330, 2020.

[8] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *ICML*, 1997.

[9] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," *Artificial Intelligence*, vol. 297, p. 103500, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370221000515

[10] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, ser. AAAI'08. AAAI Press, 2008, p. 1433–1438.

[11] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, ser. IJCAI'07. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, p. 2586–2591.

[12] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with gaussian processes," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011. [Online]. Available: https://proceedings.neurips.cc/paper/2011/file/c51ce410c124a10e0db5e4b97fc2af39-Paper.pdf

[13] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 729–736. [Online]. Available: https://doi.org/10.1145/1143844.1143936

[14] J. Bagnell, J. Chestnutt, D. Bradley, and N. Ratliff, "Boosting structured prediction for imitation learning," in *Advances in Neural Information Processing Systems*, B. Schölkopf, J. Platt, and T. Hoffman, Eds., vol. 19. MIT Press, 2006. [Online]. Available: https://proceedings.neurips.cc/paper/2006/file/fdbd31f2027f20378b1a80125fc862db-Paper.pdf

[15] D. Sadigh, A. D. Dragan, S. S. Sastry, and S. A. Seshia, "Active preference-based learning of reward functions," in *Robotics: Science and Systems*, 2017.

[16] C. Basu, E. Biyik, Z. He, M. Singhal, and D. Sadigh, "Active learning of reward dynamics from hierarchical queries," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019.

[17] E. Biyik, D. P. Losey, M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh, "Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences," *The International Journal of Robotics Research (IJRR)*, 2021.

[18] S. Nikolaidis, R. Ramakrishnan, K. Gu, and J. Shah, "Efficient model learning from joint-action demonstrations for human-robot collaborative tasks," in *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2015.

[19] H. Nemlekar, J. Modi, S. K. Gupta, and S. Nikolaidis, "Two-stage clustering of human preferences for action prediction in assembly tasks," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 3487–3494.

[20] H. Nemlekar, R. Guan, G. Luo, S. K. Gupta, and S. Nikolaidis, "Towards transferring human preferences from canonical to actual assembly tasks." [Online]. Available: https://arxiv.org/abs/2111.06454

[21] Z. Wu, W. Lian, V. Unhelkar, M. Tomizuka, and S. Schaal, "Learning dense rewards for contact-rich manipulation tasks," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6214–6221.

[22] X. Zhang, L. Sun, Z. Kuang, and M. Tomizuka, "Learning variable impedance control via inverse reinforcement learning for force-related tasks," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2225–2232, 2021.

[23] Y. Sugisawa, K. Takasugi, and N. Asakawa, "Machining sequence learning via inverse reinforcement learning," *Precision Engineering*, vol. 73, pp. 477–487, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141635921002440

[24] L. Peternel, T. Petrič, and J. Babič, "Human-in-the-loop approach for teaching robot assembly tasks using impedance control interface," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1497–1502.

[25] C. W. X. Ng, K. H. K. Chan, W. K. Teo, and I.-M. Chen, "A method for capturing the tacit knowledge in the surface finishing skill by demonstration for programming a robot," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1374–1379.

[26] W. X. Ng, H. K. Chan, W. K. Teo, and I.-M. Chen, "Programming a robot for conformance grinding of complex shapes by capturing the tacit knowledge of a skilled operator," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 1020–1030, 2017.

[27] J. Hu, A. M. Kabir, S. M. Hartford, S. K. Gupta, and P. R. Pagilla, "Robotic deburring and chamfering of complex geometries in high-mix/low-volume production applications," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, 2020, pp. 1155–1160.

[28] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the Twenty-First International Conference on Machine Learning*, ser. ICML '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 1. [Online]. Available: https://doi.org/10.1145/1015330.1015430

[29] R. H. Myers and D. C. Montgomery, *Response Surface Methodology: Process and Product in Optimization Using Designed Experiments*, 1st ed. USA: John Wiley & Sons, Inc., 1995.

[30] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin, "Learning structured prediction models: A large margin approach," in *Proceedings of the 22nd International Conference on Machine Learning*, ser. ICML '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 896–903. [Online]. Available: https://doi.org/10.1145/1102351.1102464

[31] D. P. Bertsekas, *Convex Optimization Algorithms*. Athena Scientific, Belmont, Massachusetts, 2016.

[32] K. Amin, N. Jiang, and S. Singh, "Repeated inverse reinforcement learning," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/8ce6790cc6a94e65f17f908f462fae85-Paper.pdf

[33] O. M. Manyar, J. Desai, N. Deogaonkar, R. J. Joesph, R. Malhan, Z. McNulty, B. Wang, J. Barbic, and S. K. Gupta, "A simulation-based grasp planner for enabling robotic grasping during composite sheet layup," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2021. [Online]. Available: https://doi.org/10.1109/icra48506.2021.9560939

[34] M. Attene, B. Falcidieno, and M. Spagnuolo, "Hierarchical mesh segmentation based on fitting primitives," *Vis. Comput.*,

vol. 22, no. 3, p. 181–193, mar 2006. [Online]. Available: https://doi.org/10.1007/s00371-006-0375-x

[35] R. K. Malhan, A. V. Shembekar, A. M. Kabir, P. M. Bhatt, B. Shah, S. Zanio, S. Nutt, and S. K. Gupta, "Automated planning for robotic layup of composite prepreg," *Robotics and Computer-Integrated Manufacturing*, vol. 67, p. 102020, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584520302313