Learning Stage-Wise Constraints from Unsegmented Demonstrations for Robotic Tasks

Baiyu Peng and Aude Billard

***Abstract*— Learning explicit task constraints from demonstrations provides reusable task specifications for planning in robotic applications. However, many existing constraint-learning methods assume that a single constraint set remains active throughout the entire trajectory, which is restrictive for multi-stage tasks where different constraints may govern different phases and the stage boundaries are unobserved. This paper studies stage-wise constraint learning from feasible, unsegmented demonstrations. Given a task-dependent feature bank, the goal is to determine, for each stage and feature, whether the feature is inactive, equality-constrained, lower-bound constrained, or upper-bound constrained, together with the corresponding target or boundary value. We propose Stage-Wise Constraint Learning (SWCL), which jointly infers shared stage-wise constraints and demonstration-specific stage boundaries. SWCL combines constraint likelihoods of four modes, a subgoal-progress likelihood and a mode prior in a hard-EM-like framework: stage boundaries are updated by dynamic programming, while shared constraint modes and parameters are estimated from the recovered segments. Experiments on three simulated multi-stage benchmarks show that SWCL achieves more accurate and consistent recovery of stage boundaries and constraint structures than representative segmentation baselines. The learned constraints are further reused for planning in modified task instances. Real-robot experiments on a KUKA iiwa bar-clearing task show that the learned constraints enable obstacle avoidance in 90% of trials, bar clearing in 100%, and table clearing in 85%, while also transferring to a modified environment.**

***Index Terms*— Constraint learning, learning from demonstration**

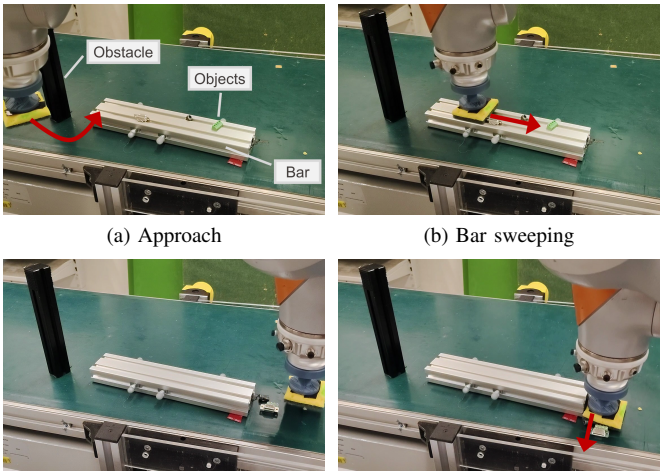
I. Introduction

Adapting industrial robot tasks to changes in workpiece geometry and workspace layout requires reusable representations of task requirements. Learning from demonstration (LfD) offers a practical way to acquire such requirements when manual specification is difficult [1]. In many manipulation tasks, however, the reusable task specification is not a policy or a reward function, but a set of constraints that must be satisfied during execution. Such constraints are central to constrained motion planning and control: they specify alignment requirements, safety margins, speed limits, and other feasibility requirements [2], [3]. For such tasks, it is therefore useful to recover these constraints explicitly from demonstrations, rather than learning only a policy. This motivates the problem of *constraint learning from demonstrations*, where the goal is to infer a reusable constraint representation from demonstrated task executions [4].

Many existing constraint-learning methods assume a constraint set that remains active and consistent throughout the trajectory. Representative examples include methods that infer constraints from locally optimal demonstrations using mixed-

This work was supported by the euROBIN project.

Baiyu Peng and Aude Billard are with the Learning Algorithms and Systems Laboratory, School of Engineering, EPFL, 1015 Lausanne, Switzerland (e-mail: baiyu.peng@epfl.ch; aude.billard@epfl.ch). Corresponding author: Baiyu Peng.



(a) Approach (b) Bar sweeping

(c) Repositioning (d) Table sweeping

Fig. 1. Real-robot execution of a bar-clearing task using the learned stage-wise constraints. A bar-shaped aluminum workpiece is placed on a conveyor, and the robot removes small objects from the bar and table before downstream processing. The task consists of five stages: (a) approaching the bar while avoiding the black obstacle; (b) sweeping the litter objects from the bar onto the table; (c) repositioning; (d) sweeping the litter objects off the table; and (e) departure (not shown).

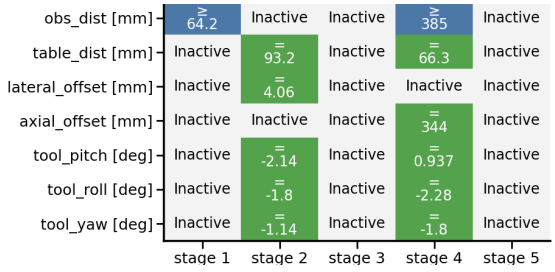


Fig. 2. Stage-wise constraints learned from demonstrations in the bar-clearing task. Rows denote seven candidate task features: obstacle clearance (obs_dist), tool height above the table (table_dist), lateral and axial offsets in the bar-fixed frame (lateral_offset, axial_offset), and tool orientation in the bar-fixed frame (tool_pitch, tool_roll, tool_yaw). Blue cells denote learned inequality constraints, green cells denote equality constraints, and white cells indicate inactive features. The constraint activation, type, and parameter for each feature at each stage are all unknown in advance and learned from demonstrations.

integer optimization, KKT conditions, or parametric and nonparametric constraint models [5]–[7], inverse constrained reinforcement learning or maximum-likelihood constraint inference methods [4], [8], [9], and a recent positive-unlabeled formulation [10]. However, this global-constraint assumption is inherently incompatible with multi-stage tasks, where different parts of a trajectory may be governed by different constraints. For example, in the bar-clearing task shown in Fig. 1, the robot must maintain obstacle clearance during approach and satisfy different tool-position and orientation constraints during bar and table sweeping.

A related but distinct line of work studies task segmentation from demonstrations, often using HMMs, changepoint detection, or neural sequence-segmentation models [11]–[13]. These approaches can decompose a demonstrated trajectory into stages, skills, or motion primitives, but their outputs are typically local dynamics or controllers rather than explicit reusable constraints. More importantly, segmentation generally does not explicitly account for active constraints, so the recovered boundaries need not align with changes in constraint structure. This motivates jointly inferring stage boundaries and stage-wise constraints from unsegmented demonstrations.

This paper addresses this gap by studying stage-wise constraint learning from demonstrations. We assume a task-dependent bank of candidate feature functions, with unknown task constraints defined on a subset of these features. For each stage, the goal is to identify the constrained features, their constraint types (equality, lower-bound inequality, or upper-bound inequality), and their target or boundary values. This is challenging because stage boundaries are unobserved and may differ across demonstrations, which may also vary in their start states, subgoals, final goals, and trajectory lengths.

We propose Stage-Wise Constraint Learning (SWCL) to jointly infer the stage boundaries and the shared constraint structure. The key idea is to segment each demonstration such that the resulting stages are consistent both with the inferred constraints and with motion toward stage-specific subgoals. SWCL combines a constraint likelihood, a directional progress likelihood, and a configurable mode prior. The resulting objective is optimized using a hard expectation-maximization (EM)-like procedure that alternates between estimating demonstration-specific stage boundaries by dynamic programming and updating the shared model parameters from the recovered segments.

Finally, we evaluate SWCL on three simulated multi-stage benchmark tasks against three baselines and further validate it on a real-robot bar-clearing task.

II. Related Work

A. Learning Constraints from Demonstrations

A substantial body of work learns a single constraint set from demonstrations through optimization-based inference [5]–[7], inverse constrained reinforcement learning and maximum-likelihood inference [4], [8], [9], positive-unlabeled learning [10], and equality-manifold learning [14].

A smaller body of work explicitly learns stage-dependent constraints. [15] extract continuous task constraints for task parameterization, including reference frames, relevant task variables, and stiffness modulation. Their method handles only equality constraints, whereas ours also learns inequalities such as maximum velocity or minimum obstacle clearance. [16] model demonstrations as a sequence of subtasks with subgoals and locally active constraints within a Bayesian nonparametric framework. However, the features within each stage are assumed to be either all constrained or all unconstrained, whereas our method allows different features to have different constraint statuses. [17] first segment demonstrations using a standard top-down procedure and then extract constraints within each segment. In contrast, our method learns segmentation and stage-wise constraints jointly. Other related works study task-specific constraint learning, such as keyframe-wise geometric constraint selection, dynamic constraint frames for hybrid force/position control, and contact-mode-specific kinematic constraints in contact-rich tasks [18]–[20]. These approaches are designed for task-specific structures, which are less generalizable.

B. Task Segmentation

HMMs and HSMMs are widely used to segment demonstrations into reusable skills or dynamical systems [11], [21]. [22] use Bayesian nonparametric autoregressive HMMs to recover finite-state task representations, while [23] combine HMMs with stable linear-parameter-varying systems to learn task sequences. Offline changepoint detection instead partitions sequences by optimizing an additive segment cost [12]. ChangeForest detects multivariate changepoints through random-forest-based segment comparison [24]. ASFormer uses a Transformer with local temporal inductive biases to predict action labels and refine segment boundaries [25]. Language-guided skill learning uses large language models to initialize trajectory segmentations, which are refined through hierarchical variational skill discovery [26].

These methods primarily recover stage labels, boundaries, local models, or controllers rather than explicit reusable constraints. In SWCL, segmentation serves to recover shared stage-wise constraint modes and parameters. Candidate segments are evaluated using constraint-specific likelihoods, subgoal progress, and a constraint model shared across demonstrations.

III. Preliminaries and Problem Formulation

We consider multi-stage tasks whose valid executions are characterized by known stage-independent feasibility requirements and unknown stage-dependent constraints on task features. Let $\mathcal{X} \subseteq \mathbb{R}^{d_x}$ denote the state space, and let $\Phi = \{\phi_j : \mathcal{X} \to \mathbb{R}\}_{j=1}^{F}$ be a given task-dependent feature bank. For a state x_t, the full feature vector is $\phi(x_t) = [\phi_1(x_t), \dots, \phi_F(x_t)]^\top \in \mathbb{R}^F$. Before learning, each feature dimension is standardized over the demonstrated data.

A task consists of K ordered stages with their own semantics. For a trajectory $\tau = \{x_t\}_{t=1}^{T}$, its cutpoints are denoted by $B = \{b_k\}_{k=0}^{K}$ and satisfy $1 = b_0 < b_1 < \cdots < b_K = T + 1$. The cutpoints induce the stage intervals $\mathcal{I}_k(B) = \{b_{k-1}, \dots, b_k - 1\}$ and the corresponding stage

subgoals $g_k(B) = x_{b_k-1}$. When learning from demonstrations, the cutpoints and intermediate stage endpoints are unknown and may differ across trajectories.

The true stage-wise constraint of a task is represented by a mode matrix $\Gamma = [\gamma_{k,j}] \in \{\varnothing, \mathrm{eq}, \mathrm{lb}, \mathrm{ub}\}^{K\times F}$. Here, $\gamma_{k,j} = \varnothing$ indicates that feature j is unconstrained in stage k, while eq, lb, and ub denote an equality constraint, a lower-bound inequality, and an upper-bound inequality, respectively.

For every constrained stage-feature pair, $\eta_{k,j} \in \mathbb{R}$ denotes its scalar constraint parameter. It specifies the target feature value for an equality constraint and the boundary value for an inequality constraint. The collection of active parameters is denoted by $\mathcal{P} = \{\eta_{k,j} : \gamma_{k,j} \neq \varnothing\}$. The mode and parameter of each stage-feature pair induce the feasible set

$$\mathcal{C}_{k,j} = \begin{cases} \mathcal{X}, & \gamma_{k,j} = \varnothing, \\ \{x \in \mathcal{X} : \phi_j(x) = \eta_{k,j}\}, & \gamma_{k,j} = \mathrm{eq}, \\ \{x \in \mathcal{X} : \phi_j(x) \geq \eta_{k,j}\}, & \gamma_{k,j} = \mathrm{lb}, \\ \{x \in \mathcal{X} : \phi_j(x) \leq \eta_{k,j}\}, & \gamma_{k,j} = \mathrm{ub}. \end{cases} \tag{1}$$

Given the stage-wise constraint $(\Gamma, \mathcal{P})$, a trajectory length T, a start state x_{start}, and execution-specific ordered subgoals $\{g_k\}_{k=1}^K$, a valid task execution is defined as a solution to the following feasibility problem:

$$\begin{aligned} \text{find} \quad & \tau = \{x_t\}_{t=1}^T, \quad B = \{b_k\}_{k=0}^K \\ \text{s.t.} \quad & 1 = b_0 < b_1 < \cdots < b_K = T+1, \\ & x_1 = x_{\mathrm{start}}, \\ & x_{b_k-1} = g_k, \qquad k = 1, \ldots, K, \\ & \tau \in \mathcal{C}_{\mathrm{known}}, \\ & x_t \in \bigcap_{j=1}^{F} \mathcal{C}_{k,j}, \qquad t \in \mathcal{I}_k(B), \quad k = 1, \ldots, K. \end{aligned} \tag{2}$$

Here, $\mathcal{C}_{\mathrm{known}}$ contains known trajectory-level feasibility requirements, such as system dynamics and known kinematic limits.

We are given a set of feasible demonstrations $\mathcal{D} = \{\tau^{(i)}\}_{i=1}^N$, where $\tau^{(i)} = \{x_t^{(i)}\}_{t=1}^{T_i}$. The demonstrations may have different lengths, start states, subgoals, and latent cutpoints, but share the same number and ordering of stages and the same stage-wise constraint model $(\Gamma, \mathcal{P})$. Each demonstration is assumed to satisfy (2) under its true latent cutpoints.

Definition 1 (Stage-wise constraint learning): Given feasible demonstrations $\mathcal{D}$, the number of stages K, and the feature bank Φ, stage-wise constraint learning aims to recover:

1) the shared constraint mode matrix $\Gamma = [\gamma_{k,j}]$, which specifies the constraint mode of every stage-feature pair;
2) the scalar constraint parameters $\mathcal{P} = \{\eta_{k,j} : \gamma_{k,j} \neq \varnothing\}$.

The demonstration-specific cutpoints and their induced intermediate subgoals are also unknown and are jointly estimated during learning as auxiliary latent variables.

IV. Method

A. Overview

The key idea of SWCL is to use a shared constraint model itself to explain and segment the demonstrations. Rather than representing a stage with a generic feature distribution, SWCL evaluates each stage-feature pair under constraint-specific modes and selects the best-fitting mode.

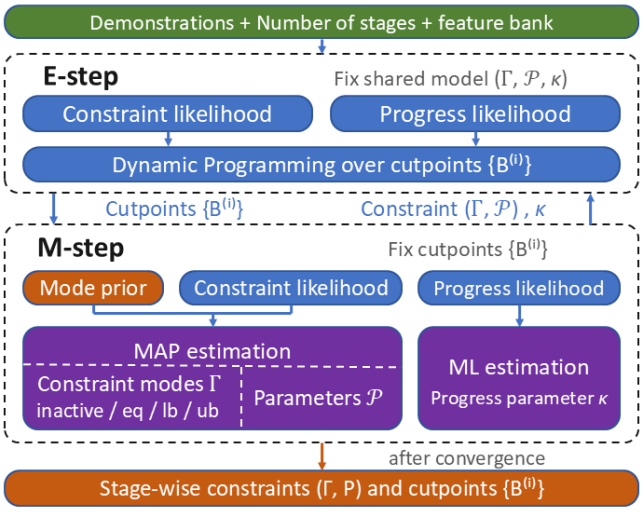


Fig. 3. Schematic of the proposed method SWCL. It formulates stage-wise constraint learning as maximizing a posterior objective that combines a constraint-mode prior, a constraint likelihood, and an auxiliary progress likelihood. Each stage–feature pair is assigned one of four constraint modes: inactive, equality (eq), lower bound (lb), or upper bound (ub), collectively forming $\boldsymbol{\Gamma}$. In the hard-EM-like procedure, the hard E-step finds demonstration-specific cutpoints by dynamic programming with the shared model fixed; the M-step uses the recovered intervals to update $(\boldsymbol{\Gamma}, \boldsymbol{\mathcal{P}})$ through MAP estimation and $\boldsymbol{\kappa}$ through ML estimation. Here, $\boldsymbol{\mathcal{P}}$ contains equality targets or inequality bounds for active constraints, and $\boldsymbol{\kappa}$ contains one auxiliary progress parameter per stage.

More specifically, let $B^{(i)}$ denote the latent cutpoints of demonstration i, and let $\mathcal{I}_k^{(i)}$ be its interval assigned to stage k. Let $\boldsymbol{\kappa} = \{\kappa_k\}_{k=1}^K$ denote the shared stage-wise progress parameters. At a conceptual level, SWCL jointly learns the stage-wise constraints and demonstration cutpoints by maximizing the posterior objective

$$p\Big(\{B^{(i)}\}_{i=1}^N, \Gamma, \mathcal{P}, \boldsymbol{\kappa} \mid \mathcal{D}\Big) \propto p(\Gamma)\, \mathcal{L}_{\mathrm{con}} \mathcal{L}_{\mathrm{prog}}, \tag{3}$$

where $p(\Gamma)$ is the constraint-mode prior, $\mathcal{L}_{\mathrm{con}}$ is the likelihood that measures how well the feature observations agree with the stage-wise constraint model, and $\mathcal{L}_{\mathrm{prog}}$ measures whether the motion within each stage advances toward its endpoint. Priors on the remaining variables are assumed to be constant and are omitted.

SWCL approximately maximizes (3) using a hard-EM-like procedure illustrated in Fig. 3. Section IV-B defines the constraint likelihood, as well as $\mathcal{L}_{\mathrm{con}}$, and $p(\Gamma)$. Section IV-C defines $\mathcal{L}_{\mathrm{prog}}$. Section IV-D then describes the two alternating updates and initialization.

B. Constraint Likelihood and Mode Prior

For a candidate stage interval, each feature is evaluated under a mode $m \in \mathcal{M} = \{\varnothing, \mathrm{eq}, \mathrm{lb}, \mathrm{ub}\}$. Every mode defines a proper density for the observed feature values. The inactive mode describes ordinary local variation, whereas the three active modes express a task-level target or boundary shared

across demonstrations. Besides the shared constraint parameter, each density may contain local distribution parameters that account for variation across individual demonstrations. Figure 4 illustrates the distributions of four constraint modes on an example feature distribution.

1) Equality Mode: An equality-constrained feature is expected to remain close to a common task-level target $\eta_{k,j}$. Its observation model is

$$p_{\mathrm{eq}}\left(\phi \mid \eta_{k,j}, \sigma_{k,j}^{(i)}\right) = \mathcal{N}\left(\phi; \eta_{k,j}, (\sigma_{k,j}^{(i)})^2\right), \qquad \sigma_{k,j}^{(i)} \le \bar{\sigma}_{\mathrm{eq}}, \tag{4}$$

where the target $\eta_{k,j}$ is shared across demonstrations, while $\sigma_{k,j}^{(i)}$ describes the local dispersion of demonstration i around that target. For a fixed target and candidate interval, we use the Gaussian maximum-likelihood scale capped at a hyperparameter $\bar{\sigma}_{\mathrm{eq}}$. This upper bound specifies how tightly the observations must concentrate around a shared target for an equality interpretation to remain plausible. The shared target itself is optimized across demonstrations in the M-step.

2) Inequality Modes: We adopt the modeling assumption that, when a one-sided inequality is active and sufficiently expressed in the demonstrations, its feature values exhibit a boundary-anchored distribution: the observations remain primarily on the feasible side, while a non-negligible amount of probability mass occurs near the boundary. This pattern provides evidence for identifying both the activation and direction of the inequality. In contrast, if all demonstrations remain far inside the feasible region and never approach the true boundary, the inequality is generally not identifiable from feasible demonstrations alone [5]. In such cases, prior knowledge about its activation and direction can instead be encoded through the configurable mode prior introduced later.

Mathematically, for an active inequality with shared boundary $\eta_{k,j}$, we define the signed slack and model the slack with a half-Student-t density. To accommodate observation noise and occasional small violations, we attach a soft exponential tail to the infeasible side.

$$y_\gamma(\phi; \eta_{k,j}) = \begin{cases} \phi - \eta_{k,j}, & \gamma = \mathrm{lb}, \\ \eta_{k,j} - \phi, & \gamma = \mathrm{ub}. \end{cases} \tag{5}$$

$$p_\gamma(\phi \mid \eta_{k,j}, \sigma_{k,j}^{(i)}) = \frac{1}{Z_{\nu,\alpha}} \begin{cases} \dfrac{2}{\sigma_{k,j}^{(i)}} t_\nu\left(\dfrac{y_\gamma}{\sigma_{k,j}^{(i)}}\right), & y_\gamma \ge 0, \\ \dfrac{2}{\sigma_{k,j}^{(i)}} t_\nu(0) \exp\left(\dfrac{y_\gamma}{\alpha \sigma_{k,j}^{(i)}}\right), & y_\gamma < 0, \end{cases} \tag{6}$$

where t_ν is the standard Student-t density with $\nu = 3$, $\sigma_{k,j}^{(i)}$ is a local scale, $\alpha = 0.1$ is a fixed parameter controlling the boundary softness, and $Z_{\nu,\alpha} = 1 + 2\alpha t_\nu(0)$ is the normalizing constant. The local scale describes how widely the feasible observations spread away from the shared boundary; it is estimated separately for every demonstration–stage interval.

Ideally, $\sigma_{k,j}^{(i)}$ would be estimated by maximizing the interval likelihood for a fixed mode and boundary. Because this would require an additional numerical optimization for every candidate interval and boundary, we instead determine it by matching the 90th percentile of the feasible slacks to that of the half-Student-t distribution.

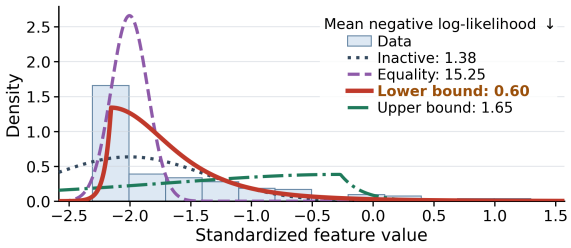


Fig. 4. Illustration of the four constraint-mode fits to an example dataset. Each curve uses its fitted parameters, and the legend reports the corresponding mean negative log-likelihood (lower is better). In this example, the lower-bound mode matches the data distribution the best and provides the strongest likelihood evidence.

3) Inactive Mode: When feature j is inactive in stage k, its values need not concentrate around a task-level shared parameter. We use a local Gaussian with a relatively large scale as a neutral reference model:

$$p_\varnothing\left(\phi \mid \mu_{k,j}^{(i)}, \sigma_{k,j}^{(i)}\right) = \mathcal{N}\left(\phi; \mu_{k,j}^{(i)}, (\sigma_{k,j}^{(i)})^2\right), \quad \sigma_{k,j}^{(i)} > c_{\mathrm{bg}} \bar{\sigma}_{\mathrm{eq}}, \tag{7}$$

where $\mu_{k,j}^{(i)}$ and $\sigma_{k,j}^{(i)}$ are estimated separately for each demonstration–stage interval by minimizing its Gaussian negative log-likelihood subject to the scale lower bound. We use $c_{\mathrm{bg}} = 1$, which prevents the locally centered inactive model from mimicking an equality constraint by shrinking its variance. Thus, inactive feature distributions may differ across demonstrations without inducing a shared task constraint.

4) Overall Constraint Likelihood with Mode Prior: For a candidate interval $\mathcal{I}$ from demonstration i assigned to stage k, define the constraint negative log-likelihood

$$\ell_{\mathrm{con},k}^{(i)}(\mathcal{I}) = \sum_{j=1}^{F} \sum_{t \in \mathcal{I}} -\log p_{\gamma_{k,j}}\left(\phi_j(x_t^{(i)}) \mid \eta_{k,j}, \hat{\psi}_{k,j}^{(i)}(\mathcal{I})\right), \tag{8}$$

where $\gamma_{k,j}$ is the single mode assigned to feature j in stage k, and $p_{\gamma_{k,j}}$ denotes the corresponding density $p_\varnothing$, p_{eq}, p_{lb}, or p_{ub}. The local parameters $\hat{\psi}_{k,j}^{(i)}(\mathcal{I})$ are fitted on the candidate interval using the rules above, while $\eta_{k,j}$ is omitted for the inactive mode.

The likelihood alone may be insufficient to identify the correct mode, particularly when an inequality boundary is weakly expressed. SWCL therefore assigns each candidate a configurable prior probability $\pi_{k,j}(\gamma)$, with

$$\pi_{k,j}(\gamma) \ge 0, \qquad \sum_{\gamma \in \{\varnothing, \mathrm{eq}, \mathrm{lb}, \mathrm{ub}\}} \pi_{k,j}(\gamma) = 1. \tag{9}$$

For example, if feature j is known to be unconstrained in stage k, the prior probabilities of its equality, lower-bound, and upper-bound modes can all be set to zero. The full mode prior is $p(\Gamma) = \prod_{k=1}^{K} \prod_{j=1}^{F} \pi_{k,j}(\gamma_{k,j})$ and is applied once to each shared stage-feature mode.

C. Subgoal Progress Likelihood

Recall from Section III that we assume each stage is a goal-reaching task whose endpoint defines its subgoal. To further reduce ambiguity in segmentation and constraint interpretation,

we augment the objective with a subgoal progress likelihood that characterizes whether the motion within a candidate stage consistently advances toward its endpoint.

A candidate interval $\mathcal{I}(s,e) = \{s,\dots,e\}$ from demonstration i induces the endpoint $g^{(i)}(\mathcal{I}) = x_e^{(i)}$. For each nonzero transition inside the interval, we define the normalized directional-progress score

$$\delta_t^{(i)}(\mathcal{I}) = \frac{\|x_t^{(i)} - g^{(i)}(\mathcal{I})\|_2 - \|x_{t+1}^{(i)} - g^{(i)}(\mathcal{I})\|_2}{\|x_{t+1}^{(i)} - x_t^{(i)}\|_2}, \tag{10}$$

where $\delta_t^{(i)} \in [-1,1]$. A value near one indicates motion directly toward the endpoint, zero indicates approximately tangential motion, and a negative value indicates motion away from it. This term therefore favors candidate intervals in which the motion consistently approaches the endpoint. Motivated by von Mises–Fisher random-walk models [27], we model the normalized progress score using the exponential-family density

$$p(\delta \mid \kappa_k) = \frac{\kappa_k \exp(\kappa_k \delta)}{2\sinh(\kappa_k)}, \qquad \delta \in [-1,1], \tag{11}$$

with the limiting uniform density $p(\delta \mid 0) = 1/2$. For an interval containing n valid transitions, its negative log-likelihood under stage k is, up to the constant cost of the uniform reference density,

$$\ell_{\mathrm{prog},k}^{(i)}(\mathcal{I}) = n \log\left(\frac{\sinh \kappa_k}{\kappa_k}\right) - \kappa_k \sum_{t\in\mathcal{I}} \delta_t^{(i)}(\mathcal{I}). \tag{12}$$

The complete progress likelihood $\mathcal{L}_{\mathrm{prog}}$ is the product of (11) over all valid within-stage transitions in all demonstrations. Thus, $\kappa_k = 0$ disables the progress contribution, whereas a larger κ_k favors intervals that move consistently toward their endpoints. This term helps determine the stage cutpoints but is not interpreted as a learned task constraint.

D. Hard-EM-like Inference

The cutpoints and shared model depend on each other: the model is estimated from the observations assigned to each stage, while the cutpoints are selected according to how well the resulting stages fit the current model. SWCL resolves this coupling by alternating between the hard E-step and M-step described below.

1) Hard E-step: Cutpoint Update: With $(\Gamma, \mathcal{P}, \boldsymbol{\kappa})$ fixed, the demonstrations can be processed independently. For a candidate interval $\mathcal{I}$ assigned to stage k, the constraint term is given by (8). Combining it with the progress term gives

$$\ell_k^{(i)}(\mathcal{I}) = \ell_{\mathrm{con},k}^{(i)}(\mathcal{I}) + \ell_{\mathrm{prog},k}^{(i)}(\mathcal{I}). \tag{13}$$

The hard E-step selects

$$\hat{B}^{(i)} = \arg\min_{B^{(i)}} \sum_{k=1}^{K} \ell_k^{(i)}(\mathcal{I}_k^{(i)}). \tag{14}$$

Because this objective is additive over consecutive stage intervals, it can be solved exactly for the current shared model by dynamic programming:

$$F_k^{(i)}(b_k) = \min_{b_{k-1}<b_k} \left[F_{k-1}^{(i)}(b_{k-1}) + \ell_k^{(i)}(\mathcal{I}_k^{(i)})\right], \tag{15}$$

where $\mathcal{I}_k^{(i)}$ is the candidate interval between b_{k-1} and b_k, and $F_0^{(i)}(1) = 0$. Backtracking from $F_K^{(i)}(T_i+1)$ recovers the complete cutpoint set $\hat{B}^{(i)}$. The update is hard because it retains this single MAP solution rather than a distribution over possible cutpoints.

2) M-step: Constraint Model Update: With the cutpoints fixed, the observations assigned to every stage are known. The M-step jointly selects the shared mode and, when active, its shared target or boundary. For candidate mode γ and shared parameter η, define the demonstration-normalized negative log-likelihood

$$\ell_{k,j}^{(i)}(\gamma,\eta) = \frac{1}{|\mathcal{I}_k^{(i)}|} \sum_{t\in\mathcal{I}_k^{(i)}} -\log p_\gamma\left(\phi_j(x_t^{(i)}) \mid \eta, \hat{\psi}_{k,j}^{(i)}(\gamma,\eta)\right), \tag{16}$$

where the local parameters $\hat{\psi}_{k,j}^{(i)}(\gamma,\eta)$ are fitted separately on each fixed demonstration–stage interval conditional on the candidate mode and shared parameter. The equality scale uses the capped Gaussian maximum-likelihood estimate, whereas the inequality scale uses the 90th-percentile rule described above. Dividing by the interval length prevents demonstrations with longer stages or denser sampling from dominating the shared constraint estimate.

The shared mode and active parameter are updated by

$$(\hat{\gamma}_{k,j}, \hat{\eta}_{k,j}) = \arg\min_{\gamma,\eta} \left[\sum_{i=1}^{N} \ell_{k,j}^{(i)}(\gamma,\eta) - \log \pi_{k,j}(\gamma)\right], \tag{17}$$

Operationally, SWCL evaluates the four modes separately. For each active mode, it first finds the target or boundary that minimizes the aggregated negative log-likelihood and retains the resulting optimized score and parameter. The inactive mode produces an analogous score without optimizing a shared parameter. SWCL then adds the corresponding negative log-prior to each optimized likelihood score and selects the mode with the smallest total; if an active mode is selected, its optimized target or boundary is retained as $\hat{\eta}_{k,j}$. The mode prior is applied once because $\gamma_{k,j}$ is one shared task-level variable. Since the lower- and upper-bound modes are separate candidates, this comparison determines both whether an inequality is active and its direction. The progress concentration of each stage is updated by

$$\hat{\kappa}_k = \arg\min_{\kappa\ge 0} \sum_{i=1}^{N} \ell_{\mathrm{prog},k}^{(i)}\left(\mathcal{I}_k^{(i)}; \kappa\right). \tag{18}$$

3) Initialization: The first hard E-step requires an initial shared model. We begin by processing each demonstration independently. With $\boldsymbol{\kappa} = \mathbf{0}$, the progress density is uniform, and every candidate interval selects its best locally fitted constraint modes without sharing modes or constraint parameters across demonstrations. Dynamic programming then returns provisional cutpoints $B^{(i)}$. Treating the resulting intervals as corresponding stages, we perform the first M-step to initialize $(\Gamma, \mathcal{P}, \boldsymbol{\kappa})$. Subsequent iterations use these shared parameters.

V. Simulation Experiment

Algorithm 1 Stage-Wise Constraint Learning

Input: demonstrations $\mathcal{D}$, number of stages K, feature bank Φ, mode priors $\{\pi_{k,j}\}$

Output: cutpoints $\{\hat{B}^{(i)}\}$, constraint model $(\hat{\Gamma}, \hat{\mathcal{P}})$, progress parameters $\hat{\boldsymbol{\kappa}}$

Extract and standardize the feature trajectories
for all demonstrations $i = 1, \ldots, N$ **do**
    Initialize $B^{(i)}$ by locally fitting candidate intervals and applying dynamic programming
end for
Initialize $(\Gamma, \mathcal{P}, \boldsymbol{\kappa})$ from the provisional stage intervals
repeat
    E-step: Update demonstration-specific cutpoints
    for all demonstrations $i = 1, \ldots, N$ **do**
        Update $B^{(i)}$ by (15) and backtracking
    end for
    M-step: Update shared model parameters
    for all stages $k = 1, \ldots, K$ **do**
        for all features j **do**
            Update $(\gamma_{k,j}, \eta_{k,j})$ by (17)
        end for
        Update κ_k by (18)
    end for
until the cutpoints and shared model converge, or the maximum iteration is reached
return $\{\hat{B}^{(i)}\}$, $\hat{\Gamma}$, $\hat{\mathcal{P}}$, and $\hat{\boldsymbol{\kappa}}$

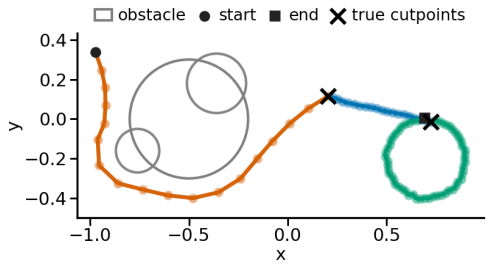


Fig. 5. Example demonstration from the `S3ObsAvoid` environment. The colored trajectory segments correspond to the three ground-truth stages: obstacle avoidance, transition, and terminal arc following. Gray ellipses denote the composite obstacle, the black circle and square mark the start and end states, and black crosses indicate the true stage cutpoints.

A. Experimental Setup

1) Benchmark Datasets: We evaluate SWCL on three simulated multi-stage benchmark tasks: `S3ObsAvoid`, `S4SlideInsert`, and `S5SphereInspect` with three baselines. The tasks have three, four, and five ordered stages, respectively, and evaluate stage-wise constraint-mode identification and parameter estimation. Code and data will be made publicly available upon acceptance.

S3ObsAvoid. A three-stage planar navigation task: obstacle avoidance → transition → terminal arc following. Fig. 5 shows a representative demonstration and its ground-truth stage cutpoints.

S4SlideInsert. A four-stage slide-and-insert task: free-space approach → surface contact and alignment → guided sliding → final insertion. The task is performed by a UR5 arm simulated in PyBullet.

S5SphereInspect. A five-stage spherical inspection task: approach → surface inspection → transition → outer-shell inspection → departure. The task is also performed by a simulated UR5 arm, and Fig. 9 visualizes its five stages.

TABLE I
FEATURE BANKS USED IN THE THREE BENCHMARK TASKS. UNITS REFER TO THE FEATURE VALUES BEFORE STANDARDIZATION.

Feature	Description	Unit
`S3ObsAvoid`		
`obs_dist`	Distance to the obstacle	cm
`speed`	Speed magnitude	cm/s
`arc_dist`	Distance to the terminal-circle center	cm
`heading`	Heading angle	rad
`line_dist`	Distance to an irrelevant straight line	cm
`noise`	Sine noise	–
`S4SlideInsert`		
`surf_dist`	Distance to the table contact surface	mm
`center_dist`	Absolute lateral distance from the slot centerline	mm
`orient_err`	Absolute orientation error relative to the slot	deg
`speed`	Speed magnitude of the tool	mm/s
`normal_force`	Normal contact force applied against the guide	N
`start_dist`	Distance to the demonstration start pose	mm
`insert_err`	Remaining insertion distance along the slot	mm
`S5SphereInspect`		
`surf_dist`	Radial distance to the sphere surface	mm
`normal_err`	Angle between the tool axis and the sphere normal	deg
`speed`	Speed magnitude of the tool	mm/s
`ang_speed`	Angular speed magnitude	deg/s
`start_dist`	Distance to the demonstration start position	mm
`goal_dist`	Distance to the demonstration final goal	mm

The feature banks of the three tasks are summarized in Table I, while their ground-truth and learned stage-wise constraints are compared in Fig. 7. Only a subset of features is constrained in each stage; the rest test identification of inactive or decoy features. All methods are trained on the same 10 demonstrations per task, generated as feasible planned trajectories satisfying the task structure in (2).

For hyperparameters of our method, we set the equality-scale upper bound $\bar{\sigma}_{\text{eq}}$ to 0.1 for `S3ObsAvoid` and `S4SlideInsert`, and to 0.05 for `S5SphereInspect`. For every stage-feature pair, we use an inactive-mode prior of 0.5 and divide the remaining probability equally among the three active modes. All the other parameters, except for the number of stages, are set to be consistent across three datasets.

2) Baselines: Since most existing constraint-learning methods are not designed to recover multi-stage constraints directly, we construct three baselines from representative trajectory-segmentation paradigms. For end-to-end constraint recovery, each baseline first segments the demonstrations using its own model and inference procedure. The recovered cutpoints are then fixed, and the M-step of SWCL, with the same hyperparameters, is applied to estimate the constraint modes and parameters.

With this common constraint-recovery backend, the comparison primarily measures the effect of segmentation on constraint recovery. SWCL is evaluated against the ground-truth

constraint modes and parameters, and through the downstream planning with the learned constraints in simulation and the real robot.

Gaussian Mixture Model Hidden Markov Model (GMM-HMM). This baseline uses a left-to-right hidden Markov model with a Gaussian-mixture emission for each stage [11]. The emissions are fitted to the standardized joint state-feature representation $(x_t, \phi(x_t))$, and the inferred latent-state sequence determines the stage cutpoints.

Autoregressive Hidden Semi-Markov Model (ARHSMM). This baseline segments the demonstrations using an autoregressive hidden semi-Markov model [21]. Its autoregressive observation model is fitted to the standardized joint state-feature representation $(x_t, \phi(x_t))$, while its explicit duration model captures the persistence of each stage.

ChangeForest. This more recent baseline applies the non-parametric random-forest classification to measure distributional differences across candidate split points [24].

3) Evaluation metrics: We evaluate the four methods mainly on constraint recovery quality and secondly on segmentation quality.

Semantic constraint intersection over union (IoU). We represent the learned and ground-truth constraint structures as sets of active stage-feature-mode tuples, where the mode distinguishes equality, lower-bound, and upper-bound constraints. Their semantic IoU is the size of the intersection divided by the size of the union. Intersection requires matching both the stage-feature pair and its constraint mode. The metric therefore penalizes missed constraints, falsely activated features, and incorrect inequality directions. Higher values indicate better constraint recovery.

Mean absolute cutpoint error. For each demonstration, we compute absolute errors, measured in timesteps, between predicted and corresponding ground-truth cutpoints. We average these errors first over cutpoints within each demonstration, then across demonstrations. Lower values indicate more accurate segmentation.

B. Results

1) Benchmark Comparison: Figure 6 compares SWCL with the three baselines on the benchmark datasets. GMM-HMM and ARHSMM are each trained from 10 random initializations, and we report the run with the highest final segmentation log-likelihood. ChangeForest is similarly evaluated with 10 random seeds, from which we select the run with the largest total change-point gain. In contrast, SWCL is run once using the initialization procedure described in Section IV-D. The hyperparameters of SWCL and the baselines are selected through method-specific parameter sweeps.

On `S3ObsAvoid`, SWCL, GMM-HMM, and ARHSMM all recover nearly exact cutpoints and achieve a constraint IoU of 1.0. On `S4SlideInsert`, SWCL obtains a cutpoint error of 0.93 and an IoU of 0.87, whereas the baselines have cutpoint errors between 8.90 and 12.53 and IoUs between 0.47 and 0.67, with ARHSMM performing best among them. On `S5SphereInspect`, SWCL achieves a cutpoint error of 0.85 and an IoU of 1.0. ChangeForest is the closest baseline

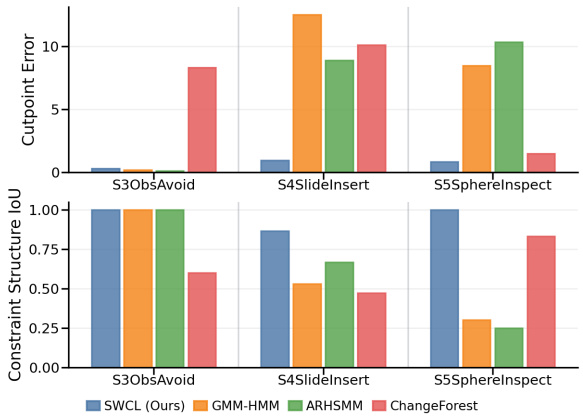


Fig. 6. Main benchmark comparison across the three datasets. Top: mean absolute cutpoint error in timesteps (lower is better). Bottom: semantic constraint-structure IoU (higher is better).

on this task, with a cutpoint error of 1.50 and an IoU of 0.83, while GMM-HMM and ARHSMM perform substantially worse.

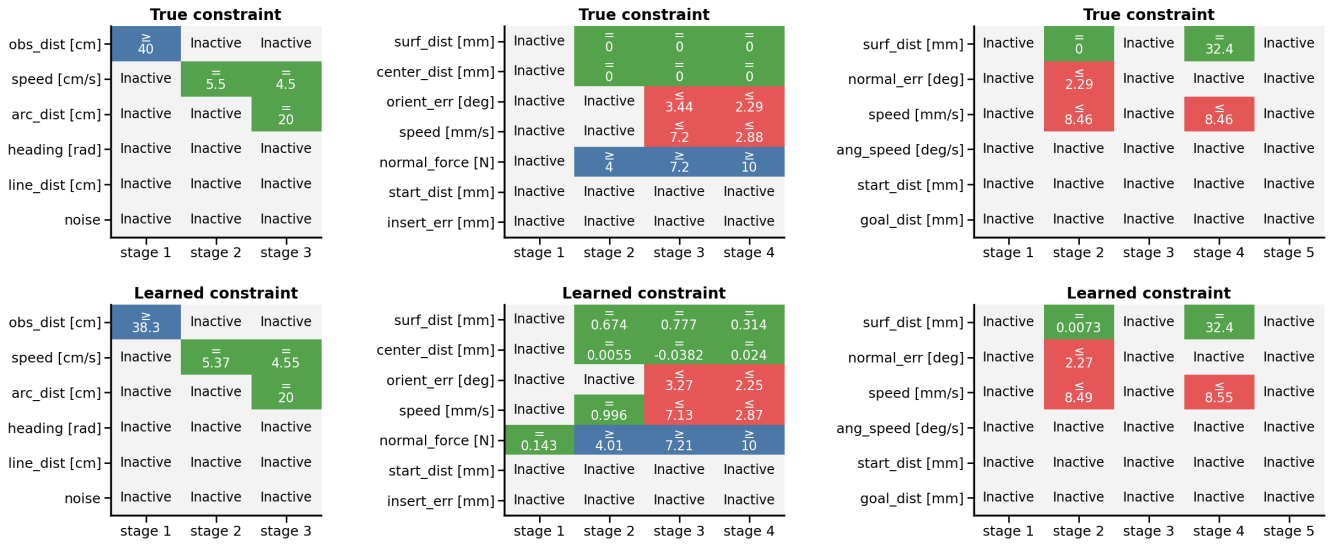
Overall, SWCL is the only method that maintains accurate constraint recovery across all three tasks. We attribute this consistency to its constraint-aware segmentation objective: unlike the baselines, which segment trajectories using generic observation or change-point models and infer constraints only afterward, SWCL directly incorporates constraint-specific likelihoods and directional subgoal progress into cutpoint inference.

2) Learned Constraint Modes and Parameters: Beyond scalar errors, we examine the stage-wise constraint structure recovered by SWCL. Fig. 7 compares the ground-truth constraint matrix with the learned shared matrix $\hat{\Gamma} = [\hat{\gamma}_{k,j}]$ for each benchmark task. Each cell represents one stage-feature pair and is labeled as inactive, equality-constrained, lower-bound constrained, or upper-bound constrained.

SWCL recovers the complete constraint-mode structure on `S3ObsAvoid` and `S5SphereInspect`. On `S4SlideInsert`, it recovers all ground-truth active modes and their directions, but introduces two additional equality constraints: normal force in stage 1 and speed in stage 2. Before contact, the normal force remains close to zero with little variation, while the stage-2 motion maintains a relatively consistent speed across demonstrations. These incidental concentrations resemble shared equality constraints in the observed data and are therefore identified as active. Despite these two false positives, SWCL recovers the dominant stage-wise constraint structure and produces interpretable values for the active constraints.

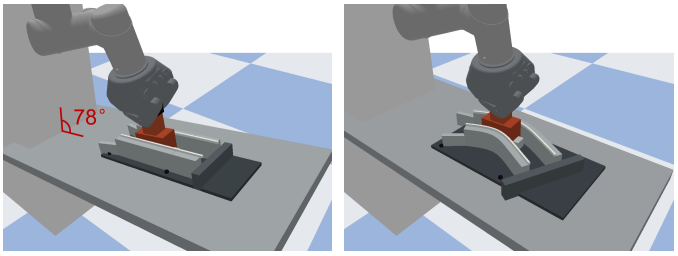
C. Reuse the Learned Constraints for Planning

We evaluate reuse of the learned stage-wise constraints for planning. Given the learned constraint modes $\hat{\Gamma}$ and parameters $\{\hat{\eta}_{k,j}\}$, we construct new task instances with prescribed start states, final goals, and trajectory lengths, and use the learned active constraints as stage-wise feasibility conditions in (2). We do not prescribe intermediate subgoals during



(a) `S3ObsAvoid` (b) `S4SlideInsert` (c) `S5SphereInspect`

Fig. 7. Comparison of the ground-truth and learned stage-wise constraint matrices for the three benchmark tasks. In each panel, the upper matrix shows the ground-truth constraints and the lower matrix shows the constraints recovered by SWCL. Each cell corresponds to one stage-feature pair. Colors indicate the constraint mode, while the cell text reports the equality target or inequality boundary in the physical units shown on the vertical axis. Inactive entries indicate unconstrained stage-feature pairs.



(a) Tilted surface transfer (b) Curved rail transfer

Fig. 8. Transfer of learned `S4SlideInsert` constraints to two modified tasks with the same stage-wise constraint semantics but different environments. (a) The guide surface is tilted. (b) The straight guide is replaced by a curved rail.

planning. Stage lengths follow the average demonstrated stage-length ratios, determining the cutpoints B. We minimize a smoothness objective subject to the learned constraints and known feasibility requirements $\mathcal{C}_{\text{known}}$. Here, $\mathcal{C}_{\text{known}}$ mainly imposes global task-agnostic velocity limits, distinct from the learned stage-wise speed constraints encoding task-specific motion requirements. The planned task-space trajectory is then converted to UR5 joint targets by inverse kinematics and tracked in PyBullet using its integrated torque control mode in `S4SlideInsert`, and position control mode in `S5SphereInspect`.

For `S4SlideInsert`, we test transferring the learned constraints to two modified environments that preserve the same stage-wise constraint semantics. While the original task uses a straight guide rail and a horizontal surface, the first uses a tilted table surface, while the second replaces the straight guide rail with a curved one. In these transfer tasks, feature functions are given for the modified geometry; SWCL transfers only the learned stage-wise constraint modes and parameters. Figure 8 shows the two transfer tasks with the learned constraints.

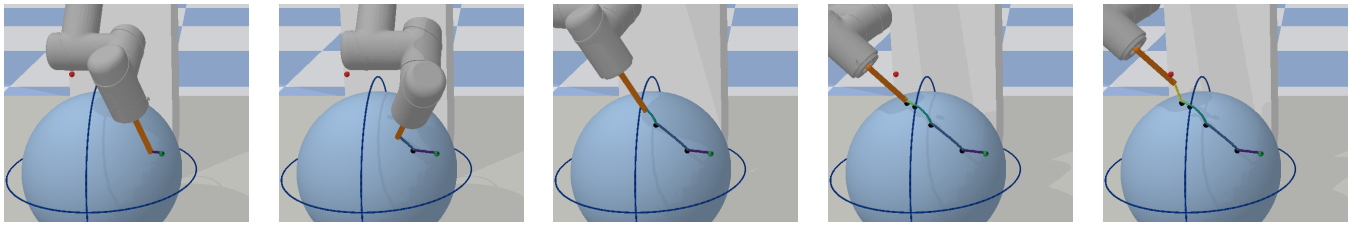
For `S5SphereInspect`, we plan and execute new trajectories in the same spherical inspection environment but with different start and terminal configurations. Figure 9 shows five snapshots from one PyBullet execution. The planned trajectory follows the learned stage-wise constraints while adapting the surface-inspection and outer-shell inspection stages to the new task instance.

We quantify execution feasibility by the per-step violation rate of true stage-wise constraints. We report the fraction of timesteps violating any truly active constraint in the corresponding stage. The violation rates are 4.37% for the tilted-table `S4SlideInsert` transfer, 0.61% for the curved-rail `S4SlideInsert` transfer, and 0.11% for `S5SphereInspect`. These results indicate that the learned constraints can be used as reusable planning constraints even beyond the original demonstrations. The video of all three experiments is attached as supplementary material.

VI. Real-robot Validation

A. Experiment Setup and Constraint Learning

We validate SWCL on a bar-clearing task motivated by workpiece preparation before downstream processing. The robot removes residual objects from the workpiece and surrounding table. Changes in workpiece geometry and obstacle placement motivate reusing stage-wise task requirements while adapting the execution trajectory. The task consists of five stages: approach the bar end while avoiding the obstacle $\rightarrow$ sweep objects off the bar surface $\rightarrow$ reposition $\rightarrow$ sweep objects off the table $\rightarrow$ departure. Fig. 1 illustrates the task and the first four stages. The task is performed by a KUKA



(a) stage 1: approach (b) stage 2: surface inspection (c) stage 3: transition (d) stage 4: outer-shell inspection (e) stage 5: departure

Fig. 9. PyBullet execution of `S5SphereInspect` with new start and terminal configurations using the learned stage-wise constraints. The surface- and outer-shell inspection stages follow spherical surfaces at different distances from the sphere.

iiwa 14 equipped with a custom sweeping tool. The conveyor is static during execution. The experiment video is attached to the submission.

We use the seven task features described in Fig. 2.

A human demonstrator provided six kinesthetic demonstrations, from which the stage-wise constraint model is learned and summarized in Fig. 2. Both sweeping stages (2 and 4) have equality constraints maintaining a heading approximately perpendicular to the bar, with a nearly constant lateral offset in stage 2 and axial offset in stage 4. The method also recovered the lower-bound obstacle-clearance constraint and its threshold in stage 1. No feature constraints are activated in stages 3 and 5.

The learned model contains a potentially redundant lower bound on `obs_dist` in stage 4. With the obstacle near one bar end, stage-4 motion from the opposite end approximately perpendicular to the bar produced an incidental one-sided distance distribution resembling a lower-bound constraint. Given only feasible demonstrations, genuine inequalities and incidental one-sided distributions are observationally indistinguishable. With stronger prior knowledge, lowering the activation prior for stage-4 `obs_dist` removed this term without changing the other constraints.

B. Planning and Execution

We reuse the learned stage-wise constraint model in an optimization-based planner. The state $x_t = (p_t, R_t) \in \mathbb{R}^3 \times SO(3)$ is the end-effector pose in the bar frame. The start state, goal state and subgoals are represented in the same frame.

Intermediate subgoals g_k are the average demonstrated stage endpoints, while $g_K = x_{\text{goal}}$ is the specified terminal state. To use the learned equality and inequality constraints in a common objective, we define the violation residual

$$r_{k,j}(x) = \begin{cases} 0, & \widehat{\gamma}_{k,j} = \varnothing, \\ \phi_j(x) - \widehat{\eta}_{k,j}, & \widehat{\gamma}_{k,j} = \text{eq}, \\ [\widehat{\eta}_{k,j} - \phi_j(x)]_+, & \widehat{\gamma}_{k,j} = \text{lb}, \\ [\phi_j(x) - \widehat{\eta}_{k,j}]_+, & \widehat{\gamma}_{k,j} = \text{ub}, \end{cases} \tag{19}$$

where $[z]_+ = \max(z, 0)$.

The planner discretizes each stage according to a fixed spatial waypoint spacing. If stage k contains T_k waypoints, its boundary is $b_k = 1 + \sum_{l=1}^{k} T_l$. B specifies waypoint allocation, not execution duration. Planning is then formulated as

$$\begin{aligned} \min_{\tau=\{x_t\}_{t=1}^{T}} \quad & J_{\text{smooth}}(\tau) + \lambda_c \sum_{k=1}^{K} \sum_{t \in \mathcal{I}_k(B)} \sum_{j=1}^{F} r_{k,j}(x_t)^2 \\ \text{s.t.} \quad & x_1 = x_{\text{start}}, \\ & x_{b_k - 1} = g_k, \qquad k = 1, \ldots, K, \\ & \tau \in \mathcal{C}_{\text{known}}. \end{aligned} \tag{20}$$

Here, J_{smooth} penalizes path length and rapid changes in end-effector position and orientation, and λ_c is a large weight penalizing constraint violation. $\mathcal{C}_{\text{known}}$ specifies the known workspace limit. We solve this nonlinear optimization problem using SciPy.

Sequential inverse kinematics (IK), initialized from the preceding waypoint, converts the optimized task-space path to a consistent joint-space path. The joint path is time-parameterized using the average demonstrated speed of each stage. A standard ROS joint-position trajectory controller executes the trajectory. It sends commands to the robot through the position-control interface at 200 Hz.

C. Results

We evaluate real-robot execution using three task-level criteria: obstacle avoidance, sweeping the objects off the bar, and sweeping the objects off the table. We conducted 20 trials with varied obstacle and bar positions and orientations. The robot passed the obstacle in 18 of 20 trials (90%); both failures involved excessively tilted initial tool poses. It successfully swept the objects off the bar in all 20 trials (100%) and swept the objects off the table in 17 of 20 trials (85%). The failures occurred when too many objects were placed or when the objects could easily roll away on the table.

The learned constraints can also be transferred to modified environments. As shown in Fig. 10, we replace the straight bar with a curved bar and introduce an additional obstacle. The known curved-bar geometry defines the corresponding bar-relative features, while the learned stage-wise constraint model remains unchanged. `obs_dist` is evaluated with respect to the nearest of the two obstacles. The robot successfully clears the bar in this transferred setting, as shown in the video.

VII. Conclusion and Limitations

This paper studied stage-wise constraint learning from feasible demonstrations, where different feature constraints may

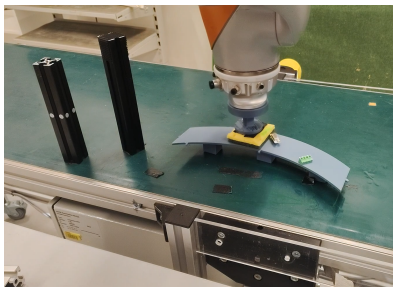


Fig. 10. Transfer setting with a curved bar and an additional obstacle.

become active in different stages and the stage cutpoints are unobserved. We proposed SWCL, which combines constraint likelihoods with a directional subgoal-progress likelihood and uses an alternating hard-EM-like procedure to jointly estimate demonstration-specific cutpoints and a shared constraint model.

Across three simulated benchmarks, SWCL achieved the most consistent joint recovery of stage cutpoints and constraint structures among the compared methods, while producing interpretable constraint modes and parameters reusable for planning. On a real-robot bar-clearing task, the learned constraints enabled the KUKA iiwa to pass the obstacle in 90% of trials, clear the bar in 100%, and clear the table in 85%.

Limitations: The current method assumes that both the task-dependent feature bank and the number of stages are known in advance, requiring relatively strong prior knowledge. This assumption is introduced because jointly inferring the stage boundaries and the underlying constraints from unsegmented demonstrations is highly ambiguous, and restricting the search to a predefined feature space helps reduce the solution space. Future work may combine the proposed framework with representation learning to automatically discover constraint-relevant features from the state space.

REFERENCES

[1] 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.

[2] D. Berenson, S. S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.

[3] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 159–185, 2018.

[4] G. Liu, Y. Luo, A. Gaurav, K. Rezaee, and P. Poupart, "Benchmarking constraint inference in inverse reinforcement learning," in *International Conference on Learning Representations*, 2023.

[5] G. Chou, N. Ozay, and D. Berenson, "Learning constraints from locally-optimal demonstrations under cost function uncertainty," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3682–3690, 2020.

[6] ——, "Learning parametric constraints in high dimensions from demonstrations," in *Proceedings of the Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 100. PMLR, 2020, pp. 1211–1230.

[7] G. Chou, H. Wang, and D. Berenson, "Gaussian process constraint learning for scalable chance-constrained motion planning from demonstrations," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3827–3834, 2022.

[8] D. R. R. Scobee and S. S. Sastry, "Maximum likelihood constraint inference for inverse reinforcement learning," in *International Conference on Learning Representations*, 2020.

[9] S. Malik, U. Anwar, A. Aghasi, and A. Ahmed, "Inverse constrained reinforcement learning," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 7390–7399.

[10] B. Peng and A. Billard, "Positive-unlabeled constraint learning for inferring nonlinear continuous constraints functions from expert demonstrations," *IEEE Robotics and Automation Letters*, vol. 10, no. 2, pp. 1593–1600, 2025.

[11] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[12] C. Truong, L. Oudre, and N. Vayatis, "Selective review of offline change point detection methods," *Signal Processing*, vol. 167, p. 107299, 2020.

[13] T. Kipf, Y. Li, H. Dai, V. Zambaldi, A. Sanchez-Gonzalez, E. Grefenstette, P. Kohli, and P. Battaglia, "CompILE: Compositional imitation learning and execution," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 3418–3428.

[14] G. Sutanto, I. R. Fernández, P. Englert, R. K. Ramachandran, and G. Sukhatme, "Learning equality constraints for motion planning on manifolds," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 155. PMLR, 2021, pp. 2292–2305.

[15] A. L. P. Ureche, K. Umezawa, Y. Nakamura, and A. Billard, "Task parameterization using continuous constraints extracted from human demonstrations," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1458–1471, 2015.

[16] D. Park, M. Noseworthy, R. Paul, S. Roy, and N. Roy, "Inferring task goals and constraints using bayesian nonparametric inverse reinforcement learning," in *Proceedings of the Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 100. PMLR, 2020, pp. 1005–1014.

[17] J. Loula, K. Allen, T. Silver, and J. Tenenbaum, "Learning constraint-based planning models from demonstrations," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 5410–5416.

[18] C. Pérez-D'Arpino and J. A. Shah, "C-LEARN: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2017, pp. 4058–4065.

[19] A. Conkey and T. Hermans, "Learning task constraints from demonstration for hybrid force/position control," in *Proceedings of the IEEE-RAS 19th International Conference on Humanoid Robots*, 2019, pp. 162–169.

[20] C. Hegeler, F. Rozzi, L. Roveda, and K. Haninger, "Teaching contact-rich tasks from visual demonstrations by constraint extraction," in *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*, 2025, pp. 3500–3505.

[21] M. J. Johnson and A. S. Willsky, "Bayesian nonparametric hidden semi-markov models," *Journal of Machine Learning Research*, vol. 14, no. 21, pp. 673–701, 2013.

[22] S. Niekum, S. Osentoski, G. D. Konidaris, S. Chitta, B. Marthi, and A. G. Barto, "Learning grounded finite-state representations from unstructured demonstrations," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 131–157, 2015.

[23] J. R. Medina and A. Billard, "Learning stable task sequences from demonstration with linear parameter varying systems and hidden markov models," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 78. PMLR, 2017, pp. 175–184.

[24] M. Londschien, P. Bühlmann, and S. Kovács, "Random forests for change point detection," *Journal of Machine Learning Research*, vol. 24, no. 216, pp. 1–45, 2023.

[25] F. Yi, H. Wen, and T. Jiang, "Asformer: Transformer for action segmentation," in *Proceedings of the 32nd British Machine Vision Conference (BMVC)*. BMVA Press, 2021, p. 236.

[26] H. Fu, P. Sharma, E. Stengel-Eskin, G. Konidaris, N. Le Roux, M.-A. Côté, and X. Yuan, "Language-guided skill learning with temporal variational inference," in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 235. PMLR, 2024, pp. 14 135–14 156.

[27] R. Gatto, "Saddlepoint approximations to the distribution of the total distance of the multivariate isotropic and von mises–fisher random walks," *Mathematical Methods of Statistics*, vol. 26, no. 1, pp. 20–36, 2017.