S1 Main Contributions

Generative Factor Chaining (GFC) is proposed with the motivation of zero-shot motion planning for long-horizon tasks. The goal is to use short-horizon skill transition distributions and efficiently compose them to structure a long-horizon task-level distribution at inference. The factorized state representation of GFC allows explicit reasoning of inter-object and skill-object interactions and satisfying spatial constraints for coordinate manipulation. The primary contributions of GFC are as follows:

1. A **generalized task representation** to formulate complex long-horizon coordination tasks as a spatial-temporal factor graph of single-arm manipulation skill sequences connected via spatial dependencies.
2. A **compositional framework** to compose short-horizon skill-level transition distributions learned via diffusion models to represent long-horizon task-level distributions.
3. **Easy plug-and-play** via learning skill distributions with skill-level data only and add it to the skill library. Any skill from the library can be plugged as temporal factors in the spatial-temporal factor graph directly at inference for a given long-horizon task.

S2 Additional Related Works

Factor-graph representation for TAMP. The graphical abstraction of a system for understanding several inter-dependencies has been used in various domains [1]. Specifically in context to task and motion planning (TAMP), such a representation allows the decomposition of multiple modalities (discrete and continuous variables) in the state of a system [2]. Solving together for discrete (logical decision variables) and continuous (motion parameters) can be formulated as a Hybrid Constraint Satisfaction Problem (H-CSP) problem, Logic-Geometric Program (LGP) [3], and more recently by advanced gradient descent methods [4]. By following the factor-graph representation, the state space can be represented as a Cartesian product of all the subspaces and the action space can be compactly represented based on the modalities they affect. We particularly follow the *dynamic factor graph* representation used by Garrett et al. [2] to represent all the objects and action parameters as the variable nodes of the graph and all the kinematic inter-dependencies as the factors of the graph.

Optimization for factor graphs. Factor graphs are graphical models where the directed and undirected factors, respectively represent the joint or conditional distribution of the variable nodes connected to them. Most directed factors graphs as used for localization [5, 6] are formulated into probabilistic graphical models of hidden-markov chains and solved for the maximum a posteriori (MAP) [5, 7] estimates of the unknown node variables. Particularly in motion planning, optimizing for all the variable nodes is often formulated as a constraint satisfaction problem [2, 8].

Additional related works on learning for TAMP. Recent works have shown that a number of components of a TAMP system benefit from powerful generative models. Wang et al [9, 10] use Gaussian Processes to learn continuous-space sampler for TAMP. Similarly, Kim et al. [11] use GANs to learn action samplers. Fang et al. [12] propose to use Diffusion Models to capture complex distributions such as Inverse Kinematics solutions, grasps, and contact dynamics. However, they still rely on an overarching TAMP system to consume the generated samples to perform planning. In contrast, our method directly forms a geometric plan sampler by chaining together factor-level diffusion models.

S3 Additional mathematical details of Generative Factor Chaining (GFC)

In this section, we provide additional detail on the mathematical formulation of the spatial-temporal probabilistic graphical model using spatial (scene) and temporal (skill) factors. First, we explain each of them individually and then discuss their composition to formulate GFC.

Spatial probabilistic graphical model. Consider an arbitrary state, s of the system based on our representation of an *undirected* factor graph, $\{\mathcal{V}, \mathcal{F}\}$. It consists of nodes ($v \in \mathcal{V}$) as the objects and robots in the scene, and certain spatial factors ($f \in \mathcal{F}$) signifying the inter-dependencies between the nodes. When we construct a probabilistic graphical model from this representation, an intuitive way of calculating the distribution of a state, $p(s)$, is the composition of all the factor distributions. Mathematically:

$$p(s) \propto \prod_{f \in \mathcal{F}} p_f(\mathcal{S}_f) \tag{S1}$$

where $p_f(\mathcal{S}_f)$ represents the joint factor potential of nodes $v \in \mathcal{S}_f \subseteq \mathcal{V}$, i.e. all nodes involved in a factor. This indicates that the joint distribution of all the nodes must satisfy each of the factors, also explored by Diffusion-CCSP [8].

Temporal probabilistic graphical model. Let us consider an arbitrary transition of an arbitrary state s_0 to s_1 after executing a skill π_0 with action parameter a_0. Based on our definition of skill as a temporal factor, the factor distribution for π_0 will be $p_{\pi_0}(s_0, a_0, s_1)$, which models the joint distribution of nodes s_0, a_0, s_1 such that the transition dynamics is satisfied. We are particularly interested in the joint distribution of a trajectory factor graph made by a temporal composition of π_0 with subsequent skills (factors) π_k in a skill skeleton $\Phi = \{\pi_0, \pi_1, \dots, \pi_K\}$, executed on s_k with action parameters a_k and leads to transitioned state s_{k+1}. The joint distribution of trajectory $\tau = \{s_0, a_0, s_1, \dots, s_K, a_K, s_{K+1}\}$ can be given similar to Equation S1 as:

$$p(\tau) \propto \prod_{\pi_k \in \Phi} p_{\pi_k}(s_k, a_k, s_{k+1}) \tag{S2}$$

However, there is a concern: While spatial factors can be composed in parallel with no conditional dependency, temporal factors have an inherent conditional dependency because we must know s_1 to execute π_1. Hence, unlike the *undirected* state factor graph, the factor graph in this case is a *directed* factor graph. Now we will explore this additional dependency with a fairly simple trajectory $\tau = \{s_0, a_0, s_1, a_1, s_2\}$.

Understanding forward-backward temporal dependency. Our goal is to construct the joint distribution of all the nodes by composing all the spatial-temporal factors in the plan. Since skill factors have a temporal dependency, we have to compensate accordingly. We follow GSC [13] and write the joint distribution of the nodes in the trajectory based on forward and backward analysis as follows:

$$p(\tau|s_0) \propto p_{\pi_0}(s_0, a_0, s_1) p_{\pi_1}(a_1, s_2|s_1) \quad \text{and} \quad p(\tau|s_2) \propto p_{\pi_0}(s_0, a_0|s_1) p_{\pi_1}(s_1, a_1, s_2)$$
$$p(\tau|s_0) \propto \frac{p_{\pi_0}(s_0, a_0, s_1) p_{\pi_1}(s_1, a_1, s_2)}{p_{\pi_1}(s_1)} \quad \text{and} \quad p(\tau|s_2) \propto \frac{p_{\pi_0}(s_0, a_0, s_1) p_{\pi_1}(s_1, a_1, s_2)}{p_{\pi_0}(s_1)}$$

This signifies that for every intermediate state connected by two temporal factors, the joint distribution of the sequence has an additional term in the denominator calculated as:

$$p(\tau|s_0, s_2) \propto \frac{p_{\pi_0}(s_0, a_0, s_1) p_{\pi_1}(s_1, a_1, s_2)}{\sqrt{p_{\pi_0}(s_1) p_{\pi_1}(s_1)}} \tag{S3}$$

Probabilistic model for spatial-temporal factor graphs. Now, we again consider the spatial graph for representing the state, where the probability of finding a state s is the joint distribution of all the nodes in the factor graph. We will now integrate the spatial factors with the temporal factors considering the compensation term introduced in Equation S3. Thus, we will not consider the state as one entity but a collection of nodes. Accordingly, Equation S3 applies for any intermediate node connected by two temporal factors. If we consider the set of intermediate nodes $\mathcal{V}_i$ connected by two skills π_{i-} and π_{i+}, we can rewrite Equation S2 as:

$$p(\tau) \propto \frac{\prod_{\pi_k \in \Phi} p_{\pi_k}(v_k \in \mathcal{V}_{pre}^{\pi_k}, a_k, v_{k+1} \in \mathcal{V}_{effect}^{\pi_k}) \prod_{k=0}^{K} \prod_{f \in \mathcal{F}_k} p_f(\mathcal{S}_f)}{\sqrt{\prod_{v_i \in \mathcal{V}_i} p_{\pi_{i-}}(v_i) p_{\pi_{i+}}(v_i)}} \tag{S4}$$

This completes the joint distribution of all the nodes in the spatial-temporal factor graph plan considering the temporal factors for all skills with their pre-condition and effect nodes, all spatial factors for all states in the plan, and all intermediate nodes in the temporal chain. We show our implementation of this formulation in algorithm 1.

Example of spatial factors. Previous work [8] considered a family of spatial factors like (`left`, `right`, `top`, `bottom`, `near` and `far`) to model collision-free object configurations. In this work, we are particularly interested in constructing a family of fixed transforms (`FixedTransform`) to model coordinated manipulation motion. For example, in order to satisfy the pre-condition of `strike(A, B)`, the transform between nodes A and B must satisfy a family of transforms signifying that B must be `Aligned` with A to strike it. Thus the factor for `strike(A, B)` with `Aligned` transforms $\mathcal{H}_A$ will look like: $f \equiv \text{distance}(\text{transform}(A, B), \mathcal{H}_A) \leq$ permisible error for at least one transform. In that case, the distribution of the factor will be: $p(f = \texttt{True}|A, B) \propto \exp[-\text{distance}(\text{transform}(A, B), \mathcal{H}_A)]$. The score of such a distribution can then be calculated as

$$\epsilon_f(A^{(t)}, B^{(t)}, t) = -\nabla_{A^{(t)}, B^{(t)}} \text{distance}(\text{transform}(A^{(t)}, B^{(t)}), h_A)$$

where $h_A \in \mathcal{H}_A$ is the closest transform to the current transform. The distance between transforms is calculated as the summation of the Cartesian distance and the quaternion distance.

Summary GFC is a new paradigm to solve complex manipulation problems using spatial-temporal factor graphs. GFC can be divided into the following segments: (1) train individual skill factor distributions individually, without any prior knowledge or data from other skills in the library (2) create spatial-temporal factor graph from a plan skeleton, (3) compose individual spatial and temporal factor distributions to construct a probabilistic graphical model, and (4) use the plan-level distribution to sample plan solutions. The proposed approach is modular as the individual skill factors and constraints can be flexibly connected to form new graphs. GFC can connect parallel skill chains with added spatial factors to solve coordinated manipulation problems directly at inference. Additional detail in algorithm algorithm 1.

S4 Generative Factor Chaining in practice: An illustrative explanation

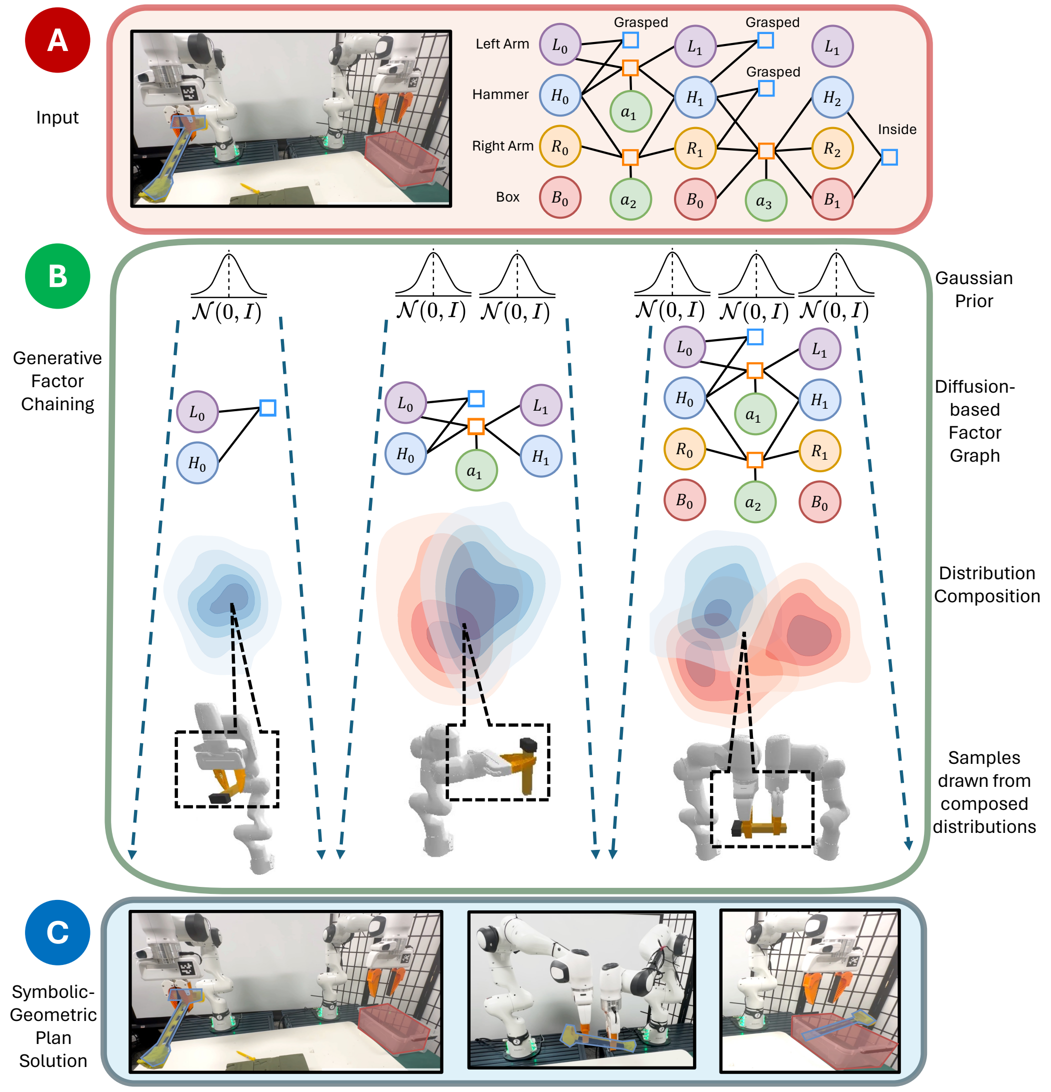
A
Input
Left Arm
Hammer
Right Arm
Box
Grasped
Grasped
Grasped
Inside
L_0 L_1 L_1 H_0 H_1 H_2 R_0 R_1 R_2 B_0 B_0 B_1 a_1 a_2 a_3
B
Generative Factor Chaining
$\mathcal{N}(0, I)$
Gaussian Prior
Diffusion-based Factor Graph
Distribution Composition
Samples drawn from composed distributions
C
Symbolic-Geometric Plan Solution

Figure S1: **Detailed methodology.** Above figure illustrates the complete pipeline to solve a long-horizon coordinated manipulation problem with our proposed method. **Task:** The task objective is to place the hammer inside the box. However, since the left arm cannot reach the box, the hammer is handed over to the right arm such that the right arm can complete the task. **(a) Inputs:** The initial scene and a symbolically feasible spatial-temporal factor graph plan to complete the goal objective. **(b) GFC:** We formulate all factors as distributions of the nodes connected to them. GFC represents spatial factors as classifiers and temporal factors as diffusion models. We leverage compositionality of diffusion models to compose spatial-temporal distributions using Equation S4 to find the joint distribution of the complete plan directly at inference. Finally, samples drawn from such a joint distribution are symbolically and geometrically feasible solutions of the whole plan. **(c) Output:** A sequence of skill choices and optimizer continuous parameters executed on robots with parameterized skill controllers.

Algorithm 1: Generative Factor Chaining (GFC) Algorithm

```
1  Hyperparameters:
2  Number of reverse diffusion steps $T$

3  Inputs:
4  Pre-defined skill library $\Pi = \{\pi_1, \pi_2, \dots, \pi_M\}$
5  Individual skill diffusion score functions $\epsilon_\pi$
6  Task skeleton $\Phi_K = \{\pi_0, \pi_1, \dots, \pi_K\}$: a sequence of skills of length $K$
7  Scene graph sequence $\Phi_S = \{s_0, s_1, \dots, s_K\}$: a sequence of scene factors of length $K + 1$ where $s_k \equiv \{\mathcal{V}_k, \mathcal{F}_k\}$
8  Goal condition $g \equiv \{\mathcal{V}_g, \mathcal{F}_g\}$
9  Noise schedule $\sigma$

10 Initialize $t = T = 1$
11 Initialize $\Delta t$
12 Initial node sequence $\mathbf{x}^{(T)} = \Big[v_k^{(T)}\ \forall\ v \in \mathcal{V}_k,\ \ a_{\pi_k}^{(T)}, \dots \forall\ k \in [0, K]\Big]$ sampled from $\mathcal{N}(\mathbf{0}, \sigma_T \mathbf{I})$

13 while $t \geq 0$ do
14     // Score of the joint distribution of all the nodes
15     $\epsilon_\Phi(v_k^{(t)}\ \forall\ v \in \mathcal{V}_k,\ \ a_{\pi_k}^{(t)}, \dots \forall\ k \in [0, K], t) = \mathbf{0}$
16     // Calculating the effective score of each node
17     $\epsilon_\Phi(x^{(t)}, t) = \sum_{k=0}^{K} \epsilon_{\pi_k}(x^{(t)}, t) + \sum_{k=0}^{K} \sum_{f \in \mathcal{F}_k} \epsilon_f(x^{(t)}, t) \quad \forall x \in \mathbf{x}$   (Computational assumption, Equation S4)
18     // Only for nodes connected with two temporal factors $f_{x,1}$ and $f_{x,2}$
19     $\epsilon_\Phi(x^{(t)}, t) =$
           $\epsilon_\Phi(x^{(t)}, t) - \frac{1}{2}\Big[\epsilon_{f_{x,1}}(x^{(t)}, t) + \epsilon_{f_{x,2}}(x^{(t)}, t)\Big]$   (Denominator compensation, Equation S4)
20     // calculating updated noised samples for the next reverse diffusion timestep
21     $\tilde{\mathbf{x}}^{(t-1)} = \mathbf{x}^{(t)} + \dot{\sigma}_t \sigma_t \epsilon_\Phi(v_k^{(t)}\ \forall\ v \in \mathcal{V}_k,\ \ a_{\pi_k}^{(t)}, \dots \forall\ k \in [0, K], t)\Delta t$
22     $t = t - \Delta t$
23 end
24 Return $\mathbf{x}^{(0)}$
```

S5 Skill Data Collection and Skill Training

We consider a finite set of parameterized skills in our skill library. While our framework supports flexible addition of new skills to the skill library, we choose skills appropriate for the considered tasks. The parameterization, data collection, and training method for each of the skills is described as follows:

1. `Pick`: Gripper picks up an object from the table and the parameters contain 6-DoF pose in the object's frame of reference. The skill diffusion models are trained on successful pick actions on all the available set of objects namely lid, cube, hammer, and nail/stake.

2. `Place`: Gripper places an object at the target location and parameters contain 6-DoF pose in the place target's frame of reference. This skill requires specifying two set of parameters, the target pose and the target object (e.g. box, table). The picked object is placed and successful placements are used to train the skill diffusion model.

3. Move: Gripper reaches a target location with an object in hand and parameters contain 6-DoF pose in the manipulator's frame of reference within the workspace. This skill captures the distribution of the reachable workspace of the robot. When composed with the Move skill of the second manipulator, the combined distribution captures the common workspace.

4. ReGrasp: Gripper grasps object mid-air and the parameters contain 6-DoF pose in the object's frame of reference. While collecting data directly for this skill is non-trivial, we consider that if an object is picked up with parameters q_1 and moved with parameters q_2, then the object can be grasped at the workspace location defined by q_2 with the ReGrasp parameters as q_1. Thus, we reuse Pick and Move data to train the skill diffusion model for ReGrasp. While this is a design choice, with appropriate skill level data, we can train this skill separately too.

5. Push: Gripper uses the grasped object to push away another object. The skill is motivated from prior work [13, 14] where a hook object is used to Push blocks. The parameters of this skill are (x, y, r, θ) such that the hook is placed at the (x, y) position on the table and pushed by a distance r in the radial direction θ w.r.t. the origin of the manipulator. The skill diffusion models is trained following GSC [13].

6. Pull: Gripper uses the grasped object to pull another object inwards. The skill is also motivated from prior work [13, 14] where a hook object is used to Pull blocks. The parameters of this skill are (x, y, r, θ) such that the hook is placed at the (x, y) position on the table and pulled by a distance r in the radial direction θ w.r.t. the origin of the manipulator. The skill diffusion models is trained following GSC [13].

7. Strike: Gripper strikes another object with one object in hand (e.g., a hammer). As a design choice, we do not train a skill diffusion model for this skill. Strike is primarily used as a terminal skill. We are only concerned about the pre-condition as their effects can be designed manually, which is similar to "subgoal skill" used in prior work. For example, in order to satisfy the pre-condition of Strike, the hammer and nail must be aligned. This can be satisfied in diverse configurations. However, the effect is achieved through a deterministic motion.

8. Pour: Gripper rotates the object in hand in a pouring fashion. Similar to Strike, we use Pour as a terminal skill too. In order to satisfy the pre-condition of Pour, the transform between the source and target mug must belong to the family of admissible distributions. We achieve the actual trajectory by designing a deterministic motion. With appropriate skill level data, we can also train skill diffusion models, however, such improvement is out of scope of this work.

S6 Model Training and Architecture

Model architecture. Our transformer-based score-network architecture is derived from the Diffusion Models with Transformers (DiT) [15] implementation, also open-sourced at: https://github.com/facebookresearch/DiT. We follow a similar concept to that of patchifying an image into many smaller patches, encoding each one of them using a common encoder and passing it as a sequence to the transformer architecture with respective positional embeddings. In our case, we consider a sequence of nodes consisting of both the object and skill parameters nodes in the factor graph as the input sequence. Each node variable is encoded into a common dimension using a common object node encoder and skill parameter encoder for object and skill parameter nodes respectively. The output is decoded into their respective dimensions using similar decoder setup.

Training. We train individual skill diffusion score-functions using the denoising score-matching (DSM) loss following algorithm 2. We collect datasets of transitions observed during the execution of a skill on an object and use them to train the score networks. The dataset size varies according to the difficulty and diversity of a skill's execution on a particular object. For example, we

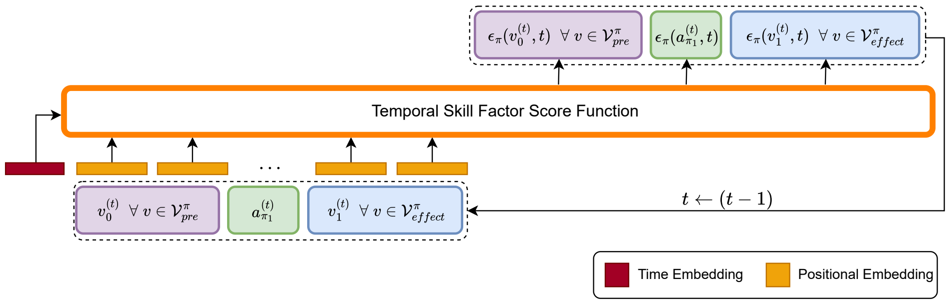
$\epsilon_\pi(v_0^{(t)}, t)\ \forall\, v \in \mathcal{V}^\pi_{pre}$
$\epsilon_\pi(a^{(t)}_{\pi_1}, t)$
$\epsilon_\pi(v_1^{(t)}, t)\ \forall\, v \in \mathcal{V}^\pi_{effect}$
Temporal Skill Factor Score Function
. . .
$v_0^{(t)}\ \forall\, v \in \mathcal{V}^\pi_{pre}$
$a^{(t)}_{\pi_1}$
$v_1^{(t)}\ \forall\, v \in \mathcal{V}^\pi_{effect}$
$t \leftarrow (t-1)$
Time Embedding
Positional Embedding

Figure S2: Transformer-based skill diffusion model. We use the noisy pre-condition, action and effect node value distribution at diffusion step t to obtain the corresponding ϵ during sampling.

need 100 successful `Pick` parameters for training the skill to pick the hammer and 300 successful `Move` parameters to cover the whole workspace of the robot. For `ReGrasp`, we use both the `Pick` and `Move` parameters.

Effect of training data coverage. If we consider "ideal" score functions and a perfect representation of the factor distributions, a solution exists if there is an overlap between two connected factor distributions. If such an overlapping segment does not exist, GFC will not be able to complete the spatial-temporal plan. Hence, the training data for each factor (here temporal factors only) must be diverse enough to ensure that the overlap exists. For example, a successful handover in *Hammer Place* and *Hammer Strike* is not possible if the training data only consists of `Pick` parameters to pick the hammer from the center of the handle. Similarly, if the training data for `Move` does not cover the common workspace of both robots, our proposed algorithm will be unable to complete the coordinated plan.

Algorithm 2: Training skill score functions for a particular skill π

```
Inputs:
Pre-condition, skill parameter and Effect nodes $(\mathcal{V}^\pi_{pre}, a_\pi, \mathcal{V}^\pi_{effect})$
Dataset of transitions $\mathcal{D}$
Parameterized skill score function $\epsilon_\phi$
Noise schedule $\sigma$
DSM loss weight schedule $\lambda$

while not converged do
    Sample batch from dataset $\mathbf{x}^{(0)} \sim \mathcal{D}$
    Sample forward diffusion timestep $t \sim [0, 1]$
    Sample Gaussian noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
    Calculate noise coefficient $\sigma_t$
    Calculate noisy data $\mathbf{x}^{(t)} = \mathbf{x}^{(0)} + \sigma_t \epsilon$
end

Optimize parameters $\phi$ using:
        $\nabla_\phi \mathbb{E}_{t,\epsilon,\mathbf{x}^{(0)}} [\lambda(t) \|\epsilon - \epsilon_\phi(\mathbf{x}^{(t)}, t)\|^2]$

Return $\epsilon_\pi \equiv$ (Optimized) $\epsilon_\phi$
```

Hyperparameters and computation. We consider the hyperparameters as shown in Table S1 for building our score-network.

For the reverse sampling steps while inference, we find the best performance using 50 steps and all results have been reported accordingly. Considering skill-object score functions with varying input nodes leads to a loss of parallel batched inference (advantage of vectorized states) and hence, an increase in computation time as compared to chaining with vectorized states. On an NVIDIA

Table S1: Hyperparameters for Score-Network with Transformer Backbone

Hyper-parameter	Value
Hidden Dimension	128
Number of Blocks	2
Number of Heads	2
MLP Ratio	2
Dropout Probability	0.1
Number of Input Channels	Varies (3-11)
Number of Output Channels	Varies (3-11)

RTX$^{\text{TM}}$ A6000 GPU, it takes 2.6 secs for the smallest horizon task *Pour Cup* and 6 secs for the longest horizon task *Hammer Nail* to give 10 candidate node variable values. These candidates are sorted based on their extent of goal-condition satisfaction and the top 5 are selected to calculate the success performance.

S7 Real Robot Experiments

Complete setup. We use two Franka Panda robot arms placed in parallel to demonstrate the coordinated tasks as illustrated in Figure S3. A pair of flexible Finray fingers [16] is attached to the parallel jaw grippers. For each of the arm, we set up a Kinect Azure camera calibrated to the origin of the arm. We use objects like mallet (hammer), stake (tent peg, nail), garden foam, a kitchen pot, two types of mugs and a rack for the considered tasks. We use segment-anything [17] and CLIP [18] to segment the objects from the RGBD image based on text descriptions and use the segmented masks to obtain the point clouds for the objects. Finally, we use ICP to align the obtained and model point clouds to calculate the transformation of the object. The procedure is done for both cameras to obtain transforms for all the detected objects in both robot's frame of reference. For a particular object, we select the transform from the arm closest to the object to get precise pose estimation (due to better depth data). We finally use the obtained transforms to recreate the physical scene in simulation, employ GFC in simulation and rollout the results in the real-world. While planning, the Frankx controller [19] is used to generate smooth motion toward the desired pose.

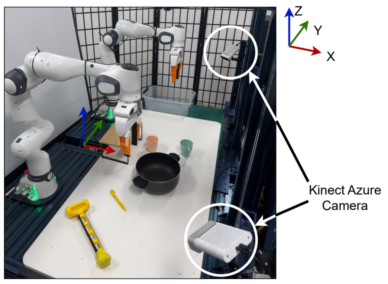

Figure S3: Real-World Experimental Setup

Qualitative analysis. We perform qualitative analysis for all four coordinated tasks using the hardware setup as shown in Figure S4,Figure S5, Figure S6 and Figure S7. We further provide detailed videos of execution in the supplementary video.

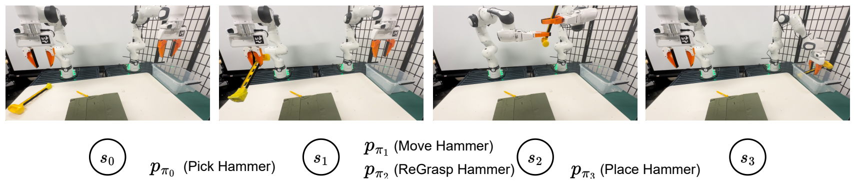
s_0
p_{π_0} (Pick Hammer)
s_1
p_{π_1} (Move Hammer)
p_{π_2} (ReGrasp Hammer)
s_2
p_{π_3} (Place Hammer)
s_3

Figure S4: **Coordination task: *Hammer Place*** The left arm must handover the hammer to the right arm such that the hammer can be placed inside the box.

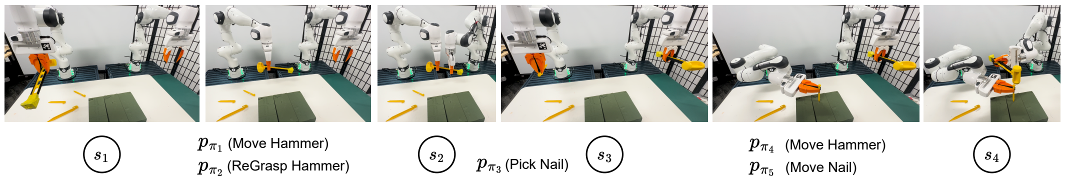
s_1
p_{π_1} (Move Hammer)
p_{π_2} (ReGrasp Hammer)
s_2
p_{π_3} (Pick Nail)
s_3
p_{π_4} (Move Hammer)
p_{π_5} (Move Nail)
s_4

Figure S5: **Coordination task: *Hammer Nail*** The left arm must handover the hammer to the right arm and pick up the nail. Both arms have to coordinate in order to move the hammer and nail to a configuration in which the hammer can strike the nail.

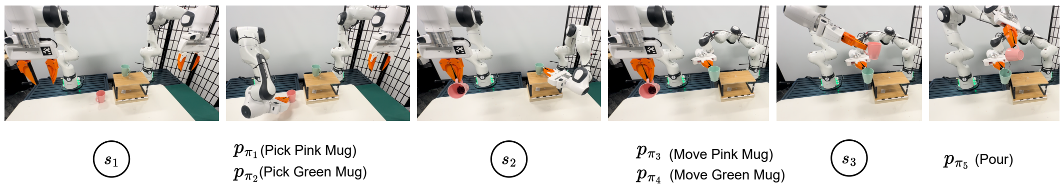
s_1
p_{π_1} (Pick Pink Mug)
p_{π_2} (Pick Green Mug)
s_2
p_{π_3} (Move Pink Mug)
p_{π_4} (Move Green Mug)
s_3
p_{π_5} (Pour)

Figure S6: **Coordination task: *Pour Cup*** The left arm and right arm must pick up the pink mug and green mug respectively. Both arms have to coordinate in order to move the mugs to a configuration in which the left arm can pour the pink mug contents into the green mug.

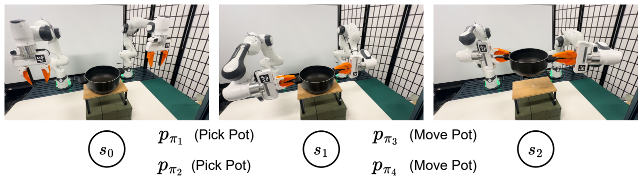

Figure S7: **Coordination task: *Bimanual Reorientation*** The left arm and right arm must pick up the pot simultaneously. Both arms have to coordinate in order to rotate the pot to a specified target reorientation angle. For the above illustration, the reorientation angle is 30deg.

Failure analysis. We try to analyze the reason for the failure of GFC in certain cases. A limiting factor of our planning framework is that the nodes denote waypoints required to be reached for completing the geometric execution and satisfying the goal condition without caring about the trajectory between them. Since we do not explicitly provide the intuition of inverse kinematics (IK) or collision, we assume that these properties are learned implicitly using the successful transitions in the training data. Hence, apart from sim-to-real gap (consisting of pose-estimation error, nature of surfaces in contact, and weight of the objects like hammer and pot), the primary reasons for failure are: (1) sampling a pose where IK cannot be computed, i.e. unreachable. (2) The sampled pose is not collision-free. We provide sim-to-real gap failures in the supplementary video.

S8 More Details on Evaluation Tasks

S8.1 Hammer Nail

Task Description: Given a scene with three boxes, a hammer in placed in one of the box covered by a lid as shown in Figure S8. There is a nail on the table. Only left arm can reach the lid, hammer and the nail. The task objective is to strike the nail by the hammer within a provided region. There is a cube in one of the boxes, picking and placing it are task-irrelevant distractions.

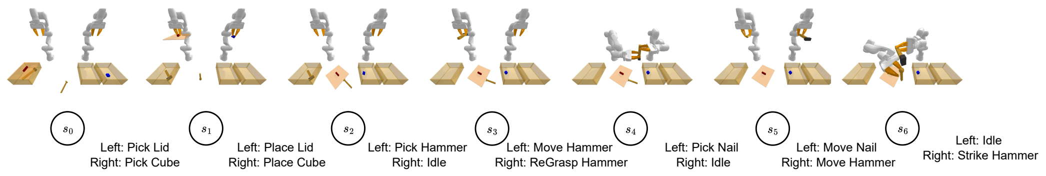

Figure S8: **Hammer Nail.** The illustration shows the *Hammer Nail* task. A successful solution to this task must complete a successful handover and coordinate to align the hammer and the nail to conduct a successful strike.

What it takes to solve? From a superficial symbolic analysis, the task can be completed if the left arm can handover the hammer to the right arm, left arm can pick up the nail to take it to the admissible region and the right arm can strike the nail by the hammer. However, the following challenges exist:

1. Hammer must be picked up and moved at a location such that the right arm can re-grasp it for a successful handover.
2. The handover must allow the right arm to satisfy the pre-condition of strike i.e. the right arm must grasp the hammer away from the head, hence the left arm must reason and pick it up by grasping close to head.
3. The re-grasp pose will affect the region where the hammer head can be reached. The left arm must reason about the hammer head's reachability to move the nail such that the hammer and nail can be aligned.

Why is this challenging? All the above reasonings are interdependent and the effect of the initial pick pose can be seen at multiple stages of the task. This makes the task challenging as the plans fails:

1. if the initial pick pose fails to reason about handover requirements.
2. if the nail move target pose fails to satisfy the reachability of the hammer-head, which actually depends on the handover.

Failure cases: The failures in the proposed method occur in the following situations:

1. *Method failure:* when it predicts in-feasible poses (where IK cannot be computed) or which does not satisfy the pre-condition of the next skill.
2. *Trajectory planning failure:* If IK can be computed for current and target poses but no collision-free trajectory can be computed (via pybullet-planning cite). This is expected as GFC only solves for high-level skill transitions.
3. *Simulation failure:* While executing `Pick` skill, sometimes the contact vectors are noisy and hence leads to pick-up failures.

S8.2 Bimanual Pot Reorientation

Task Description: Given a pot on a table, the task is to reorient the pot to some target orientation angle (along z-axis) using two manipulators as shown in Figure S10. It is worth noting that we have

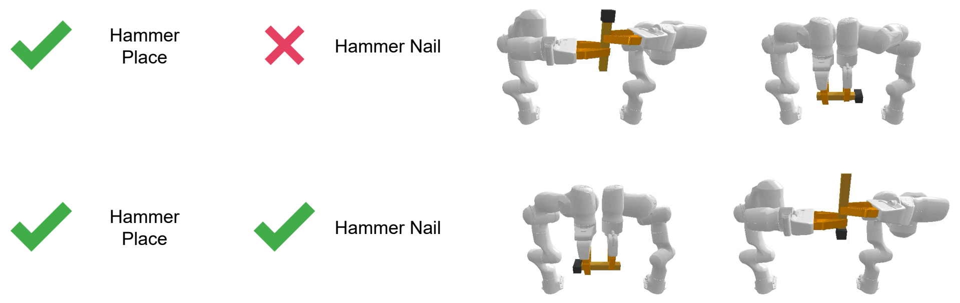

Figure S9: **Handover variations.** The hammer handover can be done in multiple ways, four of which are shown above. While placement of the hammer in the box for *Hammer Place* task can be done by re-grasping the hammer anywhere, for hammer strike in *Hammer Nail*, the hammer is encouraged to be regrasped near the tail of the handle.

`Pick` and `Move` skills for individual manipulators such that we know where the pot can be grasped and the reachable workspace of the manipulator.

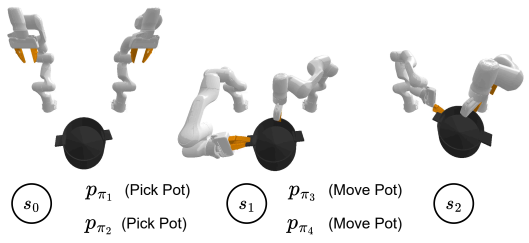

Figure S10: **Bimanual Pot Reorientation.** The task is to coordinate planning strategies to grasp a pot using two manipulators and rotate it to a target reorientation angle. The task must be done with only single-manipulator data.

What it takes to solve? This particular task can be completed if:

1. we find pick poses for both the manipulators.
2. we find feasible move poses in the workspace that satisfies the target orientation.
3. we ensure that the relative transform between two gripper poses while picking and in the predicted move target poses is the same, because the grasp poses relative to the pot cannot change while moving.

Why is this challenging? The task is challenging because the algorithm must decide the initial pick pose by considering sequential and parallel dependencies:

1. the same pick pose relative to the pot must exist for the target reorientation angle
2. the move pose for both manipulators must satisfy both the workspace reachability for individual manipulators and also have the same fixed transform as the pick poses.

Failure cases: The failures in the proposed method occur in the following situations:

1. *Method failure:* when it predicts in-feasible poses (where IK cannot be computed) or which does not satisfy the fixed transform condition.
2. *Trajectory planning failure:* If IK can be computed for current and target poses but no collision-free trajectory can be computed (via pybullet-planning cite). This is expected as GFC only solves for high-level skill transitions.
3. *Simulation failure:* While executing `Pick` skill, sometimes the contact vectors are noisy and hence lead to pick-up failures.

S9 Extending Hammer Nail task to longer horizons

In order to evaluate the extensive long-horizon planning capabilities of our proposed algorithm, we have further extended the Hammer Nail task to longer horizons as shown in Figure S11. The extended tasks particularly emphasize adding a second handover such that the hammer is handed back to the left arm after a successful hammer strike.

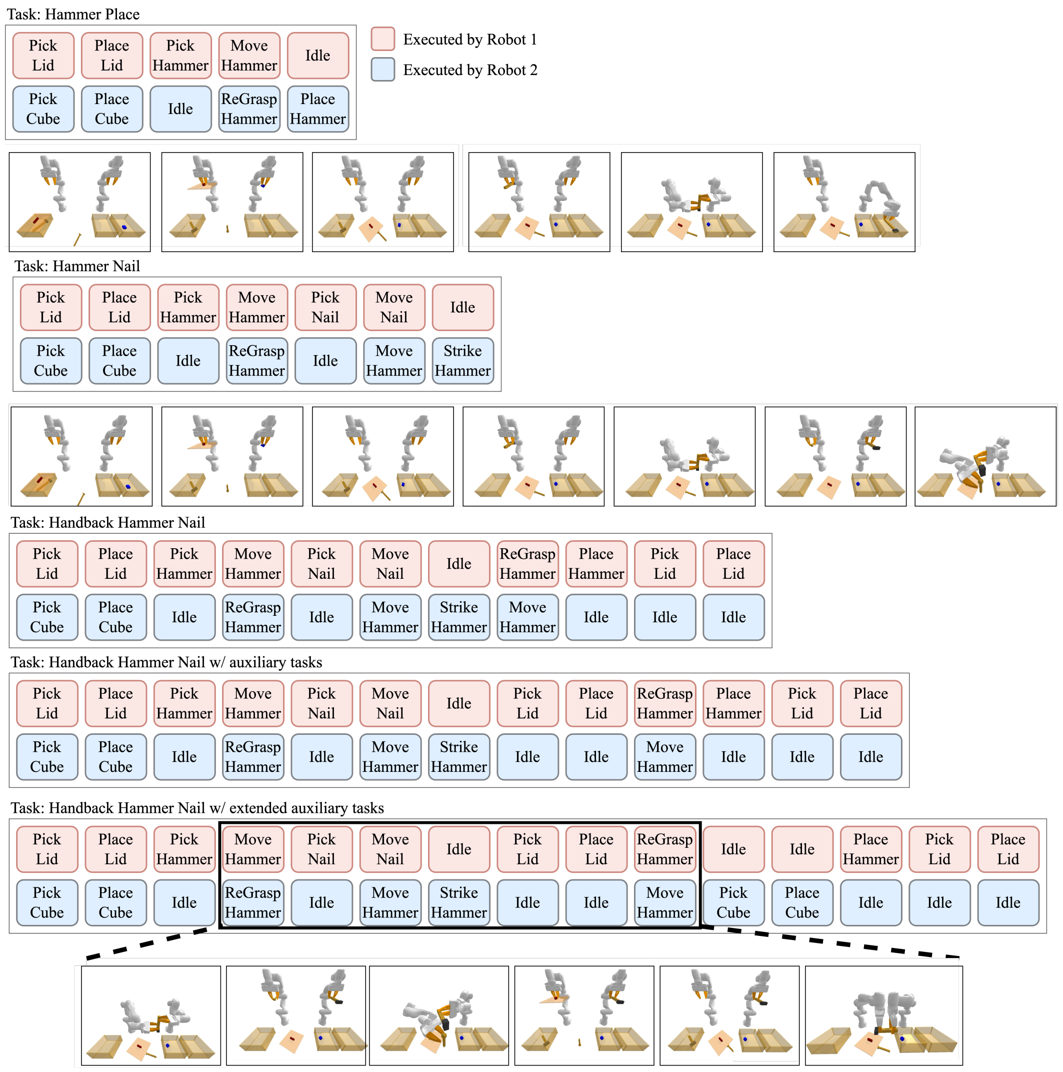
Task: Hammer Place
Pick Lid | Place Lid | Pick Hammer | Move Hammer | Idle
Pick Cube | Place Cube | Idle | ReGrasp Hammer | Place Hammer
Executed by Robot 1
Executed by Robot 2
Task: Hammer Nail
Pick Lid | Place Lid | Pick Hammer | Move Hammer | Pick Nail | Move Nail | Idle
Pick Cube | Place Cube | Idle | ReGrasp Hammer | Idle | Move Hammer | Strike Hammer
Task: Handback Hammer Nail
Pick Lid | Place Lid | Pick Hammer | Move Hammer | Pick Nail | Move Nail | Idle | ReGrasp Hammer | Place Hammer | Pick Lid | Place Lid
Pick Cube | Place Cube | Idle | ReGrasp Hammer | Idle | Move Hammer | Strike Hammer | Move Hammer | Idle | Idle | Idle
Task: Handback Hammer Nail w/ auxiliary tasks
Pick Lid | Place Lid | Pick Hammer | Move Hammer | Pick Nail | Move Nail | Idle | Pick Lid | Place Lid | ReGrasp Hammer | Place Hammer | Pick Lid | Place Lid
Pick Cube | Place Cube | Idle | ReGrasp Hammer | Idle | Move Hammer | Strike Hammer | Idle | Idle | Move Hammer | Idle | Idle | Idle
Task: Handback Hammer Nail w/ extended auxiliary tasks
Pick Lid | Place Lid | Pick Hammer | Move Hammer | Pick Nail | Move Nail | Idle | Pick Lid | Place Lid | ReGrasp Hammer | Idle | Idle | Place Hammer | Pick Lid | Place Lid
Pick Cube | Place Cube | Idle | ReGrasp Hammer | Idle | Move Hammer | Strike Hammer | Idle | Idle | Move Hammer | Pick Cube | Place Cube | Idle | Idle | Idle

Figure S11: **Extension of Hammer Nail task.** We have added three new extensions to the *Hammer Nail* task. All of the new tasks focus on handling a second handover. The nature of the first handover adds further constraints into possible ways to perform the second handover. Further, we add task-irrelevant skills in between the plan skeleton to evaluate the robustness of GFC and the spatial-temporal factor graph plan representation.

We classify the failure cases as:

- Type 1: Method failure i.e. when the proposed algorithm fails to find suitable target parameters.
- Type 2: Trajectory planning failure i.e. no collision-free trajectory can be computed between two suitable poses.
- Type 3: Simulation failure i.e. when simulator fails to detect suitable contacts.

Now, we show the failure breakdown and task success for all the considered *Hammer Nail* task and their extensions in Table S2. While we see a drop in success rates by adding a second handover to the vanilla *Hammer Nail* task, GFC proved to be robust for all other task-irrelevant skills in the chain. The task success of all "two handover" variants is similar even with an increasing task horizon.

Table S2: Failure breakdown and task success analysis of hammer nail task and its extensions with two handovers (based on 100 trials)

Task	Task Horizon	Type 1 failure	Type 2 failure	Type 3 failure	Task Success
Hammer Nail	11	42	14	10	**34**
Extended Hammer Nail v1	16	43	28	5	**24**
Extended Hammer Nail v2	18	44	21	10	**25**
Extended Hammer Nail v3	20	41	25	13	**21**

S10 Analyzing Inter-step dependencies

Our work focuses on solving long-horizon tasks that have strong inter-step dependencies [20] and requirements for coordinated manipulation [21, 22, 23]. For example, hammering a nail not only requires extensive affordance planning to perform a handover but also requires allowing sufficient reachable workspace to align the hammer head with the nail. This also affects the success of the second handover, thus increasing the action-dependency horizon. Our framework is able to compose learned factors (diffusion models) to solve a wide variety of tasks, as long as their solutions fall in the combinatorial space.

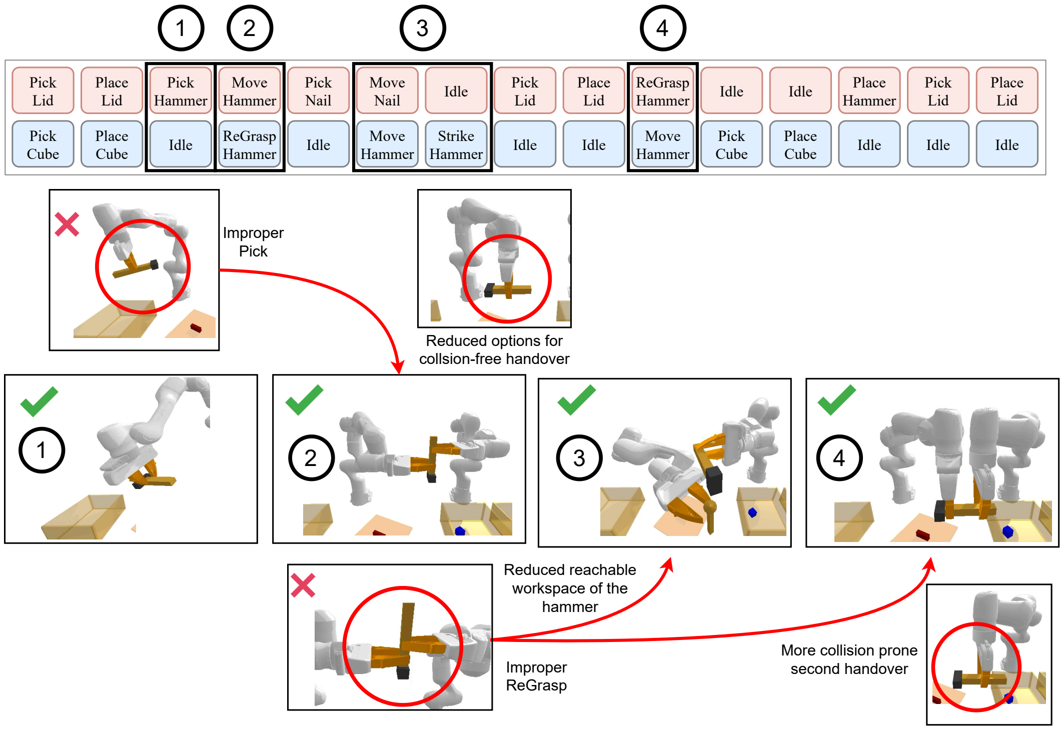

Figure S12: **Inter-step dependencies.** We show the steps and reasoning required to solve the *Hammer Nail* task. An improper initial pick can lead to a failed or unfavorable handover which might lead to difficulty in performing `Strike` and the second handover. Thus the algorithm must reason about inter-step action dependency over longer horizons to solve the task successfully.

S11 Justifying success rates with breakdowns

We elaborate on the failure and success breakdown for the vanilla *Hammer Nail* task in Table S3. Revisiting the failure categories, we classify the failure cases as:

- Type 1: Method failure i.e. when the proposed algorithm fails to find suitable target parameters.
- Type 2: Trajectory planning failure i.e. no collision-free trajectory can be computed between two suitable poses.
- Type 3: Simulation failure i.e. when simulator fails to detect suitable contacts.

Table S3: Failure breakdown and task success analysis per skill-step of hammer nail task (based on 100 trials)

Skill.No.	Skills	Type 1 failure	Type 2 failure	Type 3 failure	Accu. Success
1	Pick Lid	5	0	0	**95**
2	Place Lid	0	0	0	**95**
3	Pick Cube	0	0	0	**95**
4	Place Cube	6	0	0	**89**
5	Pick Hammer	3	0	2	**84**
6-7	Move Hammer - Regrasp Hammer	8	6	0	**70**
8	Pick Nail	4	0	8	**58**
9-10	Move Nail - Move Hammer	11	8	0	**39**
11	Hammer Strike	5	0	0	**34**

We also elaborate on the failure and success breakdown for the bimanual reorientation task in Table S4. It is worth to be noted that the skills are executed in parallel and the serialized representation of the skill sequence is shown only as a part of the analysis.

Table S4: Failure breakdown and task success analysis per skill step of bimanual pot reorientation (based on 100 trials)

Skill.No.	Skills	Type 1 failure	Type 2 failure	Type 3 failure	Accu. Success
1	Grasp Pot Left	13	0	4	**83**
2	Grasp Pot Right	12	0	3	**68**
3-4	Move Pot Left - Move Pot Right	13	12	0	**53**

We further continue the analysis for all the two handover extensions of the *Hammer Nail* task, namely for *Extended Hammer Nail v1* in Table S5, for *Extended Hammer Nail v2* in Table S6, and for *Extended Hammer Nail v3* in Table S7. We primarily note the accumulative success at the first handover, coordination for the hammer `Strike`, and the second handover. With an increasing task horizon, the proposed approach is invariant to task-irrelevant distractions and maintains similar success.

Table S5: Failure breakdown and task success analysis per skill-step of hammer nail task extension v1 with two handovers (based on 100 trials)

Skill.No.	Skills	Type 1 failure	Type 2 failure	Type 3 failure	Accu. Success
1	Pick Lid	4	0	0	**96**
2	Place Lid	0	0	0	**96**
3	Pick Cube	0	0	0	**96**
4	Place Cube	5	0	0	**91**
5	Pick Hammer	4	0	2	**85**
6-7	Move Hammer - Regrasp Hammer	11	13	0	**61**
8	Pick Nail	3	0	3	**55**
9-10	Move Nail - Move Hammer	7	9	0	**39**
11	Hammer Strike	3	0	0	**36**
12-13	Move Hammer - Regrasp Hammer	4	6	0	**26**
14	Place Hammer	0	0	0	**26**
15	Pick Lid	2	0	0	**24**
16	Place Lid	0	0	0	**24**

Table S6: Failure breakdown and task success analysis per skill-step of hammer nail task extension v2 with two handovers and some task-irrelevant skills (based on 100 trials)

Skill No.	Skills	Type 1 failure	Type 2 failure	Type 3 failure	Accu. Success
1	Pick Lid	4	0	0	**96**
2	Place Lid	0	0	0	**96**
3	Pick cube	0	0	0	**96**
4	Place Cube	4	0	0	**92**
5	Pick Hammer	5	0	2	**85**
6-7	Move Hammer - Regrasp Hammer	12	14	0	**59**
8	Pick Nail	2	0	1	**56**
9-10	Move Nail - Move Hammer	4	0	7	**45**
11	Hammer Strike	1	0	0	**44**
12	Pick Lid	3	0	0	**41**
13	Place Lid	0	0	0	**41**
14-15	Move Hammer - Regrasp Hammer	6	7	0	**28**
16	Place Hammer	0	0	0	**28**
17	Pick Lid	3	0	0	**25**
18	Place Lid	0	0	0	**25**

Table S7: Failure breakdown and task success analysis per skill-step of hammer nail task extension v3 with two handovers and many task-irrelevant skills (based on 100 trials)

Skill No.	Skills	Type 1 failure	Type 2 failure	Type 3 failure	Accu. Success
1	Pick Lid	5	0	0	**95**
2	Place Lid	0	0	0	**95**
3	Pick cube	0	0	2	**93**
4	Place Cube	4	0	0	**89**
5	Pick Hammer	3	0	2	**84**
6-7	Move Hammer - Regrasp Hammer	4	8	0	**72**
8	Pick Nail	3	0	6	**63**
9-10	Move Nail - Move Hammer	7	9	0	**47**
11	Hammer Strike	5	0	0	**42**
12	Pick Lid	1	0	2	**39**
13	Place Lid	0	0	0	**39**
14-15	Move Hammer - Regrasp Hammer	5	8	0	**26**
16	Pick cube	0	0	0	**26**
17	Place Cube	1	0	0	**25**
18	Place Hammer	3	0	0	**22**
19	Pick Lid	0	0	1	**21**
20	Place Lid	0	0	0	**21**

References

[1] F. Dellaert. Factor graphs: Exploiting structure in robotics. *Annu. Rev. Control. Robotics Auton. Syst.*, 4:141–166, 2021. URL https://api.semanticscholar.org/CorpusID:234254791.

[2] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021.

[3] M. Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *IJCAI*, pages 1930–1936, 2015.

[4] Y. Lee, P. Huang, K. M. Jatavallabhula, A. Z. Li, F. Damken, E. Heiden, K. Smith, D. Nowrouzezahrai, F. Ramos, and F. Shkurti. Stamp: Differentiable task and motion planning via stein variational gradient descent. *arXiv preprint arXiv:2310.01775*, 2023.

[5] F. Dellaert. Factor graphs: Exploiting structure in robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):141–166, 2021. doi: 10.1146/annurev-control-061520-010504. URL https://doi.org/10.1146/annurev-control-061520-010504.

[6] Q. Xiao, Z. Zaidi, and M. Gombolay. Multi-camera asynchronous ball localization and trajectory prediction with factor graphs and human poses. *arXiv preprint arXiv:2401.17185*, 2024.

[7] Y. Hao, Y. Gan, B. Yu, Q. Liu, S.-S. Liu, and Y. Zhu. Blitzcrank: Factor graph accelerator for motion planning. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2023.

[8] Z. Yang, J. Mao, Y. Du, J. Wu, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling. Compositional diffusion-based continuous constraint solvers. *arXiv preprint arXiv:2309.00966*, 2023.

[9] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez. Active model learning and diverse action sampling for task and motion planning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4107–4114. IEEE, 2018.

[10] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez. Learning compositional models of robot skills for task and motion planning. *The International Journal of Robotics Research*, 40(6-7):866–894, 2021.

[11] B. Kim, L. P. Kaelbling, and T. Lozano-Perez. Guiding the search in continuous state-action spaces by learning an action sampling distribution from off-target samples. *arXiv preprint arXiv:1711.01391*, 2017.

[12] X. Fang, C. R. Garrett, C. Eppner, T. Lozano-Pérez, L. P. Kaelbling, and D. Fox. Dimsam: Diffusion models as samplers for task and motion planning under partial observability. *arXiv preprint arXiv:2306.13196*, 2023.

[13] U. A. Mishra, S. Xue, Y. Chen, and D. Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=HtJE9ly5dT.

[14] C. Agia, T. Migimatsu, J. Wu, and J. Bohg. Taps: Task-agnostic policy sequencing. *arXiv preprint arXiv:2210.12250*, 2022.

[15] W. Peebles and S. Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.

[16] W. Crooks, G. Vukasin, M. O'Sullivan, W. Messner, and C. Rogers. Fin ray® effect inspired soft robotic gripper: From the robosoft grand challenge toward optimization. *Frontiers in Robotics and AI*, 3:70, 2016.

[17] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[18] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[19] frankx. `https://github.com/pantor/frankx`, 2021.

[20] D. Driess, J.-S. Ha, and M. Toussaint. Learning to solve sequential physical reasoning problems from a scene image. *The International Journal of Robotics Research*, 40(12-14):1435–1466, 2021.

[21] J. Chen, J. Li, Y. Huang, C. Garrett, D. Sun, C. Fan, A. Hofmann, C. Mueller, S. Koenig, and B. C. Williams. Cooperative task and motion planning for multi-arm assembly systems. *arXiv preprint arXiv:2203.02475*, 2022.

[22] L. Nägele, A. Hoffmann, A. Schierl, and W. Reif. Legobot: Automated planning for coordinated multi-robot assembly of lego structures. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9088–9095. IEEE, 2020.

[23] L. P. Ureche and A. Billard. Constraints extraction from asymmetrical bimanual tasks and their use in coordinated behavior. *Robotics and autonomous systems*, 103:222–235, 2018.