Learning Discrete Abstractions for Visual Rearrangement Tasks Using Vision-Guided Graph Coloring

Abhiroop Ajith¹ and Constantinos Chamzas¹

Abstract—Learning abstractions directly from data is a core challenge in robotics. Humans naturally operate at an abstract level, reasoning over high-level subgoals while delegating execution to low-level motor skills—an ability that enables efficient problem solving in complex environments. In robotics, abstractions and hierarchical reasoning have long been central to planning, yet they are typically hand-engineered, demanding significant human effort and limiting scalability. Automating the discovery of useful abstractions directly from visual data would make planning frameworks more scalable and more applicable to real-world robotic domains. In this work, we focus on rearrangement tasks where the state is represented with raw images, and propose a method to induce discrete, graphstructured abstractions by combining structural constraints with an attention-guided visual distance. Our approach leverages the inherent bipartite structure of rearrangement problems, integrating structural constraints and visual embeddings into a unified framework. This enables the autonomous discovery of abstractions from vision alone, which can subsequently support high-level planning. We evaluate our method on two rearrangement tasks in simulation and show that it consistently identifies meaningful abstractions that facilitate effective planning and outperform existing approaches.

I. INTRODUCTION

A central challenge in artificial intelligence and robotics is enabling agents to reason at an abstract, goal-directed level while acting in a continuous, high-dimensional sensorimotor world. Humans naturally bridge this gap [1]: when packing a lunch, one thinks in terms of subgoals—open the backpack, place the lunch inside, zip it closed—rather than orchestrating every joint movement. This ability to rely on abstract reasoning while delegating execution to low-level motor skills is key to human intelligence. In contrast, robots are often forced to operate directly on raw observations (e.g., images) and low-level controls, making even seemingly simple tasks computationally expensive, brittle, and hard to generalize. Bridging symbolic planning and sensorimotor control thus remains a fundamental obstacle to building autonomous and adaptable robots [2].

Within robotics, task and motion planning (TAMP) frameworks have long exploited symbolic abstractions to decompose long-horizon problems into tractable subproblems [3]. These approaches integrate high-level task reasoning with geometric motion planning and have shown increased generalizability and long-horizon performance [4], [5] in robotic manipulation. However, their applicability is limited because they require manually specified abstractions and precise world models, which are rarely available in real-world settings.

In this work, we focus on learning abstractions for *visual* rearrangement problems, as illustrated in Fig. 1. The robot

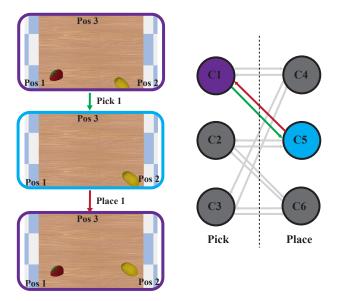


Fig. 1: From images and actions to an abstract task graph. Left: example RGB observations o from the execution trace \mathcal{D} . Right: the induced bipartite transition graph over learned clusters, split by role into pick (left of dotted line) and place (right). Clusters are obtained by constrained graph coloring guided by attention-weighted visual distance; edges are action-labeled transitions observed in \mathcal{D} .

is provided with execution traces consisting of raw visual observations paired with high-level action labels corresponding to manipulation primitives such as pick and place. Our objective is to induce an abstract transition graph Fig. 1 (right), whose nodes represent equivalence classes of task indistinguishable states and whose edges correspond to actionlabeled transitions. Such a graph constitutes a minimal and *sufficient* representation [6] for computing task satisfying high-level action plans.

Prior work on visual task planning [7] typically learns a latent representation of images [8], [9] and then performs clustering in the latent space to induce an abstract task graph. However, abstract graphs that encode real world tasks are subject to structural constraints. For example, there must be at most one high-level action between any pair of states, and states connected by an action must belong to distinct abstract nodes. Existing methods do not strictly enforce these constraints during representation learning, often producing graphs that violate them.

To overcome this limitation, we explicitly encode general hard structural constraints that any rearrangement abstraction must satisfy and restrict the learned graph to be feasible under these rules. Since these constraints alone underdetermine the abstraction (an *underdetermined problem*), we additionally leverage an unsupervised latent-space similarity metric to guide graph discovery, yielding abstractions that are both structurally valid and visually coherent.

We evaluate our method on two rearrangement manipulation tasks [10]. Assuming minimal prior knowledge—only access to a generic pick-and-place skill—we demonstrate that the proposed framework can autonomously construct higher-level abstractions that enable planning directly from raw images.

II. RELATED WORK

In robotics, abstractions can provide compact, interpretable models that reduce the complexity of raw sensorimotor data, making long-horizon reasoning and planning both tractable and generalizable across tasks [6], [11]. Learning such abstractions directly from data remains an open challenge, and has motivated learning different abstraction representations.

Higher-order abstractions: The early foundational work of [6] sought to induce STRIPS-style symbolic rules from experience, but assumed object-centric state descriptions and perfect knowledge of action executability. More recently, [12] extended this idea to learning expressive PDDL representations capable of capturing first-order logic, though their method likewise required object-centric states and known operator dynamics. In contrast, the work of [13] pioneered learning STRIPS-style abstractions directly from raw images, removing the need for predefined object models. However, their framework did not leverage action labels, which are readily available in robotics domains. Our approach builds on this direction by combining visual information with action-labeled transitions, enabling the discovery of structured abstractions tailored to rearrangement tasks.

Lower-order abstractions: At the opposite end of the spectrum, the simplest form of abstraction over a real world continuous domain is a flat graph representation, where nodes correspond to discrete states and edges encode discrete action transitions. Such graphs have been explored in robotics. Two main families of methods can be distinguished. The first follows a two-stage pipeline: learn a representation of states from sensory data and then cluster embeddings to generate a connectivity graph, with edges defined via execution traces or latent distances [8], [14]-[16]. The second jointly learns state embeddings together with forward or inverse dynamics models, which can encode some form of forward dynamics consistency, of the resulting graphs [9], [13]. While these approaches have shown promise in producing visually grounded graph abstractions, they provide no guarantees that the learned abstractions respect the underlying action constraints. Our work addresses these limitations by incorporating structural constraints into the graph induction process, while grounding state similarity directly in visual data.

Rearrangement Planning: Rearrangement of objects has been a long-standing focus in robotics [10]. Most existing approaches assume access to object-centric representations, either by relying on known object positions [17], performing object segmentation in the scene [18], or establishing image

correspondences across views. While effective in scaling to several objects [17], these methods all depend on explicit object models. In this work, we address rearrangement under a weaker assumption: the robot is equipped only with high-level pick-and-place actions, and perceives the world solely through raw images. We do not assume access to object identities, poses, or segmentation masks. Instead, our goal is to recover a relevant abstraction in the form of a symbolic graph, discovered in a fully unsupervised manner from visual observations and action-labeled transitions.

III. PROBLEM FORMULATION AND NOTATION

A. Problem Formulation

We begin by defining a planning problem, and then specialize it to the case of rearrangement tasks, followed by the vision-based setting considered in this work.

Classical Planning Problem Let \mathcal{S} be a finite set of discrete states, and let \mathcal{A} be a finite set of discrete actions. The environment is governed by a deterministic transition function $T: \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, which maps a state $s \in \mathcal{S}$ and an action $a \in \mathcal{A}$ to a successor state s' = T(s, a). Given a start state $s_{\text{start}} \in \mathcal{S}$ and a goal state $s_{\text{goal}} \in \mathcal{S}$, the objective is to compute a *plan*, i.e., a sequence of actions $\pi = (a_1, \ldots, a_T)$ with $a_t \in \mathcal{A}$, such that executing π transitions the system from s_{start} to s_{goal} under T.

Rearrangement Planning Problem A rearrangement problem can be viewed as a special case of the classical planning formulation with the additional structure:

- 1) The action set consists of high-level manipulation primitives, specifically pick and place actions, i.e., $\mathcal{A} = \{ \texttt{pick}_i, \texttt{place}_i \}$, where the index i distinguishes different pick/place actions, as would be the case if picking or placing in different regions.
- 2) The transition structure is bipartite: every pick action must be followed by a place (and vice-versa), and no two consecutive actions of the same type are valid.

In this work we assume that object identity is irrelevant for planning. Concretely, we treat all objects as interchangeable (permutation-invariant), and use object-agnostic action templates $\mathcal{A} = \{\texttt{pick}_i, \texttt{place}_i\}$ rather than object-indexed actions. This models tasks where only the spatial arrangement matters (e.g., "some fruit here, some fruit there"), not which specific instance occupies a location.

If object identities must be respected, the action set can be expanded to object-indexed labels,

$$\mathcal{A}_{\mathrm{inst}} = \big\{ \, \mathrm{pick}_i^{\mathrm{obj}}, \, \, \mathrm{place}_i^{\mathrm{obj}} \, \big\}.$$

Visual Rearrangement Planning Problem

In the visual rearrangement setting, the robot does not observe the discrete state space $\mathcal S$ directly. Instead, it perceives the world through raw visual observations $o \in \mathcal O$. We assume that each observation is generated from the underlying state via an (unknown) observation function $\phi: \mathcal S \to \mathcal O$, such that $o=\phi(s)$. Thus, while every observation corresponds to some state $s\in \mathcal S$, the mapping ϕ , the discrete state space $\mathcal S$, and the transition function T are considered unknown. Instead

of direct access to the task model, we assume the robot is provided with a dataset of observed execution traces:

$$\mathcal{D} = \{(o_0, a_0, o_1), (o_1, a_1, o_2), \dots, (o_{T-1}, a_{T-1}, o_T)\},\$$

where each o_t is a visual observation and a_t is the corresponding high-level action applied between successive observations.

Concretely, we define the visual rearrangement planning problem as: Given a start observation $o_{\text{start}} = \phi(s_{\text{start}})$ and a goal observation $o_{\text{goal}} = \phi(s_{\text{goal}})$, find a sequence of high-level actions $\pi = (a_1, \ldots, a_T)$ that transitions the system from s_{start} to s_{goal} .

IV. APPROACH

To be able to produce a feasible high-level plan, our method infers an action-labeled, bipartite transition graph from the observation–action trace \mathcal{D} ,

$$\hat{\mathcal{G}} = (\hat{\mathcal{V}}_{pick} \cup \hat{\mathcal{V}}_{place}, \, \hat{\mathcal{E}}),$$

together with an assignment map $f: \mathcal{O} \to \hat{\mathcal{V}}$ that sends any image $o \in \mathcal{O}$ to an abstract state f(o). In essence, every vertex v corresponds to a discrete abstract state $s \in \mathcal{S}$ and every edge corresponds to a discrete action $a \in \mathcal{A}$.

Bipartiteness ensures the pick \(\to place \) alternation: edges always cross roles (pick \(\to place \) place or place \(\to pick); there are no pick \(\to pick \) or place \(\to place \) edges.

At deployment, f is implemented by a classifier on fixed visual embeddings trained with the cluster labels discovered during learning; thus f approximates the inverse observation map up to abstraction: $f \approx \phi^{-1}$.

Given $(o_{\text{start}}, o_{\text{goal}})$, we compute $(f(o_{\text{start}}), f(o_{\text{goal}}))$ and run a standard graph search (e.g., BFS) in $\hat{\mathcal{G}}$ to recover a high-level action sequence. To learn the graph from the execution trace \mathcal{D} , we follow two principles:

- 1) enforce the structural constraints in Sec. V-A;
- 2) minimize the intra-cluster visual distance in Sec. V-B.

V. GRAPH CONSTRAINTS AND VISUAL DISTANCE

A. Structural Constraints

We learn $\hat{\mathcal{G}}$ under three structural constraints:

(i) Bipartiteness. Edges always cross roles (pick→place or place→pick); no pick→pick or place→place edges can exist.
 (ii) Action-uniqueness (Exactly-One). Rearrangement graphs only have one unique action between two states. This constraint is shown in Fig. 2:

$$\forall v \in \hat{\mathcal{V}}, \ \forall a \in \mathcal{A}: \quad \left| \left\{ v' : (v, a, v') \in \hat{\mathcal{E}} \right\} \right| \le 1. \quad (1)$$

(Equivalently: between any ordered pair (u, v) there is at most one action label; cf. Fig. 2.)

(iii) **Bounded action variety.** As an *optional* structural prior, a state supports only a small set of distinct outgoing action labels:

$$\forall v \in \hat{\mathcal{V}}: \quad \left| \left\{ a : \exists v' (v, a, v') \in \hat{\mathcal{E}} \right\} \right| \le K_{\text{cap}}. \quad (2)$$

We treat $K_{\rm cap}$ as unknown and select it from a small discrete sweep. We found finite caps helpful in practice for producing compact, well-connected graphs.

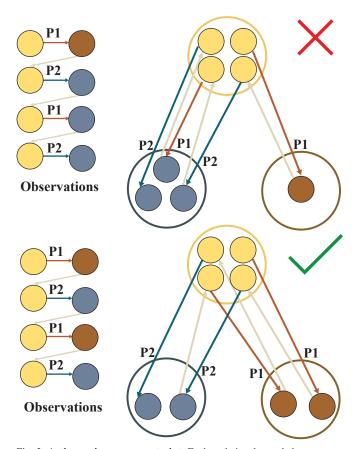


Fig. 2: **Action–uniqueness constraint:** Each node is a learned abstract state. Top: invalid—between the same two clusters we observe two different actions (e.g., p_1 and p_2); this violates our rule that there must be a single action between any fixed pair of abstract states. Bottom: valid—only one action connects the pair; any second action must lead to a different destination cluster.

These constraints create an underdetermined problem with many feasible graphs. To select among them, we integrate an attention-guided visual distance (Sec. V-B).

B. Attention-Guided Visual Distance

To guide the coloring search in a completely unsupervised manner, we require a visual distance that favors grouping images which are *indistinguishable for the task*, while being insensitive to superficial appearance changes and tolerant to minor alignment jitter.

Attention-derived spatial weights.: Each RGB observation o is processed by a frozen DINO-v2 ViT-G/14 encoder. From the final transformer block we take the [CLS]—patch attention and normalize it into a non-negative spatial probability map M(o) over the image grid. To suppress background, we keep the largest connected components in this map and lightly smooth/renormalize so that $\sum_{ij} M_{ij}(o) = 1$. The result is a soft spatial weighting that highlights the task-relevant region without relying on object identities or semantics.

Given two maps $M(o_i)$ and $M(o_j)$ supported on a common pixel grid with ground cost $C_{uv} = \|x_u - x_v\|_2$ (Euclidean distance between pixel centers), we compare them via the entropically regularized optimal-transport cost

 $\mathrm{OT}_{\varepsilon}(p,q)$ and its debiased Sinkhorn divergence:

$$S_{\varepsilon}(p,q) = \mathrm{OT}_{\varepsilon}(p,q) - \frac{1}{2}\mathrm{OT}_{\varepsilon}(p,p) - \frac{1}{2}\mathrm{OT}_{\varepsilon}(q,q).$$
 (3)

$$d_{\text{vis}}(o_i, o_j) = S_{\varepsilon}(M(o_i), M(o_j)). \tag{4}$$

where p,q are the vectorized maps. Intuitively, $d_{\rm vis}$ measures how much spatial "mass" must move to align the attention patterns. Two images that place the same item (or a visually different item, e.g., strawberry vs. lemon) in the same region yield small cost (the purple states in Fig. 1 are similar even though the object moved); moving mass across regions incurs a larger cost. This aligns well with rearrangement abstractions and the bipartite structure in Fig. 3.

We call $d_{\rm vis}$ the attention-guided OT distance. For coloring we convert distances to an affinity $K_{\rm vis}(o_i,o_j)=\exp(-d_{\rm vis}(o_i,o_j)/ au)$ and score a partition $\mathcal C$ by the average intra-cluster visual distance

$$\mathcal{V}(\mathcal{C}) = \sum_{C \in \mathcal{C}} \frac{1}{|C|(|C|-1)} \sum_{o \neq o' \in C} d_{\text{vis}}(o, o'), \quad (5)$$

where lower is better. We refer to V as the intra-cluster visual distance (V).

VI. METHODOLOGY

A naive strategy to generate a structurally feasible graph would be to enumerate every partition of the pick images into $k_{\rm pick}$ clusters and every partition of the place images into $k_{\rm place}$ clusters, keep only those that satisfy the constraints, and then select the visually most coherent solution (Sec. V-B). Even for the Fruit domains with 15 pick and 15 place images and just $k_{\rm pick} = k_{\rm place} = 3$, the number of partitions explodes: the count per side is the Stirling number of the second kind S(n,k), so the joint search size is

$$S(15,3)^2 = (2,375,101)^2 \approx 5.6 \times 10^{12}$$

before checking constraints or evaluating visual scores. If k were unknown, the space becomes all set partitions: the Bell number $B_{15}=1.38\times 10^9$ per side, yielding $B_{15}^2\approx 1.9\times 10^{18}$ candidates. This is clearly intractable. Instead of brute force, we solve a constraint-guided clustering problem via graph coloring (Sec. VI-A), where feasibility is enforced by construction and vision guides choices among feasible options.

A. Vision-Guided Graph Coloring

We treat state discovery as coloring two role-indexed sets of observations—*pick* and *place*—subject to the structural constraints (1)–(2). Fig. 3 walks through the four stages of our procedure.

Stage 1: Vision-only seeding of the pick side (Fig. 3, top-left): We seed the pick partition using visual affinity (high $K_{\rm vis}$) alone. Concretely, we rank uncolored pick images by visual centrality (highest mean affinity $K_{\rm vis}$ to the remainder), start a new color with the current seed, and then grow that color greedily by repeatedly adding the unassigned image with minimum $d_{\rm vis}$ to the cluster—provided the mean intra-cluster distance does not increase. This yields compact, visually coherent seeds without yet checking relational constraints. The number of colors is bounded by $k_{\rm pick}$.

Stage 2: Place-side DSATUR under constraints (Fig. 3, top-middle): Holding the Stage-1 pick partition fixed, we build the place conflict graph: two place images are adjacent if putting them in the same cluster would violate either constraint (1)–(2) when combined with the already colored pick partition and the observed transitions (o_t, a_t, o_{t+1}) in \mathcal{D} . We then color this graph with at most k_{place} colors using DSATUR [19].

- **Ordering.** At each step we select the place image with the highest saturation degree (most distinct colors seen among its neighbors), breaking ties by total degree.
- Vision-guided color choice. Among *feasible* colors, we choose the one with the highest mean $K_{\rm vis}$ affinity to its current members, thereby integrating vision into the combinatorial step.

Stage 3: Pick-side DSATUR with place frozen (Fig. 3, bottom-middle): After obtaining a feasible coloring for the place side in Stage 2, we discard the greedy seeds on the pick side from Stage 1 (i.e., uncolor all pick nodes) and recolor pick from scratch while holding the place partition fixed. Concretely, we build the pick conflict graph: two pick images are adjacent if merging them into the same cluster would violate either structural constraint (1)–(2) when combined with the frozen place partition and the observed transitions $(o_t, a_t, o_{t+1}) \in \mathcal{D}$. We then color this graph with at most k_{pick} colors using DSATUR, using the same policy as in Stage 2: After this stage, both sides have been colored against each other and the resulting action-labeled bipartite graph satisfies the constraints by construction.

Feasibility and backtracking: When testing an assignment $v \to c$ during DSATUR (Stages 2–3), we maintain a tiny *action–successor table* Λ that records, for each cluster c and action a, the unique successor $\Lambda(c,a)$ already fixed by earlier decisions. An assignment is *feasible* iff:

- 1) No neighbor conflict. No neighbor of v in the current role's conflict graph already has color c.
- 2) **Exactly-One.** For every observed transition (v, a, v') whose opposite-role partner v' is already colored, either $\Lambda(c, a)$ is unset or equals the color of v'.
- 3) Out-degree cap. Adding v to c would keep the number of distinct outgoing labels from $c \leq K_{\text{cap}}$.

If several colors are feasible we use the vision-guided rule above. If no existing color is feasible and we have not yet used $k_{\rm pick}/k_{\rm place}$ colors, we *open* a new color. Otherwise we *backtrack* to the most recent undecided vertex and try its next

¹We downsample the maps before OT for efficiency and cache all pairwise distances.

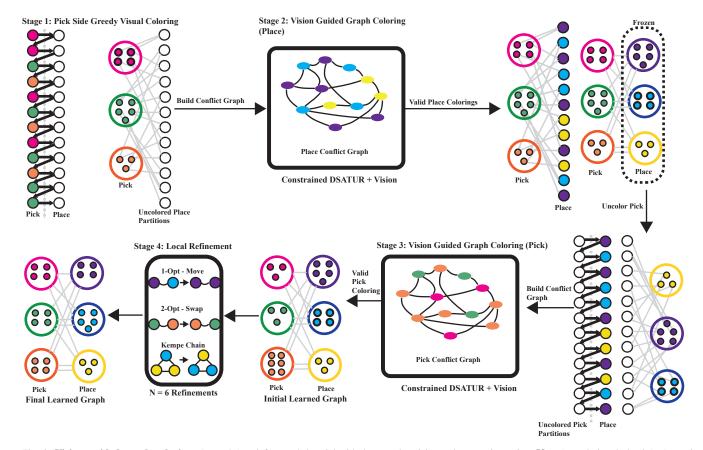


Fig. 3: **Vision-guided graph coloring.** Stage 1 (top-left): seed the pick side by greedy, vision-only grouping using K_{vis} (no relational checks). Stage 2 (top-middle): build the place conflict graph whose edges indicate merges that would violate Exactly-One or exceed K_{cap} ; color with DSATUR, choosing among admissible colors by visual affinity. Stage 3 (bottom-middle): freeze place and recolor the pick side the same way; the bipartite graph now satisfies the constraints. Stage 4 (bottom-left): local refinement (1-opt, 2-opt, Kempe chains) preserves feasibility and reduces the intra-cluster distance, producing the final learned graph.

feasible color. In practice, DSATUR's "most-constrained-first" ordering keeps backtracks shallow.

Stage 4: Local refinement (both roles) (Fig. 3, bottom-left): Starting from the first feasible coloring obtained from Stage 3, we run a short alternating local search that refines both sides. We freeze one role(pick or place) and improve the other by proposing only admissible edits: (i) 1-opt (move a single image between clusters), (ii) 2-opt (swap two images across clusters), and (iii) two-color Kempe flips (swap colors along a connected two-color component to free capacity). An edit is accepted iff (a) all conflict edges remain satisfied, (b) the action–successor table Λ remains consistent with Exactly-One, (c) the out-degree cap $K_{\rm cap}$ is not violated, and (d) the role's visual score $\mathcal V$ strictly decreases.

After *any* accepted edit on the active role (say, place), we immediately *re-solve the opposite role once* with DSATUR (re-color pick against the updated place partition) to keep the two sides synchronized and feasibility intact (Fig. 3, Stage 4). We then switch roles and repeat. We alternate $place \leftrightarrow pick$ for N rounds (we use N=6) or stop early when no admissible edit improves \mathcal{V} . In practice, one to three alternations suffice.

Algorithm 1: Vision-guided constrained coloring with grid sweep (see Fig. 4).

```
Input: trace \mathcal{D}, role labels, K_{\operatorname{cap}}, grid \mathcal{K} = \{(k_{\operatorname{pick}}, k_{\operatorname{place}})\}
Output: partition \mathcal{C}^* and graph \hat{\mathcal{G}}^*

1 foreach (k_{\operatorname{pick}}, k_{\operatorname{place}}) \in \mathcal{K} do

2 \mathcal{S} \leftarrow \operatorname{COLORWITHDSATUR}(\mathcal{D}, k_{\operatorname{pick}}, k_{\operatorname{place}}) if \mathcal{S} = \emptyset then

3 \operatorname{mark} cell infeasible; continue

4 foreach partition \mathcal{C} \in \mathcal{S} do

5 \operatorname{compute} \mathcal{V}(\mathcal{C}) with d_{\operatorname{vis}}; keep the \operatorname{arg\,min}_{\mathcal{C}} \mathcal{V}

6 Pick \mathcal{C}^* by: feasibility \succ minimal k_{\operatorname{pick}} + k_{\operatorname{place}} \succ minimal \mathcal{V}; return \mathcal{C}^* and its \hat{\mathcal{G}}^*.
```

B. Learned Graph Selection

Selection across (k_{pick}, k_{place}) : We repeat the four-stage procedure above (as outlined in algorithm 1) across a small grid of (k_{pick}, k_{place}) . For each grid cell we keep the best feasible partition (lowest intra-cluster visual distance \mathcal{V}) and then select the final abstraction by (i) feasibility, (ii) fewest total clusters $k_{pick}+k_{place}$, and (iii) minimum \mathcal{V} to break ties. The resulting sweep is summarized in Fig. 4; the selected

graph is the one we use for planning and evaluation.

C. Few-Shot Visual-to-State Classification

To localize any *new* observation o_{new} on the graph, we need to train a vision classifier that maps an RGB image to a node in \hat{V} , in essence learning the inverse of the observation function ϕ^{-1} .

Label bootstrapping: Labels come directly from the learned partition (Sec. VI-A). Let $\hat{\mathcal{V}} = \hat{\mathcal{V}}_{\text{pick}} \cup \hat{\mathcal{V}}_{\text{place}}$ and fix an index map $\mathrm{idx}: \hat{\mathcal{V}} \to \{1,\ldots,K\}$ with $K = k_{\mathrm{pick}} + k_{\mathrm{place}}$ (e.g., all pick nodes first, followed by place nodes). For every training image o_i , the classifier label is the index of its learned state,

$$y_i := idx(f(o_i)) \in \{1, \dots, K\}.$$

Thus $\mathcal{D}_{cls} = \{(o_i, y_i)\}$ requires no extra annotation and is fully consistent with the abstraction.

Model: Because each node has few labeled images, we adopt a few-shot regimen: a DINO-v2 ViT-G/14 encoder is used to extract features $\Phi(o) \in \mathbb{R}^d$, while only the last transformer block is unfrozen. On top we place a cosine classifier with temperature γ :

$$z_k(o) = \frac{1}{\gamma} \frac{w_k^{\top} \Phi(o)}{\|w_k\| \|\Phi(o)\|}, \qquad \hat{y}(o) = \arg\max_k z_k(o),$$
(6)

where $w_k \in \mathbb{R}^d$ is the learnable vector for node k. We optimize the class weights and the last transformer block with cross-entropy over the logits $\{z_k(o_i)\}$.

Training protocol: Classes (nodes) are few and imbalanced, so we use stratified K-fold cross-validation (default K=5) to choose hyperparameters and guard against overfitting. Within each fold we train with Adam using cosine-head LR 10^{-3} , last-block LR 10^{-4} , weight decay 10^{-4} , batch size 32, and early stopping (patience 10, max 100 epochs). We apply light augmentation (random resized crop, horizontal flip, mild color jitter) to improve robustness to appearance changes (e.g., color or category swaps at the same placement). After model selection, we retrain on all (o_i, y_i) and deploy the classifier. At inference, a single forward pass maps a novel observation o_{new} to a node index $\hat{y}(o_{\text{new}}) \in \{1, \ldots, K\}$ on $\hat{\mathcal{G}}$, which is then used for high-level planning.

Defaults: Unless stated otherwise, we set $\gamma{=}0.1$ and keep only the last ViT block trainable; all other encoder weights remain frozen.

VII. EXPERIMENTS AND RESULTS

A. Experimental Setup

Simulator and data.: We evaluate in **PyBullet** [20] on table-top rearrangement tasks. Let $\mathcal O$ denote RGB observations and $\mathcal A$ the discrete, high-level action labels (pick/place variants). Each dataset consists of observation–action transitions

$$\mathcal{D} = \{(o_t, a_t, o_{t+1})\},\$$

where $a_t \in \mathcal{A}$ and the role alternates (pick \rightarrow place \rightarrow pick). A key property of our scenes is that placements are *continuous* within a target region: the workspace is partitioned into a

Intra-cluster visual distance

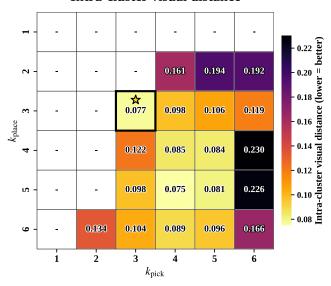


Fig. 4: **Grid sweep.** Evaluating (k_{pick}, k_{place}) with Algorithm 1. Lower $\mathcal V$ is better. For Fruit (homogeneous), the best is (3,3). Blank cells indicate no solution was found for that pair.

small set of regions, and when an object is placed into a region its planar pose (x,y) is sampled uniformly inside a fixed-radius disk around that region's nominal center. Thus many distinct images can correspond to the same abstract state.

Environments:

- Fruit rearrangement (hom./het.). Two object slots and three placement regions (e.g., two corners and center); camera is fixed. We instantiate a homogeneous variant (two identical fruits) and a heterogeneous variant (e.g., strawberry vs. lemon). All valid pick—place moves between regions are permitted (six unique action labels). Dataset size: 30 observations.
- **Blocks World (2 blocks).** Two blocks and three positions (left/center/right) with stacking/unstacking enabled. We use 12 unique high-level actions in this domain. To test appearance robustness, we swap block colors across runs (geometry and viewpoints held fixed). *Dataset size:* 45 observations.
- **Blocks World (3 blocks).** Three blocks over the same three positions, yielding a larger configuration space with stacking. We use 18 unique high-level actions. Color swaps are likewise applied only in this Blocks World setting. *Dataset size: 150 observations*.

Baselines. We evaluate two clustering baselines that follow the spirit of [9], [21], where a representation of observations is learned, and then a graph is created based on the learned latent distances. Both baselines operate on the same attention-guided visual embeddings described in Sec. V-B and induce abstract-graph edges by mapping observed transitions (o_t, a_t, o_{t+1}) to cluster indices.

Pre-partitioning (implicit contrast). To mimic the contrastive separation between action roles, we pre-partition the

TABLE I: Optimal Path (Opt \uparrow), *Intra-cluster visual distance* ($V \downarrow$), and Transitions (Trans \uparrow).

Method	Fruit (Hom)				Fruit (Het)				Blocks (2)				Blocks (3)			
	Opt	AnyPath	Trans	\mathcal{V}	Opt	AnyPath	Trans	\mathcal{V}	Opt	AnyPath	Trans	\mathcal{V}	Opt	AnyPath	Trans	\mathcal{V}
HDBSCAN	6%	10%	74.5%	0.0653	4%	4%	74.5%	0.0942	8%	8%	76.5%	0.047	2%	2%	80%	0.0869
Agglomerative	37.5%	37.5%	50.00%	0.1351	35.0%	35.0%	37.5%	0.1502	33%	33%	33%	0.0839	6%	6%	50%	0.1206
Ours	86%	100%	88%	0.077	82%	100%	83.33%	0.08	66%	93%	80%	0.091	45%	81%	45%	0.098

TABLE II: Per-environment class counts and classifier accuracy.

Environment	# Classes	Classifier Accuracy (%)
Fruit (Hom)	6	88.9
Fruit (Het)	6	88.9
Blocks (2)	9	72.2
Blocks (3)	16	62.2

observations into pick and place subsets using the role indicated by the action sequence (pick \rightarrow place \rightarrow pick). Clustering is performed *separately* within each subset; this prevents cross-role merging and acts as an implicit contrastive constraint.

Agglomerative. We run hierarchical agglomerative clustering on the role-specific embeddings and select the linkage and cut level via a small sweep to balance over- and under-segmentation (targeting a cluster count comparable to the domain's state budget). No metric details are assumed beyond a fixed feature-space affinity on the embeddings. After clustering, edges in the abstract graph are induced directly from (o_t, a_t, o_{t+1}) by mapping endpoints to their clusters.

HDBSCAN. [22] We apply hierarchical density-based clustering to the same role-specific embeddings. We tune min_cluster_size and min_samples per dataset to control granularity, and (when needed) assign outliers to the nearest existing cluster in embedding space to obtain a total labeling consistent with graph induction from transitions.

Difference from ours.: All methods use the same attention—guided features. Baselines cluster within pre-partitioned pick/place subsets and only then induce edges from transitions. In contrast, our approach integrates role information during coloring via the conflict graphs and enforces constraints at assignment time; visual affinity is used only to choose among feasible colorings. This yields graphs that are structurally valid by construction.

B. Evaluation Metrics

We evaluate along two axes: (i) planning success on the learned abstraction and (ii) intra-cluster visual distance of the clusters.

- a) Planning Metrics: For each environment we sample $K{=}500$ start–goal pairs (c_0, c_G) from distinct nodes of the learned graph $\hat{\mathcal{G}}$. For each pair, we (1) enumerate all shortest paths with BFS; if no path is found, we run DFS to search for non-shortest alternatives; (2) deem a path valid only if its entire action sequence executes in the ground-truth transition graph \mathcal{G}^* (one invalid step \Rightarrow failure). We report:
 - OptPath%: fraction of pairs for which at least one BFS shortest path in \(\hat{G}\) is valid in \(\mathcal{G}^*\).

- AnyPath%: fraction of pairs for which *any* discovered path (shortest or not) is valid in \mathcal{G}^* .
- Transition%: fraction of directed edges $(c_i \to c_j)$ in $\hat{\mathcal{G}}$ whose labeled action yields a valid transition in \mathcal{G}^* (edge counted correct only if validation succeeds).
- b) Intra-cluster visual distance: , we report mean intra-cluster visual distance (lower is better; 0 indicates visually indistinguishable, values are in [0,1]). This measures whether visually similar (task-indistinguishable) observations are grouped together.

C. Results

For each environment we sweep $(k_{\rm pick}, k_{\rm place})$ and select a solution by (i) feasibility with the fewest clusters and (ii) lowest intra-cluster visual distance \mathcal{V} (Sec. V-B). Fig. 4 (grid sweep) illustrates this selection on Fruit (hom.); we use the same procedure in all domains. We then train the few-shot localizer to map novel images to the learned graph (Sec. VI-C). See Table I for the full metrics.

- a) Planning quality: Across domains our method consistently yields graphs that are connected and plan-worthy. On both fruit tasks the abstraction supports successful planning between essentially all start–goal pairs, with most plans also shortest. In Blocks (2) we retain high planning success, and even in the harder Blocks (3) (16 learned states) our abstraction still connects the space reliably, while the baselines rarely find any valid path at all. In short: enforcing Exactly-One plus a small action-variety cap produces globally navigable graphs; the baselines often produce visually similar but disconnected ones. (Compare AnyPath/Opt trends in Table I.)
- b) Why "low \mathcal{V} " or "high Trans" can mislead: HDBSCAN (and, at times, Agglomerative) sometimes reports lower visual intra-cluster distance ($\mathcal{V}\downarrow$) and higher edge validity (TRANS↑) than ours in the most challenging setting. This is largely an artifact of *fragmentation*: over-splitting the state space into many tiny clusters (often near duplicates) drives intra-cluster distances down by construction, and the few edges that remain are locally valid, inflating TRANS. However, the graph breaks into small components, so end-to-end connectivity collapses (very low AnyPath). Our method accepts slightly higher \mathcal{V} and a modest drop in local edge purity in exchange for *global* connectivity and planning success. See the contrast between TRANS and AnyPath on Blocks (3) in Table I.
- c) Cluster cohesion: On Fruit (Het) and Blocks (2), our abstractions are at least as visually coherent as the baselines (similar or lower \mathcal{V}). On Blocks (3), the baselines' slightly lower \mathcal{V} stems from the same over-segmentation discussed

above and does not translate into usable plans (AnyPath \leq a few percent).

- d) Few-shot localization: Despite few and imbalanced labels, the graph localizer achieves $\approx 89\%$ on both fruit domains, $\approx 72\%$ on Blocks (2), and $\approx 62\%$ on Blocks (3), tracking task difficulty and class count (Table II). This indicates that the learned abstraction provides a practical label space for visual planning.
- e) Takeaways: (i) Structural constraints (Exactly-One, bounded action variety) are decisive for building connected abstractions that support planning; (ii) visual affinity is best used to choose among feasible colorings, not to define the graph alone; (iii) even with noisy, few labels, a lightweight few-shot classifier suffices to localize observations on the learned graph.

VIII. CONCLUSIONS

This work introduces one of the first approaches to enforce graph-structural constraints in representation learning, yielding action-consistent abstract graphs by construction. While promising, our method faces several limitations. First, scalability remains a challenge: solving the underlying graph coloring problem grows combinatorially with data size, limiting applicability to small domains. Future work could address this by adopting hierarchical or factored representations [23], enabling compact abstractions that scale with more objects and actions. Also, our experiments are limited to simulation. Evaluating on real robots will test robustness under noise and uncertainty, and incorporating richer task priors (e.g., forward/inverse dynamics) may further ground the abstractions in physical interaction.

REFERENCES

- [1] M. T. Mason, "Toward Robotic Manipulation," Annual Review of Control, Robotics, and Autonomous Systems, vol. 1, no. 1, pp. 1–28, May 2018. [Online]. Available: https://www.annualreviews.org/doi/10. 1146/annurev-control-060117-104848
- [2] O. Kroemer, S. Niekum, and G. Konidaris, "A review of robot learning for manipulation: challenges, representations, and algorithms," *J. Mach. Learn. Res.*, vol. 22, no. 1, Jan. 2021.
- [3] 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*, vol. 4, no. 1, pp. 265–293, May 2021. [Online]. Available: https://www.annualreviews.org/doi/10.1146/annurev-control-091420-084139
- [4] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "An incremental constraint-based framework for task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1134–1151, 2018.
- [5] W. Thomason, M. P. Strub, and J. D. Gammell, "Task and motion informed trees (tmit*): Almost-surely asymptotically optimal integrated task and motion planning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11370–11377, 2022.
- [6] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, "From Skills to Symbols: Learning Symbolic Representations for Abstract High-Level Planning," *Journal of Artificial Intelligence Research*, vol. 61, pp. 215–289, Jan. 2018. [Online]. Available: https://jair.org/index.php/jair/article/view/11175
- [7] C. Paxton, Y. Barnoy, K. Katyal, R. Arora, and G. D. Hager, "Visual robot task planning," in 2019 international conference on robotics and automation (ICRA). IEEE, 2019, pp. 8832–8838.
- [8] M. Lippi, P. Poklukar, M. C. Welle, A. Varava, H. Yin, A. Marino, and D. Kragic, "Enabling Visual Action Planning for Object Manipulation Through Latent Space Roadmap," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 57–75, Feb. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9833914/

- [9] M. Bagatella, M. Olšák, M. Rolínek, and G. Martius, "Planning from pixels in environments with combinatorially hard search spaces," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24707–24718, 2021.
- [10] D. Batra, A. X. Chang, S. Chernova, A. J. Davison, J. Deng, V. Koltun, S. Levine, J. Malik, I. Mordatch, R. Mottaghi, M. Savva, and H. Su, "Rearrangement: A Challenge for Embodied AI," Nov. 2020, arXiv:2011.01975 [cs]. [Online]. Available: http://arxiv.org/abs/2011.01975
- [11] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic planning and control of robot motion [Grand Challenges of Robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 61–70, Mar. 2007. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4141034
- [12] T. Silver, R. Chitnis, N. Kumar, W. McClinton, T. Lozano-Perez, L. P. Kaelbling, and J. Tenenbaum, "Predicate Invention for Bilevel Planning," May 2025, arXiv:2203.09634 [cs]. [Online]. Available: http://arxiv.org/abs/2203.09634
- [13] M. Asai, H. Kajino, A. Fukunaga, and C. Muise, "Classical Planning in Deep Latent Space," *Journal of Artificial Intelligence Research*, vol. 74, pp. 1599–1686, Aug. 2022. [Online]. Available: https://jair.org/index.php/jair/article/view/13768
- [14] C. Chamzas*, M. Lippi*, M. C. Welle*, A. Varava, L. E. Kavraki, and D. Kragic, "Comparing reconstruction-and contrastive-based models for visual task planning," in *IEEE/RSJ International Conference* on *Intelligent Robots and Systems*, Oct. 2022, pp. 12550–12557. [Online]. Available: https://doi.org/10.1109/IROS47612.2022.9981533
- [15] G. Yang, A. Zhang, A. Morcos, J. Pineau, P. Abbeel, and R. Calandra, "Plan2Vec: Unsupervised Representation Learning by Latent Plans," in Proceedings of the 2nd Conference on Learning for Dynamics and Control. PMLR, Jul. 2020, pp. 935–946, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v120/yang20b.html
- [16] A. Zhang, S. Sukhbaatar, A. Lerer, A. Szlam, and R. Fergus, "Composable planning with attributes," in *International Conference on Machine Learning*. Pmlr, 2018, pp. 5842–5851.
- [17] R. Wang, K. Gao, J. Yu, and K. Bekris, "Lazy rearrangement planning in confined spaces," in *Proceedings of the International Conference* on Automated Planning and Scheduling, vol. 32, 2022, pp. 385–393.
- [18] Y. Zhu, J. Tremblay, S. Birchfield, and Y. Zhu, "Hierarchical Planning for Long-Horizon Manipulation with Geometric and Symbolic Scene Graphs," in 2021 IEEE International Conference on Robotics and Automation (ICRA). Xi'an, China: IEEE, May 2021, pp. 6541–6548. [Online]. Available: https://ieeexplore.ieee.org/document/9561548/
- [19] D. Brélaz, "New methods to color the vertices of a graph," Communications of the ACM, vol. 22, no. 4, pp. 251–256, 1979.
- [20] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," Technical report / documentation, 2016. [Online]. Available: https://pybullet.org
- [21] M. Lippi, M. C. Welle, A. Varava, and D. Kragic, "Enabling visual action planning for object manipulation via latent space roadmaps," arXiv:2103.02554, 2021. [Online]. Available: https://visual-action-planning.github.io/lsr/
- [22] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017.
- [23] Bonet Blai and Geffner Hector, "Learning First-Order Symbolic Representations for Planning from the Structure of the State Space," in *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2020. [Online]. Available: https://www.medra.org/servlet/aliasResolver?alias=iospressISBN&isbn=978-1-64368-100-9&spage=2322&doi=10.3233/FAIA200361