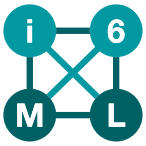

Chair of Machine Learning and Reasoning (Computer Science 6)

RWTH AACHEN UNIVERSITY

This thesis was submitted to the Chair of Machine Learning and Reasoning.

Integrating Symbolic Planning and Continuous Control via Multi-Modal Goal Embeddings

Master Thesis

Presented by

Fabian Hamm
376759

Supervised by Daniel Swoboda, M.Sc.

1st Examiner Prof. Hector Geffner, Ph.D.

2nd Examiner Prof. Dr. Christopher Morris

Aachen, July 20, 2026

Abstract

Long-horizon robotic manipulation requires reasoning over long sequences of interdependent subtasks, demanding both high-level reasoning about which subtasks to achieve and low-level continuous control to achieve them. Pure reinforcement learning struggles to discover long-term structure from raw observations, while symbolic planners reason cleanly about structure but rely on abstractions that are hard to ground in perception. Goal-conditioned contrastive reinforcement learning learns reward-free, goal-directed behavior by aligning states and goals in a shared embedding space, but it commonly conditions on *goal images*, which are impractical to obtain at deployment and entangle *what* should be achieved with irrelevant visual detail. Symbolic and language goals instead offer the compositionality that long-horizon control needs, which makes them a more natural interface.

This thesis asks whether such cheaper, more abstract goals encoded by CLIP carry enough information to condition the same contrastive policy that image goals do, and how much the goal representation alone matters. To study this, it builds an integrated pipeline in which a symbolic planner feeds a language module, CLIP, and finally a contrastive policy, coupling symbolic planning to continuous control through a shared CLIP embedding space. The low-level policy is stable contrastive reinforcement learning on a simulated Sawyer drawer, pick-and-place, and push domain, trained and evaluated with a two-stage image-then-text curriculum across three sub-goal granularities. We compare against a raw-image baseline and several goal-representation ablations.

We find that CLIP-encoded text and symbolic goals can indeed drive the policy, hinting at the potential of abstract goal representations for long-horizon robotic manipulation. Our approach solves all five tasks and matches or beats the raw-image baseline on the drawer tasks; a single fixed policy and sub-goal granularity trails it overall, mainly on the harder composite tasks. Finer sub-goal granularity does not help uniformly but inverts with the goal type — it hurts the image goal yet helps every abstract goal at a medium decomposition level. Adapting the CLIP encoder and the image-then-text curriculum do not improve downstream success — a policy trained from scratch does at least as well as the curriculum — making both matters of convenience rather than performance.

The study is confined to a single simulated domain with a frozen CLIP vision-language backbone and a reproducible five-task protocol following the baseline paper; the symbolic sub-goals are computed directly from the environment state in place of a PDDL planner, off-the-shelf CLIP grounds the rendered scenes only weakly, and there is no transfer to a real robot.

Contents

1 Introduction

1.1 Motivation

Long-horizon robotic manipulation — tidying a workspace, setting a table, assembling parts — remains one of the central challenges in autonomous robotics [11, 22]. Such tasks require reasoning over long sequences of *interdependent* subtasks: deciding *which* subtasks to achieve, and in what order, is a high-level problem, while actually achieving each one is a low-level control problem. The two are coupled — a geometric detail abstracted away at the high level can decide the correct order of the subtasks below it — and the longer the horizon, the more such couplings accumulate. Solving these tasks therefore demands both high-level reasoning about which subtasks to achieve and low-level continuous control to achieve them, and neither paradigm covers both well. Pure reinforcement learning struggles to discover this long-horizon structure from raw observations [12], and the sparse rewards typical of such tasks only compound the difficulty [1]. On the other hand, symbolic and model-based planners reason cleanly about structure but rely on hand-crafted abstractions that are laborious to specify and hard to ground in perception [35]. A natural middle ground is to let a symbolic planner supply the structure and a learned policy supply the control [16] — provided the two can be made to talk to one another.

Goal-conditioned reinforcement learning can be a natural foundation for the low-level part: rather than a hand-designed reward it conditions the policy on a goal representation and contrastive formulations align state-action pairs and goals in a shared embedding space so that the critic's similarity score estimates how reachable a goal is, with no task-specific reward design [9]. These methods often times condition on a *goal image*. A goal image is an inconvenient interface at deployment — one rarely has a visualization of the exact desired outcome to hand — and it entangles *what* should be achieved with visual detail that is irrelevant to it [30]. Symbolic predicates and natural language are, in principle, more natural carriers of an abstract goal, because they are compositional in just the way a planner's sub-goal sequence is [17]. Contrastive image–text models such as CLIP [32] offer a shared text-image space that could serve as a bridge [28] between abstract goals and visual observations, but whether an abstract-goal embedding preserves enough of the goal to drive a visual policy is an open question.

This motivates the central question of the thesis: can a contrastive embedding space act as a shared *interface* between a low-level visual policy and high-level symbolic goals? If it can, an agent could carry out abstract, compositional instructions without a hand-crafted, task-specific

reward. The approach pursued here couples symbolic planning to continuous control through exactly that interface: an assumed planner emits a sequence of symbolic sub-goal states, a language module renders each state as a short textual description, CLIP embeds that description alongside the current visual observation, and a contrastive policy acts on the two embeddings. The pipeline is deliberately not trained end-to-end — the planner is assumed and the encoder is frozen — so that the study isolates the goal interface itself.

1.2 Problem Statement

The policy is conditioned only on embeddings: the critic compares the current state and the goal as vectors in CLIP's shared space, so that space is the entire goal interface, and what the policy can achieve is bounded by what the space preserves about the goal. This creates a difficulty that is specific to abstract goals. An image goal and an abstract goal that describe the *same* target are unlikely to land at the same place in the embedding space: image and text embeddings might occupy separate regions — the *modality gap* [24] — and off-the-shelf CLIP grounds spatial relations only weakly [18], so an abstract goal hands the policy a conditioning signal that can differ from the image goal it stands in for.

The thesis therefore asks two nested questions. First, can the *trained* contrastive critic, learning on top of these embeddings, still recover enough goal-distinguishing structure from an abstract-goal embedding to drive the policy as well as a CLIP-image goal does? Second, once an abstract goal is made to carry exactly the same facts as that CLIP-image goal, is any remaining performance gap a property of the goal *encoding* — how the facts are represented — rather than of missing *information* about which facts are available?

1.3 Research Questions

The problem is decomposed into four research questions, together with a comparison against the raw-image baseline the thesis builds on.

- **RQ1 (Feasibility).** Can a contrastive policy conditioned on the *CLIP embedding* of an abstract goal — a CLIP-encoded sentence or a symbolic predicate vector — reach the goals at all, and how close does it come to the baseline [46] and the upper bound set by an oracle CLIP-image goal — the goal frame encoded into the same space? Four interchangeable goal representations — oracle CLIP-image, CLIP-text, raw predicate vector (`pddl-direct`), and voxel multi-hot — are the vehicle for answering it.
- **RQ2 (Encoding vs. information).** When the language and symbolic goals are matched to express exactly the same predicates, is any performance gap between them a property of the *encoding* — a rendered sentence versus a raw bit-vector — rather than of the *information* the goal carries?

- **RQ3 (Granularity).** Does decomposing a long task into finer sub-goals — one goal per skill, or one per manipulation phase — improve performance over a single whole-task goal, both in the training signal and at evaluation?
- **RQ4 (Pretraining).** Two pretraining choices feed the text-goal policy. Does adapting the CLIP image encoder to the manipulation scenes with LoRA [14] improve text-goal conditioning over off-the-shelf CLIP? And does the two-stage image-then-text curriculum — warm-starting the text stage from the best-checkpoint CLIP-image controller with the state-action encoder frozen — improve over training the text-goal policy from scratch?

The first three questions are the core of this thesis: they ask about the feasibility of the goal interface, the encoding-versus-information distinction, and compare the impact of different goal granularities. RQ4 and a separate, policy-free *template study* (Section 6.6) — which measures how the wording of a language goal shapes its CLIP embedding before any policy is trained — are supporting questions rather than the central focus.

1.4 Scope and Delimitations

The study is deliberately narrow so that the goal-representation question can be answered cleanly: it stays within a single simulated Sawyer manipulation domain (drawer, pick-and-place, and push) with no real-robot transfer, assumes a correct high-level planner, and builds on one offline contrastive-RL learner with a frozen CLIP ViT-B/32 backbone. These delimitations, and what they imply for the results, are discussed in full in the conclusion (Section 7.2).

1.5 Contributions

The thesis makes the following contributions.

- An integrated planner → language → CLIP → contrastive-policy pipeline that couples symbolic planning to continuous control through a shared CLIP embedding space, trained with a two-stage image-then-text curriculum so that one controller accepts language goals at deployment.
- Language- and symbolic-conditioned variants of stable contrastive RL: a CLIP-text goal encoder and two non-neural symbolic goal encoders — a raw predicate vector and a voxel multi-hot — with the symbolic goals read directly from the environment state.
- A controlled ablation that separates goal *encoding* from goal *information*: the predicate vector and the rendered sentence are level-matched to the same facts and the shared state-action encoder is frozen, so that only the goal encoder differs across arms.
- A sub-goal-aware training and evaluation protocol at three granularities (coarse, medium, fine), together with an analysis of how per-sub-goal success composes into whole-task success.

1.6 Thesis Outline

The remainder of the thesis is organized as follows. Chapter 2 reviews the foundations: goal-conditioned and contrastive reinforcement learning, symbolic planning, and CLIP. Chapter 3 situates the work among contrastive goal-conditioned reinforcement learning, multi-modal and symbolic goals, and hierarchical and neuro-symbolic planning. Chapter 4 describes the approach itself: the hierarchical architecture, the high-level planner, the language and vision-language modules, the goal representations, the low-level policy, and the sub-goal granularity at which it is driven. Chapter 5 then sets out the experimental setup and evaluation methodology in three parts: the task domain and training data, the training and model selection, and the evaluation methodology itself (the goal-representation ablation, the rollouts, and the metrics and protocol). Chapter 6 reports the experiments, from the Stage-1 CLIP-image policy the others build on through the abstract-goal studies, and Chapter 7 summarizes the findings, limitations, and directions for future work.

2 Background

This chapter collects the background the rest of the thesis builds on. We first set up goal-conditioned reinforcement learning, then contrastive representation learning, and show how the two combine into a contrastive reinforcement-learning algorithm that estimates a Q-function; we then introduce the CLIP image–text model that grounds the goals in a shared embedding space, and finally summarize the symbolic planning that supplies the sub-goals. The development of goal-conditioned and contrastive reinforcement learning follows Eysenbach et al. [9].

2.1 Goal-Conditioned Reinforcement Learning

Reinforcement learning studies an agent that acts in an environment and learns, from a scalar reward, a policy mapping states to actions that maximizes its long-run return; ordinarily that reward encodes one fixed task. *Goal-conditioned* reinforcement learning generalizes this to a family of tasks: the policy is given a goal alongside the state and must learn to reach *whichever* goal it is handed. This is attractive here because one controller can then pursue many targets, and because a hand-designed per-task reward is replaced by the goal itself — at the cost of having to say how a goal is specified and rewarded, which the following formalization makes precise.

A goal-conditioned reinforcement-learning problem is defined by a state space with states s_t, actions a_t, an initial state distribution $p_0(s)$, dynamics $p(s_{t+1} \mid s_t, a_t)$, a distribution over goals $p_g(s_g)$, and a per-goal reward $r_g(s, a)$. It is naturally read as a multi-task problem in which each task is to reach one goal, so a single policy must learn to reach any goal it is given rather than to maximize one fixed reward.

Rather than hand-design a distance between states and goals, the reward is taken to be the probability density of reaching the goal at the next step [9],

$$r_g(s_t, a_t) \triangleq (1-\gamma)\, p\big(s_{t+1} = s_g \mid s_t, a_t\big). \tag{2.1}$$

This choice removes the need for a geometric metric in the state space and ties the reward directly to the dynamics. Writing $\pi\big(\tau \mid s_g\big)$ for the probability of a trajectory $\tau = (s_0, a_0, s_1, a_1, \dots)$ under the goal-conditioned policy, the objective is the usual expected discounted return,

$$\max_{\pi}\ \mathbb{E}_{p_g(s_g),\, \pi(\tau \mid s_g)}\Big[\textstyle\sum_{t=0}^{\infty} \gamma^t\, r_g(s_t, a_t)\Big], \tag{2.2}$$

and the corresponding goal-conditioned value function is

$$Q^{\pi}_{s_g}(s,a) \triangleq \mathbb{E}_{\pi(\tau|s_g)}\big[\textstyle\sum_{t'=t}^{\infty} \gamma^{t'-t}\, r_g(s_{t'}, a_{t'}) \mid s_t = s,\ a_t = a\big]. \tag{2.3}$$

A convenient way to reason about this value is the discounted state-occupancy measure,

$$p^{\pi(\cdot|s_g)}(s_t = s) \triangleq (1-\gamma)\textstyle\sum_{t=0}^{\infty} \gamma^t\, p^{\pi}_t\big(s_t = s \mid s_g\big),$$

which is the distribution of states the policy visits, weighted so that nearer futures count more. Sampling from it is the same as drawing a time offset from a geometric distribution, $t \sim \mathrm{Geom}(1-\gamma)$, and then taking the state the policy reaches after t steps. Because the same transitions are useful for many goals, it is convenient to average over the target goals, both at the level of the visited states,

$$p^{\pi(\cdot|\cdot)}(s_{t+1} = s \mid s,a) \triangleq \int p^{\pi(\cdot|s_g)}(s_{t+1} = s \mid s,a)\, p^{\pi}\big(s_g \mid s,a\big)\, ds_g,$$

and at the level of the policy marginalized over the goals it was conditioned on,

$$\pi(a \mid s) \triangleq \int \pi\big(a \mid s, s_g\big)\, p^{\pi}\big(s_g \mid s\big)\, ds_g,$$

which lets experience collected for one goal inform the value of others.

2.2 Contrastive Representation Learning

Contrastive representation learning trains embeddings that pull *positive* pairs together and push *negative* pairs apart. A pair (u, v) is positive when it is drawn from a joint distribution $p(u, v)$ — for instance an image and an augmented view of the same image — and negative when u and v are drawn independently from their marginals $p(u)\,p(v)$. Similarity is measured by a *critic* that factorizes as an inner product of two learned encoders,

$$f(u,v) = \phi(u)^{\top}\psi(v),$$

and is trained with the binary NCE objective (also called InfoNCE),

$$\max_f\ \mathbb{E}_{(u,v^+)\sim p(u,v)}\,\mathbb{E}_{v^-\sim p(v)}[\log\sigma(f(u,v^+)) + \log(1-\sigma(f(u,v^-)))],$$

where $\sigma(\cdot)$ is the sigmoid. Intuitively the critic learns to answer whether a candidate v truly goes with u, which is exactly the kind of yes/no question the next section turns into a value estimate.

2.3 Contrastive Learning as an RL Algorithm

The derivation in this section follows Eysenbach et al. [9]. The link between the two ideas is that the goal-conditioned Q-function of Equation (2.3) is, under the reward of Equation (2.1), the discounted probability of reaching the goal,

$$Q^{\pi}_{s_g}(s,a) = p^{\pi(\cdot\mid\cdot,s_g)}\big(s_{t+} = s_g \mid s,a\big). \tag{2.4}$$

Any method that estimates this probability therefore estimates the Q-value, and contrastive learning is well suited to estimating exactly such a probability. The contrastive inputs are chosen accordingly: the first input is a state-action pair $u = (s_t, a_t)$ — the action is included because it shapes which futures are reachable — and the second is a future state $v = s_f$. For a positive pair the future state is drawn from the discounted occupancy measure of the policy,

$$s_f \sim p^{\pi(\cdot\mid\cdot)}(s_{t+1} \mid s_t, a_t), \tag{2.5}$$

so that it is a state the policy actually tends to visit after (s_t, a_t). For a negative pair the future state is drawn independently from the marginal over all state-action pairs,

$$s_f \sim p(s_{t+}) \triangleq \int p^{\pi(\cdot\mid\cdot)}(s_{t+1} \mid s,a)\, p(s,a)\, ds\, da, \tag{2.6}$$

which in practice is a state from a different trajectory. The critic $f\big(s,a,s_f\big)$ is then trained to separate the two with the same binary objective as before,

$$\max_f \mathbb{E}_{(s,a)\sim p(s,a),\, s_f^{\pm}\sim p^{\pi(\cdot\mid\cdot)}(s_{t+1}\mid s,a)}[\mathcal{L}\big(s,a,s_f^{+},s_f^{-}\big)], \quad \mathcal{L} = \log\sigma(f\big(s,a,s_f^{+}\big)) + \log(1-\sigma(f\big(s,a,s_f^{-}\big))). \tag{2.7}$$

Under this construction the contrastive objective has a closed-form maximizer [9]: at the optimum the critic scores a future state by the logarithm of the policy's chance of reaching s_g relative to how common s_g is overall,

$$f^{*}\big(s,a,s_g\big) = \log\Big(\frac{p^{\pi}\big(s_{t+} = s_g \mid s,a\big)}{p\big(s_g\big)}\Big). \tag{2.8}$$

Its exponential then matches the Q-function of Equation (2.4) up to the constant factor $1/p\big(s_g\big)$,

$$\exp(f^{*}\big(s,a,s_g\big)) = \frac{1}{p\big(s_g\big)}\, Q^{\pi}_{s_g}(s,a). \tag{2.9}$$

Consequently, learning encoders $\phi(s,a)$ and $\psi\big(s_g\big)$ so that their inner product $\phi(s,a)\cdot\psi\big(s_g\big)$ matches this critic yields a Q-value estimate without a separate reward model. Figure 2.1 illustrates the sampling scheme.

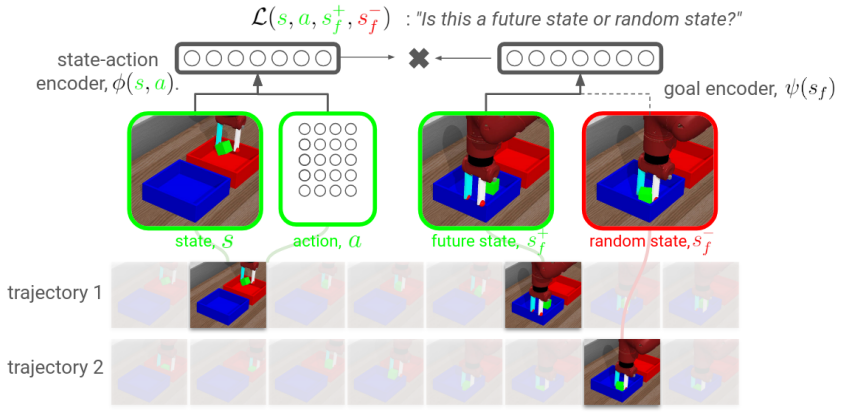

Figure 2.1: Time-step sampling behind the contrastive objective. From an anchor (s_t, a_t), positive goals are future states drawn from the discounted state-occupancy measure ($\Delta t \sim \text{Geom}(1-\gamma)$), while negatives are states sampled from the marginal (typically a different trajectory). The critic learns to score positives above negatives, yielding a Q-proportional similarity. Credit: [9].

This places contrastive reinforcement learning between two familiar alternatives. Hindsight experience replay [1] also conditions on goals and relabels episodes with the goals actually reached, but it keeps a classical, sparse reward; goal-conditioned behavior cloning imitates expert actions by supervised learning and uses no reward at all. Contrastive reinforcement learning instead turns goal reaching into a dense similarity in embedding space, which is what makes a learned embedding usable both as the reward signal and as the interface for abstract goals later in this thesis.

2.4 CLIP

CLIP (Contrastive Language-Image Pretraining) is a contrastive image–text model that learns visual representations from natural-language supervision rather than from fixed class labels [32]. It consists of two encoders that produce embeddings in a shared vector space: an image encoder, realized either as a Vision Transformer that splits an image into patches and processes them with self-attention or as a ResNet, and a Transformer text encoder that maps a tokenized sentence into the same space. Both encoders output normalized embeddings of the same dimension, so an image and a text can be compared directly by cosine similarity.

Training uses a large collection of image-text pairs — on the order of 400 million pairs collected from the web — and a symmetric cross-entropy (InfoNCE) objective [32]. For each pair the image is encoded to $\mathbf{v}$ and the text to $\mathbf{t}$, and within a batch the similarity of each matched pair

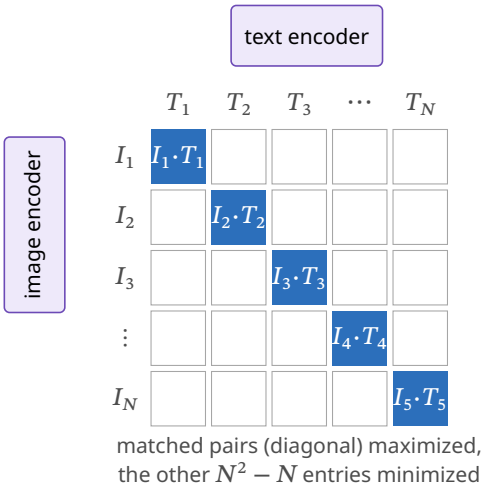

Figure 2.2: CLIP trains an image encoder and a text encoder jointly: within a batch of N (image, text) pairs it computes all $N \times N$ cosine similarities $I_i \cdot T_j$ and, with a symmetric InfoNCE objective, pushes the N matched (diagonal) pairs up and the $N^2 - N$ mismatched pairs down. The result is a shared, normalized space in which matching images and texts are close — the interface used to ground language goals against visual observations in this work. Adapted from [32].

is maximized against all mismatched pairs in both directions,

$$\mathcal{L} = \tfrac{1}{2}(\mathcal{L}_{I2T} + \mathcal{L}_{T2I}), \qquad \mathcal{L}_{I2T} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp(\mathbf{v}_i \cdot \mathbf{t}_i/\tau)}{\sum_{j=1}^{N} \exp(\mathbf{v}_i \cdot \mathbf{t}_j/\tau)},$$

where τ is a learnable temperature and $\mathcal{L}_{T2I}$ is the symmetric text-to-image term (Figure 2.2). The result is a shared embedding space in which an image and a description of it land close together, which is what lets a sentence stand in for a goal image in this work.

Once trained, CLIP performs zero-shot classification by encoding candidate class names as text prompts (for example, “a photo of a cat”) and choosing the class whose embedding is closest to the image, and it supports image-text retrieval in either direction; its embeddings also serve as transferable features for downstream tasks [32]. It is this shared space, rather than any single zero-shot task, that is used here as the interface to ground language goals against visual observations.

One property of that space matters throughout: there is a *modality gap* — image and text embeddings occupy two separate cones, so even a correctly matched image-text pair has a cosine well below one [24]. Absolute cosine values are therefore read against this gap rather than against a notional maximum of one.

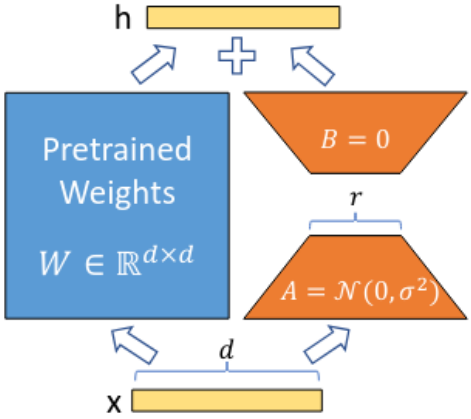

Figure 2.3: Low-Rank Adaptation (LoRA): trainable low-rank matrices A, B are injected into a transformer layer while the original weights W stay frozen, giving parameter-efficient domain adaptation without catastrophic forgetting. Here only the CLIP image encoder is adapted; the text encoder stays frozen so both modalities remain in one shared space. Image credit: [14].

2.4.1 Low-Rank Adaptation

A pretrained encoder can be specialized to a new visual domain without retraining all of its weights. Low-Rank Adaptation (LoRA) [14] freezes the original weight matrix W of a layer and learns a low-rank update $\Delta W = BA$ alongside it, where A and B are small trainable matrices whose shared inner dimension — the *rank* — is far smaller than the dimensions of W. Only A and B are trained, so adaptation touches a tiny fraction of the parameters, and because the base weights are untouched it largely avoids the catastrophic forgetting of a full fine-tune; once trained, the update can be merged back into W so that inference carries no extra cost (Figure 2.3). Applied to CLIP, LoRA can move the *image* encoder toward a new image domain while the *text* encoder is kept frozen, so both modalities remain in one shared space — the setting this thesis uses as a secondary arm (RQ4) to adapt CLIP to the simulator renders.

2.5 Planning

Symbolic planning searches for a sequence of actions that turns an initial symbolic state into one satisfying a goal condition. It reasons over abstract propositions rather than continuous states, which is what makes it a natural source of structure for decomposing a long task into sub-goals. A planning problem is commonly written as a STRIPS tuple

$$\Pi = \langle P, O, I, G \rangle,$$

where P is a set of propositional predicates, O a set of operators, $I \subseteq P$ the initial state, and $G \subseteq P$ the goal condition. Each operator $o \in O$ is a triple $o = \langle \text{pre}(o), \text{add}(o), \text{del}(o) \rangle$ of preconditions,

positive effects, and negative effects. The operator applies in a state $s \subseteq P$ when $\text{pre}(o) \subseteq s$, and its application yields the successor state $s' = (s \setminus \text{del}(o)) \cup \text{add}(o)$. A plan $\pi = \langle o_1, \dots, o_n \rangle$ applied from I then induces a sequence of symbolic states

$$s_0 \xrightarrow{o_1} s_1 \xrightarrow{o_2} \cdots \xrightarrow{o_n} s_n,$$

with $s_0 = I$ and $G \subseteq s_n$. It is precisely this implied state sequence that the thesis treats as a sequence of sub-goals for the low-level policy, with each intermediate symbolic state becoming one goal to reach.

3 Related Work

This chapter places the thesis among three lines of work: multi-modal and symbolic goal representations, contrastive goal-conditioned reinforcement learning, and hierarchical or neuro-symbolic planning. How the present study differs from each of these lines is drawn out within the respective sections below.

3.1 Multi-Modal and Symbolic Goals

One prominent line of work conditions control on goals expressed in modalities other than positions or images. Several methods exploit contrastive language-image embeddings for embodied agents. Khandelwal et al. [20] show that simply fine-tuning policies on top of a frozen CLIP visual backbone can outperform far more specialized navigation and rearrangement systems, underscoring the value of such embeddings for control. Majumdar et al. [28] use CLIP embeddings for zero-shot object-goal navigation, letting an agent move toward objects described by text prompts such as “find the sink”. Shridhar et al. [34] propose Perceiver-Actor, a transformer that conditions robotic manipulation jointly on 3D visual input and free-form language instructions. The examples show that shared representations let goal-conditioned policies be driven by language goals.

The dominant version of this idea is the vision-language-action (VLA) model, which conditions an end-to-end policy directly on language and treats actions as another output modality entirely. RT-1 demonstrates that a Transformer mapping images and free-form instructions to discretized actions generalizes well as data, model size, and data diversity grow, using a large multi-task real-robot dataset [3]. RT-2 extends this by co-fine-tuning an internet-scale vision-language backbone on both web VQA data and robot trajectories, expressing actions as text tokens so that a single model emits language and actions alike, and reports emergent generalization to novel objects, unseen instructions, and basic reasoning [47]. Nair et al. show with LOReL that a language-conditioned success classifier trained on crowd-sourced annotations of offline, sub-optimal data outperforms both goal-image specification and language-conditioned imitation by more than 25% on drawer-opening and related tasks [30] — the same argument this thesis makes for why language and symbolic goals are cheaper and less over-specified than images, evaluated on a closely related task family. Mees et al.’s CALVIN benchmark further shows that flat imitation baselines struggle badly once instructions are unconstrained and horizons grow long [29], motivating precisely the kind of sub-goal decomposition this thesis studies.

A related group moves toward more structured, symbolic goal spaces. Symbolic planners can generate sub-goals that guide reinforcement-learning policies: Illanes et al. [16], for instance, use automated planning to synthesize high-level plans that steer a hierarchical RL agent, learning a separate option, or skill policy, per high-level action and transferring these skills across tasks. Other work uses symbolic planning as a high-level skill-sequence generator, learning symbolic skills from unlabeled play data and building a transition graph in symbolic space from which a planner produces a sequence of visual sub-goals at test time [43]. Scene-graph methods provide another bridge from perception to symbols: Chalvatzaki et al. [4] fine-tune GPT-2 to generate action sequences from scene graphs. Herzog et al. [13] argue that large multimodal models remain weak at granular, domain-specific state grounding, and instead construct a domain-conditioned scene graph and map it directly onto a symbolic PDDL state. Earlier work such as LinkNet [42] extracts relational embeddings from images to construct such graphs.

Finally, several studies probe how well vision-language models support these goals. Action-aware adaptations such as Robotic-CLIP [31] fine-tune CLIP on hundreds of thousands of action videos to capture temporal and semantic correspondences relevant to manipulation. A recurring caveat, however, is that even state-of-the-art models are weak at reasoning about spatial relations [18]. Winoground sharpens this concern by showing that state-of-the-art vision-language models, asked to match two images to two captions built from the identical words in different order, perform close to chance [38], and Yuksekgonul et al. trace this failure to a "bag-of-words" shortcut encouraged by the contrastive pre-training objective itself, showing that composition-aware hard-negative mining is a partial remedy [44]. Song et al.'s RoboSpatial dataset shows that general-purpose vision-language models trained without dedicated spatial and reference-frame supervision underperform on the spatial-affordance and relationship predictions that manipulation actually requires [36]. This has both motivated more spatically capable models such as SIGLIP 2 [39], a multilingual vision-language encoder that improves spatial and semantic understanding, as well as architectures that separate *what* from *where*. Examples include CLIPort [33], a language-conditioned imitation-learning agent that combines the semantic understanding of CLIP with the spatial precision of Transporter networks [45], and spatially supervised models such as SpatialVLM [5], which is pretrained on spatially grounded captions to better capture the relations relevant for robotics.

Where these symbolic-sub-goal and scene-graph methods [4, 13, 16] *use* symbols to drive control, and where the VLA family [3, 47] folds language directly into an end-to-end action decoder, this thesis instead *isolates* goal *encoding* from goal *information* by level-matching a predicate vector and a rendered sentence to the same facts and feeding both through a single frozen CLIP encoder into an otherwise unchanged contrastive critic.

3.2 Contrastive Goal-Conditioned RL

In contrast to the language and symbolic goals above, contrastive goal-conditioned reinforcement learning has developed almost entirely on image goals. Eysenbach et al.'s recursive classification objective, C-Learning, is the direct conceptual precursor of the contrastive formulation: it casts goal-conditioned RL as learning a classifier that predicts whether a future observation was drawn from the policy's future-state distribution, then recovers a goal-conditioned value function from the classifier via Bayes' rule, together with a principled account of the optimal goal-sampling ratio [8].

The starting point for the present thesis is the formulation of Eysenbach et al. [9], who cast goal-conditioned reinforcement learning as contrastive learning. In their approach the state-action pair and the goal are each embedded into a shared space by neural encoders, and the inner product $\phi(s, a) \cdot \psi(g)$ between the two embeddings serves as a learned similarity score that, after further derivation, acts as a value function; the agent then selects the action that maximizes this score for the target goal. An alternative to this classification view is offered by Wang et al.'s quasimetric RL (QRL), which instead imposes the asymmetric, quasimetric geometry of the optimal goal-reaching value function directly on the learned embedding space, improving sample efficiency on both state- and image-based goal-reaching benchmarks [41]; where the contrastive critic used in this thesis learns a similarity *score*, QRL learns a *distance*, illustrating that the geometry imposed on the goal embedding space is itself a design axis independent of what fills that space. Building on the classification view, Liu et al. [26] demonstrate that even a single fixed goal can be enough for manipulation skills such as pick-and-place to emerge through online, on-policy training, without intermediate rewards or demonstrations. Closest to this thesis, Zheng et al. [46] stabilize contrastive reinforcement learning from offline data on vision-based manipulation tasks such as drawer opening and box picking; their method is the direct baseline here. Closer still to the present setting in goal *modality*, Ma et al.'s LIV trains a single vision-language representation from action-free, language-annotated human video contrastively with a dual-reinforcement-learning objective similar to mutual information contrastive learning, so that the same representation implicitly serves as a universal value function whether the target goal is an image or a language description [27]. LIV is thus a prior demonstration that a shared, contrastively trained embedding can support both goal modalities. It differs by deriving its representation and reward from unlabeled human video rather than from a frozen CLIP encoder driven by a symbolic planner, and by learning a reward/value signal for downstream policy learning rather than serving directly as the goal embedding inside an offline contrastive critic. Across these works the factored critic $\phi_\theta(s, a) \cdot \psi_\theta(g)$ tends to outperform hindsight experience replay [1], which instead relabels failed episodes with the goals actually reached while keeping a classical reward. Across all of these methods the goal g is supplied as an image — the assumption this thesis departs from. It keeps the *same* contrastive

critic but conditions it on abstract language and symbolic goals in place of images, asking how much the goal *representation* alone matters when the shared state-action encoder is held fixed.

3.3 Hierarchical and Neuro-Symbolic Planning

Another line of work composes high-level structure with low-level control. A prominent formalization of this composition is task-and-motion planning (TAMP), which interleaves a symbolic task planner with a continuous motion planner so that each symbolic action is realized by a geometrically feasible motion [11]. Because hand-specifying the symbolic operators and feasibility checks is laborious, a growing body of learning-based work replaces parts of the TAMP stack with learned components, for example learning the symbolic operators, samplers, or feasibility predicates the planner relies on [35]. This thesis shares the TAMP aim of coupling symbolic structure with continuous control, but *assumes* the high-level planner and instead studies how the resulting sub-goals are communicated to the low-level policy. Eysenbach et al. [7] propose *Search on the Replay Buffer*, in which the agent builds a graph of previously visited states and runs a planner over it to find feasible paths to a distant goal. Fang et al. [10] instead combine planning with policy learning by setting latent sub-goals that the policy learns to reach, effectively planning in latent space; this work also provides the simulated environment used in this thesis.

A large and fast-moving body of work uses large language models directly as the high-level planner. Ahn et al.'s SayCan restricts an LLM's proposed subtasks to those a learned affordance function scores as feasible in the current state, letting it decompose abstract instructions into sequences of pre-defined skills on a real mobile manipulator [2]. Huang et al.'s Inner Monologue closes the loop further by injecting natural-language success/failure feedback and scene descriptions back into the LLM planner with no additional training, markedly improving long-horizon instruction completion in simulated and real rearrangement and kitchen tasks [15]. Liang et al.'s Code as Policies instead has the LLM emit executable code that combines perception and control primitives with logic and loops, moving spatial-geometric reasoning directly into the generated program [23]. Closer to the symbolic-planner assumption made in this thesis, Liu et al.'s LLM+P translates a natural-language problem into PDDL, hands it to a classical planner, and translates the resulting plan back into language, achieving optimal solutions on problems where pure LLMs fail to produce even a feasible plan [25]. Working the same natural-language–PDDL interface from the opposite direction, Stein et al. [37] use an LLM to automatically convert PDDL problems into natural-language prompts. They find that these generated natural-language prompts elicit stronger planning than the raw PDDL or simple template-based prompts, even though LLM planning still trails classical symbolic planners.

A closely related question is how such sub-goals, however produced, should be sequenced over long horizons. Lee et al. show that naively chaining independently trained skill policies fail once a downstream policy meets a starting state outside its training distribution. They address this by adversarially regularizing each skill's *terminal*-state distribution to remain inside the next skill's initial-state distribution, reporting the first model-free RL method to solve two long-horizon furniture-assembly tasks [22]. Gupta et al.'s Relay Policy Learning instead learns a hierarchy directly from unsegmented human demonstrations via a data-relabeling scheme, in which the low-level policy acts for a fixed number of steps regardless of the goal actually reached [12] — different from the sub-goal execution without a fixed number of steps this thesis uses. Jiang et al.'s use of language as the abstraction layer in hierarchical RL is very close to the approach of this thesis: a high-level policy issues language sub-goals that a language-conditioned low-level policy must reach [17]. This compositional structure of language seems to generalize combinatorially better than non-compositional latent abstractions on temporally extended, pixel-based sorting and rearrangement tasks. More explicitly neuro-symbolic frameworks couple the two more tightly: LOOP [40], for example, treats planning as an iterative exchange between neural and symbolic components, combining graph neural networks for spatial reasoning, hierarchical task decomposition, and a causal memory for iterative PDDL refinement. Keller et al.'s neuro-symbolic imitation-learning framework learns a symbolic abstraction of the state-action space from demonstrations, plans over it symbolically to decompose long tasks into subtasks, and learns neural skills that refine the resulting abstract plan into robot actions [19] — similar to what this thesis studies, but with imitation-learned low-level skills and a learned, rather than assumed, symbolic vocabulary. Finally, Chen et al. [6] show that prompting large vision-language models yields richer representations than raw image embeddings and improves performance on long-horizon reinforcement-learning tasks. Together these results suggest that prompting, symbolic conditioning, and structured goal representations can all serve as effective guidance for goal-conditioned agents.

This thesis builds directly on the environment and learned-sub-goal setting of Fang et al. [10], but *assumes* the high-level planner rather than integrating one, so that its contribution is the goal interface and the granularity at which it is used. Relative to the LLM-as-planner line [2, 15, 23, 25, 37] and the long-horizon skill-composition line [12, 17, 19, 22], the present study is narrower on the planning side and goal-sequencing strategy, but broader on the representation side, since it is the only one of these to hold the low-level contrastive critic fixed while systematically varying the goal's *encoding* (image, symbolic vector, or CLIP-embedded text) and its *granularity*.

4 Methods

This chapter describes the approach: the hierarchical pipeline that turns a symbolic goal into low-level actions, and the design of each of its stages — the assumed planner, the language module, the vision-language encoder, the goal representations, and the contrastive low-level policy.

4.1 Overview of the Hierarchical Architecture

The thesis studies a single integrated pipeline. A high-level symbolic planner emits sub-goal states, a language module renders each as a short PDDL-style or natural-language description, CLIP embeds that text into a space shared with the visual observation, and a contrastive goal-conditioned policy turns the resulting goal and state embeddings into low-level actions (Figures 4.1 and 4.2). Crucially, the pipeline is *not* trained end-to-end: the planner is assumed, the vision-language encoder is frozen, and only the low-level policy is learned, so that each stage — and in particular the goal interface — can be studied in isolation. The decomposition into a high-level planner that emits sub-goals and a low-level controller that learns to reach them is what makes long-horizon control tractable, and the CLIP embedding space is the interface that couples the two: natural-language descriptions are flexible and compositional, yet through CLIP they live in the same space as the visual observations the policy already understands. The language module is, in essence, a serializer of the planner's output — but a purposeful one: instead of a raw symbol string, it renders the active predicates into natural-language (or PDDL-style) text of the kind CLIP was trained to ground, which makes a better goal input than the bare symbols. It is this rendered text, not the predicate set itself, whose effect on the policy the thesis measures, and the fixed templates could later be replaced by a learned generator such as a large language model.

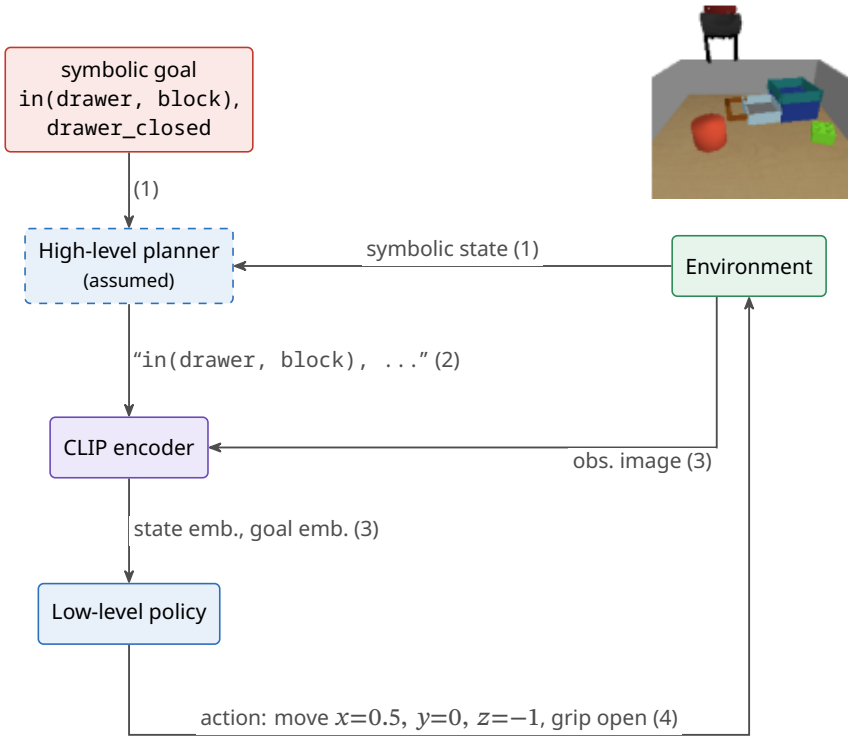

Figure 4.1: The integration spine on the running example “place the small block in the drawer”. (1) the symbolic goal and the current symbolic state are fed to the (assumed) planner, which emits the next symbolic sub-goal; (2) the language module renders that sub-goal as a PDDL-style string and CLIP embeds it; (3) CLIP also embeds the current observation image, and the policy receives the state and goal embeddings; (4) the policy outputs a low-level action executed in the environment, which returns the next symbolic state.

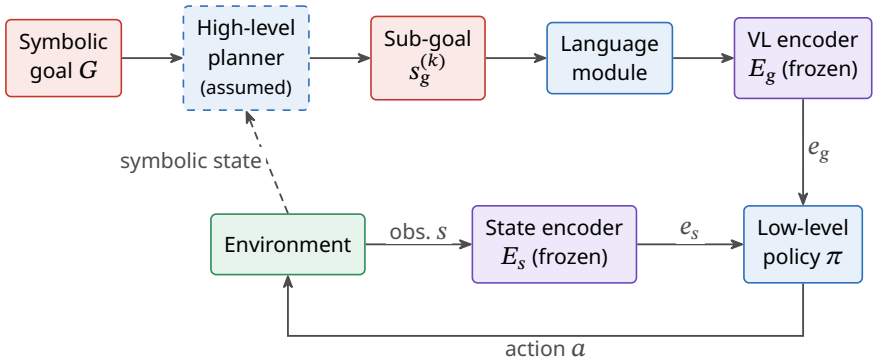

Figure 4.2: End-to-end hierarchical pipeline. A symbolic goal is decomposed by the high-level planner into sub-goals; the language module renders each sub-goal as text, which the frozen vision-language encoder maps into the shared embedding space that conditions the low-level contrastive policy. The policy closes a control loop with the environment through the frozen state encoder.

4.2 High-Level Planner

The high-level planner is assumed rather than contributed: the thesis takes its output as given and studies everything downstream of it. Conceptually, a PDDL or STRIPS planner takes the current symbolic state together with a symbolic goal and returns an action plan, together with the sequence of symbolic states that plan passes through; each intermediate state in that sequence becomes the target for one invocation of the low-level policy. Here the planner is treated as a correct oracle — plan generation itself is out of scope — and the symbolic states are read directly from the environment state. Fixing the planner this way is what lets the study isolate the goal interface: the question is not whether a plan can be found, but whether its sub-goals can be communicated to a continuous policy through a shared embedding space. There are different granularities in which a task can be decomposed into sub-goals. For example, a pick-and-place task can expressed by a single whole-task goal state — a state in which the object is already at its target location — or by a sequence of two sub-goals — one for the pick and one for the place. We call those different levels of *sub-goal granularity*. The next section describes how those predicates are rendered into text.

4.3 Language Module

The Language Module renders the symbolic predicates into text, which can then be embedded by the Vision-Language Encoder. Depending on the sub-goals granularity that the High-Level Planner uses, the predicates which need to be rendered into text are different. For example in a pick-and-place task, the sub-goal sequence of two sub-goals — one for the pick and one for the place — needs a predicate that describes the grasping of the object, which is not needed in a single whole-task goal state. At the same time, it is perfectly valid to also render the whole-task goal state with the ”grasping” predicate, even though it is not needed for the whole-task goal state. We call those different levels of *text-detail*.

Figure 4.3 makes the the different *granularities* and *text-detail* levels concrete on a concrete example. The task is to open the drawer and then push the cylinder. At ”coarse” granularity the policy receives a single whole-task goal; at ”medium”, there are two sub-goals; at ”fine” there are four sub-goals. Each sub-goal is shown with a short text: different from the ”coarse” and ”medium” granularity texts, the ”fine” texts include information needed to describe the intermediate contact states (for example, “gripper at drawer handle”, needing higher *text-detail*).

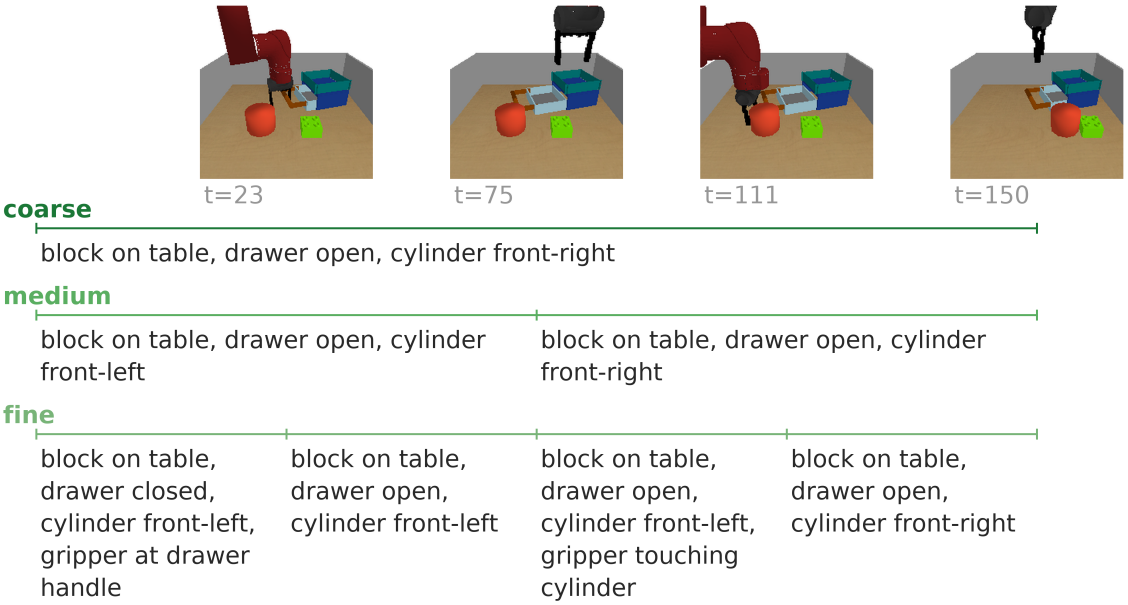

Figure 4.3: A two step task (open the drawer, then push the cylinder) decomposed at the three sub-goal granularities. The images show the (sub-)goal frames. The top row shows the goal frame of each sub-goal; the three rows below mark the sub-goal spans the planner would emit different granularities. Every span is annotated with a short-style sentence rendered from the goal frame's symbolic state. Different text-detail levels are needed.

4.4 Vision-Language Encoder

The vision-language encoder is the bridge between the symbolic goal and the visual policy. CLIP's text encoder E_g maps a goal sentence g to an embedding $e_g \in \mathbb{R}^d$, and its image encoder E_s maps an observation s to an embedding $e_s \in \mathbb{R}^d$ in the same space, so that similarity between the two measures how well the current state satisfies the goal. These embeddings are what parameterize the low-level policy. Four goal encoders are compared in the thesis: CLIP's image encoder, the CLIP text encoder (for language goals), and two non-neural symbolic encoders that read the raw predicate vector or the voxel multi-hot directly.

The backbone is CLIP with a ViT-B/32 image encoder, which produces 512-dimensional embeddings at a 224×224 input resolution. CLIP is chosen because its image and text encoders share one space, so a language goal and a visual observation become directly comparable without any task-specific alignment step, and because its embeddings are cheap to precompute and cache. The encoder is otherwise treated as a swappable component: other image–text encoders were explored during development — notably SigLIP 2 [39], a higher-resolution So400M model trained with a sigmoid rather than a softmax image–text loss — but CLIP ViT-B/32 is used as the default backbone for all reported experiments because it provided better results in early tests. Because CLIP is pretrained on natural web images rather than the simulator renders the

policy sees, its image embeddings need not cleanly separate the manipulation configurations that matter here. As a secondary arm (RQ4) the CLIP *image* encoder is therefore adapted to the rendered domain with a low-rank adapter (LoRA, Section 2.4.1), while the *text* encoder is kept frozen so that the language goals encoded at deployment stay in the same shared space. The adapted encoder is then frozen for the policy training, making the adaptation a preprocessing stage; its training procedure is detailed in Section 5.2.1.

4.5 Low-Level Policy

The low-level policy is a goal-conditioned controller $\pi(a \mid e_s, e_g)$ that acts on the CLIP embeddings of the observation and the goal and outputs a four-tuple action $(x, y, z, \text{open/close})$ — continuous arm motion and a binary gripper action. It is trained with the contrastive actor-critic of Eysenbach et al. and Zheng et al. [9, 46]. The critic scores (state-action, future-goal) pairs through the factored form $\phi(e_{s'}) \cdot \psi(e_g)$, which doubles as the similarity reward, and the actor is a Gaussian policy trained against a twin contrastive Q. In the implementation, ϕ and ψ are small multi-layer perceptrons with layer normalization that map into a shared low-dimensional space (Figure 4.4).

Concretely, the action is a four-dimensional vector — a Cartesian end-effector displacement $(\Delta x, \Delta y, \Delta z)$ together with a scalar gripper action thresholded to open or close. The actor is a diagonal-Gaussian policy that emits a mean and standard deviation over this vector; an action is sampled during training and the mean is used at deployment. The policy sees the observation and the goal only through their CLIP embeddings e_s and e_g, never the raw simulator state, so one architecture serves every goal representation — only the goal-input width differs between the embedding-based and the symbolic arms.

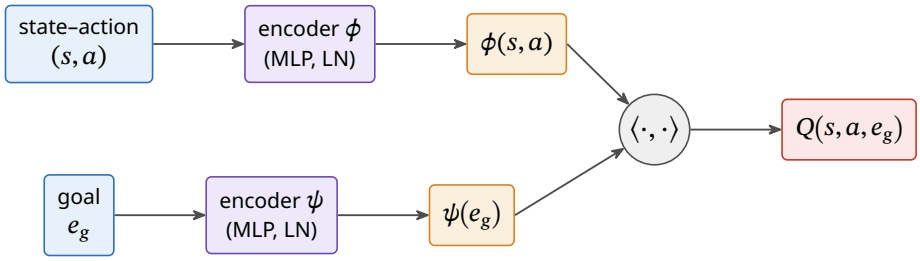

Figure 4.4: Two-branch contrastive critic. The state–action pair and the goal embedding are mapped by separate encoders ϕ and ψ into a shared low-dimensional space; their inner product estimates the goal-conditioned Q-value, which doubles as the similarity reward $\phi(e_{s'}) \cdot \psi(e_g)$.

5 Experimental Setup and Evaluation

5.1 Task Domain and Training Data

This section fixes the concrete setting the experiments run in: the simulated domain and its five evaluation tasks, the demonstration trajectories the policies are trained on and how the symbolic predicates are computed and checked.

5.1.1 Environment and Tasks

The experiments use the simulated manipulation environment of Fang et al. [10]: a 7-DOF Sawyer arm with a parallel gripper on a bordered tabletop, holding a large non-graspable orange cylinder, a small graspable green block, and a drawer, all simulated in PyBullet. Five evaluation tasks are built on this environment (Figure 5.1). Observations are $224 \times 224 \times 3$ RGB renders from a fixed camera — the CLIP input resolution — and the policy conditions on their CLIP embeddings; the action is the four-tuple $(x, y, z, \text{open/close})$, where (x, y, z) is the relative position of the end-effector in centimeters.

The symbolic side uses the two-tier predicate vocabulary of Table 5.2: nine skill-level configuration predicates (for example `in_drawer`, `drawer_open`, and `quadrant_k`) and three sub-skill-level contact predicates (`grasping_small_object`, `gripper_around_handle`, and `touching_large_object`). All predicates have arity at most one under a closed-world assumption and are computed directly from the environment state, with no PDDL planner in the loop (see Chapter 4). Evaluation uses the five fixed tasks of the reference paper, kept for direct comparability.

The Five Evaluation Tasks

The five tasks span the manipulation skills the environment supports — drawer articulation (*drawer*), grasping and placing the small block (*pick-and-place*/*pnp*), and pushing the large cylinder (*push*) — and have both single and composite tasks (Table 5.1). The first two are single-skill tasks: closing the drawer, and picking the block up from the table and placing the block on top of the drawer (unit). The last three are composite, requiring the small block to be picked up from the drawer, placed on top of the drawer *and* the drawer to be closed, the cylinder to be pushed into a target quadrant *and* the drawer to be operated; here the order of the two skills matters, since the cylinder and the drawer can block one another and the small block can only be grasped while the drawer is open. The composite tasks are therefore not just

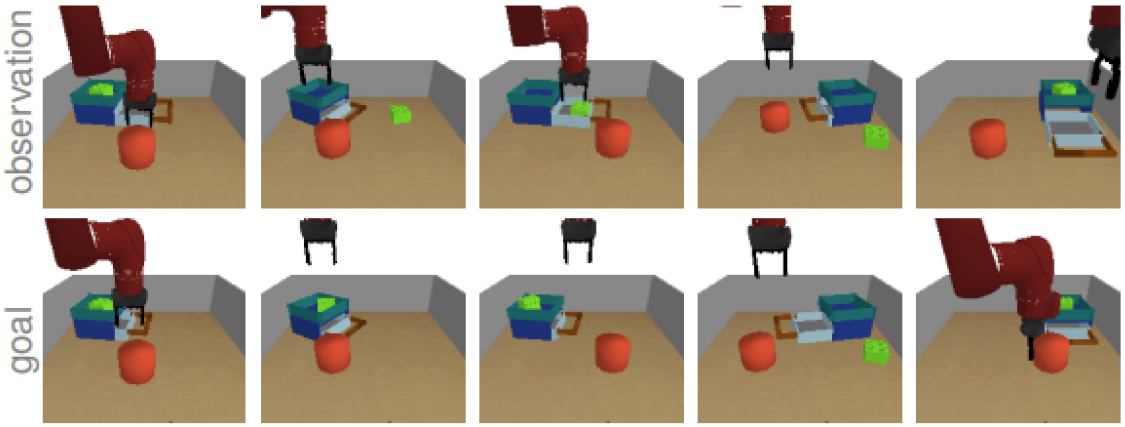

Figure 5.1: The five evaluation tasks, each shown as the initial observation (top row) and the goal configuration (bottom row). From left to right: drawer, pnp, pnp+drawer, push+drawer, and drawer+push (Table 5.1). Credit: [46].

Table 5.1: The five evaluation tasks. The goal is the listed predicate assignment; `quadrant_k` denotes the cylinder's target table quadrant.

Task	Description	Goal predicate(s)
drawer	Close the open drawer	`drawer_closed`
pnp	Pick the block from the table and place it on the top of the drawer	`on_drawer`
pnp+drawer	Pick the block from inside the drawer, place the block on top of the drawer, then close the drawer	`on_drawer` ∧ `drawer_closed`
push+drawer	Push the cylinder to a target quadrant, then open the drawer	`quadrant_k` ∧ `drawer_open`
drawer+push	Close the drawer, then push the cylinder to a target quadrant	`drawer_closed` ∧ `quadrant_k`

a conjunction of two independent skills but require a specific sequence of actions to succeed. The symbolic goal of a task is defined by the goal predicate(s) listed in Table 5.1.

5.1.2 Symbolic Predicates and Goals

When the symbolic planner divides a task into sub-goals, it can do so at different levels of granularity. We chose to evaluate three *levels of granularity*: *medium*, *fine*, and *coarse*. Medium is the level where each skill is a single sub-goal, eg. the pnp+drawer task is divided into two sub-goals: the first describes the state where the block sits on top of the drawer, and the second describes the state where the block is on top of the drawer and the drawer is closed. The fine level divides each skill into two sub-goals, eg. the pnp+drawer task is divided into four sub-goals: the first describes the state where the block is in the drawer, the second describes the state where the block is grasped, the third describes the state where the block is on top of the drawer, and the fourth describes the state where the block is on top of the drawer and the drawer is closed. The coarse granularity level is the level where each task is a single sub-goal, eg.

Table 5.2: The twelve symbolic predicates and how each is detected from the low-dimensional state. The nine skill-level predicates name a resting configuration; the three sub-skill-level predicates name a contact event.

Predicate	Tier	Meaning (geometric test on the state)
`on_drawer`	skill	small object rests on the drawer top
`in_drawer`	skill	small object rests inside the drawer
`on_table`	skill	small object rests on the table surface
`drawer_open`	skill	drawer handle displaced from its closed pose
`drawer_closed`	skill	negation of `drawer_open`
`quadrant_k`	skill	large object lies in table quadrant $k \in \{0, 1, 2, 3\}$ (nearest center)
`grasping_small_object`	sub-skill	end-effector and small object co-located (held)
`gripper_around_handle`	sub-skill	end-effector at the drawer handle (tight tolerance)
`touching_large_object`	sub-skill	end-effector descended onto and inside the large object

the pnp+drawer task is described by a single goal that describes the state where the block is on top of the drawer and the drawer is closed. The coarse level corresponds to having no planner at all, which is what the baseline method uses, and is therefore the level that the baseline method is evaluated on. The higher the granularity, the more detailed the predicate vocabulary needs to be, because finer-grained (sub-)goals require more specific information to be accurately described. This motivates a set of predicates that both support the fine-grained sub-goals and are also compatible with the medium and coarse levels, so that the same predicate vocabulary can be used across all three levels of granularity.

Predicate sets. Table 5.2 lists the twelve symbolic predicates used in this thesis. Nine predicates are motivated by the three primitive skills which make up the five tasks, and three predicates are motivated by breaking each skill into two sub-skills. The skill-level motivated predicates can also describe the coarse granularity level, but the sub-skill-level predicates are only used for the fine-grained sub-goals.

The predicates have arity at most one and we follow the closed-world assumption, so any fact not explicitly true is taken to be false. However, we use both `drawer_closed` and `drawer_open` predicates. This gives us the possibility to explicitly describe the drawer state both with and without negations ("`drawer_open`" instead of "¬`drawer_closed`") in different templates while keeping the implementation logic simple.

Converting predicates to text. To convert predicates into text we chose to use a *template*-based approach, where each predicate is converted into a text description using a fixed template. The strings converted that way get concatenated, optionally with a conjunction symbol and optionally there is a prefix for the whole description defined in the template. Some templates also assign negated predicates a string. Specifically, we created three types of templates: short —

Table 5.3: Predicates and the text fragment each contributes under the three template styles. A sentence is the fragments of the active predicates joined in canonical order (the PDDL style additionally wraps them in `(and ...)`).

Predicate	PDDL	short	long
`on_drawer`	`(on small-block drawer-top)`	block on drawer	A small green block is resting on top of the drawer.
`drawer_open`	`(open drawer)`	drawer open	The drawer is open.
`quadrant_0`	`(at cylinder back-right)`	cylinder back-right	An orange cylinder is in the back-right area of the table.
`grasping_small_object`	`(grasping gripper small-block)`	gripper grasping block	The robot gripper is grasping the small green block.
`touching_large_object`	`(touching gripper cylinder)`	gripper touching cylinder	The robot gripper is touching the orange cylinder.

which uses short phrases, long — which uses longer phrases, and pddl — which uses pddl-like phrases. Table 5.3 shows some examples of the templates.

Since the medium and coarse granularity levels do not require the sub-skill-level predicates, we it is perfectly fine to not convert those into text.

coarse and medium sub-goal granularities dont use sub-skill-level predicates in the goals during rollout except when the template explicitely renders negative predicates. However, during training the the states would be more descriptive which might help the policy learn better.

However, it is interesting to see what happens when we do and how it affects the policy. For that reason we try out both predicates sets (the nine predicates set and the extended twelve predicates set). When using the nine predicate set we call it the *skill-level* text-detail and when using the twelve predicate set we call it the *sub-skill-level* text-detail. Figure 5.2 shows which text detail suffices for which sub-goal granularity.

Each *template style* — PDDL-like, short, or long — is realized by one *template*: a fixed mapping from a predicate to a text fragment, with the fragments joined in a canonical order. The template style is independent of the detail tier (the subject of the template study) and is the only part of the pipeline a learned generator, such as a large language model, would replace.

5.1.3 Demonstration Data

Following [46], our training trajectories come from a scripted expert in the simulated environment. Each rollout iteratively selects one of the three primitive skills and executes it in the current scene. Each skill is rolled for a fixed horizon of 75 steps. A base rollout chains 4 skills in one continuous scene and then resets the scene.

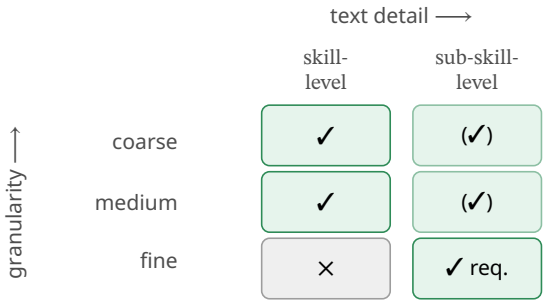

Figure 5.2: Which text detail suffices for which sub-goal granularity. A check marks a sufficient detail level, a parenthesized check an admissible over-specification, and a cross an insufficient one. Coarse and medium goals are skill boundaries that the skill-level tier already names (so they share one detail level); only fine granularity targets mid-skill states that require the sub-skill-level tier.

A subtle but important point is that the simulation runs on between skills while the storage is segmented (following [46]). The simulator reloads a fresh scene only periodically — after a block of four skills, not after each one — and in between it samples a new skill while the object state *carries over* from the previous skill. Consecutive skills therefore unfold without an intervening reset, in one uninterrupted physical rollout, yet each skill segment is written to the dataset as a separate trajectory (Figure 5.4). As a result, no stored trajectory covers a full two-skill task; each is a short scripted primitive segment.

Because the baseline [46] cuts the rollouts into trajectories that span only one skill — and thus never see a full composite task in one trajectory — we limit ourselves to the same. We create two datasets of trajectories, a skill-trajectory dataset (which holds the same type of trajectories as the baseline), and a sub-skill-trajectory dataset that splits each skill into two sub-skills. (Figure 5.3). The idea behind splitting skills into shorter sub-skill trajectories is to improve the sampling rate on finer-grained segments to improve learning on the fine sub-goal granularity. For each policy training, we select one of the two datasets to train on.

Each stored transition keeps the rendered image observation, the low-dimensional simulator state, the frozen-encoder image embedding, and a small pool of precomputed crop embeddings for latent augmentation; a dataset is built from 20 scene variants of 200 rollouts each.

5.1.4 Predicate Detection and Splitting Skills

Each predicate is computed from the low-dimensional simulator state by geometric tests. The nine skill-level predicates are computable following the threshold used by the environment for computing task success. The three contact predicates (`grasping_small_object`, `gripper_around_handle`, and `touching_large_object`) are defined by experimenting with different thresholds. However, dividing skills into sub-skills for the sub-skill-trajectory dataset turned out to be noisy, so we could not directly use them to determine the skill divider. That

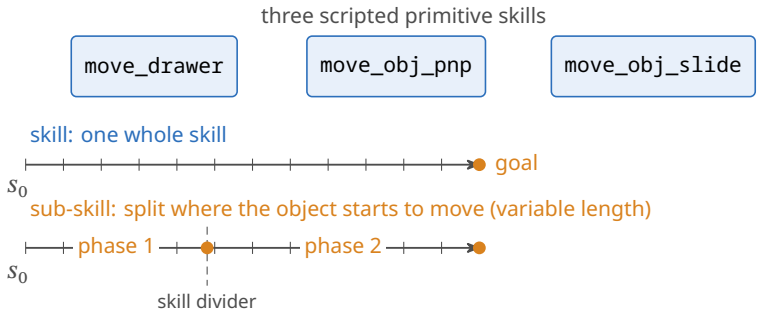

Figure 5.3: The scripted expert issues one of three primitive skills. A *skill* trajectory is a single 75-step skill; a *sub-skill* trajectory splits that skill where the manipulated object first starts to move — the boundary between approaching and manipulating — into two variable-length segments.

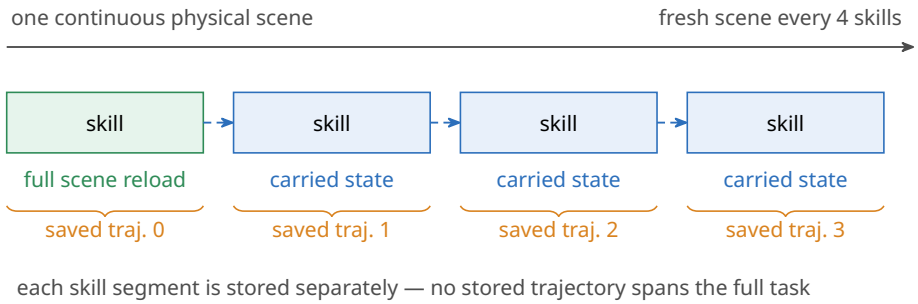

Figure 5.4: The simulator reloads a fresh scene every 4 skills; intervening skills reuse the carried object state, so each 4-skill block unfolds in one continuous scene. Each skill is nonetheless stored as a separate trajectory, so no stored trajectory spans a full composite task.

is why we choose to determine the skill divider by the first frame where the manipulated object starts to move, which is a more reliable signal. The contact predicates are then checked against this divider to ensure that they are true at the divider and not true during the approach phase, as expected from physics. For example, when pushing, the contact predicate `touching_large_object` might fire before being in the final push position so slicing there would cut the skill before the manipulation begins and mislabel the boundary. The object's first motion, by contrast, is the unambiguous physical effect of the skill and is computed independently of the contact predicate, so it doubles as a clean ground truth for checking that predicate.

To determine where to divide a skill, it passes through three events in a fixed causal order (Figures 5.5 and 5.6):

1. the scripted policy reaches its pre-contact *approach waypoint* and flips an internal alignment flag;
2. the gripper makes *contact* and the respective contact predicate turns true;
3. the object *starts to move* — the skill divider.

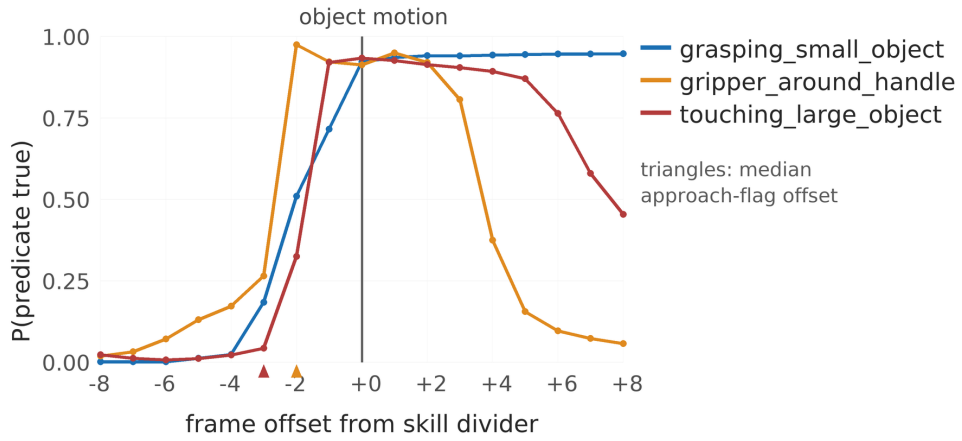

Figure 5.5: Probability that each contact predicate is true as a function of frame offset from the skill divider (at 0), over the demonstration set. *Grasping* and *touching* stay near zero through the approach and step up to about 90 % at the divider; the triangles mark the median position of the scripted approach flag, two to three frames earlier. *Around-handle* is a positional precondition, so it ramps on slightly sooner and falls off once the drawer is pulled.

To check consistency, we evaluate each contact predicate at all frames relative to the skill divider. Across the demonstration set the predicate is true at the divider in over 90 % of skills, is essentially never true deep in the approach, and switches on one to two frames *before* the divider — exactly as physics dictates, since contact precedes motion — while the scripted approach flag sits two to three frames earlier still, recovering the expected ordering of the three events. The contact predicates differ slightly in shape. *Grasping* and *touching* are sharp contact events that step on cleanly at the divider with no approach leakage, whereas *around-handle* is a positional precondition: the gripper reaches the handle and dwells there before pulling, so it ramps on a few frames earlier and falls off once the drawer is open. Both behaviors are correct, and only a loose proximity test would be true throughout the approach. Finally, the contact predicates are transient by design, because a predicate describes the boundary state rather than the whole sub-skill — a grasp ends when the object is released, and a push ends when the gripper retracts. A predicate correctly turning false again later is therefore expected, which is why the check asks only that it hold *at* the divider and stay silent through the approach, not that it persist to the skill's end.

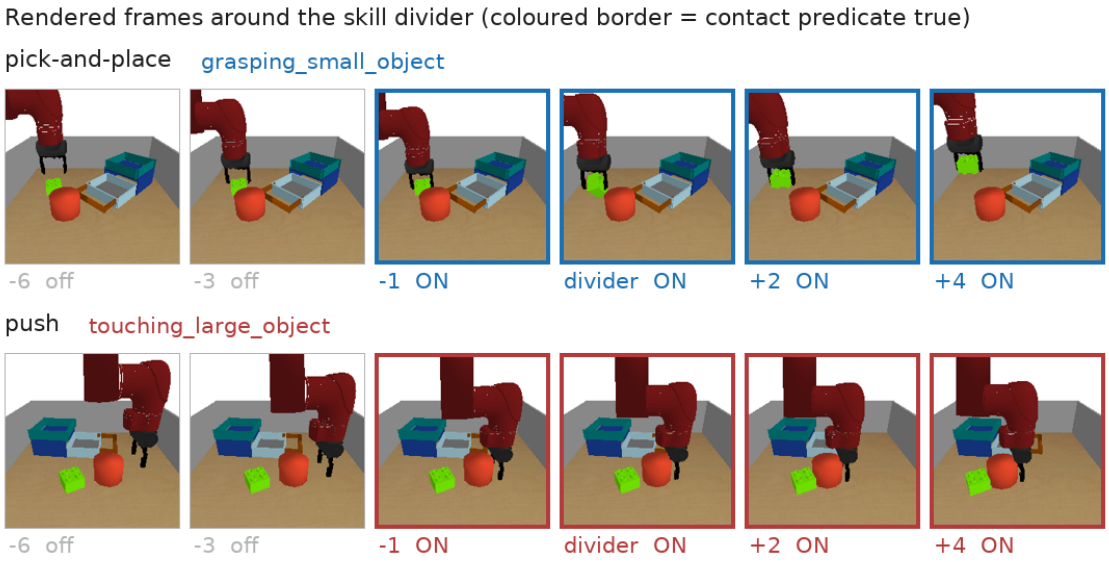

Figure 5.6: Rendered frames around the skill divider for a pick-and-place skill (top) and a push skill (bottom). A colored border marks frames where the contact predicate is true. Through the approach (offsets −6, −3) the gripper is not yet in contact; at the divider and the frames around it the small object is grasped, or the gripper presses into the large object.

5.2 Training and Model Selection

With the data in place, this section describes how the policies are obtained: the two-stage image-then-text training curriculum and the choices around it, and how the base CLIP-image controller is selected from a hyperparameter sweep.

5.2.1 Training Strategy

This section describes the two-stage training curriculum, the augmentation strategy and encoder augmentation.

Two-stage curriculum. The policy is trained in two stages over the *same* contrastive objective, differing only in the goal column. Stage 1 conditions on *CLIP-image* goals; stage 2 continues from the two best-checkpoint stage-1 models on *text*-goals (or other representations in case of ablations) with the state-action encoder frozen. The idea behind the curriculum is that a CLIP-image goal shares a modality with the image state, so early in training it sits naturally close to the state embedding and gives a denser, more direct learning signal. Once the policy has learned to reach goals in the shared space, we freeze the state-action encoder and switch to the sparser text goals which teaches it to interpret symbolic instructions without re-learning control. Because text and image share one CLIP space, the same controller that

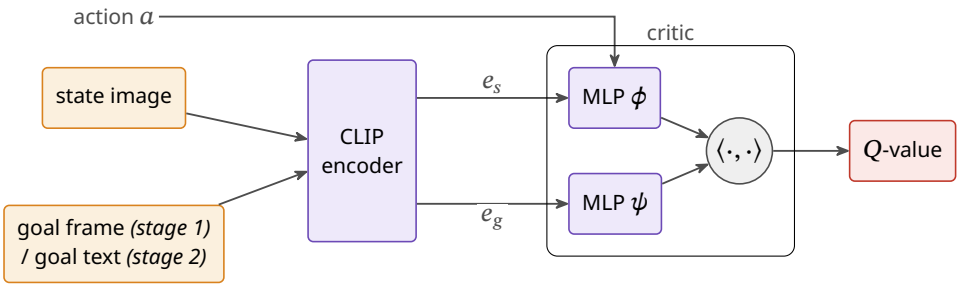

Figure 5.7: The critic conditions on the state image (with the action) and a goal, both embedded by CLIP and mapped by separate MLPs ϕ and ψ whose inner product is the Q-value. The two training stages differ only in the goal column: *stage 1* uses *goal images*, then *stage 2* switches to *goal text* while the state-action encoder stays frozen — the image-then-text curriculum that lets one controller accept language goals at deployment. Adapted from [46].

accepts image goals in the first stage also accepts language goals in the second stage without having to retrain the goal-encoder from scratch. (Figure 5.7).

Sweep and Model Selection The base Stage-1 CLIP-image controller is chosen from a sweep of 48 runs that crosses five factors — the trajectory dataset (skill or sub-skill), batch size (1024 or 2048), the hidden-layer configuration ($512 \cdot 512 \cdot 256$, 1024×4, or 2048×4), learning rate (3×10^{-4} or 1×10^{-4}), and the augmentation mode (augmenting the observation only, or both observation and goal) — at a single training seed, with every checkpoint scored on the five evaluation tasks, where only the final goal is given as the goal. Each run is trained for a fixed budget of 500 epochs.

All checkpoints get ranked by best cross-task mean success; the two best checkpoints of different runs are carried into the text-goal stage, where they warm-start the Stage-2 policies. Each of the two checkpoints starts a sweep over two factors — text-template (two per template style, that is six in total) and text-detail level (skill-level and sub-skill-level) — for a total of 12 Stage-2 runs per Stage-1 checkpoint.

Augmentation. We follow the baseline and use random crop augmentation. The crop augmentation is applied to the rendered images before they are passed through the frozen CLIP encoder, and the resulting embeddings are used for training the policy [46]. The augmentation is only applied during training, and at evaluation time, the original unaugmented images are used. While the baseline pads with edge replication and then randomly crops, we instead found upscaling through interpolation and then cropping to be more effective. In initial tests we saw that the edge-replication artifacts seemed to confuse the CLIP encoder. For efficiency, we precompute a small pool of ten crop embeddings for each frame in the dataset, and sample from that pool during training.

Our crop augmentation of the latent observations regularizes *only* the behavior-cloning term of the actor loss; the contrastive critic and the actor-Q term always train on the unaugmented latents, following the stabilizing techniques of Zheng et al. [46]. The reasoning is that the critic's similarity matrix *is* the reward target — its diagonal positives and off-diagonal negatives define goal reaching — so perturbing its inputs would distort that target, and the actor-Q term back-propagates through the critic, so it must see the same clean inputs. Behavior cloning, by contrast, is plain supervised regression to the demonstrator's action, where an input crop is a standard regularizer that improves robustness without touching the reinforcement-learning targets [21]. Within that behavior-cloning term, *which* inputs are augmented is a hyperparameter that we try out: an *observation-only* variant augments the observation and keeps the goal unaugmented (the conservative choice, since the goal also serves as the contrastive reward target), whereas an *observation-and-goal* variant augments both, as baseline trainer does. Both leave the critic and actor-Q on clean latents, and the sweep runs both so the choice is decided empirically rather than assumed.

Encoder adaptation. In addition to off-the-shelf CLIP as a vision-language encoder, we also pretrain a CLIP image encoder so it adapts to the simulator renders using LoRA (Section 2.4.1); only the image encoder is adapted and the text encoder is left frozen, so both modalities stay in one shared space. Training uses a symmetric InfoNCE loss over a batch of (render, sentence) pairs drawn from the demonstration rollouts: each rendered frame is paired with the sentence rendered from its *true* symbolic state, and the matching sentence must score above every other sentence in the batch, and vice versa — so the adaptation reuses the same symbolic-to-text module as the rest of the pipeline, with the other sentences in the batch as negatives. No explicit hard-negative mining is used: because every frame and sentence comes from the same narrow manipulation domain, the in-batch negatives are already closely related to the positive — they differ only by which object moved or which sub-goal is active — so a large batch supplies fine-grained contrast on its own, at the cost that near-identical goal states may be separated less sharply than under explicit negative-mining.

5.3 Evaluation Methodology

This section sets out how the trained policies are evaluated: the two rollout protocols, the metrics and the peak convention used to summarize a run, the selection-versus-reporting pipeline, and the three ablation studies — goal representation, encoder adaptation, and curriculum — that support the research questions (Section 1.3).

5.3.1 Evaluation Rollouts

We use two types of evaluation rollout, *sequence* and *isolation*. A sequence rollout runs the full ordered sub-goal sequence from a canonical start; an isolation rollout runs each sub-goal on its

own, from its canonical start state, so it is scored from a correct predecessor. Isolation rollouts apply only at the medium and fine granularities, where a task has more than one sub-goal; we denote these two isolation cells *iso-medium* and *iso-fine*, respectively. Sequence rollouts apply at every granularity, so a coarse sequence rollout is just a single whole-task goal. In both cases the policy always receives the current sub-goal as its goal input, and that goal advances to the next one as soon as it is reached. A sequence rollout ends when the final sub-goal is reached or after at most 400 steps; an isolation rollout ends when its sub-goal is reached or after 150 steps. Success is read from the final frame and judged by the simulation environment, not by the symbolic state.

5.3.2 Metrics and Protocol

Unless stated otherwise, "success" is the *final-frame* overall-task success: the environment's task predicate evaluated at the last step of a rollout, reported per-task rate or cross-task mean over the five evaluation tasks. Because a run's success is strongly non-monotonic over training (Section 6.1), every rate uses the *peak convention*: the best checkpoint per cell, and, where several runs or template tiers share a cell, the mean over them. This is a reporting convention only — the held-out diagnostics below are never used to select a checkpoint.

Three further views are added where a question calls for them. The *granularity-aware* metrics decompose a long-horizon task both per isolated sub-goal (can each one be reached on its own?) and as a full ordered sequence (does the chain complete?), which localizes *where* performance is lost. A *qualitative* analysis reads the rollouts directly — annotated filmstrips of representative successes and failures and a taxonomy of the recurring failure modes — turning a low rate into an account of *what* the policy does wrong. Finally, a set of *diagnostics* reports the health of the critic and the language module rather than task success: the contrastive categorical accuracy and the goal-conditioned behavior-cloning (GCBC) loss during Stage-1 training, together with the GCBC loss on an out-of-sample set of held-out rollouts from a disjoint generation seed — whose growing gap to the training loss flags overfitting but is only monitored, never used for selection — and the image–text cosine alignment of the template study (Section 6.6).

The raw-image contrastive-RL baseline of Zheng et al. [46] (also called the *raw-image baseline*) is the external reference the abstract-goal method aims to match or beat, and appears wherever a headline success is reported; the oracle CLIP-image goal is the practical upper bound in the encoding and granularity comparisons. Table 5.4 collects these metrics.

Determinism. The evaluation tasks and training seeds are fixed and the CPU and GPU random number generators are made deterministic, so runs are reproducible. For efficiency a rollout is terminated early once its task predicate is satisfied.

Table 5.4: The reported metrics and their definitions. "Final-frame" means the goal predicate is evaluated at the last rollout step; all rates are per-task means then a cross-task mean.

Metric	Type	Definition / use
Sequence success	primary/granularity	whole decomposed chain completes
Sub-goal isolation success	granularity	each target sub-goal reached in isolation
Qualitative failures	qualitative	annotated rollout filmstrips + failure-mode taxonomy
Categorical acc. / GCBC-loss	diagnostic	critic categorical accuracy, GCBC loss train vs. held-out
Image-text cosine	diagnostic	language-module alignment (Section 6.6)

5.3.3 Evaluation Scheme

Training, evaluation and selection can be seen in Figure 5.8. *Selection&training-time* evaluations narrow the search space defensibly — which templates and which base policies advance — the two training stages then fit the policies, and *reporting-time* evaluations answer the research questions.

The*template eval* is an offline, policy-free pass that scores each candidate sentence by how well it separates the goals in CLIP's shared embedding space, keeping the two strongest versions of each template style (Section 6.6.1); it vets the language module before any policy is conditioned on it and fixes the templates the text stage is trained on. A *stage-1 selection eval* then ranks the sweep cells on their in-loop cross-task mean success and carries the two best CLIP-image configurations into the text stage (Section 5.2.1). The language goals themselves are *not* selected against a success metric: every shortlisted template is trained and reported, so no phrasing is quietly dropped for under-performing.

The *final whole-task eval* gives the headline success per configuration; the *sub-goal and sequence eval* (Section 5.3.1) decomposes that number to localize *where* a long-horizon task breaks; the *goal-representation ablation* (Section 5.3.4) swaps in the four goal encodings to separate *encoding* from *information* (RQ2); and a *zero-shot language-transfer* check evaluates the carried Stage-1 policies on CLIP-text goals with no text training, measuring the cross-modal gap directly (Figure 6.6). These four share the frozen stage-1 state-action encoder; only the first three additionally use the stage-2-trained goal encoder (Figure 5.8). Two further comparisons which are not depicted in the figure probe the *training* itself rather than the goal, and so retrain the policy instead of reusing the frozen encoder: a *curriculum ablation* retrains stage 2 from scratch — no warm start and no frozen encoder — to read off the gain of the image-then-text curriculum over the warm-started policy (Section 5.3.6), and an *encoder-adaptation eval* (RQ4) carries a LoRA-adapted CLIP image encoder through the identical pipeline to test whether adapting CLIP to the rendered domain improves downstream success (Section 5.3.5).

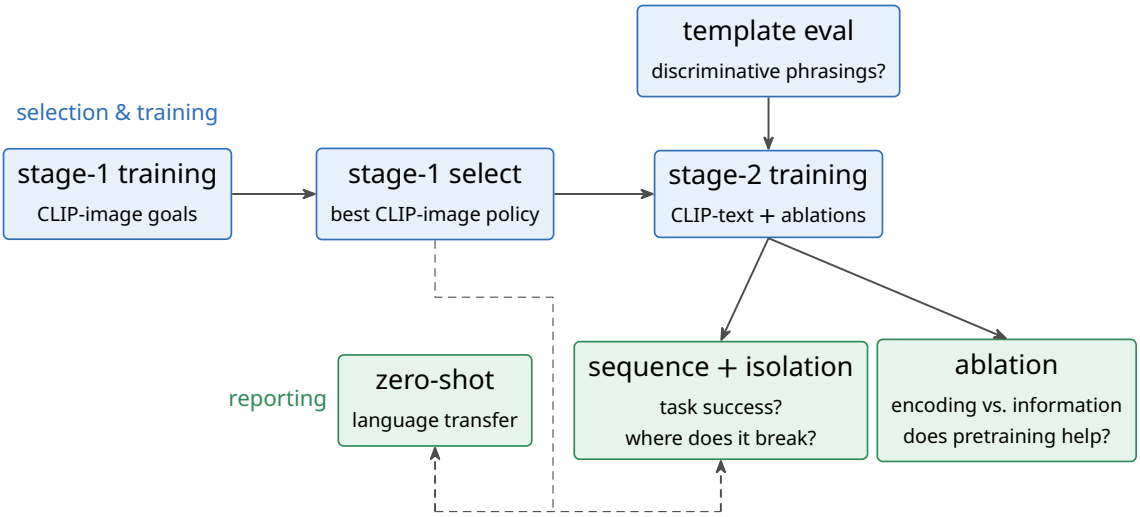

Figure 5.8: The evaluation scheme. Stage-1 training and the template eval feed *selection*: the best CLIP-image policy is carried into stage-2 training, which fits the CLIP-text goal encoder and the goal-representation ablations. The *reporting* evaluations then answer the research questions — the sequence and sub-goal isolation evaluation, and the goal-representation ablation are run on the stage-2-trained policy (solid arrows), while the *zero-shot* check evaluates the selected stage-1 policy on CLIP-text goals with no text training. The dashed arrows mark the frozen stage-1 state-action encoder that every reporting evaluation shares.

5.3.4 Goal-Representation Ablation

The goal-representation ablation asks whether an abstract goal can drive the policy as a CLIP-image goal does, and pins down what a gap means.

Two image references must not be conflated. The *raw-image baseline* is the original contrastive-RL method, which conditions the policy directly on a raw goal *frame* — its pixels — with no CLIP encoding; it is the external method the abstract-goal arms aim to match or beat. The thesis's own image goals are never raw pixels: a goal frame encoded into CLIP's shared space is a *CLIP-image goal*, and as the most direct goal the interface allows it is the *oracle CLIP-image goal*, the practical upper bound. It enters this ablation as one of the interchangeable goal representations rather than as the baseline. Throughout the thesis, then, "goal frame" denotes the raw rendered pixels, "raw-image baseline" the external policy conditioned on them, and "CLIP-image goal" the same frame once encoded by CLIP.

The four representations are the CLIP-text goal (the thesis's main approach), the oracle CLIP-image goal (hypothesized to be the upper bound), the raw predicate one-hot encoding (`pddl-direct`), and a *voxel multi-hot* of the task-relevant object cells (Figure 5.9); they span the spectrum from a fully visual goal to a purely symbolic one.

The voxel goal is the only non-obvious one: it discretizes each task-relevant object's position into a coarse workspace grid and concatenates the per-object one-hot cells into a 700-dimensional

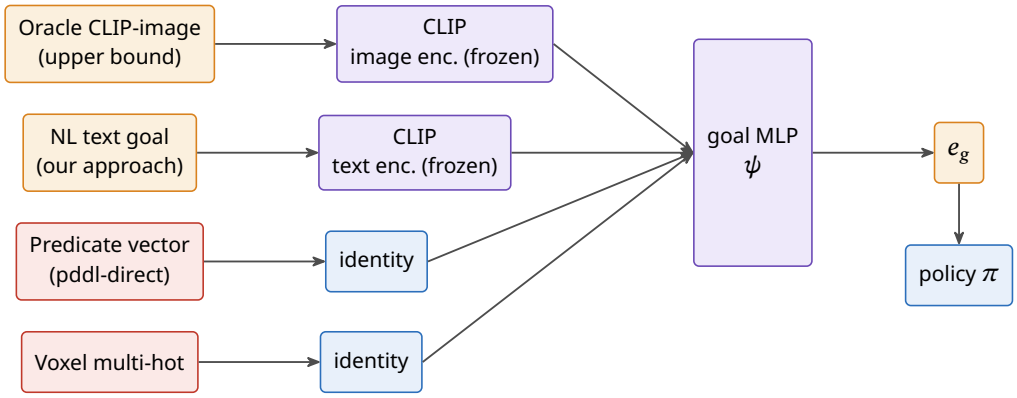

Figure 5.9: The four goal representations (Section 5.3.4) share one frozen state-action encoder and one downstream goal MLP ψ; only the goal column changes. Image and text goals pass through frozen CLIP encoders; the symbolic predicate vector and the voxel multi-hot enter directly. Level-matching the predicates to the language reference makes any performance gap attributable to *encoding* rather than missing *information*. The state-action encoder is omitted in the diagram for clarity, but it is shared across all four goal representations.

multi-hot with exactly three active entries (Figure 5.10). Two things are held fixed so that any difference can only come from the goal encoding.

First, the goals are *information-matched* (RQ2): the predicate vector and the rendered sentence are produced from the *same* active predicates, at the text-detail level the granularity demands (Figure 5.2), so the symbolic and language goals state identical facts and any gap between them reflects the *encoding* rather than missing information. Second, the *policy is held fixed*: all four representations run with the same frozen stage-1 state-action encoder and a single downstream goal MLP, so only the goal representation changes (the frozen-encoder stage-2 procedure is described in Section 5.2.1). A performance difference is then attributable to the goal representation alone, rather than to an incidentally different policy. This attribution comes at a cost: the curriculum ablation of Section 5.3.6 shows that a policy trained from scratch on the goal representation outperforms the warm-started, frozen-encoder one. Freezing the state-action encoder is therefore what makes the comparison clean, and the ablation runs on a deliberately conservative substrate as a result.

5.3.5 Encoder Adaptation Evaluation

Pretraining is judged on two levels that answer different questions. The *intrinsic* metrics ask whether the adapted encoder represents the rendered domain better; the *extrinsic* metric asks whether that better representation actually helps the policy. The intrinsic metrics are necessary but not sufficient — a sharper embedding is only useful if it improves the policy — so the downstream task metric is the one that decides.

Table 5.5: The goal representations evaluated in this thesis, ordered from fully perceptual to purely symbolic. The four representations share one frozen state-action encoder and a single downstream goal MLP; only the goal column differs (Figure 5.9). The raw-image baseline is the external reference — a separate policy with no encoder — and the oracle CLIP-image goal is the upper bound. CLIP-text is the thesis's approach; `pddl-direct` and voxel are the non-neural ablations, level-matched in information to the language goal.

Representation	Goal input	Dim.	Nature	Goal encoder
Raw-image baseline	raw goal frame	224×224×3	perceptual (pixels)	none (CNN policy)
Oracle CLIP-image (ceiling)	CLIP embedding of goal frame	512	perceptual, upper bound	CLIP image (frozen)
CLIP-text (ours)	CLIP embedding of goal sentence	512	semantic / language	CLIP text (frozen)
`pddl-direct`	0/1 predicate vector	9–12	relational / symbolic	identity
voxel	object-position multi-hot	700	geometric / metric	identity

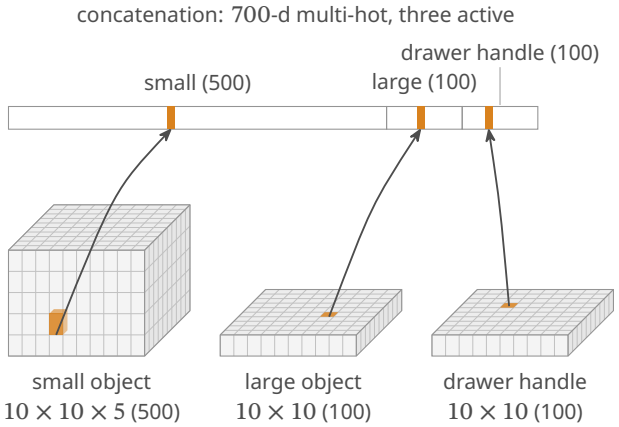

Figure 5.10: The voxel goal representation discretizes each task-relevant object's position into a coarse workspace grid and concatenates the resulting per-object one-hot cells. The small graspable object is binned in three dimensions ($10 \times 10 \times 5 = 500$ cells, the height axis separating on-table, on-drawer, and in-drawer); the large object and the drawer handle are binned in the ground plane only ($10 \times 10 = 100$ cells each). The concatenation is a 700-dimensional multi-hot with exactly three active entries — a purely geometric, non-semantic goal encoding that states only *where* the objects should be.

Two intrinsic checks are used. The primary one is a *retrieval lift*: on a held-out split of (render, sentence) pairs — held out by whole rollout, so no frame of an evaluation trajectory is seen during adaptation — top-1 in-batch image-to-text retrieval accuracy is measured for the adapted encoder and compared against the same metric for the unadapted, zero-shot CLIP encoder, so that a positive lift is direct evidence that the adaptation captured rendered-domain structure the off-the-shelf encoder missed. Because many frames of a rollout share an identical templated sentence, the in-batch metric is capped well below one, so it is read as a relative lift rather than

an accuracy target. Second, the gap between training and held-out retrieval is monitored as an over-fitting check, and, since the text encoder is frozen, image-to-text retrieval staying high also confirms that adapting the image encoder did not pull its embeddings out of the shared space the text goals depend on. Good adaptation therefore means retrieval up over zero-shot with a small train/held-out gap.

The decisive question for RQ4 is extrinsic: whether conditioning the policy on adapted-CLIP image embeddings beats the unadapted baseline on *task success*. Each pretraining variant is carried through the identical policy-training pipeline — its own skill-span stage-1 hyperparameter sweep and model selection, mirroring the off-the-shelf arm, so the adapted encoder is judged on a policy chosen on its own merits rather than on the run-configuration chosen by the models trained with the unadapted encoder. The adaptation is judged to help only if downstream success at least matches, and ideally exceeds, the baseline across tasks. Because training a policy for every hyperparameter setting is prohibitively expensive, the LoRA configuration — adapter rank and learning rate — and the training epoch are all selected on the held-out retrieval metric rather than on the last epoch, over a small grid. Downstream task success is then measured only for that single selected encoder and is the quantity that decides RQ4.

5.3.6 Curriculum (Cold-Start) Ablation

The two-stage curriculum (Section 5.2.1) warm-starts the text stage from the best-checkpoint Stage-1 CLIP-image controller and freezes its state-action encoder. To measure what that curriculum buys, we compare it against a *cold-start* policy of the same architecture trained on the text goals from scratch — no warm start and no frozen encoder — so the benefit of reusing the Stage-1 controller can be read off directly. Both architectures are swept over the *same* twelve text templates (the three template styles short, long, and PDDL, two templates each, at skill- and sub-skill-level detail) and compared at a single architecture, so the contrast is averaged over identical templates rather than a single prompt. Both are read under the peak convention at every granularity, and the outcome — whether warm-starting and freezing the encoder helps downstream success — is reported with the pretraining results.

5.3.7 Template Study

The template study is a secondary, policy-free diagnostic that characterizes how a template's wording shapes the goal embedding *before any policy is trained*. A language goal is produced by a template acting on the active predicates, in one of the three template styles (PDDL-like, short, or long). Because the policy's state is a CLIP image embedding and the critic compares state and goal in CLIP's shared space, a template is judged "good" to the extent that it turns distinct symbolic goal states into goal embeddings that are (i) *discriminable* — distinct goals must not collapse onto one another, or the policy cannot tell them apart; (ii) *aligned* — a language goal should land near the image embedding of the goal it must match; and (iii) *distinct* in text

space — the sentences themselves should spread different goals apart. All three are intrinsic properties, measured without training a policy by encoding rendered goal frames and their templated sentences with CLIP. Because these metrics are only a weak proxy for task success — the decisive evidence is the downstream evaluation — the study is used as supporting context rather than a headline result. Its role in the pipeline is to fix the template wording: it ran before the main experiments, scoring the full grid of phrasing variants on these intrinsic metrics and keeping the two strongest of each style for the text stage (reported in Section 6.6).

6 Results

6.1 Stage-1 CLIP-Image Goal Policy

Every experiment in this chapter builds on a contrastive policy that is first trained to reach *CLIP-image* goals, selected from the 48-run sweep described in Section 5.2.1. Before turning to abstract goals, this section reports the outcome of that sweep: which factors decide success, how the runs train, how uneven success is across the evaluation tasks, and which checkpoints are carried into the text-goal stage.

What works. Figure 6.1 shows a clear ordering of the factors. Trajectory granularity is decisive: models trained on longer skill trajectories reach a mean peak success of 0.51 (best 0.76), whereas models trained on the short sub-skill trajectories never leave the floor at 0.02 (best 0.06). We hypothesize that these short trajectories are too short to learn meaningful representations that generalize to the full task, and that the longer skill trajectories provide a more informative training signal. Network capacity is the next strongest lever — peak success climbs from 0.34 for the small $512 \cdot 512 \cdot 256$ network to 0.58 for 1024×4 and 0.61 for 2048×4. Augmenting the goal as well as the observation helps modestly (0.54 versus 0.48), and the larger learning rate is slightly preferable (0.55 versus 0.47); batch size makes essentially no difference. In short, a skill-level, wide-network, goal-augmented configuration is what a strong Stage-1 policy needs. These trends echo the ablations of the contrastive-RL baseline [46], which likewise favors a wider MLP, a large batch, and augmenting both the state and the goal image.

Training dynamics. Figure 6.2 looks inside three representative runs. The contrastive critic trains successfully everywhere — even the stalled sub-skill run reaches a comparable categorical accuracy — so the failure of the weak runs is one of control, not representation. The middle panel is the clearest diagnostic: the training GCBC loss falls monotonically, but the out-of-sample held-out BC loss bottoms out early and then moves upward, which can be a sign of overfitting. By design, we use these metrics purely as a diagnostic, never as a selection criterion.

Task-to-task spread. A single cross-task mean hides how uneven success is across the evaluation tasks. Figure 6.3 draws one bar per task for the headline models: even the strongest policies range from as low as 0.3 on some tasks to a perfect rate on others. The five tasks behave like five different task instances, and no single model dominates all of them — motivation for reporting cross-task means with their spread throughout.

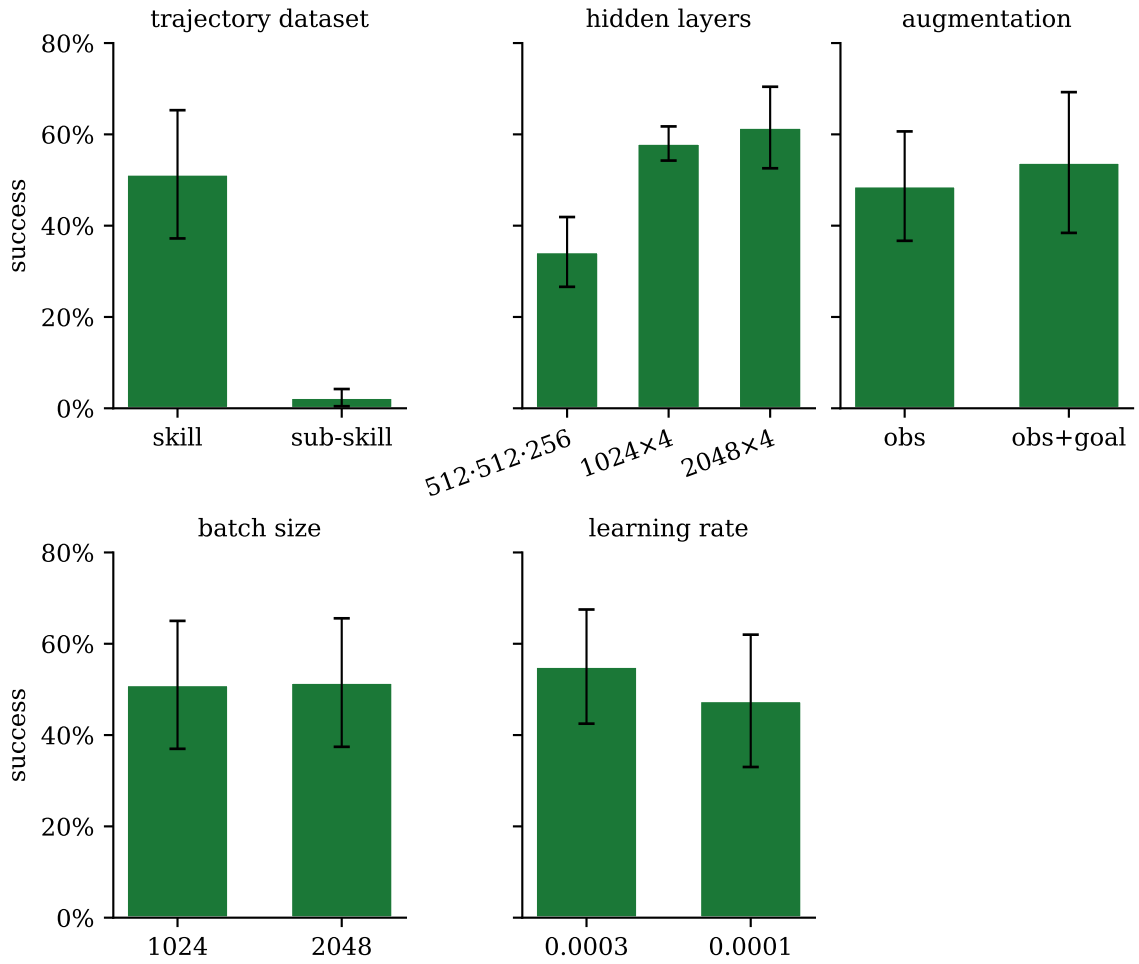

Figure 6.1: Marginal effect of each swept factor on the best checkpoint success of each run. The trajectory dataset panel is over all 48 runs; the remaining panels are restricted to those 24 runs, that were trained on the skill trajectory dataset. Bars are group means, error bars one standard deviation across the runs in the group.

Checkpoint and model selection. The two runs carried into Stage-2 are therefore run A (run 09 in Table 6.1; 2048 × 4, best success 0.76) and run B (run 23; 2048 × 4, 0.70), which both peak at epoch 360 — the very checkpoint that seeds the warm-started Stage-2 policies. Each carried model seeds not a single Stage-2 run but a *sweep* of twelve warm-started text policies — template style (short, long, `pddl`-worded) × version $\{0, 1\}$ × text-detail (skill- or sub-skill-level). A single warm-started Stage-2 run is written `A_style_version` (for example `A_pddl_0`), and a value pooled over a model's whole text sweep is labeled *warm-started from A*. Figure 6.4 traces each carried model's per-task CLIP-image goal success over the training: success is strongly non-monotonic and *unstable*. This is in line with the known instability of contrastive RL that Zheng et al. try to improve [46]. Comparing the success curves of our CLIP-image models to those of the raw-image baseline (Figure 6.5) shows that the instability is not a property of the CLIP-image goal but rather of the contrastive RL training itself. Our two CLIP-image goal runs

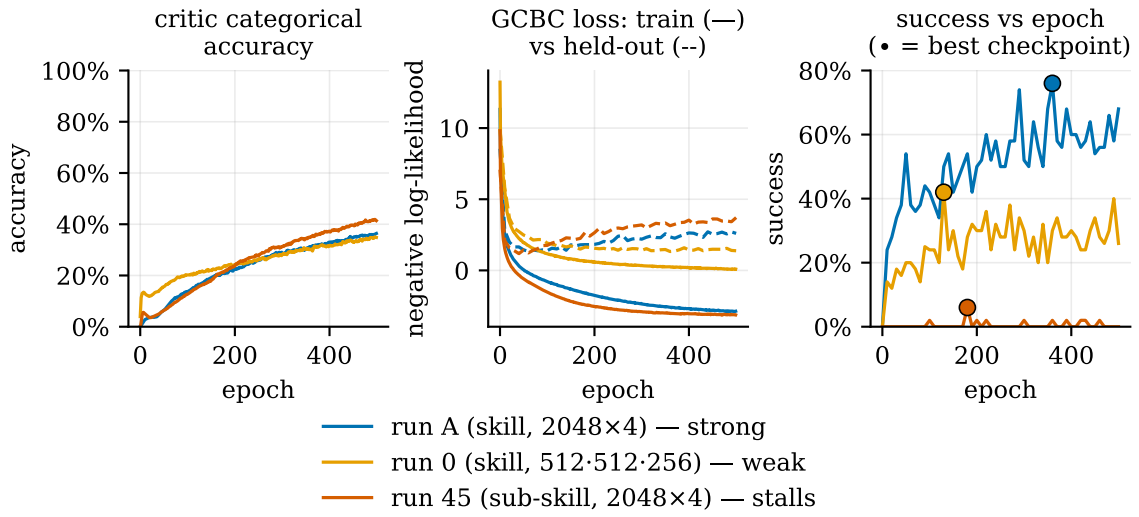

Figure 6.2: Three representative runs over the training axis. *Left:* critic categorical accuracy. *Middle:* training GCBC loss (solid) versus out-of-sample held-out GCBC loss (dashed). *Right:* success versus epoch, with each run's best checkpoint marked.

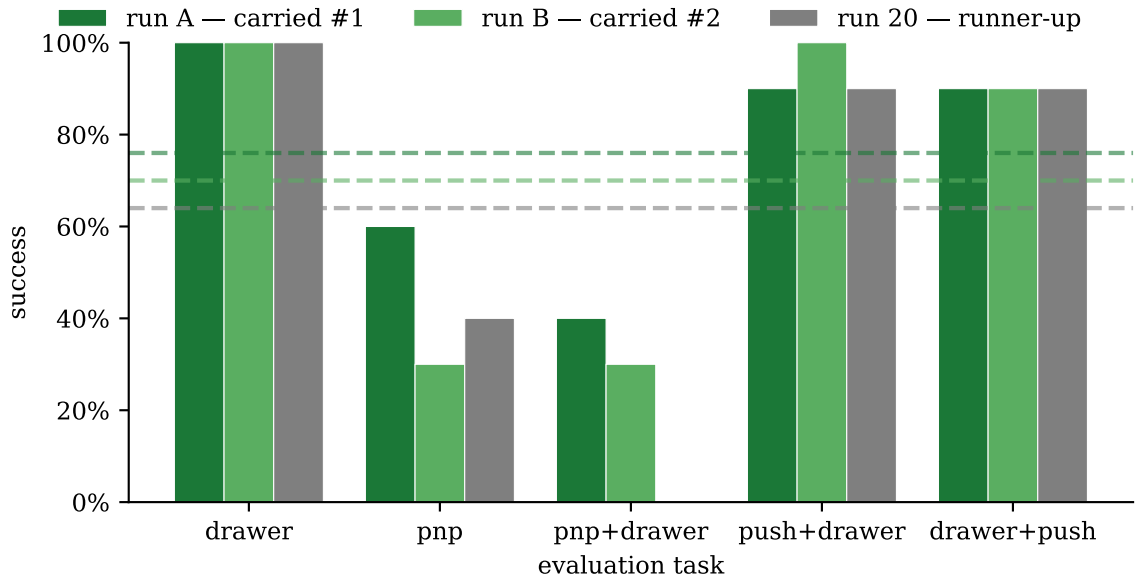

Figure 6.3: CLIP-image goal success per evaluation task for the two carried models (runs A and B) and the runner-up (run 20), each at its best checkpoint — the epoch with the highest *cross-task mean* success, the value reported for that run. CLIP-image goal evaluation at the coarse goal only (no sub-goal decomposition), scored by final-frame success. Dashed lines are per-model means.

have a mean absolute change in per-task success between successive checkpoints of ~ 0.15 (the mean absolute successive difference, MASD), while the raw-image baseline has a MASD of ~ 0.18 and is thus slightly more unstable.

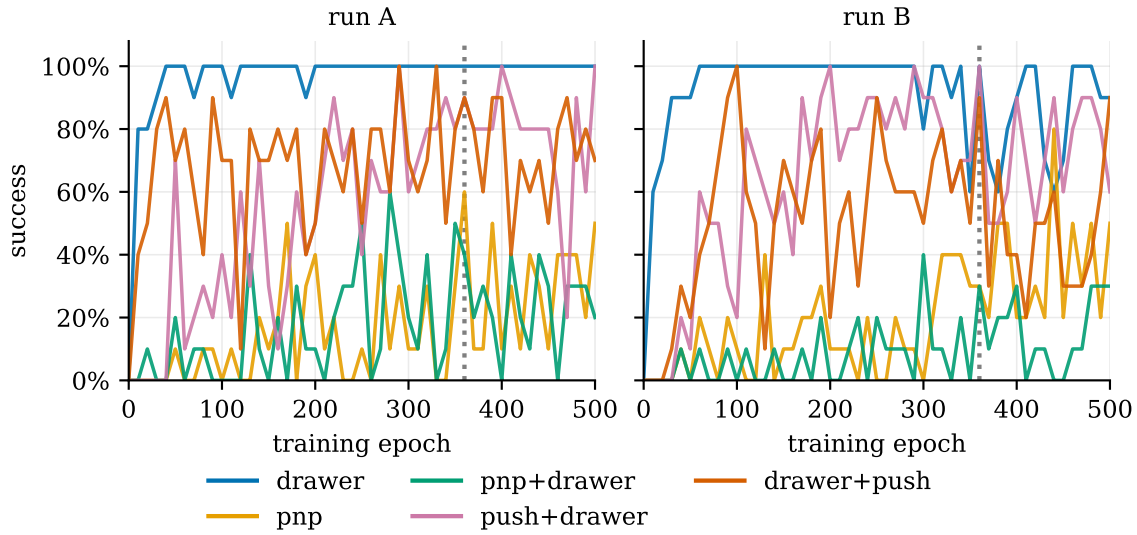

Figure 6.4: CLIP-image goal success over training for the two carried models, one line per evaluation task. Success is non-monotonic and the per-task peaks fall at different epochs — so no single fixed-budget checkpoint is best for every task. The dotted line marks epoch 360, the checkpoint carried into Stage-2.

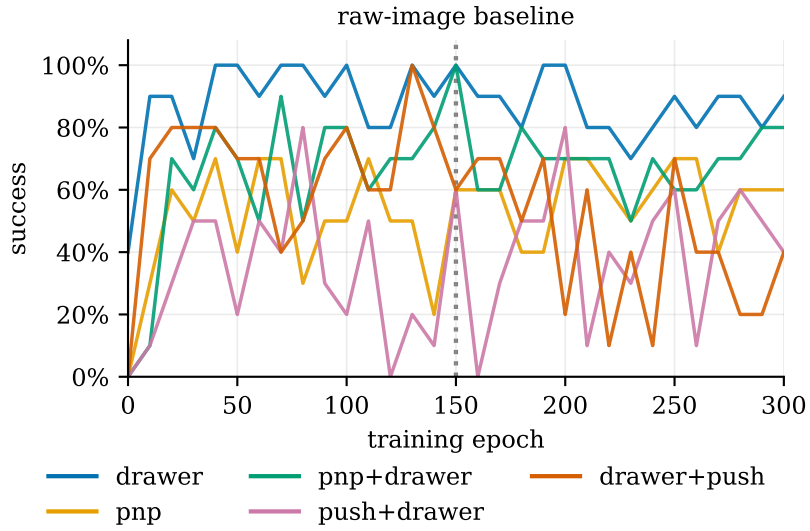

Figure 6.5: Raw-image baseline success over training, one line per evaluation task. As for the carried CLIP-image models (Figure 6.4), success is non-monotonic and the per-task peaks fall at different epochs. The dotted line marks its best checkpoint (epoch 150).

Zero-shot language transfer. A more interesting question is what a finished Stage-1 policy does when its *image* goal is swapped for a *text* goal it was never trained on. Because the carried policies read their goal from CLIP's shared space, a CLIP-text embedding can be dropped in for the CLIP-image goal directly, with no text training and no change to the network. Figure 6.6 compares the same Stage-1 policy under the two goal encodings at the coarse whole-task granularity. With its native CLIP-image goal each carried model solves 76% (run A) and 70%

Table 6.1: Best CLIP-image checkpoint per run (on skill-trajectory dataset only). Superscripts mark the two runs carried into Stage-2.

run	hidden	b	lr	aug	succ.	epoch
09[1]	2048×4	1024	0.0003	obs+goal	0.76	360
23[2]	2048×4	2048	0.0001	obs+goal	0.70	360
21	2048×4	2048	0.0003	obs+goal	0.66	240
20	2048×4	2048	0.0003	obs	0.64	500
05	1024×4	1024	0.0003	obs+goal	0.64	420
17	1024×4	2048	0.0003	obs+goal	0.62	220
04	1024×4	1024	0.0003	obs	0.60	490
22	2048×4	2048	0.0001	obs	0.60	480
19	1024×4	2048	0.0001	obs+goal	0.58	450
07	1024×4	1024	0.0001	obs+goal	0.58	360
06	1024×4	1024	0.0001	obs	0.56	460
08	2048×4	1024	0.0003	obs	0.56	420

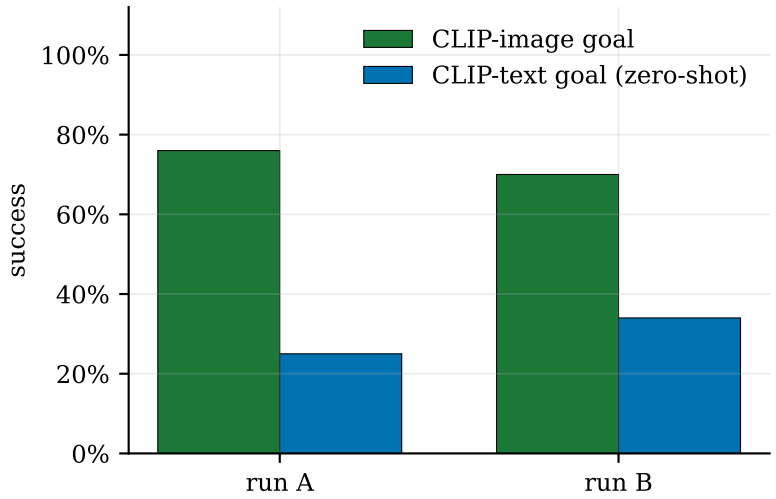

Figure 6.6: The two carried Stage-1 policies evaluated under both goal encodings at the coarse whole-task granularity, on the same carried checkpoint: the native *CLIP-image* goal (green) versus a *CLIP-text* goal supplied zero-shot, with no text training (blue). Swapping the image goal for a sentence drops success from 76/70% to 25/34% — the zero-shot cross-modal gap — but language alone already solves a quarter to a third of episodes. Values use the peak convention (best checkpoint per cell); the CLIP-text bar is read at the epoch-0 warm-start checkpoint.

(run B) of episodes; replacing that goal with a CLIP-text sentence drops success to 25% and 34% — the zero-shot cross-modal gap. Although it has lower success rates, the policy is indeed able to succeed on some tasks, which confirms the use of CLIP as a shared embedding space.

6.2 Feasibility and Baseline Comparison (RQ1, Baseline)

The first experiment asks the most basic question: does the approach work at all? Concretely, can a contrastive policy conditioned on the CLIP embedding of an abstract goal — a sentence

or pddl-like predicates — actually reach the five tasks using sub-goals? Performance is read against the raw-image baseline of Zheng et al. [46] — the external method this thesis aims to match or beat — and against the oracle CLIP-image goal, the hypothesized upper bound.

The hypothesis is that the abstract-goal policy solves the single-skill tasks at rates comparable to the raw-image baseline and trails it on the two composite push+drawer tasks, where the longer horizon and the spatial push are hardest.

Our method versus the baseline. The headline question here is whether the CLIP-text policy with sub-goal decomposition matches the raw-image baseline it is built to replace. Figure 6.7 compares them at each method's *best*: for each of the two models carried from the first stage, the CLIP-text bar is its single best configuration — the checkpoint and granularity (*medium* or *fine* only) are selected by best cross-task performance, which for both models is the *medium* granularity and skill-level text-detail — against the baseline's single best checkpoint. White diamonds mark the oracle CLIP-image goal read at the *same* granularity as each model's configuration (medium), so the diamond isolates the goal *encoding* at a matched decomposition rather than an independently tuned one.

- At its single best configuration CLIP-text trails the baseline overall: ∼58% task success for both carried models against the baseline's ∼76%.
- It *ties* the baseline on opening the drawer (100/100 vs 100) and *overtakes* it on the drawer+push composite (80/80 vs 60).
- It trails on pick-and-place (30/40 vs 60) and pnp+drawer (70/50 vs 100), and most sharply on push+drawer (10/20 vs 60).

Read at the matched medium granularity the oracle CLIP-image goal is no stronger: it too drops to 20/10 on push+drawer and to 30/10 on pnp+drawer, and CLIP-text actually *exceeds* it on pnp+drawer. So the single-configuration shortfall on the push composites is in large part a property of the medium decomposition itself — the per-skill hand-off is hard even for a privileged image goal — rather than of the text encoding.

Fixing the granularity and the checkpoint understates what our approach can achieve on any *one* task. We relax this within the abstract medium/fine goal decomposition: keeping the granularity fixed and holding the run fixed, we select the best checkpoint per task within that single run (Figure 6.8). This corresponds to an optimistic stable policy while staying within our main approach. This lifts push+drawer from the 10/20 of Figure 6.7 to 30/30 and pnp to 50/50, but the fixed medium decomposition still trails the baseline on the push composite.

For the most optimistic read, Figure A.1 in the appendix lifts every restriction at once — any warm-started run, any granularity (the coarse whole-task goal included) and any checkpoint per task. Relative to the fixed-decomposition read of Figure 6.8, which is pinned to a single medium run and still reaches only 30% on push+drawer, this recovers push+drawer to 80/90

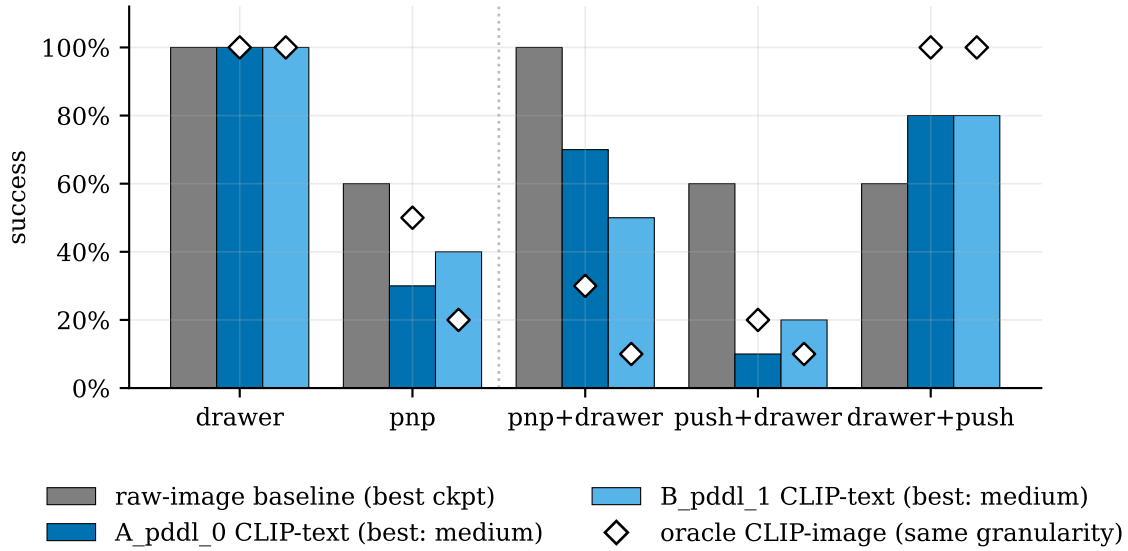

Figure 6.7: Whole-task success per evaluation task for our CLIP-text method against the raw-image baseline it aims to match, each at its single best deployable configuration. Every CLIP-text bar is that model's best *configuration* — the checkpoint and granularity with the highest cross-task mean, which for both carried models is a single `pddl`-worded warm-started run (`A_pddl_0` and `B_pddl_1`) at the medium (per-skill) granularity with skill-level text — and the baseline is its single best checkpoint. White diamonds mark the oracle CLIP-image goal at the *same* granularity as each model's configuration (medium) and at the selected stage-1 checkpoint (`itr_-140`): a matched-granularity encoding reference held at one fixed checkpoint, never cherry-picked per task. Single-skill tasks left of the dotted divider, composite tasks right. CLIP-text ties or beats the baseline on the drawer tasks but trails on the push composites; at the matched medium granularity the oracle image goal collapses on push+drawer too.

— matching or slightly exceeding the per-task baseline there — and lifts pnp+drawer to 80/70; that final lift on the push hand-off is closed only once the coarse whole-task goal is allowed back in. The single-configuration shortfall in Figure 6.7 is therefore in large part a selection effect — which run, checkpoint and granularity are chosen — rather than a ceiling on the goal representation.

Qualitative failure analysis. The success rates say *how often* a task is solved but not *how* it is lost. Figure 6.9 shows representative filmstrips and Table 6.2 groups the failures into six recurring modes by manual evaluation. The drawer opens and closes reliably (10/10), which is why the drawer tasks match or beat the baseline. On pnp the block is usually reached but the grasp fails or the arm pushes the large distractor instead of picking, so seven of ten rollouts miss — this is where CLIP-text trails the baseline most. The composites are more subtle. On push+drawer the push almost always succeeds and the drawer is the sticking point. Only *four* out of nine failures are genuine and can be attributed to the arm pushing correctly but leaving

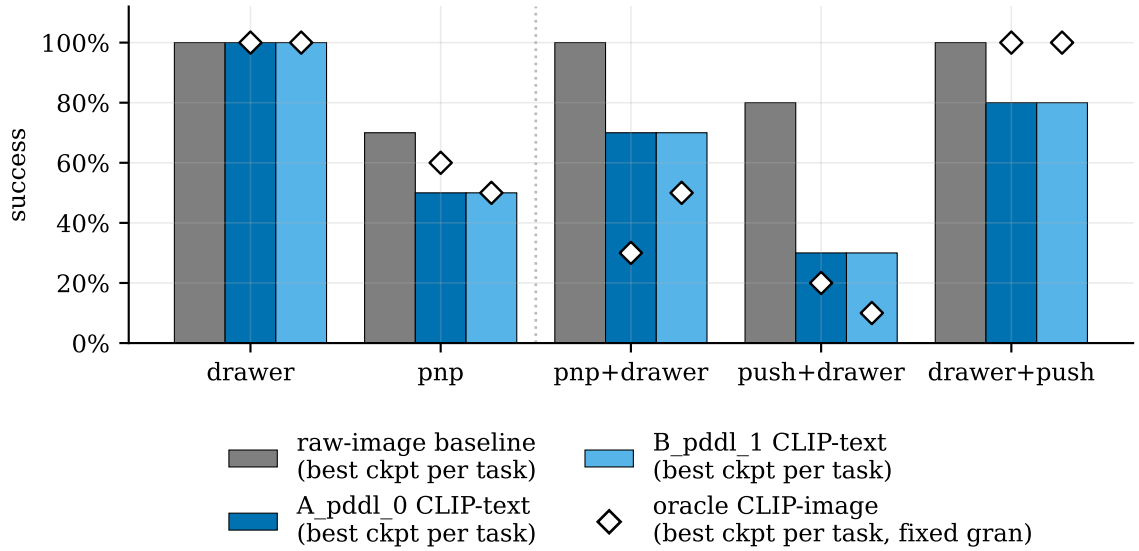

Figure 6.8: Per-task success when each model's goal *decomposition* is held fixed and only the checkpoint is chosen per task. For each carried model we pin the single best no-coarse run and then, for every task, we read the best checkpoint *within that fixed run*. Gray bars are the raw-image baseline (best checkpoint per task); white diamonds are the oracle CLIP-image goal at the same fixed granularity, best checkpoint per task. Relative to the single deployable configuration of Figure 6.7 this recovers pnp and push+drawer somewhat, but the fixed medium decomposition still trails the baseline on the push composite.

without operating the drawer. The remaining *five* are measurement artifacts — the target symbolic goal is reached and the episode terminates, yet the geometric environment success criterion still returns false. Counting those goal-reaching episodes, push+drawer completes roughly *six* of ten rather than the one in ten the headline number reports, in line with the other composites; the apparent push-composite collapse is thus as much a strict scoring threshold as a policy failure. This measurement anomaly appears only at the fine and medium granularities and only for the push+drawer composite task. For drawer+push, two failures are initialization or freeze anomalies, which are not policy errors but rather a mis-specified initial state or a simulator freeze. Thus, most of the genuine failures (9/14) are failures of *goal identity* or *commitment to the goal*. Only five are failures of *control* — the arm starts reaching the goal but cannot complete it.

Summary. This experiment answers the feasibility question in the affirmative: a contrastive policy conditioned on a CLIP-text goal with sub-goal decomposition reaches all five tasks and never collapses to zero on any cell. At a single deployable configuration it ties or beats the raw-image baseline on the drawer tasks but trails it overall (∼58% vs ∼76%), mostly because of the push composites — and roughly half of *that* shortfall is the strict environment success criterion rejecting goal-reaching rollouts rather than a policy failure (Table 6.2). The full per-task ceiling

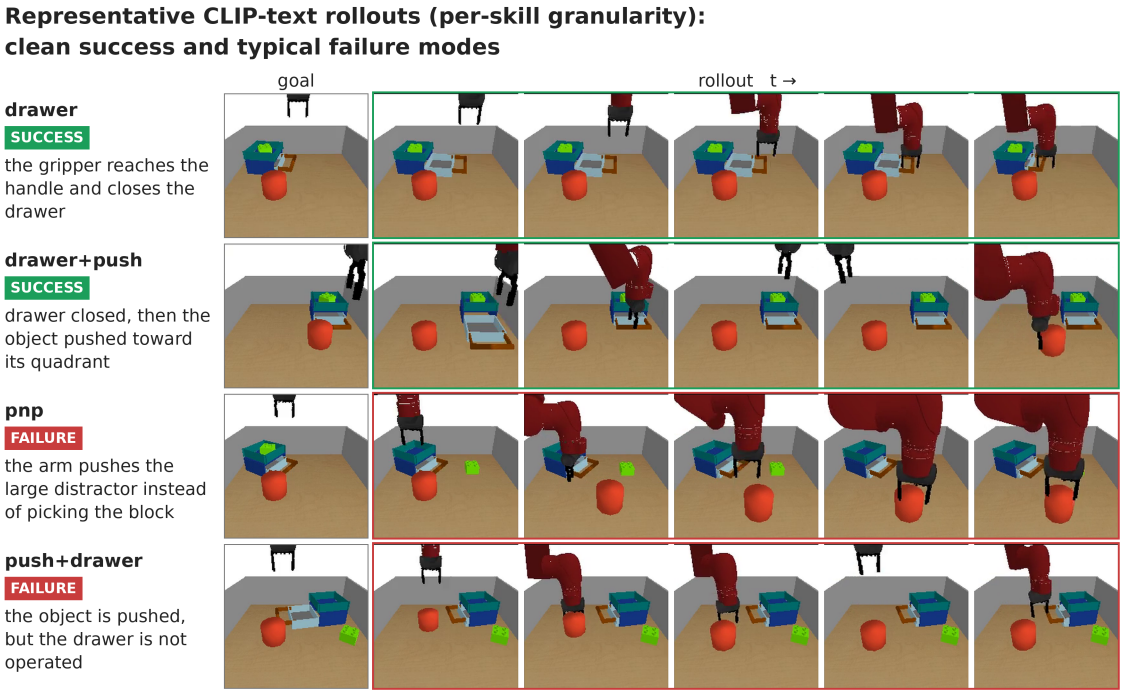

Figure 6.9: Representative rollouts of the carried CLIP-text policy (`A_pddl_0`, per-skill granularity, best CLIP-text checkpoint — the exact rollouts behind Figure 6.7). Each row shows the task's final goal frame (left) and the observation at evenly spaced timesteps; a green border marks a success, red a failure. Drawer and drawer+push complete cleanly, while pick-and-place pushes the large distractor instead of grasping the block and push+drawer pushes correctly but does not operate the drawer — the failure modes of Table 6.2.

that also admits the coarse whole-task goal matches or beats the baseline on all tasks except pnp and pnp+drawer (Figure A.1, appendix). That shows that one can achieve high accuracy with symbolic goals. At the matched medium granularity the oracle is no stronger than CLIP-text on the push composites; whether the remaining coarse-goal gap is a property of the goal *encoding*, and how it narrows with sub-goal granularity, are the subjects of the next two sections.

Table 6.2: The recurring failure modes of the carried CLIP-text policy at the per-skill granularity, read from all fifty evaluation rollouts (Figure 6.9); # counts rollouts, summing to the 21 failures across the five tasks. The fifth mode is a scoring artifact — the target goal is reached but the geometric success criterion rejects it — so the push+drawer success rate understates true completion.

Failure mode	Affected task(s)	#	What happens
Failed grasp	pnp	4	the block is reached but the grasp fails, so it is never picked up
No commitment to grasp	pnp	1	the arm moves above the block and stays hovering there
Spurious push of the distractor	pnp, pnp+drawer	4	the arm pushes the large distractor object instead of, or after, handling the target block
No release after grasp	pnp+drawer	1	the block is grasped but never placed
Drawer not operated after push	push+drawer	4	the push succeeds but the gripper leaves without operating the drawer
Symbolic goal reached, detector rejects	push+drawer	5	the target goal is achieved and the episode ends, yet the geometric success criterion still scores a failure (a measurement artifact, not a policy error)
Initialization / freeze anomaly	drawer+push	2	a wrong initial state, or the arm freezes after closing the drawer

6.3 Encoding versus Information (RQ2)

Having established feasibility, the second experiment isolates how much the goal *representation* alone matters. The same trained policy is evaluated with the four interchangeable goal representations of Section 5.3.4, with the symbolic and language goals level-matched to carry the same predicates. Because the information is held equal, any gap between the predicate vector and the rendered sentence is a property of the *encoding* rather than of missing *information*.

Figure 6.10 reports the peak cross-task success of the three abstract goal encodings (best checkpoint per cell) for both carried models. Three patterns hold across both models.

- **CLIP-text is the strongest abstract encoding** at every sequential granularity: coarse (45/52 vs `pddl` 41/29, voxel 38/36), medium (58/57 vs `pddl` 51/53, voxel 44/56), and fine (54/50 vs `pddl` 50/50, voxel 40/28).
- **Matching the information closes the gap only for isolated skills.** On the single-skill isolation cells the predicate vector reaches CLIP-text's performance — iso-medium 76/72 vs 81/82, iso-fine 83/81 vs 86/86 — so once the goal is a single reachable predicate, the raw predicate vector is as good as the rendered sentence. The gap re-opens on the coarse and sequential goals, where composing several predicates is what the CLIP encoding does better.

- **The geometric voxel encoding is weakest at nearly every granularity**, below both the symbolic and the language goal (e.g. fine 40/28 vs `pddl` 50/50). A purely metric "where the objects go" goal, with no relational or semantic structure, carries the least usable signal. This is partly by construction: the voxel cells are coarse relative to the environment's own geometric success tolerance — in the horizontal plane a cell (3–7 cm) is comparable to the success radius (6.5–8 cm), but the graspable object's height axis has only five ≈ 7 cm bins against a 1 cm vertical placement tolerance (Figure 5.10) — so the goal fixes a coarse target region rather than a precise position.

Table 6.3 resolves the coarse result per evaluation task, all five arms side by side — the raw-image baseline, the oracle CLIP-image ceiling, and the three abstract encodings.

- **On the drawer-opening tasks the symbolic arms are competitive.** Opening the drawer is a single discrete predicate: there `pddl-direct` and voxel match or beat CLIP-text (open drawer 80/90 vs 60) and stay close on the drawer+push composite (72/50 vs 68) — when the goal is one binary fact, a raw predicate vector encodes it about as well as a sentence.
- **The encoding gap opens on precise and composite tasks.** On pick-and-place and the two hand-off composites the symbolic arms fall well behind CLIP-text: pick-and-place 38 vs `pddl` 8/voxel 20, push+drawer 43 vs 15/10, and pnp+drawer 35 vs 0/15. Precise placement and composing a skill hand-off are exactly where the CLIP encoding helps.
- **Voxel is weakest wherever composition or precision is required**, at or near the floor on pick-and-place (20), push+drawer (10), and pnp+drawer (15): a purely metric goal carries the least usable structure for the hardest tasks.
- **Neither image reference is a strict per-task ceiling.** The oracle CLIP-image goal dominates the drawer composites (drawer+push 90 vs 60, push+drawer 95 vs 60) yet the raw-image baseline wins pnp+drawer (100 vs 35) and pick-and-place (60 vs 45) — the two image references trade off across tasks rather than one bounding the other.

Summary. With the information held equal, the *encoding* still matters — CLIP-text > `pddl-direct` > voxel on the composite goals — but the symbolic and language encodings almost converge on isolated fine skills, so the RQ2 gap is specifically one of *composing* goals rather than reaching them. For example, on the discrete drawer-opening goals a raw predicate vector or voxel is as good as a sentence, while on precise placement and the two skill hand-offs only the CLIP encoding keeps pace. All three abstract encodings nonetheless stay below the oracle CLIP-image goal and the raw-image baseline, leaving headroom to attribute the remaining differences to sub-goal granularity rather than encoding. However, these abstract encodings are measured on the deliberately conservative frozen-encoder substrate. Section 5.3.6 shows that training the policy end-to-end from scratch on the goal representation does better — so whether

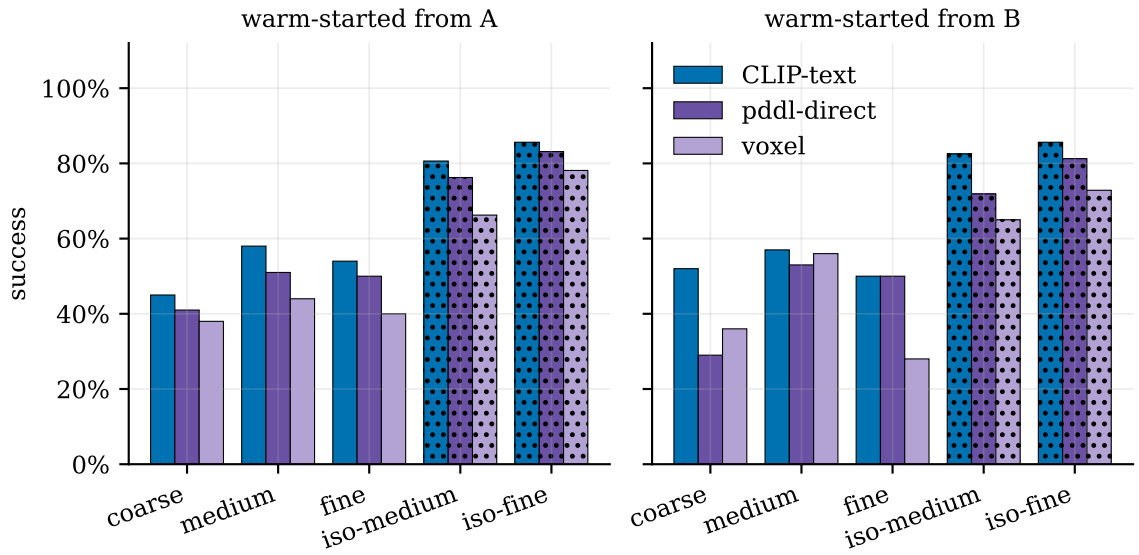

Figure 6.10: Goal-encoding comparison: peak cross-task success (best checkpoint per cell) of the three abstract goal encodings that share a frozen state-action encoder and training data and differ only in the goal representation — the CLIP text embedding (*CLIP-text*, ours), the raw symbolic predicate vector (`pddl-direct`), and the geometric object-position multi-hot (*voxel*, Figure 5.10). Shown for both carried models. CLIP-text leads on the coarse and sequential goals; `pddl-direct` catches up on the isolated single-skill cells; the voxel encoding trails throughout.

Table 6.3: Success (%, at each model's single best cross-task-mean checkpoint) per evaluation task at the coarse granularity: the raw-image baseline, the oracle CLIP-image ceiling, and the three abstract goal encodings (mean over the two carried models, and over the skill/sub-skill text tiers for the text and symbolic arms). Top: single-skill tasks; bottom: composite tasks. The `pddl-direct` and voxel columns are the symbolic ablations; they collapse on the two hand-off-sensitive composites.

task	Raw-image	Oracle CLIP-image	CLIP-text	pddl-direct	voxel
drawer	100	100	60	80	90
pnp	60	45	38	8	20
pnp+drawer	100	35	35	0	15
push+drawer	60	95	43	15	10
drawer+push	60	90	68	72	50

an end-to-end-trained goal representation would close the gap to the two image references is untested here and left open.

6.4 Granularity (RQ3)

This section compares the coarse, medium, and fine granularities, scoring each both per subgoal and as a full sequence rollout (Section 5.3.1). The hypothesis is that the medium and fine decompositions outperform coarse granularity on the composite tasks: a coarse, whole-task

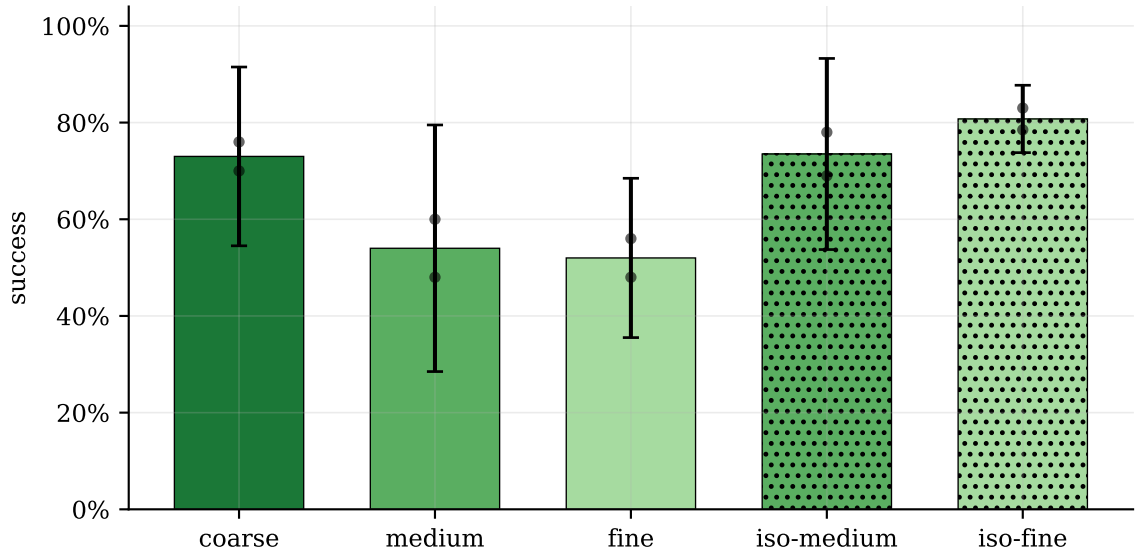

Figure 6.11: CLIP-image (oracle) success by goal granularity for the two carried Stage-1 policies (mean over 2 models × 5 tasks; error bars are 95% confidence intervals over the per-(model, task) rates; dots are the two per-model means). The three solid bars are whole-sequence success at coarse/medium/fine granularity — finer decomposition does *not* help here; the two dotted bars (iso-medium, iso-fine) sit well above the sequential bars — the ≈20–30-point cost of chaining sub-goals (error compounding).

goal fixes the final object positions but not the order in which they are reached, whereas finer sub-goals can elicit that order. We read the granularity effect in three steps: first the CLIP-image oracle upper bound across granularities, then whether the same holds for the main CLIP-text method, and finally whether the goal-text detail changes the picture.

Oracle CLIP-image upper bound across granularities

Before the text and symbolic arms, we establish the granularity behavior of the *CLIP-image* policies, which are the upper bound those arms are compared against. The two selected CLIP-image models are evaluated with CLIP-image goals at every granularity, as a full-sequence rollout and — for the medium and fine decompositions — in the iso-medium and iso-fine cells (Figure 6.11, Table 6.4).

Finer is not better. Finer goal decomposition does *not* improve whole-task completion for the image oracle: coarse whole-task goals succeed 73% of the time, well ahead of the medium (54%) and fine (52%) sequences. Coarse is the best granularity for *both* selected models (76 and 70), and finer decompositions are uniformly worse — so the hypothesized finer-is-better trend is not merely absent but reversed for this upper-bound arm.

Table 6.4: CLIP-image (oracle) success by granularity: whole-sequence completion (coarse / medium / fine) and per-sub-goal success in isolation. Mean over the two carried models and five tasks; 95% CI over the per-(model, task) rates.

Granularity / cell	Success (%)	95% CI	n
Whole-sequence success (goal sequence completed)			
coarse	73.0	18.5	10
medium	54.0	25.5	10
fine	52.0	16.5	10
Per-sub-goal success in isolation (predecessor restored)			
skill sub-goals	73.5	19.8	10
sub-skill sub-goals	80.7	7.0	10

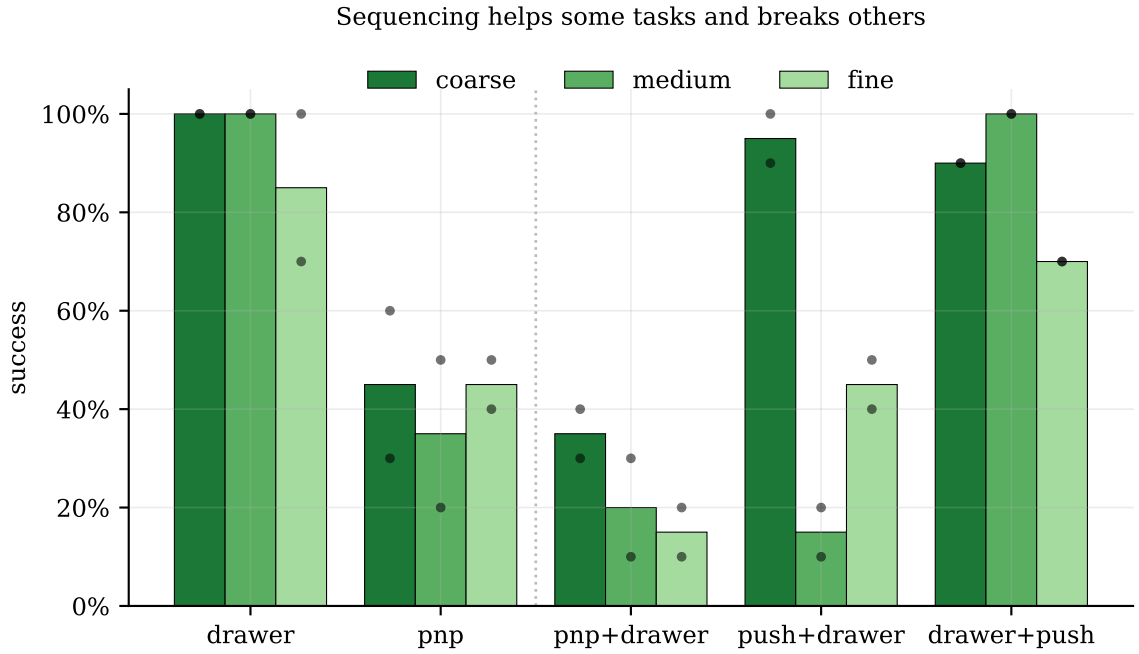

Figure 6.12: Success per evaluation task across goal granularities for the CLIP-image oracle (bars are the mean of the two carried models; dots the individual models). The near-flat average granularity effect hides opposing per-task behavior: coarse goals are best or tied on all tasks but drawer+push, and medium decomposition collapses the push+drawer task (95 → 15%).

Decomposition costs unevenly. The average, however, hides strongly opposing per-task behavior (Figure 6.12). For the CLIP-image oracle, coarse goals are best or tied on every task except drawer+push (where medium edges coarse, 90 → 100%). Its sharpest failure is the push+drawer task, which falls from 95% under a single coarse goal to 15% once it is split into medium sub-goals, recovering only partially (45%) at the fine granularity. The pnp+drawer task also declines with decomposition (35 → 15%), while easy single-skill tasks (drawer) stay near the ceiling throughout. Goal granularity is therefore not uniformly bad but task-dependent.

Table 6.5: Success (%, best checkpoint per cell, mean over the two carried models) by goal granularity for the CLIP-image oracle and the three abstract goals. Bold marks the best granularity for each goal (highest per column). The abstract goals peak at the medium (per-skill) granularity — the opposite of the oracle's coarse preference — and CLIP-text at medium overtakes the oracle CLIP-image goal.

granularity	Oracle CLIP-image	CLIP-text	pddl-direct	voxel
coarse	**73**	48	35	37
medium	54	**58**	**52**	**50**
fine	52	52	50	34

The penalty is compounding, not broken chaining. Why does decomposition cost so much? The per-sub-goal skill is in fact high — 74% (skill) and 81% (phase) in isolation, with the finer, shorter sub-goals the easier ones — yet chaining those same sub-goals into a sequence costs twenty to thirty points (74 → 54 and 81 → 52). That drop is close to what *independent* composition alone predicts. If the sub-goals were independent hurdles, whole-sequence success would equal the *product* of the isolated per-sub-goal rates, and it largely does (Figure 6.13): across the ten task × granularity cells the product tracks the actual success with a mean absolute error of 0.12 and a correlation of r=0.87 once a single outlier is removed (r=0.62 with it), and $0.74^2 \approx 0.55$, $0.81^3 \approx 0.53$ recover the sequential rates (54 and 52) to within a point. So the sequence penalty is mostly the arithmetic of needing *every* sub-goal to succeed rather than a policy that breaks when it hands one sub-goal to the next — per-sub-goal skill is intact and, with a single exception, the transitions are clean. That exception is push+drawer at medium granularity: its two sub-goals are individually easy (product 0.86) yet the sequence collapses to 15%, far *below* the independence prediction, because reaching the first skill's goal leaves the arm in a state from which the second is unreachable — a genuine broken *hand-off*. It recovers at the fine granularity, so the fault is one specific skill boundary, not chaining in general. The isolated sub-goal metric is thus both a predictor of sequential success *and* a diagnostic that pinpoints which boundary breaks.

CLIP-text policy across granularities

Abstract goals invert the oracle's preference. The headline result flips for the abstract arms. Table 6.5 runs the same coarse/medium/fine comparison for the abstract goals against the CLIP-image oracle upper bound: where the image goal is best at coarse (73, falling to 54/52), all three abstract goals peak at the *medium* (per-skill) granularity — CLIP-text 57 (vs 48 coarse, 52 fine), `pddl-direct` 52, and voxel 50. Decomposing the goal into medium sentences therefore helps where a single coarse sentence is hard to ground, though the fine, per-phase split gives that gain back, so finer is not uniformly better. At the medium granularity CLIP-text (57) even *exceeds* the oracle CLIP-image goal (54): a medium *sentence* is easier to follow than a medium *image*, the one regime where the language goal overtakes its own visual upper bound.

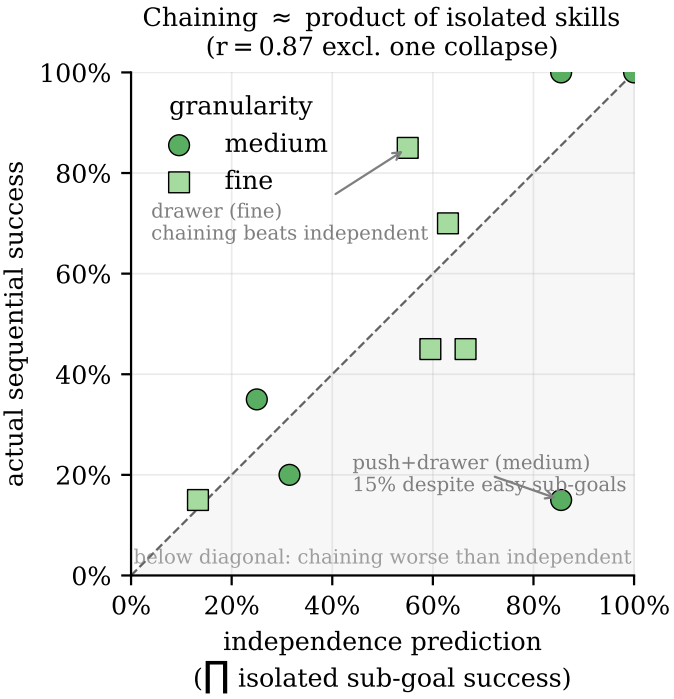

Figure 6.13: Actual sequential success versus the independence prediction — the product of the isolated per-sub-goal success rates — for the CLIP-image oracle, one point per task and granularity. Points hug the diagonal (r=0.87, mean absolute error 0.12, excluding the outlier), so chaining mostly behaves like independent composition. The two labeled points are the informative deviations: push+drawer at medium granularity collapses to 15% despite easy isolated sub-goals (a broken goal hand-off, not weak skill), while drawer at fine granularity clears the diagonal — there chaining *beats* independent composition.

The gap to the ceiling is a sequencing gap. Aggregated across tasks, Figure 6.14 places CLIP-text against each model's CLIP-image ceiling at every granularity. CLIP-text trails the ceiling on the coarse goal (45/52 vs 76/70) but *closes* the gap for isolated sub-goals — matching it at phase (86/86) and reaching it at skill (81/82 vs 70/84). So, exactly as for the oracle, what CLIP-text loses is in *sequencing* several goals, not in reaching a single one.

The same skill boundary breaks. Per task (Figure 6.15, warm-started from A), behavior is heterogeneous — the drawer task is easiest and pick-and-place hardest — and no single granularity wins everywhere. Per-skill is strongest on the drawer tasks yet collapses on push+drawer (5%), the very skill-boundary hand-off the oracle already flagged as the one place chaining breaks (Figure 6.13). The image ceiling makes the cause explicit: run A solves push+drawer perfectly with a CLIP-image goal (100%) yet it is among the hardest for text, so it is the text goal at that boundary, not the policy, that fails. The second carried model is analogous, with a milder push+drawer dip (25% versus the 5% warm-started from A); its per-task breakdown, and

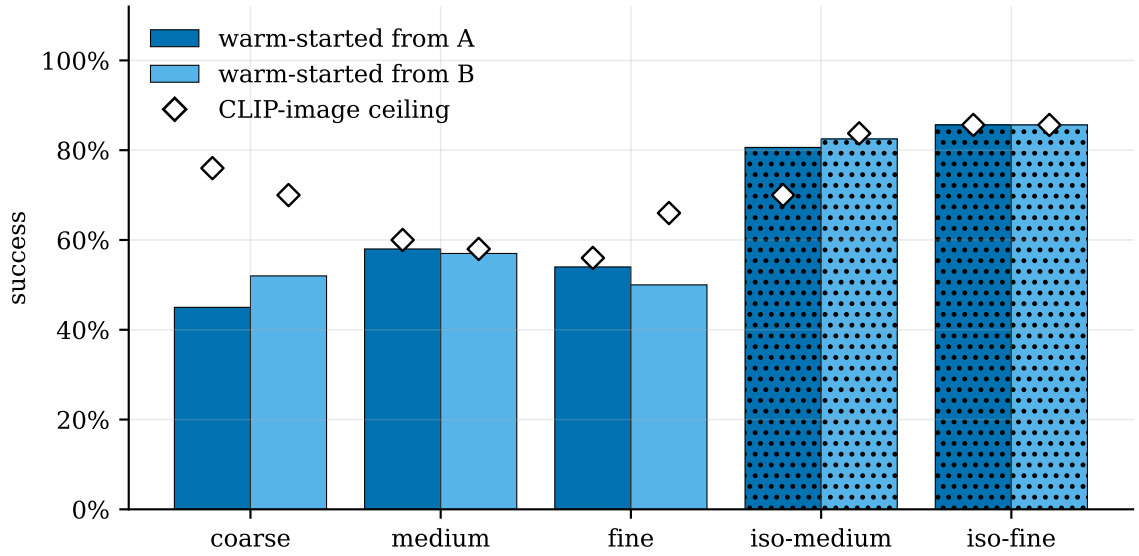

Figure 6.14: Stage-2 CLIP-text (our method, warm-start with frozen state-action encoder) success per goal granularity, warm-started from A and from B, at each granularity's best checkpoint over the template sweep (mean of the two text-detail tiers), against each model's CLIP-image ceiling (diamonds). CLIP-text trails the ceiling on the coarse goal but closes the gap for isolated sub-goals, where it matches or exceeds the CLIP-image goal.

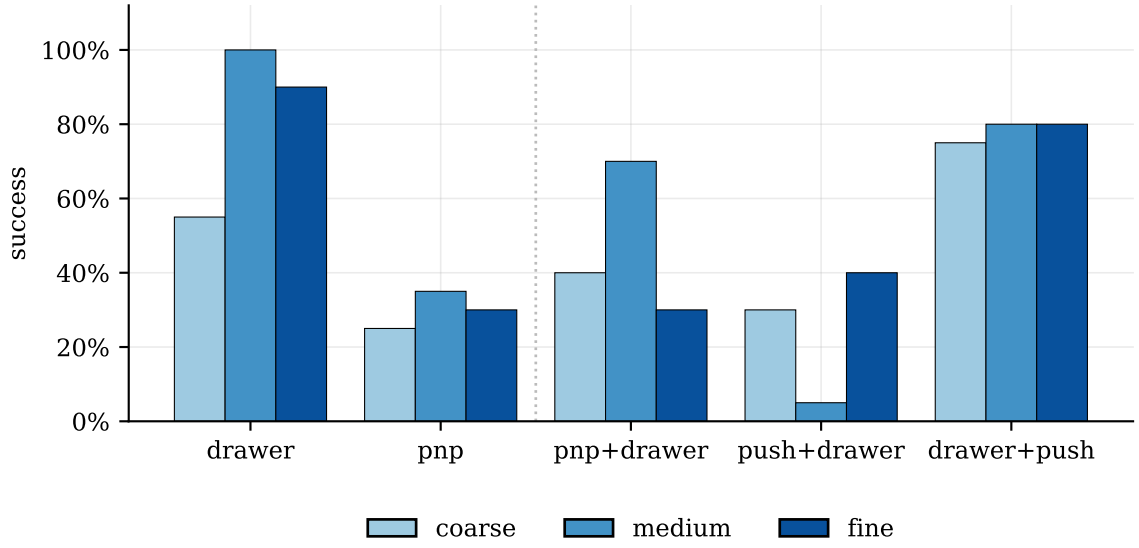

Figure 6.15: Per-task stage-2 CLIP-text success for the two-stage warm-start curriculum (our main method, warm-started from A), for the three sequential granularities (coarse, medium, fine), at each cell's best checkpoint over the template sweep (mean of the two text-detail tiers). No single granularity dominates across tasks; medium collapses on the push+drawer skill hand-off.

a per-template-style view showing no style dominates across tasks, are given in the appendix (Figures A.2 and A.3).

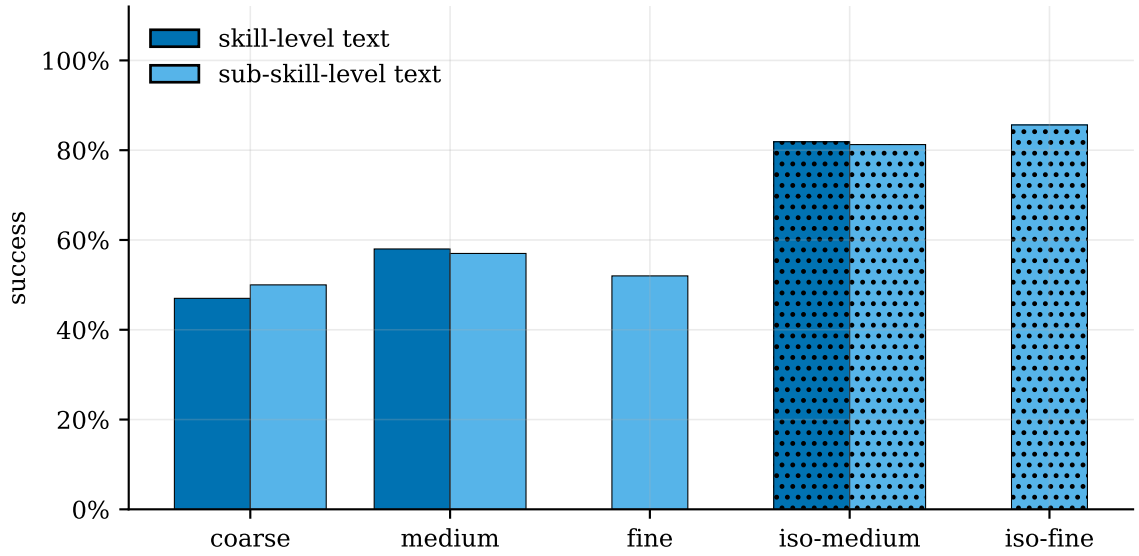

Figure 6.16: Whole-task CLIP-text success by goal granularity (coarse / medium / fine, and the two isolation cells) and, within each granularity, by goal-text *detail* tier — skill-level (dark) vs sub-skill-level (light) sentences. Both tiers are CLIP-text; fine and iso-fine carry only the sub-skill-level rendering. Mean over the two carried models at each cell's best checkpoint. The bars move a lot *across* granularities but barely *within* one when only the text detail changes: granularity matters, text detail does not.

Granularity matters, text detail does not. A text goal has two knobs — *which* decomposition it uses and *how* detailed the sentence is — and they carry very different weight. Figure 6.16 plots both at once: success by granularity, and within each granularity by goal-text detail tier (skill- vs sub-skill-level). Granularity moves success substantially — from ~48% at the coarse goal to ~57% at medium, and up to ~82–86% once each sub-goal is scored in isolation — whereas the detail tier barely registers: skill- and sub-skill-level sentences land within a few points at every granularity (coarse 47/50, medium 58/57, iso-medium 82/81). Describing goals in more detail than necessary does not help; what matters is the granularity at which the goal is *issued*, not the verbosity of its description.

Summary. Granularity acts oppositely on the image and the abstract goals. For the CLIP-image oracle, finer is *worse* — coarse whole-task goals are best (73 vs 54/52) — and that cost is mostly the compounding of independent sub-goal successes rather than a chaining defect, bar the single push+drawer skill hand-off that genuinely breaks. For the abstract goals the effect *inverts*: all three encodings — CLIP-text, `pddl-direct`, and voxel — peak at the *medium* granularity, so decomposing an abstract goal into per-skill descriptions helps where a single whole-task description is hard to ground. CLIP-text benefits most, overtaking the oracle image at medium (57 vs 54). The gain is not monotone — the finer per-phase split hands it back — and it comes from the granularity at which the goal is *issued*, not the detail of the sentence:

skill- and sub-skill-level sentences are within a few points everywhere. In short, decomposition helps every abstract goal, CLIP-text included, up to the one skill boundary that breaks for image and text alike.

6.5 Pretraining (RQ4)

This is a secondary arm rather than a central claim. It covers the two forms of pretraining that feed the text-goal policy, both set up in the evaluation chapter: the *encoder* — off-the-shelf CLIP versus a LoRA-finetuned CLIP for text goals (Section 5.3.5) — and the *policy* — the two-stage image-then-text curriculum, warm-starting stage 2 from the best-checkpoint stage-1 model with the state-action encoder frozen, against a from-scratch ablation (Section 5.3.6).

Encoder adaptation

Adapting the CLIP image encoder with LoRA sharpens held-out image-to-text retrieval by roughly an order of magnitude: the selected adapter (rank 4, learning rate 10^{-4}; Table A.1) raises top-1 in-batch image-to-text retrieval on the held-out split from 0.01–0.02 for the unadapted, zero-shot encoder — near the in-batch chance rate of 1/128 at our batch size — to 0.14, with the strongest configurations across the rank/learning-rate grid clustering around an eight- to tenfold lift. What RQ4 turns on, though, is whether that sharper representation helps *control*. Figure 6.17 runs the comparison end-to-end: the entire two-stage pipeline — stage-1 CLIP-image pretraining and the stage-2 text-goal warm-start — built on the LoRA-adapted CLIP against the off-the-shelf CLIP, with each pipeline carried through its own stage-1 sweep and model selection (Section 5.3.5). Despite the order-of-magnitude retrieval gain, adapting the encoder does *not* improve downstream success. On the sequential goals the LoRA pipeline is *below* off-the-shelf CLIP — its best single configuration reaches 42% against 58%, and per granularity (mean over the two models) it trails at coarse (38 vs 49), medium (40 vs 58), and fine (40 vs 52). It matches or exceeds off-the-shelf CLIP only on the isolated single-skill cells (iso-medium 87 vs 82, iso-fine 100 vs 86). Sharper image-to-text alignment thus helps *reach* a single named sub-goal but not *compose* a sequence of them, and end-to-end it slightly hurts — so encoder adaptation is not worthwhile for the text-goal policy in our experiment.

Policy: two-stage curriculum vs from scratch

The second pretraining axis is the policy. Our two-stage curriculum warm-starts stage 2 from the best-checkpoint CLIP-image model and freezes the state-action encoder and continues training with CLIP-text goals. Against this, a single-stage policy of the same architecture is trained from scratch. Figure 6.18 compares them at each cell's best checkpoint (peak cross-task success). The curriculum does *not* improve downstream success: the from-scratch policy is level with or above the warm-start+frozen-SA policy at every granularity — coarse 56 vs 45,

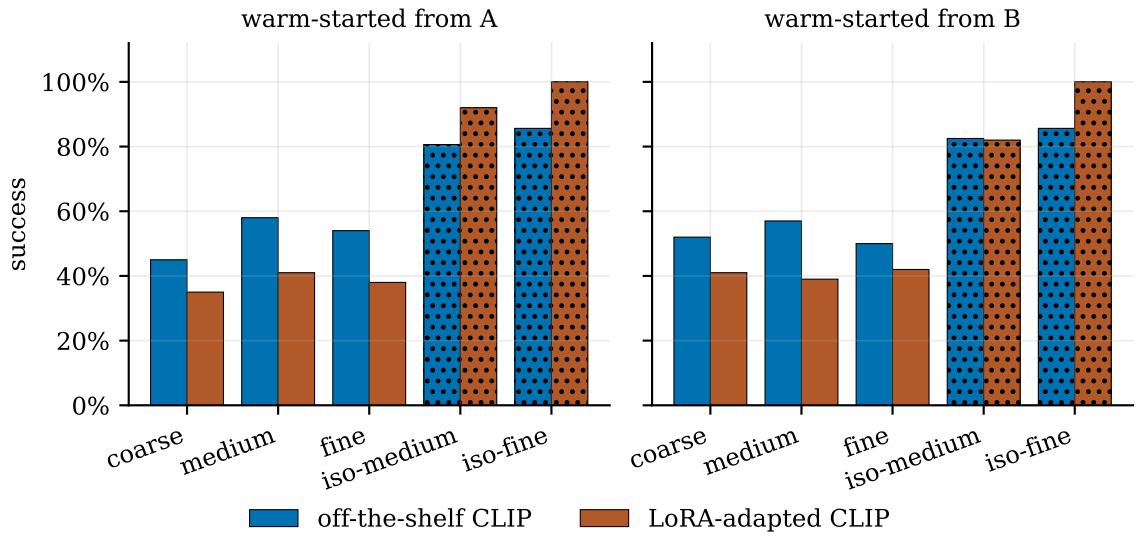

Figure 6.17: Downstream text-goal success per granularity for the full two-stage pipeline built on the LoRA-adapted CLIP versus the off-the-shelf CLIP, at each carried architecture (warm-started from A and B; best checkpoint per cell). The LoRA pipeline trails on the sequential goals (coarse/medium/fine) and matches or exceeds only on the isolated single-skill cells — the order-of-magnitude retrieval gain does not transfer to control.

medium 58 vs 58, and fine 56 vs 54% on the sequential goals, and clearly ahead on the isolated single skills (iso-medium 100 vs 81, iso-fine 100 vs 86). Freezing the CLIP-image state-action encoder thus seems a mild handicap rather than a help for text goals — letting the policy adapt end-to-end matches or beats it. The two-stage curriculum is therefore justified by convenience, not a task gain, and that convenience comes from *warm-starting* — a single stage-1 backbone seeds all twelve text runs — rather than from freezing. Freezing the state-action encoder buys no downstream gain; if anything it slightly lowers success.

Figure 6.19 recasts the comparison as a per-task best-config view, the analogue of Figure 6.7: each arm at its single best (checkpoint, granularity, text-detail) configuration. The two policies pick *different* best configs — the curriculum a medium (per-skill) goal, from-scratch a coarse whole-task goal — and their per-task profiles trade off rather than dominate: the curriculum leads on drawer (100 vs 70) and pnp+drawer (70 vs 20), while from-scratch wins the push+drawer hand-off outright (90 vs 10). Neither is uniformly better, consistent with the near-tied aggregate, and both stay below the oracle CLIP-image ceiling.

Summary. Neither form of pretraining moves the downstream task metric, so RQ4 is largely a negative result. Adapting the CLIP image encoder with LoRA sharpens image-to-text retrieval by an order of magnitude yet does not carry over to control — it helps *reach* a single named sub-goal but not *compose* a sequence, and end-to-end it slightly hurts. The two-stage image-then-text curriculum is likewise no better than training the text-goal policy from scratch:

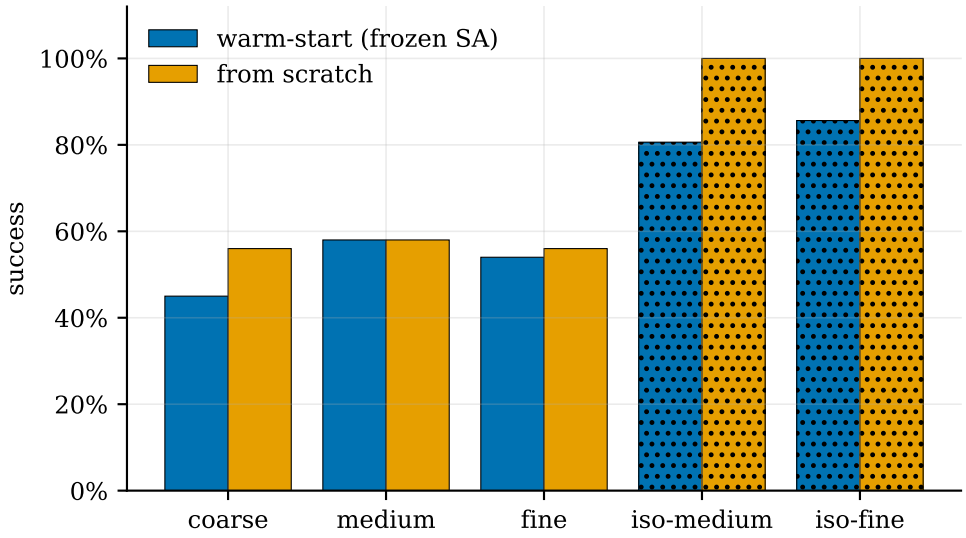

Figure 6.18: Stage-2 CLIP-text success per granularity for the two-stage warm-start (frozen state-action encoder) curriculum (warm-started from A) against a from-scratch policy of the same architecture, each at its best checkpoint per cell. From-scratch matches or exceeds the curriculum at every granularity — and clearly so on the isolated single-skill cells — so the two-stage curriculum does not improve downstream success.

from-scratch matches or beats the warm-started, frozen-encoder policy at every granularity — clearly so on the isolated single skills — and the two trade off per task rather than one dominating. Both design choices are therefore made for convenience rather than performance — an off-the-shelf CLIP and a shared, reusable stage-1 backbone — and the headroom to the oracle CLIP-image ceiling is not closed by pretraining at this scale.

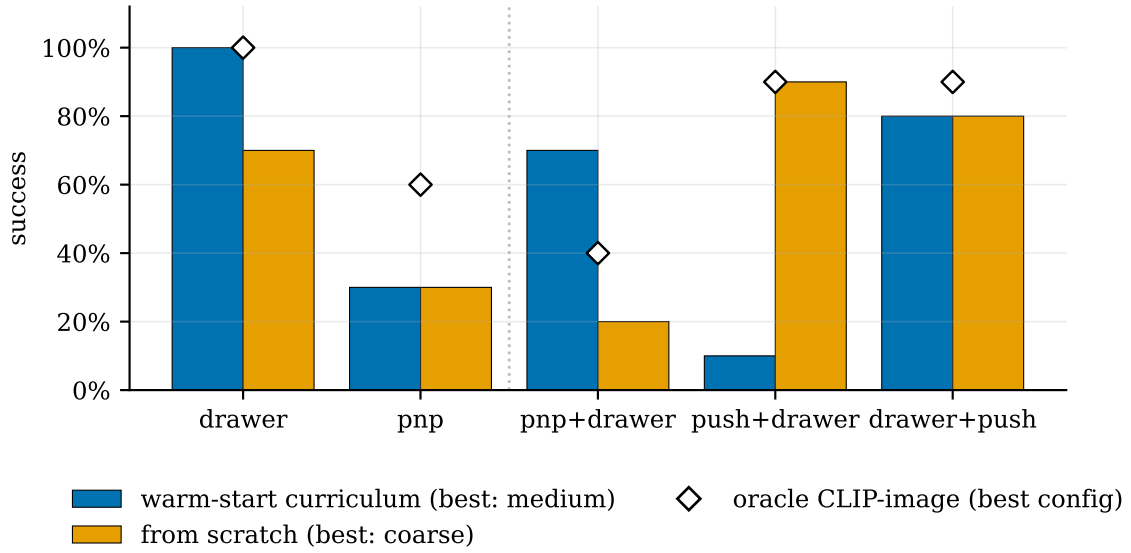

Figure 6.19: Per-task success for the two-stage warm-start curriculum and the from-scratch policy, each at its own single best (checkpoint, granularity, text-detail) configuration (warm-started from A), with the oracle CLIP-image ceiling as diamonds. Single-skill tasks left of the dotted divider, composite tasks right. The two arms choose different best configs and trade off across tasks — the curriculum leads on the drawer and pnp+drawer tasks, from-scratch on the push+drawer hand-off — consistent with their near-identical aggregate (Figure 6.18).

6.6 Template Study

The template study is the secondary, policy-free diagnostic set out in Section 5.3.7: it scores how each template's wording shapes the goal embedding *before any policy is trained*, on the three intrinsic properties defined there — discriminability, image–text alignment, and text-space distinctness. Its metrics are only a weak proxy for task success — the decisive evidence is the downstream evaluation — so it is reported here as supporting context, not a headline result. It ran before the main experiments, and its findings fixed the template wording that the text stage was trained on: two templates per style were carried forward.

6.6.1 Searching for the Best Templates per Template Style

To choose the wording for each template style rather than commit to a single phrasing, we enumerated the full grid of 324 phrasing variants spanning the three styles — PDDL-like tuples, short phrases, and long narratives — by varying how objects are named, how the table quadrant is worded, how the predicate fragments are joined, whether a scene-framing prefix is prepended, and whether inactive predicates are spelled out. Each variant was scored by text-distinctness and image–text alignment across the goal states, and the two best templates of each template style were kept for the main experiments, with the single strongest of each ringed in Figure 6.20. The clearest negative effect was spelling out the inactive predicates: phrases such as "the block

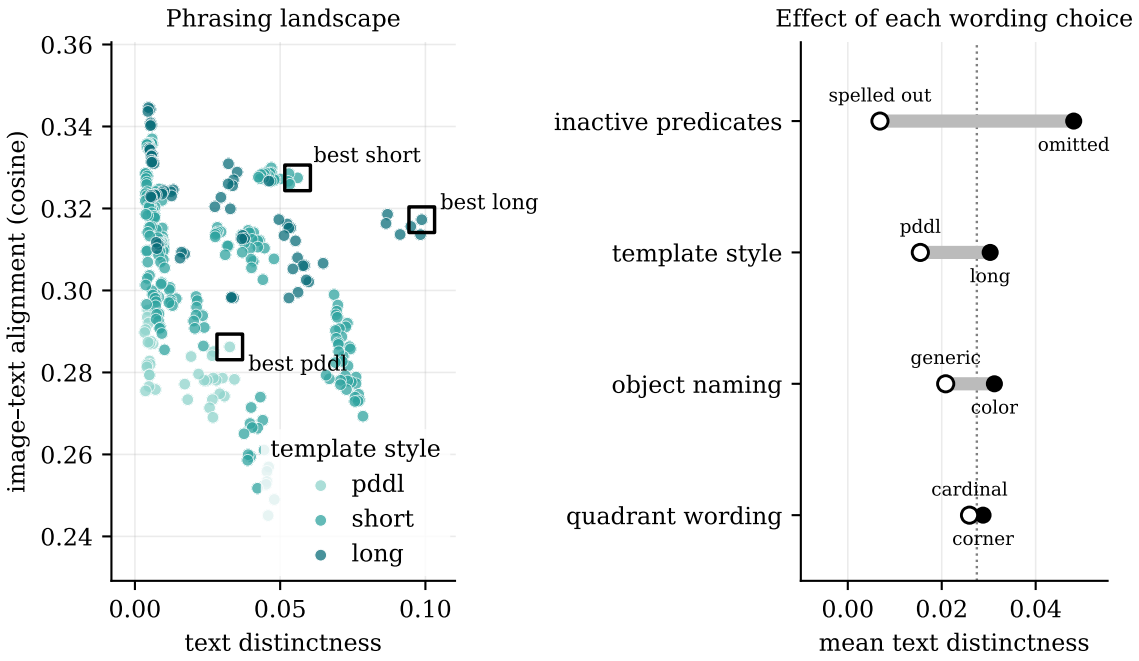

Figure 6.20: All 324 phrasing variants, scored on the goal states. *Left:* each variant placed by its text-distinctness (x) and image–text alignment (y), colored by template style; the tuple phrasings sit in the low-distinctness corner and the full sentences in the high-distinctness, high-alignment corner, with the best template of each template style ringed by a hollow square. *Right:* the distinctness swing each wording axis produces, from its worst (open marker) to its best (filled marker) level (dotted line is the overall mean); spelling out inactive predicates is by far the most harmful choice.

is not in the drawer" repeat across almost every goal and pull the sentences together. Across the three template styles the long narrative separates goals best, and no wording closes the gap to the visual goal, so the limiting factor is CLIP's grounding of the rendered images rather than the choice of words.

6.6.2 Intrinsic Quality of the Selected Templates

The best template of each template style — selected by the search above — is compared on the intrinsic metrics in Figure 6.21 and discussed below.

- **Two levels: extrinsic vs. intrinsic.** Template quality can be assessed intrinsically (the three properties of Section 5.3.7) and extrinsically (downstream task success when the policy is conditioned on that template style; the per-task extrinsic outcome by style is reported in Figure A.3). The intrinsic metrics are heuristic and not a definite measure. The working hypothesis is that they *rank* the template styles the same way the policy eventually does, so a cheap text/image encoding can give an idea of which template styles are worth an expensive policy run.

- **Alignment and CLIP's modality gap.** Figure 6.21 (left) shows the two natural-language styles align comparably with the goal image (cosine 0.32 for long and 0.33 for short), both above the PDDL tuples (0.29). These values must be read against CLIP's modality gap: image and text embeddings occupy two separate cones, so even a correctly matched image–caption pair stays well below cosine 1 [24]. An alignment around 0.32 is therefore about as close as the encoder's geometry allows for a correct pair, not a weak match.
- **Text distinctness.** The distinctness values are small in absolute terms (0.10 for long, 0.03 for PDDL) for the same geometric reason: each modality is confined to a narrow cone [24], so even unrelated sentences share a high baseline cosine and the informative quantity is the *relative* spread. There the gap is clear: the long style separates goals about three times further than the PDDL tuples, which differ only in a few parenthesized tokens.
- **Discriminability and retrieval.** A direct image-to-text retrieval check — does a goal frame's CLIP image embedding return its own rendered sentence over those of other goals? — pools to roughly the chance rate ($1/n_{\text{states}}$) for every template style, and its per-style ordering is *not* stable: which style scores highest flips with the arbitrary subset of goal states used to measure it, so it gives no reliable ranking and is not plotted. This is the expected regime for zero-shot CLIP: the goal frames are rendered simulator images far from its training distribution, and the goals turn on the back/front and left/right quadrant relations that vision–language models handle poorly [5, 18]. This untrained nearest-neighbor test on the raw embeddings need not bound what the policy does: the critic is trained on top of the frozen embeddings and can use structure a plain cosine lookup discards.

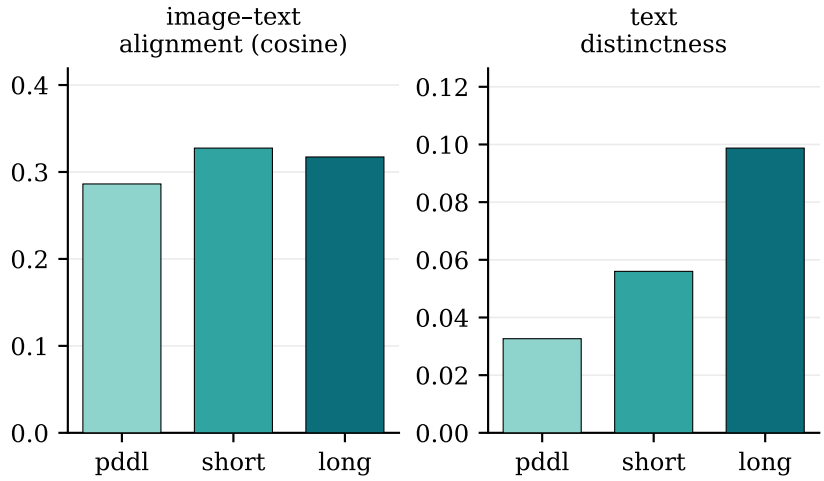

Figure 6.21: Intrinsic quality of the best template of each template style (PDDL, short, long) at skill-level detail, computed by encoding rendered goal frames and their templated sentences with CLIP (no policy involved). *Left:* cosine alignment between a goal's image embedding and its own sentence — read against CLIP's modality gap, where even matched image–text pairs sit well below cosine 1 [24]. *Right:* text-space distinctness (1− mean pairwise cosine). The long style separates goals best in text space (distinctness) and is level with the short style on alignment; the PDDL tuples are the least distinct and least aligned, differing only in a few parenthesized tokens. Image-to-text discriminability is not shown: it sits at chance for every style and its ordering is unstable across goal-state subsets (see text).

7 Conclusion

7.1 Summary

This thesis asked whether a contrastive embedding space can act as a shared *interface* between a low-level visual policy and high-level, abstract goals: whether a CLIP-encoded sentence or symbolic predicate vector carries enough of a goal to condition the same contrastive policy that a goal image does. To answer it we assembled a planner → language → CLIP → contrastive-policy pipeline on a simulated Sawyer drawer, pick-and-place, and push domain, froze the encoder, and trained a single controller with a two-stage image-then-text curriculum, then compared four interchangeable goal representations across three sub-goal granularities. The evidence gives a *qualified yes*: abstract CLIP goals do drive the policy, but they fall short of the oracle CLIP-image goal (stage 1) and the raw-image baseline.

- **RQ0 (CLIP interface).** The CLIP embedding space indeed drives a single goal-conditioned policy across all five tasks with a mean success rate of 0.76, the same as the raw-image baseline. When training the policy with (CLIP embedded) image goals and then switching to (CLIP embedded) text goals without further training, the policy continues to succeed, but worsens significantly. That shows that in principle the CLIP embedding space can serve as a shared interface between a low-level policy and high-level, abstract goals.
- **RQ1 (feasibility) and the baseline.** A policy conditioned on the CLIP embedding of symbolic predicates reaches all five tasks and never collapses. At a single CLIP-text checkpoint with a fixed sub-goal granularity, our approach ties or beats the raw-image baseline on the two tasks but trails it overall (∼58% against ∼76%), mainly on the pnp+drawer and push+drawer composites — though roughly half of that shortfall is the strict success criterion rejecting goal-reaching rollouts rather than a policy failure. The optimistic ceiling of the CLIP-text policy — the best checkpoint per task and the best granularity per task — matches or beats the baseline on three tasks, leaving only pnp and pnp+drawer below it. That shows that one can achieve high accuracy with symbolic goals.
- **RQ2 (encoding vs. information).** With the goals level-matched to the same information, the *encoding* still matters on the composite goals (CLIP-text > `pddl-direct` > voxel), yet the symbolic and language encodings almost converge on the isolated single skills. The residual gap is therefore one of *composing* goals, not of the information the goal representation carries.

- **RQ3 (granularity).** Finer decomposition does not help uniformly; it helps the abstract goals (CLIP-text, pddl-direct and voxel) but not the oracle CLIP-image goals. For the oracle CLIP-image, supplying just the one final goal is best and providing sub-goals is worse. For every other goal type the effect reverses and sub-goals do help: all three peak at the *medium* granularity, where CLIP-text edges the oracle CLIP-image goal at that same medium decomposition (57 against 54) — but only because medium sub-goals already weaken the image goal; the image goal's own best (coarse, 73) stays well above CLIP-text's best. The gain is non-monotone and comes from the granularity at which the goal is issued — overspecifying the sub-goals with additional non-essential information does not help.
- **RQ4 (pretraining).** Neither CLIP finetuning nor the two-stage training curriculum improves the downstream metric. Adapting the CLIP image encoder with LoRA sharpens image-to-text retrieval by an order of magnitude but does not carry over to control, and the two-stage curriculum is no better than training the text-goal policy from scratch. Both are therefore justified by convenience — an off-the-shelf encoder and a shared, reusable backbone — rather than performance, and the headroom to the oracle ceiling is not closed by pretraining at this scale.

Taken together, the CLIP embedding space can indeed serve as a shared interface between a low-level policy and high-level, abstract goals and dividing a composite task into multiple sub-goals can help the policy in that interface. The policy that was trained with CLIP embedded images (stage 1) with its best granularity — which does not issue sub-goals at all — ties with the raw-image baseline on mean success. When switching to CLIP embedded text — either using a two-stage curriculum or training from scratch — the policy falls by about 18%. The decomposition gives mixed signals: the CLIP embedded *image* goals do best without sub-goals, while the CLIP embedded *text* goals do best with sub-goals. Finetuning the CLIP image encoder improves the image-to-text retrieval but does not improve the downstream control performance. The main bottleneck is reaching several sub-goals one after another: on a single sub-goal CLIP-text matches the oracle image goal, but both drop once the sub-goals must be chained into a full task. The goal encoding still matters too: even for one whole-task goal given at once, CLIP-text trails the image (48 against 73), so part of its weakness is capturing a rich goal in text, not just chaining sub-goals.

7.2 Limitations

The following limitations of the study are worth noting:

- **Weak visual grounding.** The rendered PyBullet scenes look very different from the natural images CLIP was trained on, so the off-the-shelf encoder aligns them poorly with text: the image-to-text retrieval of Section 6.6 is at chance. Part of the gap between the

abstract and image goals is therefore due to this weak encoding of the domain rather than the goal interface itself, and a policy trained on a more realistic environment could have yielded different results.

- **Contrastive adaptation without hard negatives.** The optional CLIP image-encoder adaptation is trained with in-batch InfoNCE and no explicit hard-negative mining. Because many frames of a rollout share an *identical* templated goal sentence, some in-batch negatives in fact state the same goal as the positive — effectively false negatives that both cap the in-batch retrieval metric and can leave near-identical goal states under-separated. We cannot rule out that a different finetuning scheme would have improved the downstream control performance.
- **Training instability and checkpoint sensitivity.** Offline contrastive-RL training is noisy: success swings substantially from checkpoint to checkpoint within a run and across seeds, so some of the reported numbers depend strongly on the checkpoint selection. With only two carried stage-1 checkpoints and five fixed evaluation seeds, the per-cell numbers carry uncertainty.
- **A strict geometric success criterion.** Success is scored by the environment's geometric goal check, which is stricter than the target symbolic goal: on the drawer composites several rollouts reach the target goal and terminate, yet are still counted as failures. On push+drawer this accounts for about half of the recorded failures (Table 6.2), so the composite success rates — and the apparent gap to the baseline on those tasks — understate the policy's true completion.

7.3 Future Work

The evaluation surfaced several concrete problems; for each we sketch how we believe it could be addressed.

- **Missed goal hand-off.** On push+drawer the policy reliably pushes but then leaves without operating the drawer (Table 6.2); because the drawer skill succeeds in isolation (10/10), this is a missed *hand-off* between sub-goals, not an inability to operate the drawer. The pipeline issues its sub-goals open-loop and ends the episode on a fixed schedule, so nothing catches the unmet drawer goal. The symbolic predicate detector validated in Section 5.1.4 supplies the missing run-time signal: checking the target predicates at each sub-goal boundary lets the controller re-issue the drawer sub-goal instead of terminating, giving the policy another attempt to complete the hand-off. Another option is to show the policy longer trajectories during training which could help it learn to complete the hand-off. Both of these ideas tackle only the four *genuine* push+drawer failures; the remaining five are a scoring artifact where the symbolic goal is reached but the geometric success criterion rejects it. This can be addressed by making sure the predicate detection aligns perfectly with the geometric success criterion.

- **Adaptive decomposition.** A single fixed sub-goal granularity is not optimal: the best granularity is non-monotone and differs per task. The oracle analysis of Figure 6.13 points to a fix — because sequential success tracks the *product* of the isolated per-sub-goal success rates except where a particular goal boundary breaks the hand-off, the isolated per-sub-goal success is a cheap diagnostic for detecting such boundaries — and hence for driving an *adaptive*, per-task choice of granularity rather than a single fixed decomposition.
- **Stronger encoder.** Off-the-shelf CLIP grounds the rendered scenes only weakly — image-to-text retrieval sits at chance — which caps how well any goal aligns with the observations. A stronger, spatial-reasoning vision-language encoder would attack this weakness from the model side.
- **Sharper goal separation.** Finetuning the vision-language encoder did not employ hard-negative mining, so sub-goals stay under-separated and the policy confuses them. Mining hard negatives that deliberately flip a single predicate would supply the sharper contrast the in-batch negatives do not have. This is independent of whether CLIP or another encoder is being used. We chose LoRA to reduce catastrophic forgetting, but a more aggressive finetuning scheme could also be explored.
- **Stronger evidence.** Given the noisy nature of offline contrastive-RL training, training with more random seeds, hyperparameters, and doing more evaluation rollouts would provide stronger evidence for the conclusions drawn in this thesis.
- **From-scratch training.** Specifically from-scratch training has so far been underexplored compared to the two-stage curriculum. The from-scratch training, since it performs at least as well as the two-stage curriculum, is worth exploring to see whether it can be improved further.

A Appendix

A.1 Implementation and Hyperparameters

This section collects the implementation details of the Stage-1 sweep grid, optimizer, and contrastive-loss settings, together with the CLIP LoRA pretraining hyperparameters.

Table A.1: CLIP image-encoder LoRA adaptation. The adapter rank and learning rate were swept, and the best model was selected by held-out image-to-text retrieval accuracy (selected values in bold: rank 4, learning rate 10^{-4}). The low-rank update is merged into the encoder's weights W after training, so the deployed encoder is a plain CLIP image encoder.

Setting	Value
Adapted encoder	CLIP image encoder (text encoder frozen)
Target modules	FFN projections `c_fc`, `c_proj`
Adapter rank r	**4** (swept $\{4, 8, 16\}$)
LoRA scaling α	1.0 (update scaled by α/r)
Adapter dropout	0.1
Learning rate	$\mathbf{10^{-4}}$ (swept $\{10^{-5}, 5\times10^{-5}, 10^{-4}\}$)
Epochs	100 (best epoch by held-out retrieval)
Batch size	128
Optimizer	AdamW, weight decay 0.01, gradient clip 1.0
InfoNCE temperature	0.07
Adaptation data	300 rollouts, one frame every 4 steps; ≈8,000 train / 2,000 held-out frames (90/10 by rollout)

Table A.2: Stage-1 CLIP-image contrastive-RL sweep: the five swept axes ($2 \times 2 \times 3 \times 2 \times 2 = 48$ runs) and the fixed settings. Each run is trained for a fixed budget of 500 epochs.

Setting	Value(s)
Swept (48 runs)	
Goal trajectory granularity	skill, sub-skill
Batch size	1024, 2048
Hidden sizes (MLP)	[512, 512, 256], [1024] × 4, [2048] × 4
Learning rate	10^{-4}, 3×10^{-4}
Augmentation mode	observation-only, observation+goal
Fixed	
Representation dim	16 (no representation normalization)
Critic	twin Q
Behavior-cloning coefficient	0.05
Augmentation probability	0.5
Policy	Gaussian, std 0.15
Observation / action dim	512 / 4
Budget	500 epochs × 1000 steps, snapshot every 10
Training seed	single ($s0$)

A.2 Task Specifications

This section gives the two task-level details deferred from Chapter 5: the per-task initial and goal symbolic states of the five evaluation tasks (Table A.3), and the geometric thresholds by which every predicate of Table 5.2 is detected from the low-dimensional simulator state (Table A.4).

Per-task initial and goal symbolic states. Table A.3 lists, for each of the five evaluation tasks, the full set of active command-level predicates at the initial frame and at the whole-task goal frame, computed by `state_to_symbolic` on the scripted expert's own start and end states. Under the closed-world assumption any predicate not listed is false. The goal column shows the *complete* symbolic state that the coarse whole-task goal encodes, of which the success-defining goal predicate(s) of Table 5.1 (bold) are a subset — the remaining facts are the scene context that carries over unchanged. The cylinder quadrant follows the predicate numbering of Table 5.2; only the two push tasks change it, and only the drawer and pick-and-place tasks change the drawer or the small-object location.

Full predicate-detection thresholds. Table A.4 gives the geometric test and numeric threshold behind every predicate. All distances are in meters in the drawer-relative simulator frame. The nine command-level thresholds are ported from the reference success metrics [46]. The small-object placement predicates (`on_drawer`, `in_drawer`, `on_table`) combine a 3-D

Table A.3: Initial and goal command-level symbolic states of the five evaluation tasks, in the canonical task order of Table 5.1. Each cell is the set of active predicates (closed-world: unlisted predicates are false), joined by ∧ in canonical order (small-object location, drawer state, cylinder quadrant). **Bold** marks the success-defining goal predicate(s) of Table 5.1; the other goal facts are unchanged scene context. † The block starts resting inside the drawer, but at its settled pose the `in_drawer` test misses the 0.01 m vertical tolerance (Table A.4), so no small-object location predicate is active at the very first frame.

Task	Initial symbolic state	Goal symbolic state
drawer	`on_drawer` ∧ `drawer_open` ∧ `quadrant_2`	`on_drawer` ∧ **`drawer_closed`** ∧ `quadrant_2`
pnp	`on_table` ∧ `drawer_closed` ∧ `quadrant_2`	**`on_drawer`** ∧ `drawer_closed` ∧ `quadrant_2`
pnp+drawer	`drawer_open` ∧ `quadrant_3`†	**`on_drawer`** ∧ **`drawer_closed`** ∧ `quadrant_3`
push+drawer	`on_table` ∧ `drawer_closed` ∧ `quadrant_1`	`on_table` ∧ **`drawer_open`** ∧ **`quadrant_2`**
drawer+push	`on_drawer` ∧ `drawer_open` ∧ `quadrant_2`	`on_drawer` ∧ **`drawer_closed`** ∧ **`quadrant_3`**

proximity radius with a tight vertical tolerance; `quadrant_k` is a nearest-center assignment and so carries no radius.

Table A.4: Geometric test and numeric threshold for each of the twelve predicates of Table 5.2, computed from the 32-d `state_observation`. Distances are Euclidean norms in meters; "planar" is the xy-distance and "vertical" the $|\Delta z|$. The command-level thresholds match the reference success metrics; the three contact thresholds were tuned on the demonstration set (Section 5.1.4).

Predicate	Geometric test on the state	Threshold
`on_drawer`	distance from small object to the drawer-top reference	0.08 m 3-D, 0.01 m vertical
`in_drawer`	distance from small object to the in-drawer reference	0.08 m 3-D, 0.01 m vertical
`on_table`	small object off both drawer references and at table height	0.08 m 3-D, 0.01 m vertical
`drawer_open`	planar displacement of the handle from its closed pose	> 0.18 m planar
`drawer_closed`	negation of `drawer_open`	—
`quadrant_k`	nearest of four table-quadrant centers ($x \in \{0.61, 0.73\}$, $y \in \{0.09, -0.089\}$)	nearest center
`grasping_small_object`	end-effector to small object, 3-D	< 0.05 m 3-D
`gripper_around_handle`	end-effector to drawer handle	< 0.03 m planar, < 0.02 m vertical
`touching_large_object`	end-effector descended onto the cylinder	< 0.085 m planar, < 0.04 m vertical

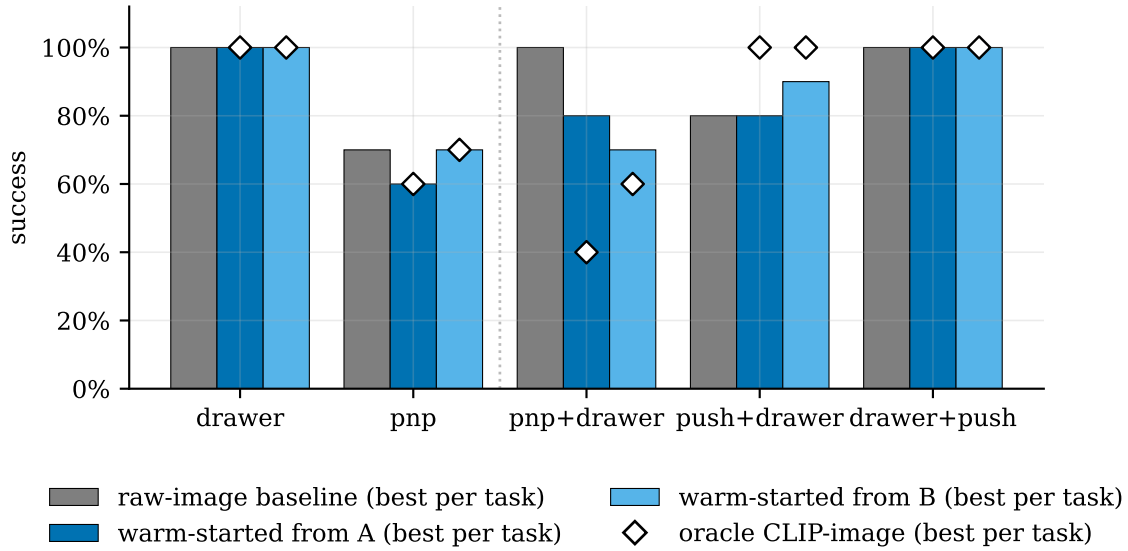

Figure A.1: Full per-task *ceiling* of CLIP-text (warm-started from A and B), the raw-image baseline and the oracle CLIP-image goal (diamonds): for each task, the best success over *any* configuration — any warm-started run, checkpoint, granularity (coarse, medium or fine) and, for CLIP-text, text-detail — taken among fully task-covered checkpoints. Because the coarse whole-task goal is *allowed* here, push+drawer recovers to 80/90 (matching or clearing the baseline's 80) by issuing a single whole-task goal there. With that freedom CLIP-text ties or beats the baseline on the drawer task and the push composite but still trails on pnp+drawer (80/70 vs 100). This is an optimistic upper bound on the hierarchical approach, not a deployable configuration.

A.3 Supplementary Figures

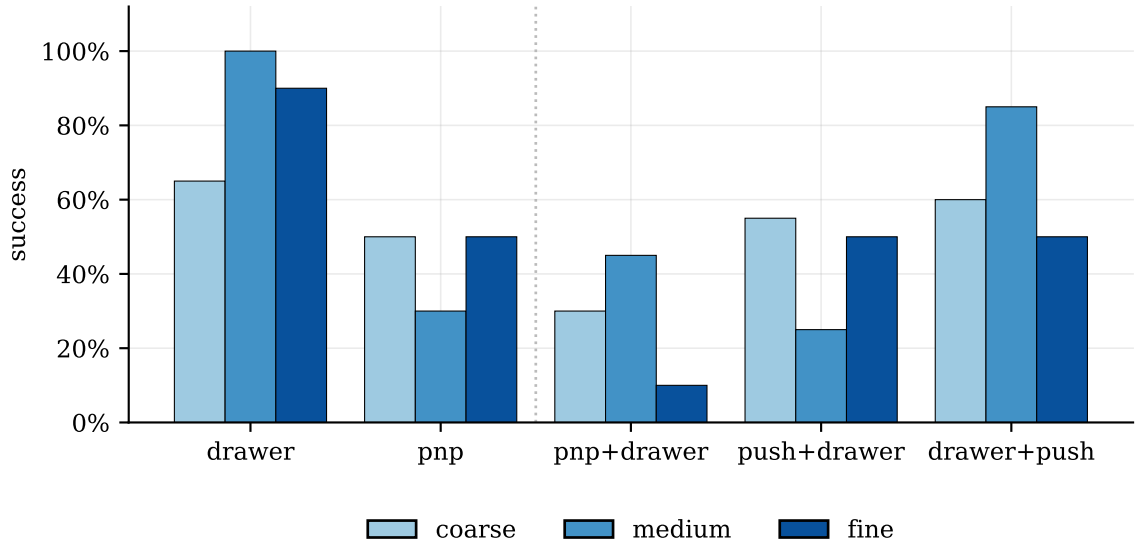

Figure A.2: The same per-task granularity breakdown as Figure 6.15, warm-started from B, at each cell's best checkpoint over the template sweep (mean of the two text-detail tiers). The per-task pattern broadly matches warm-started from A, with medium strongest on the drawer tasks; the push+drawer skill hand-off is milder here (25%) than the collapse warm-started from A.

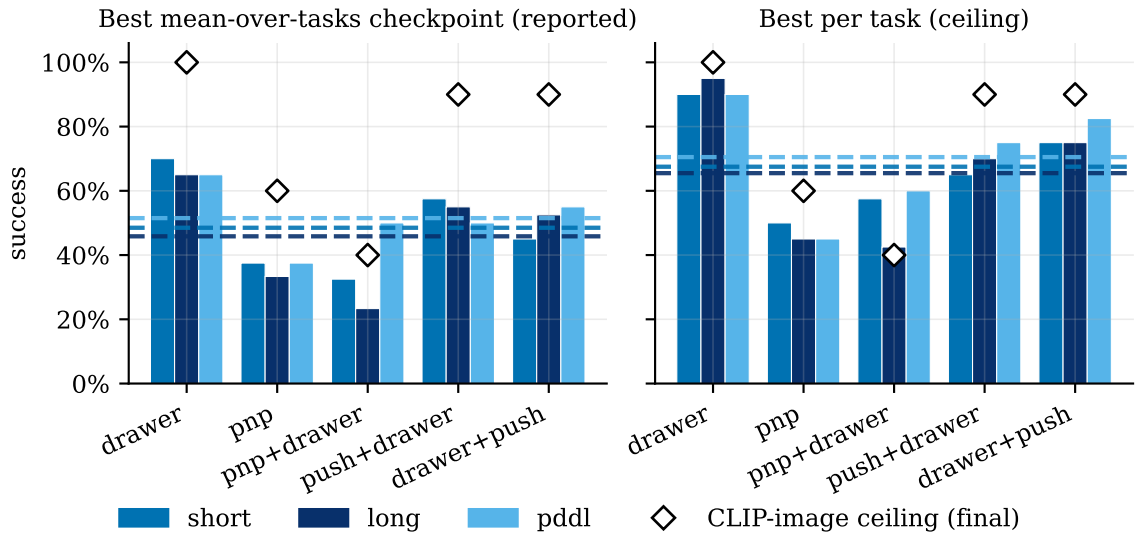

Figure A.3: Stage-2 CLIP-text whole-task success per evaluation task, grouped by template style (mean over the two versions and the two trained-detail tiers), under two checkpoint-selection conventions: the single best mean-over-tasks checkpoint (left — the reported, deployable value) and the best-per-task ceiling (right — each task picks its own best checkpoint). Here the checkpoint is the *only* optimistic axis: the granularity stays fixed at whole-task and the two versions and two trained-detail tiers are averaged, not maximized over. This mirrors the per-task best-checkpoint convention of Figure A.1, whose ceiling additionally frees granularity and text detail. The blue dashed lines are per-style means; white diamonds mark the same model's CLIP-image ceiling (coarse, best checkpoint). No template style dominates across tasks. Shown warm-started from A (fixed, not optimized over); warm-started from B is analogous.

List of References

[1] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, pp. 5055–5065, ISBN: 9781510860964.

[2] b. ichter brian, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, D. Kalashnikov, S. Levine, Y. Lu, C. Parada, K. Rao, P. Sermanet, A. T. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, M. Yan, N. Brown, M. Ahn, O. Cortes, N. Sievers, C. Tan, S. Xu, D. Reyes, J. Rettinghouse, J. Quiambao, P. Pastor, L. Luu, K.-H. Lee, Y. Kuang, S. Jesmonth, N. J. Joshi, K. Jeffrey, R. J. Ruano, J. Hsu, K. Gopalakrishnan, B. David, A. Zeng, and C. K. Fu, "Do as i can, not as i say: Grounding language in robotic affordances," in *Proceedings of The 6th Conference on Robot Learning*, K. Liu, D. Kulic, and J. Ichnowski, Eds., ser. Proceedings of Machine Learning Research, vol. 205, PMLR, 14–18 Dec 2023, pp. 287–318. [Online]. Available: https://proceedings.mlr.press/v205/ichter23a.html.

[3] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. S. Ryoo, G. Salazar, P. R. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. H. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, "RT-1: Robotics Transformer for Real-World Control at Scale," in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, Jul. 2023. DOI: 10.15607/RSS.2023.XIX.025.

[4] G. Chalvatzaki, A. Younes, D. Nandha, A. T. Le, L. Ribeiro, and I. Gurevych, "Learning to reason over scene graphs: A case study of finetuning gpt-2 into a robot language model for grounded task planning," 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2023.1221739/full?&utm_source=Email_to_authors_&utm_medium=Email&utm_content=T1_11.5e1_author&utm_campaign=Email_publication&field=&journalName=Frontiers_in_Robotics_and_AI&id=1221739.

[5] B. Chen, Z. Xu, S. Kirmani, B. Ichter, D. Sadigh, L. Guibas, and F. Xia, "Spatialvlm: Endowing vision-language models with spatial reasoning capabilities," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 14 455–14 465. DOI: 10.1109/CVPR52733.2024.01370.

[6] W. Chen, O. Mees, A. Kumar, and S. Levine, *Vision-language models provide promptable representations for reinforcement learning*, 2024. arXiv: 2402.02651 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2402.02651.

[7] B. Eysenbach, R. R. Salakhutdinov, and S. Levine, "Search on the replay buffer: Bridging planning and reinforcement learning," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/5c48ff18e0a47baaf81d8b8ea51eec92-Paper.pdf.

[8] B. Eysenbach, R. Salakhutdinov, and S. Levine, *C-learning: Learning to achieve goals via recursive classification*, ICLR 2021, 2021. arXiv: 2011.08909 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2011.08909.

[9] B. Eysenbach, T. Zhang, S. Levine, and R. R. Salakhutdinov, "Contrastive learning as goal-conditioned reinforcement learning," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., 2022, pp. 35 603–35 620. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/e7663e974c4ee7a2b475a4775201ce1f-Paper-Conference.pdf.

[10] K. Fang, P. Yin, A. Nair, and S. Levine, "Planning to practice: Efficient online fine-tuning by composing goals in latent space," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 4076–4083. DOI: 10.1109/IROS47612.2022.9981999.

[11] 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. Volume 4, 2021, pp. 265–293, 2021, ISSN: 2573-5144. DOI: 10.1146/annurev-control-091420-084139. [Online]. Available: https://www.annualreviews.org/content/journals/10.1146/annurev-control-091420-084139.

[12] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," in *Proceedings of the Conference on Robot Learning*, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., ser. Proceedings of Machine Learning Research, vol. 100, PMLR, 30 Oct–01 Nov 2020,

pp. 1025–1037. [Online]. Available: https://proceedings.mlr.press/v100/gupta20a.html.

[13] J. Herzog, J. Liu, and Y. Wang, *Domain-conditioned scene graphs for state-grounded task planning*, 2025. arXiv: 2504.06661v2 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2504.06661v2.

[14] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, *Lora: Low-rank adaptation of large language models*, 2021. arXiv: 2106.09685 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2106.09685.

[15] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, T. Jackson, N. Brown, L. Luu, S. Levine, K. Hausman, and b. ichter brian, "Inner monologue: Embodied reasoning through planning with language models," in *Proceedings of The 6th Conference on Robot Learning*, K. Liu, D. Kulic, and J. Ichnowski, Eds., ser. Proceedings of Machine Learning Research, vol. 205, PMLR, 14–18 Dec 2023, pp. 1769–1782. [Online]. Available: https://proceedings.mlr.press/v205/huang23c.html.

[16] L. Illanes, X. Yan, R. Toro Icarte, and S. A. McIlraith, "Symbolic plans as high-level instructions for reinforcement learning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 540–550. DOI: 10.1609/icaps.v30i1.6750. [Online]. Available: https://ojs.aaai.org/index.php/ICAPS/article/view/6750.

[17] Y. Jiang, S. Gu, K. Murphy, and C. Finn, "Language as an abstraction for hierarchical deep reinforcement learning," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[18] A. Kamath, J. Hessel, and K.-W. Chang, "What's "up" with vision-language models? investigating their struggle with spatial reasoning," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 9161–9175. DOI: 10.18653/v1/2023.emnlp-main.568. [Online]. Available: https://aclanthology.org/2023.emnlp-main.568/.

[19] L. Keller, D. Tanneberg, and J. Peters, "Neuro-symbolic imitation learning: Discovering symbolic abstractions for skill learning," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 6519–6526. DOI: 10.1109/ICRA55743.2025.11127692.

[20] A. Khandelwal, L. Weihs, R. Mottaghi, and A. Kembhavi, "Simple but effective: Clip embeddings for embodied ai," in *2022 IEEE/CVF Conference on Computer Vision and*

Pattern Recognition (CVPR), 2022, pp. 14 809–14 818. DOI: 10.1109/CVPR52688.2022.01441.

[21] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, *Reinforcement learning with augmented data*, 2020. arXiv: 2004.14990 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2004.14990.

[22] Y. Lee,
bibinitperiod Lim, A. Anandkumar, and Y. Zhu, "Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization," English, *Proceedings of Machine Learning Research*, vol. 164, pp. 406–416, 2021, Publisher Copyright: © 2021 Proceedings of Machine Learning Research. All rights reserved.; 5th Conference on Robot Learning, CoRL 2021 ; Conference date: 08-11-2021 Through 11-11-2021, ISSN: 2640-3498.

[23] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 9493–9500. DOI: 10.1109/ICRA48891.2023.10160591.

[24] W. Liang, Y. Zhang, Y. Kwon, S. Yeung, and J. Zou, "Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22, New Orleans, LA, USA: Curran Associates Inc., 2022, ISBN: 9781713871088.

[25] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, *Llm+p: Empowering large language models with optimal planning proficiency*, 2023. arXiv: 2304.11477 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2304.11477.

[26] G. Liu, M. Tang, and B. Eysenbach, *A single goal is all you need: Skills and exploration emerge from contrastive rl without rewards, demonstrations, or subgoals*, 2024. arXiv: 2408.05804 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2408.05804.

[27] Y. J. Ma, V. Kumar, A. Zhang, O. Bastani, and D. Jayaraman, "Liv: Language-image representations and rewards for robotic control," in *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML'23, Honolulu, Hawaii, USA: JMLR.org, 2023.

[28] A. Majumdar, G. Aggarwal, B. Devnani, J. Hoffman, and D. Batra, "Zson: Zero-shot object-goal navigation using multimodal goal embeddings," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22, New Orleans, LA, USA: Curran Associates Inc., 2022, ISBN: 9781713871088.

[29] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7327–7334, 2022. DOI: 10.1109/LRA.2022.3180108.

[30] S. Nair, E. Mitchell, K. Chen, B. Ichter, S. Savarese, and C. Finn, "Learning language-conditioned robot behavior from offline data and crowd-sourced annotation," in *Proceedings of the 5th Conference on Robot Learning*, A. Faust, D. Hsu, and G. Neumann, Eds., ser. Proceedings of Machine Learning Research, vol. 164, PMLR, Aug. 2022, pp. 1303–1315. [Online]. Available: https://proceedings.mlr.press/v164/nair22a.html.

[31] N. Nguyen, M. N. Vu, T. D. Ta, B. Huang, T. Vo, N. Le, and A. Nguyen, "Robotic-clip: Fine-tuning clip on action data for robotic applications," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 5930–5936. DOI: 10.1109/ICRA55743.2025.11127829.

[32] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 18–24 Jul 2021, pp. 8748–8763. [Online]. Available: https://proceedings.mlr.press/v139/radford21a.html.

[33] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Proceedings of the 5th Conference on Robot Learning*, A. Faust, D. Hsu, and G. Neumann, Eds., ser. Proceedings of Machine Learning Research, vol. 164, PMLR, Aug. 2022, pp. 894–906. [Online]. Available: https://proceedings.mlr.press/v164/shridhar22a.html.

[34] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Proceedings of The 6th Conference on Robot Learning*, K. Liu, D. Kulic, and J. Ichnowski, Eds., ser. Proceedings of Machine Learning Research, vol. 205, PMLR, 14–18 Dec 2023, pp. 785–799. [Online]. Available: https://proceedings.mlr.press/v205/shridhar23a.html.

[35] T. Silver, R. Chitnis, J. Tenenbaum, L. P. Kaelbling, and T. Lozano-Pérez, "Learning symbolic operators for task and motion planning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Prague, Czech Republic: IEEE Press, 2021, pp. 3182–3189. DOI: 10.1109/IROS51168.2021.9635941. [Online]. Available: https://doi.org/10.1109/IROS51168.2021.9635941.

[36] C. H. Song, V. Blukis, J. Tremblay, S. Tyree, Y. Su, and S. Birchfield, *Robospatial: Teaching spatial understanding to 2d and 3d vision-language models for robotics*, CVPR 2025 (Oral), 2024. arXiv: 2411.16537 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2411.16537.

[37] K. Stein, D. Fišer, J. Hoffmann, and A. Koller, "Automating the generation of prompts for llm-based action choice in pddl planning," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 35, no. 1, pp. 250–259, Sep. 2025. DOI:

10.1609/icaps.v35i1.36126. [Online]. Available: https://ojs.aaai.org/index.php/ICAPS/article/view/36126.

[38] T. Thrush, R. Jiang, M. Bartolo, A. Singh, A. Williams, D. Kiela, and C. Ross, "Winoground: Probing vision and language models for visio-linguistic compositionality," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 5228–5238. DOI: 10.1109/CVPR52688.2022.00517.

[39] M. Tschannen, A. Gritsenko, X. Wang, M. F. Naeem, I. Alabdulmohsin, N. Parthasarathy, T. Evans, L. Beyer, Y. Xia, B. Mustafa, O. Hénaff, J. Harmsen, A. Steiner, and X. Zhai, *Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features*, 2025. arXiv: 2502.14786 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2502.14786.

[40] R. Virwani and R. Suryawanshi, *Loop: A plug-and-play neuro-symbolic framework for enhancing planning in autonomous systems*, 2025. arXiv: 2508.13371 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2508.13371.

[41] T. Wang, A. Torralba, P. Isola, and A. Zhang, "Optimal goal-reaching reinforcement learning via quasimetric learning," in *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML'23, Honolulu, Hawaii, USA: JMLR.org, 2023.

[42] S. Woo, D. Kim, D. Cho, and I. S. Kweon, "Linknet: Relational embedding for scene graph," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/58238e9ae2dd305d79c2ebc8c1883422-Paper.pdf.

[43] W. Yang, A. Tikna, Y. Zhao, Y. Zhang, L. Palopoli, M. Roveri, and J. Pajarinen, "Extracting visual plans from unlabeled videos via symbolic guidance," in *9th Annual Conference on Robot Learning*, 2025. [Online]. Available: https://openreview.net/forum?id=HMcBBIg1Th.

[44] M. Yüksekgönül, F. Bianchi, P. Kalluri, D. Jurafsky, and J. Zou, "When and why vision-language models behave like bags-of-words, and what to do about it?" In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, OpenReview.net, 2023. [Online]. Available: https://openreview.net/forum?id=KRLUvxh8uaX.

[45] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, and J. Lee, "Transporter networks: Rearranging the visual world for robotic manipulation," in *Proceedings of the 2020 Conference on Robot Learning*, J. Kober, F. Ramos, and C. Tomlin, Eds., ser. Proceedings of Machine Learning Research, vol. 155, PMLR, 16–18 Nov 2021, pp. 726–747. [Online]. Available: https://proceedings.mlr.press/v155/zeng21a.html.

[46] C. Zheng, B. Eysenbach, H. Walke, P. Yin, K. Fang, R. Salakhutdinov, and S. Levine, "Stabilizing contrastive rl: Techniques for robotic goal reaching from offline data," in *International Conference on Learning Representations*, vol. 2024, 2024, pp. 28 236–28 264. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2024/file/78b9d95f6bb13b080c2a68bdea54cdbb-Paper-Conference.pdf.

[47] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, Q. Vuong, V. Vanhoucke, H. Tran, R. Soricut, A. Singh, J. Singh, P. Sermanet, P. R. Sanketi, G. Salazar, M. S. Ryoo, K. Reymann, K. Rao, K. Pertsch, I. Mordatch, H. Michalewski, Y. Lu, S. Levine, L. Lee, T.-W. E. Lee, I. Leal, Y. Kuang, D. Kalashnikov, R. Julian, N. J. Joshi, A. Irpan, B. Ichter, J. Hsu, A. Herzog, K. Hausman, K. Gopalakrishnan, C. Fu, P. Florence, C. Finn, K. A. Dubey, D. Driess, T. Ding, K. M. Choromanski, X. Chen, Y. Chebotar, J. Carbajal, N. Brown, A. Brohan, M. G. Arenas, and K. Han, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Proceedings of The 7th Conference on Robot Learning*, J. Tan, M. Toussaint, and K. Darvish, Eds., ser. Proceedings of Machine Learning Research, vol. 229, PMLR, Jun. 2023, pp. 2165–2183. [Online]. Available: https://proceedings.mlr.press/v229/zitkovich23a.html.

Declaration on the Use of AI-Based Tools

In the course of preparing this thesis, the author made use of AI-based assistance tools. Specifically, the author used **GitHub Copilot** and **Claude** (Anthropic) to assist with:

- phrasing, grammatical refinement, and
- writing and debugging source code.