COARSE INTENT, FINE PHYSICS: EQUIPPING VLM AGENTS WITH LATENT WORLD MODEL FORESIGHT FOR OPEN-LOOP MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Frontier agents such as Claude Code with Claude Opus 5.5 and Codex with GPT-6 Astra are increasingly used out of the box in robotics, yet they cannot reliably anticipate the physical consequences of low-level actions. JEPA-style latent world models predict these consequences quickly, but planners built on them need goal images and exploit optimistic prediction errors, a failure we call hallucination. We give Claude Code access to DDP-WM through two tools, one previewing a proposed plan as keypoint clouds predicted under one hundred perturbations and one running a local cross-entropy search seeded by that plan, and the agent learns to use both from a single exploratory episode. On a Push-T protocol that requires moving the block 61 pixels and rotating it 77 degrees on average within one open-loop plan 2.4 times shorter than human demonstrations, MPC with DDP-WM solves 2 of 21 episodes and the agent alone 7, while the two tools raise this to 14 and 16. Replacing the clouds by single predictions lowers these numbers to 12 and 13, showing that the agent uses the predicted distribution to avoid hallucinated plans.

1 INTRODUCTION

Over the past year, general-purpose agents built on frontier models, such as Claude Code with Claude Opus 5.5 and Codex with GPT-6 Astra, have moved well beyond software engineering. Building on research into language agents that interleave reasoning with tool use (Yao et al., 2023; Schick et al., 2023) and operate software through purpose-built interfaces (Yang et al., 2024a), these agents write and debug their own tools and revise their strategy from feedback, and a growing number of publicly shared demonstrations show them driving simulated and real robots out of the box in zero-shot or few-shot settings. Earlier work used vision-language models (VLMs) in robotics as one component of a hand-designed pipeline, for instance as a task planner, a code generator or a source of spatial constraints (Ahn et al., 2022; Liang et al., 2023; Huang et al., 2023; 2024), or fine-tuned them on robot data into vision-language-action models (Brohan et al., 2023; Kim et al., 2024). Despite the attention that the new generation of agents attracts, they are rarely evaluated under controlled protocols, and it remains unclear how well they cope with tasks whose success hinges on precise physical prediction.

Physical prediction remains a weak point of language-centered models (Bear et al., 2021; Gao et al., 2024; Chow et al., 2025), and LeCun has repeatedly argued that the largest language models understand the physical world less well than a house cat (Mims, 2024). The alternative he advocates learns world models that predict in a latent representation space (LeCun, 2022; Assran et al., 2023), and recent JEPA-style instances such as DINO-WM (Zhou et al., 2025) and the more efficient DDP-WM (Yin et al., 2026) predict how frozen DINOv2 features (Oquab et al., 2024) evolve under candidate actions and plan with the cross-entropy method (CEM) (Rubinstein, 1999) inside a model predictive control (MPC) loop. These models forecast contact-rich dynamics quickly and precisely, yet they carry no semantic prior about the task, need a goal image to define the planning cost, and expose their planners to a failure mode that we call hallucination. A prediction error on its own is unbiased, whereas a planner that keeps the best of many predicted outcomes preferentially retains the samples whose error happens to be optimistic. This bias, known as the optimizer's curse in decision analysis (Smith & Winkler, 2006) and as model exploitation in model-based reinforcement learning (Kuru-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

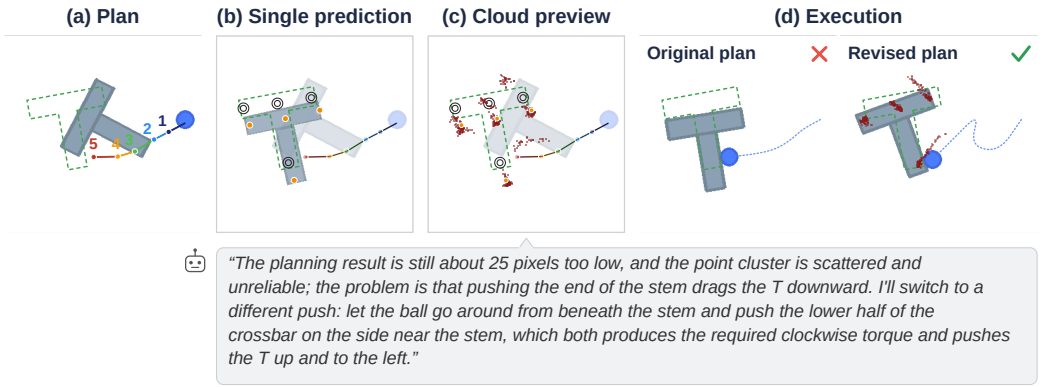

Figure 1: The same world model shown to the agent in two ways. (a) The agent proposes five macro actions, drawn as a numbered path, to push the block toward the goal pose shown as a dashed outline. (b) The DDP-WM rollout of the unperturbed plan places all four keypoints on their targets, and executing this plan fails. (c) One hundred rollouts of slightly perturbed versions of the same plan reveal a scattered, bimodal keypoint cloud, which the agent reads as an unreliable contact that may drag the block along or leave it behind. (d) Final states after executing the plan accepted on the basis of the single prediction (left) and the plan the agent adopts after revising it until all four clouds are tight (right). Only the revised plan reaches the goal. The speech bubble quotes the agent's reasoning after it has seen (c).

tach et al., 2018; Janner et al., 2019), appears as narrow spikes in the cost landscape and is strongest for CEM, which repeatedly concentrates its samples around the best-scoring candidates.

We study how the two lines of work can support each other on Push-T (Florence et al., 2021; Chi et al., 2023), where an agent pushes a T-shaped block to a target pose. Push-T is the contact-rich benchmark on which DINO-WM and DDP-WM are evaluated, and it tests whether a complete system understands collision, friction, rotation and rigid-body motion well enough to act on them. The protocol commonly used with latent world models samples 25-step segments from the test trajectories, yet the block does not move at all in 15% of them, the start already satisfies the success criterion in 31%, and the generous action budget lets a planner spend many actions on jitter and detours. Our protocol connects the first and middle frames of each of the 21 test trajectories, so the block must travel 61 pixels on 448-pixel observations and rotate by 77 degrees on average (Figure 2). Human demonstrations need about 59 low-level steps for this, while our agents must commit a single open-loop plan of 25 steps, a 2.4-fold gain in action efficiency that imitation learning cannot reach by construction and that search-based methods such as MPC or reinforcement learning can in principle attain. MPC with DDP-WM, which succeeds in 98% of the standard episodes (Yin et al., 2026), solves only 2 of our 21 episodes, and Claude Code alone solves 7.

We connect the two lines by exposing DDP-WM to Claude Code through two tools of increasing strength (Figure 3), without fine-tuning the agent or the world model. The preview tool rolls out one hundred slightly perturbed copies of a plan proposed by the agent and draws the predicted positions of four keypoints on the block, read out with a correspondence model trained on DINOv2 features, as point clouds over the observation, and the planning tool additionally runs a small CEM search seeded at the plan toward keypoint targets set by the agent under a rigid-body constraint. The agent keeps every decision, learns to use both tools from a single exploratory episode, and reasons explicitly about spread, clustering and prediction confidence, declining plans whose keypoints scatter widely even when the unperturbed prediction satisfies every constraint (Figure 1).

With the preview tool the agent solves 14 episodes and with the planning tool 16, against 2 for MPC alone and 7 for the bare agent. When each cloud is replaced by a single noise-free prediction, the two settings drop to 12 and 13 successes and in some episodes fall below the bare agent, because the world model then misleads the agent with precise-looking wrong predictions. The agent needs the world model in the first place because its own physical predictions are coarse, and a direct test shows that Claude Opus 5.5 predicts the keypoints of the block after five open-loop steps with an

error of 5.7 pixels on 224-pixel observations, against 1.9 pixels for DDP-WM (Section 4.3). Our contributions are as follows.

- A Push-T protocol that requires goal-directed, action-efficient pushing in a single open-loop plan, under which neither a frontier agent nor MPC with a latent world model performs well alone.
- A tool interface through which an off-the-shelf frontier agent uses a latent world model for distributional previews and for local planning seeded by its own proposals.
- An analysis of hallucination in world-model planning, together with evidence that distributional previews allow the agent to recognize and avoid it.

The code for the protocol and the tools will be released upon publication.

2 THE PUSH-T CHALLENGE

Environment and data. In Push-T, a circular pusher moves in the plane and has to push a T-shaped block into a target pose. We use the Push-T simulator, action space and dataset of DINO-WM (Zhou et al., 2025), which build on the environment of Chi et al. (2023), and control the pusher at 10 Hz. Following DINO-WM, the world model groups five consecutive low-level actions into one macro action, and every agent in our protocol outputs five macro actions that are linearly interpolated into 25 low-level actions and executed without feedback.

The standard protocol. Latent world models are commonly evaluated on Push-T by sampling a 25-step segment from a test trajectory and using its first and last frames as start and goal, which matches a planning horizon of five macro actions (Zhou et al., 2025; Yin et al., 2026). This protocol measures short-horizon planning fidelity well, and it poses a limited physical challenge. In 15% of the segments the block does not move at all and only the pusher moves, and in 31% the start pose already satisfies the success criterion, so a successful plan does not need to touch the block, and because a demonstration reached every goal within the same 25 steps, a planner can spend a large share of its actions on jitter or detours and still succeed. Under this protocol, MPC with DDP-WM succeeds in 98% of the episodes (Yin et al., 2026).

Our protocol. We take the first and the middle frame of each of the 21 test trajectories as start and goal. Every demonstration ends with the block on the green target region that the environment renders in the background, so world models trained on these demonstrations acquire a systematic pull toward it, and their predictions for pushes near the green target drift toward alignment with it. Restricting the episodes to the first half of each demonstration keeps the block away from this region, where the predictions of the world model are largely unbiased, and lets MPC be evaluated on its planning ability without a dataset bias that would make it believe the block had reached the target while it remained far from it. Reaching these goals requires translating the block by 61 pixels and rotating it by 77 degrees on average on 448-pixel observations, and the human demonstrations need 59 low-level steps on average to do so (Figure 2). The action budget stays at 25 low-level steps, committed as a single open-loop plan, so a successful plan has to be 2.4 times more action-efficient than the demonstration of the same episode. Imitation learning cannot exceed the efficiency of its demonstrations by construction, whereas search-based methods such as MPC and reinforcement learning can in principle, which makes the protocol a test of physical foresight and planning. We evaluate open-loop execution for three reasons. It isolates the ability to foresee the consequences of a complete action sequence, which closed-loop correction would mask. It matches how frontier agents operate, since they deliberate for a long time before each decision and cannot follow a 10 Hz control loop. It also places the agent, the world model and their combination behind one interface, so that their results are directly comparable.

Evaluation. An episode succeeds when the final pose of the block lies within 17.5 pixels and 20 degrees of the goal pose. Every agent configuration first plays one calibration episode, which is excluded from evaluation and lets the agent explore the environment and its tools, and then plays the 21 episodes in one continuous session.

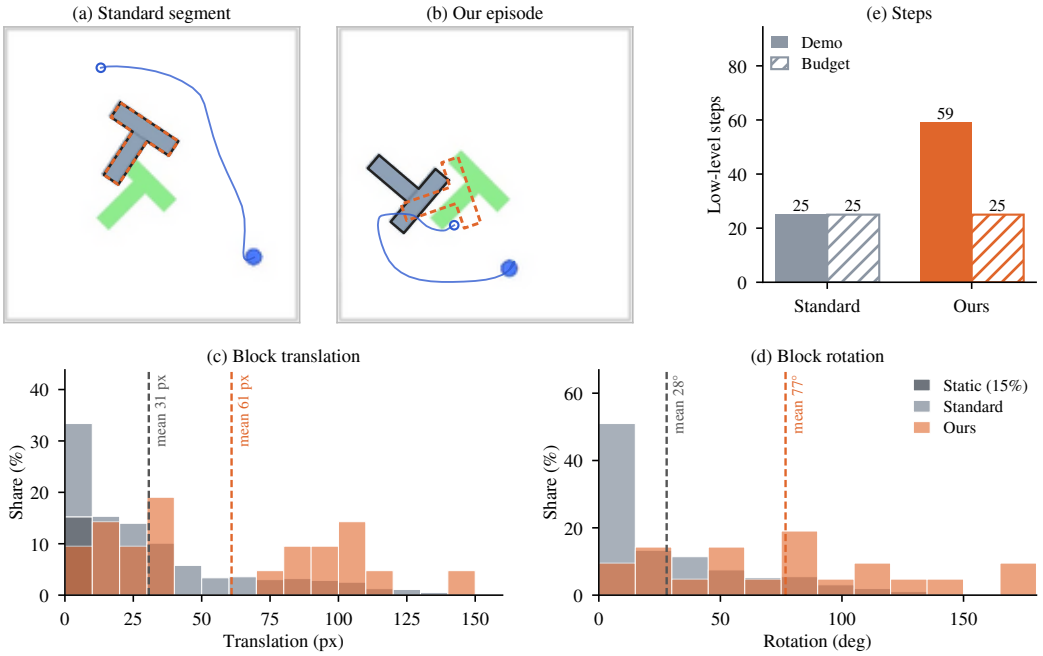

Figure 2: Standard and proposed Push-T protocols. (a) A typical 25-step segment of the standard protocol, in which the start pose (solid) and the goal pose (dashed) of the block coincide and only the pusher moves. (b) A typical episode of our protocol, connecting the first and middle frames of a test trajectory. (c, d) Distributions of block translation in pixels on 448-pixel observations and of block rotation in degrees between start and goal. The block does not move at all in 15% of the standard segments, whereas our episodes require 61 pixels of translation and 77 degrees of rotation on average. (e) Demonstration length and action budget, which are both 25 low-level steps in the standard protocol, while our demonstrations take 59 steps on average against the same budget of 25.

3 Method

Figure 3 summarizes the system. The agent receives the start observation o_0 and the goal image o_g and outputs a plan $\boldsymbol{a} = (a_1, \ldots, a_5)$ of five macro actions. Before committing, it may call two tools built on a latent world model, which we describe after the world model and the keypoint readout they share.

3.1 Latent World Model and Keypoint Readout

Let E denote the frozen DINOv2 encoder and f the DDP-WM transition model, which maps the current feature map $z_0 = E(o_0)$ and a plan $\boldsymbol{a}$ to predicted feature maps $\hat{z}_1, \ldots, \hat{z}_5$. DDP-WM confines most of its computation to the sparse regions where physical interaction changes the scene and updates the background with a low-rank correction, which gives it a single-step throughput of 1563 samples per second on Push-T compared with 170 for DINO-WM (Yin et al., 2026). A batch of one hundred perturbed plans therefore takes only seconds, which makes it practical to query many variants of a plan inside the decision loop of the agent.

Feature maps are hard for an agent to interpret, and decoding them into pixels requires a heavy generative decoder whose reconstructions distort fine geometry. We read out keypoints directly from the latent space with a correspondence model C in the style of COTR (Jiang et al., 2021), trained on DINOv2 feature maps of images from the training trajectories. Given the current feature map z_0, a query location k in the current observation and a predicted feature map $\hat{z}$, the model returns the location $C(z_0, k, \hat{z})$ that corresponds to k in the predicted state. We track four keypoints $k^1, \ldots, k^4$ on the block, so the predicted outcome of a plan becomes four image coordinates $\hat{k}^j(\boldsymbol{a}) = C(z_0, k^j, \hat{z}_5(\boldsymbol{a}))$ that can be drawn on the current observation.

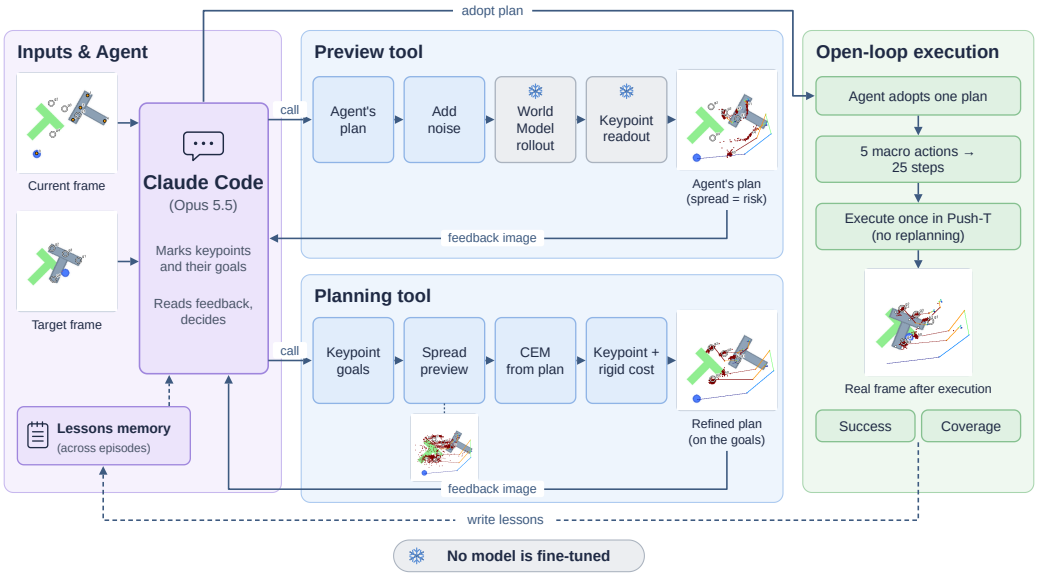

Figure 3: Overview of the system. Claude Code receives the current observation and the goal image and may call two tools built on DDP-WM before it commits one open-loop plan of five macro actions, which are linearly interpolated into 25 low-level steps. The preview tool perturbs the agent's plan, rolls out one hundred variants in the DINOv2 latent space, reads out four keypoints on the block with a correspondence model and draws their clouds on the observation. The planning tool widens the perturbation around the agent's plan, runs a small CEM search toward keypoint targets set by the agent under a rigid-body constraint, and returns the result as clouds in the same format. The agent annotates the four keypoints and their goals on the observations, reads the feedback images returned by the tools, and carries the lessons of earlier episodes into the next one. Snowflakes mark frozen models. Neither the agent nor the world model is fine-tuned, and every decision on what to execute remains with the agent.

3.2 Preview Tool

The preview tool perturbs a plan $\boldsymbol{a}$ proposed by the agent into one hundred variants $\boldsymbol{a}^{(i)} = \boldsymbol{a} + \sigma\epsilon^{(i)}$ with independent zero-mean noise $\epsilon^{(i)}$ and $\sigma = 0.01$, rolls them out in one batch and renders the predicted keypoints $\{\hat{k}^j(\boldsymbol{a}^{(i)})\}_{i,j}$ as four colored point clouds over o_0, together with the prediction for the unperturbed plan. At this noise level the clouds describe how the outcome responds to small variations of the same plan.

The shape of a cloud carries information that a single prediction lacks. A tight cloud indicates that the outcome is insensitive to small variations of the plan, an elongated cloud shows the direction in which the outcome is sensitive, and a cloud that splits into separate clusters reveals a bifurcation, typically a contact that the pusher may or may not make. When the pusher grazes an edge of the block, some perturbed plans drag the block along and others leave it in place, and the corresponding keypoint cloud becomes bimodal. An open-loop plan that depends on such a contact fails whenever execution falls on the wrong side, and the cloud makes this risk visible before any action is taken.

3.3 Planning Tool

The planning tool gives the agent control over a local search. The agent marks target positions $g^1, \ldots, g^4$ for the four keypoints, which specify where the block should end up, and the tool widens the perturbation to $\sigma = 0.02$ and first renders the resulting cloud, so that a cloud covering the targets tells the agent that a plan within this neighborhood can reach the goal. The tool then runs CEM with

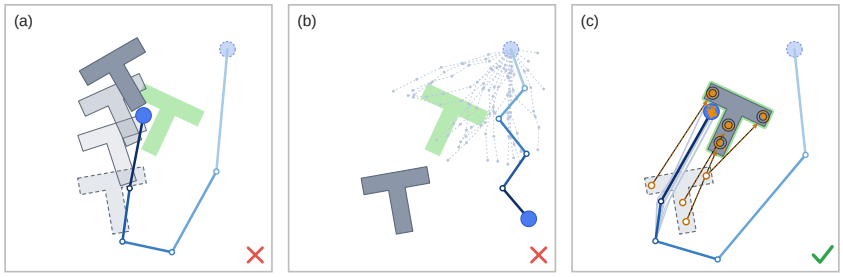

Figure 4: Schematic of how the tools turn a plan into a decision on Push-T. The blue disk is the pusher and the gray T is the block, with the green T marking the goal pose. (a) The agent's plan, drawn as a path whose color darkens with the macro action index, moves the block through the faded poses but ends away from the goal. (b) The preview tool perturbs the plan into many variants, drawn as light gray paths that fan out from the start position of the pusher, and none of the predicted outcomes is a reliable route to the goal. (c) With keypoint targets (double circles) set by the agent, the planning tool searches around the plan until the cloud of predicted keypoints (orange dots) covers the targets, and the resulting plan places the block on the goal.

its sampling distribution initialized at mean $\boldsymbol{a}$ and scale σ and minimizes the keypoint cost

$$J(\boldsymbol{a}) = \sum_{j=1}^{4} \left\| \hat{k}^j(\boldsymbol{a}) - g^j \right\|_2. \tag{1}$$

Because the world model occasionally predicts a deformed or blurred block, and such predictions can reach a spuriously low cost, candidates are kept only if they preserve the rigid shape of the block,

$$\max_{j<l} \left| \left\| \hat{k}^j(\boldsymbol{a}) - \hat{k}^l(\boldsymbol{a}) \right\|_2 - \left\| k^j - k^l \right\|_2 \right| \leq \delta, \tag{2}$$

where δ is a fixed tolerance. The tool returns the best plan together with its preview cloud, and the agent decides whether to adopt, adjust or discard it. Figure 4 illustrates this progression from a single plan through its perturbed variants to keypoint targets.

Keypoint targets have two further advantages over the goal-image cost used by DINO-WM. They constrain only the task-relevant state, so the pusher is free to end anywhere, whereas a goal-image cost penalizes it for leaving the position it happens to occupy in the goal frame. They also do not require a goal image at all. Our protocol provides o_g only because a goal image is the most natural way to specify a Push-T task, and the agent derives the keypoint targets from it.

3.4 The Agent

We use Claude Code with Claude Opus 5.5 as the agent. It interacts with the tools and the environment by reading rendered images and writing plans as JSON files, may call the tools any number of times within an episode, and ends the episode by submitting five macro actions. During the calibration episode it explores the effect of macro actions and the meaning of the tool outputs, and because it plays the 21 evaluation episodes in one session, experience from earlier episodes remains available in its context. Each tool call returns within a few seconds on eight RTX 2080 GPUs, so the running time of an episode is dominated by the deliberation of the agent, which is longest for the bare agent. The bare-agent baseline runs the same agent under the same interface without the two tools.

3.5 Why Distributions Counter Hallucination

Let $c(\boldsymbol{a})$ denote the true cost of a plan and $\hat{c}(\boldsymbol{a}) = c(\boldsymbol{a}) + e(\boldsymbol{a})$ the cost predicted through the world model, with zero-mean error e. A planner that evaluates N candidates and keeps $\boldsymbol{a}^\star = \arg\min_i \hat{c}(\boldsymbol{a}^{(i)})$ obtains

$$\mathbb{E}\big[\hat{c}(\boldsymbol{a}^\star)\big] \leq \mathbb{E}\big[c(\boldsymbol{a}^\star)\big] \tag{3}$$

Table 1: Results on the 21 episodes of our Push-T protocol. An episode succeeds when the final block pose lies within 17.5 pixels and 20 degrees of the goal pose.

Configuration	Decision maker	Successes (of 21)
MPC with DDP-WM	CEM	2
Bare agent	Claude Code	7
Agent with preview tool	Claude Code	14
Agent with preview and planning tools	Claude Code	16

whenever the errors of the candidates are independent of their true costs, and the gap grows with N and with the variance of e, because the minimum preferentially selects candidates whose error is negative (Smith & Winkler, 2006). The selected plan therefore looks better in imagination than it performs in execution. We call this systematic optimism hallucination and distinguish it from ordinary prediction error, which is unbiased. CEM compounds the effect because every iteration refits its sampling distribution to the best-scoring candidates, and an agent that calls the preview tool repeatedly and keeps the most promising plan performs the same kind of selection.

Hallucinated optima tend to be narrow spikes of the predicted cost landscape, where the error happens to be strongly negative over a small region of plan space. Perturbing a plan that sits on such a spike moves most samples off it, so their predicted keypoints disagree and the cloud scatters, while a plan inside a broad basin of genuinely low cost produces a tight cloud. A single rollout of the unperturbed plan predicts success in both cases and cannot tell them apart. The preview cloud thus acts as a local consistency check that the agent reads directly, and the rigid-body test in Equation 2 removes a second source of optimism, predictions that meet the targets only because the block has been deformed.

4 Experiments

4.1 Setup

We compare four configurations on the 21 episodes of our protocol. The Push-T test split released with DINO-WM contains exactly 21 trajectories, so these episodes cover the complete test set, and every configuration is evaluated on all of them without selecting a subset. MPC plans all five macro actions with DDP-WM, CEM and the goal-image cost of DINO-WM (Zhou et al., 2025), which is the distance between the predicted and the goal feature maps, and executes them open loop. The bare agent is Claude Code with Claude Opus 5.5 without world-model tools. The preview configuration gives the agent the preview tool, and the planning configuration gives it both tools. For the two tool configurations we also run an ablation in which every tool output shows only the prediction for the unperturbed plan, so that the agent receives the same world model without its distribution. DDP-WM is trained on the Push-T dataset as in Yin et al. (2026) and kept fixed throughout.

4.2 Main Results

Table 1 shows that performance increases with every capability added to the agent. MPC alone succeeds in 2 episodes. Far goals leave most sampled action sequences without any contact with the block, which gives CEM no informative cost to follow, and the plans it selects are prone to hallucination. The bare agent succeeds in 7 episodes, which shows that its priors about pushing already yield a good coarse plan. The preview tool raises the number of successes to 14, and the planning tool reaches 16. The agent thus supplies a plan near a good solution and the judgement of when to trust a prediction, and the world model supplies fast physical foresight and local precision.

4.3 Physical Prediction by the Agent and by the World Model

The premise of our approach is that a frontier agent predicts the physical consequences of its actions less precisely than a world model does, and we test this directly. We show Claude Opus 5.5 an observation with the four keypoints of the block marked on the image together with their coordinates,

Table 2: Successful episodes (of 21) when the tools return point clouds from one hundred perturbed rollouts or only the prediction for the unperturbed plan. The bare agent without tools solves 7 episodes.

Configuration	Point clouds	Single prediction
Agent with preview tool	14	12
Agent with preview and planning tools	16	13

give it ten in-context examples of the same form, and ask it to predict where the keypoints will be after a given open-loop plan of five macro actions. On the test set at a resolution of 224 pixels, the mean error of the agent is 3.12 pixels over the five steps and 5.7 pixels at the fifth step. On the same plans, DDP-WM reaches 1.9 pixels at the fifth step and DINO-WM 3.1 pixels, even though their keypoints come from the correspondence readout and therefore include its own small error. The agent thus predicts the motion of the block reasonably well, yet clearly less precisely than the world models, and the gap widens when the block is pushed far, where the errors of the agent grow to more than 20 pixels. Physical prediction is therefore the part of the task that the agent is best advised to delegate to the world model.

4.4 Distributions Versus Single Predictions

Replacing the clouds by the prediction for the unperturbed plan reduces the preview configuration from 14 to 12 successes and the planning configuration from 16 to 13 (Table 2). With single predictions, both tool configurations end below the bare agent in some episodes, which is the signature of hallucination. The agent receives a precise-looking prediction that places the block on its target, trusts it, and executes a plan that the world model has scored optimistically. The planning configuration suffers in the same way, since CEM returns exactly the plans that its own selection has made optimistic, and without the cloud the agent cannot tell a robust optimum from a spike. With clouds, the agent declines plans whose keypoints scatter even when the unperturbed prediction meets every target, and it keeps adjusting the plan until the clouds are tight (Figure 1).

4.5 How the Agent Uses the Tools

The transcripts show that the agent adopts the tools within the calibration episode and reads their outputs in probabilistic terms. It repeatedly describes the clouds by their spread, clustering and bimodality, refers to the confidence of a prediction, and reasons about the distribution of outcomes when choosing between plans. It treats a tight cloud on the targets as evidence that a plan can be executed safely, and when a cloud scatters or splits it revises the plan and verifies the revision with another preview, even when the unperturbed prediction already satisfies the targets. In the planning configuration it uses the widened cloud to check whether its targets are reachable from the current plan before starting the search, and it inspects the returned plan with the same criteria before adopting it. Appendix A follows one episode of the planning configuration through these steps.

5 Related Work

Language agents and robot manipulation. Language agents interleave reasoning with actions and tool calls (Yao et al., 2023; Schick et al., 2023), and the interface through which an agent observes and edits its environment largely determines what it can accomplish (Yang et al., 2024a). In robotics, language models have served as planners grounded in affordances or environment feedback (Ahn et al., 2022; Huang et al., 2022; Driess et al., 2023), as writers of policy, skill and reward code refined from execution feedback (Liang et al., 2023; Wang et al., 2024; Ma et al., 2024), and as sources of value maps or keypoint constraints for motion planners (Huang et al., 2023; 2024; Pan et al., 2025). Visual prompting methods draw candidate points or actions on the image and let a VLM choose among them (Fang et al., 2024; Nasiriany et al., 2024), showing that VLMs reason well about spatial quantities once these are rendered in their input modality. Another line fine-tunes such models into vision-language-action policies (Brohan et al., 2023; Kim et al., 2024; Gemini Robotics Team, 2025). Benchmarks nevertheless find that VLMs grasp physical properties

and dynamics poorly (Bear et al., 2021; Gao et al., 2024; Chow et al., 2025), and several systems therefore pair a VLM with a predictive model. FOREWARN aligns a fine-tuned VLM with latent predictions to select among the plans of a generative policy (Wu et al., 2025), VLMPC scores video predictions of VLM-sampled actions inside an MPC loop (Zhao et al., 2024), and KUDA turns keypoint targets from a VLM into costs for a learned dynamics model (Liu et al., 2025). In these systems the language model fills a fixed slot of a designed pipeline, whereas the agent we study is handed a world model as a pair of tools, decides when and how to call them, reads their predictions as images of keypoint distributions, and requires no fine-tuning of either model.

World models and planning. Planning through learned predictions of visual outcomes dates back to visual foresight, which combined action-conditioned video prediction with CEM-based MPC to push objects (Finn & Levine, 2017; Ebert et al., 2018), and latent dynamics models have since become a standard basis for planning and control (Ha & Schmidhuber, 2018; Hafner et al., 2019; 2020; Hansen et al., 2022; 2024). Generative world models now simulate interactive environments at scale (Yang et al., 2024b; Bruce et al., 2024) and support planning for navigation (Bar et al., 2025). The joint-embedding predictive architectures advocated by LeCun predict in a learned representation space that discards unpredictable detail (LeCun, 2022; Assran et al., 2023). Within this family, V-JEPA 2 plans toward image goals on real robot arms (Assran et al., 2025), DINO-WM plans with CEM on frozen DINOv2 features (Zhou et al., 2025; Oquab et al., 2024), C-JEPA extends masked latent prediction to object-centric representations (Nam et al., 2026), and DDP-WM separates sparse interaction-driven dynamics from a low-rank background update, which yields a roughly ninefold speedup over DINO-WM on Push-T and a smoother optimization landscape for planning (Yin et al., 2026). We build on DDP-WM because a hundred perturbed rollouts and a short CEM search both finish within seconds, well inside the time the agent spends deliberating. Push-T was adapted by Chi et al. (2023) from the pushing tasks of Florence et al. (2021).

Keypoints and correspondence. Keypoints and dense correspondences give a compact and interpretable state for manipulation. Dense Object Nets learn pixel descriptors that remain consistent across object poses (Florence et al., 2018), and keypoints tracked through such descriptors serve as the state of learned dynamics models for model-based control (Manuelli et al., 2020). Correspondence transformers match arbitrary query locations across images (Jiang et al., 2021). Our readout applies such matching between observed and predicted DINOv2 feature maps, which turns latent predictions into keypoint positions without decoding pixels.

Optimism under selection. Selecting the best of several noisy estimates yields an estimate of the winner that is biased toward optimism, a phenomenon studied as the optimizer's curse in decision analysis (Smith & Winkler, 2006) and as overestimation bias in value-based reinforcement learning (van Hasselt, 2010). In model-based reinforcement learning, planners exploit regions where the learned model is optimistic (Kurutach et al., 2018; Janner et al., 2019), and ensembles or penalties on model uncertainty are common remedies (Chua et al., 2018; Yu et al., 2020). Our preview tool offers a lightweight counterpart without training additional models, since the spread of keypoints under small action perturbations shows whether a predicted success rests on a broad basin or on a narrow spike.

6 Conclusion

Frontier agents and latent world models have developed along separate lines, and our results show that each supplies what the other lacks. On our Push-T protocol, MPC with a strong latent world model solves 2 of 21 episodes and Claude Code alone solves 7, while the agent equipped with two tools built on DDP-WM solves 16. The agent learns to use these tools from a single exploratory episode, and it benefits from them because they present the predictions of the world model as distributions it can reason about, which protects it from the hallucinated plans that selection on a learned model produces. Our protocol isolates contact-rich pushing under a strict action budget, which lets us measure how much physical understanding a complete system brings to a single committed plan. Because the agent expresses goals as keypoint constraints, the approach does not depend on goal images and is compatible with goals stated in language or in any other form that an agent can translate into such constraints. Extending the tool interface to other agents such as Codex with GPT-6 Astra, to closed-loop execution and to real robots is a natural next step.

AI USE STATEMENT

This work studies a generative AI system, Claude Code with Claude Opus 5.5, as the subject of its experiments, and all agent results reported in this paper were produced by this system under the protocol of Section 2. We also used generative AI tools to assist with drafting and editing the text of the manuscript. The authors reviewed all AI-assisted work and take responsibility for the final content of this paper, including text, claims and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All experiments are conducted in simulation and involve no human subjects and no personal data.

REPRODUCIBILITY STATEMENT

Section 2 specifies the environment, the construction of the 21 episodes, the calibration episode and the success criterion. Section 3 specifies the world model, the keypoint readout, the noise levels, cost and rigid-body test of the tools, and the interface of the agent, and Section 4 lists all compared configurations. The world model is the publicly available DDP-WM, and the simulator and test trajectories are those released with DINO-WM. We will release the code for the evaluation protocol and for both tools, including the keypoint readout, upon publication.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning (CoRL)*, 2022.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Daniel M. Bear, Elias Wang, Damian Mrowca, Felix J. Binder, Hsiao-Yu Fish Tung, R. T. Pramod, et al. Physion: Evaluating physical prediction from vision in humans and machines. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, et al. Genie: Generative interactive environments. In *International Conference on Machine Learning (ICML)*, 2024.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

Wei Chow, Jiageng Mao, Boyi Li, Daniel Seita, Vitor Guizilini, and Yue Wang. PhysBench: Benchmarking and enhancing vision-language models for physical world understanding. In *International Conference on Learning Representations (ICLR)*, 2025.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. PaLM-E: An embodied multimodal language model. In *International Conference on Machine Learning (ICML)*, 2023.

Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

Kuan Fang, Fangchen Liu, Pieter Abbeel, and Sergey Levine. MOKA: Open-world robotic manipulation through mark-based visual prompting. In *Robotics: Science and Systems (RSS)*, 2024.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

Peter R. Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. In *Conference on Robot Learning (CoRL)*, 2018.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A. Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning (CoRL)*, 2021.

Jensen Gao, Bidipta Sarkar, Fei Xia, Ted Xiao, Jiajun Wu, Brian Ichter, Anirudha Majumdar, and Dorsa Sadigh. Physically grounded vision-language models for robotic manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

Gemini Robotics Team. Gemini Robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning (ICML)*, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*, 2020.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning (ICML)*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, et al. Inner monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning (CoRL)*, 2022.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D value maps for robotic manipulation with language models. In *Conference on Robot Learning (CoRL)*, 2023.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. ReKep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *Conference on Robot Learning (CoRL)*, 2024.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Wei Jiang, Eduard Trulls, Jan Hosang, Andrea Tagliasacchi, and Kwang Moo Yi. COTR: Correspondence transformer for matching across images. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*, 2024.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations (ICLR)*, 2018.

Yann LeCun. A path towards autonomous machine intelligence. *OpenReview preprint*, 2022.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

Zixian Liu, Mingtong Zhang, and Yunzhu Li. KUDA: Keypoints to unify dynamics learning and visual prompting for open-vocabulary robotic manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, et al. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Lucas Manuelli, Yunzhu Li, Pete Florence, and Russ Tedrake. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2020.

Christopher Mims. This AI pioneer thinks AI is dumber than a cat. *The Wall Street Journal*, October 2024.

Heejeong Nam, Quentin Le Lidec, Lucas Maes, Yann LeCun, and Randall Balestriero. Causal-JEPA: Learning world models through object-level latent masking. In *International Conference on Machine Learning (ICML)*, 2026.

Soroush Nasiriany, Fei Xia, Wenhao Yu, Ted Xiao, Jacky Liang, Ishita Dasgupta, et al. PIVOT: Iterative visual prompting elicits actionable knowledge for VLMs. In *International Conference on Machine Learning (ICML)*, 2024.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Mingjie Pan, Jiyao Zhang, Tianshu Wu, Yinghao Zhao, Wenlong Gao, and Hao Dong. OmniManip: Towards general robotic manipulation via object-centric interaction primitives as spatial constraints. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Reuven Y. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, et al. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

James E. Smith and Robert L. Winkler. The optimizer's curse: Skepticism and postdecision surprise in decision analysis. *Management Science*, 52(3):311–322, 2006.

Hado van Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

Yilin Wu, Thomas Tian, Gokul Swamy, and Andrea Bajcsy. From foresight to forethought: VLM-in-the-loop policy steering via latent alignment. In *Robotics: Science and Systems (RSS)*, 2025.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a.

Sherry Yang, Yilun Du, Seyed Kamyar Seyed Ghasemipour, Jonathan Tompson, Leslie Pack Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. In *International Conference on Learning Representations (ICLR)*, 2024b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Shicheng Yin, Kaixuan Yin, Weixing Chen, Yang Liu, Guanbin Li, and Liang Lin. DDP-WM: Disentangled dynamics prediction for efficient world models. In *International Conference on Machine Learning (ICML)*, 2026.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Wentao Zhao, Jiaming Chen, Ziyu Meng, Donghui Mao, Ran Song, and Wei Zhang. VLMPC: Vision-language model predictive control for robotic manipulation. In *Robotics: Science and Systems (RSS)*, 2024.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *International Conference on Machine Learning (ICML)*, 2025.

A Case Studies

We illustrate how the agent uses the planning tool with a representative episode. All panels are the feedback images that the tools and the environment returned to the agent during the episode, cropped to the region around the block.

From near miss to success. Figure 5 follows episode 2 in the planning configuration. After five previews, the best plan of the agent brings three keypoints of the block into their targets, while the cloud of the fourth misses its target. The agent then requests the widened cloud, confirms that it covers all targets, and runs CEM from its plan. The cloud of the returned plan is tightly concentrated on all four targets. The agent runs five further previews before adopting a plan, and the block ends within 17.5 pixels and 20 degrees of the goal pose with a final coverage of 95%.

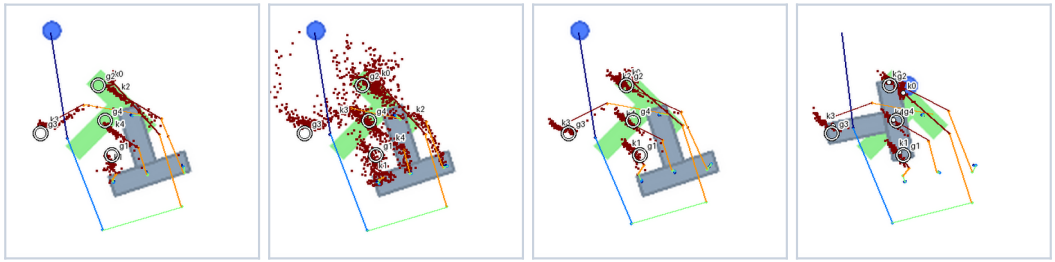

Figure 5: The planning tool turns a near miss into a success. Hollow circles mark the keypoint targets set by the agent, dark red clouds the keypoints predicted after the fifth macro action, and colored polylines the unperturbed plan by macro action. (a) Preview of the best plan of the agent before planning, in which the cloud of one keypoint misses its target. (b) The same plan under the widened perturbation, whose cloud covers all targets. (c) Preview of the plan returned by CEM, tightly concentrated on the targets. (d) Frame after executing the plan adopted after five further previews, with white dots marking the keypoints found in the frame. The block ends within 17.5 pixels and 20 degrees of the goal pose.