2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

RoboSQ: Semantic Queries for Task-Aligned Robot Training Data

†Kaiyuan Chen[1*], Shuangyu Xie[1*], Kush Hari[1], Andrew Goldberg[1], Kavish Kondap[1], Ken Goldberg[1,3]

***Abstract*— Training robot policies often requires extracting appropriate subsets of data from large and noisy datasets. For example, one might want to extract only robot demonstrations with accurate captions or only those related to cooking. We present RoboSQ, a robot data management system that allows semantic queries. RoboSQ samples temporally distributed frames and overlays projected sensor information from robot trajectories and constructs structured Visual Question Answering (VQA) prompts for Vision-Language Models (VLMs). RoboSQ efficiently handles queries by pipelining data loading, frame extraction, and VLM inference. We evaluate RoboSQ on the DROID dataset with three semantic queries: 1) failure detection, 2) calibration error detection and 3) visual complexity scoring. It filters out the failure trajectories with 78% accuracy and 86% F1 score, and identifies the trajectories with incorrect extrinsic calibration between camera frame and end effector frame at 86% accuracy and 88% F1 score. We evaluate RoboSQ by training a pick-and-place Action Chunking Transformer policy with a UR5 robot arm with mixed quality demonstration data. Data extracted by RoboSQ is closely aligned with the expert-curated data. A policy trained on RoboSQ-selected data achieves 13 successes out of 15 trials, compared to only 1 out of 15 when trained on the full mixed dataset. Code, video and supplementary information can be found on website `https://berkeleyautomation.github.io/robosq/`**

I. Introduction

Vision-Language-Action models [1–3] require large-scale human teleoperated demonstration trajectories, such as Open X-Embodiment [4] and Distributed Robot Interaction Dataset (DROID) [5] curated from research institutions worldwide. At tera or even penta-byte scale, which is sometimes characterized as Big Data [6], manually inspecting and filtering robot demonstrations becomes prohibitively expensive. Emerging robot data management systems, such as RoboDM [7] and LeRobot [8], efficiently store the data and load them for policy training. However, these systems rely primarily on predefined metadata and do not support semantic querying, such as "does the robot perform a successful grasp?", "is the extrinsic calibration correct?", or "does the task occur in low-light conditions?". This limits their ability to handle semi-structured, heterogeneous robot datasets where critical information is often embedded in raw visual or sensor data rather than explicit labels. To address this challenge, we present RoboSQ, a robot data management system that supports semantic queries – finding subsets of data that satisfy semantic conditions described in natural language.

[1]Department of Electrical Engineering and Computer Science
[2]Department of Industrial Engineering and Operations Research
[1,2]University of California, Berkeley, CA, USA
∗ Equal Contribution.
†For correspondence and questions: kych@berkeley.edu

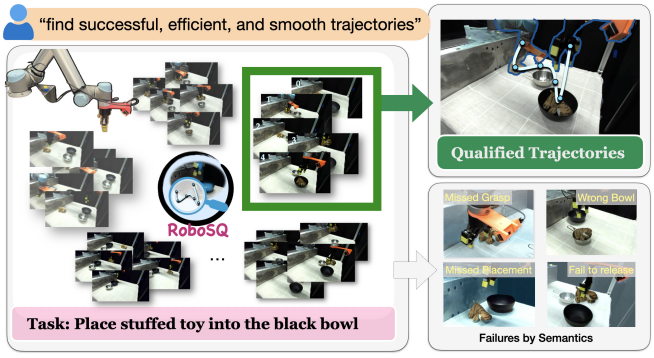


Fig. 1: **RoboSQ**: A semantic query system with efficient frame selection and VLM to extract high-quality, task-specific data for effective robot learning.

Recent Vision-Language Models (VLMs) [9–15] are increasingly capable of performing spatial and semantic analysis of images, and semantic querying of image datasets. VLMs are trained from Internet-scale data, which enables generalizable understanding of complex and diverse contexts in robotics data, as tested and evaluated in recent efforts [16, 17], and can be further improved with more robotics data [18]. Visual Question Answering (VQA) is an interface to VLMs that structures the language and image as input, and allow users to pose complex visual and semantic queries. In robotics, this can be applied to data management tasks such as identifying failure cases, summarizing behaviors, or checking spatial relations without requiring manual annotations.

In this paper, we present RoboSQ, a semantic query system for robot data based on VLM using VQA. RoboSQ structures multi-modal robot trajectory data into structured VQAs by sensor projection, image concatenation, and streaming. To handle the heterogeneous semantic query pipeline, RoboSQ organize these modules and VQAs into a combination data transformation primitives, such as sort, filter, as the output of the semantic query response. The pipeline of RoboSQ efficiently parallelizes data frame extraction and VLM inference.

We evaluate the semantic query capability of RoboSQ on DROID [5] with three unique case studies: (1) Given only one camera stream, RoboSQ filters out the trajectories that fail to complete the task. RoboSQ agrees with the metadata annotated by original human teleoperators at 0.86 F1 score; (2) RoboSQ could identify the many demonstrations with incorrect extrinsic calibration between camera frame and end effector frame; (3) RoboSQ ranks the trajectory semantic quality with a metric based on visual and task complexity.

We integrate RoboSQ into a robot learning task where a UR5 arm is to pick up a stuffed animal and place

it into a black bowl. The dataset includes trajectories of varying qualities, containing both pick-and-place failures and inefficient motions. The data extracted by RoboSQ closely matches the expert-curated set. Training a policy on RoboSQ-selected data achieved 13 successes out of 15 trials, in contrast to only 1 out of 15 when trained on the full mixed dataset.

The paper makes the following contributions: (1) RoboSQ, a semantic query-enabled robot data management system; (2) an efficient parallel processing pipeline to structure Visual Question Answering for Vision Language Models; (3) empirical evaluation data of RoboSQ on large scale open datasets and a real robot learning task.

II. Related Work

Big Robot Data The robot learning community is actively building a number of open-source robot learning datasets [19–21]. Recent work, such as Octo [2] and Open-VLA [3], is trained on large datasets such as RT-1 [19], RT-2 [1], Open-X-Embodiment [4], Distributed Robot Interaction Dataset (DROID) [5]. Also, data augmentation effort using human videos [22] and robot mask inpainting [23] scales data quickly. Initial results suggest training with large and diverse robotics datasets can enhance robot capabilities and generalization in handling multiple settings in diverse environments. In this work, we present an efficient data query pipeline for managing large and diverse robot datasets.

Robot Data Management Traditional robot data management systems, such as LeRobot and RoboDM [7], only use given metadata (e.g., robot type, task type, success/failure). While effective for basic querying on existing manually labeled metadata, these systems lack the ability to perform semantic queries involving vision and language. This limits their utility for filtering or inspecting robot trajectories at scale.

Multi-Modal Information Retrieval. Table I shows a comparison of existing approaches to curate and retrieve multi-modal data. Earlier machine learning approaches addressed retrieval with learned similarity metrics, but these were generally unimodal and not robust to visual diversity or temporal structure. More recently, vector databases (e.g., FAISS [24], Milvus [25]) have enabled embedding-based retrieval by storing dense representations of text or images for fast nearest-neighbor queries. Retrieval-Augmented Generation (RAG) [26] builds on vector DBs by pairing retrieval with language models, enabling generation conditioned on retrieved context. However, early RAG systems primarily support text-based or image-based retrieval and have limited capabilities for robot perception tasks such as captioning.

VLM-based Retrieval and RoboSQ. Recent frameworks like LlamaIndex [27] integrate VLMs with vector databases to support natural language querying over multi-modal data. These systems expand RAG's capabilities into the visual domain, enabling semantic access to video, image, and sensor streams. RoboSQ builds on this trend by introducing a scalable VLM-based query engine for robot trajectories.

Robot Semantic Query Type	Index	Vector	RoboSQ
Task Success/Failure	✗	✓	✓
Find Objects	✗	✓	✓
Task Captioning	✗	✗	✓
Extrinsic Calibration Detection	✗	✗	✓
Trajectory Quality Scoring	✗	✗	✓
Object Occlusion Detection	✗	✗	✓
Visual Clarity Assessment	✗	✗	✓

TABLE I: **Supported Semantic Query Types with Different Database Designs** Conventional Index-based databases use metadata and indices that can only handle semantic queries if explicitly labeled. Emerging Vector databases compares the similarity of foundational model vector embeddings between the input query and the image or text data in the database. Existing Vector databases can handle course-grained queries efficiently, but it is challenging to handle non-visual and non-textual embodiment data. RoboSQ processes multi-modal robot data and use semantic capability of VLM to achieve fine-grained robot data processing.

III. Problem Formulation

A robot demonstration trajectory $\mathcal{T}$ as a temporally ordered sequence of data frames x_t, where $t \in \{1, 2, \ldots, T\}$ indexes discrete time steps and T denotes the trajectory length. Each trajectory is paired with a natural language task description l, resulting in the representation:

$$\mathcal{T} = \{x_{1:T},\ l\}$$

A trajectory database is defined as a collection of such demonstrations:

$$\mathcal{D} = \{\mathcal{T}_i\}_{i=1}^{N}$$

We study the following problems:

Definition III.1 (**Semantic Data Query**). *Given a trajectory database $\mathcal{D}$ and a user-specified semantic query condition Q expressed in natural language, the goal is to retrieve a subset of* qualified goal trajectories $\{\mathcal{T}_g\} \subseteq \mathcal{D}$ *such that each trajectory satisfies the semantic condition:*

$$\{\mathcal{T}_g\} = \{\mathcal{T} \in \mathcal{D} \mid f(\mathcal{T}, Q) > \tau\}$$

Each semantic query Q can be composed and transformed to K textual conditions q_k with optional importance weights $\alpha_k \in [0, 1], \sum_k \alpha_k = 1 : Q = \{(q_1, \alpha_1), \ldots, (q_K, \alpha_K)\}$. A trajectory $\mathcal{T}$ is considered to satisfy Q if the predicted relevance score $s = f(\mathcal{T}, Q)$ exceeds a user-defined threshold τ. Each condition q_k corresponds to a high-level semantic property of interest (e.g. "contains a successful grasp", "extrinsic calibration is correct", or "takes place in low-light conditions").

Definition III.1.1 (**Top-K Qualification**). *For ranked retrieval, we return the top-k most relevant trajectories based on the scoring function:* $\{\mathcal{T}_g\} = \textit{Top-}k\,(\mathcal{T} \in \mathcal{D},\ s)$.

IV. Method

A. Overview

We present RoboSQ, a robot data semantic query framework (as defined in Definition III.1) using VLM and a pipelined execution engine. As shown in Fig. 2, RoboSQ processes user queries (e.g., "give me high-quality data considering clarity, occlusion, etc.") by constructing sequential visual prompts, and using a VLM (e.g., Gemini, GPT-4o, Qwen) to produce scalar ratings over task-relevant axes (e.g.,

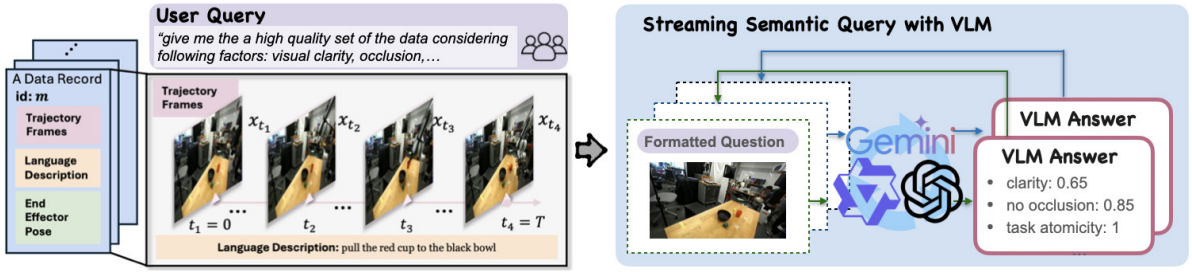


Fig. 2: RoboSQ pipeline. The user issues a natural language prompt asking whether a trajectory is successful. RoboSQ uses Vision-Language Models (VLMs) to analyze sampled frames from the trajectory and generate a structured answer. This VQA interface enables expressive multi-modal filtering based on high-level task semantics.

clarity, atomicity, and goal visibility). To scale across large datasets, RoboSQ pipelines disk-bound data loading, CPU-bound frame decoding, and GPU-bound VLM inference, enabling high-throughput, resource-efficient search through millions of robot trajectories.

B. Robot Trajectory Visual Question Answering

Building on the Robo2VLM framework [18], which transforms frames in trajectories into VQA, we introduce how a VLM can be used for semantic queries that are fully aware of the complete trajectory context.

1) Heterogeneous Sensor Projection: Some semantic queries require not only visual input but also additional sensor modalities such as robot joint poses. Heterogeneous sensor projection provides the link between low-level kinematic metadata and the perceptual image space where semantic queries are executed. Formally, any point of interest on the robot expressed in world coordinates ${}^{W}\mathbf{p} \in \mathbb{R}^3$ can be projected into the image $(u, v) \in I$ through a projection function

$$(u, v) = g({}^{W}\mathbf{p}).$$

As an illustrative example, we consider the query "is the extrinsic calibration correct?". To answer this, we project the robot end-effector pose from robot trajectory onto the camera view and render it as an overlay. When the calibration is correct, the projection coincides with the visible end-effector geometry in the image; in contrast, miscalibration produces a noticeable offset.

Each demonstration record includes the robot end-effector pose in the world coordinate system, computed via forward kinematics. Let ${}^{W}\mathbf{T}_E \in SE(3)$ denote the homogeneous transformation matrix representing the end-effector pose in the world frame $\{W\}$. The metadata also contains camera parameters: the intrinsic matrix $\mathbf{K}$ and the extrinsic transformation ${}^{W}\mathbf{T}_C \in SE(3)$, which maps the camera coordinate frame $\{C\}$ into the shared world coordinate frame. To project the 3D end-effector position into the 2D image, we first transform the end-effector point (tool center point) $\mathbf{p}_E = [x_E, y_E, z_E, 1]^\top$ into the camera coordinate system:

$$\mathbf{p}_C = ({}^{W}\mathbf{T}_C)^{-1} \cdot {}^{W}\mathbf{T}_E \cdot \mathbf{p}_E.$$

The pixel coordinates (u, v) are obtained by applying the projection function f to the world point ${}^{W}\mathbf{p}$:

$$(u, v) = g({}^{W}\mathbf{p}) = \pi\big(\mathbf{K} \cdot ({}^{W}\mathbf{T}_C)^{-1} \cdot {}^{W}\mathbf{p}_h\big),$$

where ${}^{W}\mathbf{p}_h = [x_W, y_W, z_W, 1]^\top$ is the homogeneous representation of the world point, and $\pi([x, y, z]^\top) = \left(\frac{x}{z}, \frac{y}{z}\right)^\top$ denotes the perspective division.

2) Temporal information handling: A robot trajectory contains the temporal evolution of the robot, environment, and their interaction. To maintain temporal coherence across long-horizon behaviors, we provide multiple sensor frames (e.g. images) from the robot trajectories in the query process. Providing multi-frames allows RoboSQ to support a broader class of semantic queries that depend on temporal logic, phase transitions, or event ordering, while maintaining interpretability and flexibility in prompt design. The overall architecture supports complex queries such as "Did the robot complete the task without occlusion?", "Was there a successful grasp followed by a lift?", or "Did a failure occur during execution?"

Recent advances in VLM prompting highlight two main paradigms for handling temporal input: (i) concatenated multi-image queries, which treat sampled frames as a single structured input, and (ii) streaming queries, which process sequences incrementally, mirroring human-like temporal reasoning. Inspired by OpenAI's streaming response protocol [28] and recent VLM prompting practices for video reasoning [29], we implement both modes in RoboSQ.

Image Concatenation To ensure the VLM receives sufficient spatial context for semantic reasoning, we uniformly sample n frames across a trajectory and concatenate them into a single tiled image. This approach preserves temporal ordering while allowing the model to observe multiple phases of a task simultaneously. Formally, the sampled trajectory indexes are

$$t_i = \left\lfloor \frac{i \cdot (T-1)}{n-1} \right\rfloor, \quad i = 0, 1, \ldots, n-1, \tag{1}$$

where T is the trajectory length and $\lfloor \cdot \rfloor$ denotes the floor function to ensure integer indexing.

Image Streaming. Streaming queries present the trajectory sequentially, similar to how a human reviews a video by watching it unfold frame by frame. Unlike concatenation, where frames are provided all at once, streaming mode

incrementally feeds frames to the VLM. This design mirrors OpenAI's streaming response protocol [28].

Given a trajectory $\mathcal{T}$ and semantic query Q, frames are sampled at regular intervals according to Eq. (1). Each sampled frame x_{t_i} is appended into the evolving context and streamed to the VLM. The model's interpretation is shaped round by round in the shared context. The final score s emerges from aggregating these context-conditioned representations, providing a temporally grounded assessment of the trajectory as a whole.

This streaming design enables the VLM to detect causal dependencies (e.g., "a grasp before a lift") and task outcomes (e.g., "a failure occurred during execution") as they emerge, without requiring all frames to be processed upfront.

Remark. We note that the selection of image streaming modes depend on the semantic query and the VLM model characteristics. For example, some VLMs based on CLIP [30] typically use fixed input image sizes. These VLMs are not appropriate for image concatination.

C. Semantic Query Pipeline

RoboSQ iterates semantic query on each robot trajectory data. Sometimes, the output of VLM can be verbose and unstructured. For example, Gemini Robotics [31] shows that reasoning step by step before generating the answer encourages the model to "think" about the problem and significantly improves robotics planning task success rate. To perform semantic data query defined in Definition III.1, performing VQA per trajectory, extracting answer from VLM answers and aggregating all answers from the dataset are challenging. RoboSQ builds on the insight that *a semantic query can be partitioned into a number of dataset transformation primitives*.

We define *dataset operators* that bridge heterogeneous robot trajectories to VLM outputs, and to the final semantic query results. This process can be defined with three key dataset operator primitives, `Filter`, `Map`, and `Sort`, as following:

Definition 1 (Dataset Operator Primitives)**.** *Let $\mathcal{D}$ be a dataset whose elements are trajectory records $r \in \mathcal{R}$ (key–value dictionaries). A database action primitive is an operator* op *parameterized by a function λ, acting on $\mathcal{D}$ as follows:*

$$
\begin{aligned}
\text{Filter}: \quad & \lambda_F : \mathcal{R} \to \{\textsc{True}, \textsc{False}\}, \\
& \text{Filter}(\mathcal{D}, \lambda_F) = \{\, r \in \mathcal{D} \mid \lambda_F(r) = \textsc{True} \,\} \\
\text{Map}: \quad & \lambda_M : \mathcal{R} \to \mathcal{R}', \\
& \text{Map}(\mathcal{D}, \lambda_M) = \{\, \lambda_M(r) \mid r \in \mathcal{D} \,\} \\
\text{Sort}: \quad & \lambda_S : \mathcal{R} \to \mathbb{R}, \\
& \text{Sort}(\mathcal{D}, \lambda_S) = [\, r \in \mathcal{D} \,] \textit{ ordered by } \lambda_S(r)
\end{aligned}
$$

where $\mathcal{R}'$ is a prescribed output type: either $\mathcal{R}$ (schema-preserving), a record type extending $\mathcal{R}$ (annotation), or a fixed value space (e.g., $\mathbb{R}^k$ or a label set).

The output type of `Map` $\mathcal{R}'$ specifies the intended use of the transform. If $\mathcal{R}' = \mathcal{R}$, the map is *schema-preserving* that it transforms records without changing their structure (e.g., normalize gripper values). If $\mathcal{R}'$ is a record type that augments $\mathcal{R}$, the map performs *annotation* by appending derived fields (e.g., a semantic relevance score in $[0, 1]$). If $\mathcal{R}'$ is a value space (e.g., $\mathbb{R}^k$ or a finite label set), the map is a *projection* that produces features or labels from each record (e.g., trajectory embeddings in $\mathbb{R}^k$).

Given these operator primitives, RoboSQ query process is a multi-stage dataset transformation:

1) RoboSQ uses a `Map` operation to load the raw dataset from disk or network.
2) RoboSQ uses a `Map` operation to seek to the time frame of interest, decode and extract the frames
3) RoboSQ uses a `Map` operation to run one or multiple VLM inferences on each trajectory, issuing structured prompts (e.g., image concatenation or streaming) and recording the model's full textual response.
4) RoboSQ applies a `Map` operation to extract structured outputs—such as binary labels, scores—from the VLM response.
5) RoboSQ compiles the extracted outputs into dataset operator functions, instantiating λ_F for `Filter`, λ_S for `Sort`, or λ_M for `Map`, depending on the prompt type and expected output format.

For instance, consider a user-issued semantic query such as "Demonstrations that the robot finishes a task." RoboSQ begins by applying a series of `Map` operations: loading raw trajectories from disk, sampling temporally distributed frames, and formatting them into structured prompts (e.g., tiled image grids or streaming sequences) for VLM inference. The VLM may return a verbose response, such as "The robot approaches the cup, tips it over, and pours all contents into the second container. Yes, the task is completed successfully." In this case, a second `Map` operation is applied to extract the terminal answer—"Yes"—using a lightweight pattern-matching rule or a secondary LLM [32] trained to extract structured outputs. This Boolean response is then compiled into a λ_F function representing a `Filter` operator, enabling RoboSQ to retain only the successful demonstrations from the full dataset.

As another example, suppose the user query is "Rate the visual clarity of the trajectory from 1 to 5." In this case, the VLM returns scalar scores such as "3 out of 5" or "I would rate this a 4 due to partial occlusion." The extracted numeric value is used in one of two ways: (i) as λ_S in a `Sort` operator, allowing RoboSQ to rank trajectories by clarity; or (ii) as λ_M in a `Map` operator, enriching each trajectory's metadata with an additional `clarity_score` field.

D. Efficient Pipeline

In RoboSQ, a semantic query is executed as a sequence of dataset transformations, each expressed as a `Map` operation. These transformations are designed to reflect the distinct computational bottlenecks in robot data processing—disk I/O, CPU-bound video decoding, and GPU-bound VLM inference—and are composed into a pipeline that enables parallel execution across heterogeneous hardware.

We illustrate this design with a three-stage example. In the first stage, RoboSQ applies a `Map` operation to load raw robot trajectories from disk. In the second stage, a second `Map` operation uses parallel CPU workers to decode compressed video streams and extract frames of interest. Finally, a third `Map` operation formats the frames into structured prompts and runs them through the VLM, issuing one inference call per trajectory.

Because the `Map` operations are independent between trajectories, the transformations can be parallelized to reduce the resource contention between different stages. In the example, disk-bound loading prefetches trajectories, CPU-bound decoders process previously loaded chunks, and GPU-bound inference saturates hardware through batched VLM calls. This modular design not only improves throughput but also provides elasticity: additional GPUs increase inference capacity without altering I/O or decoding behavior, while faster storage accelerates data loading independently of downstream stages.

V. Evaluation

A. Experimental Setup

We evaluate RoboSQ on the Distributed Robot Interaction Dataset (DROID) [5], a large-scale in-the-wild robot manipulation dataset containing 76,000 human teleoperated demonstrations on a Franka robot with a Robotiq gripper (350 hours) collected across 564 scenes and 86 tasks. Each DROID trajectory contains three synchronized RGB camera streams, camera calibration parameters, depth information, 7-DOF proprioceptive data, end-effector pose, gripper state, and natural language task instructions. The trajectories are manually annotated with success/failure labels and contain multiple langauge annotations per successful episode. We resize all DROID images to 640x320 resolution for VLM inference efficiency.

Unless specified otherwise, we use Qwen2.5-VL-32B-Instruct as our VLM backbone, which has demonstrated strong performance on visual understanding benchmarks [18]. Our evaluation runs on a cluster with a 256-core CPU and 2.0TB memory. The VLM inference pipeline distributes frames across 8× NVIDIA A100 GPUs (80GB VRAM each) using SGLang with tensor parallelism, enabling efficient processing of thousands of trajectories per hour.

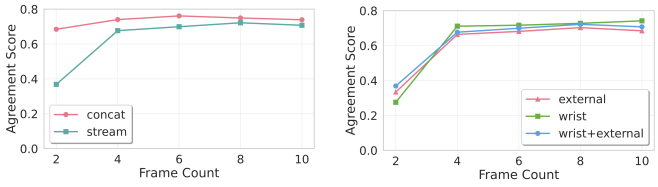
B. Task Failure Detection

Filtering out failed demonstrations is essential for training robot policies that could negatively impact model training. Commonly, robotics researchers detect task failures by visually monitoring the robot demonstration and filtering out the failed robot trajectories. Our first evaluation examines RoboSQ's capability in this task using purely visual data of robot trajectories. Given a video stream from a single camera, we provide the semantic query "Did the robot successfully complete the task? Answer yes or no.".

Detection Agreement Score. We note that the success or failure conditions in DROID data collection are not explicitly stated. For example, in a pick-and-place demonstration, some trajectories succeed by only removing one cube from a container, while other trajectories need to remove all cubes. Failures are purely determined by the human teleoperator. We evaluate RoboSQ by comparing how it agrees with the labels provided by the human teleoperator, as an indicator of how RoboSQ agrees with human in semantic qualification. For each experiment, we randomly sample 1400 trajectories (2% of the DROID dataset) with the same success-failure proportion as the raw DROID dataset. We repeat each experiment 3 times.

System	Model	Accuracy	Precision	Recall	F1 Score
Random	Known Dist.	0.58	0.70	0.70	0.70
VectorDB	OpenCLIP	0.63	0.82	0.70	0.76
VectorDB	SigLip2	0.61	0.83	0.68	0.75
Random	Coin Flip	0.50	0.50	0.50	0.50
RoboSQ	GPT-4o	0.68	0.68	**1.00**	0.81
RoboSQ	Qwen2.5-32B	**0.78**	**0.92**	0.81	**0.86**

TABLE II: **Task Failure Detection Agreement Score compared to Vector Database and different VLM models** Since Vector Databases (VectorDB) produce only relative similarity scores, we use the top k trajectories which k is the number of true successes. Even given the number of successes apriori, vector database performance is close to random guessing with known distribution. In contrast, without knowing the groundtruth failure distribution, RoboSQ with Qwen2.5-32B achieves a significantly higher F1 score, and RoboSQ with GPT-4o is more conservative in labeling trajectories as failures



(a) Image Input Method (b) Camera View

Fig. 3: **Sensitivity Analysis of Task Failure Agreement Score under different input methods.**

Comparison with alternative systems and models. In failure detection, Table III suggests RoboSQ demonstrates strong performance in detecting task failures, with balanced precision and recall across both classes.

Image Input Method Comparison. Across all frame counts, image concatenation consistently outperforms streaming when the same camera setup is used. At N=2, the performance gap is largest: concatenation achieves 0.68 accuracy and 0.82 F1, while streaming lags at 0.36/0.54. This advantage persists across larger N: concatenation leads streaming by 6–7 points in accuracy and 4–5 points in F1 at N=4 and N=6, and still maintains smaller gains at N=8 and N=10.

Camera Selection For the *concatenation* method using the external and wrist views, both accuracy and F1 score increase with more frames up to N=6, peaking at 0.76 accuracy and 0.87 F1. Beyond that, performance slightly degrades at N=8 and N=10, suggesting diminishing returns and possible context dilution in the larger image. In contrast, the *streaming* method shows more gradual improvement, with accuracy and F1 climbing from 0.36/0.54 at N=2 to a maximum of 0.72/0.84 at N=8. A slight drop is observed at N=10, po-

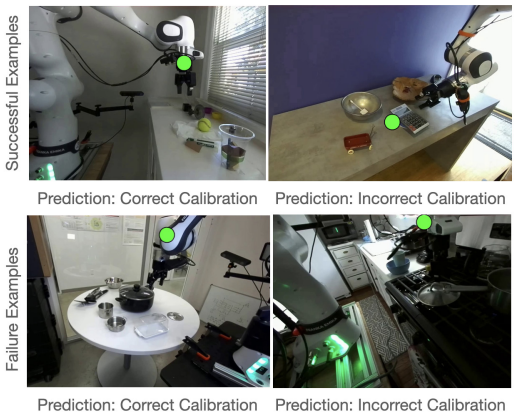


Fig. 4: **Examples of Extrinsic Calibration Error Detection By Reprojection** We show two successful examples and failure examples. RoboSQ can predict calibration errors successfully if both the main body of the Franka arm is visible. In failure examples, (Left) RoboSQ fails to detect the calibration error because the end effector is occluded by the robot joints. (Right) RoboSQ misdetects because the majority of robot end-effector is not in the image frame but can be inferred by the gripper position.

		RoboSQ Prediction	
		Correct Cali.	Incorrect Cali.
Ground Truth	Correct Cali.	51	10
	Incorrect Cali.	4	35

TABLE III: **Confusion Matrix for Calibration (Cali.) Error Detection.** RoboSQ achieves 86.2% accuracy and 87.9% F1 score in identifying calibration error for 100 demonstrations.

tentially due to prompt fatigue or over-conditioning. Interestingly, for wrist-only views, streaming performance continues to improve consistently with more frames—reaching 0.74 accuracy and 0.86 F1 at N=10—indicating that egocentric temporal context is especially useful when using streaming-style prompts.

C. Extrinsic Calibration Error Detection

Incorrect calibration between camera and end-effector frames is a common but difficult-to-detect issue in robot datasets. Automatically identifying calibration issues between camera and end effector state recordings is important. Identifying such errors typically require human to project a known cartesian position to the image and visually verify the calibration. We note that this process is considered as a visual sanity check over a large dataset instead of fine-grained specialist calibration detection models, such as CtRNet-X[33].

At time t, given the end effector cartesian position, camera intrinsics and extrinsics, RoboSQ identifies trajectories with calibration errors by combining image projection and VLM inference. For each frame, we project the end-effector pose from the robot's kinematic chain to the camera frame using the provided extrinsic calibration matrix, and query RoboSQ: "This image shows a robot's end effector position (large green circle) projected onto a camera view. The green dot is positioned where you would expect the robot's end effector at the end of the robot's arm connecting to the gripper. Is the dot correctly positioned? "

We evaluate RoboSQ with 100 trajectories from DROID-100, an author-curated diverse subsection of DROID dataset. We use its raw calibration information. As noted in DROID [5], some cameras in its raw robot demonstrations are not calibrated. We manually label the calibration error in the randomly sampled trajectories and remove the ambiguous image frames, such as if the end effector is not visible or the projection is only centimeter from the center of the end effector. Table III shows the confusion matrix of the calibration success prediction. RoboSQ achieves 86.2% accuracy and 87.9% F1 score in identifying calibration error.

Figure 4 shows 4 qualitative examples of the RoboSQ prediction compared to groundtruth. In failure examples, we recognize that the semantic difference is important for VLM to respond correctly. When the end effector is occluded by the robot joints in the left image, the VLM fails to detect because it's unable to distinguish between end effector and robot joints. The failure example on the right requires sophisticated spatial reasoning capability that human annotator can infer the robot end-effector position when the majority of the robot body is not in the image frame.

D. Visual Complexity Scoring

Understanding the visual complexity of the dataset is important to train a robust robot policy. Existing research by Saxena et al. [34] shows that co-training robot policies with diverse visual complexities improves the policy robustness.

We implement a semantic relevance score (Definition III.1) with RoboSQ that provides fine-grained semantic scoring of robot trajectories. RoboSQ outputs a scalar semantic relevance score s that quantifies the overall quality of a trajectory with respect to a set of visual and task-related factors. We define a weighted semantic relevance score based on five criteria: *Visual Clarity* (lighting conditions and contrast), *Occlusion* (degree of self-occlusion caused by the robot arm), *Target Object Quality* (clarity and visibility of the target object), *Task Relevance* (alignment between observed behavior and task description), and *Scene Complexity* (number of objects and clutter). Each criterion is assigned a corresponding weight, forming the vector $\boldsymbol{\alpha} = [0.30, 0.20, 0.20, 0.15, 0.15]$.

Figure 5 illustrates the semantic relevance scores produced by RoboSQ on DROID100, a 100-trajectory subset of the full DROID dataset that captures diverse lighting, occlusion, and scene layouts. Qualitatively, we observe that low-scoring trajectories tend to contain occlusions, poor lighting, or ambiguous robot behavior, while high-scoring samples exhibit clear visual conditions.

E. Policy Learning with RoboSQ

We study how semantic qualification affects the performance of a learned policy with RoboSQ. Specifically, we test whether using qualified data that filtered or scored by RoboSQ can lead to improved policy performance compared to unfiltered or randomly selected data. The robot is tasked to place a stuffed animal toy to a black bowl in the presence of a distractor pot.

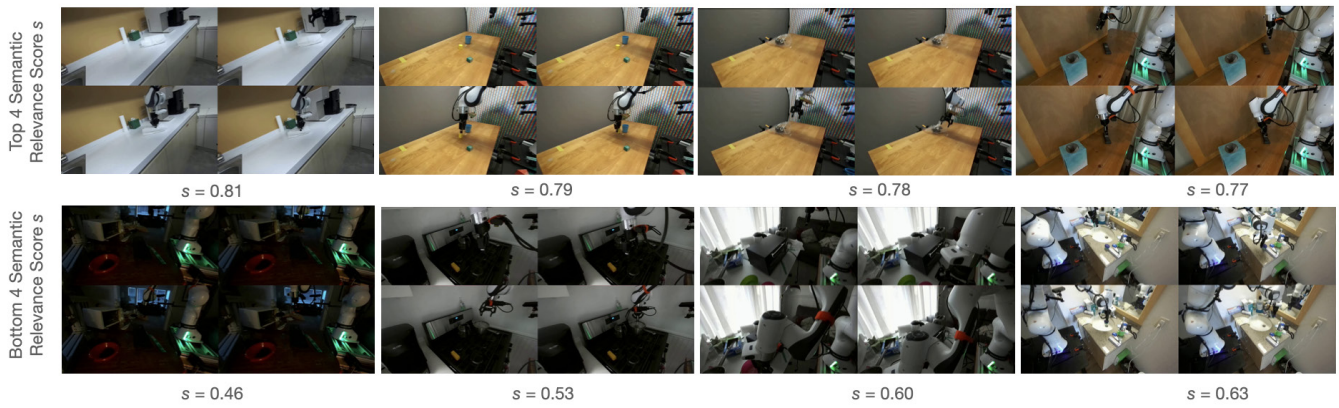


Fig. 5: **Visual Complexity Score s on DROID100** We run RoboSQ on DROID100, a 100-trajectory subset curated by DROID authors. The visual complexity score s of RoboSQ shows a strong relevance to the lighting, camera occlusion and scene complexity condition of the robot trajectory.

Setup. The task is picking a tiger and placing in a black bowl. Each dataset is used to train the same policy architecture: the Action Chunking Transformer (ACT) [35], under identical training hyperparameters. The ACT implementation is from [36]. The physical setup with UR5e can be found in Figure 1.

Data Distribution. We curate a training dataset of 90 trajectories. The dataset includes 50 high-quality demonstrations where the robot successfully picks up the stuffed animal and places it directly in the bowl. The other 40 trajectories exhibit low-quality demonstrations commonly seen in the Open X-embodiment dataset [4]. These failure cases are distributed across four categories that: (a) fails to grasp the stuffed animal, (b) successfully picks up the object but places it on the table, (c) the object in the wrong container (the pot instead of the bowl), and (d) low quality demonstration such as stalls, drops the object incorrectly and has to recover, or takes an indirect path to reach the target.

Dataset Selections (1) *Expert* selection uses a expert curated subset of high quality robot trajectories, (2) *Full* dataset selection includes all the trajectories, (3) *RoboSQ-Selection* with Definition. III.1, which prompts RoboSQ with a simple binary Selection. *RoboSQ-Selection 1* was prompted with "Does the robot successfully and smoothly pick a tiger and place it into the black bowl?" and *RoboSQ-Selection 2* was prompted with "Does the robot successfully and smoothly pick a brown stuffed toy and place it into the black bowl?" (4) *RoboSQ Top-K* using Definition. III.1.1 that rate the trajectory in rating successfulness, smoothness and efficiency, and rank with top 50 trajectories.

Metric For each method, we compute the sucess rate and Intersection over Union (IoU) for all dataset selection between the expert curated dataset (1) and other dataset selections in (2),(3),(4). IoU measures the proportion of overlap between the two sets, calculated as the amount of shared data divided by the total amount of data across both sets.

Results Table IV shows the policy performance under different dataset selection. The all-data policy trained with dataset selection (2) displayed the various failure modes present in the mixed dataset, including semantic failures where the stuffed animal was placed in the pot rather than the bowl, grasp failures preventing pickup of the toy, place failures resulting in the tiger being dropped on the table, and motion failures causing the robot to stall mid-trajectory.

Data Mix	# Traj.	IoU w/ Expert	Success Rate
Expert	50	100%	14 / 15
All Data	90	56%	1 / 15
RoboSQ Selection #1	44	84%	8 / 15
RoboSQ Selection #2	56	80%	11 / 15
RoboSQ Top k	50	**96%**	**13 / 15**

TABLE IV: **Policy Performance on Tiger Pick and Place Task** The base policy on Expert curated data with the 50 good demonstrations achieved a 14/15 success rate, while the policy trained on all 90 demonstrations managed only 1/15 success rate. RoboSQ with Top-K achieved performance comparable to expert selection with a 13/15 success rate.

All instances of RoboSQ outperformed the all-data policy. RoboSQ Selection 1 and RoboSQ Selection 2 used identical data selection methods, leading to similar IoU with the expert data. The success rates varied, with 8 out of 15 for Selection 1 and 11 out of 15 for Selection 2. The difference selected two prompts spans across different categories of failures, but the second prompt include more training data (56 versus 44 trajectories), indicating that a larger set of high-quality demonstrations can improve policy performance.

Moreover, RoboSQ Top k outperformed other RoboSQ methods with a 13/15 success rate approaching the source policy's 14/15 success rate. Interestingly, despite RoboSQ Selection 2 having an additional 6 trajectories, it performed worse than RoboSQ Top k. One advantage of RoboSQ Top k is that it identified trajectories where the robot initially misgrasped the stuffed animal but subsequently recovered. This inclusive data selection approach enables the resulting policy to develop recovery behaviors not seen in the source policy, suggesting that carefully selected data with mixed quality can enhance robustness.

VI. Conclusion and Future Work

In this paper, we propose RoboSQ, a semantic data qualification framework for robot data management. We use VLMs to perform visual question answering on robot trajectory

data to handle the semantic queries. We perform three case studies on DROID dataset, and results show that RoboSQ achieves high agreement with human-labeled metadata. We use a semantic data qualification metric for robot physical data that demonstrates comparable performance to human expert. We acknowledge the limitation that VLM prediction can be subject to prompt quality, leading to different behaviors in handling filtering or ranking. We consider it as future work to improve the stability and spatial understanding of VLM to improve the robot dataset queries. We also consider it as future work to identify a complete set of semantic queries for robot policy training.

REFERENCES

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, B. Zitkovich, *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning*, PMLR, 2023, pp. 2165–2183.

[2] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, *et al.*, *Octo: An open-source generalist robot policy*, `https://octo-models.github.io`, 2023.

[3] M. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, *et al.*, "Openvla: An open-source vision-language-action model," *Conference on Robot Learning (CoRL)*, 2024.

[4] O. X.-E. Collaboration *et al.*, *Open X-Embodiment: Robotic learning datasets and RT-X models*, 2024.

[5] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, *et al.*, "Droid: A large-scale in-the-wild robot manipulation dataset," in *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[6] S. Sagiroglu and D. Sinanc, "Big data: A review," in *2013 international conference on collaboration technologies and systems (CTS)*, IEEE, 2013, pp. 42–47.

[7] K. Chen, L. Fu, D. Huang, Y. Zhang, L. Y. Chen, H. Huang, K. Hari, A. Balakrishna, T. Xiao, P. R. Sanketi, *et al.*, "Robo-dm: Data management for large robot datasets," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2025.

[8] R. Cadene, S. Alibert, A. Soare, Q. Gallouedec, and T. Wolf, *Lerobot: Making ai for robotics more accessible with end-to-end learning*, `https://github.com/huggingface/lerobot`, 2024.

[9] "Gpt-4v(ision) system card," 2023.

[10] Google, "Gemini: A family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.

[11] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," *Advances in neural information processing systems*, vol. 36, 2024.

[12] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, R. Ring, E. Rutherford, *et al.*, "Flamingo: A visual language model for few-shot learning," *Advances in neural information processing systems*, vol. 35, pp. 23 716–23 736, 2022.

[13] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, *et al.*, "Palm-e: An embodied multimodal language model," in *International Conference on Machine Learning*, PMLR, 2023, pp. 8469–8488.

[14] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[15] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[16] E. Zhao, V. Raval, H. Zhang, J. Mao, Z. Shangguan, S. Nikolaidis, Y. Wang, and D. Seita, *Manipbench: Benchmarking vision-language models for low-level robot manipulation*, 2025. arXiv: `2505.09698 [cs.RO]`.

[17] R. Yang, H. Chen, J. Zhang, M. Zhao, C. Qian, K. Wang, Q. Wang, T. V. Koripella, M. Movahedi, M. Li, H. Ji, H. Zhang, *et al.*, *Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents*, 2025. arXiv: `2502.09560 [cs.AI]`.

[18] K. Chen, S. Xie, Z. Ma, P. R. Sanketi, and K. Goldberg, *Robo2vlm: Visual question answering from large-scale in-the-wild robot manipulation datasets*, 2025. arXiv: `2505.15517 [cs.RO]`.

[19] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, B. Zitkovich, *et al.*, "RT-1: Robotics transformer for real-world control at scale," *Robotics: Science and Systems (RSS)*, 2023.

[20] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on robot learning*, PMLR, 2018, pp. 651–673.

[21] H. Walke, K. Black, A. Lee, M. J. Kim, M. Du, C. Zheng, T. Zhao, P. Hansen-Estruch, Q. Vuong, A. He, V. Myers, K. Fang, *et al.*, "Bridgedata v2: A dataset for robot learning at scale," in *Conference on Robot Learning*, PMLR, 2023, pp. 1723–1736.

[22] M. Lepert, J. Fang, and J. Bohg, *Phantom: Training robots without robots using only human videos*, 2025.

[23] L. Y. Chen, C. Xu, K. Dharmarajan, M. Z. Irshad, R. Cheng, K. Keutzer, M. Tomizuka, Q. Vuong, and K. Goldberg, "Rovi-aug: Robot and viewpoint augmentation for cross-embodiment robot learning," in *Conference on Robot Learning (CoRL)*, Munich, Germany, 2024.

[24] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The faiss library," *arXiv preprint arXiv:2401.08281*, 2024.

[25] J. Wang, X. Yi, R. Guo, H. Jin, P. Xu, S. Li, X. Wang, X. Guo, C. Li, X. Xu, *et al.*, "Milvus: A purpose-built vector data management system," in *Proceedings of the 2021 international conference on management of data*, 2021, pp. 2614–2627.

[26] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[27] *Llama Index*, `https://github.com/run-llama/llama_index`.

[28] OpenAI, *Streaming responses*, Accessed: 2025-08-28, 2025.

[29] K. Feng, K. Gong, B. Li, Z. Guo, Y. Wang, T. Peng, B. Wang, and X. Yue, "Video-r1: Reinforcing video reasoning in mllms," *arXiv preprint arXiv:2503.21776*, 2025.

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

[31] G. R. Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl, *et al.*, "Gemini robotics: Bringing ai into the physical world," *arXiv preprint arXiv:2503.20020*, 2025.

[32] Q. Yu, Z. Zheng, S. Song, Z. Li, F. Xiong, B. Tang, and D. Chen, "Xfinder: Large language models as automated evaluators for reliable evaluation," *arXiv preprint arXiv:2405.11874*, 2024.

[33] J. Lu, Z. Liang, T. Xie, F. Ritcher, S. Lin, S. Liu, and M. C. Yip, "Ctrnet-x: Camera-to-robot pose estimation in real-world conditions using a single camera," *arXiv preprint arXiv:2409.10441*, 2024.

[34] V. Saxena, M. Bronars, N. R. Arachchige, K. Wang, W. C. Shin, S. Nasiriany, A. Mandlekar, and D. Xu, "What matters in learning from large-scale datasets for robot manipulation," in *International Conference on Learning Representations (ICLR)*, 2025.

[35] T. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *RSS*, vol. abs/2304.13705, 2023.

[36] J. Kerr, K. Hari, E. Weber, C. M. Kim, B. Yi, T. Bonnen, K. Goldberg, and A. Kanazawa, "Eye, robot: Learning to look to act with a bc-rl perception-action loop," *arXiv preprint arXiv:2506.10968*, 2025.