R+X: Retrieval and Execution from Everyday Human Videos

Georgios Papagiannis* **Norman Di Palo*** **Pietro Vitiello** **Edward Johns**

The **Robot Learning Lab** at **Imperial College London**

Abstract: We present R+X, a framework which enables robots to learn skills from long, unlabelled first-person videos of humans performing everyday tasks. Given a language command from a human, R+X first retrieves short video clips containing relevant behaviour, and then conditions an in-context imitation learning technique on this behaviour to execute the skill. By leveraging a Vision Language Model (VLM) for retrieval, R+X does not require any manual annotation of the videos, and by leveraging in-context learning for execution, robots can perform commanded skills immediately, without requiring a period of training on the retrieved videos. Experiments studying a range of everyday household tasks show that R+X succeeds at translating unlabelled human videos into robust robot skills, and that R+X outperforms several recent alternative methods. Videos are available at this anonymised website https://sites.google.com/view/r-plus-x.

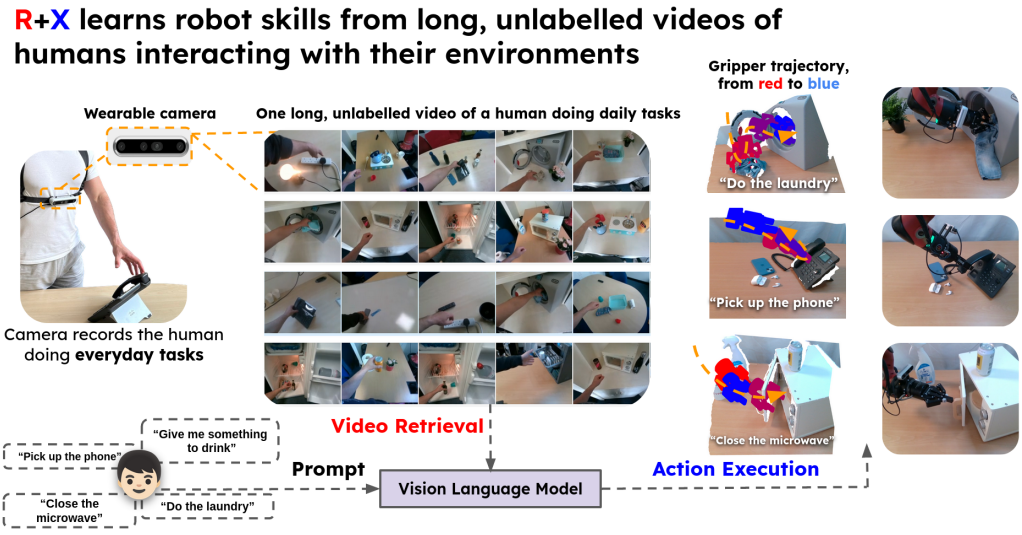

Figure 1: Given a language prompt, R+X first retrieves short relevant video clips extracted from a long unlabelled video of a human performing everyday tasks, recorded with a wearable camera. With the use of the retrieved video clips and a VLM, R+X performs in-context imitation learning allowing it to immediately generate and execute the desired behaviour on the robot.

1 Introduction

Robot learning of diverse, everyday tasks in natural environments, is a significant challenge. Imitation learning is a promising solution which has been widely adopted in recent years [1, 2], but it remains difficult to scale up due to the cost of hardware and human demonstration time. If robots could

*Equal contribution, random order.
*Contact at: {g.papagiannis21, n.di-palo20}@imperial.ac.uk

instead learn from data which does not require any robot hardware or dedicated human time, the difficulty of scaling up data collection would be lowered substantially.

In this work, we study the problem of learning robot skills from long, unlabelled, first-person videos of humans performing everyday tasks in everyday environments. *"Long, unlabelled videos"* means that a human simply goes by their everyday life and passively records videos of themselves without the need to specify which behaviour is being performed. *"Everyday environments"* means that videos contain diversity in tasks, scenarios, objects, distractors, illumination, and camera viewpoints and motion. It is likely that such videos will become abundant through the adoption of wearable devices such as AR headsets and glasses [3, 4, 5, 6], and thus offer a significant opportunity for future scalability if robots could learn from such videos.

Previous approaches to learning from videos of humans have often relied on a set of strong constraints and assumptions, such as human videos manually aligned with robot videos, human videos manually labelled with language descriptions or demonstrations on robot or MoCap hardware [7, 8, 2, 9, 10, 11, 12]. In this work, we remove all of these constraints and present a framework that requires only an unlabelled first-person video depicting tens of tasks.

Our framework, **R+X**, is a two-stage pipeline of **R**etrieval and E**X**ecution shown in Figure 1 that uses Foundation Models for both stages. Upon receiving a language command from a user, a Vision Language Model retrieves all clips where the human executes the specified task. By extracting the trajectories of the human's hand in each clip, we then employ a few-shot in-context imitation learning method to condition on these trajectories, which enables a robot to ingest, learn from, and replicate the retrieved behaviours to previously unseen settings and objects. The recent literature demonstrated that, by finetuning large Vision Language Models on robotics data, they can transfer their common knowledge and ability to reason and plan to robotics settings [13, 14]. This, however, requires very expensive finetuning of often intractably large models. With our proposed framework, we can equally leverage these abilities but now via retrieval and video understanding, thus without the need for any finetuning. Rather than requiring explicit policy training on the retrieved videos, skills can be learned, and executed immediately following the language command. In particular, we demonstrate the benefits that this brings over spatial and language generalization, compared to large monolithic policy networks.

In summary, the two main properties of R+X are: **1)** it enables robots to execute everyday tasks from language commands, given long, unlabelled, first-person videos collected naturally by recording a human's daily activities, and **2)** it achieves this without the need for any training or finetuning of models, allowing it to learn and execute tasks immediately.

2 Related Work

Collecting Robotics Data. Scaling data collection has been proven to be a successful path towards increasingly more general machine learning models [15, 16, 17, 18]. To collect robotics data, the most common paradigm is to teleoperate robots, collecting datasets of paired observations and actions [1]. This, however, needs dedicated teleoperation hardware, and needs human operators to allocate their time to *actively teach a robot new tasks*. In our framework, a human user interacts with their environments as usual, completing the tasks they wish, while a robot *passively learns to emulate it*, resulting in a more scalable and time efficient paradigm.

	Multi task no label/align videos	No robot data	Non-prehensile tasks	New obj gener.	Distractors (both train & test)	No MoCap hardware
Vid2Robot	✗	✗	✓	✓	✓	✓
WHIRL	✗	✗	✓	✗	✓	✓
DITTO	✗	✓	✗	✗	✗	✓
ScrewMimic	✗	✗	✗	✓	✓	✓
Orion	✗	✓	✗	✗	✗	✓
DexCap	✗	✓	✓	✓	✓	✗
R+X	✓	✓	✓	✓	✓	✓

Figure 2: The main assumptions and constraints of many recent Learning from Observation methods.

Learning from Human Videos. Many recent works have proposed solutions to teach new skills to robots by observing a human executing such tasks. However, as we illustrate in Fig. 2, they often

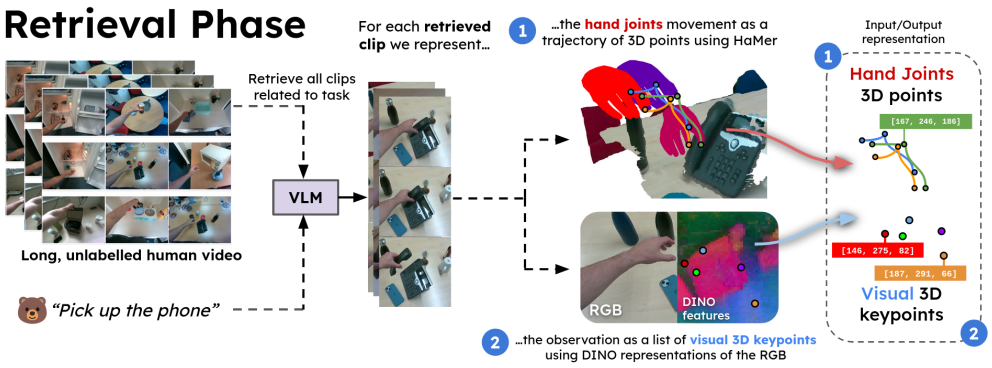

Figure 3: Upon receiving a language command, R+X retrieves all the relevant clips from the human video. Each retrieved clip is transformed from pixels to a sparse 3D points representations of the hand joints movement and salient parts of the visual observation.

relied on a set of assumptions and constraints that limited their use "in-the-wild". Bahl et al. [8], Wang et al. [19] require a combination of human data and either robot exploration or teleoperation, therefore needing active assistance of the user in teaching the robot, and videos are recorded from a fixed, third-person camera. R+X relies entirely on the videos recorded by the user with a mobile camera in their natural environments. Heppert et al. [9], Zhu et al. [11], Bahety et al. [10] can learn robot skills entirely from human videos. Heppert et al. [9], Zhu et al. [11], however, focus on replicating the object trajectory from the demo, and cannot perform tasks that do not involve grasping and moving, such as pushing or pressing, that R+X can execute. Bahety et al. [10] can learn a larger repertoire of skills, but still relies on learning single tasks in isolation from a fixed camera, while R+X can replicate tasks from a given a language command after receiving a single, long, unlabelled video depicting tens of tasks, without the need to specify which clip demonstrated which behaviour. Methods like Wang et al. [12] allow a user to naturally interact with their environment while collecting data that can teach a robot new skills. However, they require additional hardware to wear, like specialised MoCap gloves. R+X only needs a single RGB-D camera, and does not require any extrinsics calibration. This means that data could be recorded using a wearable camera, smart glasses, AR visors, and more.

Language-Conditioned Policy Learning. Robots should learn a wide repertoire of skills, and be able to execute many different tasks while deployed. Language is considered a viable way to instruct robots and guide their task execution [1, 14, 13]. The common approach is to train a large, language and image conditioned policy that can, at test time, receive user commands and camera observations [1]. This, however, presents some criticalities: training such models can require enormous computational effort [13], and unlike other machine learning applications, robots should learn new skills over time. Therefore, re-training or finetuning such networks should be avoided. R+X therefore does not train or finetune any networks, but takes advantage of pre-trained models able to retrieve examples of tasks [16], and to learn to emulate and execute new behaviour directly at deployment via in-context learning [20].

Additional papers are discussed in the Supplementary Material.

3 R+X: Retrieval and Execution

We now describe R+X, our proposed method to learn robot skills from long, unlabelled videos of humans interacting with their environments. We assume R+X has access to a long, unlabelled video of a user performing a multitude of tasks in many different locations. We call this long, first-person video of everyday tasks the "*human video*" $\mathcal{H}$ in the rest of the paper. The goal of our method is to learn to emulate the behaviours recorded in the human video upon receiving a language command from the user.

Overall, from a high-level perspective, our pipeline takes three inputs: **1)** a single, long, unlabelled video of all the interactions recorded by the humans, $\mathcal{H}$ **2)** a task to execute in language form, $\mathcal{L}$ and

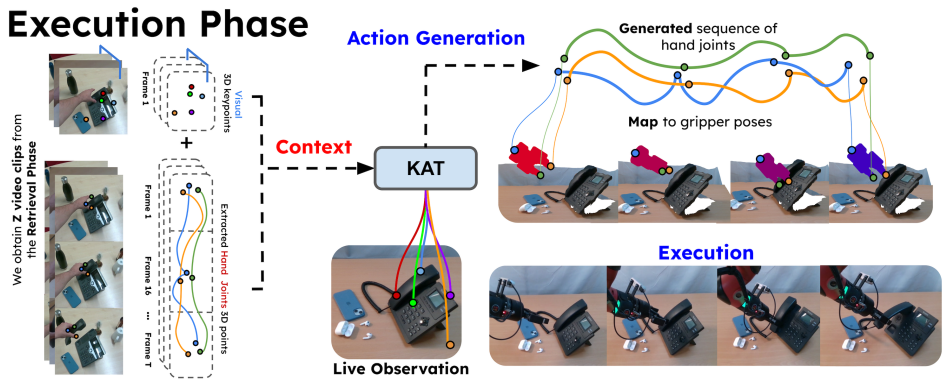

Figure 4: The visual 3D keypoints of the first frame of each of the Z videos obtained from the retrieval phase along with each extracted hand joint trajectory are used as context for KAT. To execute a skill, visual 3D keypoints are extracted from the live observation and used as input to KAT which generates a sequence of hand joints. By mapping this sequence to gripper poses the robot executes the task.

3) the current observation of the robot as an RGB-D image, $\mathcal{O}_{live}$. It then outputs a trajectory of 6-DoF gripper poses, which are executed by the robot to tackle the task at hand.

There are however a set of non-trivial challenges: while receiving a language command to execute at deployment, the robot receives **no language information before deployment**: the recorded video contains no more information than the recorded visual frames. Additionally, the robot receives **no action information**: unlike the case of teleoperation, no joints or end-effector position/velocity/acceleration data are recorded, and the correct movements need to be inferred by the videos alone: the problem is additionally complex due to the cross-embodiment between human videos and final robot actions. Furthermore, as the user is interacting with their natural environment, the visual observations can be **filled with distractor objects** typical of household and offices, unrelated to the task the user is performing. To tackle these challenges, we leverage the abilities of Foundation Models.

Specifically, at the first phase of R+X, which we call the **retrieval phase**, we use a VLM to extract from the human video all the examples of the desired behaviour described in the prompt $\mathcal{L}$ as a list of Z shorter video clips $[\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_Z]$. We map the Z videos into a lower dimensional 3D representation, extracting for each video: (1) a list of K *visual 3D keypoints* that describe the scene, $[k_1, k_2, \ldots, k_K]$ where $k = (x_k, y_k, z_k)$, and (2) the movement of the user's hand as a trajectory of length T of 21 hand joints that parametrise the MANO hand model [21], $\mathcal{J} = [j_1, j_2, \ldots, j_T]$, where $j = [[x_0, y_0, z_0], \ldots, [x_{21}, y_{21}, z_{21}]]$. Finally, at the second stage of R+X, which we call the **execution phase**, to emulate the behaviours observed in the retrieved video clips, we condition a few-shot in-context imitation learning model on this data that, given a live observation of the environment, it generates a trajectory of 3D hand joints to execute the desired behaviour described in the prompt. To map such joints to gripper poses, we designed a heuristic that we describe in the Supplementary Material.

3.1 Retrieval: Extracting visual examples from a long, unlabelled video

The first main phase of R+X is **retrieval**, shown in Figure 3. Upon receiving a language command $\mathcal{L}$, and given the human video $\mathcal{H}$, the goal is to retrieve all video clips from the human video that depict the execution of the requested task. This is accomplished using a recent Vision Language Model (VLM), Gemini Pro 1.5 Flash [22]. Gemini, which we denote $\mathcal{G}$, is natively multi-modal and can take as input images, videos, and text, and outputs text. We prompt the model with the human video, and ask it to retrieve the starting and ending seconds at which the received task happens. The inputs of this phase is therefore a language command and the human video, and the output is a list of Z shorter video clips demonstrating the desired task, $\mathcal{G}(\mathcal{H}, \mathcal{L}) \rightarrow [\mathcal{V}_1, \ldots, \mathcal{V}_Z]$. Each clip comprises T RGB-D frames, where T can vary across different clips. At the time of writing, Gemini can accept up to 2 hours of videos per call, and longer human videos can be trivially split into 2 hour chunks and processed in parallel. There is therefore no limit to the length of the human video.

3.1.1 Preprocessing Videos into a Sparse 3D Representation

Given the Z extracted video clips, $[\mathcal{V}_1, \ldots, \mathcal{V}_Z]$, we apply a preprocessing step that converts each video clip from a list of RGB-D frames to a set of 3D points describing the visual scene and the trajectory of the human's hand, a representation that we will then feed to our few-shot in-context imitation learning model.

Visual 3D keypoints. To transform the complex RGB-D observations into a lower-dimensional, easier to interpret input, we harness the powerful vision representations generated by DINO [23], a recent Vision Foundation Model. As proposed in [24, 20], given the Z clips retrieved as described before, we find a set of K common 3D visual keypoints that capture interesting semantic or geometrical aspects of the scene. We extract these keypoints from the *first frame* of each of the Z videos only.

We first compute the DINO descriptors for the first frame of each of the Z videos, obtaining a list of Z different $N \times 384$ outputs, where N is the number of patches of each image [25, 26, 23]. Via a clustering and matching algorithm [24], we select from the $N \times 384$ descriptors of the first frame of the first video, $\mathcal{O}_{\mathcal{V}_1,1}$, a list of the K descriptors that are the most common in all the remaining $Z-1$ frames, as shown in Figure 5. We denote these descriptors as $\mathcal{D} \in \mathbb{R}^{K \times 384}$. Therefore, this way, we autonomously extract descriptors that focus on the object of interest, as its appearance is common among videos, while distractors and overall scene will vary [24, 20].

Finally, for each of the K descriptors in $\mathcal{D}$, we extract keypoints by finding the K nearest neighbours between the $N \times 384$ descriptors of each the Z frames, and compute their 2D coordinates. We then project these in 3D using each frame's depth image and known camera intrinsics. As such, given the retrieved clips, we obtain and store a list $\Lambda = [\mathcal{K}_{V_1}, \ldots, \mathcal{K}_{V_z}] \in \mathbb{R}^{Z \times K \times 3}$ of visual 3D keypoints, K for each clip, where each $\mathcal{K}_{V_i} = [k_1, ..., k_K]$ and $k_j = (x_j, y_j, z_j)$. For a more detailed description please refer to Amir et al. [24].

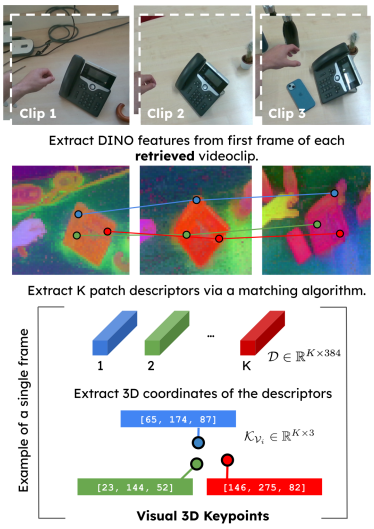

Figure 5: Visual keypoints are extracted autonomously from RGB via a matching algorithm [24].

Hand Joint Actions. To extract human actions from each retrieved video clip we use the HaMeR model [27], a recent technique for 3D hand pose estimation from images based on DINO. In particular, using HaMeR we extract from each video frame $\mathcal{O}_{\mathcal{V}_z,t}$ (where $1 \leq t \leq T$) of each of the Z clips the 3D hand pose, represented as a set of 21 3D points, $j_{\mathcal{V}_z,t}$, describing the hand joints that parameterise the MANO hand model, as it is commonly done in the literature [21]. As the camera moves between frames, we design a stabilisation technique to compute transformations between camera poses that is robust to dynamic scenes (for more details please see our Supplementary Material). This enables us to express the extracted hand joints relative to a single reference frame, that of the first camera frame of each clip. For each video clip $\mathcal{V}_z$, this process results in a sequence of 3D hand joint actions $\mathcal{J}_{\mathcal{V}_z} = [j_{\mathcal{V}_z,1}, \ldots, j_{\mathcal{V}_z,T}]$, expressed relative to the first frame of each video $\mathcal{O}_{\mathcal{V}_z,1}$ where the visual 3D keypoints are also expressed in. As such, at the end of this process we are left with a list $\mathcal{M} = [\mathcal{J}_{\mathcal{V}_1}, ...\mathcal{J}_{\mathcal{V}_Z}]$ of Z hand joint action sequences.

In summary, from the Z retrieved video clips, we extracted the list of hand joints actions $\mathcal{M}$, along with the list of visual 3D keypoints Λ, that will be used as context for our in-context imitation learning model, as Z input-output pairs.

3.2 Execution: Few-Shot, In-Context Imitation from Video Examples

The second main phase of R+X is **execution**. Our framework is based on the use of a model capable of performing few-shot, in-context imitation learning, receiving a few examples of desired inputs and outputs pairs describing a desired behaviour, and able to replicate such behaviour immediately upon receiving a new input. We use **Keypoint Action Tokens (KAT)** to achieve this, a recently proposed technique that takes 3D visual keypoints as input and outputs a trajectory of 3D points describing

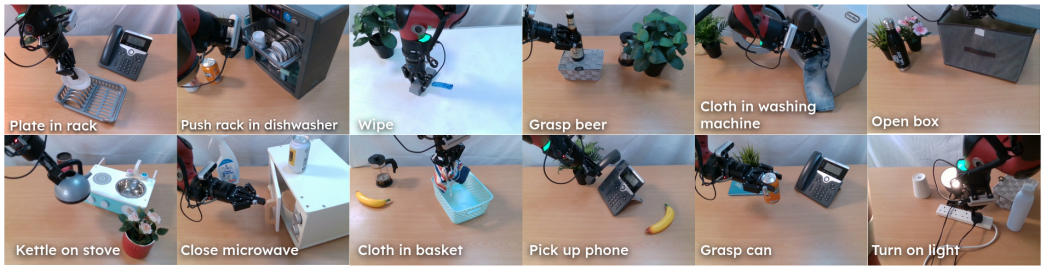

Figure 6: We test R+X on 12 everyday tasks, executed by a human user in different rooms and with different distractors.

the gripper movement. Instead of explicitly training a model on robot data, KAT demonstrates that recent, off-the-shelf Large Language Models are able to extract such numerical patterns, and behave as few-shot, in-context imitation learning machines, without the need for any further finetuning.

Given the output of the retrieval phase $[\Lambda, \mathcal{M}]$ and a new, live RGB-D observation collected by the robot $\mathcal{O}_{live}$, we extract its visual 3D keypoints representation $\mathcal{K}_{live}$ by first extracting its $N \times 384$ DINO descriptors, and then finding the K nearest neighbours to each of the K descriptors in $\mathcal{D}$, that we obtained in the retrieval phase. This results into K 2D coordinates, that we project in 3D. We then input to KAT as context $[\Lambda, \mathcal{M}]$, as Z examples of the desired input-output mapping, and the new visual 3D keypoints, and generate a new trajectory of desired hand joint actions, $\text{KAT}([\Lambda, \mathcal{M}], \mathcal{K}_{live}) \rightarrow \mathcal{J}_{live}$ with $\mathcal{J}_{live} = [j_1, \ldots, j_T]$, as shown in Figure 4. We then map the predicted trajectories of hand joints actions into gripper poses as described in our Supplementary Material, and execute them.

To summarize, given a language command $\mathcal{L}$ and a human video $\mathcal{H}$, R+X first **retrieves** Z videos depicting the described behaviour using a VLM. From the retrieved videos, a list of $[\Lambda, \mathcal{M}]$ visual 3D points and hand joint actions are extracted. Then, to execute the desired behaviour described in the prompt $\mathcal{L}$, $[\Lambda, \mathcal{M}]$ along with the visual 3D keypoints of the live observation $\mathcal{K}_{live}$ are used as context for KAT that performs few-shot in-context imitation learning to generate a trajectory of hand joints actions, which is mapped to gripper poses and **executed** on the robot.

4 Experiments

Human Video. We collect the human video $\mathcal{H}$ using an Intel RealSense 455, worn by the user on their chest as shown in Figure 1. To reduce downstream computational time, we filter out each frame in which human hands are not visible right after recording. As our robot is single-armed, we limit ourselves to single hand tasks. However, our method could identically be applied to bimanual settings and dexterous manipulators. The video is collected in many different rooms and buildings. In the Supplementary Material, we also discuss the use of multiple views beyond a chest-camera.

Robot Setup. At execution, we use a Sawyer robot equipped with a RealSense 415 head-camera. The robot is equipped with a two-fingered parallel gripper, the Robotiq 2F-85. As the robot is not mobile, we setup different scenes in front of it with variations of the tasks recorded by the user, placing several different distractors for each task, while the human video was recorded in many different rooms. Although we have a wrist-camera mounted, we do not use that in our work.

Tasks. To evaluate our proposed framework, we use a set of 12 everyday tasks, where the user interacts with a series of common household objects, listed in Fig. 6. We include movements like grasping, opening, inserting, pushing, pressing, and wiping.

Baselines. We compare R+X, and its retrieval and execution design, to training a single, language-conditioned policy. To obtain language captions from the human video, we use Gemini to autonomously caption snippets of the video, obtaining a *(observation, actions, language)* dataset. We finetune R3M (ResNet-50 version [28]) [29] and Octo [30] on this data. We extend R3M to also encode language via SentenceBERT and use a Diffusion Policy [31] head to predict actions from

Method / Task	Plate	Push	Wipe	Beer	Wash	Box	Kettle	Micro.	Basket	Phone	Can	Light	Avg.
R3M-DiffLang	0.5	0.7	0.4	0.7	0.5	0.5	0.4	0.8	0.7	0.4	0.7	0.3	0.55
Octo	0.5	0.8	0.5	0.6	0.5	0.5	0.4	0.7	0.6	0.4	0.6	0.3	0.53
R+X	**0.6**	0.8	**0.7**	**0.8**	**0.6**	**0.7**	**0.6**	0.8	0.7	**0.7**	**0.8**	**0.6**	**0.7**

Table 1: Result of the various methods on the 12 proposed tasks.

intermediate representations. We denote this version as R3M-DiffLang. More details on all aspects are provided in the Supplementary Material.

4.1 Results

Can R+X learn robot skills from long, unlabelled videos? How does it perform with respect to a monolithic language-conditioned policy? In these experiments, we evaluate the performance of R+X in learning a set of everyday skills. At deployment, we place the object to be interacted with in front of the robot, and issue a language command. We then run the retrieval and execution pipeline using the human video, the current observation and the language command. We run 10 episodes per task, randomising 1) the object pose 2) the type and number of distractors 3) for tasks where it is possible, we swap the object to interact with, with another one from the same class to test for generalisation (e.g. a different can, a different piece of clothing, a different telephone). More details are provided in the Supplementary Material. In Table 1, we report the performance of R+X on these tasks, together with the baselines. As we demonstrate, the framework is able to tackle a wide range of everyday tasks, surpassing the monolithic policy baselines. These results prove the benefit of modeling language-conditioned learning from observation as distinct retrieval and execution phases, to fully leverage the abilities of recent Foundation Models. In the Supplementary Material, we investigate more in depth the performance of R+X on unseen objects and in the presence of distractors.

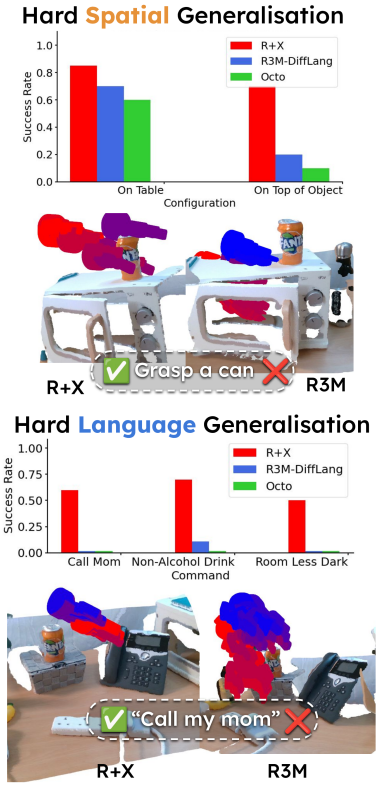

Figure 7: Examples of hard spatial and language generalisation. Gripper moves from red to blue.

What are the main sources of difference in performance between R+X and a monolithic policy? In these experiments, we investigate more in detail what changes in the inputs lead to the most noticeable difference in performance between R+X and the baselines, R3M-DiffLang and Octo. We explored two aspects, related to two properties of R+X:

Hard Spatial Generalisation: R+X, by retrieving videos of the desired task, can also extract a series of relevant keypoints $\mathcal{K}_{live}$ from the current observation $\mathcal{O}_{live}$, something not possible when using a single policy network. Going from RGB-D to a list of 3D points for inputs and outputs allows us to apply simple geometric data augmentation techniques for KAT, such as normalisation or random translations and rotations. This leads to a stronger spatial generalisation: in the *"grasp a can"* and *"grasp a beer"* tasks we test performance of each method when the objects are on the table, or when they are positioned on top of other objects (a box, a microwave). By running 5 test episodes for each case, we demonstrate how R+X retains strong performance, while the performance of the baselines drop. Results and an example of the predicted actions can be seen in Fig. 7, top.

Hard Language Generalisation: By leveraging the language and video understanding abilities of recent large Vision Language Models, such as Gemini, R+X can interpret and execute nuanced commands. To evaluate this, we setup a scene with many objects, and ask the robot to perform three tasks: *"give me something to call my mom"*, *"give me a non-alcoholic drink"* and *"make the room less dark"*. We run 5 test episodes for each of these commands, modifying the position of the objects

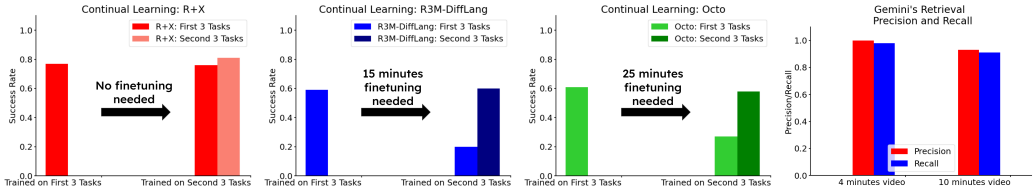

Figure 8: **Left (barplots 1-3)**: We compare R+X and the baselines' ability to learn tasks in succession, and the time needed to learn such new tasks. **Rightmost barplot**: Gemini's retrieval performance.

and the distractors. The language-conditioned policies struggle to interpret these commands, that are strongly out of distribution. R+X, on the other hand, leverages Gemini's ability to understand the meaning of these commands, as it is able to retrieve useful video clips from the human video (respectively, picking up the phone, grasping a Fanta can, and turning on the light). Results and an example of output gripper trajectories can be seen in Fig. 7, bottom.

Can R+X learn task sequentially over time? In these experiments, we demonstrate how R+X can learn tasks continually, with no need for any additional training or finetuning, while obtaining strong performance both on the new task and on the old ones, a desirable ability for a robot learning from an ever increasing dataset of human experience. To measure this ability, and highlight the difference behaviour with respect to a single language-conditioned policy, we first collect 10 demos for 3 tasks. We train the baselines on these tasks (after extracting captions via Gemini as described for the experiments of Table 1) and evaluate them and R+X. Then, we add 10 demos of 3 new tasks, finetune the baselines, then measure performance on the 3 new tasks, and on the 3 old tasks for all methods. In Fig. 8, Left, we see how the performance of R3M-DiffLang and Octo deteriorates on the old task, due to the well known effect of catastrophic forgetting [32]. On the other hand, R+X performance does not deteriorate, due to the ability to retrieve from an ever-growing video of tasks and adapting the behaviour model at test time on the retrieved data.

If instead we train the baselines on all the data each time, the performance does not deteriorate: however, this leads to a substantial growth in time needed to train. As the dataset grows, those networks would need an increasing amount of time per each new added task in order to be retrained, while R+X does not need any training or finetuning.

How does the length of the videos affect the retrieval performance? In this experiments, we first collect a video with 10 demos of 5 tasks (*kettle on stove, cloth in basket, pick up phone, cloth in washing machine, close microwave*), and measure Gemini's ability to retrieve clips for each task. We then add 10 demos for 5 additional tasks, and compute again the precision and recall of retrieval of the original 5 tasks, to study how length of video and number of tasks affects it. The original video is close to 4 minutes in length, while the second is close to 10. In Fig. 8, Right, we show the mean over all tasks of retrieval's precision and recall. Results demonstrate how Gemini's performance remains high while increasing the length of the video, allowing to scale R+X as the human records more and more tasks.

5 Discussion and Limitations

We presented R+X, a method to learn robot skills from long, unlabelled videos of users interacting with their environments, demonstrating its clear benefits over training monolithic language-conditioned policy networks. There are, however, still a series of limitations that we here describe, proposing possible avenues for future work. The method is currently bottlenecked by the performance of Vision Language Model, that has however seen a drastic improvement in recent time and will likely still increase. While we can learn a large series of everyday tasks, the errors arising from the hand pose prediction currently inhibits us from learning precise tasks. Furthermore, while the keypoints extraction leads to strong spatial generalisation properties, it can become a bottleneck in the presence of many similar objects: methods that find a dynamic number of keypoints might tackle this issue. Finally, due to the use of Foundation Models for retrieval and execution, our method has a few seconds delay between the issuing of the command and execution, that however has reduced dramatically over the last months with the release of faster models such as GPT-4o [33] or Gemini Flash [22].

References

[1] O. X.-E. Collaboration. Open x-embodiment: Robotic learning datasets and rt-x models, 2023. arXiv:2310.08864.

[2] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning, 2022.

[3] Apple. Apple vision pro. URL https://www.apple.com/apple-vision-pro/.

[4] Meta. Meta quest 3. URL https://www.meta.com/gb/quest/quest-3/.

[5] M. Ray-Ban. Ray-ban smart glasses. URL https://www.ray-ban.com/uk/ray-ban-meta-smart-glasses.

[6] MagicLeap. Magicleap. URL https://www.magicleap.com/.

[7] V. Jain, M. Attarian, N. J. Joshi, A. Wahid, D. Driess, Q. Vuong, P. R. Sanketi, P. Sermanet, S. Welker, C. Chan, et al. Vid2robot: End-to-end video-conditioned policy learning with cross-attention transformers. *arXiv e-prints*, pages arXiv–2403, 2024.

[8] S. Bahl, A. Gupta, and D. Pathak. Human-to-robot imitation in the wild. *arXiv preprint arXiv:2207.09450*, 2022.

[9] N. Heppert, M. Argus, T. Welschehold, T. Brox, and A. Valada. Ditto: Demonstration imitation by trajectory transformation. *arXiv preprint arXiv:2403.15203*, 2024.

[10] A. Bahety, P. Mandikal, B. Abbatematteo, and R. Martín-Martín. Screwmimic: Bimanual imitation from human videos with screw space projection. *arXiv preprint arXiv:2405.03666*, 2024.

[11] Y. Zhu, A. Lim, P. Stone, and Y. Zhu. Vision-based manipulation from single human video with open-world object graphs. *arXiv preprint arXiv:2405.20321*, 2024.

[12] C. Wang, H. Shi, W. Wang, R. Zhang, L. Fei-Fei, and C. K. Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *arXiv preprint arXiv:2403.07788*, 2024.

[13] A. Brohan et al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. *arXiv e-prints*, art. arXiv:2307.15818, July 2023. doi:10.48550/arXiv.2307.15818.

[14] D. Driess et al. PaLM-e: An embodied multimodal language model. In A. Krause et al., editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8469–8488. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/driess23a.html.

[15] T. Brown et al. Language models are few-shot learners. In H. Larochelle et al., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[16] Gemini-Team. Gemini: A family of highly capable multimodal models, 2023.

[17] S. Mirchandani et al. Large Language Models as General Pattern Machines. *arXiv e-prints*, art. arXiv:2307.04721, July 2023. doi:10.48550/arXiv.2307.04721.

[18] J. Hoffmann et al. Training Compute-Optimal Large Language Models. *arXiv e-prints*, art. arXiv:2203.15556, Mar. 2022. doi:10.48550/arXiv.2203.15556.

[19] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. *arXiv preprint arXiv:2302.12422*, 2023.

[20] N. Di Palo and E. Johns. Keypoint Action Tokens Enable In-Context Imitation Learning in Robotics. *arXiv e-prints*, art. arXiv:2403.19578, Mar. 2024. doi:10.48550/arXiv.2403.19578.

[21] J. Romero, D. Tzionas, and M. J. Black. Embodied hands: modeling and capturing hands and bodies together. *ACM Transactions on Graphics*, 36(6):1–17, Nov. 2017. ISSN 1557-7368. doi:10.1145/3130800.3130883. URL http://dx.doi.org/10.1145/3130800.3130883.

[22] G. Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024.

[23] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers, 2021. arXiv:2104.14294.

[24] S. Amir, Y. Gandelsman, S. Bagon, and T. Dekel. Deep vit features as dense visual descriptors, 2022.

[25] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[26] OpenAI. GPT-4 Technical Report. *arXiv e-prints*, art. arXiv:2303.08774, Mar. 2023. doi:10.48550/arXiv.2303.08774.

[27] G. Pavlakos, D. Shan, I. Radosavovic, A. Kanazawa, D. Fouhey, and J. Malik. Reconstructing hands in 3d with transformers, 2023.

[28] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[29] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation, 2022.

[30] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy, 2024.

[31] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2023.

[32] L. Wang, X. Zhang, H. Su, and J. Zhu. A comprehensive survey of continual learning: Theory, method and application, 2024.

[33] OpenAI. Gpt-4o. URL https://openai.com/index/hello-gpt-4o/.

[34] Y. Zhu, Z. Ou, X. Mou, and J. Tang. Retrieval-augmented embodied agents, 2024.

[35] M. Du, S. Nair, D. Sadigh, and C. Finn. Behavior retrieval: Few-shot imitation learning by querying unlabeled datasets. *arXiv preprint arXiv:2304.08742*, 2023.

[36] N. Di Palo and E. Johns. DINOBot: Robot Manipulation via Retrieval and Alignment with Vision Foundation Models. *arXiv e-prints*, art. arXiv:2402.13181, Feb. 2024. doi:10.48550/arXiv.2402.13181.

[37] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. van den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, D. de Las Casas, A. Guy, J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. W. Rae, E. Elsen, and L. Sifre. Improving language models by retrieving from trillions of tokens, 2022.

[38] H. Bharadhwaj, R. Mottaghi, A. Gupta, and S. Tulsiani. Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation, 2024. URL https://arxiv.org/abs/2405.01527.

[39] C. Yuan, C. Wen, T. Zhang, and Y. Gao. General flow as foundation affordance for scalable robot learning, 2024. URL https://arxiv.org/abs/2401.11439.

[40] S. Bahl, R. Mendonca, L. Chen, U. Jain, and D. Pathak. Affordances from human videos as a versatile representation for robotics, 2023. URL https://arxiv.org/abs/2304.08488.

[41] A. S. Chen, S. Nair, and C. Finn. Learning generalizable robotic reward functions from "in-the-wild" human videos, 2021. URL https://arxiv.org/abs/2103.16817.

[42] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. *The International Journal of Robotics Research*, 40:1419 – 1434, 2020. URL https://api.semanticscholar.org/CorpusID:220069237.

[43] T. Lüddecke and A. S. Ecker. Image segmentation using text and image prompts, 2022.

[44] H. Matsuki, R. Murai, P. H. J. Kelly, and A. J. Davison. Gaussian splatting slam, 2024.

[45] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct. 2015. ISSN 1941-0468. doi:10.1109/tro.2015.2463671. URL http://dx.doi.org/10.1109/TRO.2015.2463671.

[46] C. Doersch, Y. Yang, M. Vecerik, D. Gokay, A. Gupta, Y. Aytar, J. Carreira, and A. Zisserman. Tapir: Tracking any point with per-frame initialization and temporal refinement, 2023.

[47] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

[48] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

6 Supplementary Material

6.1 Additional Related Work

In this section, we discuss some additional related papers from the literature.

Retrieval in Robotics. Other papers have proposed the use of retrieval in robotics. Zhu et al. [34] retrieves directly from a bank of *policies* trained on robotics data. Du et al. [35] retrieves from an unlabelled data of robotics experience, not human videos, and needs an additional example at test time to guide the retrieval phase, while we leverage a language command alone. Di Palo and Johns [36] retrieves a goal observation to then perform visual servoing and replay a pre-recorded demo. R+X leverages large Vision Language Models to retrieve directly from long videos given a language query, without requiring pre-trained policies or new demonstrations as queries.

Retrieval Augmented Generation in Language Modelling: Large Language Models, through extensive pre-training and finetuning on web-scale datasets, are able to implicitly record countless facts and concepts in their weights [26, 33, 16]. However, it has been proved useful to add to their implicit knowledge the ability to explicitly access datasets of information, either offline or through web search [37]. This allows models to be more factually correct. In this work, we do not deal with factuality or information retrieval, but we take inspiration from these ideas to form a sort of "retrieval augmented execution", which is the main idea behind R+X.

Further Learning from Observation Papers Recently [38, 39] propose methods to learn skills from videos, similarly to our paper, but focusing on optical flow between frames. [38] however learns a visual-goal-conditioned policy, while we focus on language-conditioned skills extracted from a long, unlabelled video. Additionally, by focusing on optical flow of relevant objects, these methods might struggle on objects that barely move for some tasks, like pressing, as we show we can do with R+X by tracking the human hand directly. [40] learns, from in-the-wild human videos, where objects can generally be grasped and in what directions they can be moved, allowing a robot to passively learn to push, pick up, pull, etc. However, their method is more limited in the amount of skills it can learn, focusing on mostly linear movement of objects, while we show we can learn more articulated and dextereous tasks. [41, 42] show that human videos can be used to learn to guide robot exploration by extracting a reward function from videos. In our method, we show we can effectively learn skill without any need for robot teleoperation or autonomous exploration.

6.2 Processing the Long, Unlabelled Human Video

After recording the video of a human performing everyday tasks, we process it to remove unnecessary frames, i.e., frames that do not contain human hands. We achieve this automatically by leveraging HaMeR [27] to detect the frames where hands are present. Consequently, we are left with one long, unlabelled video of smaller video clips, concatenated together, each containing only frames where a human interacts with various objects with their hands. This video is then passed to Gemini to proceed with the retrieval phase of R+X, as discussed in section 3.1. Fig. 9 shows a short segment of the uncut video we recorded. The frames marked with red correspond to frames where no human hands were detected. The frames where human hands were detected are marked with green. After processing the video with HaMeR [27] we remove the frames marked with red and are left with one long video of the green frames concatenated together.

6.3 Stabilisation of First Person Videos

As we record videos using a wearable camera, the point of view of the camera changes substantially frame by frame. However, our method is based on representing the visual inputs as fixed, 3D keypoints, and the actions as a series of 3D hand joints actions expressed in the same frame of the keypoints. Therefore, we need to compute the relative movement of the camera at each step in order to express everything in the original frame of reference, the first of the recorded video clip.

The computer vision literature has proposed several methods to address this problem [44, 45]. However, most of these techniques assume that the scene in front of the camera is fixed, and only the camera is moving. This is in sharp contrast with our setting, where the arms and hands of the user move in front of the camera, together with some of the objects. Classic techniques, in our experiments, failed to achieve a robust estimation of the movement of the camera, a task we hereby

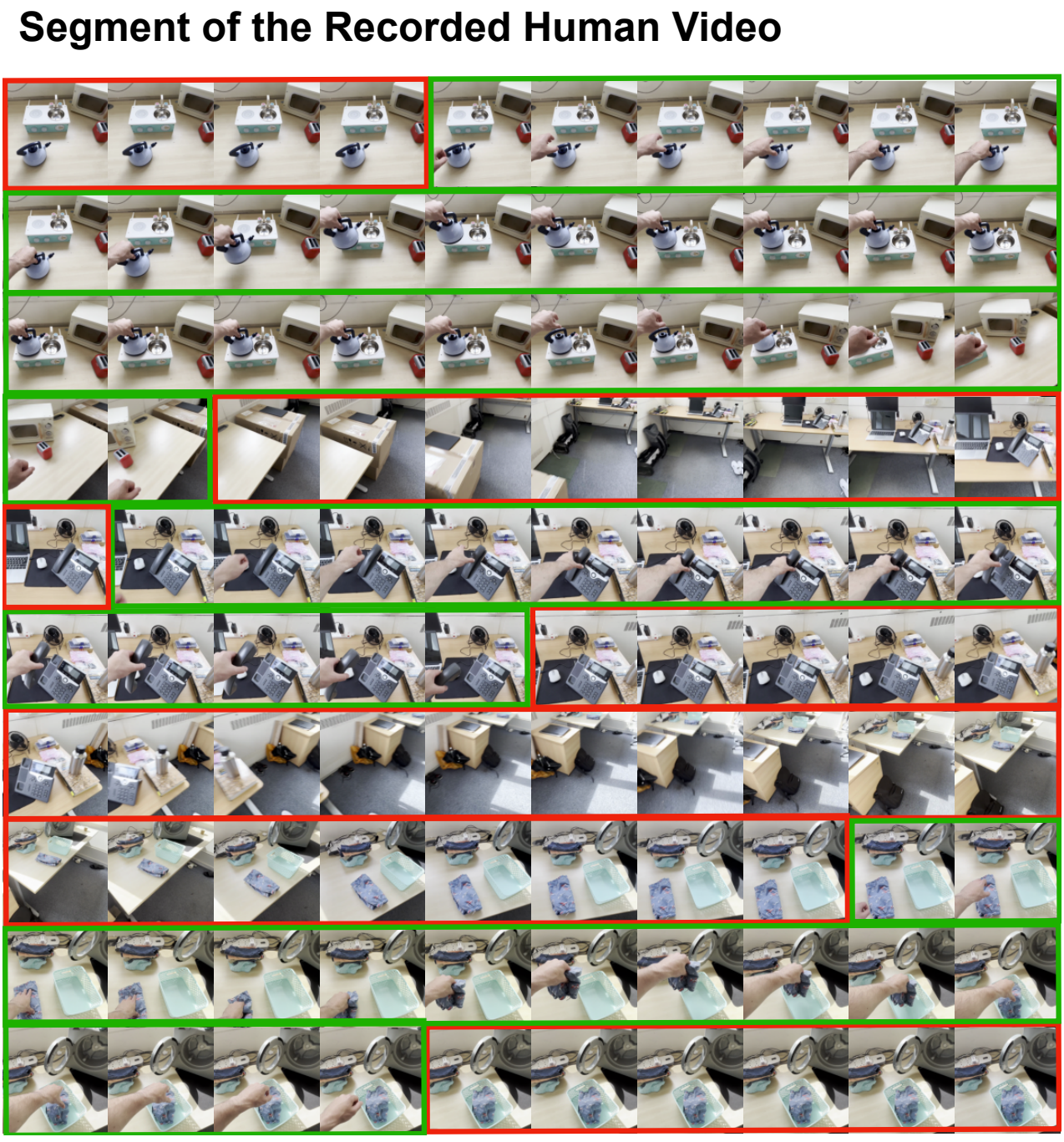

Figure 9: A short segment of our recorded long, unlabelled video of a human performing everyday tasks. The frames marked with red correspond to frames where no human hands were detected, while the frames marked with green correspond to frames where HaMeR [27] detected human hands. In practice, we discard the frames where no human hands are present and only retain a long video of frames concatenated together depicting only the human interacting with various objects.

Figure 10: An example of keypoints being tracked in a retrieved video clip. Notice how, thanks to the segmentation model [43], we can sample keypoints that stick to static parts of the scene, like the table.

refer has the *stabilisation of the camera*. In the following section, we describe how we stabilise each video clip we retrieve from $\mathcal{H}$ while using R+X.

In order to tackle the aforementioned problem, we leverage the use of a series of recent computer vision models. The first one, CLIPSeg [43], is an open-vocabulary image segmentation model. The model receives an image and a series of language queries as an input, and outputs a segmentation mask localising the pixels of the desired objects, $\text{CLIPSeg}(\mathcal{O}_{live}, [\mathcal{L}_{obj,0}, \ldots, \mathcal{L}_{obj,n}]) \rightarrow M_{live} \in \mathbb{R}^{H \times W \times 1}$, where H and W are the height and width of the observation. We query CLIPSeg with some static parts of any scene, namely *"floor", "wall", "table"*. In addition, we query it with *"arm", "hand", "person"* in order to obtain segmentation masks of parts of the user arms, and remove it from the segmentation mask of the static objects.

Given the segmentation of such static objects, we sample a set of random 2D pixel keypoints in that area. We then track the movement of such keypoints frame by frame using TAPIR [46], a recent keypoint-tracking network. The position of these keypoints in 2D at each frame allows us to then project them in 3D using the depth-channel from the RGB-D camera we use. Given these sparse point-clouds, we can compute the relative $SE(3)$ rigid transformation between each new frame and the first one. We perform this process separately for each video clip where human hands where detected as discussed in section 6.2.

This pipeline, in our experiments, has proven to be better than classic techniques [45], as it can handle the presence of the moving hands and arms of a person, together with the movement of some objects the user interacts with, as it learns to only focus on static parts of the environment, ignore the user, and track the selected keypoints over time.

6.4 Extracting Gripper Actions from Human Hands

To map the human hand joints to gripper actions we deploy different heuristics based on the type of interaction the human performs in the retrieved video clips. In scenarios where a robot is equipped with an end-effector with human-like fingers, such as a dexterous 5-fingered hand, such heuristics are not needed as the hand joints would map directly to the robotic, human-like fingers.

Fig. 11 shows instances of extracted hand joints using HaMeR [27] mapped to different gripper actions for grasping, pressing, and pushing tasks. For ease of illustration, this figure demonstrates how human hand joints are mapped to gripper poses from images captured in the human video. In practice, we deploy the following heuristics on the predictions made by each method. Our heuristics are applied to the Robotiq Gripper 2F85, but can be adapted accordingly to any other parallel jaw gripper.

Grasping. (Fig. 11, top) For tasks that require grasping we align spatially the tips of the gripper's fingers with the index and thumb tips of the hand. The vector joining the two tips defines an axis of rotation around which we rotate the gripper such that the bottom of the gripper is as close as possible to the middle point between the index mcp and the thumb dip. Aligning the gripper to these points enables us to obtain the gripper's pose. Consequently, for grasping tasks each method predicts, the index tip, thumb tip, index mcp and thumb dip joints. To compute the gripper close/open action, we

Figure 11: Examples of different human hand interactions and extracted gripper actions based on three different heuristics for grasping, pressing, and pushing. The blue dots in the third column correspond to the 21 hand joints extracted using HaMeR [27]. Which heuristic to use is determined by Gemini at the Retrieval phase based on Gemini's understanding of the language command provided by the human. Note that for the pressing and pushing task the gripper is rendered as open, but in practice the gripper is closed as if it grasping something.

employ an heuristic that measures the distance of the index and thumb tips and compares it to the robot gripper width.

Pressing. (Fig. 11, middle) For pressing tasks, we assume that the gripper is *closed*. To obtain the pose of the gripper we use three points at the point of contact of the two gripper fingers when the gripper is closed. Specifically, one point is at the tip of the fingers, the other in the middle, and the last at the bottom of the gripper's fingers. We align these three points to the index tip, index pip, index mcp, and index dip of the human hand respectively. Consequently, each method predicts the index tip, index pip, index mcp, and index dip.

Pushing. (Fig. 11, bottom) Pushing tasks, are similar to pressing tasks, where we use the same three points on the gripper but to obtain the gripper's pose, we instead align these points with the joints on the hand halfway between the index and middle finger tip, pip, mcp and dip.

For all heuristics, we match the pose of the gripper with the joints on the hand using singular value decomposition.

Detecting which heuristic to use is done automatically by Gemini based on the human's language command during task execution. As Gemini exhibits strong semantic language understanding, it can determine whether the requested task from the human involves grasping, pressing, or pushing.

6.5 Tasks Details and Success Criteria

We here list more details about the tasks we use in R+X, the generalisation abilities we study, and the success criteria.

- **Plate in Rack:** We randomise the position of the rack on the table in a 40cm by 30cm (width, height) area and its orientation in a $[-30, 30]$ degrees range. The robot starts with a plate grasped in its gripper. We place random distractors in half the runs, while half are distractors free. The task is successful if the plate can stand in one of the slots of the rack without falling over.
- **Push Rack in Dishwasher:** Due to the kinematics of the robot, we always put the dishwasher on the right side of the table, and randomise its position in a 35cm by 20cm area, and its orientation in a $[-20, 20]$ degrees range. As the object is considerably large, we cannot randomise excessively its starting point or it will end out of view/out of the robot's working area. We place random distractors in half the runs, while half are distractors free. The task is successful if the rack is no more than 5cm outside of the dishwasher, and starts being around 15cm outside of it.
- **Wipe Marker from Table:** We draw a random spot with a marker on an horizontally placed whiteboard, that emulates a mark on a table, some dirt, or something spilled. We randomise the shape of the spot, that is generally $15cm^2$ in area, and its position in a 40cm by 30cm area. The robot starts with a cleaning tool grasped. We place random distractors in half the runs, while half are distractors free. The task is successful if more than 80% of the spot it erased and cleaned.
- **Grasp Beer**: We place a beer in a random spot on the table in a 40cm by 30cm area. In this case, we also place it 2 out of 10 times on top of other objects to test for hard spatial generalisation. This setting is better studied in the "*Hard Spatial Generalisation*" subsection of the main paper. We place random distractors in half the runs, while half are distractors free. The task is solved if the robot has grasped and lifted the beer.
- **Cloth in Washing Machine**: We place the washing machine on the right side of the table, randomising its position in a 20cm by 20cm area, and its orientation in a $[-20, 20]$ degrees range. We then place a random piece of clothing (generally clothing for kids/infants) on the left side of the table, and randomise its position in a 25cm by 25cm area, therefore testing for both spatial and object generalisation. We place random distractors in half the runs, while half are distractors free. The task is completed if the robot can place the cloth inside the hole of the washing machine, also if part is outside: as our toy washing machine is very small and its receptacle is shallow (around 15cm), it is hard also for a human to fit a cloth entirely in there.
- **Open Box:** We place one of two possible boxes of different sizes in a 30cm by 30cm area on the table, and randomise its orientation in a $[-20, 20]$ degrees range. We place random distractors in half the runs, while half are distractors free. The task is successful if, during the episode, the robot has partially opened the box. Opening it completely is often challenging due to kinematics constraints.
- **Kettle on Stove:** We place a toy stove on the table, randomising its position in a 30cm by 20cm area, and its orientation in a $[-20, 20]$ degrees range. We then randomly place a random kettle in a 20cm by 20cm area and randomise its orientation in a $[-30, 30]$ degrees range. By placing seen and unseen kettles, we test both for spatial and object generalisation. We place random distractors in half the runs, while half are distractors free. The task is successful if the kettle is on top of the stove, which we consider true if the intersection of the stove area and the kettle base is more than 50% than the stove area.
- **Close Microwave:** We place a toy microwave on the right side of the table, randomising its position in a 20cm by 20cm area and its orientation in a $[-20, 20]$ degrees range. Additionally, we open its door in a range between $[10, 45]$ degrees, We place random distractors in

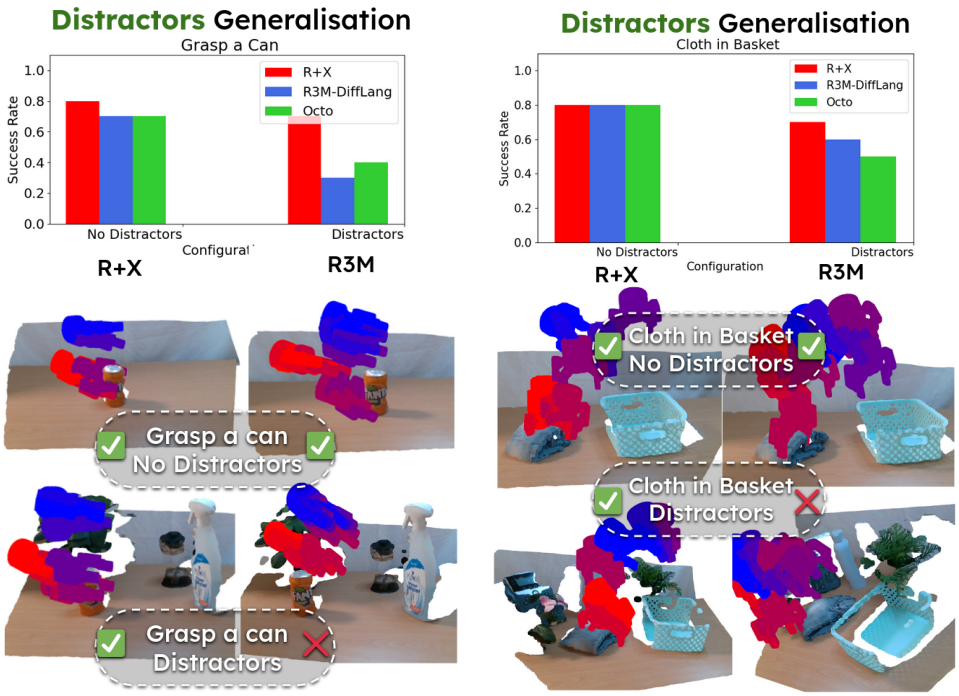

Figure 12: Examples of generalisation to distractors. Gripper moves from red to blue.

half the runs, while half are distractors free. We consider the task successful if the microwave door is completely closed (it has a magnetic system that keeps it closed once it is completely pushed in position).

- **Cloth in Basket:** We place a basket on the right side of the table in a 20cm by 20cm area, and randomise its orientation in a $[-20, 20]$ degrees range. We place a random cloth (same as for the washing machine task) on the left side of the table, randomising its position in a 25cm by 25cm area.We place random distractors in half the runs, while half are distractors free. We consider the task successful if the cloth is contained in the basket, i.e. if lifting the basket, the cloth is lifted as well without falling on the table.
- **Pick Up Phone:** We randomly place one out of two possible phones on the table, randomising its position in a 30cm by 30cm area, and its orientation in a $[-30, 30]$ degrees range. By using different phones, one of which unseen at training, we also test for object generalisation. We place random distractors in half the runs, while half are distractors free. We consider the task successful if the robot has grasped and lifted the phone from its base.
- **Grasp Can:** Same setting as for Grasp Beer. In this case, we also use other unseen cans at test time to test for object generalisation.
- **Turn On Light:** We place a light bulb and a socket on the table, randomising their positions independently in a 30cm by 30cm area, and the socket orientation in a $[-20, 20]$ degrees range. We place random distractors in half the runs, while half are distractors free. The task is successful if the robot can press the light bulb's plug entirely in the socket, turning on the light.

Further Investigation: What is the performance of R+X on unseen objects? In many of our tasks, as described before, we test R+X on unseen objects of the same categories, like unseen cans, kettles, clothes, etc. In this section, we perform a more detailed study on the performance of our method on seen and unseen objects. For the tasks that can be tested on unseen objects, we run 10 trajectories on objects seen in the human video, and 10 with unseen objects. Results in Figure 13 demonstrate that our method is robust to novel, unseen objects of the same category. This is thanks to the keypoint

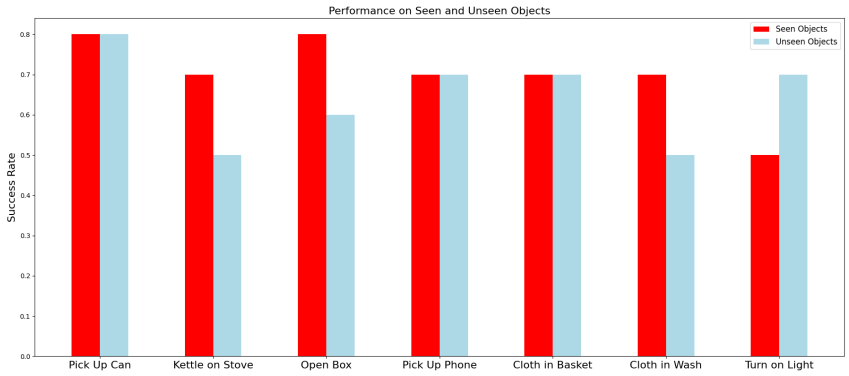

Figure 13: Success rate of R+X on seen and unseen objects from different tasks.

Figure 14: Examples of keypoints extracted for the same tasks, but with different views, settings, and target objects. Images are grouped in pairs column-wise. Keypoints are extracted between top and bottom images of each column.

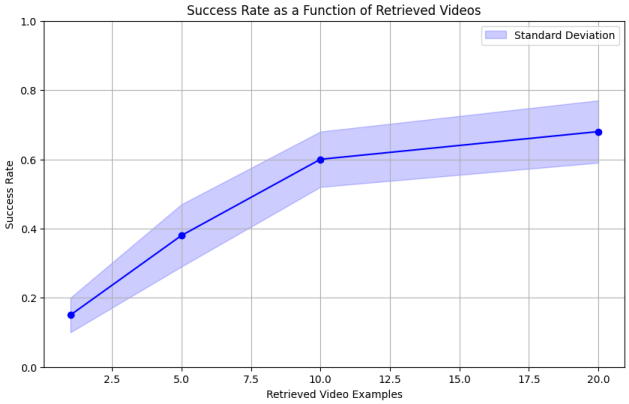

Figure 15: Success rate as a function of the amount of retrieved videos. We plot mean and standard deviation on 4 tasks, described below.

extraction pipeline, that can extract from images a list of semantically and geometrically meaningful keypoints, that transfer across objects. Examples of such keypoints can be observed in Figure 14.

How does the performance change with the number of retrieved videos? R+X is conditioned on a set of video clips depicting the requested tasks. How many videos clips need to be retrieved from the full human video to obtain strong performance? In this experiments, we manually limit the amount of videos that are retrieved for the "*pick up can, pick up telephone, cloth in basket and open box*" tasks, changing it between 1, 5, 10 and 20. We then run 10 test episodes for each of these values and compute the overall success rate. In Figure 15 we see how R+X can reach a strong performance also with as few as 10 retrieved videos.

Further Investigation: What are the main sources of difference in performance between R+X and a monolithic policy? Here we further investigate sources of differences in performance between R+X and the baselines, extending the experimental investigation of the main paper. In particular, here we focus on the presence of distractors.

Distractors Generalisation: One of the main differences between R+X and the baselines is that the former, after retrieving a set of video clips from $\mathcal{H}$ depicting the requested task, can extract a set of keypoints that are semantically and geometrically meaningful for the objects to interact with, as described in the main paper. These keypoints are generally robust to distractors, as they focus only on the DINO features that are common in all images, whereas distractors generally through out the human video $\mathcal{H}$ and the execution. The baselines, being monolithic policies, receive as input the observation as it is, and must learn to generalise to the presence of distractors during the long training phase.

To evaluate the effect of distractors on the scene, we test R+X and the baselines on two tasks, "*grasp a can*" and "*put the cloth in the basket*", emulating the experimental scenario of the "*Hard Generalisations*" sections. We run 10 runs without distractors, and 10 with, explicitly measuring the performance in the two cases. We can see in the results of Fig. 12 how R+X is more robust to the presence of random distractor objects, highlighting the advantage of extracting semantically and geometrically meaningful keypoints *after* having received a language command and having retrieved the corresponding video clips.

6.6 Further Details on Baselines:

6.6.1 Octo

One of baselines that we have compared against is a fine tuned version of Octo [30]. We have used the original code provided by the authors and adapted it for the specific input and output relevant to this work. More specifically, we designed the input of the model to be the RGB image of size 256×256 of the live observation, $\mathcal{O}_{live}$ and a language description of the task $\mathcal{L}$. The output, instead, is $\mathcal{J}_{live}$, the trajectory of desired hand joints 3D points to be translated to gripper poses as described before. Unlike KAT, these methods require a fixed length of the output trajectory. We therefore pad it to 40 times steps. The output is ultimately a tensor of dimension 480, which corresponds to a flattened tensor of dimension $(40, 4, 3)$. The 4 predicted hand joints are described in section 6.4 along with how they are mapped to robot actions.

Having set the input and output dimensions, as well as modalities, the model has been fine tuned starting from the provided checkpoint "octo-small-1.5" and it was trained until convergence.

6.6.2 R3M-DiffLang

The other baseline we have compared against has been the combination of R3M [29] and [31], with the inclusion of language conditioning. The resulting model leveraged the ResNet-50 variant of R3M to encode the 256×256 live observation RGB image $\mathcal{O}_{live}$ and Sentence-BERT [47] to encode the language description of the task $\mathcal{L}$. The resulting feature vectors for the image and language description was then concatenated and passed to a UNet [48] trained via diffusion following the original code of [31]. The output of the diffusion head has the same size as the output of Octo, as we pad the trajectory $\mathcal{J}_{live}$ to 40 timesteps. In order to take advantage of the pre-trained vision knowledge of R3M, the encoder has been initialised with the provided weights and kept frozen, as it

has been done for the language encoder. The diffusion policy head has been trained from scratch using our own dataset of trajectories until convergence was reached.