2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

RoboTAP: Tracking Arbitrary Points for Few-Shot Visual Imitation

Mel Vecerik[1,2,*], Carl Doersch[1,*], Yi Yang[1], Todor Davchev[1], Yusuf Aytar[1], Guangyao Zhou[1]
Raia Hadsell[1], Lourdes Agapito[2], Jon Scholz[1,*]

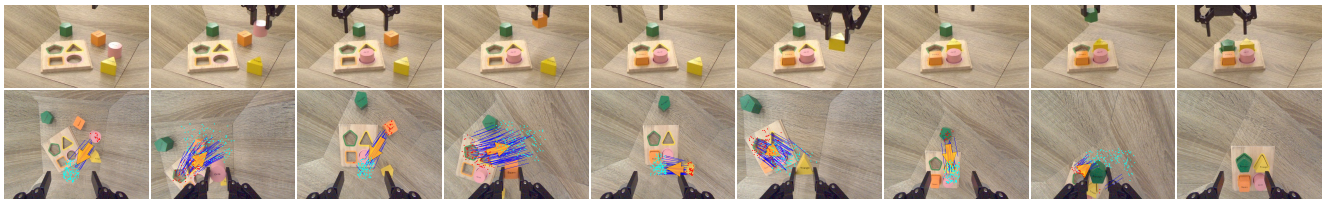

Fig. 1: An example of RoboTAP using points automatically selected from few demos (≤ 6) to define a long horizon behaviour. First row is illustrative, second row is what the agent sees. At every stage, the system identifies the current location of "active" points relevant to the stage (red). Given the goal locations for each point, from the demos (cyan), a desired motion for each point is produced (blue lines), and converted to a robot action using a generalized 4D visual-servoing primitive, which operates with arbitrary points.

***Abstract*— For robots to be useful outside labs and specialized factories we need a way to teach them new useful behaviors quickly. Current approaches lack either the generality to onboard new tasks without task-specific engineering, or else lack the data-efficiency to do so in an amount of time that enables practical use. In this work we explore *dense tracking* as a representational vehicle to allow faster and more general learning from demonstration. Our approach utilizes Track-Any-Point (TAP) models to isolate the relevant motion in a demonstration, and parameterize a low-level controller to reproduce this motion across changes in the scene configuration. We show this results in robust robot policies that can solve complex object-arrangement tasks such as shape-matching, stacking, and even full path-following tasks such as applying glue and sticking objects together, all from demonstrations that can be collected in minutes.**

I. INTRODUCTION

Imagine if you could track an arbitrary number of points in space, in any scene, through occlusions, motion, and deformation – how might it simplify the manipulation problem? Recent advancements in *dense-tracking* [1] have provided exactly this capability, but it has yet to be explored in a manipulation setting. In this paper we investigate dense-tracking as a perceptual primitive for manipulation, with the goal of producing a single system that can *quickly* solve a wide range of problems without task-specific engineering.

This objective is closely related to recent work on foundation-models for robotics [2], [3], which treats manipulation across embodiments and tasks as a large-scale sequence-modeling problem. These approaches are incredibly general and powerful, especially when utilizing models pre-trained on large multi-modal datasets [4], [5], but they tend to be data-hungry and expensive to train. We therefore move away from direct neural-network training and instead follow the recent trend towards "prompting" pretrained models. However, instead of an LLM, we explore a general-purpose point tracker as the pretrained model.

*Main contacts: {vec,doersch,jscholz}@google.com
[1]Google DeepMind
[2]Department of Computer Science at University College London
Project website: https://robotap.github.io

Our hypothesis is that much of the complexity in low-level manipulation can be reduced to three fundamental operations: (1) identifying **what** is relevant in the current frame, (2) identifying **where** it is, and (3) identifying **how** to move it in desired directions. We show that all three operations can be parameterized via dense-tracking, yielding a general formulation for manipulation that does not require task-specific engineering. Furthermore, we find that point tracks can provide an *interface* between these different operations, allowing us to factorize the problem into simple low-dimensional functions which, together, can express a large space of possible behaviors.

The behaviors of interest span a range of real-world problems involving precise multi-object rearrangement, *e.g.* pick-and-place, (high-clearance) insertion, and stacking. Our approach, RoboTAP, allows few-shot imitation learning of these behaviors in minutes, which we highlight via a task that involves grasping a glue-stick, applying it to a surface, mating the two parts, and placing them at a desired location. Solving this task requires over 1000 precise real-valued actions, and is messy and irreversible, which renders it infeasible for approaches requiring large-scale data-gathering.

For all of these tasks, RoboTAP automatically extracts the individual motions, the relevant points for each motion, goal locations for those points, and a generates a plan that can be tracked by a low-level visual servoing primitive. While currently less general than fully end-to-end approaches, our approach can be trained on as few as 4-6 demonstrations per task, does not require action-supervision, and effortlessly generalizes across clutter and object pose randomization.

Our main contributions are as follows: (1) RoboTAP, a formulation of the multi-task manipulation problem in terms of dense-tracking, (2) Concrete implementations of RoboTAP's *what*, *where*, and *how* problems in the form of visual-saliency, temporal-alignment, and visual-servoing, (3) A new dense-tracking dataset with ground-truth human annotations tailored for our tasks and evaluated on the TAP-Vid Benchmark focusing on Real-World Robotics Manipulation, and (4) Empirical results that characterize the success and

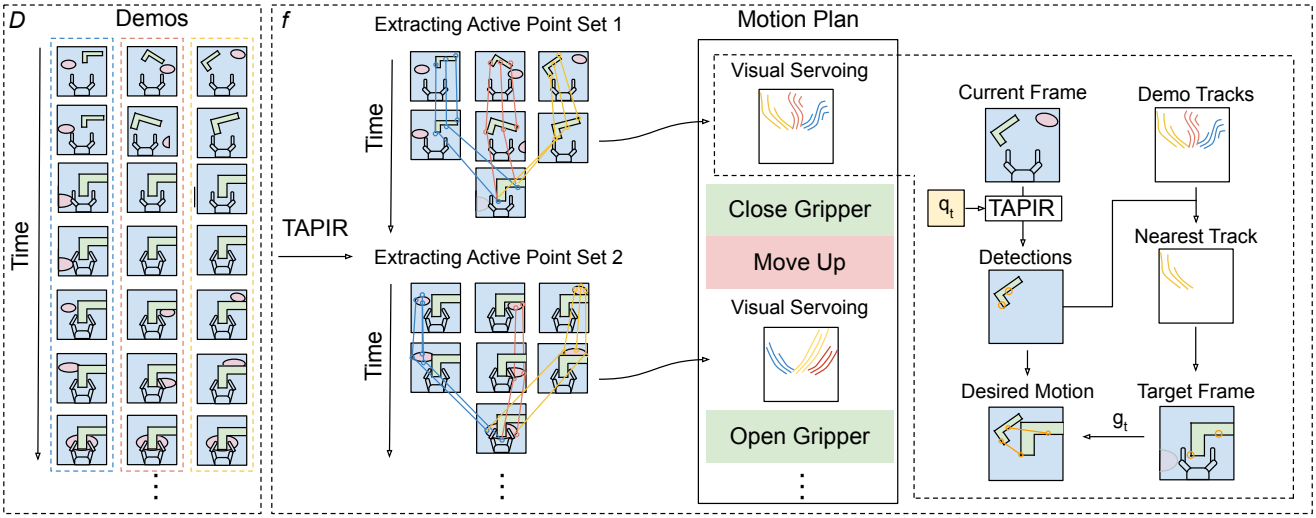

Fig. 2: Here we describe the core of the RoboTAP approach. Given a set of demonstrations D, we first track densely using TAPIR. Next, we temporally segment the demonstrations into stages, and then automatically discover the *active point set* q for each stage, which covers the object whose motion is relevant at that stage of the action. We then form a motion plan that can be executed on the robot, which consists of stages of servoing to imitate visual motions, and basic motor primitives like closing and opening the gripper. Visual servoing is accomplished by detecting points q using TAPIR, finding the nearest demonstration which shows how those points should move, and finding a single nearby frame that can be used as a motion target. The displacement between the points in the target frame (g) and the online TAPIR detections is used as a motion target for classical visual servoing, yielding surprisingly complex and robust behavior.

failure modes of RoboTAP on a range of manipulation tasks involving precise multi-body rearrangement, deformable objects, and irreversible-actions.

II. Related work

A. Vision-based Manipulation

1) End-to-End Approaches: Regressing actions directly from images in an end-to-end manner is one established way for utilising visual inputs [3], [6], [7], [8], [9]. While in theory this allows for creation of arbitrarily complex policies in practice this often requires a large amount of data including the target objects or classes.

2) Pose-based Approaches: Another approach is to define a policy on top of object poses regressed from images [10], [11], [12], [13], [14]. Pose is a strong signal for object state, but is not well-defined for deformable or symmetric objects, and is difficult and time-consuming to supervise accurately and generally.

3) Keypoint-based Approaches: Within RoboTAP, point locations are detected from an RGB camera and converted to arm motions in a visual feedback-loop, which is commonly referred to as image-based visual servoing [15], [16]. Existing autonomous visual servoing techniques typically rely on hand-designed target detectors or confidence maps [17], [18], [19], which lack the generality required for few-shot imitation in arbitrary scenes.

A closely-related approach is to define a small sparse set of keypoints for a given object *class* [20], [21], [22]. Choosing a different point set however requires a retraining of the whole model. Methods such as TACK [23] or DON [24] generalize this approach by learning an embedding for any point on an *object* in the image. This still falls short of representing arbitrary points in scenes, and must be retrained in order to adapt to new class of objects.

As a result, keypoint-based approaches can perform very well in specific settings, but act as an extra barrier to deploy robotic systems on novel tasks. In this work, we aim to demonstrate that recent advances on point tracking models such as [25] are good enough to act as a sole perception model for both motion tracking, cross scene correspondence and segmentation. This system doesn't use any depth information during training or evaluation and uses only a single non-calibrated gripper mounted camera.

Lastly, the "what/where" factorization we explore in this paper has several precedents in robotics, *e.g.* [26], and has been more broadly explored in neuroscience [27], [28].

B. Point Tracking in Computer Vision

Within computer vision there have been approaches which extract points from any scene without any pretraining. For example non-learned descriptors such as SIFT [29] or ORB [30] require no class specific training, but they produce many spurious matches which cannot be easily filtered on non-rigid scenes, making them difficult to use as a target for visual servoing. Recently there have been several advances in high-performance long-horizon visual tracking such as OmniMotion [31], PIPs [32], TAP-Net [1] or TAPIR [25], which provide high-quality tracks with very few outliers. This means that Euclidean distance in point space, with no outlier removal, can serve as a reliable metric of whether two spatial configurations are similar, or whether two motions are similar. While all of these approaches are made to run on whole videos, we note that the setup (especially TAPIR) can be modified to run online on a frame-by-frame basis, making it suitable for robotics applications.

III. Approach

Recall that we seek to define a *single system* that can be instructed to solve as many manipulation tasks as pos-

sible without requiring any per-task engineering or technical knowledge. To enable this instruction mechanism we begin by considering the class of semi-parametric policies $\pi(s_t, D)$, in which the policy has access to a "context dataset" D at test-time, *e.g.* demonstration trajectories.

Conceptually, RoboTAP can be viewed as a factorization of π in which we extract a sufficient statistic for the current action from the full "context" D, according to the current state s_t, *i.e.* $\pi(s_t, D) = \pi(TAP(s_t, q_t), g_t) = \pi(p_t, o_t, g_t)$, where $q_t \in \mathbb{R}^{n,c}$ represents the c-dimensional descriptors for n points (a.k.a. the query), $g_t \in Z^{n,2}$ represents the target locations for those points in the image. Given a query q_t and an image in the current state s_t, *TAP* represents a dense point-tracker that outputs detections $p_t \in Z^{n,2}$ and an occlusion probability o_t for each point.

The quantities q_t and g_t represent the "what" and "where", respectively, of the current action, and are extracted from D via a procedure $g_t, q_t = f(s_t, D)$, described in Section III-A. For simplicity, in this paper we implement f as a one-time plan that summarizes the *objectives* common across D. An overview of this procedure and it's connection to the control is illustrated in Fig. 2. The high-level outline for our overall solution is as follows:

1) Sample large number of query points from D and track those points across all trajectories using TAPIR.
2) Segment the trajectories into phases, and discover a descriptor-set q_t for each phase which characterize the motion.
3) Pack the sequence of q_t and corresponding trajectory slices (representing the goals) into a "motion-plan".
4) Execute this motion plan using the low-level controller in a sequential fashion, advancing stages based on a simple final-error criterion.

A. Temporal and Spatial Decomposition

The first core step of the RoboTAP approach, is to extract important motion from demonstrations in order to construct a motion-plan. This seen in Fig. 2. At a high level, the problem we must solve is how to decompose K demonstrations into a N primitive motions, where each motion can be parameterized by a visual servoing controller with a specific set of query features, a.k.a. "active points". We solve this problem in two steps. First we temporally segment the demonstrations in to a set of phases such that the (unknown) active points can be assumed to be constant. Second we look across the demos for each phase for the points with consistent motion, and save their query features as the *active points* for that phase.

Temporal alignment. Temporal alignment has been well-studied [33], [34], [35], but for simplicity, we rely on the assumption that our tasks consist of grasping and releasing objects or making contacts, which means that temporal segments are trivially obtained by thresholding gripper actions and forces. The overall number of phases is determined by the number of times the gripper opens and closes.

Active Point Selection. Given a temporal segment, identifying active-points involves asking what points $q_t \in Q$ the low-level controller π_l should move in order to generate motions that accomplish a similar result as observed in the segment. Our criteria for point selection are illustrated in Fig. 3. The key insight is that the active points may begin at diverse locations, but for goal-directed behavior they tend to *end up* the same place across all demos at the end of the relevant segment. Therefore, we select points which end at the same place at the end of the demo segment (relative to the camera), and remove points which don't move across the segment (i.e., the gripper and any already grasped object).

Fig. 3: Active point selection. We exploit the funneling-nature of control to identify relevant points based on their variance, and remove the gripper by filtering static points. Remaining points are used to vote on the *motion cluster*, which we use to sample 128 points throughout the motion-segment to serve as the salient features for that step of the motion plan.

While this alone can identify many of the desired active points, not all of the useful points are guaranteed to be visible at the end of each motion segment, either due to occlusion or simply failures by the detector; furthermore, there may be inconsistencies due to imprecise demos that make it difficult to set thresholds on whether a point is moving. Therefore, we recast the active point discovery as votes on which *object* (or object-part) is being manipulated, which we substantially improves robustness. However, this means we must perform unsupervised object discovery from the demos, a classically difficult problem which we find is rendered surprisingly reliable given robust dense point tracks. We then combine these heuristics with a voting-based scheme, whereby we select the clusters which contains the most points which are "active" according to the first two criteria.

Clustering. There are numerous approaches to object-based clustering, ranging from semantic segmentation [36] to generative modeling [37], but for simplicity, we use motion estimates extracted from TAPIR, since this means we do not require semantic labels, and we find it can work reliably from a remarkably small amount of data.

We first randomly select many points from the image, and track all of them using our online TAPIR model. We assume that all of the points belong to one of several approximately-rigid objects in the scene, and can therefore be explained by a set of 3D motions followed by reprojection (note that we model camera motion as a motion of all objects in the scene). Let $p_{i,t}$ be the TAPIR-predicted location for point i at time t in the demos (for simplicity, t indexes both time and demos, i.e., we concatenate all demos into a single long sequence), and $v_{i,t} \in \{0, 1\}$ be a thresholded version of occlusion probability o_t: 1 if o_t is less than 0.5, 0 otherwise. We assume there are K different rigid objects in the scene. The model infers a 3D location $P_{i,k} \in \mathbb{R}^3$ for i'th point in the k'th object (initially, the model does not know which object the point should be assigned to). It also proposes a rigid 3D transformation for each object at each time $A_{t,k} \in SE(3)$, represented as a 3×4 matrix. We then optimize the

transforms and the 3D points to minimize the simple squared error of the reprojection $R(x) = [x[0]/x[2], x[1]/x[2]]$ as follows:

$$\arg\min_{A,P} \min_{k} \sum_{i,t} v_{i,t} \|R(A_{t,k} P_{i,k}) - p_{i,t}\|^2$$

The simplicity of this equation is somewhat remarkable, as it is essentially standard bundle adjustment [38], but *we do not model outliers*. Outlier rejection is a critical component of prior structure-from-motion methods as they are typically based on descriptors like SIFT [39], which have a high proportion of extreme errors.

We parameterize both $P_{i,k}$ and $A_{t,k}$ using neural networks, which aim to capture the inductive biases that points nearby in 2D space, and also frames nearby in time, should have similar 3D configurations. Specifically, $P_{i,k} = \boldsymbol{P}(p_i, o_i|\theta_1)_k$, where θ_1 parameterizes the neural network $\boldsymbol{P}$ which outputs a $k \times 3$ matrix, and $A_{t,k} = \boldsymbol{A}(\phi_t|\theta_2)_k$, where ϕ_t is a temporally-smooth learned descriptor for frame t, θ_2 is a neural network parameter for neural network $\boldsymbol{A}$ which outputs a $k \times 3 \times 4$ tensor representing rigid transforms.

The above optimization is, unsurprisingly, somewhat difficult due to local minima; we find that these local minima can be avoided by *splitting* clusters during optimization. For details on this and the neural networks, see our wepbpage.

Given a clustering, we select clusters by voting. Every previously selected active point casts a vote for a cluster and we merge clusters with largest number of votes. To further avoid selecting points from the gripper we compute average point movement for each cluster and remove clusters where this movement is below a threshold. Finally, we discard any points are not visible in the current phase, or which are not nearby other points on most frames, as these are likely to be tracking failures. For details, see our wepbpage.

B. Robot controller

The foundation of RoboTAP is a general-purpose controller that can align points in a scene. To define this controller we consider a set of image points, with current locations p_t, and goal locations g_t as derived above. Note that p_t must be detected *online* at test time, which TAPIR was not designed for; however, we find that its temporal convolutions can be replaced with *causal* convolutions with minimal loss in accuracy to produce an online model. The objective of this controller is to move the gripper with velocity v_t (4 DoF: $\dot{x}, \dot{y}, \dot{z}, \dot{rz}$) such that the points will move towards the desired location, while also being robust to noise and occlusions. To do this we need an estimate of how the points will move if the gripper moves in a certain direction. For a wrist-camera this is simply the image Jacobian $\frac{dp}{d\xi} \in \mathbb{R}^{n,2,4}$, where n denotes the number of points and ξ contains the relevant dimensions (x, y, z, rz) of the gripper pose.

In theory, to obtain the correct image Jacobian we need to consider the camera intrinsics and the extrinsics. However in our case we only require that it points in the correct direction, and we then tune the rotation and translation gains of the controller for stability. To demonstrate this in all of our experiments we assume vertical field of view of 90° and unit depth which leads the following image Jacobian for a single point:

$$J = \begin{bmatrix} 1 & 0 & -u & -v \\ 0 & 1 & -v & u \end{bmatrix} \quad (1)$$

Where the columns correspond to end-effector-frame linear and angular velocity of the gripper, and u, v are normalized image coordinates such that vertical dimension is within [-1, 1]. Using this Jacobian, we then compute the gripper motion that would by minimize the L2 error between the current detections p and goal locations g under the linear approximation, following standard visual servoing. Because the error is typically dominated by translation, applying the Jacobian naively would result in the controller explaining translation via rotation and scaling; therefore, we use Gram-Schmidt orthogonalization [40] to eliminate the average translation before computing the Jacobian with respect to rotation and scaling.

If TAPIR provided perfect point locations, the above algorithm would work effectively without modification. However, for precise tasks, TAPIR errors can still lead to two specific failure modes. First, outliers from TAPIR can overwhelm the controller when true errors are low. To deal with this, we leverage TAPIR's uncertainty outputs, and only use points which are confidently predicted for both the current frame and in the demo. Second, noisy detections introduce a statistical bias towards minimizing the spread of the point cloud in order to reduce unexplainable errors. This presents itself as a bias towards moving the camera further back. Therefore, we modify the controller by leveraging the relation between the problems of aligning p to g and the inverse alignment of g to p. This changes the z-axis update to perform variance matching rather than directly optimizing the visual servoing objective. To further reduce variance near the end of each servoing stage (when the demo locations are assumed to agree), we use an average of point locations across all demos as a target to further reduce variance, while we use a single demo as a target earlier in each stage. See our webpage for details.

IV. EVALUATION

In this section, we aim to demonstrate the strengths and limitations of our system and its components. The capabilities of RoboTAP overall are directly linked to the precision of TAPIR itself on the domain we are interested in. Therefore to evaluate and advance its abilities we introduce a new point tracking dataset focused on robotics settings.

Following this, we show evaluations of the full RoboTAP system. In particular, we show that RoboTAP can tackle complex tasks with many stages from just a handful of demos, can deal with start and end poses significantly different from those in the demos, can both place objects and follow trajectories, has strong invariance to distractors in the workspace, and can deal with non-rigid objects.

We choose a variety of tasks involving placement, insertion, and gluing, that are representative of real-world assembly problems. While all of the demonstrations were gathered without occlusions and distractor objects we explore how the ability of RoboTAP is affected when these are

Dataset	TAP-Vid-RGB-Stacking	RoboTAP
# Videos	50	265
# Points	30	43.7
# Frames	250	271.9
# Human Clicks	N/A	8.5
Sim/Real	Sim	Real
Eval resolution	256x256	256x256

TABLE I: Statistics of RoboTAP dataset. Comparing to existing point tracking dataset on Robotics: TAP-Vid-RGB-Stacking, RoboTAP dataset produces realistic challenging videos with more point annotations and longer video durations. Note, # Points is average number of annotated points per video; # Frames is the average number of frames per video. # Human Clicks is the average number of human clicks per video.

present. Finally we also study the servoing precision using a calibrated real-world setup.

A. Robotics TAP-Vid Dataset addition

The ability to track and relate points in the scene is what enables RoboTAP to generalize to novel scenes and poses. To enable progress in this core capability we introduce an addition to the previous TAP-Vid benchmark in a form of a new dataset which focuses specifically on robotic manipulation.

Specifically we collect 265 real world robotics manipulation videos. The data source mainly comes from teleoperated episdoes in the public DeepMind robotics videos [41], [21]. We annotate each video with human groundtruth point trajectories, following the same instruction in the previous TAP-Vid dataset [1], with 5 points per object and 5 points on the background. To make the evaluation closer to the realistic manipulation setup, we sampled the videos from different camera viewpoints (i.e. basket view, wrist view) where basket view is static and wrist view moves along with the gripper.

Table I shows the overall statistics of the newly collected RoboTAP dataset. Comparing to the existing simulated TAP-Vid-RGB-Stacking dataset, it contains more videos, more points, and more frames on average. For more information, examples of videos and quantitative evaluation of current models as well as the modified online TAPIR model on this dataset please see our website.

B. Robot Setup

To run our system in the real world we use the Franka Panda Emika robot in the impedance control mode, 2f-85 Robotiq gripper and FT 300 Force Torque Sensor. The control signal was sent at 10Hz and interpreted as velocity of the impedance controller setpoint. To gather images we used a Basler Dart daA1280-54ucm at a resolution of 640x480 pixels mounted to the wrist using a custom 3D printed adapter. Our camera images were undistorted, but we did not use intrinsic or extrinsic calibration of our camera.

To collect the demonstrations we move the robot via the Franka Panda cuff. RoboTAP is compatible with kinesthetic teaching because, unlike much other imitation learning work, it does not require access to actions beyond gripper opening/closing state. During teaching we record robot positions,

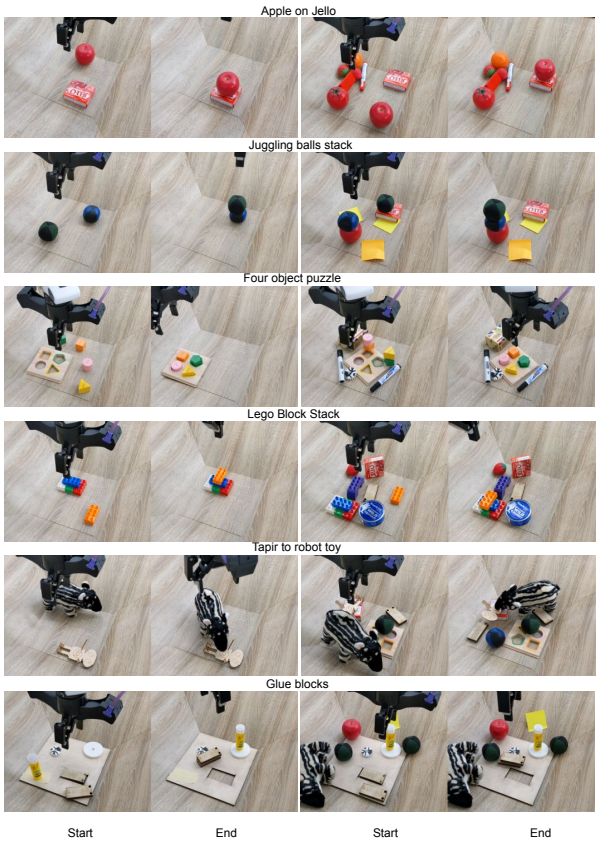

Fig. 4: Examples of successfully solved real robot tasks. In order to challenge the system and demonstrate its robustness we show its performance on scenes with clutter, distracting objects and partial occlusions.

Task name	Description	# Demos
Apple on jello	Pick up the apple and place it on jello.	5
Juggling ball stack	Pick up a green juggling ball and place it on a blue ball.	3
Four object puzzle	Place four wooden objects into their cutouts on a puzzle board.	6
Lego Stack	Put orange LEGO brick on top of a blue brick and push them together.	4
Tapir to robot	Pick up the plush tapir toy and place it next to the wooden robot.	4
Gluing	Glue the 2 wooden block together and place them beside the white gear.	5
Gear on grid	Pick up the white gear and place it on the grid.	6
Four object stack	Stack 4 wooden objects on top of each other.	4
Pass the butter	Pick up a butter and place it in human's hand.	5

TABLE II: Tasks details.

forces from a wrist force-torque sensor, and wrist camera videos at 10Hz. We used a keyboard to open and close the gripper. It takes about 30 seconds to record a demonstration for a pick and place task, 1 minute for the gluing task and up to 2 minutes for the 4-object insertion task.

We used a total of 9 tasks described in Table II, some of which are illustrated in Figure 4 - see our website for a full list of task videos. When running this controller we have noticed repeatable patterns of cases where the controller succeeds and fails. For most tasks we have attempted, we observe robust performance provided the algorithm's basic assumptions are met, yet we have seen it fail in a few specific cases. Therefore we believe that a simple success metric would not appropriately capture its behaviour and we aim to provide other avenues to express its performance.

In all cases we aimed to demonstrate the performance of the controller in a clean environment, and then show how performance degrades with clutter and occlusions. The

Fig. 5: Examples of states where the system failed to reach the desired final state. Tasks which require sub 5mm precision cannot always be reliably solved (*e.g.* shape-matching). In addition, our use of a purely-visual control paradigm makes it difficult to solve tasks that require reasoning over visual and force modalities simultaneously (*e.g.* lego part-mating). Lastly, our controller is unable to reason about the validity of a motion-plan at runtime, which can lead to failures if certain motions are invalid (*e.g.* the bunny's ear is covering the jello too much to place the apple).

only tasks where we have not observed reliable performance were the LEGO stack, in which the controller lacked the precision to stack the bricks, and the 4 object stack, where the compounding of the errors often lead the last object to be dropped in the wrong location. Overall we observed 4 factors which caused failures: *(1) Occlusions* cause failures when the active points cannot be seen (*e.g.* due to the gripper occlusions). *(2) Scale changes* cause failures when the gripper moves too far away and none of the currently-tracked features can be matched because the objects are too small. *(3) Distractors* can cause failures when the scene contains a similar object to the essential object such as a red apple vs a tomato. *(4) Collisions* can happen with the gripper or currently-held object as the controller does not have a way to perceive clutter. However in absence of these cases, we observed robustness to novel arrangements and even dynamic changes of the scene while the controller is being executed.

C. Real world precision

Previous experiments demonstrated the qualitative behavior of several long-horizon manipulation tasks. We conclude by evaluating the precision of our visual-servoing controller quantitatively. For this we take inspiration from position-repeatability analyses performed on commercial industrial robot arms. We gathered 6 demonstrations of putting a textured white plastic gear on a square of graph paper, which allowed us to precisely compute of the placement error using OpenCV [42]. For the demonstrations we used 3 different configurations of the white gear and the target pattern. The target had a cross in the middle which allowed us to extract precise location by taking pictures of the centre of the gear.

Then we ran the controller 30 times for each of 3 novel goal locations. First goal location, *Near*, was positioned in the middle between the locations which were used for demonstrations. The second one, *Far*, was on the opposite side of our workspace and the last one, *Rotated*, was on the side but rotated by 90 degrees. Notably the last 2 goal positions are outside of the distribution of the demonstrations.

We can see the results of this evaluation in Fig. 6. Based on the figure we can see that the error of the controller is

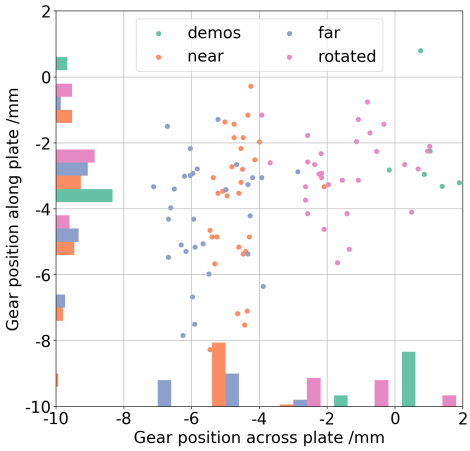

Fig. 6: Evaluation of the precision of controller. We designed a pick-and-place task which allowed us to easily measure exact final location of the placed object relative to movable target. We gathered 6 demonstrations and ran RoboTAP 30 times for 3 different locations of the placement target. The figure shows the locations of where the object was placed in each of the trials. When the goal was located between the demonstrated position the distribution is similar to what was observed during the demonstrations ($<$ 4mm). We can see a slight decrease in precision when the target was moved to a novel location or rotated by 90 degrees ($<$ 1cm).

comparable to the error seen across the demonstrations. The error along the plate, orthogonal to the grasp direction is on the order of several milimeters. Based on these results we can see that the controller in its current form would not be suitable for sub-mm insertion tasks, however we have not attempted to extensively tune it for this purpose.

V. CONCLUSION

We presented RoboTAP, a manipulation system that can solve novel visuomotor tasks from just a few minutes of robot interaction. RoboTAP does not require any task-specific training or neural-network fine-tuning. Thanks largely to the generality of TAP, we found that adding new tasks (including tuning hyper-parameters) took minutes, which is orders of magnitude faster than any manipulation system we are familiar with. We believe that this capability could be useful for large-scale autonomous data-gathering, and perhaps as a solution real-world tasks in its own right. RoboTAP is most useful in scenarios where quick teaching of visuomotor skills is required, and where it is easy to demonstrate the desired behavior a few times.

There are several important limitations of RoboTAP. First, the low-level controller is purely visual, which precludes complex motion planning or force-control behavior. Second, we currently compute the motion plan once and execute it without re-planning, which could fail if individual behaviors fail or if the environment changes unexpectedly.

Several of the ideas in RoboTAP, *e.g.* explicit spatial representation and short-horizon visuo-motor control, could also be applicable in more general settings. In the future we would like to explore whether RoboTAP models and insights can be combined with larger-scale end-to-end models to increase their efficiency and interpretability.

REFERENCES

[1] C. Doersch, A. Gupta, L. Markeeva, A. Recasens, L. Smaira, Y. Aytar, J. Carreira, A. Zisserman, and Y. Yang, "Tap-vid: A benchmark for tracking any point in a video," *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2022.

[2] A. Michael et al, "Do as i can and not as i say: Grounding language in robotic affordances," in *arXiv:2204.01691*, 2022.

[3] K. Bousmalis, G. Vezzani, R. Dushyant, C. Devin, A. X. Lee, M. Bauza, T. Davchev, Y. Zhou, and A. Gupta et al, "Robocat: A self-improving foundation agent for robotic manipulation," *arXiv:2306.11706*, 2023.

[4] A. Michael et al, "RT-1: Robotics transformer for real-world control at scale," *arXiv:2212.06817*, 2022.

[5] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *ICRA*. IEEE, 2023, pp. 9493–9500.

[6] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *arXiv:1806.10293*, 2018.

[7] L. Yen-Chen, A. Zeng, S. Song, P. Isola, and T.-Y. Lin, "Learning to see before learning to act: Visual pre-training for manipulation," in *ICRA*. IEEE, 2020, pp. 7286–7293.

[8] M. A. Lee, Y. Zhu, P. Zachares, M. Tan, K. Srinivasan, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making sense of vision and touch: Learning multimodal representations for contact-rich tasks," *T:RO*, vol. 36, no. 3, pp. 582–596, 2020.

[9] J. Luo, O. Sushkov, R. Pevceviciute, W. Lian, C. Su, M. Vecerik, N. Ye, S. Schaal, and J. Scholz, "Robust multi-modal policies for industrial assembly via reinforcement learning and demonstrations: A large-scale study," *arXiv:2103.11512*, 2021.

[10] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis, "Single image 3d object detection and pose estimation for grasping," in *ICRA*. IEEE, 2014, pp. 3936–3943.

[11] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv:1711.00199*, 2017.

[12] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox, "Self-supervised 6d object pose estimation for robot manipulation," in *ICRA*. IEEE, 2020, pp. 3665–3671.

[13] K. Chen, R. Cao, S. James, Y. Li, Y.-H. Liu, P. Abbeel, and Q. Dou, "Sim-to-real 6d object pose estimation via iterative self-training for robotic bin picking," in *ECCV*. Springer, 2022, pp. 533–550.

[14] H. Chen, F. Manhardt, N. Navab, and B. Busam, "Texpose: Neural texture learning for self-supervised 6d object pose estimation," in *CVPR*. IEEE/CVF, 2023, pp. 4841–4852.

[15] J. Hill, "Real time control of a robot with a mobile camera," in *Proc. 9th Int. Symp. on Industrial Robots*, 1979, pp. 233–245.

[16] J. Pomares, "Visual servoing in robotics," p. 1298, 2019.

[17] N. Vahrenkamp, S. Wieland, P. Azad, D. Gonzalez, T. Asfour, and R. Dillmann, "Visual servoing for humanoid grasping and manipulation tasks," in *ICHR*. IEEE-RAS, 2008, pp. 406–412.

[18] D. Kragic, H. I. Christensen, *et al.*, "Survey on visual servoing for manipulation," *Computational Vision and Active Perception Laboratory, Fiskartorpsv*, vol. 15, p. 2002, 2002.

[19] E. G. Ribeiro, R. de Queiroz Mendes, and V. Grassi Jr, "Real-time deep learning approach to visual servo control and grasp detection for autonomous robotic manipulation," *RAS*, vol. 139, p. 103757, 2021.

[20] L. Manuelli, W. Gao, P. Florence, and R. Tedrake, "kpam: Keypoint affordances for category-level robotic manipulation," in *The International Symposium of Robotics Research*. Springer, 2019, pp. 132–157.

[21] M. Vecerik, J.-B. Regli, O. Sushkov, D. Barker, R. Pevceviciute, T. Rothörl, C. Schuster, R. Hadsell, L. Agapito, and J. Scholz, "S3k: Self-supervised semantic keypoints for robotic manipulation via multi-view consistency," *Conference on Robotic Learning (CoRL)*, 2020.

[22] N. Das, S. Bechtle, T. Davchev, D. Jayaraman, A. Rai, and F. Meier, "Model-based inverse reinforcement learning from visual demonstrations," in *CoRL*. PMLR, 2021, pp. 1930–1942.

[23] M. Vecerik, J. Kay, R. Hadsell, L. Agapito, and J. Scholz, "Few-shot keypoint detection as task adaptation via latent embeddings," in *ICRA*. IEEE, 2022, pp. 1251–1257.

[24] P. R. Florence, L. Manuelli, and R. Tedrake, "Dense object nets: Learning dense visual object descriptors by and for robotic manipulation," *arXiv:1806.08756*, 2018.

[25] 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," *arXiv:2306.08637*, 2023.

[26] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *CoRL*. PMLR, 2022.

[27] M. A. Goodale, "Visuomotor control: Where does vision end and action begin?" *Current Biology*, vol. 8, no. 14, pp. R489–R491, 1998.

[28] D. Milner and M. Goodale, *The visual brain in action*. OUP Oxford, 2006, vol. 27.

[29] D. G. Lowe, "Object recognition from local scale-invariant features," in *ICCV*, vol. 2. Ieee, 1999, pp. 1150–1157.

[30] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *ICCV*. Ieee, 2011, pp. 2564–2571.

[31] Q. Wang, Y.-Y. Chang, R. Cai, Z. Li, B. Hariharan, A. Holynski, and N. Snavely, "Tracking everything everywhere all at once," *arXiv:2306.05422*, 2023.

[32] A. W. Harley, Z. Fang, and K. Fragkiadaki, "Particle video revisited: Tracking through occlusions using point trajectories," in *ECCV*. Springer, 2022, pp. 59–75.

[33] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman, "Temporal cycle-consistency learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 1801–1810.

[34] T. Davchev, O. Sushkov, J.-B. Regli, S. Schaal, Y. Aytar, M. Wulfmeier, and J. Scholz, "Wish you were here: Hindsight goal selection for long-horizon dexterous manipulation," *ICLR*, 2022.

[35] K. Zakka, A. Zeng, P. Florence, J. Tompson, J. Bohg, and D. Dwibedi, "Xirl: Cross-embodiment inverse reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 537–546.

[36] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*. IEEE, 2015, pp. 3431–3440.

[37] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf, "Object-centric learning with slot attention," *NeurIPS*, pp. 11 525–11 538, 2020.

[38] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu*. Springer, 2000, pp. 298–372.

[39] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, pp. 91–110, 2004.

[40] Å. Björck, "Solving linear least squares problems by gram-schmidt orthogonalization," *BIT Numerical Mathematics*, vol. 7, no. 1, pp. 1–21, 1967.

[41] S. Cabi, S. G. Colmenarejo, A. Novikov, K. Konyushkova, S. Reed, R. Jeong, K. Zolna, Y. Aytar, D. Budden, M. Vecerik, *et al.*, "Scaling data-driven robotics with reward sketching and batch reinforcement learning," *arXiv preprint arXiv:1909.12200*, 2019.

[42] G. Bradski, "The opencv library." *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 25, no. 11, pp. 120–123, 2000.