Recognising Affordances in Predicted Futures to Plan With Consideration of Non-Canonical Affordance Effects

Solvi Arnold, Mami Kuroishi, Rin Karashima, Tadashi Adachi, and Kimitoshi Yamazaki, *Member, IEEE*

***Abstract*—We propose a novel system for action sequence planning based on a combination of affordance recognition and a neural forward model predicting the effects of affordance execution. By performing affordance recognition on predicted futures, we avoid reliance on explicit affordance effect definitions for multi-step planning. Because the system learns affordance effects from experience data, the system can foresee not just the canonical effects of an affordance, but also situation-specific side-effects. This allows the system to avoid planning failures due to such non-canonical effects, and makes it possible to exploit non-canonical effects for realising a given goal. We evaluate the system in simulation, on a set of test tasks that require consideration of canonical and non-canonical affordance effects.**

***Index Terms*—Affordances, cognitive control architectures, deep learning methods, manipulation planning, predictive modelling.**

I. Introduction

THE concept of affordances, first introduced by Gibson [1] in the field of psychology, has found application in various areas of robotics [2], [3], [4]. Numerous conceptualisations exist, but affordances can broadly be defined as the action opportunities arising from the combination of an agent's action capabilities with the environment the agent is placed in. Affordances appear to play a central role in structuring high-level behaviour in humans and other animals. Key areas for this work are affordance recognition and affordance-based planning.

Within the area of affordance recognition, the introduction of neural networks (NNs) has produced substantial progress over the past years. In [5] and [6], object recognition-style architectures are applied to the problem of detecting affordances in images. In [7], NNs are used to extract object features that inform object manipulation. CLIPort [8] finds affordances defined as motion start and end points in various scenes on basis of natural language instructions.

Manuscript received 17 June 2022; accepted 10 January 2023. Date of publication 24 January 2023; date of current version 2 February 2023. This letter was recommended for publication by Associate Editor S. Chernova and Editor D. Kulic upon evaluation of the reviewers' comments. *(Corresponding author: Solvi Arnold.)*

Solvi Arnold and Kimitoshi Yamazaki are with the Department of Mechanical Systems Engineering, Faculty of Engineering, Shinshu University, Nagano 380-8553, Japan (e-mail: s_arnold@shinshu-u.ac.jp; kyamazaki@shinshu-u.ac.jp).

Mami Kuroishi, Rin Karashima, and Tadashi Adachi are with EPSON AVASYS, Ueda 386-1214, Japan (e-mail: kuroishi.mami@exc.epson.co.jp; karashima.rin@exc.epson.co.jp; adachi.tadashi2@exc.epson.co.jp).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3239308, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3239308

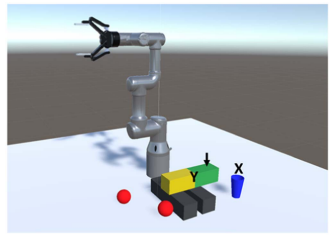

Fig. 1. Snapshot of the simulated environment, showing the virtual UR3 robot and task objects. Letters and arrow marker added.

Early approaches in planning descend from Situation Calculus [9], and employ grammars to describe action effects and preconditions [10]. The grammatical approach is effective if the task environment can be fully formalised, but hard to adapt to task domains with complex physical effects. This makes rule-based approaches notoriously brittle. Consider the task of stacking objects. We may specify the *canonical* effect of placing one object on another as follows:

$$PLACE\_ON\,(X,Y) \rightarrow ON\,(Y,X) \tag{1}$$

This rule may fail to hold for an instance as shown in Fig. 1. Object X can be placed on Y at the arrow marker, but (depending on the relative weights of X and Y) this may not result in X sitting atop Y. We can precondition the rule on stability of the object arrangement, but in practice this implies importing a substantial amount of physics into the precondition check, which quickly makes planning computationally infeasible.

The need to define complex physical preconditions on affordance effects may be avoidable by learning to predict action effects instead. In [11], [12], affordance effects in an object-stacking scenario are learned in a rule-based format for symbolic planning. However, effective use of fine quantitative state features for affordance effect prediction requires that prediction is learned at finer granularity than symbolic rules (learned or given) can provide. In early work in this direction under the nomer of *internal rehearsal* [13], action effects for a traversability affordance are learned using a Gaussian Mixture Model in a goal-specific manner.

Outside the context of affordances, physical prediction has been explored extensively in recent years. In [14], NNs are trained to predict stability (and block trajectories) for block towers, and [15] predicts object motion from real-world images. Models with action input are found in [16] (predictive models for billiard) and [17] (3D rigid body motion). The prediction NN in the present work fits in this line of inquiry.

NN approaches for affordance-based action planning remain scarce, but a notable example is found in *Deep Affordance Foresight* (DAF), which proposes a neural latent dynamics model for predicting future affordance availability [18]. However, future states are only represented in latent form, necessitating task-specific learning to associate states and goals. A second example is found in [19], which combines symbolic goal and state descriptors with learned transition models. However, visual input is masked to isolate target objects, discarding environmental context that may be relevant for predicting affordance effects. Task-specific learning is used, but shown to be accelerated by goal-agnostic pre-training. Learned skill effects are also found in [20], which focuses on incremental expansion of the skill repertoire. In contrast to the present work, states are represented in a manually designed analytical format instead of as raw observations. None of these works explore active use of non-canonical effects.

The purpose of this work is to combine the affordance concept's potential to structure complex action spaces with the strength of neural prediction for modelling action effects. A key aspect of the approach is that affordance recognition is applied identically on current (actual) states and predicted ("imagined") future states. Hence, future affordances are grounded in the same low-level, sub-symbolic, high-dimensional state representation as current affordances. By carrying sub-symbolic state representations forward through the planning process, we avoid the need for brittle symbolic effect specifications, and avoid planning failures due to hard-to-specify effects and preconditions. Furthermore, predicting future states in human-interpretable form facilitates interpretability of a robot's behaviour: a visualised plan provides users an intuitive explanation of the robot's action choices and outcome expectations.

Through simulation experiments, we show that the system produces suitable action sequences for goals provided at run-time, effectively predicting and avoiding non-canonical action effects that would cause failure of logically sensible action choices. We also demonstrate exploitation of such effects to reach otherwise inaccessible goal states.

A. Canonical and Non-Canonical Effects

Symbolic formalisation of affordance effects creates a distinction between *canonical* (formalised) and *non-canonical* (non-formalised) effects. Since we do not formalise effects, our approach makes no inherent distinction between the two. However, the distinction remains conceptually useful. Below we use the distinction informally as follows. Canonical effects are those effects one expects on basis of an affordance's label. Non-canonical effects are any other effects that may accompany execution of an affordance in particular circumstances.

Constraining a planner to formalised effects limits it to action outcomes foreseen and deemed intentional by the rule-designer. However, what may be an undesirable side-effect in one problem setting may be a clever solution in another problem setting. In systems where affordance effects are learned in the context of a specific goal, the distinction between canonical and non-canonical effects is less explicit, but learning of effects not pertinent to the goal will be deemphasised. Hence, we pursue fully goal-agnostic learning, and do not set goals until run-time.

B. Contributions

1) We propose a neural architecture that integrates affordance recognition and affordance effect prediction, capable of predicting future affordances through repeat application of scene prediction and affordance recognition.
2) We propose a planning algorithm that uses the neural architecture to plan action sequences that realise positive and negative goal conditions set at run-time.
3) We demonstrate that the system can plan action sequences in consideration of non-canonical affordance effects, avoiding and exploiting such effects as necessary.

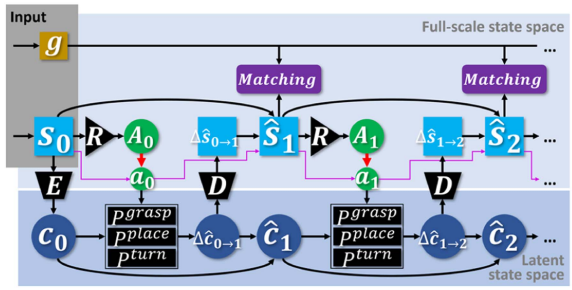

Fig. 2. Global system structure, rolled out for processing affordance sequences of length two, with $n_{aff} = 3$ and $A = \{Grasp, Place, Turn\}$. Processing for subsequent affordances is identical to that for the second. Light blue: full-scale state representations, dark blue: latent state representations, green: affordances, black: NN modules (R: recognition net, E: encoder, P: prediction module, D: decoder), yellow: goal image, purple arrows: affordance pixel data.

II. System Architecture

Fig. 2 shows the global structure of our setup. At the core of the system are two neural network modules: an affordance recognition module, and an affordance effect prediction module. We denote states as $s \in S$, let s_0 be (the observation of) the current state of the task environment, and use s_i, $i \geq 1$ to refer to possible future states. A denotes the set of affordance types in the robot's repertoire and n_{aff} its size. In our experiments, states are RGBD images, and $A = \{Grasp, Place, Turn\}$. Below we explain our setup in detail, starting with our interpretation of the affordance notion.

A. Affordance Concept

The affordance concept has been defined in numerous ways [2]. Since we purposely exclude definitions of intended effects, we arrive at the following minimalistic definition. In this work, an affordance of type $a \in A$ exists at point p in the problem space IFF the agent is capable of *commencing* an action of type a at p. In our experiments, p is a 4D vector consisting of a point in 3D space and a gripper angle, and an affordance of type a exists at p IFF (1) an entity of the type affordances of type a act on exists at the spatial location, and (2) the robot's gripper can reach this location in the orientation specified by the angle, i.e., the gripper can assume the starting pose for the affordance's execution. Given this definition, affordance presence can be evaluated from observation of the current scene alone, without consideration of physical effects beyond commencement. The exclusion of effectiveness at producing a specific outcome from the affordance conceptualisation echoes [18].

B. Affordance Recognition

Some existing affordance recognition methods perform recognition at pixel-level granularity [5], alongside semantic

object recognition. However, for integration in our planning architecture, it is desirable to recognise affordances in a format that can be interpreted as a physical action directly, without further processing. This approach is consistent with the finding that visual processing for object-directed action in humans occurs separate from and independent of semantic recognition [21]. Our recognition result format consists of the following values: (1) the affordance type, (2) a target point in 3D space, (3) a gripper angle. For the *grasp* and *turn* affordances in our experiments, the target point indicates the centre of the object to be grasped or turned, and for the *place* affordance, it indicates the point (on some surface) that the currently held object is to be placed at. Affordances may have additional parameters determining details of their execution. In the current repertoire, the *turn* affordance has a second angle parameter defining the turn amount and direction. However, because we define presence of an affordance in terms of *possibility to commence*, such parameters are not part of the recognition format.

We additionally include a value indicating the affordance's symmetry w.r.t. the gripper angle. This value indicates whether the affordance is qualitatively distinct from other valid affordances that differ only by the gripper angle. Consider grasping or placing a rotationally symmetrical object. The object will allow multiple (potentially infinitely many) valid gripper angles, but these angles produce qualitatively equivalent results. Recognising such equivalence allows us to improve efficiency of the planning process.

Our recognition format (a point in space plus some parameters) resembles that of the YOLO [22] family of object recognition architectures (a point in space plus bounding box dimensions). We adapt the ScaledYOLOv4 architecture [23] to perform affordance recognition instead of object recognition. In the interest of space, we limit our discussion of the architecture to our main modifications. We change the input format from RGB to RGBD. We drop the bounding box width and height outputs, replacing them with the affordance angle. ScaledYOLOv4 originally uses bounding box anchors to allow for multiple different detections at nearby locations. We adapt the anchor logic to allow for multiple affordances with different angles at nearby locations, setting anchors at 90° intervals. Additionally, we let detections include a z-coordinate and a symmetry value. As depth input is given as a depth map and z-coordinates are a simple additional detection feature, the basic network structure remains 2D. To adapt the net to our domain, bandwidth (channel count) of hidden layers was reduced (our domain is visually simple and the number of classes (affordances) is small), while cell count of the output was increased to facilitate detection of multiple affordances in close proximity. MSE losses are used for angles, z-coordinates, and symmetry values. We repurpose YOLO's 'objectness' as detection confidence. Weight initialisation follows the original ScaledYOLOv4 implementation. We filter detections of the same affordance type by proximity in state-angle space. When multiple detections are in overly close proximity, we retain the detection with the highest confidence.

C. Affordance Effect Prediction

The prediction pathway is based on the EM∗D architecture [24], [25]. The present work extends this architecture with differential prediction and affordance-specific sub-modules. The motivation for introducing differential prediction is that complex scenes may contain any number of elements that remain unaffected by execution of a given affordance. Reconstructing these elements in predicted states is unnecessary, increases the difficulty of the prediction problem, and degrades prediction quality. Instead, we let the system indicate where changes occurred and fill in the changed parts of the state. This allows for unchanged elements to be retained in their original fidelity.

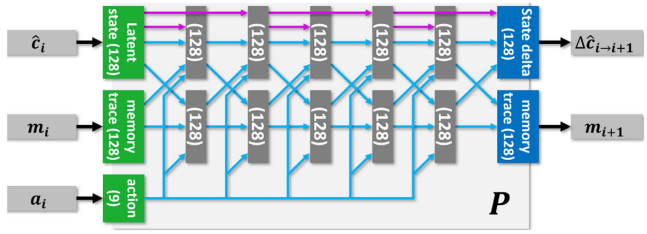

Fig. 3. *P* module architecture. Each block is a set of neurons. Numbers in brackets indicate neuron counts. Green: input, blue: output, dark grey: hidden. Action input is identical across layers. Blue lines indicate full connectivity between sets. Purple lines indicate 1-to-1 copying of activation values.

The prediction pathway consists of Encoder (E), Decoder (D), and n_{aff} prediction modules $P^a, a \in A$. E maps input state s_0 (an RGBD image in our implementation) to its latent representation c_0. P^a takes a (predicted or actual) latent representation c_i and a parametrisation a_i of affordance a as input and generates prediction $\Delta\hat{c}_{i\to i+1}$ of the difference $\Delta c_{i\to i+1}$ between the latent representation c_{i+1} of the state s_{i+1} resulting from performing a_i in s_i and the latent representation c_i of state s_i. Latent prediction $\hat{c}_{i+1}$ is obtained from c_i and $\Delta\hat{c}_{i\to i+1}$ through simple summing:

$$\hat{c}_{i+1} = c_i + \Delta\hat{c}_{i\to i+1} \tag{2}$$

D maps latent difference representation $\Delta\hat{c}_{i\to i+1}$ to full-scale representation $\Delta\hat{s}_{i\to i+1}$ of the state difference described therein.

The format of the full-scale difference representation is as follows. For an n-channel (e.g. 4 for RGBD images) state format, we use an $(n+1)$-channel difference descriptor, with the additional channel encoding a mask determining where the other channels overwrite the original state. Full-scale difference representations are applied to state representations as follows:

$$\hat{s}_{i+1} = s^{c+1}_{i\to i+1} * s^{1:c}_{i\to i+1} + \left(1 - s^{c+1}_{i\to i+1}\right) * s_i \tag{3}$$

Where c is the channel count of the state representation, and superscripts indicate channel selection.

Architectures in the prediction pathway are as follows. E consists of 2 convolutional layers with kernel size 3, stride 2, and output channel count 8, followed by 3 dense layers of 8192, 4096, and 128 neurons. D consists of 3 dense layers followed by 2 up-convolutional layers, with channel and neuron counts mirroring E. Following [26], we use 2x nearest neighbour upscaling to increase resolution between up-convolutional layers.

Each P^a module duplicates the architecture shown in Fig. 3. The P^a module used to process a given action input is determined by the affordance type value in that input. Hence actions are not simply passive input signals, but actively determine the course of signal propagation through the prediction pathway. The remaining values are input to the selected module. The 5th action input value is only used for the *turn* affordance. The last four inputs contain the RGBD values of a single pixel from the image representing the state in which the affordance is to be performed. For the initial state, this image is already part of the

input. For subsequent states, we use the generated prediction, so no additional external information is required to provide the pixel input. We project the 3D position of the affordance to pixel coordinates, and retrieve the RGBD values from the corresponding pixel. This RGBD input provides some minimal context about the world state at the affordance position.

As shown in Fig. 3, the P^a module has I/O blocks for a memory trace. The memory trace is passed between subsequent P^a modules in the pathway. When input is received from *E*, memory trace input is a zero vector.

All networks in the prediction pathway use the hyperbolic tangent activation function at all layers, except for the output layer of D, which simply clamps output activation to the [0, 1] range. The prediction pathway is implemented in JAX [27].

D. Planning & Execution Logic

Action sequences are planned as follows. We define the planning goal using an RGBD or RGB goal image depicting the objective to be achieved. By using a goal image of size smaller than the state image resolution, we can define objectives without having to specify the full goal state. For example, we can specify the goal of placing a cup on a block without specifying positions for other objects in the scene.

We expand a search tree of possible futures from the current state by repeated recognition and prediction. Each edge of the search tree corresponds to a fully parametrised affordance, and each node corresponds to a state. Given a state, recognition thus provides the edges extending from that state. To predict child state *s* for a given edge, we run prediction from the current state (root node of the search tree), with the chain of affordances leading up to *s* as action input.

Affordance parameters not affecting existence of the affordance at a given location are filled in between recognition and prediction. Our experiments feature one such parameter: the turn angle of the *turn* affordance. Such affordances are duplicated with different parametrisations. For the turn affordance, we parametrise the turn angle to $\pm 90°$. The recognition process may return multiple affordances that differ only in their gripper angle, and are rotationally symmetrical (i.e., symmetry parameter value > 0.5). For any set of such affordances, we process only the affordance with the highest confidence, to avoid unnecessary branching of the search tree.

Each predicted state is evaluated against the goal image (*matching* in Fig. 2) with a simple sliding window method. For a state image resolution of $w^s \times h^s$, we define a goal image as an image of resolution $w^g \times h^g$, $w^g \leq w^s$, $h^g \leq h^s$. We can then define the planning problem as follows.

$$plan = \underset{a_{0:n-1}}{\operatorname{argmin}} L\left(g, Pr\left(s_0, a_{0:n-1}\right)\right),\ n < n_{\max} \tag{4}$$

$$L\ (g, \hat{s}) = \min_{0<x<w^s-w^g,\ 0<y<h^s-h^g} MSE\left(g, \hat{s}_{x:x+w^g, y:y+h^g}\right) \tag{5}$$

Where $Pr(s_0, a_{0:n-1})$ applies the prediction pathway with state s_0 and parametrised affordance sequence $a_{0:n-1}$, n_{max} is the maximum sequence length to consider, and $w^s \times h^s$ is the state image resolution. So, the loss for a given plan (sequence of parametrised affordances) is the smallest MSE loss between the goal image and any patch of size $w^g \times h^g$ in the prediction generated by that plan. Typically, we return the plan with the lowest residual loss. However, we can also define "negative" goals, i.e., objectives to avoid. By selecting the affordance sequence producing the *highest* residual loss (i.e., replacing *argmin* with *argmax* in (4)), we find plans for avoiding or eliminating the situation expressed by the goal image.

Each process (recognition, prediction, evaluation) is parallelised per depth level of the search tree by batching.

III. Experimental Setup

A. Task Space

Fig. 1 shows a snapshot of our task scene. The scene is implemented in Unity [28]. The agent controls a virtual UR3 (Universal Robots) robot placed on a table. Motion planning is performed using MoveIt via a ROS connection, as provided by the Unity Robotics Hub [29]. The scene contains 0–3 cups, 0–3 balls, 0–2 grey "support" blocks, and one long coloured block, arranged in a rectangular area in front of the robot.

We define the set of affordances A as *{Grasp, Place, Turn}*. *Grasp* affordances exist at the centre point of each object, except for the support blocks. *Place* affordances exist at the centre of the top surface of each support block, and on 3 positions on the top surface of the coloured block. A *turn* affordance exists at the centre of the coloured block. The turn affordance turns the block 90° clockwise or counter-clockwise in the XY plane. After execution of an affordance, the robot returns to a default pose with its gripper pointing upward. After execution of a grasp affordance, the gripper holds the grasped object.

Following our *possibility to commence* affordance existence criterion, the existence of the above affordances is conditioned on (1) the current state of the gripper (*grasp* and *turn* affordances only exist if the gripper is empty, and *place* affordances only exist when the gripper is holding something), and (2) the starting pose being collision-free and within reach of the robot. No consideration is given to whether execution would produce the affordance's canonical result.

Various interactions between objects occur in this setup. Turning the block can cause nearby cups to fall over, and nearby balls to roll away. It will also cause any balls on top of the block to roll away, but a cup on top of the block will stay put and turn along with the block. Lifting the block when there is another object on top of it will cause that object to fall off. Placing an object on top of the coloured block when the coloured block is on top of a support block may produce an unstable situation, causing the block to tilt over and the object to fall off or roll away. Hence while the scene is simple, it produces a variety of non-canonical effects.

State images are captured from a top-down perspective to minimise occlusion, and rescaled to 128×128 resolution. The robot itself is visible in state images, and consequently states encode any object the gripper is holding.

B. Data Generation

We generate data for training the recognition and prediction networks as follows. We randomise the arrangement of objects in the scene (position of each object, and number of instances for each object except the coloured block). Block positions are discretised to produce stable configurations, while randomisation of ball and cup positions is continuous. We then perform a simple affordance detection routine. Each affordance location is represented in the simulation environment as an invisible marker attached to its host object. We check whether the marker is within

reach of the robot, and if so, check whether the affordance's commencement pose is accessible by placing a gripper-shaped object at the marker position and checking for collisions. If the gripper pose is collision-free, we record the affordance. For *grasp* and *turn* affordances on the block, the gripper angle is aligned to the block's orientation. For *grasp* affordances on radially symmetrical objects, as well as all *place* affordances, we perform the accessibility check at 90° intervals of the gripper angles. For *place* affordances, we let the gripper-shaped object hold a copy of the currently held object when checking for collisions. We record all found affordances as the *affordance list* for the state. We then randomly select and perform one affordance from the list, obtaining a new state. We repeat this process four times for each data sequence, or until no affordances remain in the scene. Each *(state, affordance list)* pair provides one example for the recognition network, while each sequence of states and executed affordances provides an example for the prediction network. We collected a dataset of 26563 sequences containing 100558 affordance executions.

C. Training

The Recognition network was trained using the standard ScaledYOLOv4 training procedure for 100 epochs. For data augmentation, we apply random small translations to states, shifting affordance positions accordingly. Test and validation sets consist of 500 sequences each.

The prediction pathway (E, P, D) was trained for 1M batches of 32 sequences each, using the SignSGD update rule [30] with automatic adjustment of the learning rate. Sequence length is varied per training batch from 1 to 4 affordances, and starting points within example sequences are selected randomly (where possible). Note that the sequence of P modules varies per example (being determined by the sequence of affordance types in the example). Training with various P module orderings ensures that the learned latent representation format is compatible between P modules. For data augmentation we use translations, mirroring, noise on action input values, and 180 degree turns of the affordance angle in cases where the angle is irrelevant.

IV. Evaluation

A. Evaluation – Recognition

We evaluate the recognition module in terms of recall and spurious detections. Recall is satisfied for a given affordance if a detection of the correct type is produced within 2.5cm from the ground truth position, with an angle error of $<5°$. This is sufficiently precise for affordances to be executable in the simulation environment. For scale, the long side of each block measures 28cm. The confidence threshold for detection is tuned on the validation set. Detections above the confidence threshold that fall outside the above ranges from a ground truth affordance are considered spurious (false positives). We measure performance on the test set and on 500 sequences from the training set. Quantitative results are shown in Table I. The "spurious" column reports the average number of spurious detections per state. Example detections are shown in Fig. 4.

Note that the robot state is visible in the state image, and recognition incorporates this information correctly. Detection errors primarily stem from 1) misjudgement of object accessibility in crowded arrangements, and 2) failure to mark grasp affordances on toppled cups as asymmetrical (most instances of graspable cups in the dataset are in upright position).

TABLE I
Affordance Recognition Accuracy

	Grasp	Place	Turn	All	Spurious
Test	0.979 5763/5886	0.999 9720/9732	0.995 1510/1518	0.99217 16993/17136	0.023 /state
Train	0.980 6048/6170	0.999 10372/10378	1.00 1568/1568	0.9929 18006/18134	0.0085 /state

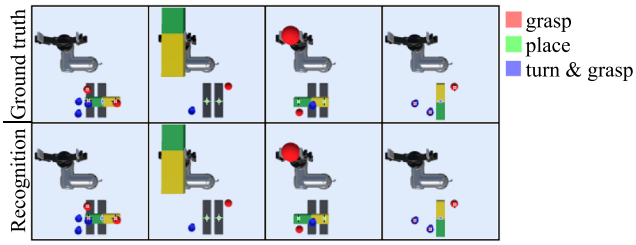

Fig. 4. Examples of affordance recognition. Lines extending from markers indicate grasp angles, with the colour of the line indicating whether the grasp angle affects the outcome (white) or not (black). Note that graspability and available grasp angles are affected by surrounding objects. *Grasp* and *turn* affordances are detected when the gripper is empty, whereas *place* affordances are detected when the gripper holds an object.

TABLE II
Prediction Accuracy

		With affordance pixel		Without affordance pixel	
Set	**Step**	**All-area accuracy**	**Changed area acc.**	**All-area accuracy**	**Changed area acc.**
Test	**1**	.0030 (.0017)	.042 (.020)	.0031 (.0018)	.044 (.025)
	2	.0039 (.0029)	.043 (.023)	.0038 (.0029)	.041 (.023)
	3	.0050 (.0047)	.042 (.022)	.0052 (.0048)	.044 (.027)
	4	.0055 (.0034)	.044 (.024)	.0054 (.0034)	.043 (.025)
	All	.0043 (.0035)	.043 (.022)	.0043 (.0035)	.043 (.025)
Train	**1**	.0029 (.0022)	.040 (.021)	.0029 (.0022)	.040 (.019)
	2	.0039 (.0028)	.041 (.023)	.0037 (.0028)	.040 (.022)
	3	.0048 (.0036)	.043 (.031)	.0048 (.0036)	.044 (.034)
	4	.0053 (.0038)	.044 (.024)	.0052 (.0038)	.043 (.026)
	All	.0042 (.0033)	.042 (.025)	.0041 (.0033)	.042 (.026)

B. Evaluation – Prediction

We evaluate the prediction pathway by performing prediction for all test set sequences and 500 training sequences, and calculating pixel value accuracy. Given our differential prediction setup, substantial areas of the state image are copied from one state to the next. To evaluate accuracy w.r.t. changed state areas specifically, we include pixel value accuracy for areas that have changed from the preceding state (in the ground truth sequence) as a secondary evaluation metric ("changed area accuracy"). Absolute mean RGBD pixel value error (range [0,1]) is given in Table II. We observe that the mean error for changed areas is ≤ 0.05 for all sequence lengths, with little overfitting.

The data includes inherently unpredictable outcomes. Uncontrolled kinetic effects (e.g. unstably placed objects falling over or rolling away) are not exactly predictable. However, perfect prediction is not necessary for effective planning. For example, as long as the prediction for an unstable placement captures that the placed object will not remain in the location where it was placed, it will suffice to avoid planning that placement. As seen in Fig. 5, predictable effects (a, c) are well predicted,

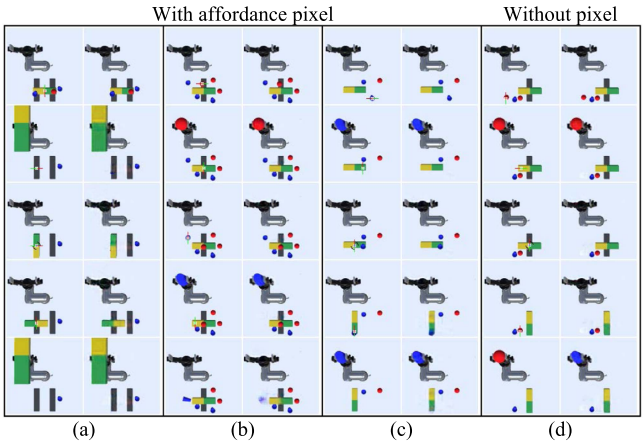

Fig. 5. Example predictions (test set). In each example, left column shows ground truth state sequence along with executed affordances, and right column shows initial and predicted states. Red and green lines on markers correspond to gripper fingers. For *turn* affordances, black quarter circles indicate the turn direction. a. Moving the block around. Note that the ball and one cup are flung away by the lifting the block. b. Balance effects. Placement of the blue cup destabilises the block, causing the cup to fall off. The resulting pose and location of the left cup is stochastic, but predicted approximately (blue blur). Tilting of the block also causes the red ball to roll off the block and out of the scene, which is predicted correctly. c. Turning a block with an object on it. The resulting displacement of the cup is predicted adequately, and subsequent grasping at the cup's new position results in a prediction where the robot holds the cup again. d. Example of object confusion between objects in near proximity by a network trained without the affordance pixel input.

TABLE III
AFFORDANCE DISCOVERY

	Grasp	Place	Turn	All	Spurious
Main	0.902	0.984	0.987	0.959	0.50/state
D-s-V	0.557	0.968	0.967	0.801	1.84/state
D-s-100	0.0752	0.150	0.144	0.119	4.63/state
D-s-1000	0.403	0.747	0.697	0.636	49.7/state
D-m-V	0.883	0.990	0.990	0.957	2.86/state
D-m-100	0.114	0.168	0.152	0.149	9.96/state
D-m-1000	0.608	0.768	0.769	0.719	99.9/state

whereas unpredictable effects (b) produce messier results, as expected. Note that when objects are placed on a block in an unstable manner (b), the block tilts and the objects fall off, but subsequently the block usually tilts back into a horizontal orientation. This behaviour is physically correct.

Affordance pixel input has no appreciable effect on the visual quality of predictions, but nets trained without this input occasionally confuse grasped objects in scenes that are crowded or contain objects in very close proximity, as seen in Fig. 5(d). This can easily cause planning failures. So, while the quantitative accuracy difference is small, we found nets trained with affordance pixel input to be more suitable for planning.

C. Evaluation – Affordance Discovery

Next, we combine prediction and recognition to quantitatively evaluate the system's ability to discover future affordances. For all sequences in the test set, we run prediction from the initial state to predict all subsequent states, and perform recognition on the predicted states. Matching criteria are as described in Section IV-A. Table III reports accuracy for prediction of future affordances per affordance, as well as the average number of spurious affordances predicted per state. We optimise the confidence threshold for recognition using the validation set.

As baseline, we implement an approximation of the DAF approach [18] ("DAF mode" for short) as follows. The encoder is unchanged. The decoder is replaced by a binary classification MLP that takes a latent state and parametrised affordance as inputs, and classifies whether or not the affordance will be available in the state (Affordance Model f_A in [18]). Hence no image representations of future states are recovered; latent states only encode affordance availability. DAF originally uses one prediction model for all affordances, whereas our setup uses a separate prediction module for each affordance. We train a single-module version (D-s in Table III) to approximate the original, and a multi-module version (D-m in Table III) for fairer comparison. In the single-module version, affordance type is given as additional input to the prediction module. DAF mode is trained on the same dataset, with ground truths for the classification model provided by the affordance list associated with each state. Negative examples are sourced from random states in the training set. Augmentation replicates that in the main setup. In contrast to the main setup, DAF does not produce affordance lists directly, but casts affordance discovery as a classification problem, with the classification model classifying whether a given parametrised affordances exists in a given state. Consequently, the confidence threshold presents a trade-off between recall and precision. To focus our evaluation on the ability to predict how affordance execution changes affordance availabilities over the course of a sequence, we optimise the threshold so as to maximise classification accuracy over affordances in the current state and the non-current affordances from the same sequence, over all sequences in the validation set. First, we measure classification accuracy on ground truth examples (D-s/m-V in Table III). The "spurious" column for this result indicates the mean number of non-current affordances from the same sequence that are classified as positives. For the single module setup, we observe that positive ground truth examples of the *place* and *turn* affordances are verified with accuracy close to the main setup, while accuracy for the more volatile *grasp* affordance lags behind. Positive ground truth verification accuracy of the multi-module setup matches the detection accuracy of the main setup. The optimised threshold came out lower here, and we do see a shift from precision towards recall.

Ground truth verification does not equal affordance discovery. DAF's affordance model is a binary classifier, so sampling is required to discover affordances. We evaluate discovery of future affordances for sampling budgets of 100 and 1000 samples per state (D-s/m-100 and D-s/m-1000 in Table III). Sample affordances are drawn from the training data to ensure that they fall within the task domain, with additional position augmentation to ensure thorough coverage. We see that with a sampling budget of 1000, a decent proportion of affordances is discovered, but substantial numbers of false positives are produced as well ("spurious" column). These results indicate that DAF mode is effective for affordance verification, but less effective for discovering a state's set of affordances in our particular task domain. Affordance characteristics play a role here. Our system is adept at finding highly localised affordances while more spatially extended affordances are hard to represent, whereas DAF more naturally represents spatially extended affordances, and appears less apt at finding highly localised ones.

D. Evaluation – Planning

To evaluate the integrated system, we prepared a set of 9 task types, each designed to require some modicum of insight in the scene's dynamics. For each task type, we generate 3 instances, for a total of 27 tasks. Instances differ in object placements, roles, and goal specifics, while leaving the basic challenge intact. Example tasks and generated solutions are shown in Fig. 6. Footage of the robot executing plans of each type can be found in the accompanying video. Note that in each plan, states after the first are predictions generated by the system. Below we briefly describe the challenge posed by each task type.

1) Requires consideration of action order, because the red ball occupies the goal position for the blue cup.
2) Access to the ball is initially obstructed by the block, requiring that the block is moved out of the way first, and restored to its original pose after moving the ball.
3) Placing the cup on the block directly would be unstable, causing the block to tilt and the cup to fall off. Moving the block allows stable placement of the cup.
4) Requires counterbalancing. In the left task instance, the ball must be placed in its target location before the cup, and vice versa in the right task instance. Note how the position of the grey block affects the generated plan.
5) Turning the block first would knock over the cup. Solving the task requires understanding that a cup placed on a block turns along with that block.
6) The red ball should be moved out of the area directly in front of the block, but the gripper cannot reach the ball due to the placement of the cup. The system must exploit the fact that a turning block pushes away balls in its path, and pick the correct turn direction to achieve this effect.
7) Negative goal: eliminate the ball. The affordance repertoire provides no straight-forward way of removing objects from the scene. Solved here by placing the ball on the block and then turning the block, causing the ball to roll away (see video for an alternative solution).
8) Negative goal: eliminate the cup from the scene. Solved here by first placing the cup on the block, and then lifting the block, causing the cup to be flung out of the scene.
9) Negative goal: eliminate the ball. The ball is inaccessible due to the cup placement. However, the robot can hide it from its own view by placing the coloured block over it.

Tasks of types 1 and 2 are solved using canonical affordance effects plus consideration of how affordance execution changes affordance availability in future states. Task types 3–9 involve consideration of non-canonical affordance effects. Task types 7–9 use negative goals. For these cases we use small goal patches, and ignore the depth channel in the plan loss calculation. This is to eliminate trivial solutions where the object is merely lifted up (lifting an object up changes its visual size and depth values). The system comes up with various ways of eliminating objects from the scene, exploiting non-canonical affordance effects to satisfy the goal condition.

Table IV shows quantitative results for each task type. "Candidates" refers to the mean number of plans considered during search. We also report the mean total time cost incurred for the recognition, prediction, and goal-matching processes in each task type (measured on a single Nvidia A100 GPU). We see that time cost remains manageable for all tasks.

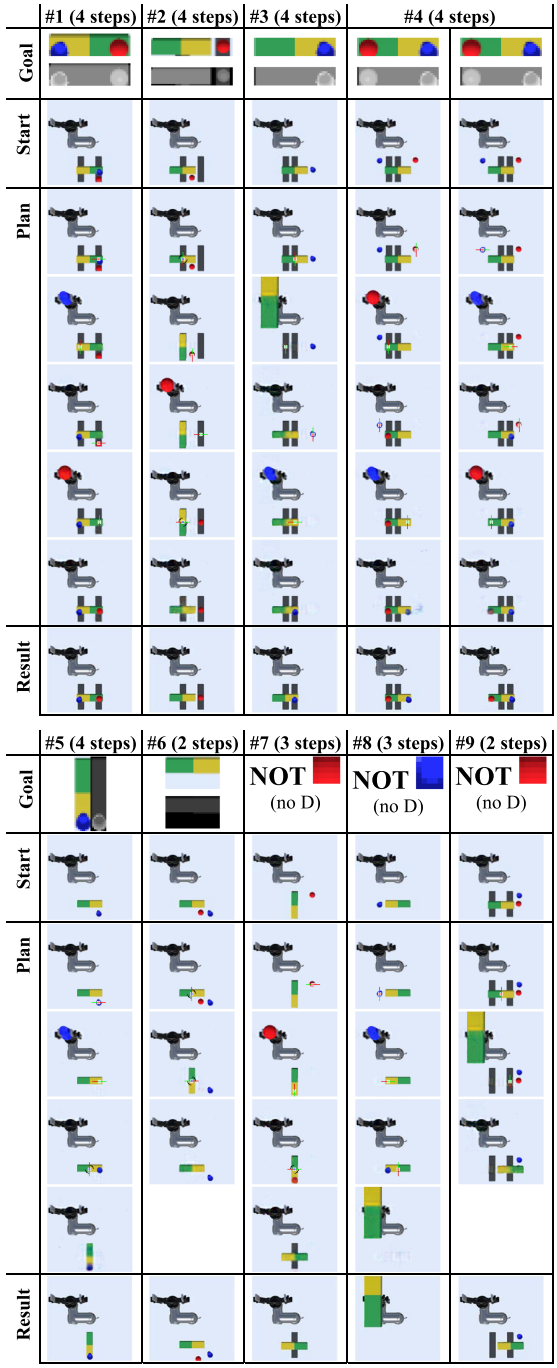

Fig. 6. Examples of generated plans and their execution results. See Section IV-D for task concepts. Goals are presented in RGBD or RGB format. To visualise the generated plan, we draw its affordance sequence on a state sequence consisting of the initial state and network-generated predictions of the subsequent states for the plan. States are in RGBD format, but the D channel is omitted for space considerations.

TABLE IV
PLANNING PERFORMANCE

Task type		1	2	3	4	5	6	7	8	9
Success rate		3/3	3/3	3/3	3/3	3/3	3/3	2/3	2/3	3/3
Candidates		513	2254	2452	2651	652	37	272	168	22
Time cost[s]	Rec.	0.202	0.285	0.290	0.296	0.204	0.097	0.146	0.144	0.098
	Pred.	0.575	1.988	2.222	2.302	0.609	0.097	0.263	0.235	0.097
	Match	0.038	0.108	0.117	0.121	0.038	0.011	0.049	0.045	0.022

V. Discussion & Future Work

Our results demonstrate affordance-structured reasoning with consideration of non-canonical effects. The system effectively avoids non-canonical effects that would obstruct access to the goal state, and allows non-canonical effects to be actively exploited to reach goals that are otherwise inaccessible.

The system still has significant limitations. We currently operate on discrete affordances, but many real-world affordances have continuous components. Place affordances would be better represented as regions instead of points, and parameters such as the turn angle should be generalised to continuous values. Another issue is how to handle actions with stochastic effects. We are exploring probabilistic prediction formats to enable reasoning through stochastic effects.

A strength of the system is that it takes goals at run-time. However, goals are currently provided as images, which is impractical in practice. As predictions are in image format, methods for quantifying image-text agreement (e.g. CLIP [31]) may be integrated to enable goal definition in natural language.

With regard to scalability, main hurdles are time cost and domain complexity. In our experimental setup, the number of affordances per state is limited, and we can feasibly expand the search tree to reasonable depth through parallelisation of computationally expensive processes, as seen in Table IV, but for more affordance-rich environments or longer sequences, we will need heuristics to constrain which branches are expanded. The system could be combined with symbolic planning methods to quickly solve straight-forward tasks, with sub-symbolic prediction supporting robustness through plan verification and, when necessary, exploration of less canonical search tree branches. Recovering full state representations from latent states has noted advantageous for goal definition and interpretability, but will be challenging for more complex domains. However, NN-based predictive modelling is a rapidly advancing field. Our results are indicative of the future potential of such methods for affordance-based planning.

References

[1] J. Gibson, *The Senses Considered As Perceptual Systems*. Boston, MA, USA: Houghton-Mifflin, 1966.

[2] P. Zech et al., "Computational models of affordance in robotics: A taxonomy and systematic classification," *Adaptive Behav.*, vol. 25, no. 5, pp. 235–271, 2017.

[3] N. Yamanobe, W. Wan, I. G. Ramirez-Alpizar, D. Petit, and T. Tsuji, "A brief review of affordance in robotic manipulation research," *Adv. Robot.*, vol. 31, pp. 1086–1101, 2017.

[4] P. Ardón, È. Pairet, K. S. Lohan, S. Ramamoorthy, and R. P. A. Petrick, "Affordances in robotic tasks—A survey," 2020, *arXiv:2004.07400*.

[5] T. Do, A. Nguyen, I. Reid, D. G. Caldwell, and N. G. Tsagarakis, "AffordanceNet: An end-to-end deep learning approach for object affordance detection," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 5882–5889.

[6] S. Deng, X. Xu, C. Wu, K. Chen, and K. Jia, "3D AffordanceNet: A benchmark for visual object affordance understanding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1778–1787.

[7] S. Nishide, T. Ogata, R. Yokoya, K. Komatani, H. G. Okuno, and J. Tani, "Structural feature extraction based on active sensing experiences," in *Proc. IEEE Int. Conf. Inform. Educ. Res. Knowl.-Circulating Soc.*, 2008, pp. 169–172.

[8] M. Shridhar, L. Manuelli, and D. Fox, "CLIPort: What and where pathways for robotic manipulation," in *Proc. 5th Conf. Robot Learn.*, 2022, pp. 849–906.

[9] J. McCarthy and P. Hayes, "Some philosophical problems from the standpoint of artificial intelligence," *Mach. Intell.*, vol. 4, pp. 473–502, 1969.

[10] M. Steedman, "Plans, affordances, and combinatory grammar," *Linguistics Philosophy*, vol. 25, no. 5, pp. 723–753, 2002.

[11] E. Ugur and J. Piater, "Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 2627–2633.

[12] E. Ugur and J. Piater, "Refining discovered symbols with multi-step interaction experience," in *Proc. IEEE-RAS 15th Int. Conf. Humanoid Robots*, 2015, pp. 1007–1012.

[13] E. Erdemir, C. B. Frankel, K. Kawamura, S. M. Gordon, S. Thornton, and B. Ulutas, "Towards a cognitive robot that uses internal rehearsal to learn affordance relations," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 2016–2021.

[14] A. Lerer, S. Gross, and R. Fergus, "Learning physical intuition of block towers by example," in *Proc. 33rd Int. Conf. Int. Conf. Mach. Learn.*, vol. 48, 2016, pp. 430–438.

[15] R. Mottaghi, H. Bagherinezhad, M. Rastegari, and A. Farhadi, "Newtonian scene understanding: Unfolding the dynamics of objects in static images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3521–3529.

[16] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik, "Learning visual predictive models of physics for playing billiards," in *Proc. 4th Int. Conf. Learn. Representations*, 2016.

[17] A. Byravan and D. Fox, "SE3-nets: Learning rigid body motion using deep neural networks," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 173–180.

[18] D. Xu, A. Mandlekar, R. Martín-Martín, Y. Zhu, and L. Fei-Fei, "Deep affordance foresight: Planning through what can be done in the future," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 6206–6213, doi: 10.1109/ICRA48506.2021.9560841.

[19] C. Wang, D. Xu, and L. Fei-Fei, "Generalizable task planning through representation pretraining," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 8299–8306, Jul. 2022.

[20] J. Liang, M. Sharma, A. LaGrassa, S. Vats, S. Saxena, and O. Kroemer, "Search-based task planning with learned skill effect models for lifelong robotic manipulation," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 6351–6357.

[21] A. D. Milner et al., "Perception and action in 'visual form agnosia'," *Brain*, vol. 114, pp. 405–428, 1991.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

[23] C. Wang, A. Bochkovskiy, and H. M. Liao, "Scaled-YOLOv4: Scaling cross stage partial network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 13029–13038.

[24] D. Tanaka, S. Arnold, and K. Yamazaki, "EMD net: An encode–manipulate–decode network for cloth manipulation," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 1771–1778, Jul. 2018.

[25] S. Arnold and K. Yamazaki, "Cloth manipulation planning on basis of mesh representations with incomplete domain knowledge and voxel-to-mesh estimation," *Frontiers Neurorobotics*, vol. 16, 2023, doi: 10.3389/fnbot.2022.1045747.

[26] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," 2016. [Online]. Available: http://distill.pub/2016/deconv-checkerboard

[27] J. Bradbury et al., "JAX: Composable transformations of Python+numpy programs," 2018. [Online]. Available: http://github.com/google/jax

[28] V. Berges et al., "Unity: A general platform for intelligent agents," 2018, *arXiv:1809.02627*.

[29] "Unity robotics hub," 2020. [Online]. Available: http://github.com/unity-technologies/unity-robotics-hub

[30] J. Bernstein, Y. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed optimisation for non-convex problems," in *Proc. 35th Int. Conf. Mach. Learn. (ICML 2018), Proc. Mach. Learn. Res.*, 2018, vol. 80, pp. 559–568.

[31] A. Radford et al., "Learning transferable visual models from natural language supervision," in *Proc. 38th Int. Conf. Mach. Learn. (ICML 2021), Proc. Mach. Learn. Res.*, 2021, vol. 139, pp. 8748–8763.