Autonomous Agents and Multi-Agent Systems (2020) 34:45
https://doi.org/10.1007/s10458-020-09471-w

From demonstrations to task-space specifications. Using causal analysis to extract rule parameterization from demonstrations

Daniel Angelov[1] · **Yordan Hristov**[1] · **Subramanian Ramamoorthy**[1]

Published online: 17 June 2020

Abstract
Learning models of user behaviour is an important problem that is broadly applicable across many application domains requiring human–robot interaction. In this work, we show that it is possible to learn generative models for distinct user behavioural types, extracted from human demonstrations, by enforcing clustering of preferred task solutions within the latent space. We use these models to differentiate between user types and to find cases with overlapping solutions. Moreover, we can alter an initially guessed solution to satisfy the preferences that constitute a particular user type by backpropagating through the learned differentiable models. An advantage of structuring generative models in this way is that we can extract causal relationships between symbols that might form part of the user's specification of the task, as manifested in the demonstrations. We further parameterize these specifications through constraint optimization in order to find a safety envelope under which motion planning can be performed. We show that the proposed method is capable of correctly distinguishing between three user types, who differ in degrees of cautiousness in their motion, while performing the task of moving objects with a kinesthetically driven robot in a tabletop environment. Our method successfully identifies the correct type, within the specified time, in 99% [97.8–99.8] of the cases, which outperforms an IRL baseline. We also show that our proposed method correctly changes a default trajectory to one satisfying a particular user specification even with unseen objects. The resulting trajectory is shown to be directly implementable on a PR2 humanoid robot completing the same task.

Keywords Human–robot interaction · Robot learning · Explainability

This research is supported by the Engineering and Physical Sciences Research Council (EPSRC), as part of the CDT in Robotics and Autonomous Systems at Heriot-Watt University and The University of Edinburgh. Grant reference EP/L016834/1.

✉ Daniel Angelov
d.angelov@ed.ac.uk

[1] School of Informatics, University of Edinburgh, Edinburgh, United Kingdom

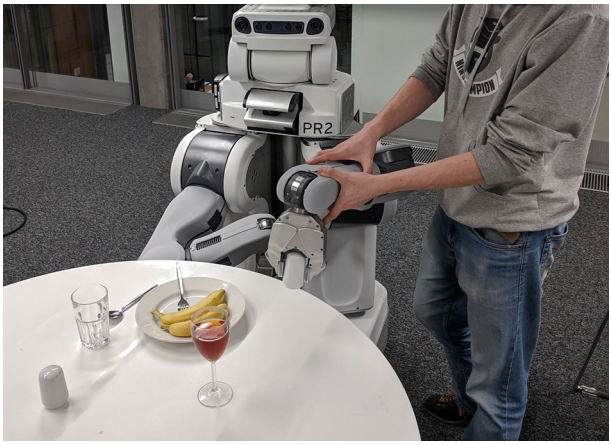


Fig. 1 Example setup—the demonstrated task is to return the pepper shaker to its original location—next to the salt shaker. Deciding which objects to avoid when performing the task can be seen as conditioning on the user specifications, implicitly given during a demonstration phase

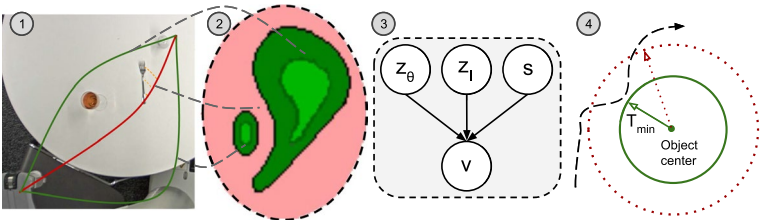


Fig. 2 **1** Demonstrations that satisfy the user task specification maintain a distance from fragile objects (i.e. a wine glass), or fail to satisfy the specification by moving over sharp items. **2** An environment can have multiple clusters of valid trajectories in the latent space, conditioned on user type. **3** The validity of trajectories can be represented as a causal model. Whether a trajectory is part of a cluster v is conditioned on the specific path z_θ, the environment z_I, and the specification s. **4** The minimum radius from the object centre—T_{min}, which would change the validity of a trajectory

1 Introduction

As we move from robots dedicated to a restricted set of pre-programmed tasks to being capable of more general purpose behaviour, there is a need for easy re-programmability of these robots. A promising approach to such easy re-programming is Learning from Demonstration, i.e., by enabling the robot to learn from and reproduce behaviors shown to it by a human expert—Fig. 1.

This paradigm lets us get away from having to handcraft rules and allows the robot to learn by itself, including modelling the specifications the teacher might have used during the demonstration. Often such innate preferences are not explicitly articulated, typically being in the form of biases resulting from experience with other potentially unrelated tasks sharing parallel environmental corpora—Fig. 2(1). The ability to notice, understand and reason causally about these 'deviations', whilst still learning to perform the demonstrated task is of significant interest.

Similarly, other methods for Learning from Demonstration as discussed by Argall et al. [2] and Wirth et al. [37] in the Reinforcement Learning domain are focused on finding a general mapping from observed state to an action, thus modeling the system or attempting to capture the high-level user intentions within a plan. The resulting policies are not

generally used as *generative models*. As highlighted by Sünderhauf et al. [34] one of the fundamental challenges with robotics is the ability to *reason* about the environment, beyond a state-action mapping.

Thus, when receiving a positive demonstration, we should aim to understand the causal reasons differentiating it from a non-preferential one, rather than merely mimicking the particular trajectory. When people demonstrate a movement associated with a concept, they rarely mean to refer to one singleton trajectory alone. Instead, that instance is typically an element of a set of trajectories sharing particular features. So, we want to find groups of trajectories with similar characteristics that may be represented as clusters in a suitable space. We are interested in learning these clusters so that subsequent new trajectories can be classified according to whether they are good representatives of the class of intended feasible behaviors. Further, we want to distill these specifications into a set of parameterized rules and find a safety envelope that can represent the learned model. For instance, one such rule may be "The robot should not get closer than T_{min} away from an object". These rules would generalize to unseen world configurations, as they are dependent on object characteristics.

It is often the case that in problems that exhibit great flexibility in possible solutions, different experts may generate solutions that are part of different clusters—Fig. 2(2). In cases where we naively attempt to perform statistical analysis, we may end up collapsing to a single mode or merging the modes in a manner that doesn't entirely reflect the underlying semantics (e.g., averaging trajectories for going left/right around an object).

When we talk about task specification, we understand the high-level descriptions of a task based trajectory and its desired behavior/interaction with a cluttered environment and its symbolic representation through causal analysis. For instance learning the manner, by which the robot end-effector may move above or around objects in the scene. The specifications, as learned by the network, are the observed regularities in the human behavior. These rules are then parameterized by performing constrained optimization based on the demonstrations or samples from the learned model.

We present a method for introspecting in the latent space of a model which allows us to relax some of the assumptions illustrated above and more concretely to:

- find varied solutions to a task by sampling a learned generative model, conditioned on ka particular user specification.
- backpropagate through the model to change an initially guessed solution towards an optimal one with respect to the user specification of the task.
- counterfactually reason about the underlying feature preferences implicit in the demonstration, given key environmental features, and to build a causal model describing this.
- find a safety envelope of parameters to sets of rules representing the specifications though constraint optimization that allows their future use in motion planning.

2 Related work

2.1 Learning from demonstration

Learning from demonstration involves a variety of different methods for approximating the policy. In some related work, the state space is partitioned and the problem is viewed as one of classification. This allows for the environment state to be in direct control of

the robot and to command its discrete actions—using Neural Networks J Matari'c [25], Bayesian Networks Inamura [19], or Gaussian Mixture Models Chernova and Veloso [8]. Alternatively, it can be used to classify the current step in a high-level plan Thomaz and Breazeal [35] and execute predetermined low-level control.

In cases where a continuous action space is preferred, regressing from the observation space can be achieved by methods such as Locally Weighted Regression Cleveland and Loader [9].

Roboticists e.g., Sünderhauf et al. [34], have long advocated the position that reasoning as part of planning is dependent on reasoning about objects, their geometric manifestations, and semantics. This is based on the view that structure within the demonstration should be exploited to better ground symbols between modalities and to the plan.

One way to learn such latent structure can be in the form of a reward function obtained from Inverse Reinforcement Learning as described in Ng et al. [28], Zhifei and Meng Joo [38], Brown and Niekum [4]. However, it is not always clear that the underlying true reward, in the sense of being *the unique reward* an expert may have used, is re-constructable or even if it can be sufficiently approximated. Combining multiple demonstrations to blend a desired expert response as in Vukoviundefined et al. [36] may not recreate an expected output with divergent multi-clustered demonstrations, which we are interested in the current work. Alternatively, Angelov et al. [1] and Gombolay et al. [14] propose a solution that is based on composing smaller policies to mitigate the search for hierarchical decomposition of the demonstration through direct learning of a goal scoring metric or through pair-wise ranking. Alternatively, preference-based reinforcement learning (PbRL), Wirth et al. [37], offers methods whose focus is on learning from non-numeric rewards, directly from the guidance of a demonstrator. Such methods are particularly useful for problems in high-dimensional domains, e.g. robotics—[18, 20, 21], where a concise numeric reward (unless highly shaped) might not be able to correctly capture the semantic subtleties and variations contained in the expert's demonstration. Thus, in the context of PbRL, the method we propose learns a user specification model using user-guided exploration and trajectory preferences as a feedback mechanism, using definitions from Wirth et al. [37].

2.2 Causality and state representation

The variability of environmental factors makes it hard to build systems relying only on correlation data statistics for specifying their state space. Methods that rely on causality, Pearl [30], Harradon et al. [15], and learning the cause and effect structure, Rojas-Carulla et al. [32], are much better suited to supporting the reasoning capabilities required for transfer of core knowledge between situations. Interacting with the environment allows robots to perform manipulations that can convey new information to update the observational distribution or change their surrounding, and in effect perform interventions within the world. Counterfactual analysis helps in a multi-agent situation with assignment of credit as shown by Foerster et al. [12]. It shows that marginalizing an agents actions in a multi-agent environment through counterfactuals allows to learn a better representative Q-function. In this work, we similarly employ a causal view of the world where we capture the expert preference in the model and evaluate it against a different set of environments, which is prohibitive if we used human subjects.

Learning sufficient state features has been highlighted by Argall et al. [2] as an open challenge for LfD. The problem of learning disentangled representations aims at generating

a good composition of the latent space, separating the different modes of variation within the data. Higgins et al. [16], Chen et al. [6] have shown promising improvements in disentangling of the latent space with few a priori assumptions, by manipulating the Kullback–Leibler divergence loss of a variational auto-encoder. Denton and Birodkar [10] show how the modes of variation for content and temporal structure should be separated and can be extracted to improve the quality of the next frame video prediction task if the temporal information is added as a learning constraint. While the disentangled representations may not directly correspond to the factors defining action choices, Johnson et al. [22] adds a factor graph and composes latent graphical models with neural network observation likelihoods.

The ability to manipulate the latent space and separate variability as well as obtain explanation about behavior is also of interest to the interpretable machine learning field, as highlighted by Doshi-Velez and Kim [11].

2.3 Constrained optimization

The ability to find an optimal solution under a set of constraints has been well studied, e.g., in [3, 5]. Moskewicz et al. [27] is one representative and state of the art method for propositional satisfiability (SAT). These methods have a history of being applied to robotics problems for high-level planning, motion planning [13] and stability analysis [24].

In this paper, we use these methods to efficiently navigate the search space whilst adhering to a set of non-linear constraints. With the development of increasingly more mature libraries for constrained optimization and SAT solving, such as [29], whose CP–SAT solver is based on [33], we can efficiently rewrite the set of specifications as parametrised channelling rules activated under different conditions, which partition the state space of the problem. As a result, we can optimize their respective parameters from the demonstrations.

3 Problem formulation

In this work, we assume that the human expert and robotic agent share multiple static tabletop environments where both the expert and the agent can fully observe the world and can interact with an object being manipulated. The agent can extract RGB images of static scenes and can also be kinesthetically driven while a demonstration is performed. The task at hand is to move an object held by the agent from an initial position p_{init} to a final position p_f on the table, while abiding by certain user-specific constraints. Both p_{init} and $p_f \in \mathbb{R}^P$. The user constraints are determined by the demonstrator's type s, where $s \in S = \{s_1, \ldots, s_n\}$ for n user types.

Let $\mathcal{D} = \{\{\mathbf{x}_1, v_1\}, \ldots, \{\mathbf{x}_N, v_N\}\}$ be a set of N expert demonstrations, where $\mathbf{x}_i = \{I, tr_i^s\}$, $I \in \mathbb{R}^M$ is an RGB image of the tabletop scene, tr_i^s is the trajectory and v_i is a binary label denoting the validity of the trajectory with respect to the user type s. Each trajectory tr_i^s is a sequence of points $\{p_0, \ldots, p_{T_i}\}$, where $p_0 = p_{init}$ and $p_{T_i} = p_f$. The length of the sequences is not constrained—i.e. T is not necessarily the same for different trajectories.

The learning task is to project each $\mathbf{I} \in \mathbb{R}^M$ into $\mathbf{Z_I} \in \mathbb{R}^K$, by an encoder $Z_I = E(I)$, and $tr_i^s \in \mathbb{R}^{PT_i}$ into $\mathbf{Z}_\theta \in \mathbb{R}^L$, by Bèzier curve reparameterization, $Z_\theta = Bz(tr_i^s)$, with significantly reduced dimensionality $K \ll M$, $L \ll PT_i$. Both Z_I and Z_θ are used in order to predict the validity $\hat{v}_i$, $\hat{v}_i = C_s(Z_I, Z_\theta)$ of the trajectory tr_i^s with respect to the user type s. With an optimally-performing agent, $\hat{v}_i \equiv v_i$. For more details see Fig. 3.

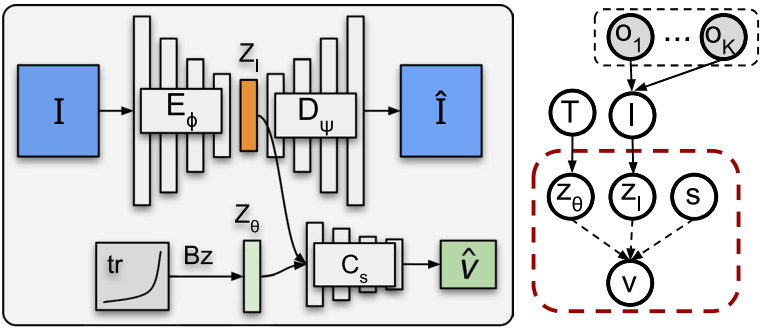


Fig. 3 Left: Specification model architecture. The environmental image I, $I \in R^{100\times100\times3}$, is passed through an Encoder–Decoder Convolutional Network, with a 16–8–4 3×3 convolutions, followed by fully connected layer, to create a compressed representation $Z_I, Z_I \in R^{15}$. It is passed along with the trajectory parameterization $Z_\theta, Z_\theta \in R^2$ through a 3-layer fully connected classifier network that checks the validity of the trajectory $C_s(z)$ with respect to the spec. s. Right: The environment, compressed to z_I, is composed of objects $(o_1, \ldots, o_K)$. A trajectory T is parameterized by z_θ, which alongside the factors z_I and user specification s are part of the specification model

In order to alter an initial trajectory, we can find the partial derivative of the model with respect to the trajectory parameters with the model conditioned on a specific user type s,

$$\Delta = \frac{\partial C_s(z|\hat{v}=1)}{\partial z_\theta}$$

We can take a gradient step Δ and re-evaluate. Upon achieving a satisfactory outcome, we can re-project z_θ back to a robot-executable trajectory $tr^s = Bz^{-1}(z_\theta)$.

The main feature we want in our model is for the the latent space to be structured in a way that would allow us to distinguish between trajectories conforming (or not) to the user specifications. In turn, this generates good trajectories. We further need the model to maintain certain kinds of variability in order to allow us to estimate the causal link between the symbols within the world and the validity of a trajectory, given a specification.

4 Specification model

We use the Deep Variational Auto-Encoder Framework—see [23]—as a base architecture. The full model consists of a convolutional encoder network q_ϕ, parametrised by ϕ, a deconvolutional decoder network p_ψ, parametrised by ψ, and a classifier network C, comprised of a set of fully-connected layers. The encoder network is used to compress the world representation I to a latent space Z_I, disjoint from the parameterization of the trajectories Z_θ. The full latent space is modeled as the concatenation of the world space and trajectory space $Z = Z_I \cup Z_\theta$ as seen on Fig. 3.

Parameters—α, β, γ—are added to the three terms of the overall loss function—see Eq. 1—so that their importance during learning can be leveraged. In order to better shape the latent space and to coerce the encoder to be more efficient, the Kullback–Leibler divergence loss term is scaled by a β parameter, as in [16].

$$\begin{aligned}\min_{\psi,\phi,\mathbf{C}} & \mathcal{L}(\psi,\phi;I,z_I,z_\theta,v)\\ &= -\alpha\mathbb{E}_{E_\phi(z_I|I)}(logD_\psi(I|z_I))\\ &\quad + \beta D_{KL}(E_\phi(z_I|I)||D_\psi(z_I))\\ &\quad - \gamma\left[v\log(C(z)) + (1-v)\log(1-C(z))\right]\end{aligned} \tag{1}$$

By tuning its value we can ensure that the distribution of the latent projections in Z_I do not diverge from a prior isotropic normal distribution and thus influence the amount of disentanglement achieved in the latent space. A fully disentangled latent space has factorised latent dimensions—i.e. each latent dimension encodes a single data-generative factor of variation. It is assumed that the factors are independent of each other. For example, one dimension would be responsible for encoding the X position of an objectin the scene, another for the Y position, third for the color, etc. Higgins et al. [17] and Chen et al. [7] argue that such low-dimensional disentangled representations, learned from high-dimensional sensory input, can be a better foundation for performing separate tasks—trajectory classification in our case. Moreover, we additionally add a binary cross-entropy loss (scaled by γ) associated with the ability of the full latent space Z to predict whether a trajectory tr^s associated a world I satisfies the semantics of the user type s—$\hat{v}$. We hypothesise that by backpropagating the classification error signal through Z_I would additionally enforce the encoder network to not only learn factorised latent representations that ease reconstruction, but also trajectory classification. The full loss can be seen in Eq. 1.

The values for the three coefficients were empirically chosen in a manner such that none of the separate loss terms overwhelms the gradient updates while optimising $\mathcal{L}$.

5 Causal modeling

Naturally, our causal understanding of the environment can only be examined through the limited set of symbols, O, that we can comprehend about the world. In this part, we work under the assumption that an object detector is available for these objects (as the focus of this work is on elucidating the effect of these objects on the trajectories rather than on the lower level computer vision task of object detection per se). Given this, we can construct specific world configurations to test a causal model and use the above-learned specification model as a surrogate to inspect the validity of proposed trajectories. We assume that by understanding the minimum number of required demonstrations per scene, we can learn a model that reflects the expert decisions.

If we perform a search in the latent space z_θ, we can find boundaries of trajectory validity. We can intervene and counterfactually alter parameters of the environment and specifications and see the changes of the trajectory boundaries. By looking at the difference of boundaries in cases where we can test for associational reasoning, we can causally infer whether

- the different specifications show alternate valid trajectories
- a particular user type reacts to the existence of a specific symbol within the world.

5.1 Specification model differences

We are interested in establishing the causal relationship within the specification model as shown on Fig. 3. We define our Structural Causal Model (SCM), following the notation of [31] as

$$\mathfrak{C} := (\mathbf{S}, P_{\mathbf{N}}), \quad S = \{X_j := f_j(\mathbf{PA}_j, N_j)\}$$

where nodes $\mathbf{X} = \{Z_\theta, Z_I, S, V\}$ and $\mathbf{PA}_j = \{X_1, X_2, \ldots X_n\} \backslash \{X_j\}$. Given some observation $\mathbf{x}$, we can define a counterfactual SCM $\mathfrak{C}_{\mathbf{X}=\mathbf{x}} := (\mathbf{S}, P_{\mathbf{N}}^{\mathfrak{C}|\mathbf{X}=\mathbf{x}})$, where $P_{\mathbf{N}}^{\mathfrak{C}|\mathbf{X}=\mathbf{x}} := P_{N|\mathbf{X}=\mathbf{x}}$.

We cannot logistically perform counterfactuals using the data and humans, but by relying on the learned models to have encapsulated the expert representations, we can perform the causal analysis on those surrogate models.

We can choose a particular user specification $s \sim p(S), s \neq s_{\mathbf{x}}$ and use the specification model to confirm that the different specification models behave differently given a set of trajectories and scenes, i.e. the causal link $s \rightarrow v$ exists by showing:

$$\mathbb{E}\left[P_v^{\mathfrak{C}|\mathbf{X}=\mathbf{x}}\right] \neq \mathbb{E}\left[P_v^{\mathfrak{C}|\mathbf{X}=\mathbf{x};do(S:=s)}\right] \tag{2}$$

We expect different user types to generate a different number of valid trajectories for a given scene. Thus, by intervening on the user type specification we anticipate the distribution of valid trajectories to be altered, signifying a causal link between the validity of a trajectory within a scene to a specification.

5.2 Symbol influence on specification models

We want to measure the response of the specification models of intervening in the scene and placing additional symbols within the world. We use the symbol types $O = \{o_1, \ldots, o_k\}$ as described in Sect. 7.1. To accomplish this, for each symbol within the set we augment the scene I, part of the observation $\mathbf{x}$ with symbol o, such that $I_{new} = I \cup o$. We do not have the ability to realistically remove objects from the scene, for this reason, our augmentation involves adding such objects, which can be interpreted as applying an additional overlay of the object on the image. If we observe that the entailed distributions of $P_v^{\mathfrak{C}|\mathbf{X}=\mathbf{x};do(Z_I:=z_{Inew})}$ changes i.e.

$$\mathbb{E}\left[P_v^{\mathfrak{C}|\mathbf{X}=\mathbf{x}}\right] \neq \mathbb{E}\left[P_v^{\mathfrak{C}|\mathbf{X}=\mathbf{x};do(Z_I:=z_{Inew})}\right] \tag{3}$$

then the introduced object o has a causal effect upon the validity of trajectories conditioned upon the task specification $s_{\mathbf{x}}$.

We investigate the intervention of all symbol types permutated with all task-space specifications to build an understanding of the relationship between the manner of execution and the influence of the symbols on it.

6 Parameterization of specifications

The aim of this work is to provide a closed system that decomposes demonstrations into a set of parametrized rules. We have shown methods for ways to construct a model that encapsulates such specifications, using causal analysis to extract symbols which influence the demonstrations. Further, relying on these outputs, we use constraint optimization to find optimal parameters for a set of predefined rules representing the specifications.

We rely on the CP-SAT solver in Or-tools, [29], and formulate a set of rules that can be understood as corresponding to a point in the trajectory being in collision with an object, being in a region of influence of an object or in free-space. We formally define this in the following manner:

$$f(p_i) = \begin{cases} inf, & \text{if } p_i \leq T_{min} \\ ||p_i - p_{obj-k}||_2, & \text{if } p_i \leq T_{min} + T_{object-k} \text{ for any object } k \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

We would have a penalty constraint that $\sum_i f(p_i) < F_{max}$ for any trajectory $tr = \{p_1, p_2, \ldots, p_{T_i}\}$, where F_{max} is chosen as the attention buffer for the demonstrator. We are interested in providing a maximum or minimum safety envelope for the trajectory and would, thus maximize/minimize $\mathcal{L}_T = \sum_k T_{object-k} + T_{min}$. We can observe how the requirements for positive or negative change with the different safety target.

For each point in a trajectory, we add a set of constraints representing the different channels as seen under Eq. 4. The sum of penalties for each trajectory is added as a constraint conditioned on the validity of the trajectory. We would then find a feasible or optimal solution for the parameters—$T_{min}, T_{object-1}, \ldots, T_{object-K}$ under the minimum/maximum cost function.

7 Experimental setup

7.1 Dataset

The environment chosen for the experiment consists of a top down view of a tabletop on which a collection of items, O = *{utensils, plates, bows, glasses}*—Fig. 4, usually found in a kitchen environment, have been randomly distributed. The task that the demonstrator has to accomplish is to kinestetically move a robotic arm gently holding a pepper shaker from one end on the table to the other (p_{init} = bottom left, p_f = top right) by demonstrating a trajectory, whilst following their human preferences around the set of objects—see Fig. 5. The demonstrators are split into user types S, $S = \{careful, normal, aggressive\}$ based on the trajectory interaction with the environment. The semantics behind the types are as follows: the *careful* user tries to avoid going near any objects while carrying the pepper shaker, the *normal* user tries to avoid only cups and the *aggressive* user avoids nothing and tries to finish the task by taking the shortest path from p_{init} to p_f.

The agent observes the tabletop world and the user demonstrations in the form of 100x100 pixel RGB images $I, I \in \mathbb{R}^{100\times100\times3}$. The demonstrator—see Fig. 1—is assigned one of the types in S, has to produce a number of possible trajectories, some that satisfy the semantics of their type and some that break it—Fig. 2(1). As specified in Sect. 3, each

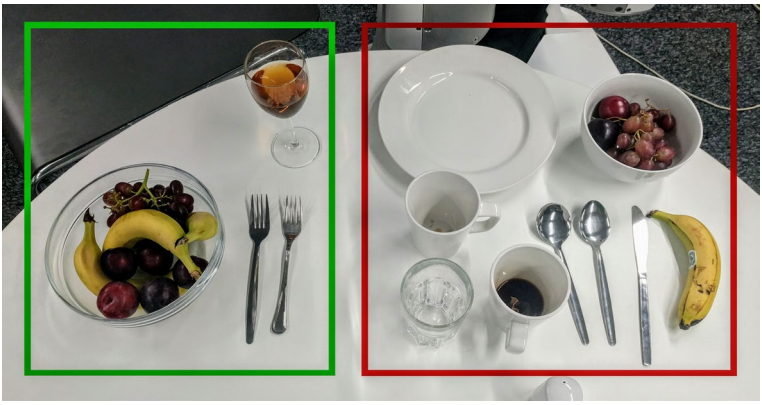


Fig. 4 Items used for the generation of the training (green/left) and test (red/right) scenes (Color figure online)

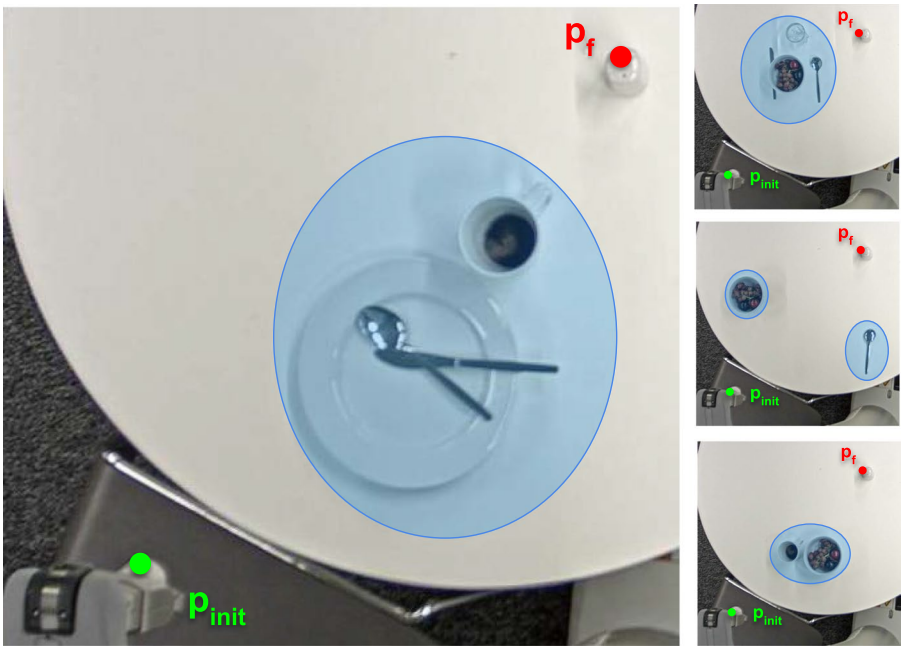


Fig. 5 Sample images used to represent example scenes. p_{init} and p_f are as defined in Sect. 3. Blue blobs represent potential obstacles in the scene, which some user types might want to avoid, and are only drawn for illustrative purposes (Color figure online)

trajectory tr^s is a sequence if points $\{p_0, \ldots, p_T\}$, where $p_0 = p_{init}$ and $p_{T_i} = p_f$. Each point p_j,$j \in \{0, \ldots, T\}$ represents the 3D position of the agent's end effector with respect to a predefined origin. However, all kinesthetic demonstrations are performed in a 2D (XY) plane above the table, meaning that the third coordinate of each point p_j carries no information ($P = 2$). An efficient way to describe the trajectories is by using a Bèzier curve representation—see [26]. The parameterization of a single trajectory becomes the 2D location

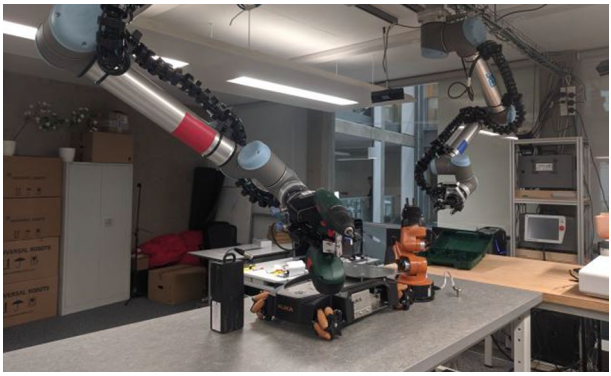


Fig. 6 An additional task of moving the the drill to the work space of the other robot

of the central control point parametrized by θ, together with p_{init} and p_f. However, the initial and final points for each trajectory are the same and we can omit them. Thus, with respect to the formulations in Sect. 3 $L = 2$ and $Z_\theta \in \mathbb{R}^2$.

In total, for each user type $s \in S$, 20 scenes are used for training, with 10 trajectories per scene. The relationship between the number of trajectories per scene and the model's performance is explored in Sect. 8. For evaluation purposes additional 20 scenes are generated, using a set of new items that have not been seen before—see Fig. 4.

7.2 Evaluation

We evaluate the performance of the model by its ability to correctly predict the validity of a trajectory with a particular specification. We perform an ablation study with the full model ($\alpha \neq 0, \beta \neq 0, \gamma \neq 0,$), AE model ($\beta = 0$), and classifier ($\alpha = 0, \beta = 0$). We investigate how the performance of the model over unseen trajectories varies with a different number of trajectories used for training per scene. We randomize the data used for training 10 times and report the mean.

As a baseline we use an IRL model $r_s(p, I)$, such that the policy π producing a trajectory tr^s that is optimal wrt:

$$\underset{tr^s}{\text{argmax}} \sum_{i=0}^{N} r_s(p_i, I)$$

Additionally, we test the ability of the learned model to alter an initially suggested trajectory to a valid representative of the user specification. We assess this on the test set with completely novel objects by taking 30 gradient steps and marking the validity of the resulting trajectory.

We perform a causal analysis of the model with respect to the different user specifications and evaluate the difference in their expected behavior. Additionally, we intervene by augmenting the images to include specific symbols and evaluate the difference of the expectation of their entailed distribution. This highlights how the different specifications react differently to certain symbols.

We conclude by finding optimal maximum and minimum parameters for a set of rules that the motion controller can use to plan with varying levels of safety vs travel time. We perform constrain optimization on the task of moving a drill on a workbench robot

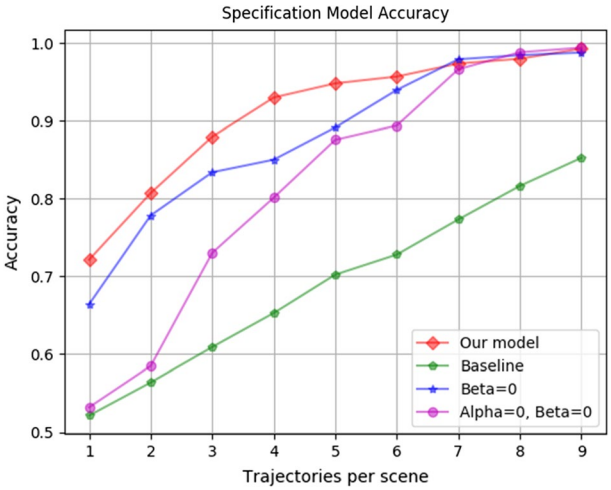


Fig. 7 The accuracy of the different models with respect the number of trajectories used within a scene. The lines indicate the mean accuracy with 10 different seed randomizations of the data. As the number of trajectories per scene increases, the performance of all models improves, but especially with a lower number of trajectories, our full model shows the biggest gains

assembly area as shown on Fig. 6 and report results in Sect. 8.4. We obtain demonstrations in a representative simulated 2D environment, such that the demonstrated trajectories no longer need to adhere to the Bèzier representation.

The aim is to find the rule parameterization based on Eq. 4, such that this representation can later on be used for motion planning optimization as an additional cost. We would aim to extract the limits of the parameters to create an envelope of possible costs and not a bound of the geometric models that represent the objects.

8 Results

In this section we show how modeling the specifications of a human demonstrator's trajectories, in a table-top manipulation scenario within a neural network model, can be later used to infer causal links through a set of known features about the environment.

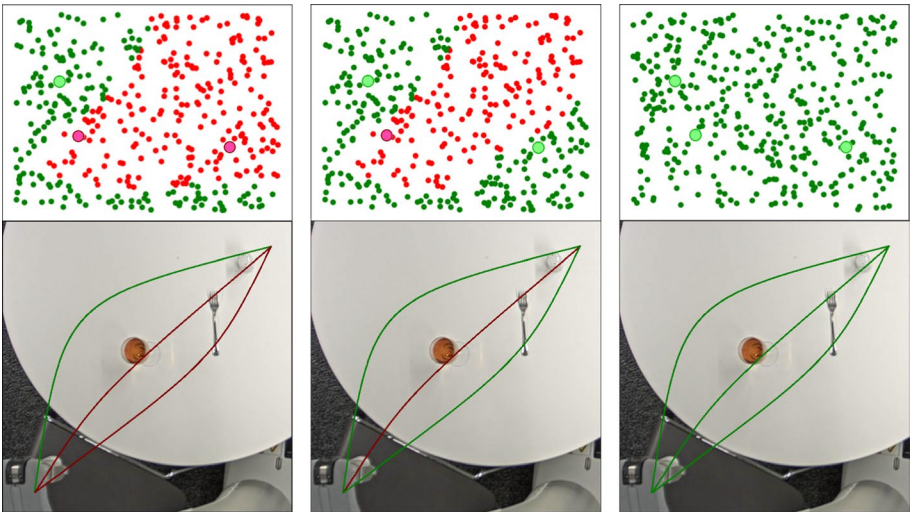
8.1 Model accuracy

We show the accuracy of the specification model in Fig. 7 and on our website.[1] Changing the number of trajectories shown within a scene has the highest influence on the performance going from 72% [67.3–77.5] for a single trajectory to 99% [97.8–99.8][2] when using 9 trajectories. The results illustrate that the models benefit from having an auto-encoder component to represent the latent space. However, they asymptotically approach perfect behavior as the number of trajectories per scene increases. Interestingly, the IRL baseline shows the need for much more information in order to create an appropriate policy.

If we look into the latent space of the trajectory—Fig. 8—we can see that the trajectory preferences have clustered and there exists an overlap between the different model

[1] Website on https://sites.google.com/view/learnspecifications.

[2] The numbers in brackets indicate the first and third quartile.



(a) Careful **(b)** Normal **(c)** Aggressive

Fig. 8 Sampling of the latent trajectory space—Z_θ—of the preference model with different specifications. It can be observed how for the same region in the latent trajectory space—e.g. bottom right—the different user types have different validity values for the same trajectory—e.g. normal versus careful user types around the cutlery and glass

Table 1 The success rate of perturbing a non valid trajectory into a valid one under different user specifications

User types	Success rate (%)
Careful	75
Normal	95
Aggressive	100

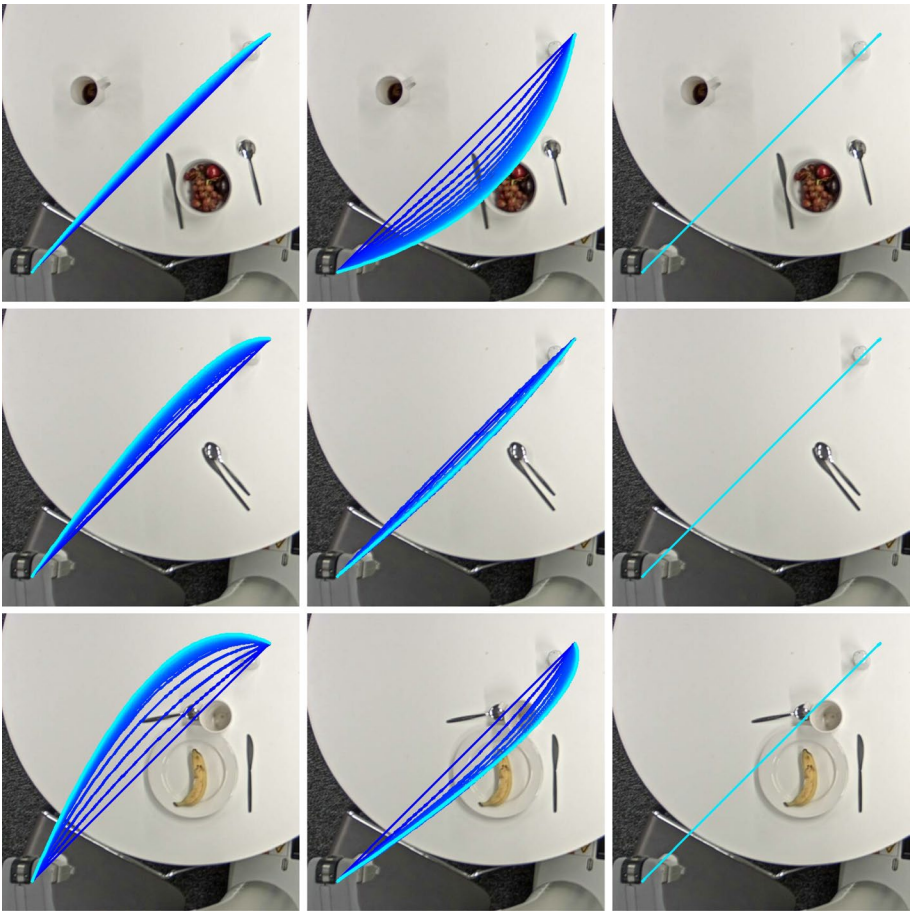
specifications. It also illustrates what the models' specifications can show about the validity of the trajectory.

8.2 Trajectory backpropagation

We can use the learned specification model and perturb an initially suggested trajectory to suit the different user types by backpropagating through it and taking gradient steps within the trajectory latent space.

Based on the unseen object test scenes, the models were evaluated under the different specifications and the results can be found in Table 1. Individual trajectory movements can be seen in Fig. 9.

The first row of Fig. 9 shows that the careful user type steering away from both the cup and bowl/cutlery, whereas in the normal user type, the model prefers to stay as far away from the cup as possible, ignoring the bowl. The model conditioned on the aggressive user type does not alter its preference of the trajectory, regardless of it passing through objects. The second row illustrates a situation, where the careful model shifts the trajectory to give more room to the cutlery, in contrast to the normal case. The final row highlights a



(a) Careful **(b)** Normal **(c)** Aggressive

Fig. 9 An initial trajectory (seen in dark blue) is used as a base solution to the task for difference scenes—rows 1, 2, 3. Furthermore, the parametrisation z_θ for each initial trajectory is continuously updated so that it better abides by the semantics of the different user specifications—columns a, b, c. It can be seen that as the gradient steps in Z_θ are taken, the resulting intermediate trajectories are shifted to accommodate the preference of the model until the final trajectory (light blue) is reached. Color change from dark to light blue designates progressive gradient steps (Color figure online)

situation, where the resulting trajectories vastly differ depending on the conditioning of the specification model.

8.3 Causal analysis

On Table 2 we can see the mean of the entailed distribution depending on the type of intervention performed. The results of Eq. 2 can be seen in the first column under "No intervention". It shows the expected likelihood $E[p(v|X = x, S = s)]$ of validity of a trajectory given a set of observations with different user specifications. Conditioning on the different types of user specifications, we can see that the validity increases (from 0.43 to 1.0), meaning a

Table 2 The respective distributions of validity $p(v|X = x, S = s)$ with different user types depending on the intervention performed for a random trajectory to be valid under the user specification

User types	No intervention	Bowl	Plate	Cutlery	Glass
Safe	0.43	**0.27**	**0.28**	**0.31**	**0.30**
Normal	0.62	0.62	0.63	0.62	**0.48**
Aggressive	1.00	1.00	1.00	1.00	1.00

The first column shows the mean distribution over the information obtained over the observations. The cells in bold indicate significant change with respect to the no intervention column. Those cells highlight a change, which is interpreted as a causal link between the intervened symbol and the user type

higher number of possible solutions can be identified. The variety of solutions can be seen in Fig. 8. This naturally follows the human assumption about the possible ways to solve a task with different degrees of carefulness. In the case of the final user type, all of the proposed trajectories have successfully solved the problem.

In the subsequent columns on Table 2 we can see the mean probability of validity for when we intervene in the world and position randomly a symbol of different type within the scene. By comparing the value with the ones in the first column (as discussed above), we can assess the inequality in Eq. 3.

In the case of a safe user specification, adding a symbol of any type decreases the probability of choosing a valid trajectory (from 0.43 down to 0.27). This indicates that the model reacts under the internalized specification to reject previously valid trajectories that interact with the intervened object.

For the normal user type, significant changes are observed only when we introduce a glass within the scene. This means it doesn't alter its behavior with respect to any of the other symbols.

In the last case, the aggressive user type doesn't reject any of the randomly proposed trajectories and that behavior doesn't change with the intervention. It suggests the specification model, in that case, is not reacting to the scene distribution.

Based on these observations, we can postulate that the specification model has internalized rules such as "If I want to be careful, I need to steer away from any objects on the table" or "To find a normal solution, look out for glass-like objects.".

This type of causal analysis allows us to introspect in the model preference and gives us an understanding of the decision making capabilities of the model.

8.4 Parameterization of task-space specifications

Based on the demonstrated trajectories, we can find parameterization of the rules specified in Eq. 4 for a world with 2 distinct objects. We can observe the resulting parameters for object distance for 3 different participants in Table 3. We are measuring the distances in pixel units, and as the camera is orthogonal to the surface, they can be transformed to real world distances.

On Fig. 10 we can observe the progression of these threshold distances when we alter the number of valid and invalid examples. This allows us to better choose where future focus should be when obtaining demonstrations for alternative tasks. If we look at Fig. 10a, b to increase the confidence that we have found a maximum safety boundary, we need to counter-intuitively provide more positive examples. Whereas if we

Table 3 The object threshold distances found from demonstrations of different participants

	T_{min}	$T_{min} + T_{object-1}$	$T_{min} + T_{object-2}$
User 1	(36) 59	(37) 135	(37) 159
User 2	(37) 45	(38) 145	(38) 145
User 3	(48) 52	(49) 152	(49) 152

The values in brackets indicate the radius when optimizing for the minimal $\mathcal{L}_T$ vs the maximum

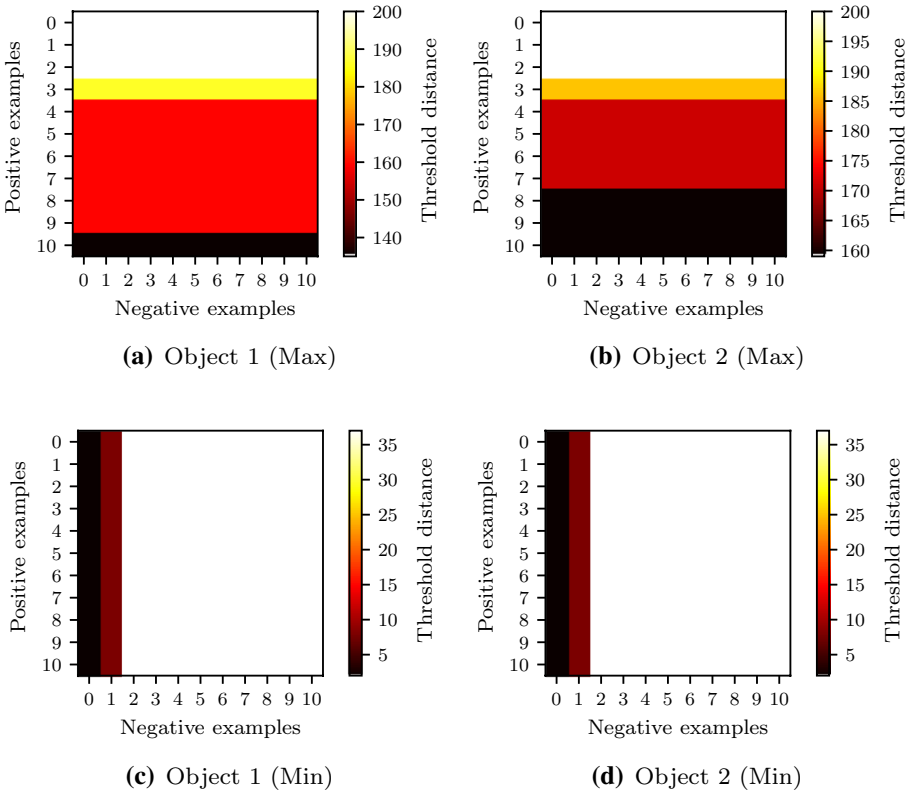


Fig. 10 The transition of the threshold distance ($T_{min} + T_{object-k}$) for different number of positive and negative examples. We can see the impact of increasing the number of trajectories when we want to find an optimally maximum/minimum distance around and object

are interested in the minimum safety envelope, Fig. 10c, d illustrates that we need to give invalid trajectories. Thus, the true underlying object distance will lie between the observed maximum and minimum boundaries.

The resulting boundaries around the symbols do not necessarily represent the object boundaries, but the expert representation of the min/max expected distance of interaction around them. Combining the rules in Eq. 4 and the values in Table 3 allows us to create an additional cost map that can be used to perform motion planning in the scene

following the user expectations. Combining this with the causal analysis gives us the ability to incorporate only the required symbols within the planning framework.

9 Conclusion

Learning behavioural types is essential for completing interactive human–robot tasks. It helps avoid nuisance and promotes better foresight into human actions and plans. Being able to decompose those user types into interpretable and reusable models is of high importance.

In this work, we demonstrate how to construct and use a generative model to differentiate between behavioral types, derived from expert demonstrations. We show how performance changes with the number of trajectories illustrated in a scene. Additionally, by using the same learned model, it is possible to change any solution to satisfy the preference of a particular user type, by taking gradient steps in the latent space of the obtained model.

Performing causal analysis allows for the extraction of causal links between the occurrence of specific symbols within the scene and the expected validity of a trajectory. The models exhibit different behaviors with regard to the different symbols within the scene leading to correctly inferring the underlying specifications that the humans were using during the demonstrations.

Further, by assuming an underlying set of specifications that users follow, it is possible to find the safety envelope boundaries for the objects within the scene. Additionally, we investigate what type of demonstrations would help move the minimum/maximum side of this boundary toward the optimum.

This paper demonstrates a method that converts demonstrations into a set of functions that represent the underlying specifications. Those are specifically linked to objects within the world and are causally discarded for uninteresting objects.

References

1. Angelov, D., Hristov, Y., Burke, M., & Ramamoorthy, S. (2020). Composing diverse policies for temporally extended tasks. *IEEE Robotics and Automation Letters*, *5*(2), 2658–2665.
2. Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, *57*(5), 469–483. https://doi.org/10.1016/j.robot.2008.10.024.
3. Bertsekas, D. P. (2014). *Constrained optimization and Lagrange multiplier methods*. Cambridge: Academic Press.
4. Brown, D.S., & Niekum, S. (2018). Machine teaching for inverse reinforcement learning: Algorithms and applications. arXiv:1805.07687
5. Byrd, R. H., Lu, P., Nocedal, J., & Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, *16*(5), 1190–1208.

6. Chen, T. Q., Li, X., Grosse, R., & Duvenaud, D. (2018). Isolating sources of disentanglement in variational autoencoders. arXiv e-prints, arXiv:1802.04942.
7. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems* (pp. 2172–2180).
8. Chernova, S., & Veloso, M. (2007). Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the 6th international joint conference on autonomous agents and multiagent systems, ACM, AAMAS '07* (pp. 233:1–233:8), New York.
9. Cleveland, W.S., & Loader, C. (1996). Smoothing by local regression: Principles and methods. In *Statistical theory and computational aspects of smoothing* (pp. 10–49) Springer.
10. Denton, E., & Birodkar, V. (2017). Unsupervised Learning of Disentangled Representations from Video. arXiv e-prints, arXiv:1705.10915.
11. Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. arXiv e-prints. arXiv:1702.08608.
12. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2017). Counterfactual multi-agent policy gradients. arXiv:1705.08926
13. Ghallab, M., Nau, D., & Traverso, P. (2004). *Automated planning: Theory and practice*. Amsterdam: Elsevier.
14. Gombolay, M., Jensen, R., Stigile, J., Son, S.H., & Shah, J. (2016). Apprenticeship scheduling: Learning to schedule from human experts. In *AAAI Press/international joint conferences on artificial intelligence*.
15. Harradon, M., Druce, J., & Ruttenberg, B. (2018). Causal learning and explanation of deep neural networks via autoencoded activations. arXiv e-prints. arXiv:1802.00541.
16. Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M. M., et al. (2017). beta-VAE: Learning basic visual concepts with a constrained variational framework. *ICLR*, *2*, 6.
17. Higgins, I., Sonnerat, N., Matthey, L., Pal, A., Burgess, C.P., Bošnjak, M., Shanahan, M., Botvinick, M., Hassabis, D., & Lerchner, A. (2018). Scan: Learning hierarchical compositional visual concepts. arXiv preprint arXiv:170703389
18. Hristov, Y., Angelov, D., Burke, M., Lascarides, A., & Ramamoorthy, S. (2019). Disentangled relational representations for explaining and learning from demonstration. In *Conference on robot learning (CoRL)*.
19. Inamura, T. (1999). Acquisition of probabilistic behavior decision model based on the interactive teaching method. In *Proceedings of the 9th international conference on advanced robotics* (pp. 523–528). https://ci.nii.ac.jp/naid/20000105704/en/.
20. Jain, A., Wojcik, B., Joachims, T., & Saxena, A. (2013). Learning trajectory preferences for manipulators via iterative improvement. In *Advances in neural information processing systems* (pp. 575–583).
21. Jain, A., Sharma, S., Joachims, T., & Saxena, A. (2015). Learning preferences for manipulation tasks from online coactive feedback. *The International Journal of Robotics Research*, *34*(10), 1296–1313.
22. Johnson, M. J., Duvenaud, D., Wiltschko, A. B., Datta, S. R., & Adams, R. P. (2016). Composing graphical models with neural networks for structured representations and fast inference. arXiv e-prints arXiv:1603.06277
23. Kingma, D. P., & Welling, M. (2013). Auto-encoding variational Bayes. arXiv e-prints. arXiv :1312.6114.
24. Koch, K. H., Mombaur, K., & Soueres, P. (2012). Optimization-based walking generation for humanoid robot. *IFAC Proceedings Volumes*, *45*(22), 498–504.
25. Matari'c, J. M. (1999). *Sensory-motor primitives as a basis for imitation: Linking perception to action and biology to robotics*. Cambridge, MA: MIT Press.
26. Mortenson, M. E. (1999). *Mathematics for computer graphics applications*. New York: Industrial Press Inc.
27. Moskewicz, M. W., Madigan, C. F., Zhao, Y., Zhang, L., & Malik, S. (2001). Chaff: Engineering an efficient sat solver. In *Proceedings of the 38th annual Design Automation Conference, ACM* (pp. 530–535).
28. Ng, A.Y., & Russell, S.J., et al. (2000). Algorithms for inverse reinforcement learning. In *Icml* (pp. 663–670)
29. Nikolaj van Omme, V. F., & Perron, Laurent. (2014). *or-tools user's manual*. Google: Tech. rep.
30. Pearl, J. (2009). *Causality: Models, reasoning and inference* (2nd ed.). New York, NY: Cambridge University Press.
31. Peters, J., Janzing, D., & Schölkopf, B. (2017). *Elements of causal inference: Foundations and learning algorithms*. Cambridge, MA: MIT Press.

32. Rojas-Carulla, M., Baroni, M., & Lopez-Paz, D. (2017). Causal discovery using proxy variables. arXiv preprint. arXiv:1702.07306.
33. Shaw, P., Furnon, V., & De Backer, B. (2003). A constraint programming toolkit for local search. In D. L. Woodruff (Ed.), *Optimization software class libraries* (pp. 219–261). Boston: Springer.
34. Sünderhauf, N., Brock, O., Scheirer, W., Hadsell, R., Fox, D., Leitner, J., et al. (2018). The limits and potentials of deep learning for robotics. arXiv e-prints. arXiv:1804.06557.
35. Thomaz, A., & Breazeal, C. (2004). Tutelage and socially guided robot learning. In *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS) (IEEE Cat No04CH37566)* (Vol. 4, pp. 3475–3480).
36. Vukoviundefined, N., Mitiundefined, M., & Miljkoviundefined, Z. (2015). Trajectory learning and reproduction for differential drive mobile robots based on gmm/hmm and dynamic time warping using learning from demonstration framework. *Engineering Applications of Artificial Intelligence*, *45*(C), 388–404. https://doi.org/10.1016/j.engappai.2015.07.002.
37. Wirth, C., Akrour, R., Neumann, G., & Fürnkranz, J. (2017). A survey of preference-based reinforcement learning methods. *The Journal of Machine Learning Research*, *18*(1), 4945–4990.
38. Zhifei, S., & Meng Joo, E. (2012). A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics*, *5*(3), 293–311.