Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Unsupervised Novelty Characterization in Physical Environments Using Qualitative Spatial Relations

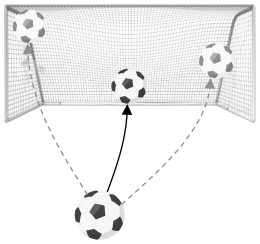
Ruiqi Li , Hua Hua , Patrik Haslum , Jochen Renz
School of Computing, Australian National University, Canberra Australia
{ruiqi.li, hua.hua, patrik.haslum, jochen.renz}@anu.edu.au

Abstract

Detecting, characterizing and adapting to novelty, whether in the form of previously unseen objects or phenomena, or unexpected changes in the behavior of known elements, is essential for Artificial Intelligence agents to operate reliably in unconstrained real-world environments. We propose an automatic, unsupervised approach to novelty characterization for dynamic domains, based on describing the behaviors and interactions of objects in terms of their possible actions. To abstract from the variety of realizations of an action that can occur in physical domains, we model states in terms of qualitative spatial relations (QSRs) between their entities. By first learning a model of actions in the non-novel environment from the state transitions observed as the agent interacts with the world, we can detect novelty by the persistent deviations from this model that it causes, and characterize the novelty by new or modified actions. We also present a new method of learning action models from observation, based on conceptual similarity and hierarchical clustering.

1 Introduction

In the real world, "novelty" refers to situations that violate implicit or explicit assumptions about agents, the environment, or their interactions (Langley 2020). As Artificial Intelligence (AI) agents such as robots become increasingly ubiquitous, they are expected to operate reliably in unconstrained environments where novelties are increasingly likely to occur. Therefore, it is essential for AI agents to be capable of detecting, characterizing and adapting to novelty in their environment (Boult et al. 2020). In this paper, we focus on the tasks of detecting and characterizing novelty, i.e., constructing an update to the agent's world model to reflect the detected novelty. Arguably, an agent could adapt to novelty through a process of trial-and-error, without an explicit characterization of the novelty. However, an explicit model of the novelty can enable more efficient adaptation, as it allows for reasoning about the novelty. The characterizations we create are expressed in terms of actions, with preconditions and effects, that describe the behavior of the novelty, and thus very close to the kind of model that is used in AI planning (Li et al. 2018). An explicit model of the novelty also enables the agent to share the knowledge it has learned about the novelty with other agents, including humans. This is key to making the agent's response to the novelty explainable to an observer, such as a human monitoring the agent.



```
(:action shoot-into-goal
 :parameters (?a - ball ?b - goal)
 :precondition (and (disconnected ?a ?b)
                    (far ?a ?b)
                    (approaching ?a ?b)
                    (stationary ?b))
 :effect (and (not (disconnected ?a ?b))
              (nt_proper_part ?a ?b)
              (not (far ?a ?b))
              (touching ?a ?b)
              (not (approaching ?a ?b))
              (stationary ?a))
```

Figure 1: Left: Some possible realizations of the action of shooting a ball into a goal. Right: The corresponding qualitative action model. Predicates are in the form of relation object triples like (disconnected ?a ?b).

Furthermore, we focus on physical domains, in which the agent and other world objects interact according to physical laws. In such domains, the behavior of novel objects is consistent, following the same laws when interacting with the world. As a case study, we apply our approach to the physics-based video game Angry Birds.

Our approach to characterizing novelty in the environment consists of two steps (see Figure 3 for an overview). In the first step, we learn, from observation, a model of the environment without novelty. This model takes the form of a set of actions, defined by their preconditions and effects, which describe the possible behaviors of objects in the environment. We call this the *base model*. Actions are generalized from state transitions observed during training. This step is done off-line, which means we can use a large volume of training data. However, because training data can never be guaranteed to be complete, neither is, in general, the model. Also, generalization from observed state transitions means the model may permit some transitions that do in fact not occur in the environment. Note that although we term them "actions", state transitions permitted by the base model are not necessarily actions taken by the agent. They can also represent events that occur spontaneously, or as delayed consequences of the agent's actions.

The second step occurs on-line, when novelty is encountered. We detect novelty by the recurring presence of state transitions that are not possible in the base model; we call such transitions *uncovered*. Because the base model cannot be guaranteed to be complete, some uncovered state tran-

sitions may occur also without novelty, i.e., there is some noise. However, because novel behaviors are persistent, uncovered state transitions caused by novelty are far more likely to be repeated as we observe more of the agent's interaction with the world, and we use this fact to distinguish when novelty appears. The repeated uncovered state transitions also become the basis for our novelty characterization. We create new actions, covering the novel behaviors, from these transitions, by a process similar to that which creates the base model.

Learning action models in physical domains is challenging because of the great variety of realizations of an action that can occur. For example, the action of kicking a ball into a goal (shown in Figure 1, left) can be taken from a myriad of different positions, with different ball trajectories, ending at different locations in the goal. Although what time a state transition occurs is ambiguous because state properties such as the relative positions of objects can change by arbitrarily small quantities, only changes that are somehow significant should be considered a transition to a different state. To address this problem, we abstract the observation of the world state into a set of qualitative spatial relations (QSRs) between observed objects.

QSRs represent relations between objects in space using qualitative terms. They represent intuitive, human-understandable concepts, such as $disconnected$, which means two objects appear separate from each other. There is a large number of formally defined QSRs (Randell, Cui, and Cohn 1992; Van de Weghe et al. 2005; Dylla et al. 2017; Renz and Nebel 2007; Cohn and Renz 2008), which cover most of the spatial aspects relevant to our world, such as relative object location, distance and movement. QSRs have also previously been used to model physical states and state transitions (Delafontaine, Cohn, and Van de Weghe 2011; Duckworth, Hogg, and Cohn 2019).

The action models we generate are defined by their typed parameters, precondition and effects, and can be expressed as a planning domain, e.g., in the Planning Domain Definition Language (PDDL) (McDermott et al. 1998), using the QSRs as predicates. An example is shown in Figure 1 (right). We make one departure from standard PDDL, by allowing for effects that assign an arbitrary relation to one of the spatial aspects. For example, in the action in Figure 1, the (center of the) ball can end up with any direction to the (center of the) goal. This is simply a shorthand for a larger number of possible actions.

Contributions The novelty, and advantage, of our approach is that it is able to compute a characterization of observed novel behavior in the environment at a qualitative level of abstraction, and to do so automatically, in an unsupervised manner.

We show that different novelties mostly give rise to highly distinct characterizations, with very little overlap in their novel actions. We also propose a new method to create the base model, combining neighborhood and hierarchical clustering to group observed state transitions likely to be caused by the same action. We show that this combination yields a better trade-off between the model's coverage, validity and size than other methods. Therefore, our approach is also relevant to applications that require automated generation of a world representation, without considering novelty.

2 Background and Related Work

In this section, we first review the literature on generating action models to characterize novelty and applying QSRs to represent actions. We then describe the qualitative spatial calculi that we use.

2.1 Action Model Extraction for Novelty Characterization

Most AI research into novelty has focused on detection, rather than characterization. For example, Sabokrou et al. (2018) proposed a Generative Adversarial Network (GAN) for anomaly and outlier detection in images, and Yahaya, Lotfi, and Mahmud (2019) used several machine learning models to classify novel behaviors in daily activities based on human-selected features, e.g. duration.

There is a long history of research on action model generation in AI. Pasula, Zettlemoyer, and Kaelbling (2007) learn models of stochastic actions in a simulated physical Block Stacking game, but assume that the actions taken (not just their consequences) are observed. Konidaris, Kaelbling, and Lozano-Perez (2015) and Andersen and Konidaris (2017) learn models of stochastic actions that unfold over time (termed "options"), in video games. Recently, researchers have attempted unsupervised/semi-supervised action model generation. Miglani (2020) apply deep reinforcement learning to produce action models from descriptions in natural language. Asai and Fukunaga (2017) and Asai and Muise (2020) propose an unsupervised approach using autoencoders to derive a representation of states and cluster state transitions into actions. Several researchers have studied learning action models from observation of symbolic actions and states (Amir and Chang 2008; Cresswell, McCluskey, and West 2013; Aineto, Jimenez, and Onaindia 2018).

2.2 Action Representation by Qualitative Spatial Relations

QSRs have been used to recognize action from video in an automatic way (Sridhar, Cohn, and Hogg 2011a; 2011b; Dubba et al. 2015). Tayyub et al. (2014) use both qualitative and quantitative spatio-temporal representations as joint features to recognize daily activities. Young and Hawes (2015) combined four kinds of qualitative spatial calculi to demonstrate the movements of agents in a soccer game simulator, like $kick$ or $shoot$. Duckworth et al. (2016) and Duckworth, Hogg, and Cohn (2019) extracted qualitative spatial relations like qualitative distance relation from robotic movement observations, and combined them as features for unsupervised clustering algorithms such as K-means (Yuan and Yang 2019), and Latent Semantic Analysis (LSA) (Deerwester et al. 1990). However, while these works use QSR features to represent states, they are used only to recognize known actions. Without a description specifying the preconditions and effects of actions, actions can not be instantiated and explained.

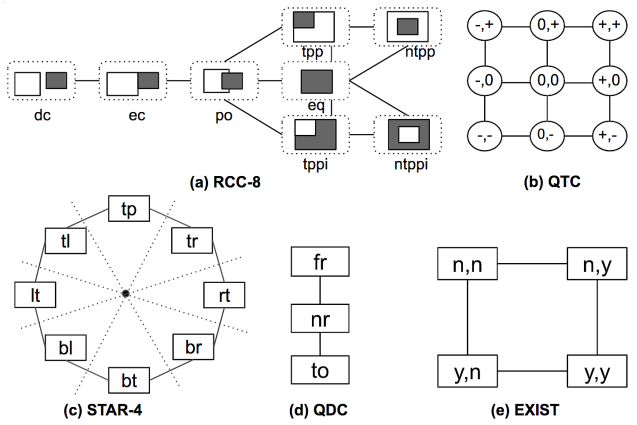


Figure 2: (a) RCC-8: topology relations; (b) QTC: movement relations; (c) STAR-4: direction relations; (d) QDC: distance relations; (e) EXISTS: existence relations. Adjacent (connected) relations are are conceptual neighbors.

2.3 Qualitative Spatial Relations

In this paper, states will be described as QSRs between objects. The QSRs are drawn from four calculi, corresponding to four essential spatial aspects of any physical domain: topology, movement, direction and distance.

RCC-8 The Region Connection Calculus (RCC) (Randell, Cui, and Cohn 1992) describes topological relations between regions. RCC-8 consists of 8 relations: dc (disconnected), ec (externally connected), po (partially overlapping), eq (equal), tpp (tangential proper part), $tppi$ (tangential proper part inverse), $ntpp$ (non-tangential proper part) and $ntppi$ (non-tangential proper part inverse) as shown in Figure 2 (a).

QTC Relations in the Qualitative Trajectory Calculus (QTC) (Van de Weghe et al. 2005) express if one object is approaching $(-)$, moving away from $(+)$, or stationary relative to (0) another object. For example, $(+,+)$ indicates two objects are moving away from each other. There are 9 QTC relations as shown in Figure 2 (b).

STAR-4 STAR-4 (Renz and Mitra 2004) consists of qualitative direction relations. Given two objects, as shown in Figure 2(c), the 2D space is uniformly divided into 8 cones by 4 dotted lines with the first object in the center and the direction relation is decided by which cone the second object locates in. The 8 cones stand for 8 direction relations: tp (top), tr (top right), rt (right), br (bottom right), bt (bottom), bl (bottom left), lt (left) and tl (top left).

QDC The Qualitative Distance Calculus (QDC) (Clementini, Di Felice, and Hernández 1997) indicates distance between objects. As shown in Figure 2 (d), we use the three relations fr (far), nr (near) and to (touching).

Conceptual Distance The conceptual distance between two QSRs can be defined by *conceptual neighborhood diagrams* (Van de Weghe and De Maeyer 2005; Dondrup et al. 2015). In such diagrams, shown in Figure 2, nodes are relations and edgeconnect conceptual neighbors. Two QSRs are conceptual neighbors if there is no other intermediate relation in a consecutive change. For example, in a scenario where two touching objects move away from each other, their QDC relation changes from to to nr and then to fr. Thus nr is a conceptual neighbor to both fr and to while to is not a neighbor of fr. Note, however, that all steps of the change may not be observed: if the objects are moving fast enough, we may see it as a transition from to to fr in consecutive video frames. We will use the conceptual distance as a (dis-)similarity measure between QSRs to cluster similar state transitions.

3 Proposed Approach

An overview of our approach is shown in Figure 3. The inputs are sequences of unlabelled frames (from videos) of the environment without novelty, for base model generation, and with novelty, for detection and characterization. Qualitative states are created by detecting and classifying objects in each frame, and computing their QSRs. Classification assigns each object a type, based on its appearance (e.g. a ragdoll is of type "cat"). If objects of a novel type appear, we have of course no knowledge of that type, but we assume that the classifier can recognize different objects with the same appearance as belonging to the same (novel) type. We do not assume that object detection, tracking or classifcation is perfect, although if they are too unreliable the quality of the generated model will suffer. Because the four QSR calculi we use are made up of binary relations, a state can be decomposed into a collection of object-pair states. State transitions are created from consecutive object-pair states where a change in at least one qualitative aspect occurs.

To create the base model, we cluster the extracted state transitions. Clustering is done in two stages, as explained later in this section, and results in a set of clusters of similar state transitions. From each cluster we create one action, which is representative of the transitions in that cluster.

In the novelty detection step, observed state transitions are matched against actions in the base model; a state transition that is not matched by any action is said to be *uncovered*. Uncovered state transitions that occur repeatedly indicate novelty, once they exceed a certain threshold. From these repeated uncovered state transitions (RUSTs), we extract a set of novel actions that characterize the novel behavior.

3.1 State and State Transition

Our qualitative state representation is made up of the QSRs of four calculi, namely RCC-8, QTC, STAR-4 and QDC, and one additional property which describes a change in object existence (EXIST). We refer to each of these five as an *aspect* of the state. Because the relations in each calculus are mutually exclusive and exhaustive, and binary, each pair of objects satisfies exactly one relation from each calculus, and a state is the union of the pair-wise states of all object pairs. The possible existence relations are EXIST $= \{(n,n),(n,y),(y,n),(y,y)\}$, i.e., it is the cross product of a binary property of each object. We represent it as a relation for uniformity with the other aspects. Non-existent

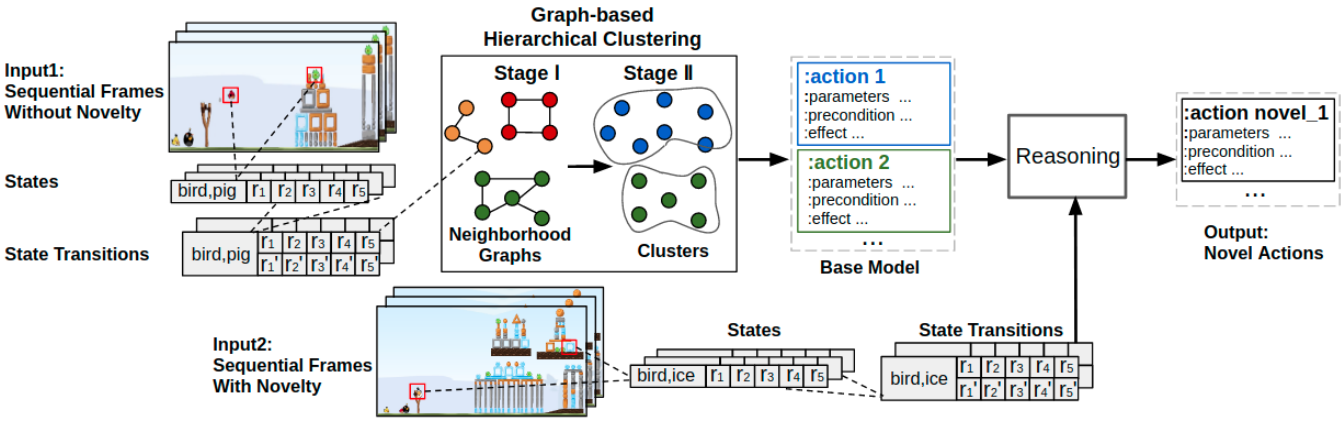


Figure 3: An overview of our proposed solution for the problem of novelty characterization by its novel actions in physical domains. The input is two sets of game-playing videos (with/without novelty) and the output is the novel actions (i.e., uncovered state transitions). Our approach consists of 4 procedures: (1) two sets of states and corresponding state transitions are extracted from both inputs; (2) state transitions are grouped into clusters by our two-stage graph-based hierarchical clustering approach: first, constructing neighborhood graphs, where each node represents a state transition and state transition with similar relation changes are connected; second, merging similar neighborhood graphs into larger clusters; (3) A base model that includes a set of actions is generated from clusters; and (4) select repeated uncovered state transitions from Input 2 as the novel actions.

objects are of course not detected, and are not represented in the state; a change in existence from y to n means the object is about to disappear (is not present in the next frame) and a change from n to y means the object has just appeared (was not present in the previous frame).

Definition 1 (Object-Pair State)**.** *An* object-pair state *is a 2-tuple $s = (op, R)$, where $op = ((o_1, t_1), (o_2, t_2))$ is a pair of typed objects, and $R = (r_1, r_2, r_3, r_4, r_5)$ is the 5-tuple of relations, one for each aspect, that hold between o_1 and o_2, i.e., $r_1 \in$ RCC-8, $r_2 \in$ QTC, $r_3 \in$ STAR-4, $r_4 \in$ QDC, $r_5 \in$ EXIST.*

For each frame, we extract one object-pair state for each pair of detected objects. Automatic state extraction can be done by object detection algorithms like Fast-RCNN (Dias et al. 2020) and QSR computation tools like QSRLib (Gatsoulis et al. 2016). A state transition occurs where the object-pair states referring to the same pair of objects in two consecutive frames differ in at least one aspect.

Definition 2 (State Transition)**.** *Given a sequence of sets of object-pair states $S_1, S_2, \ldots$, where S_i is the set of object-pair states extracted from frame i, a* state transition *is a pair of object-pair states $(s = (op_s, R_S), s' = (op_{s'}, R_{s'})$ such that: (1) $s \in S_i$ and $s' \in S_{i+1}$ for some i; (2) $op_s = op_{s'}$; and (3) $R_s \neq R_{s'}$, We refer to s and s' as the* before *and* after *states of the transition.*

In the following, we will use a slightly different representation of state transitions: instead of a pair of object-pair states, (s, s'), we describe a transition as $((t_1, t_2), r_1\text{-}r'_1, \ldots, r_5\text{-}r'_5)$, where (t_1, t_2) are the types of the object pair and r_i and r'_i the relation of the i-th aspect that holds between the objects in the before state s and after state s', respectively. That is, once we have identified state transitions, the individual objects' identities are no longer important; we regard it as a transition of the objects' types. We refer to the pair r_i-r'_i as the transition in aspect i. Note that r_i may equal r'_i for some i, since a state transition requires only that there is a change in at least one aspect.

3.2 Graph-based Hierarchical Clustering

The clustering of state transitions proceeds in two stages. In the first stage, we construct a neighborhood graph, based on conceptual similarity, and form initial clusters from the connected components of this graph. In the second stage, we iteratively merge pairs of clusters that satisfy certain conditions to form larger clusters. At both stages, we cluster only state transitions with the same pair of object types. The complete clustering procedure is shown in Algorithm 1.

Neighborhood Graph Construction Due to the qualitative abstraction, and because state transitions are identified with pairs of object types, not objects, it is common to observe the same state transition more than once in any data set of sufficient size. Hence, we first group all identical state transitions together. These initial sets can be represented as pairs of a state transition and a repetition count, (st, n_{st}).

The neighbourhood graph is an undirected graph whose nodes are the unique state transitions. Two state transitions $st_1 = (otp_1, r_{1,1}\text{-}r'_{1,1}, \ldots, r_{1,5}\text{-}r'_{1,5})$ and $st_2 = (otp_2, r_{2,1}\text{-}r'_{2,1}, \ldots, r_{2,5}\text{-}r'_{2,5})$ are connected iff $otp_1 = otp_2$ and $d(st_1, st_2) = \sum_{i=1}^{5}(d_i(r_{1,i}, r_{2,i}) + d_i(r'_{1,i}, r'_{2,i})) = 1$, where $d_i(r_{1,i}, r_{2,i})$ is the conceptual distance between relations $r_{1,i}$ and $r_{2,i}$ in the i:th aspect (i.e., the length of the shortest path between them in the corresponding conceptual distance graph shown in Figure 2). In other words, transitions are connected in the neighbourhood graph iff they differ in exactly one aspect, and in that aspect either their before or after state relations are conceptual neighbours. The initial clusters are the connected components of this graph.

The first clustering stage improves efficiency by reducing

the number of initial clusters for the next stage. It can also improve model generalization, since it may cluster some transitions not considered mergeable in the next stage.

Hierarchical Agglomerative Clustering Hierarchical agglomerative clustering starts from an initial set of clusters, and iteratively select a most similar pair of clusters to be merged into one, until no more mergers are possible (Rokach and Maimon 2005).

In our application of this idea to clustering state transitions, we impose two conditions that a pair of clusters must meet to be merged: (1) their sets of state transitions must refer to the same object type pair; and (2) for at least δ of the aspects, a fraction τ of transitions in their union must have the same before and after relation. The parameters $\delta \in \{0, \ldots, 5\}$ and $\tau \in [0, 1]$ control how much the actions generated from the clusters are permitted to generalize from the observed state transitons. To state this more precisely, we need the following definitions:

Definition 3. *Let C be a set of state transitions, and $i \in \{1, \ldots, 5\}$. The* representative transition in aspect i for C *is a relation pair r_i-r'_i in aspect i that occurs most frequently in C, taking into account repetitions. If two or more relation pairs in aspect i are tied for highest frequency, one of them is chosen arbitrarily.*

The agreement in aspect i of C *is the number of state transitions in C whose relation pair in aspect i is the representative transition in that aspect for C divided by the size of C, again taking into account repetitions.*

We denote the agreement in aspect i of C by $\eta_i(C)$.

Definition 4. *Two sets of state transitions C_1 and C_2 are* mergeable*, under parameters $\delta \in \{0, \ldots, 5\}$ and $\tau \in [0, 1]$, iff (1) all state transitions in $C_1 \cup C_2$ have the same object type pair, and (2) there are at least δ different aspects $i_1, \ldots, i_\delta$ such that $\eta_i(C_1 \cup C_2) \geq \tau$ for each i in $i_1, \ldots, i_\delta$, i.e., the agreement in each of those aspects of $C_1 \cup C_2$ is at least τ.*

Definition 4 tells us which clusters may be merged, and therefore also provides the stopping condition for the hierarchical clustering process (when there are no more mergeable clusters). It remains to specify which pair of mergeable clusters are selected in each iteration, i.e., the similarity measure. Given two mergeable candidate clusters C_1 and C_2, their similarity $\theta(C_1, C_2)$ is the average agreement in the δ aspects that have the highest agreement. The cluster pair with the highest similarity score is selected to be merged.

3.3 Base Model Formation

The output of clustering is a set of clusters; each cluster is a set of state transitions. By construction, all state transitions in each cluster have the same object type pair. We generate one action from each cluster. How this is done is described in this section. The resulting set of actions is our base model.

Representative State Transition Given a cluster C, we form a single representative state transition, $RST_C = (otp_C, r_1\text{-}r'_1, \ldots, r_5\text{-}r'_5)$, where otp_C is the object type pair common to all state transitions in C and r_i-r'_i is the representative transition in aspect i for C if $\eta_i(C) \geq \tau$. If $\eta_i(C) < \tau$, the representative state transition has a special value $\star$ in place of a relation pair for aspect i. The meaning of this value is that any transition is considered possible in this aspect.

Algorithm 1 Graph-based Hierarchical Agglomerative Clustering.

```
procedure G-HAC
    # Stage 1: Neighbourhood graph clustering
    # Collect unique state transitions with counts:
    nodes(G) = {(st_1, n_{st_1}), ..., (st_n, n_{st_n})}
    for (st_i, n_{st_i}), (st_j, n_{st_j}) ∈ nodes(G) do
        if st_i ≠ st_j ∧ d(st_i, st_j) = 1 then
            connect (st_i, n_{st_i}) and (st_j, n_{st_j}) in G.
    C_S = connected_components(G) # initial clusters
    # Stage 2: Hierarchical clustering
    # mergeable^{δ,τ}(C_S) denotes cluster pairs that are
    # mergeable under params δ and τ (cf. Definition 4)
    while ∃C_i, C_j ∈ mergeable^{δ,τ}(C_S) do
        # Select pair with highest similarity:
        C_i, C_j ← argmax{θ(C_i, C_j) |
                          C_i, C_j ∈ mergeable^{δ,τ}(C_S)}
        C_S = (C_S \ {C_i, C_j}) ∪ {C_i ∪ C_j}
    return C_S
```

An example is shown in Figure 4. The cluster consists of 5 state transitions (note that the third and fourth row are two repetitions of the same state transition). The agreements in each of the five aspects are $1, 0.8, 0.8, 0.6$ and 0.8 respectively. Assuming $\tau = 0.7$, we obtain the representative state transition $((\text{bird}, \text{pig}), dc\text{-}dc, (0,0)\text{-}(-,0), bl\text{-}lt, \star, (y,y)\text{-}(y,n))$.

Actions An action consists of three components: parameters, precondition and effect. The action generated from a cluster C has two parameters, whose types are given by the common object type pair of the cluster. The action's precondition and effect are determined by the before and after relations of the representative state transition RST_C. When RST_C has a relation pair r_i-r'_i in aspect i, the relation in the before state, r_i, becomes an action precondition, and the relation in the after state, r'_i, becomes an effect if it is different from r_i. This follows the planning convention that persistence of facts is implicit (McDermott et al. 1998). When RST_C has an arbitrary change, $\star$, in aspect i, the action has no corresponding precondition, and an effect that allows any one of the relations in this aspect to become true. This can be seen as a shorthand, representing one action for each possible relation that can hold in the after state. The action created in the example above is also shown in Figure 4.

The predicates used by these actions are simply one for each possible relation in each of the five aspects that make up our qualitative state representation.

3.4 Novelty Characterization

The output of the first step is the base model: a set of actions, A_0, that describe the possible state transitions in the environment without any novelty. In the second step, we

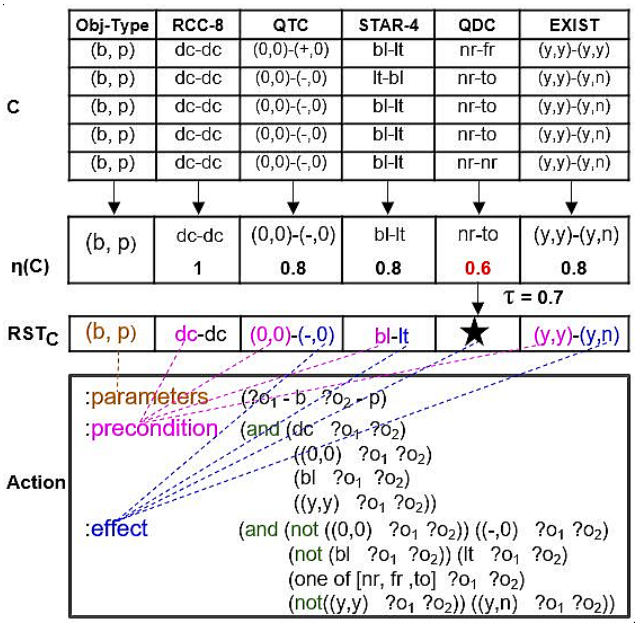


Figure 4: Example of how an action is created from a cluster. There are 5 state transitions in the cluster C as shown in the table. These state transitions share the same object type pair $(\text{bird}, \text{pig})$. Below the table is shown the most frequent relation pair for each aspect and the corresponding agreement ($\eta(C)$). The representative state transition (RST_C) also has the most frequent transition in each aspect, except for the distance aspect where it has an arbitrary change effect (marked by a $\star$) because the agreement in this aspect in C is below the threshold $\tau = 0.7$. Finally, the generated action.

observe new state transitions, as the agent interacts with the world, and attempt to match them to actions in this set.

Definition 5. *A state transition (s, s') for the typed object pair $((o_1, t_1), (o_2, t_2))$ is* covered *by action a iff (1) t_1 and t_2 are the types of a's parameters; (2) the before state s satisfies a's precondition; and (3) a's effects can be applied to s in a way that yields the after state s'.*

A state transition is covered by a set A of actions iff it is covered by some action $a \in A$.

If an action has an arbitrary change effect in an aspect, we can choose any of the possible relations for this aspect in the after state; hence the wording of condition (3) above.

A state transition not covered by A_0 is said to be *uncovered*. Because the base model is learned from a finite set of observations, it is not perfect, and hence there will normally be some uncovered state transitions also when observing the environment without novelty. However, because novelties in a physical environment cause a persistent change to the behavior of the environment, or some types of objects in it, we claim that *repeated* uncovered state transitions (RUSTs) will be more frequent when a novelty is introduced, compared to the non-novel environment, and that this difference is often sufficient to detect novelty. We demonstrate this claim holds for most of the novelties in our case study in Section 5.

The RUSTs are also the basis for generating the characterization of the novelty once detected. From these we create a set of *novel actions*, following the same process as described in Section 3.3. Note, however, that we do not apply any clustering to the RUSTs before generating the novel actions.

4 Experiment 1: Base Model Generation

We conduct two sets of experiments: First, we evaluate how well the generated base model describes the non-novel environment. We compare our method with three other methods from the literature, and evaluate the impact of the δ and τ parameters on the model. In the second set of experiments (Section 5), we introduce novelties and measure how well our approach can detect and characterize them.

4.1 Dataset

Our testbed is a classical physics-based video game – Angry Birds. In this game, the player uses a slingshot to shoot birds at structures made up of different kinds of building blocks, with the aim of destroying all pigs to clear a level. The game can be described as a "physics-based puzzle": the behavior of objects as they fly, collide, and so on is calculated by an internal physics simulation, and solving a level requires an element of logical thinking. Playing Angry Birds has been a challenge for AI agents since 2012 (Renz et al. 2015).

We use the Science Birds open source clone of the game[1] (Ferreira and Toledo 2014), since this allows to introduce novel behaviors. We extract objects and the properties needed to compute the qualitative state in each frame from the game engine directly, instead of indirectly from the rendered video frame. Thresholds for qualitative distances (QDC relations) were set based on the distribution of all object pair distances in the training data set, as follows: touching $to \in [0, 10]$, near $nr \in [10, 50]$, and far $fr \in [50, \infty]$.

Training Data is the set of qualitative state transitions collected from 170 game levels, each played 10 times. To generate training data exhibing a broad range of possible object behaviors, we use a random game-playing agent to play these levels. This agent shoots each bird randomly, without any strategy or target. This means each time the agent replays a game level, different actions are likely to happen, which helps capture more infrequent actions.

To limit the size of the experiment, we focus on actions related to birds. These are the most interesting from the agent's perspective, since it is the birds that the player controls. This means we keep only the state transitions in which at least one of the two objects is of the type "bird", resulting in 6233 state transitions in total in the training data set.

Test Data is extracted from playing another 30 unseen game levels, each 3 times. These are played by an agent that is more goal-directed, targetting the pigs. The agent's strategy is not sophisticated: it selects which pig to shoot at randomly, ignoring obstacles. Hence, it is called the "naive" agent. Nevertheless, it is a better representative of how an intelligent agent may act than the random agent used in training. Only state transitions in which at least one of the two

[1] https://gitlab.com/aibirds/sciencebirdsframework/-/tree/release/alpha0.4.1

objects is of type "bird" are kept, resulting in 520 state transitions in the test data set.

4.2 Evaluation Metrics

The performance of the base model is evaluated on three criteria: (1) the validity of its actions, meaning the degree to which they represent state transitions that can actually happen in the game; (2) coverage on the test set, i.e., on unseen game levels without novelty; and (3) size, meaning the number of actions. A perfect model would achieve 100% validity and coverage, with a small number of actions. Actions corresponding exactly to state transitions in the training set are sure to be valid, but their coverage outside the training set may be low. Introducing some generalization increases coverage on the unseen test set, but also risks lowering validity.

Action Validity Since we cannot know the set of all state transitions that are possible in the environment, we approximate it with the union of the training and test sets; call this set S_U. Actions without arbitrary change effects represent only one state transition, and thus have a validity score of either 1 (if found in S_U) or 0 (if not). An action a that has an arbitrary change effect in one or more aspects can result in a set of state transitions, all $(otp_a, r_1\text{-}r'_1, \ldots, r_5\text{-}r'_5)$ such that either r_i is in the precondition of a and r'_i in the effect of a or equal to r_i, or a has an arbitrary change effect on aspect i. The validity of the action is the fraction (in $[0, 1]$) of these state transitions found in S_U. The validity score of the base model is the average validity of all its actions.

Coverage on Test Data Coverage on test data measures how well the base model describes game levels outside the training set, i.e., how general it is. This is measured simply as the fraction of state transitions in the test set that are covered (cf. Definition 5) by the base model's action set.

4.3 Baseline Approaches

We compare our proposed graph-based hierarchical clustering approach (G-HAC) with three baseline approaches: (1) the Cube-space Action Auto-Encoder (AAE), proposed by Asai and Muise (2020); (2) the widely-used clustering method K-means and (3) Latent Semantic Analysis (LSA). We also apply hierarchical agglomerative clustering (HAC) alone as a comparison. Without the neighborhood graph-based clustering stage, the initial clusters each contain only repetitions of a unique state transition.

AAE is a neural network that produces action models by learning the object distribution changes as state transitions. State transitions are represented as numeric vectors by embedding the pixel distribution of consecutive game state images. With the state transitions as the input to AAE, the result is which action the input state transition belongs to, where the total number of actions K is predefined. K-means and LSA were both successfully used by Duckworth, Hogg, and Cohn (2019) to classify human activities based on the similarity of their QSR changes. Given a predefined number of clusters K (the same as in AAE), they both return K clusters (i.e., activities).

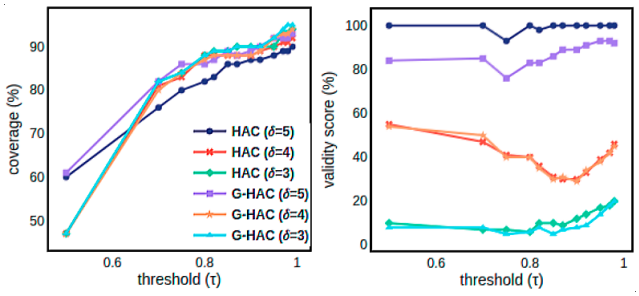


Figure 5: Coverage on test data (left) and validity (right) of the base models generated by G-HAC and HAC with different parameter values. $\delta \in \{3, 4, 5\}$, $\tau \in [0.5, 0.99]$.

4.4 Experiment Results and Analysis

Both G-HAC and HAC are tested with different combinations of $\tau \in$ [0.5, 0.99] and $\delta \in \{3, 4, 5\}$. Figure 5 shows the results. Increasing τ boosts coverage for both methods, while varying δ does not affect coverage much. This is because as τ increases, fewer clusters are mergeable, and more actions are generated. For the same parameter settings, the model generated by G-HAC has slightly better coverage than the one generated by HAC alone. Validity, on the other hand, is strongly affected by δ. When $\delta < 5$, hierarchical clustering can generate actions with arbitrary change effects, and all the possible state transitions represented by those actions are rarely present in the combined training and test set.

The three other methods all require a number of clusters K as an input parameter. We use values of K equal to the number of clusters generated by G-HAC with all different δ and τ values. Figure 6 shows the coverage–validity trade-off achieved by the models generated by all five methods, under all tested parameter settings. The closer a model is to the top right corner, the better its performance. Models generated by AAE and LSA have visibly lower validity scores, though their coverage is high. In other words, these methods generate models that over-generalize.

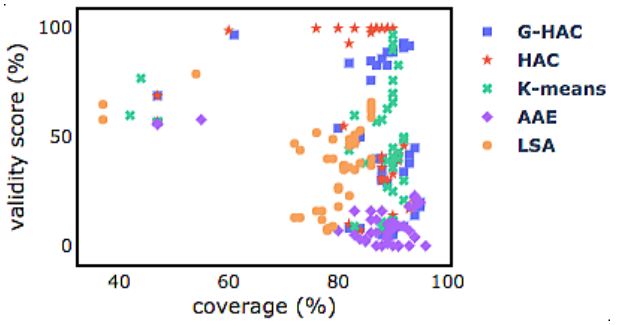


Figure 6: Comparison of coverage and validity achieved by all models generated using G-HAC, HAC, AAE, K-means and LSA.

Table 1 shows the size, coverage and validity of the best (i.e., closest to the top right corner in Figure 6) model generated by each method. While no method dominates in all three performance criteria, the G-HAC method offers the best trade-off. The model generated by AAE has the high-

	G-HAC	HAC	K-means	AAE	LSA
No. of actions	593	662	663	166	496
Coverage (%)	93.0	90.4	90.4	94.1	91.0
Validity (%)	95.2	100.0	94.6	18.7	66.6

Table 1: Size, coverage and validity of the best model generated by each method. These correspond to the points that are closest to the top right corner in Figure 6.

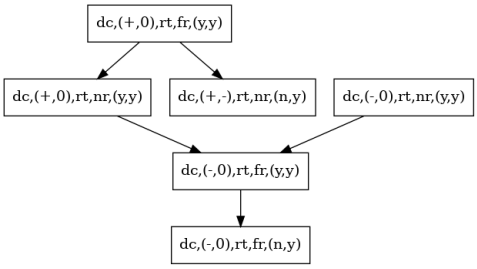


Figure 7: Part of the model of interactions between "red bird" and "fat ice rect". The aspects, from left to right, are RCC8, QTC, STAR4, QDC and EXISTS.

est coverage, but it is only slightly higher than that of the G-HAC model and its validity is vastly lower.

Clustering with G-HAC and action generation take around 20 seconds. HAC alone takes about 10 times longer.

4.5 Using the Model

The generated model cannot be used directly to plan the actions of an Angry Birds playing agent, because it does not distinguish which actions are actions of the agent and which are uncontrollable events that happen as delayed consequences. However, plan existence can be used as a predictor of what future states can occur. As an example, Figure 7 shows a small part of the model of interactions between objects of the "red bird" type and the "fat ice rect" type. It can be seen that, for example, these particular starting conditions never result in the ice block being destroyed.

5 Experiment 2: Novelty Detection and Characterization

We select the best model generated by G-HAC, as shown in Table 1, as the base model for testing novelty detection and characterization.

5.1 Novelties

For this experiment, we introduce five novelties, of three different kinds, in the Angry Birds game. Novelties 1–3 are changes to the properties of some object type, which changes its behavior; novelty 4 is a "global" change – rotating the screen 180 degrees – which causes a change in apparent behavior of almost all objects; novelty 5 introduces a new object type, with a different appearance but which is behaviorally the same as the red bird type in the non-novel game. These correspond to levels 2, 3 and 1 of the classification proposed by DARPA (2019). Figure 8(a) shows an example of one action in the standard game, and how this action may change with each of the novelties (b–f).

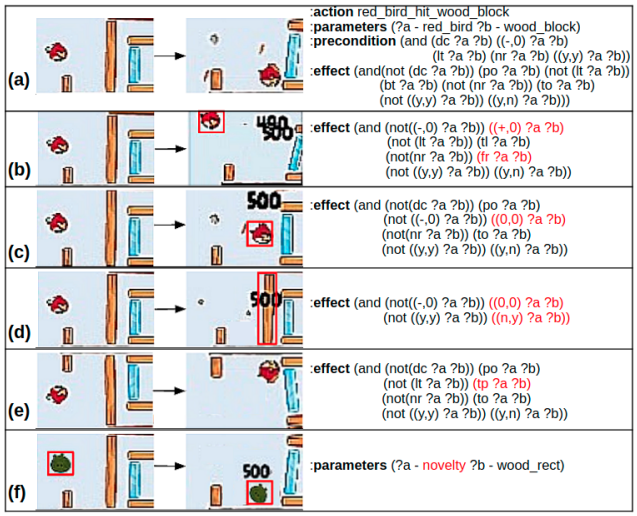


Figure 8: (a) A red bird hits a wood block in the environment without novelty; right: the corresponding action model. (b) Novelty 1: Increasing the bounciness of the red bird causes it to bounce back after hitting the wooden block. QTC and QDC in the effect change accordingly. (c) Novelty 2: Increasing the linear drag of the red bird causes it to fall slowly and more vertically. QTC in the effect is different. (d) Novelty 3: Increasing health points of the wooden block, a single hit from the bird is not sufficient to destroy it. The EXIST relation changes in the effect. (e) Novelty 4: Rotating the view of the environment 180 degrees. The STAR-4 relation in the effect changes to tp. (f) Novelty 5: Introducing a new object type with different appearance but the same properties as the red bird.

5.2 Novelty Detection

Detecting novelty is a prerequisite for characterization. To determine the detection ability of our approach and the number of levels needed to detect these novelties, we insert the novelties into the 30 unseen game levels used to generate the test data set. For each novelty, and also for the unseen levels without novelty, we randomly sample $n_l = \{5, 10, 20\}$ of the levels, observe the naive agent play each of them three times, and collect all state transitions which involve an object of type red bird. We then calculate the number of repeated uncovered, by the base model, state transitions (RUSTs) in this set. We repeat the experiment 100 times for each novelty (including no novelty) to obtain a distribution of the number of RUSTs, which is shown in Figure 9.

Without novelty, the number of RUSTs is always small (< 10). Novelties cause a greater number of RUSTs. Novelties 4 and 5 are clearly detectable even at $n_l = 5$, while novelties 1–3, which change the behavior of only one object type, are harder to detect. Nevertheless, at $n_l = 20$ we can detect all novelties except #2 with a high recall rate without false positives, by setting the decision boundary at maximum number of RUSTs seen in the non-novel environment (shown by the dashed line in the graph).

Novelty detection and characterization time is linear in data size (length of the frame sequence), and at $n_l = 20$

takes around 40 seconds, over 99% of which is extracting object-pair states and transitions.

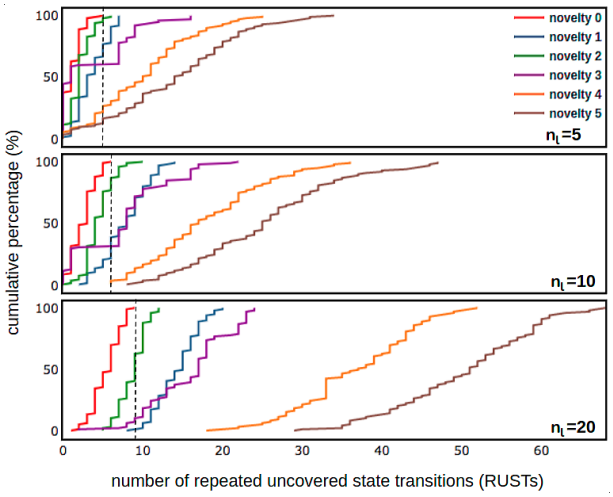


Figure 9: Cumulative distribution of the number of repeated uncovered state transitions (RUSTs) for each novelty (including the non-novel environment as #0) under different number of game levels observed (n_l) over 100 trials. The vertical dashed line in each graph marks the maximum number of RUSTs in non-novel levels.

5.3 Novelty Characterization

We create characterizations of each of the novelties from the RUSTs collected after observing $n_l = 20$ game levels. The characterization of novelty k is a set of novel actions, NA_k, each corresponding to one of the RUSTs.

We evaluate the specificity of these characterizations by measuring their overlap. The overlap of two novel action sets, $O(NA_i, NA_j)$, is the fraction (in $[0, 1]$) of actions in NA_i that are also found in NA_j, i.e., $|NA_i \cap NA_j|/|NA_i|$. Note that this is not symmetric. The overlap is an indication of the risk that given the characterization of novelty j, subsequent observations of game levels with novelty i would not be recognized as a different novelty, or, in other words, the degree to which novelty i fails to stand out.

	$\mathbf{NA}_1$	$\mathbf{NA}_2$	$\mathbf{NA}_3$	$\mathbf{NA}_4$	$\mathbf{NA}_5$
$\mathbf{NA}_1$		0.22 (0.06)	0.02 (0.03)	0	0
$\mathbf{NA}_2$	0.36 (0.11)		0	0	0
$\mathbf{NA}_3$	0.02 (0.03)	0		0	0
$\mathbf{NA}_4$	0	0	0		0
$\mathbf{NA}_5$	0	0	0	0	

Table 2: The average overlap rate $OR(NA_i, NA_j)$. NA_i is the row and NA_j is the column. In parenthesis, the standard deviation.

Table 2 shows the average overlap, and standard deviation, obtained over the 100 repeats of the experiment. Except for novelties 1 and 2, the overlaps are zero or close to zero, indicating that the novel actions are highly distinct. Novelties 1 and 2, which both change properties of red bird objects, produce some visually similar behaviors, and therefore a greater overlap. This can be seen Figure 8(b) and (c), as the position of the bird in one can be an intermediate state on course to the other. This indicates that our action-based novelty characterization can not only distinguish novelties, but also see similarities between them.

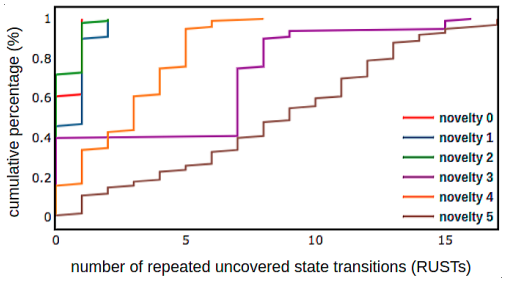


Figure 10: Cumulative distribution of the number of repeated state transitions which are uncovered by $A_0 \cup NA_1$. For each novelty, these are taken over the 10 remaining unseen game levels.

To further examine the quality of the characterization of each novelty, we test how well it can help to detect the same novelty and still distinguish other novelties in new, unseen game levels. We take the 10 remaining unseen game levels, after selecting $n_l = 20$ that the characterization is based on, and collect the state transitions from three plays each of these in each environment version, i.e., with each novelty as well as without novelty. From each of these sets, we calculate the number of repeated state transitions that are uncovered by $A_0 \cup NA_i$. Figure 10 shows the result for novelty 1. As can be seen, the number of RUSTs relative to $A_0 \cup NA_1$ for novelty 1 as well as no novelty are now small (smaller in fact than what we observed in the graph for $n_l = 10$ in Figure 9). This means that after we incorporate the novel actions into the model, new, unseen levels with novelty 1 no longer appear novel; in other words, the characterization recognizes new instances with the same novelty. The other novelties, except #2, still stand out, and thus would be detected as different novelties after learning of novelty 1. Novelty 2, which is the most difficult to detect and also has a high overlap with novelty 1, does not.

6 Conclusion

We propose an approach to automatically generating a characterization of novelty in physical environments, based on the novel observable behavior (actions) it creates. We demonstrated, using the physics-based puzzle game Angry Birds as our testbed, that our approach can detect and distinguish most of the novelties.

The more precisely the base model describes the non-novel environment, the more accurate our novelty detection and characterization will be. Thus, refining the base model with information not just about what actions are possible, but also what sequences of them can occur, how likely they are, and quantitative delays between them is a direction for future work. We also seek to evaluate how well the novelty characterization can guide adaptation of an agent's behavior, for example, using the extended model to predict possible action outcomes.

Acknowledgments

This work was supported by the DARPA SAIL-ON program under contract W911NF-20-2-0002. The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

References

Aineto, D.; Jimenez, S.; and Onaindia, E. 2018. Learning STRIPS action models with classical planning. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 399–407.

Amir, E., and Chang, A. 2008. Learning partially observable deterministic action models. *Journal of AI Research* 33:349–402.

Andersen, G., and Konidaris, G. 2017. Active exploration for learning symbolic representations. In *Advances in neural information processing systems*, 5009–5019.

Asai, M., and Fukunaga, A. 2017. Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary. *arXiv preprint arXiv:1705.00154*.

Asai, M., and Muise, C. 2020. Learning neural-symbolic descriptive planning models via cube-space priors: The voyage home (to strips). *arXiv preprint arXiv:2004.12850*.

Boult, T.; Grabowicz, P.; Prijatelj, D.; Stern, R.; Holder, L.; Alspector, J.; Jafarzadeh, M.; Ahmad, T.; Dhamija, A.; Li, C.; et al. 2020. A unifying framework for formal theories of novelty: Framework, examples and discussion. *arXiv preprint arXiv:2012.04226*.

Clementini, E.; Di Felice, P.; and Hernández, D. 1997. Qualitative representation of positional information. *Artificial intelligence* 95(2):317–356.

Cohn, A. G., and Renz, J. 2008. Qualitative spatial representation and reasoning. *Foundations of Artificial Intelligence* 3:551–596.

Cresswell, S. N.; McCluskey, T. L.; and West, M. M. 2013. Acquiring planning domain models using LOCM. *Knowledge Engineering Review* 28(2):195–213.

DARPA, Defense Sciences Office. 2019. Broad agency announcement: Science of artificial intelligence and learning for open-world novelty (SAIL-ON). https://sam.gov/api/prod/opps/v3/opportunities/resources/files/109e04254a2dda31bf362f46fe47bce8/download?api_key=null&status=archived&token=.

Deerwester, S.; Dumais, S. T.; Furnas, G. W.; Landauer, T. K.; and Harshman, R. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41(6):391–407.

Delafontaine, M.; Cohn, A. G.; and Van de Weghe, N. 2011. Implementing a qualitative calculus to analyse moving point objects. *Expert Systems with Applications*.

Dias, L. O.; Bom, C. R.; Faria, E. L.; Valentín, M. B.; Correia, M. D.; Márcio, P.; Marcelo, P.; and Coelho, J. M. 2020. Automatic detection of fractures and breakouts patterns in acoustic borehole image logs using fast-region convolutional neural networks. *Journal of Petroleum Science and Engineering* 107099.

Dondrup, C.; Bellotto, N.; Hanheide, M.; Eder, K.; and Leonards, U. 2015. A computational model of human-robot spatial interactions based on a qualitative trajectory calculus. *Robotics* 4(1):63–102.

Dubba, K. S.; Cohn, A. G.; Hogg, D. C.; Bhatt, M.; and Dylla, F. 2015. Learning relational event models from video. *Journal of Artificial Intelligence Research* 53:41–90.

Duckworth, P.; Gatsoulis, Y.; Jovan, F.; Hawes, N.; Hogg, D. C.; and Cohn, A. G. 2016. Unsupervised learning of qualitative motion behaviours by a mobile robot. In *AAMAS*, 1043–1051. AAMAS.

Duckworth, P.; Hogg, D. C.; and Cohn, A. G. 2019. Unsupervised human activity analysis for intelligent mobile robots. *Artificial Intelligence* 270:67–92.

Dylla, F.; Lee, J. H.; Mossakowski, T.; Schneider, T.; Delden, A. V.; Ven, J. V. D.; and Wolter, D. 2017. A survey of qualitative spatial and temporal calculi: algebraic and computational properties. *ACM Computing Surveys (CSUR)* 50(1):7.

Ferreira, L., and Toledo, C. 2014. A search-based approach for generating angry birds levels. In *Proceedings of the 9th IEEE International Conference on Computational Intelligence in Games*, CIG'14.

Gatsoulis, Y.; Alomari, M.; Burbridge, C.; Dondrup, C.; Duckworth, P.; Lightbody, P.; Hanheide, M.; Hawes, N.; Hogg, D.; Cohn, A.; et al. 2016. Qsrlib: a software library for online acquisition of qualitative spatial relations from video.

Konidaris, G.; Kaelbling, L. P.; and Lozano-Perez, T. 2015. Symbol acquisition for probabilistic high-level planning. In *Proceedings of the 24th International Conference on Artificial Intelligence*.

Langley, P. 2020. Open-world learning for radically autonomous agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 13539–13543.

Li, D.; Scala, E.; Haslum, P.; and Bogomolov, S. 2018. Effect-abstraction based relaxation for linear numeric planning. In *IJCAI*, 4787–4793.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. Pddl–the planning domain definition language–version 1.2. *Yale Center for Computational Vision and Control, Tech. Rep. CVC TR-98-003/DCS TR-1165*.

Miglani, S. 2020. Nltopddl: One-shot learning of pddl models from natural language process manuals. In *Working Notes of the ICAPS'20 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS'20)*. ICAPS.

Pasula, H. M.; Zettlemoyer, L. S.; and Kaelbling, L. P. 2007. Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research* 29:309–352.

Randell, D. A.; Cui, Z.; and Cohn, A. G. 1992. A spatial logic based on regions and connection. *KR* 92:165–176.

Renz, J., and Mitra, D. 2004. Qualitative direction calculi with arbitrary granularity. In *PRICAI*, volume 3157, 65–74.

Renz, J., and Nebel, B. 2007. Qualitative spatial reasoning using constraint calculi. In *Handbook of spatial logics*. Springer. 161–215.

Renz, J.; Ge, X.; Gould, S.; and Zhang, P. 2015. The Angry Birds AI competition. *AI Magazine*.

Rokach, L., and Maimon, O. 2005. Clustering methods. In *Data mining and knowledge discovery handbook*. Springer. 321–352.

Sabokrou, M.; Khalooei, M.; Fathy, M.; and Adeli, E. 2018. Adversarially learned one-class classifier for novelty detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3379–3388.

Sridhar, M.; Cohn, A. G.; and Hogg, D. C. 2011a. Benchmarking qualitative spatial calculi for video activity analysis. In *Proceedings ijcai workshop benchmarks and applications of spatial reasoning*, 15–20. Leeds.

Sridhar, M.; Cohn, A. G.; and Hogg, D. C. 2011b. From video to rcc8: exploiting a distance based semantics to stabilise the interpretation of mereotopological relations. In *COSIT*, 110–125. Springer.

Tayyub, J.; Tavanai, A.; Gatsoulis, Y.; Cohn, A. G.; and Hogg, D. C. 2014. Qualitative and quantitative spatio-temporal relations in daily living activity recognition. In *ACCV*, 115–130. Springer.

Van de Weghe, N., and De Maeyer, P. 2005. Conceptual neighbourhood diagrams for representing moving objects. In *International Conference on Conceptual Modeling*, 228–238. Springer.

Van de Weghe, N.; Kuijpers, B.; Bogaert, P.; and De Maeyer, P. 2005. A qualitative trajectory calculus and the composition of its relations. In *International Conference on GeoSpatial Sematics*, 60–76. Springer.

Yahaya, S. W.; Lotfi, A.; and Mahmud, M. 2019. A consensus novelty detection ensemble approach for anomaly detection in activities of daily living. *Applied Soft Computing* 83:105613.

Young, J., and Hawes, N. 2015. Learning by observation using qualitative spatial relations. In *AAMAS*, 745–751. AAMAS.

Yuan, C., and Yang, H. 2019. Research on k-value selection method of k-means clustering algorithm. *J—Multidisciplinary Scientific Journal* 2(2):226–235.