Spatial Representation and Reasoning in an Architecture for Embodied Agents

Pat Langley
Institute for the Study of Learning and Expertise
2164 Staunton Court, Palo Alto, CA 94306 USA

Edward P. Katz
Stanford Intelligent Systems Laboratory
Stanford University, Stanford, CA 94305 USA

Abstract

In this paper, we review PUG, a cognitive architecture for embodied agents, and report extensions that let it represent and reason about spatial relations. The framework posits graded concepts that are grounded in perception and integrates symbolic reasoning with continuous control. After describing the architecture, we discuss how an extended version supports places, encoded as virtual objects defined by distances to reference entities, and reasons about them as if they were visible. We demonstrate PUG's control of a simulated robot that approaches targets and avoids obstacles in a simple two-dimensional environment. In closing, we discuss related research on agent architectures, robotic control, and spatial cognition, along with our plans to extend the framework's capabilities for spatial representation and reasoning.

1 Introduction

Research on cognitive architectures aims to develop unified theories of the mind that explain and support a broad range of abilities (Newell, 1990; Langley, Laird, & Rogers, 2009). Such computational theories include postulates about what mental structures and processes remain constant across different domains and tasks. Just as a building architecture indicates which elements are fixed (e.g., floors, rooms) and which ones are mutable (e.g., furniture), so a cognitive architecture specifies which facets of an intelligent agent remain the same and which ones change across settings and over time. For this reason, work in the area typically makes strong assumptions about the representation of content and the mechanisms that operate over it, but, in the interests of generality, not about the content itself.

Most cognitive architectures import many of their core ideas from psychological theories. They had their origin in accounts of human problem solving (Newell & Simon, 1972), which focused on problem-space search, but work soon incorporated ideas about memory, attention, and learning. Architectures usually include assumptions that: dynamic short-term memories are distinct from stable long-term stores; both memories contain modular elements that are cast as discrete symbol structures; long-term structures are accessed through pattern relational matching against short-term elements; processing revolves around discrete cycles that involve retrieval, selection, and application of knowledge structures; and cognition – both performance and learning – involves the dynamic composition of these mental structures. This paradigm has been used to explain many human abilities, including reasoning, language processing, and problem solving. A cognitive architecture usually comes with a programming language whose syntax and interpreter reflects its assumptions about representation and processing, providing an infrastructure for intelligent systems.

Research in this area has been active for nearly five decades. Early efforts focused on production systems (Neches, Langley, & Klahr, 1987), which encoded long-term knowledge in terms of condition-action rules that matched and modified elements in working memory, but the field has evolved to include a wide range of frameworks. Well-known architectures include ACT-R (Anderson, 1993), Soar (Laird, 2012), and ICARUS (Choi & Langley, 2018), but the field includes many alternative designs. As Langley (2017) notes, the paradigm has produced many successes but also has important limitations. One drawback is that a concern with generality has led researchers to avoid making commitments about some topics. In particular, cognitive architectures seldom incorporate strong assumptions about spatial representation and reasoning, even when used to develop systems that operate in simulated environments (e.g., Jones et al., 1999) and that control physical robots (e.g., Trafton et al., 2013). Thus, although architectures have been used to *model* embodied agents, they have not incorporated these ideas into their core *theoretical* commitments.

In this paper, we report our progress toward addressing this oversight by incorporating strong assumptions about spatial representation and processing into an existing framework. We start by reviewing PUG, an existing architecture for embodied agents that retains many ideas from earlier frameworks but that grounds symbolic relations in quantitative descriptions and that combines symbolic reasoning with continuous control. We will see that the current theory supports some aspects of spatial cognition, but only those based on the agent's immediate perceptions. In response, we report extensions to the framework that let it represent and use long-term knowledge about *places*. We illustrate the augmented PUG's assumptions and abilities with examples from a simulated two-dimensional robotic domain. In addition, we discuss its relation to previous work on spatial reasoning and outline plans for further research.

We should reiterate that our objective is to extend an existing cognitive architecture to support aspects of spatial cognition. Thus, our main contributions will be theoretical and they should be judged by the criteria usually applied to architectural research. These include the ability to provide qualitative explanations of target behaviors, but also features like consistency, generality, and elegance. The purpose is not to demonstrate new abilities or to show the approach outperforms previous ones, but rather to provide a broad theory of intelligent agency that takes seriously the importance of spatial cognition. Applied roboticists may not appreciate our concern with strong architectural commitments and experimental psychologists may not understand our emphasis on qualitative accounts, but we believe that architectural researchers will value our contributions to the literature on unified theories of the mind.

2 The PUG Architecture for Embodied Agents

PUG (Langley et al., 2016) is a cognitive architecture for embodied agents that extends ideas from earlier frameworks.[1] The theory retains the classic assumptions that short-term and long-term memories are distinct, that they contain modular symbol structures, that long-term elements are accessed through pattern matching, and that mental processing occurs in discrete cycles. However, it also introduces additional constraints about representation and processing, including:

[1] PUG expands to *Planning with Utilities and Goals*, which reflects the framework's concern with task planning that is guided by both symbolic goals and numeric utilities associated with these structures. Here we focus on the most recent version of the architecture, PUG/C (Langley & Katz, 2022), which extends previous incarnations (Langley et al., 2016, 2017) to support continuous control. Earlier papers provide details about task-level plan generation, execution, and monitoring, which we will not address here, as they have few direct implications for spatial cognition.

- Symbolic structures relations are *grounded* in quantitative descriptions of physical situations;
- All short-term elements are *instances* of structures defined in long-term memory; and
- Cognition involves *cascaded processing*, with higher levels using results from lower levels.

PUG borrows all three assumptions from ICARUS (Choi & Langley, 2018), but it moves beyond this precursor in important ways. In this section, we review the framework's cognitive structures and then examine the computational mechanisms that operate on them, using examples that illustrate its basic capacities for spatial representation and reasoning.

2.1 PUG's Representation of Knowledge

Any intelligent system depends on stored expertise to drive its reasoning, decision making, and external behavior. In cognitive architectures, this content resides in one or more long-term memories, which are reasonably stable in that they change only slowly through learning. The PUG architecture incorporates four distinct types of long-term structures that encode different forms of such knowledge:

- *Concepts*, which define categories that specify classes of entities and relations among them;
- *Skills*, which specify how the agent should act to achieve a given target concept;
- *Processes*, which predict how some aspects of the environment will change over time; and
- *Motives*, which indicate the utility of particular relations conditioned on beliefs about the situation.

PUG uses these elements for conceptual inference, reactive control, heuristic evaluation, and plan generation. As discussed later, PUG operates in discrete cycles, so it uses these structures to encode content for a given step and applies them repeatedly to produce behavior over time. We will focus on concepts and skills in this paper, as they have the most relevance to spatial cognition. We will touch on processes and motives only in passing, as they mainly impact the architecture's higher-level mechanisms. Langley and Katz (2022) describe the latter and their use in some detail.

PUG incorporates three theoretical postulates about concepts, two of which elaborate on those noted above. First, they *ground* symbols in terms of *percepts*, which describe quantitative attributes of objects in the environment. Second, inferred *beliefs* are always instances of these defined concepts that refer to perceived or imagined objects. Finally, concepts are *graded* in that they match against situations to greater or lesser degrees, with each belief having an associated *veracity* that denotes this score. Like other cognitive architectures, PUG's syntax reflects its theoretical commitments. Each concept is encoded by a separate rule, similar to a Prolog clause, that includes a head and a set of antecedents. The head specifies the concept name, a set of arguments, and a set of attributes. Each antecedent includes an entity type, an identifier, and a set of attributes, along with optional Boolean tests on those attributes' values. In addition, each concept has a veracity function, often a conditional one, that specifies how to compute its degree of match and that refers to variables bound in earlier fields.

Table 1 (a) shows two conceptual rules that define relations – *robot-at* and *robot-facing* – between the robot and another object. The *:elements* field describes a set of typed objects, each with an identifier and numeric attribute values, *:binds* assigns values to new variables, and *:veracity* specifies a function for computing the degree of match. Veracity for *robot-at* is one minus the distance *?d* (between object boundaries) divided by ten when *?d* is less than ten and zero otherwise. Veracity for *robot-facing* depends on the agent's angle *?a* to object *?o*, with different values depending on whether *?a* is positive, negative, or zero.[2] Ta-

[2] In these cases, veracity is a piecewise linear function of attribute values, but other relations are possible, such as inverse square.

Table 1: (a) Two PUG concepts for a two-dimensional robot domain, each including a head, a set of observed elements, and a veracity function. Each head and element comprise a predicate and set of attribute-value pairs, with ^ marking attributes and ? denoting variables that match specific symbols or numbers. (b) Four percepts for the domain that describe perceived objects and four beliefs inferred from them. Each includes a predicate and a set of attributes and values, the latter observed for percepts and inferred for beliefs.

```
(a) ((robot-at ^id (?r ?o) ^distance ?od)
     :elements   ((robot ^id ?r ^radius ?rr)
                  (object ^id ?o ^distance ?od ^radius ?or))
     :binds      (?d (- ?od (+ ?rr ?or)))
     :veracity   ((cond ((> ?d 10.0) 0.0)
                        (t (- 1.0 (/ ?d 10.0))))))
    ((robot-facing ^id (?r ?o) ^angle ?a)
     :elements   ((robot ^id ?r) (object ^id ?o ^angle ?a))
     :veracity   ((cond ((< ?a 0.0) (- 1.0 (/ ?a -180.0)))
                        ((> ?a 0.0) (- 1.0 (/ ?a 180.0)))
                        ((= ?a 0.0) 1.0))))
```

```
(b) (robot ^id R1 ^radius 0.15 ^move-rate 0.0 ^turn-rate 0.0) [1.0]
    (object ^id O1 ^distance 2.0 ^angle 0.0 ^radius 0.4) [1.0]
    (object ^id O2 ^distance 4.123 ^angle 14.032 ^radius 0.4) [1.0]
    (object ^id O3 ^distance 6.0 ^angle 0.0 ^radius 0.4) [1.0]
    (robot-at ^id (R1 O1) ^distance 2.0) [0.847]
    (robot-at ^id (R1 O2) ^distance 4.123) [0.622]
    (robot-facing ^id (R1 O1) ^angle 0.0) [1.0]
    (robot-facing ^id (R1 O2) ^angle 14.032) [0.688]
```

ble 1 (b) also shows eight beliefs from the same domain, the first four describing objects the agent perceives in the environment and the others corresponding to inferred beliefs that are instances of defined concepts. Each belief includes a predicate, an identifier that may specify a relation, a set of attribute-value pairs, and a veracity score that falls between zero and one. Beliefs in PUG are always ground instances of defined concepts that relate specific entities.

In a similar way, PUG incorporates four theoretical assumptions about skills, another key type of knowledge structure. First, they *ground* symbols that refer to activities in terms of percepts, concepts, and their associated numeric attributes. Second, *intentions* are always instances of defined skills that refer to perceived or imagined objects. Third, each skill specifies a graded *target concept* that it aims to achieve. Finally, it includes equations for *control attributes* that are functions of the *mismatch* between the target concept and the situation. As a result, each intention has an associated *activation* that is one minus the veracity of its target. Again, PUG's syntax for skills reflects these theoretical assumptions. This shares some features with concepts but also introduces new ones. The head specifies the skill's name and arguments, while the antecedents include a set of elements described by an entity type, an identifier, and attribute values, along with optional Boolean tests. However, it also includes fields for the target concept and control equations that determine how to act in the environment.

Table 2 (a) shows two skills – *move-to* and *turn-to* – that an agent can use to approach an object. Each takes two arguments (the robot and the object) and an *:elements* field that refers to the latter's attributes, such as distance and angle. The first skill also includes two Boolean tests, which state that the angle to the object

Table 2: (a) Two skills for the two-dimensional robot domain, each of which includes a relational head, a set of observed entities, Boolean tests, control equations, and a target concept. As with concepts, each entity is described by a predicate and a set of attribute-value pairs, with ^ indicating an attribute and ? denoting a variable that matches against a specific symbol or number. The first skill influences the control attribute *move-rate*, while the second affects *turn-rate*. (b) Four intentions for approaching the target object $O3$ and for avoiding the obstacles $O1$ and $O2$, the first two being instances of the skills in (a). The bracketed numbers denote each intention's activation, which is one minus the veracity of its matched target concept.

```
(a) ((move-to ?r ?o)
     :elements ((robot ^id ?r ^turn-rate ?t) (object ^id ?o ^angle ?a))
     :tests    ((> ?a -90) (< ?a 90))
     :control  ((robot ^id ?r ^move-rate (* 0.3 $MISMATCH)))
     :target   ((robot-at ^id (?r ?o))))
    ((turn-to ?r ?o)
     :elements ((robot ^id ?r) (object ^id ?o ^angle ?a))
     :control  ((robot ^id ?r ^turn-rate (* 5.0 (sign ?a) $MISMATCH)))
     :target   ((robot-facing ^id (?r ?o))))
```

```
(b) (move-to R1 O3) [0.675]
    (turn-to R1 O3) [0.812]
    (avoid-on-left R1 O1) [0.793]
    (avoid-on-right R1 O2) [0.0]
```

falls between -90 and 90 degrees. Both skills also have a *:target* field that specifies a desired relation, *robot-at* and *robot-facing*, respectively, and a *:control* field with a conditional function for computing values of control attributes, *move-rate* and *turn-rate*. This function includes the symbol *$MISMATCH*, which denotes one minus the veracity of the target belief. Table 2 (b) presents two intentions that correspond to instances of these skills and two others for avoiding obstacles, although they show only the instantiated heads and activation levels, rather than their entire matched structures. We will discuss the mechanisms that generate such intentions shortly.

As noted earlier, PUG also includes two other forms of long-term structures. Processes describe how the values of control and state attributes influence the current values of state attributes, such as how turning changes the agent's angle to a given object. Motives specify how to compute the utility of a particular belief as a function of the agent's beliefs about the current situation. Their syntax is similar to that for concepts and skills, but differ slightly in accordance with their purposes. Again, we will not discuss either of them at length in this paper because they come into play at the architecture's higher processing levels and they are not central to how it addresses spatial cognition.

2.2 Cascaded Processing in PUG

Like other cognitive architectures, PUG operates in discrete cycles that match long-term knowledge structures against short-term elements to produce new instances of the latter, as well as produce behavior in the agent's environment. An important difference is that the framework relies on five levels that involve distinct levels of temporal resolution. These include:

- *Belief processing*, which matches conceptual knowledge against the agent's perceptions and beliefs to draw inferences and thus generate new or updated beliefs;
- *State processing*, which matches skills against perceptions and beliefs to calculate values for intentions' control attributes and matches processes against perceptions and control values to predict effects;
- *Reactive execution*, which generates motion trajectories either in the environment, using feedback control, or in the agent's mind, using mental simulation of such reactive control;
- *Motion planning*, which carries out heuristic search through a space of possible motion trajectories, generated by sets of intentions, guided by utilities associated with their states; and
- *Task planning*, which searches for candidate sequences of motion plans, again guided by their utilities, that achieve some or all of the agent's top-level goals.

PUG applies these levels in a cascaded manner, with each one using mental structures produced by those below it. For instance, state processing draws on beliefs generated by conceptual inference, whereas execution uses results from state processing to generate trajectories over multiple cycles. These operate at different temporal scales, in that multiple cycles of belief processing occur for each step of state processing and multiple cycles of the latter occur for each execution or mental simulation run. We will discuss motion and task planning only in passing here, as spatial reasoning takes place primarily in the lower three layers.

At the first level of processing, conceptual inference derives beliefs that are consistent with the agent's perceptions or with its predictions. This involves matching antecedents of conceptual rules against elements like *(robot ^id R1 ^radius 0.1)* and *(object ^id O1 ^distance 6.3 ^angle 0.0 ^radius 0.2)* to infer beliefs like *(robot-at ^id (R1 O1) ^distance 6.0)*. The mechanism is similar to that in production systems (Neches et al., 1987) and logic programming languages like Prolog (Clocksin & Mellish, 1981). For each matched rule, PUG instantiates the head, computes values for numeric attributes, and adds the resulting belief, using the *:veracity* field to compute its belief's degree of match. Once the architecture has handled all concepts in this way, it applies the process recursively until it generates the full deductive closure of the rules and percepts.[3] Together, the resulting beliefs describe the current state as a point in N-dimensional space, with one dimension for each belief and its position specified by veracity scores.

Figure 1 illustrates the calculation of veracity scores for the two concepts in Table 1 for a scenario that involves two percepts: *(robot ^id R1 ^radius 0.1)* and *(object ^id O2 ^distance 6.3 ^angle 45.0 ^radius 0.2)*. The conceptual rule for *robot-at* matches against these two descriptions, with *?r* binding to *R1*, *o* to *O2*, *?od* to 6.3, *?rr* to 0.1, *?or* to 0.2, and *?d* to 6.0. The first condition of the veracity field applies, giving $(- \ 1.0 \ (* \ 0.1 \ 6.0)) = 0.4$ as the score for belief *(robot-at ^id (R1 O1) ^distance 6.0)*. The rule for *robot-facing* matches against the same percepts, in this case binding *?r* to *R1*, *o* to *O1*, and *?a* to 45.0. Here the veracity field's second condition applies, giving $(- \ 1.0 \ (/ \ 45.0 \ 180.0)) = 0.75$ as the score for *(robot-at ^id (R1 O2) ^angle 45.0)*. These inferred beliefs could in turn lead other inference rules to fire, but that does not happen in this simple case. In more complex scenarios that involve other objects, the two concepts would apply for each of them, producing additional beliefs like those in Table 1 (b).

The beliefs produced by conceptual inference form the basis for state processing, the second level, which assumes PUG has a set of intentions (skill instances) that were provided by higher levels or by the system's user. For each such intention I, the architecture: checks whether I's conditions match the current percepts

[3] Clearly this will not scale to complex environments and an important direction for future work is developing an anytime approach to conceptual inference that is guided by utility, like that reported by Asgharbeygi et al. (2005). The architecture should also take advantage of truth-maintenance methods (Stanojevic et al., 1994) to eliminate redundant inferences across cycles.

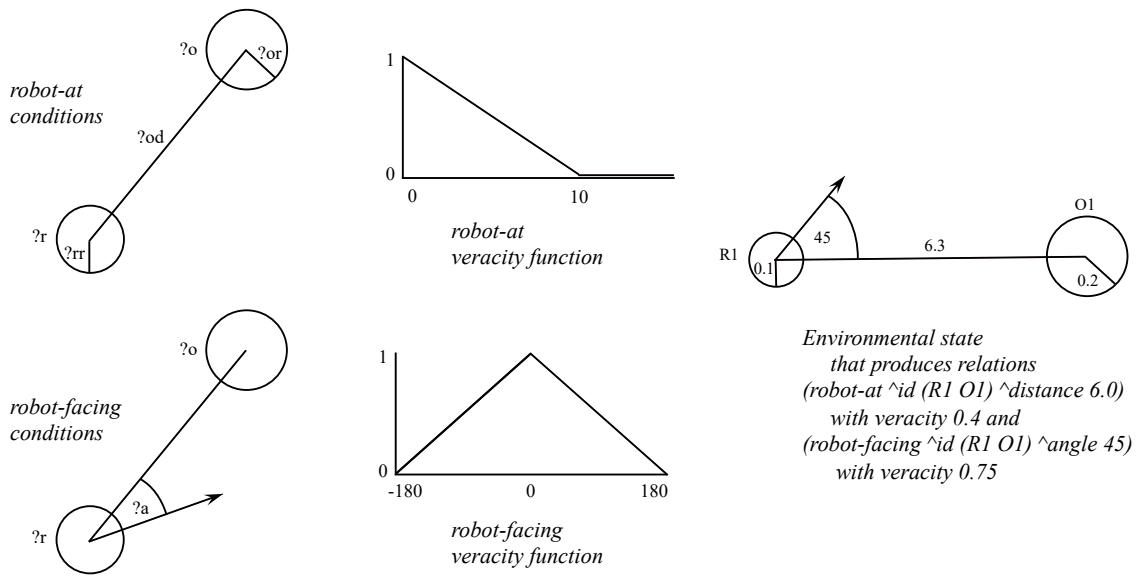

Figure 1: Graphical depiction of the relations (left) involved in the concepts *robot-at* and *robot-facing* from Table 1 (a), with variables for robot *?r*, object *?o*, the distances *?od*, *?rr*, and *?or*, and the angle *?a*; veracity functions (center) for the concepts *robot-at* and *robot-facing*; and instances of these relations (right) that lead to veracity scores for beliefs *(robot-at ^id (R1 O1) ^distance 6.0)* and *(robot-facing ^id (R1 O1) ^angle 45.0)*.

and beliefs; finds the veracity V of I's target belief and computes its mismatch $M = 1 - V$; and, if M is above a threshold, inserts M into I's equations to find values for control attributes. The mismatch provides an error signal that supports a form of feedback control. For instance, we have seen that, given the situation in Figure 1, the target concept for *move-to* would match *(robot-at ^id (R1 O1) ^distance 6.0)* with veracity 0.4, giving the activation 0.6 and the value 0.18 for control attribute *move-rate*. Similarly, the target concept *turn-to* would match *(robot-facing ^id (R1 O1) ^angle)* with veracity 0.75, giving the activation 0.25 and the value 1.25 for control attribute *turn-rate*. If multiple intentions apply on a given cycle, then PUG takes the sum of the resulting control values, much as in potential-field methods (Khatib, 1985).[4]

At the third level, PUG invokes belief and state processing repeatedly to generate a motion trajectory over space and time. When the architecture carries out actions in the environment, this corresponds to reactive control. Because it focuses on the current situation, this mode of operation does not invoke predictive processes, as it does not require expectations about future states, but it does assume that perceptual processing – not part of the architecture – provides a unique identifier for each object across time steps. This is a strong supposition, but object tracking is not the focus of our research. When the agent instead carries out mental simulation of motion trajectories, it relies on process knowledge to produce predicted changes that, when applied repeatedly, serve as the basis for successive belief and state processing. Such a trajectory follows deterministically from an initial state and a set of intentions. For this reason, we refer to both this sequence of states and the set of intentions that produces it as a *motion plan*.

[4]The architecture also sums the effects of matched processes, which we have described elsewhere (Langley & Katz, 2022), to predict the combined influence of these control settings, as well as those of natural events.

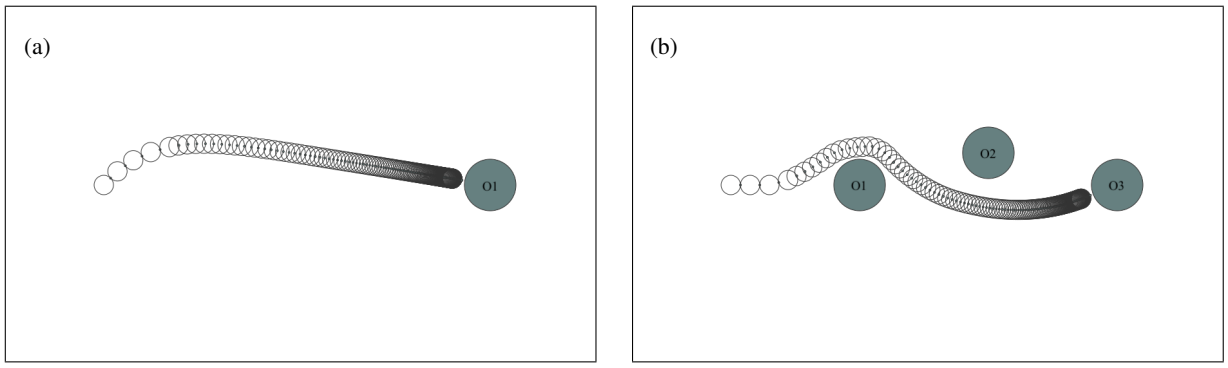

Figure 2: Two scenarios in which a robot must approach a nearby object: (a) when the robot is facing away from the target, it turns and moves forward simultaneously; (b) when two obstacles fall on the robot's path to the target, it veers left to avoid the first and right to evade the second. Traces show the robot's pose at equal time intervals, illustrating that position and orientation change more slowly as it approaches the target.

Figure 2 shows two trajectories that PUG generates in this manner. In the left scenario, the robot is facing slightly away from the target object on the right and adopts two intentions, *(move-to R1 O1)* and *(turn-to R1 O1)*, instances of the skills in Table 2. The architecture repeatedly invokes belief and state processing to turn the robot and move it towards the target, which continues until it reaches the objective, slowing on its approach. In the right scenario, the robot is facing the target object but the direct path is blocked by two obstacles. Here there are four intentions – *(move-to R1 O3)*, *(turn-to R1 O3)*, *(avoid-on-left R1 O1)*, and *(avoid-on-right R1 O2)* – that produce the trajectory depicted. Initially, only the first intention influences behavior, but the third becomes active as the robot nears the first object, causing it to veer left. Once past the obstacle, the robot turns back toward the target but then the next object leads it to turn right. After collision is no longer imminent, the *move-to* and *turn-to* intentions become dominant, taking the robot to the target.

The intentions in this example arise from PUG's fourth level of processing, which carries out heuristic search through a space of motion plans. We will not discuss it in detail here, but, briefly, it considers alternative sets of intentions, each of which deterministically produces a trajectory by repeatedly invoking reactive execution in simulation mode. This mechanism examines the utilities of simulated trajectories to identify problems (e.g., collision with obstacles), add new intentions to address them, and select the most promising trajectory. The utility for each trajectory is the average utility for each state, which in turn are the summed utilities of beliefs in that state, with these scores computed by the agent's motives. As Langley and Katz (2022) report, the fourth level repeatedly invokes simulated reactive execution to guide its search through the space of motion plans, each corresponding to a set of agent intentions.

The fifth processing level generates task plans, each of which comprises a sequence of motion plans with associated trajectories. This employs heuristic search through a space of such plans, calling on motion planning repeatedly to add steps to the current plan and continuing until it achieves as many goals as possible. The architecture invokes utility here as well to guide its search for task plans that achieve the agent's objectives, with scores computed as the average utility of component motion plans. Langley et al. (2016) describe the process of task planning and its reliance on mental simulation, so we will not repeat the details here. Both motion planning and task planning take advantage of spatial inferences and decisions made at the lower levels, so we will focus on the latter in the remaining pages.

3 Encoding and Using Place Knowledge

We have seen that the PUG framework supports the construction of embodied agents that operate in simulated physical environments. This necessarily means that it encodes and uses some forms of spatial information, but, like other cognitive architectures, to date it has made no theoretical commitments along these lines. Instead, PUG developers have introduced assumptions about how agents encode spatial information that are convenient to achieve their ends, but they have not introduced explicit constraints, which means they do not count as elements of the theory. In this section, we present an elaborated version of the architecture that incorporates such postulates. These include assumptions about the representation of agent perceptions, knowledge about places, and processes for making spatial inferences.

We will illustrate these ideas in the same two-dimensional robotics environment discussed earlier in the paper. This is clearly an idealization of the situations encountered by humans and physical robots, who must deal with far more cluttered settings and much more complex shapes, not to mention complicating factors like occlusion. However, as noted previously, our aim is to extend an existing cognitive architecture to deal with spatial cognition, which means that our contribution is best clarified with simple examples. Whether the extended framework can handle more difficult, realistic environments is a question that is best delayed for future work. In Section 5, we discuss some ways such research could move beyond our initial results.

3.1 Perceptual Assumptions

The updated theory introduces a number of statements about how PUG agents characterize their perceptions of the immediate environment. These include assumptions that:

- Percepts describe objects in the agent's environment that are visible to its sensors;
- Each percept is encoded as a set of attributes and their associated values, some of them numeric;
- Spatial attributes specify an object's distance and angle from the agent in egocentric polar coordinates;
- Intrinsic object attributes (e.g., size) remain constant, but spatial attributes may change over time.

These constraints are consistent with the examples presented in the previous section, but we have now elevated them to core principles of the cognitive architecture.

The framework does not rule out other representations of spatial content, such as allocentric encodings, but it requires that they be inferred rather than perceived directly. Neither does the framework take a position on how object-centered descriptions are computed from lower-level perceptions like image pixels or range data. PUG has a stronger commitment to grounded representations than most cognitive architectures, but it does not attempt to explain how percepts are generated. Finally, the theory takes no stance on whether the agent can perceive multi-component objects directly, but we focus here on simple ones that can be described as simple geometric shapes with associated parameters.

The expanded framework does not commit to particular types of entities or their associated intrinsic attributes. The examples in Table 1 (b) describe two-dimensional circular objects, but other shapes like ellipses, rectangles, and polygons are allowed provided they can be described with intrinsic parameters. We have also developed PUG agents that perceive and interact with edges, which are extended one-dimensional objects that can describe road limits and similar boundaries. The common feature is that the architecture describes spatial facets of entities in terms of their egocentric distance and angle to the agent, although it defines these measures differently across entity types. For instance, the distance of a circular object O from

a circular agent A is the length of the line between their center points. In contrast, the distance of an edge E from a circular agent A that is facing E is the length of the line from A's center to its intersection with E.

3.2 Representing Places

Such perceptual structures and the beliefs they produce are sufficient for simple control tasks like avoiding obstacles while moving toward a target object, but not for more complex activities. We know that people organize their spatial knowledge in terms of *places*, if only because we assign names to them like 'kitchen' and 'dining room'. They can also imagine attributes of entities they cannot perceive, such as their relative position. Thus, it is natural to ask how PUG might represent such content. Whenever attempting to replicate an ability in a cognitive architecture, we should first ask whether the existing framework can handle it or whether changes are needed. In the case of place knowledge, the current structures appear adequate.

In fact, we can define a place as a 'virtual' object that is specified by its distance to a set of reference objects, themselves either visible or virtual.[5] For example, the place P in Figure 2 can be defined by its distance to the reference objects X and Y, both of which are perceivable. Two landmarks suffice because the agent's pose provides additional constraints. The corresponding conceptual rule is:

```
((object ^id P ^distance ?d3 ^angle ?a3 ^radius 0.2)
 :elements   ((object ^id X ^distance ?d1 ^angle ?a1)
              (object ^id Y ^distance ?d2 ^angle ?a2))
 :binds      (?d3 (*distance-R-to-V ?d1 ?d2 ?a1 ?a2 3.0 3.0)
              ?a3 (*angle-R-to-V ?d1 ?d2 ?a1 ?a2 3.0 3.0)))
```

This has the same general form as other concepts, but it does not specify a new predicate. Instead, the head refers to the known predicate *object* but introduces a new constant identifier, *P*. The rule also mentions two entities in the *:elements* field, one for each of the named reference objects, along with variables that denote their distances from the virtual entity. Finally, it has a *:binds* field with functional expressions for calculating *P*'s distance and angle from the agent. These invoke the functions $\star$*distance-to-virtual* and $\star$*angle-to-virtual*, which take as arguments the place's distances from the references, in this case 11.0 and 9.17.

Recall that PUG's beliefs are either primitive percepts or instances of defined concepts, and beliefs about places involve both varieties. For example, based on the robot *R*'s distances to objects *X* and *Y* in Figure 2, the corresponding belief would be *(object ^id P ^distance 10.57 ^angle 7.17)*. This takes the form of a primitive percept, even though it is derived from measurements about the agent's relation to other entities. The values for this belief's spatial attributes – distance and angles – will change as the robot moves in the environment, just as it would for visible objects like *X* and *Y*.[6] The rule might also specify values for intrinsic attributes, such as radius, either as constants or in terms of reference objects, that will not change over time. Such percepts lead to inferred beliefs about places, such as *(robot-at ^id (R1 P) ^distance 0.25)*.

3.3 Inferring and Using Place Beliefs

The PUG architecture can use the same mechanism to derive beliefs about defined places as for other relations. On each inference cycle, it finds rules whose conditions match percepts or beliefs already in memory, computes the values for numeric attributes, and adds new beliefs for each rule instantiation. In this case, the

[5] The examples here assume that reference objects are described as points, but the architectural framework allows more complex entities that comprise multiple points or surfaces.

[6] The definition for place *P* has no *:veracity* field, so its associated beliefs will have the default score of one, as with percepts.

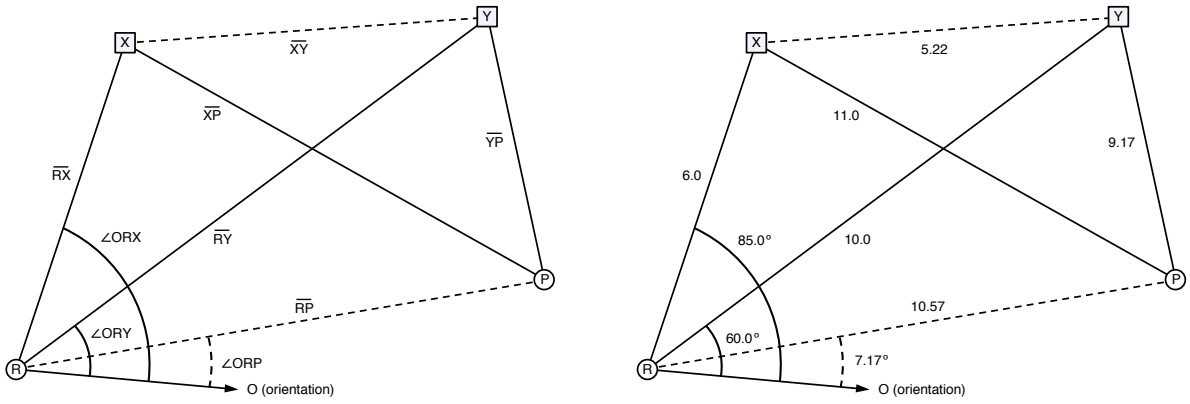

Figure 3: A scenario in which the extended PUG architecture computes the robot R's distance and angle to a place (the virtual object P) based on perceived distances and angles to two reference objects (X and Y). The left diagram shows symbolic notations; the right gives numeric values (not to scale). Solid lines and angles are perceived or given in P's place definition; dashed lines are derived from them.

match process itself is straightforward and depends only on the place's reference objects being present in memory, which means they are visible to the agent or have themselves been inferred. But calculation of a place's distance and angle relies on trigonometric equations that merit elaboration.

Let us return to the diagram in Figure 2, which shows the distance $\overline{RX}$ and $\overline{RY}$ from the robot to reference objects X and Y, as well as the angles $\angle ORX$ and $\angle ORY$ between them and the robot's orientation. Given these quantities and distances $\overline{XP}$ and $\overline{YP}$ from the virtual object and the references, we have

$$\overline{RP} = SAS(\overline{RY}, |SSS(\overline{XY}, \overline{XP}, \overline{YP}) - SSS(\overline{XY}, \overline{RX}, \overline{RY})|, \overline{YP}) \ ,$$

where $\overline{XY} = SAS(\overline{RX}, |\angle ORX - \angle ORY|, \overline{RY})$, and where the equation for the angle is

$$\angle ORP = |\angle ORY - SSS(\overline{RY}, \overline{YP}, \overline{RP})| \ .$$

Here the function $SSS(a, b, c) = acos([(a^2 + c^2) - b^2]/[2 \cdot a \cdot c])$ and the function $SAS(b, A, c) = sqrt([b^2 + c^2] - [2 \cdot b \cdot c \cdot cos(A)])$, which draw on the standard 'Side Side Side' and 'Side Angle Side' formulae from trigonometry.

These functions let PUG use the conceptual definition for place P to calculate the agent's distance and angle to P from its egocentric relations to the reference objects X and Y. Given the situation depicted in the figure, the distance $\overline{RP}$ from the robot R to P is 10.57, whereas its angle $\angle ORP$ is 7.17 degrees. Thus, the inference process generates the belief *(object ^id P ^distance 10.57 ^angle 7.17)*, but again these spatial attributes will vary as the robot moves and as its distances to the landmarks X and Y change. Note that these calculations work as intended only when X is to the left of Y from the agent's perspective; when their relationship is reversed, then their roles in the equations must be reversed.

Once the architecture has produced a belief about the agent's relation to a virtual object like P, it can use this content in higher-level processing. For instance, it can use skills like those in Table 2 to approach P even though it is not visible to the agent, using the derived distance and angle to compute values for control variables. Similarly, if the agent has an intention to avoid P, then it can treat it as an obstacle and use its

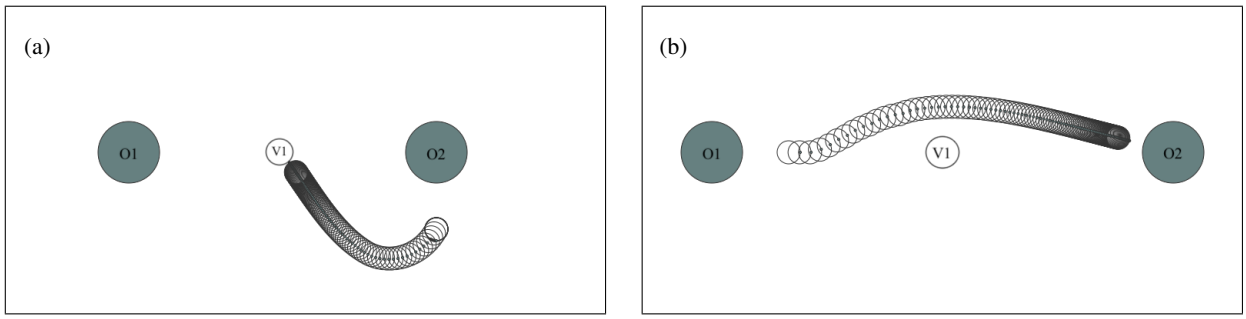

Figure 4: Two scenarios that involve maneuvering a robot with respect to a virtual object. In the left situation, the agent initially faces downward to the left and moves to a place that is defined as midway between two visible objects. In the right situation, it skirts a virtual obstacle that lies on the path to the target object.

derived attributes to evaluate candidate trajectories during motion planning. The system can even use the description for P to compute the angle and distance to other places that refer to it as a reference object. This opens the door to encoding topological networks of places and hierarchical descriptions of space, as we discuss later in Section 5.

Figure 3 shows trajectories that demonstrate the first two of these abilities. The left diagram shows a scenario in which a place $V1$ is defined as midway between two visible objects, $O1$ and $O2$, the robot is initially below $O2$ and faces downward to the left, and has the intentions *(move-to R1 V1)* and *(turn-to R1 V1)*. Using its observed distances and angles to the two reference objects, it turns and moves toward $V1$ until it achieves its objective by reaching the target. The right diagram depicts a scenario in which a virtual obstacle $V1$ is defined as halfway between objects $O1$ and $O2$, the robot begins by facing $O2$ between $O1$ and $V1$, and it has intentions *(move-to R1 O2)*, *(turn-to R1 O2)*, and *(avoid-on-left R1 V1)*. In this case, the agent moves toward $O2$ until it comes close to $V1$, which it treats as an obstacle even though it is imperceptible. The robot veers around the virtual object, after which it continues onward to reach $O2$.

3.4 Evaluating the Extensions

As noted earlier, our objective has not been to demonstrate entirely new capabilities for spatial cognition or to show that our approach is superior to ones that have already appeared in the literature. Rather, our aim has been to extend a cognitive architecture – which aspires to provide a unified theory of the mind – to explain spatial abilities that it could not previously handle. The demonstrations in this section have illustrated how our extensions let the PUG framework encode and use virtual objects in terms of visible ones, which in turn let it specify and reason about places, both of which are spatial abilities it did not possess before. The spatial reasoning is shallow but, when combined with other inference steps, it can produce complex results. A more important question concerns whether these extensions meet the criteria for cognitive architectures that we listed in the introduction – consistency, generality, and elegance.

The extensions are clearly *consistent* with PUG's previous theoretical postulates in that they do not conflict with them in any way. Like other symbols, virtual objects like places are grounded in quantitative descriptions of physical situations, the instances of such objects that appear in mental states are linked to terms defined in long-term memory, and spatial inference fits naturally into the architecture's cascaded processing, with its results feeding into higher levels of cognition. The architecture's new assumptions are

elaborations of those already present rather than replacements of them. These extensions are also *general* in that they should apply to any virtual objects that can be defined in terms of perceivable entities. The examples we have presented were limited to objects that can be described as single points rather than more complex structures, but there is nothing in the theory that forbids the latter.

Finally, the extensions to PUG are *elegant* in that they take advantage of existing structures and processes rather than introducing new types of memory elements or modifying the mechanisms that operate over current ones. Instead, the revised architecture encodes places and other virtual objects in terms of conceptual rules, although it uses them in a novel way, to define new named entities in terms of existing ones rather than to specify new relations in terms of percepts. The processes for inference and control that operate over them also remain unchanged, although they can now introduce and update descriptions of unobservable entities. Thus, the theoretical contribution largely involves stating how places fit into the existing framework, offering a simple, elegant solution to representing and processing spatial content.

4 Related Research

The framework described in the preceding sections brings together ideas from multiple traditions. We have already discussed research on cognitive architectures (Newell, 1990; Langley et al., 2009) and noted that PUG incorporates many classic assumptions from that paradigm, such as separation of long-term memories from short-term content and operation in discrete cycles. However, it introduces the further constraints that all symbols must be grounded in quantitative descriptions of the environment and that processing involves cascaded layers, extending ideas from its precursor, ICARUS (Choi & Langley, 2018). As noted earlier, specific instances of other architectures, such as Soar (Jones et al., 1999) and ACT-R (Trafton et al., 2013), have used grounded encodings, but they do not elevate them to theoretical commitments.

The emphasis on grounded representations has led us to incorporate notions from another paradigm, continuous control (Bennett, 1996), which relies on error signals to modulate the values of control variables. PUG supports this idea by providing skills with target concepts whose mismatches influence their control equations to produce smooth, adaptive behavior.[7] In this way, the architecture retains the notions of relational concepts and skills but imbues them with the continuous character of control systems, extending Newell's vision for unified theories in a new direction. This leads naturally to PUG agents' egocentric encoding of spatial relations to other objects, their behavior in response to targets and obstacles, and their mental simulation of trajectories in specific scenarios. Although we have focused on navigation, Bartels et al. (2013) describe a similar approach to robotic manipulation that incorporates soft constraints, analogous to our target concepts, into control laws. Two key differences are that they involve dynamics rather than kinematics and they use a variety of geometric object features, both of which we should incorporate in future work.

In contrast, some robotics research has addressed continuous settings very differently. A reasonably common approach discretizes the environment by dividing it into grid cells (Moravec & Elfes, 1985), which are widely used for map construction and localization (Yamauchi, Schultz, & Adams, 1998). In a similar spirit, work on route planning and navigation often introduces waypoints that are linked in graphs, as in probabilistic road maps (Kavraki et al., 1996) and rapidly-exploring random trees (LaValle & Kuffner, 2001). Moreover, these typically use allocentric rectangular coordinates, whereas PUG's use of egocentric polar encoding comes closer to robotics work by Kuipers and Byun (1991) and by Yeap (2011), which, like our framework, encode the environment in terms of discrete places. In addition, the architecture's approach to

[7] This unified approach differs from that in Soar (Laird, personal communication), which calls on a PID controller as a subroutine.

control also shares features with Khatib's (1986) use of potential fields. Both allow multiple influences on motion that vary over time, some stronger than others, with their effects being summed. A key difference is that PUG associates fields with intentions, instances of skills that support high-level planning.

Our approach has some overlap with qualitative approaches to robot planning and control. Examples here include Brenner et al.'s (2007) use of potential fields to position objects to satisfy qualitative spatial relations and Wiley et al.'s (2016) use of a qualitative formalism with landmark values to represent states and control rules. However, these efforts invoke multi-step planning to generate sequences of discrete actions rather than PUG's continuous motion plans. Rather, our theory comes much closer to Kuipers' (2004) spatial semantic hierarchy, which associates quantitative control laws with qualitative regions that drive a mobile robot to distinctive states. PUG's definition of places in terms of quantitative perceptual attributes also bears similarities to Kuipers' framework, although his account is more complete. On the other hand, our framework differs sharply from Brooks' (1986) subsumption architecture, which relied on qualitative control knowledge to support reactive behavior but combined its elements in other ways. Moreover, Brooks explicitly rejected the use of explicit spatial representations, including place knowledge.

The architecture's use of logical rules to transform continuous object descriptions into qualitative relations also has precedents in the literature on robotics and agent architectures. For instance, Deeken et al. (2018) use description logics to infer relational descriptions like *left-of* from sensor readings, but their concepts are true or false, rather than true to varying degrees, as in our framework. Another architectural theory that addresses spatial cognition is Shultheiss and Barkowsky's (2011) Casimir, which extracts qualitative relations from image-like encodings, but again these relations are matched in an all-or-none, rather than graded, fashion and the system does not use them to control agent motion. A related idea appears in Laird's (2012) Soar, which has an iconic spatial memory that it links to a relational working memory and that it integrates with processes for problem solving and execution. In summary, our approach to spatial cognition bears similarities to earlier work but combines them in a novel architectural theory.

5 Directions for Future Work

Although our architectural account of spatial cognition shows promise, there are clear limitations that we should address in future work. One drawback lies in the impoverished representation for objects, which is constrained to simple parametric shapes like circles and lines. We could extend these to polygons for two dimensions and to polyhedra for three, encoding entities as collections of vertices and edges or surfaces, then computing distances and angles to them. Generalized cylinders (Binford, 1971) and non-uniform rational basis splines (Piegl, 1991) are alternatives that encode continuous object surfaces. We must also describe complex objects with multiple components, which requires some way to specify the latter's spatial relations. One promising candidate is the Region Connection Calculus (Cohn et al., 1997), which provides a set of qualitatively different relations among shapes, although we may need to augment them with quantitative descriptors. This would let the architecture move beyond stating that the agent is 'at' a point-like place to a greater or lesser degree and describe its relation to the place's boundaries.

Another limitation is that our place definitions require precisely two reference objects, but human places clearly involve richer descriptions. One reason is that only some references are visible at a time, but there may also be variation in their locations. Future versions of PUG should not only support place concepts with a greater number of landmarks, but also alter processing to take advantage of them. The simplest change would use each pair of visible entities to calculate the agent's distance and angle to a virtual object, then take

their means, which would be useful with noisy sensors. However, there are $\binom{n}{2}$ for n references, so a more realistic approach would average a sample of such pairs. Another complication occurs when landmarks have shifted, in which case the system might find a subset of estimates that agree and ignore the anomalies. More generally, we should extend the framework to support places with many objects, some of whose positions vary, as occurs in manipulation tasks for cluttered spaces.

In addition, we should extend our treatment of spatial cognition to dynamic settings in which entities besides the agent move. For example, a PUG vehicle that is driving on a road might define a place that is five units from the right shoulder and 20 units behind the vehicle ahead of it, regardless of the latter's speed. The current architecture can already handle such scenarios, but it cannot define a place at which the robot's path will intersect the trajectory of another object, such as a flying ball. Given processes that describe an object's movement, it can predict both trajectories, but identifying their intersection point requires more sophisticated reasoning. One can also imagine defining a 'place' that is a particular point in space time and specifying skills that let the agent rendezvous with it, or even designating a trajectory with specific start and end times that the agent's movements should follow.

Finally, we should augment the framework to represent and reason not only about the agent's immediate surroundings but also about large-scale space. We have already noted that PUG's conceptual rules can define places in terms of other places, which would constitute, in effect, a topological network. Given a goal for the robot to be at a target place, the task planner would find a sequence of motion plans that takes it there through a series of intermediate places, each adjacent pair sharing at least one reference object. This approach is similar to those reported by Kuipers (2004), Yamauchi et al. (1998), and Yeap (2011), all of which use of a topological network of places for large-scale navigation, although our scheme differs in that the network would be stored implicitly in place definitions, rather than as a separate mental structure.

This should suffice for basic navigation tasks, say within one floor of a building, but the architecture currently supports interplace connections only at a single level of abstraction. We will need something more to specify larger spaces, such as an entire floor or building rather than individual rooms. One response would be to view these as composite objects and to introduce a notation for specifying hierarchical entities in terms of their components. This would also require adapting the architecture's conceptual inference mechanism, which can only apply rules when all conditions match, which in turn means that it can only recognize small-scale places. Some variety of abductive reasoning (e.g., Langley & Meadows, 2019) that can apply rules with missing antecedents appears necessary, along with the ability to introduce default assumptions, which would support top-down predictions about nearby places that are not observable.

6 Concluding Remarks

In this paper, we described PUG, an architecture for embodied agents that combines relational concepts with numeric descriptors and that joins symbolic reasoning with continuous control. We reviewed the architecture's core postulates and clarified their support for basic spatial cognition, illustrating their operation in a two-dimensional robotic setting. We then presented an elaborated version of the theory that makes explicit assumptions about spatial representation and reasoning, including the form of perceptual inputs from the environment. In addition, we discussed how to encode knowledge about places as PUG conceptual rules, how the architecture infers beliefs about them from perceived entities, and how skill execution uses the results to interact with these virtual objects. Examples included moving to a defined target place and avoiding an unseen but known obstacle on a path elsewhere.

We also considered how PUG's theoretical assumptions relate to ideas in earlier research. The framework shares features with other cognitive architectures, but differs in its commitment to embodied agency and spatial cognition. To this end, it incorporates ideas from feedback control, which supports smooth behavior in continuous environments. Moreover, PUG differs from robotic systems that encode space as a discretized grid or network, but it comes closer to ones that adopt egocentric perspectives and continuous representations. Finally, we proposed extensions to the framework that use richer formalisms for entities and regions, allow redundant reference objects in place definitions, define places as space-time points or trajectories, and handle large-scale spatial knowledge as hierarchical mental structures. Taken together, these additions will offer a more complete architectural account of spatial representation and reasoning.

Recall that the objective of our research has not been to support new capabilities for spatial cognition or to demonstrate that our approach is superior to prior ones. Instead, the goal was to extend an existing architectural theory for embodied agents so that it can encode and use knowledge about places. We argued that the extensions we introduced are consistent with the original framework, they hold considerable generality, and they are elegant in that they build upon the theory and elaborate it rather than revise its core postulates. Taken together, these suggest that the new architecture offers a promising springboard for future research on spatial cognition for embodied agents.

Acknowledgements

The research reported in this article was carried out while the first author was at Stanford University's Center for Design Research and it was supported by Grant No. FA9550-20-1-0130 from the US Air Force Office of Scientific Research, which is not responsible for its contents. We thank participants in the Dagstuhl Seminar on Representing and Solving Spatial Problems for useful discussions that contributed to the ideas it reports, as well as the reviewers for their constructive feedback.

References

Anderson, J. R. 1993. *Rules of the mind*. Hillsdale, NJ: Lawrence Erlbaum.

Asgharbeygi, N., Nejati, N., Langley, P., & Arai, S. (2005). Guiding inference through relational reinforcement learning. *Proceedings of the Fifteenth International Conference on Inductive Logic Programming* (pp. 20–37). Bonn, Germany: Springer.

Bartels, G., Kresse, I., & Beetz, M. (2013). Constraint-based movement representation grounded in geometric features. *Proceedings of the IEEE-RAS International Conference on Humanoid Robots* (pp. 547–554). Atlanta, GA: IEEE Press.

Binford, T. O. (1971). Visual perception by computer. *Proceedings of the IEEE Conference on Systems and Control*. Miami, FL: IEEE Press.

Bennett, S. (1996). A brief history of automatic control. *IEEE Control Systems Magazine*, *16*, 17–25.

Brenner, M., Hawes, N., Kelleher, J., & Wyatt, J. (2007). Mediating between qualitative and quantitative representations for task-orientated human-robot interaction. *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence* (pp. 2072–2077). Hyderabad, India.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, *2*, 14–23.

Choi, D., & Langley, P. (2018). Evolution of the ICARUS cognitive architecture. *Cognitive Systems Research*, *48*, 25–38.

Clocksin, W. F., & Mellish, C. S. (1981). *Programming in Prolog*. Berlin: Springer-Verlag.

Cohn, A. G., Bennett, B., Gooday, J., & Gotts, M. M. (1997). Qualitative spatial representation and reasoning with the Region Connection Calculus. *GeoInformatica*, *1*, 275–316.

Deeken, H., Wiemann, T., & Hertzberg, J. (2018). Grounding semantic maps in spatial databases. *Robotics and Autonomous Systems*, *105*, 146–165

Jones, R. M., Laird, J. E., Nielsen P. E., Coulter, K., Kenny, P., & Koss, F. (1999). Automated intelligent pilots for combat flight simulation. *AI Magazine*, *20*, 27–42.

Kavraki, L. E., Svestka, P., Latombe, J.-C., Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, *12*, 566–580.

Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. *Proceedings of the 1985 IEEE International Conference on Robotics and Automation* (pp. 500–550). St. Louis, MO.

Kuipers, B. (2004). An intellectual history of the Spatial Semantic Hierarchy. In M. E. Jeffries & W. K. Yeap (Eds.), *Robotics and cognitive approaches to spatial mapping*. Berlin: Springer.

Kuipers, B. J., & Byun, Y.-T. (1991). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, *8*, 47–63.

Laird, J. E. (2012). *The Soar cognitive architecture*. Cambridge, MA: MIT Press.

Langley, P. (2017). Progress and challenges in research on cognitive architectures. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (pp. 4870–4876). San Francisco, CA: AAAI Press.

Langley, P., Barley, M., Meadows, B., Choi, D., & Katz, E. P. (2016). Goals, utilities, and mental simulation in continuous planning. *Proceedings of the Fourth Annual Conference on Cognitive Systems*. Evanston, IL.

Langley, P., Choi, D., Barley, M., Meadows, B., & Katz, E. P. (2017). Generating, executing, and monitoring plans with goal-based utilities in continuous domains. *Proceedings of the Fifth Annual Conference on Cognitive Systems*. Troy, NY.

Langley, P., & Katz, E. P. (2022). Motion planning and continuous control in a unified cognitive architecture. *Proceedings of the Tenth Annual Conference on Advances in Cognitive Systems*. Arlington, VA.

Langley, P., Laird, J. E., & Rogers, S. (2009). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, *10*, 141–160.

Langley, P., & Meadows, B. (2019). Heuristic construction of explanations through associative abduction. *Advances in Cognitive Systems*, *8*, 93–112.

LaValle, S. M., & Kuffner, J. J. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research*, *20*, 378–400.

Moravec, H., & Elfes, A. (1985). High resolution maps from wide angle sonar. *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 116–121). St. Louis, MO.

Neches, R., Langley, P., & Klahr, D. (1987). Learning, development, and production systems. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development*. Cambridge, MA: MIT Press.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Newell, A., & Simon, H. A. (1976). Computer science as empirical enquiry: Symbols and search. *Communications of the ACM*, *19*, 113–126.

Piegl, L. (1991). On NURBS: A survey. *IEEE Computer Graphics and Applications*, *11*, 55–71.

Shultheiss, H., & Barkowsky, T. (2011). Casimir: An architecture for mental spatial knowledge processing. *Topics in Cognitive Science*, *3*, 778–795.

Stanojevic, M., Vranes, S., & Velasevic, D. (1994). Using truth maintenance systems: A tutorial. *IEEE Expert*, *9*, 46–56.

Trafton, J. G., Hiatt, L. M., Harrison, A. M., Tamborello, F., Khemlani, S. S., & Schultz, A. C. (2013). ACT-R/E: An embodied cognitive architecture for human robot interaction. *Journal of Human-Robot Interaction*, *2*, 30–55.

Yamauchi, B., Schultz, A., & Adams, W. (1998). Mobile robot exploration and map-building with continuous localization. *Proceedings of the 1998 IEEE International Conference on Robotics and Automation* (pp. 3715–3720). Leuven, Belgium: IEEE Press.

Yeap, W. K. (2011). How Albot0 finds its way home: A novel approach to cognitive mapping using robots. *Topics in Cognitive Science*, *3*, 707–721.