Spatial Representation and Reasoning in an Architecture for Embodied Agents

Pat Langley
Center for Design Research, Stanford University
Stanford, California 94305 USA

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

Abstract

In this paper, we describe review PUG, a cognitive architecture for embodied agents that posits graded concepts that are grounded in perception and that integrates symbolic reasoning with continuous control. We review the framework's core assumptions about representation and processing, as well as how they enable spatial cognition. After this, we discuss how the architecture supports places, which it encodes as virtual objects defined by distances to reference entities, and reasons about them as if they were visible. We demonstrate PUG's abilities using a simulated robot that must approach targets and avoid obstacles. In closing, we discuss related research on agent architectures, robotic control, and spatial cognition, along with 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 mental abilities (Newell, 1990; Langley, Laird, & Rogers, 2009). Such theories include postulates about what structures and processes are constant across different domains. Just as a building architecture indicates which elements are fixed and which ones are mutable, 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 not about the content itself.

Most cognitive architectures import many of their core ideas from psychological theories. These usually include assumptions that: short-term memories are distinct from long-term stores; both memories contain modular elements cast as symbol structures; long-term structures are accessed through pattern relational matching; processing revolves around discrete cycles that involve retrieval, selection, and action; and cognition involves the dynamic composition of mental structures. A cognitive architecture usually comes with a programming language whose syntax and interpreter reflects these and other assumptions about mental representation and processing. Thus, it provides an infrastructure for developing intelligent systems.

Research in this area has been active for nearly five decades. Early work focused on production systems (Neches, Langley, & Klahr, 1987), 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 many researchers' concern with generality has led them to avoid making theoretical commitments about some topics. In particular, cognitive

architectures seldom incorporate strong assumptions about spatial representation and reasoning, even when used to develop agents that operate in physical settings (e.g., Jones, 1999).

In this paper, we report our efforts to address this oversight. We start by reviewing PUG, an architecture for embodied agents that retains many ideas from earlier frameworks but that grounds symbolic relations in quantitative descriptions and that enables continuous control. The theory supports basic aspects of spatial cognition, but only that based on the agent's immediate perceptions, so we also report extensions that allow representation and use of place knowledge. We illustrate 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.

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. It 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:

- 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 results from lower levels being used by higher levels.

PUG borrows all three assumptions from ICARUS (Choi & Langley, 2018), but it moves beyond this precursor in important ways.[1] 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 knowledge to drive its reasoning, decision making, and external behavior. 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 the agent's situation.

We will focus on concepts and skills in this paper, as they have the most relevance to spatial cognition. We will touch on processes only in passing and sidestep motives entirely, as they impact only the architecture's higher levels. 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

[1] Here we focus on the most recent version of the architecture, PUG/X (Langley & Katz, 2022), which extends previous incarnations (Langley et al., 2016, 2017). The earlier papers provide details about task-level plan generation, execution, and monitoring, which will not address here, as they have few direct implications for spatial cognition.

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 of the entities' attributes, and (b) four percepts for the same domain that describe perceived objects and five beliefs inferred from them.

```
(a) ((robot-at ^id (?r ?o) ^distance ?d)
     :elements   ((robot ^id ?r ^radius ?rr)
                  (object ^id ?o ^distance ?d ^radius ?or))
     :veracity   ((linear ?d (+ ?rr ?or) 10.0)))
    ((robot-facing ^id (?r ?o) ^angle ?a)
     :elements   ((robot ^id ?r) (object ^id ?o ^angle ?a))
     :veracity   ((linear ?a 0.0 45.0)))
```

```
(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]
    (robot-facing ^id (R1 O3) ^angle 0.0) [1.0]
```

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. Finally, each each concept has a veracity function that specifies how to compute its degree of match.

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, whereas the *:veracity* field specifies a function for computing the degree of match. Here the function $(linear\ obs\ max\ min)$ returns 1.0 if the observed value $obs \leq max$, returns 0.0 if $obs \geq min$, and returns $obs/(max - min)$ when $max < obs < min$. The expression is symmetrical for positive and negative values, as can occur with angles. Table 1 (b) also shows nine 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.

In a similar way, PUG incorporates four thoretical 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 specifes 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 indicates the degree to which the agent is pursuing it on a given cycle.

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. 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 avoding the obstacles $O1$ and $O2$, the first two being instances of the skills in (a).

```
(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]
```

PUG's syntax for skills 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 attributes, along with optional Boolean tests. Table 2 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 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 function for computing values of control attributes, *move-rate* and *turn-rate*. These functions include the symbol *$MISMATCH*, which is one minus the veracity of the target belief.

2.2 Cascaded Processing in PUG

Like other cognitive architectures, PUG operates in discrete cycles that use knowledge to produce new short-term structures, as well as behavior in the agent's environment. However, the framework differs in that it relies on five distinct levels:

- *Belief processing*, which uses conceptual knowledge to draw inferences from perceptions and predictions;
- *State processing*, which uses skills to calculate values of control attributes and processes to predict effects;
- *Execution / Simulation*, which generates motion trajectories in the environment or in the agent's mind;
- *Motion planning*, which carries out utility-guided search through a space of motion trajectories; and
- *Task planning*, which searches for candidate sequences of motion plans that achieve the agent's objectives.

PUG applies these levels in a cascaded manner, with each one using mental structures produced by those below it. We will discuss motion and task planning only in passing here, as spatial reasoning takes place primarily at the lower three layers.

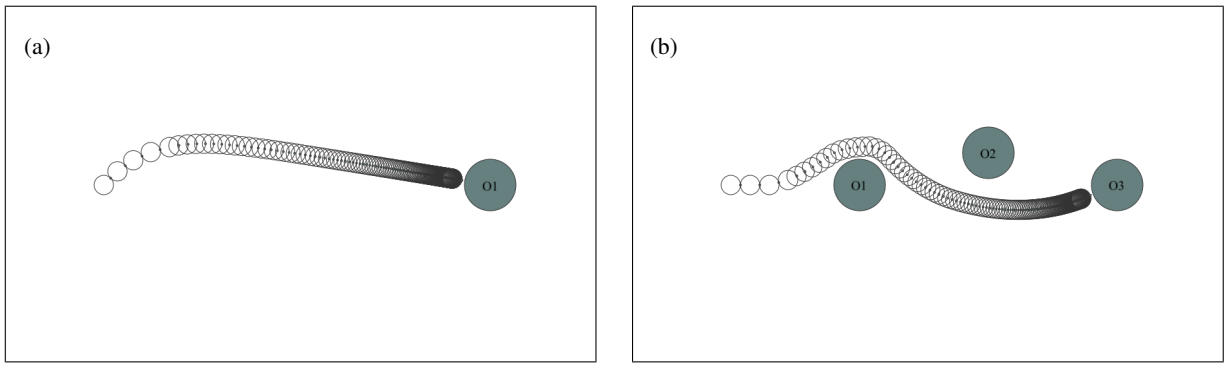

Figure 1: Two scenarios in which a robot must approach a nearby object: (a) when the robot is facing away from the target, it turns and moves forward simultaneously; (b) when two obstacles fall on the robot's path to the target, it veers left to avoid the first and right to evade the second. Traces show the robot's pose at equal time intervals, illustrating that position and orientation change more slowly as it approaches the target.

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 0.15)* and *(object ^id O1 ^distance 2.0 ^angle 0.0 ^radius 0.4)* to infer beliefs like *(robot-at ^id (R1 O1) ^distance 2.0)*. 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.

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 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. If multiple intentions apply on a given cycle, then PUG takes the vector sum of the resulting control values, much as in potential-field methods (Khatib, 1985). The system also uses the vector sum of matched processes, described elsewhere (Langley & Katz, 2022) to predict the effects of these control settings, as well as those of natural events the agent cannot influence.

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 processes, as it does not need predictions about future states. However, mental simulation of motion trajectories relies on them to produce a sequence of predicted changes that, when applied repeatedly, serve as the basis or successive belief and state processing. We should note that a trajectory follows deterministically from an initial state and a set of intentions; for this reason, we will refer to this sequence of states as a *motion plan*.

Figure 1 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 assumes 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 we specified 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.

PUG's fourth level of processing carries out heuristic search through a space of motion plans to select a set of intentions. This examines the utilities of trajectories produced by mental simulation to identity problems (e.g., collision with obstacles) and to select among alternatives. The fifth processing level generates task plans, each of which comprises a sequence of motion plans with associated trajectories. The architecture invokes utility here as well, in this case to guide search for task plans that achieve the agent's objectives. Earlier papers (Langley et al., 2016; Langley & Katz, 2022) provide details of these activities, so we will not repeat them here. Both forms of planning takes 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 physical environments. This necessarily means that it encodes and uses some forms of spatial information, but, like other cognitive architectures, it makes no theoretical commitments along these lines. Instead, PUG developers introduce assumptions about space that are convenient to achieve their ends, but they are not limited by explicit constraints about the topic. In this section, we present an elaborated version of the theory that incorporates such postulates. These require no changes to the implementation, only to its use. We first discuss how the architecture encodes perceived objects and then how it represents and uses place knowledge.

3.1 Perceptual Assumptions

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

- 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 from 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. 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. Beliefs about places have the same relationship to the rules that specify the latter. 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 7.07)*. This takes the same form as a primitive percept, even though it is derived from measurements about the agent's relation to other entitites. 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*.[2] However, the rule might also specify values for intrinsic attributes, such as radius, either as constants or in terms of the reference objects, that will not change over time.

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

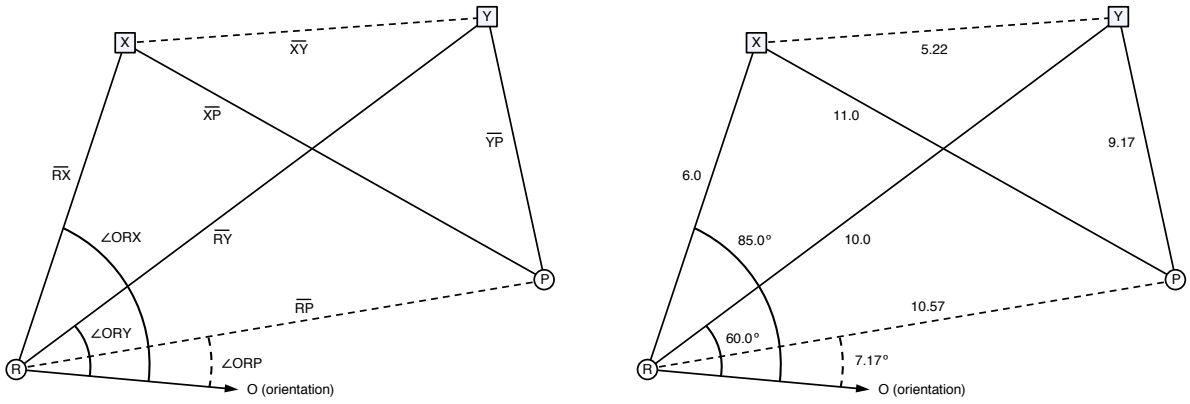

Figure 2: A scenario in which the extended PUG architecture computes the robot R's distance and angle to a place (the virtual object P) based on perceived distances and angles to two reference objects (X and Y). The left diagram shows symbolic notations; the right gives numeric values (not to scale). Solid lines and angles are perceived or given in P's place definition; dashed lines are derived from them.

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 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 7.07)*, 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.

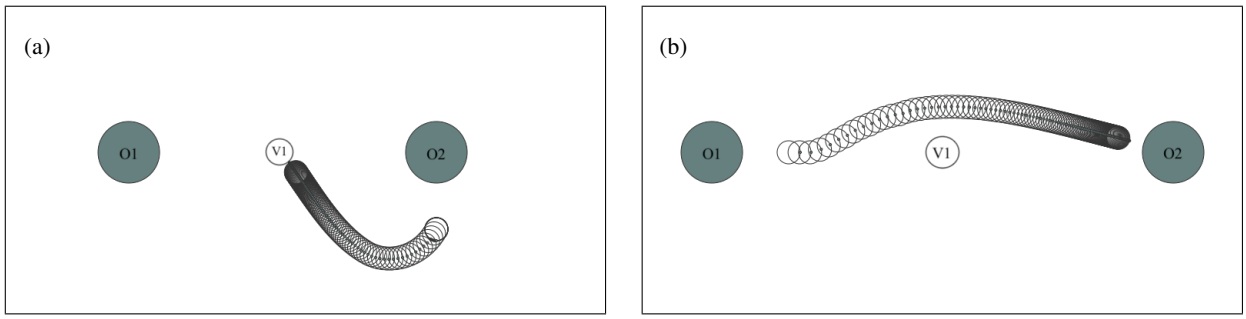

Figure 3: Two scenarios that involve maneuvering a robot with respect to a virtual object. In the left situation, the agent initially faces downward to the left and moves to a place that is defined as midway between two visible objects. In the right situation, it skirts a virtual obstacle that lies on the path to the target object.

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 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. As we discuss later, this opens the door to topological networks of places and hierarchical descriptions of space.

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 *(robot-at R1 V1)* and *(robot-facing 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 $O2$.

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).

Our response to the grounding requirement was 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. 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's 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.

In contrast, some robotics research has addressed continuous settings very differently. A 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). In addition, the architecture's reliance on vector sums to compute control values is similar to Khatib's (1986) use of potential fields.

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' (2000) 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.

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 currently 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. 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 describe 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 it with quantitative descriptors.

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 moved, in which case the system might find a subset of estimates that agree and ignore the anomalies.

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 about not only the agent's immediate surroundings but also about large-scale space. We have already shown that PUG's conceptual rules can define places in terms of other places, but our examples were at the same level of aggregation. This suggests that we need something more to specify larger spaces, such as a floor or building rather than its rooms. One response would be to view these as composite objects, which we need for other reasons, and introduce a notation for specifying them in terms of their components. This would also require adapting the architecture's mechanisms for conceptual inference, since the percepts available to the agent would only let it recognize small-scale places. Some variety of abductive reasoning 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 those 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.

Acknowledgements

The research reported here 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 improved the ideas in this paper.

References

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

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.

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.

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. (2020). The Spatial Semantic Hierarchy. *Artificial Intelligence*, *1–2*, 191–233.

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.

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.

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

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.