Motion Planning and Continuous Control in a Unified Cognitive Architecture

Pat Langley

Center for Design Research
Stanford University

Edward P. Katz

Stanford Intelligent Systems Laboratory
Stanford University

Thanks to Mike Barley, Dongkyu, Choi, Ben Meadows, and Mohan Sridharan for useful discussions. This research was supported by Grant No. FA9550-20-1-0130 from AFOSR, which is not responsible for its contents.

Cognitive Architectures

A *cognitive architecture* is a unified theory of the mind that:

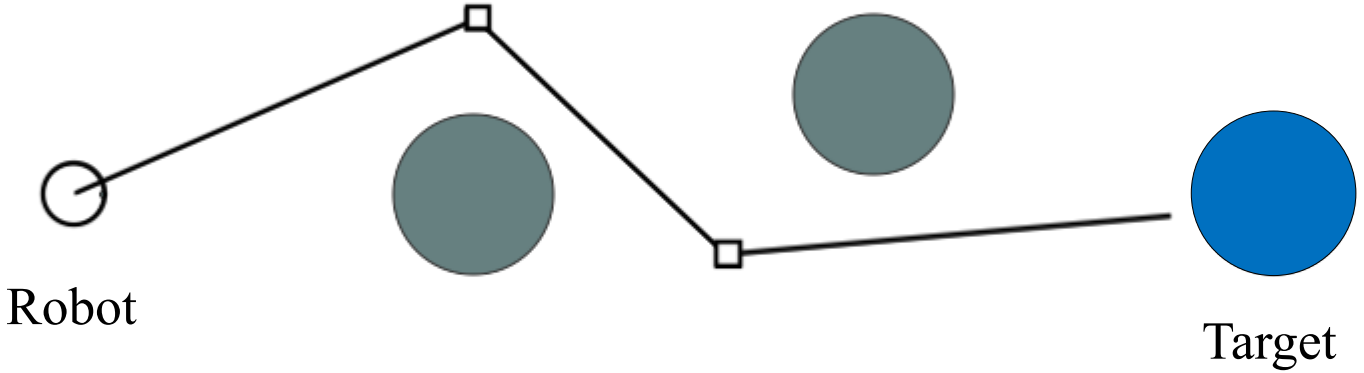
- Specifies what remains *constant* across different domains
- Incorporates many assumptions from *cognitive psychology*
- Offers a *programming language* for building intelligent systems

The PUG architecture (Langley et al., 2016) assumes that:

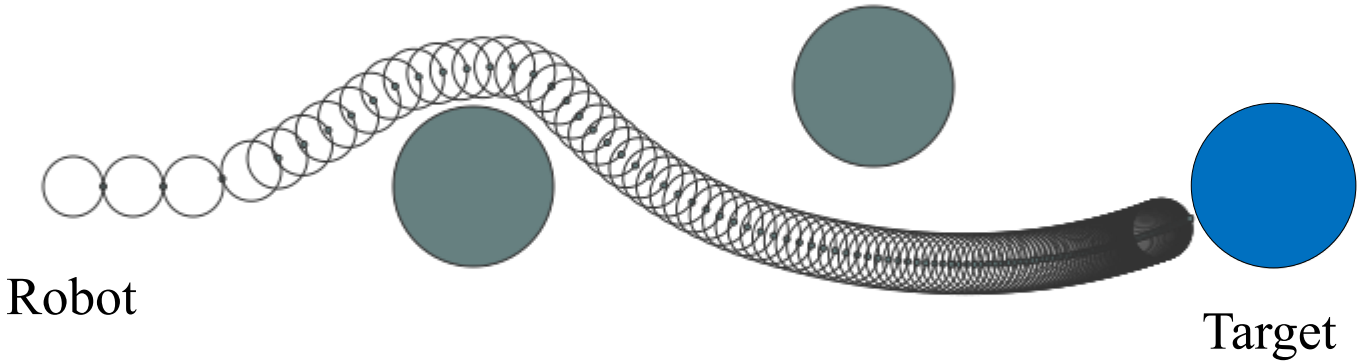
- *Symbolic relations are grounded in quantitative descriptions*
- *Relations have associated utilities that reflect tradeoffs*
- *Discrete skills have associated control equations*
- *Mental simulation creates trajectories to guide planning*

This talk reports a recent extension – PUG/C – that unifies symbolic and numeric processing more deeply.

PUG's Navigation Behavior



Desired Navigation Behavior



PUG's Knowledge Structures

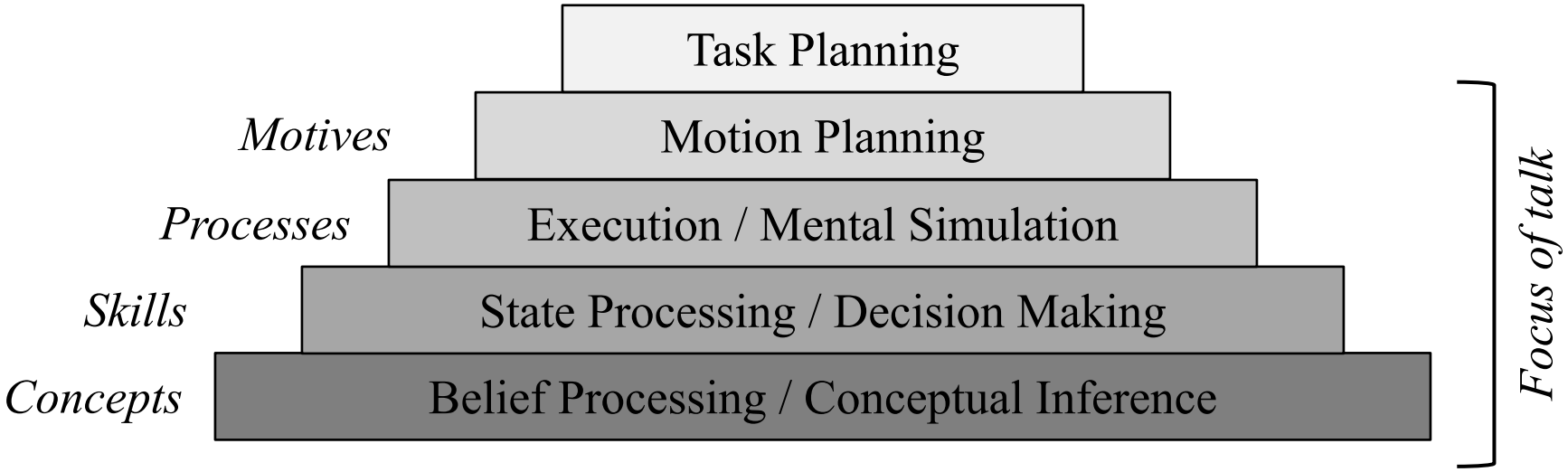
PUG/C incorporates four distinct types of generic, long-term knowledge structures:

- ***Concepts*** – Define relational categories, attributes, and *veracity*
- ***Skills*** – Specify *control values* based on match to *target concepts*
- ***Processes*** – Predict *changes* in attributes given current values
- ***Motives*** – Indicate *utility* of relations conditioned on situation

The architecture uses these elements for conceptual inference, reactive control, heuristic evaluation, and plan generation.

PUG's Layered Processes

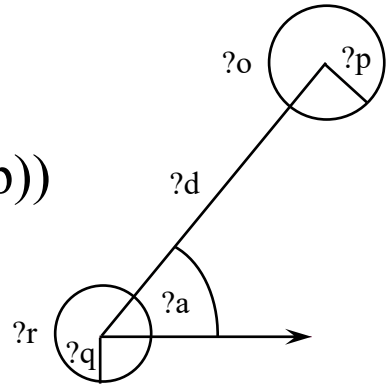
Like other cognitive architectures, PUG/C operates in *cycles* that use knowledge to produce new short-term structures.



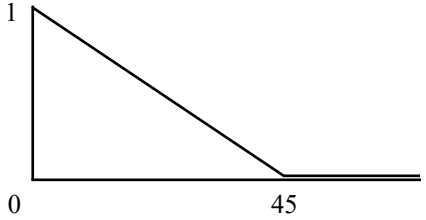
These levels are organized in a ***cascaded*** manner, with each one using results produced by those below it.

PUG Concepts

```
((robot-at ^id (?r ?o) ^distance ?d)
   :elements    ((robot ^id ?r ^radius ?q)
                 (object ^id ?o ^distance ?d ^radius ?p))
   :veracity    ((linear ?d (+ ?p ?q) 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)) )
```



Here the function *(linear obs max min)* returns *1* if the observed value *obs* ≤ *max*, *0* if *obs* ≥ *min*, and *obs/(max − min)* when *max* < *obs* < *min*.

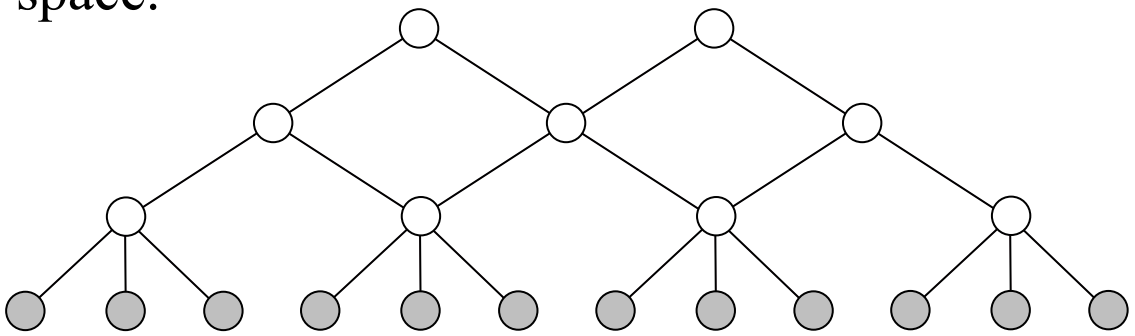
Beliefs are ground instances of concepts that relate specific entities.

Conceptual Inference in PUG

At the lowest level, *conceptual inference* derives beliefs that are consistent with perceptions / predictions:

- Matches conceptual rules to infer beliefs like (*robot-at R1 O1*)
  - Computes ***values*** of numeric attributes associated with beliefs
  - Calculates ***veracity*** (degree of match) for each inferred belief
- Applies this recursively to generate the full deductive closure

Together, the resulting beliefs describe the current ***state*** as a point in N-dimensional space.

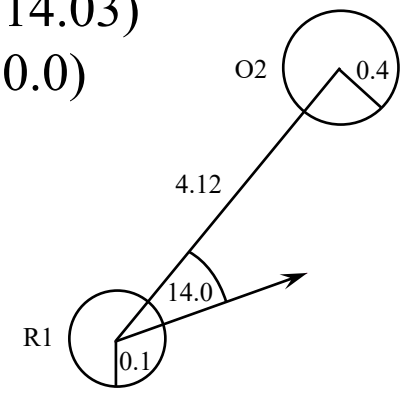


PUG Beliefs

Perceptions:	**Veracity**	**Utility**
(robot ^id R1 ^radius 0.15 ^move-rate 0.0 ^turn-rate 0.0)	1.00	0.0
(object ^id O1 ^distance 2.0 ^angle 0.0 ^radius 0.4)	1.00	0.0
(object ^id O2 ^distance 4.123 ^angle 14.03 ^radius 0.4)	1.00	0.0
(object ^id O3 ^distance 6.0 ^angle 0.0 ^radius 0.4)	1.00	0.0

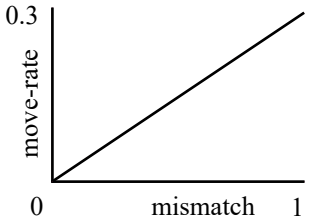
Inferred Beliefs:

(robot-at ^id (R1 O1) ^distance 2.0)	0.85	0.0
(robot-at ^id (R1 O2) ^distance 4.12)	0.62	0.0
(robot-facing ^id (R1 O1) ^angle 0.0)	1.00	0.0
(robot-facing ^id (R1 O2) ^angle 14.03)	0.69	0.0
(robot-facing ^id (R1 O3) ^angle 0.0)	1.00	0.0
(approaching ^id (R1 O1))	0.91	−20.0

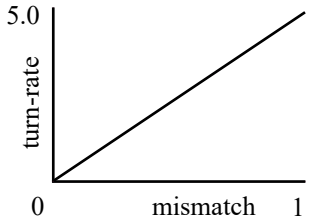


PUG Skills

```
((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 ^distance ?d))
  :control     ((robot ^id ?r ^turn-rate (* 5.0 (sign ?a) $MISMATCH)))
  :target      ((robot-facing ^id (?r ?o))) )
```



Here the symbol *$MISMATCH* stands for one minus the *veracity* of the matched target concept.

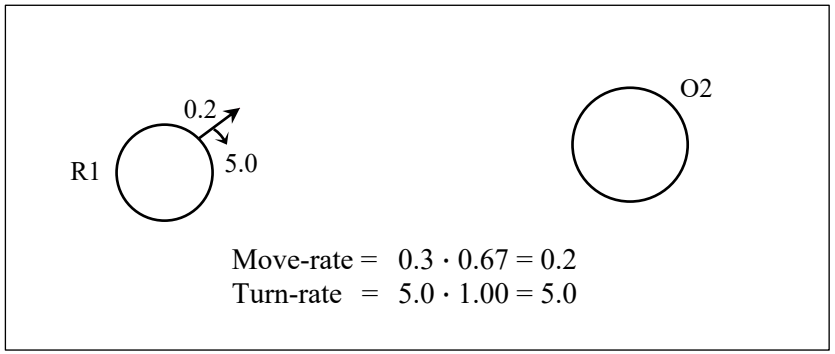
Intentions are ground instances of skills that involve specific entities.

State Processing / Decision Making

When PUG carries an out an active intention associated with skill S, whether mentally or externally, it:

- Checks that S's conditions match the current beliefs
- Finds degree of *mismatch* M to S's target belief
- Ensures the mismatch does not fall below threshold
- Else inserts M into S's equations to find control values

If *multiple* intentions apply, then PUG takes the *vector sum* of control values (as with potential fields).



PUG Processes

```
((move-relative ?r ?o)
  :elements    ((robot ^id ?r ^move-rate ?m)
                (object ^id ?o ^distance ?d ^angle ?a))
  :changes     ((object ^id ?o ^distance (*dd ?d ?a ?m)
                                ^angle    (*da ?d ?a ?m))) )

((turn-relative ?r ?o)
  :elements    ((robot ^id ?r ^turn-rate ?t)
                (object ^id ?o ^angle ?a))
  :changes     ((object ^id ?o ^angle (- ?t))) )
```

Here the functions **dd* and **da* calculate change in the robot's *distance* and *angle* relative to an object as it moves forward.

Reactive Execution / Mental Simulation

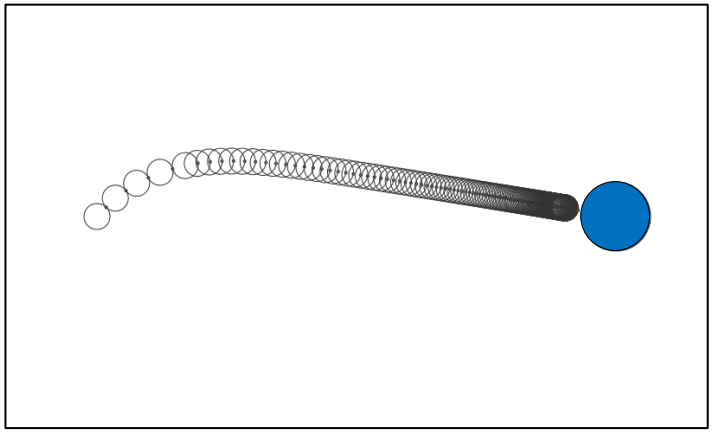
PUG/C applies skills and invokes state processing repeatedly to generate ***motion trajectories***.

- When the architecture carries out actions in the environment, this corresponds to ***reactive control***.
  - This operation does ***not*** require processes, as it has no need to predict future states.
- When the system imagines carrying out actions in its mind, it corresponds to ***mental simulation***.
  - This operation ***does*** rely on processes (applied in parallel) to predict the succeeding state.

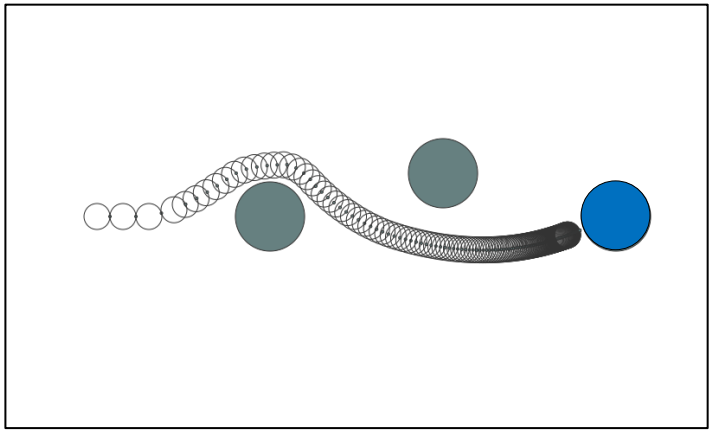
Each trajectory follows deterministically from an intention set (i.e., skill instances) that constitute a ***motion plan***.

Reactive Execution / Mental Simulation

PUG/C applies skills and invokes state processing repeatedly to generate ***motion trajectories***.



(move-to R1 O1)
(turn-to R1 O1)



(move-to R1 O3)
(turn-to R1 O3)
(avoid-on-left R1 O1)
(avoid-on-right R1 O2)

Concurrent Intentions

Each trajectory follows deterministically from an intention set (i.e., skill instances) that constitute a ***motion plan***.

PUG Motives

```
((robot-at ^id (?r ?o))
  :conditions  ((robot ^id ?r ^radius ?rr)
                (object ^id ?o ^type target ^distance ?d ^radius ?or))
  :utility     (cond ((< ?d (+ ?rr ?or 0.25))  10.0)
                     (t  0.0))
  :type        achievement )

((approaching ^id (?r ?o))
  :conditions  ((robot ^id ?r ^radius ?rr)
                (object ^id ?o ^type obstacle ^distance ?d ^radius ?or))
  :utility     (cond ((< ?d (+ ?rr ?or))  −20.0)
                     (t  0.0))
  :type        maintenance )
```

An *achievement* motive assigns utility to a belief only on its first match, where a *maintenance* motive does so repeatedly.

Heuristic Search for Motion Plans

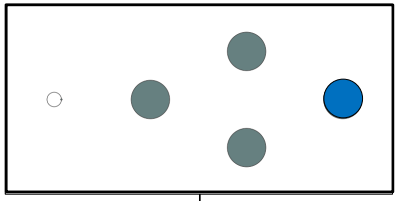
PUG/C combines motives with mental simulation to carry out greedy search through a space of motion plans.

- Starting from an empty plan, it retrieves intentions that would ***achieve its target beliefs***.
- Mental simulation produces a trajectory and uses motives to assign ***utilities*** to each belief and state.
- If some of these beliefs have ***negative utility***, PUG retrieves intentions that would help ***avoid*** them.
- When multiple repairs are possible, it selects the candidate plan / trajectory with the ***highest utility***.

This search continues until it finds an acceptable motion plan.

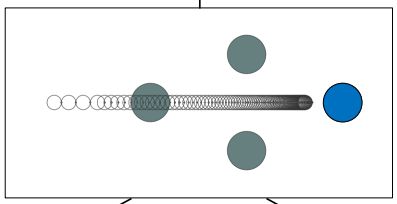
Heuristic Search for Motion Plans

Empty Plan

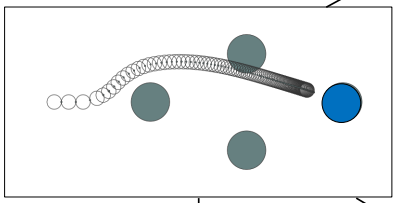


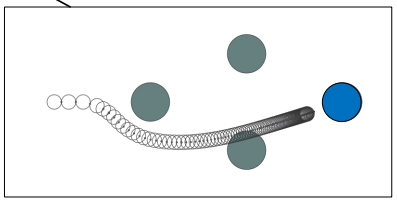
Here the agent wants to reach a target object.

But the initial trajectory collides with an obstacle.

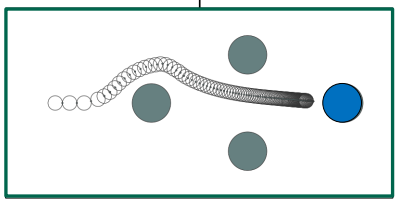


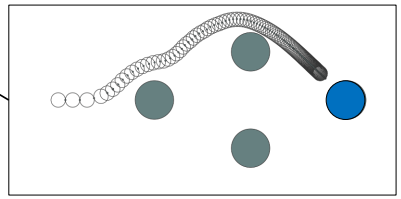
Search adds new intentions that improve utility.





Final Plan





Related and Future Research

PUG/C incorporates insights from a number of paradigms:

- Cognitive architectures (Soar, ICARUS, teleoreactive systems)
- Error-driven feedback control and potential fields
- Qualitative reasoning and quantitative simulation

But it combines ideas from these efforts in a unified framework.

Future research will add support for ***places*** (virtual objects) and large-scale ***maps*** (topological networks).

We will also integrate PUG/C with PUG/X, an earlier extension that combined ***task planning***, ***execution***, and ***monitoring***.

Summary Remarks

PUG/C is a cognitive architecture for embodied, human-like agents that incorporates:

- Concepts, motives, skills, and processes that have both ***symbolic*** and ***numeric*** elements
- ***Cascaded processing*** with layers for inference, state processing, mental simulation, motion planning, and task planning
- Skills' target concepts match to different degrees, which serve as error signals that enable continuous control
- Mental simulation and motivational processing support ***greedy search*** through a space of continuous motion plans

We demonstrated PUG/C's behavior on scenarios that parallel skill application, obstacle avoidance, and motion planning.

References

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

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

Goebel, R., Sanfelice, R. G., & Teel, A. R. (2009). Hybrid dynamical systems. *IEEE Control Systems Magazine*, *29*, 28–93.

Hasan, M., Warburton, M., Agboh, W. C., et al. (2020). Human-like planning for reaching in cluttered environments. *Proceedings of the 2020 IEEE International Conference on Robotics and Automation* (pp. 7784–7790). Paris: IEEE Press.

Katz, E. P. (1997). Extending the teleo-reactive paradigm for robotic agent task control using Zadehan (fuzzy) logic. *Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation* (pp. 282–286). Monterey.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. In I. J. Cox & G. T. Wilfong (Eds.), *Autonomous robot vehicles*. New York: Springer.

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

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

Nilsson, N. (1994). Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, *1*, 139–158.

PUG's Layered Processes

Like other cognitive architectures, PUG/C operates in *cycles* that use knowledge to produce new short-term structures.

The framework differs in that it relies on five processing levels:

- ***Belief processing*** – Inference from perceptions / predictions
- ***State processing*** – Applies skills, processes, motives
- ***Execution / Mental simulation*** – Generates trajectories
- ***Motion planning*** – Heuristic search for an intention set
- ***Task planning*** – Search for a sequence of motion plans

These levels are organized in a ***cascaded*** manner, with each one using results produced by those below it.