Regression Planning Networks

Danfei Xu, Roberto Martín-Martín, De-An Huang, Yuke Zhu, Silvio Savarese, Li Fei-Fei
Stanford Vision and Learning Lab, Stanford AI Lab

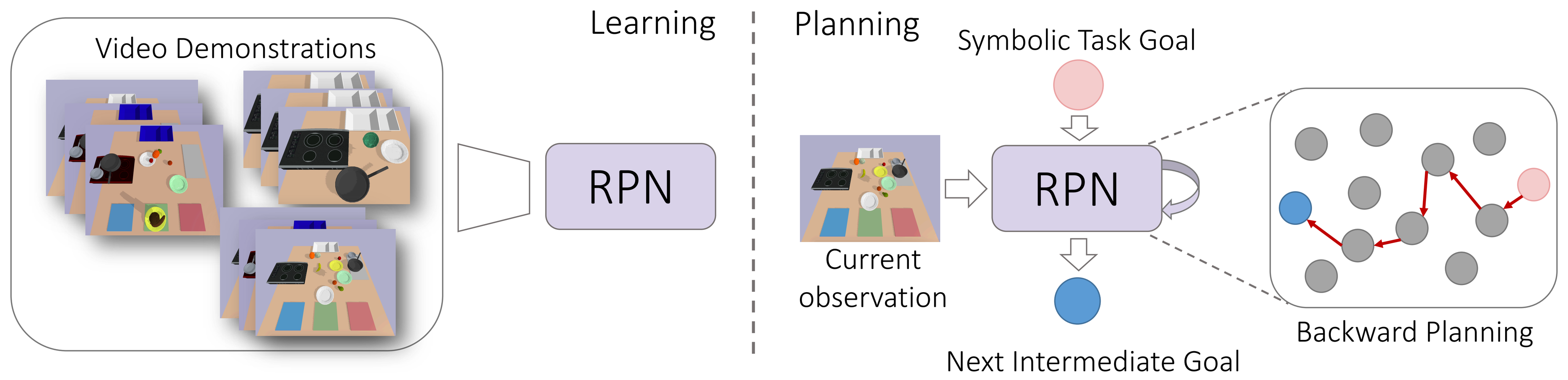
github.com/danfeiX/RPN

Overview

Goal: Plan for long-horizon robotics tasks with high-dimensional observations.

Challenges:
- Long-term predictions in **high-dimensional** space (images).
- Learning to plan towards **unseen goals**.

Key ideas:
- Learn to plan in a low-dimensional **symbolic space** conditioning on high-dimensional observations.
- Learn to break a complex task goal to sub-goals and plan **backwards (regression planning)** starting from the goal.
- Learn to model regression planning steps with a recursive neural network.



Background: Symbolic Regression Planning

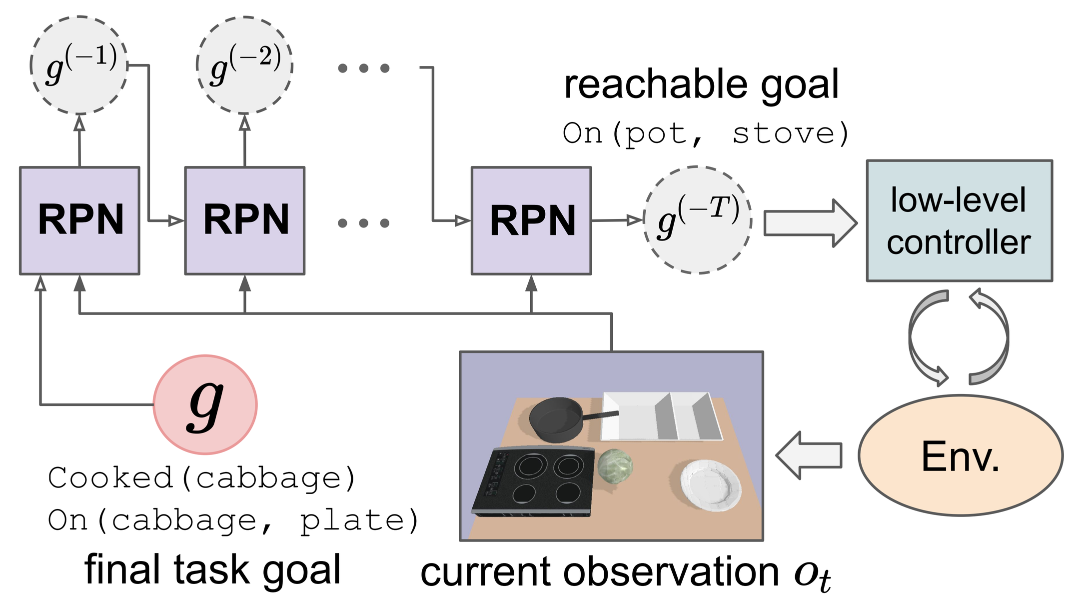
- **Regression planning [1]** is a type of classic symbolic planning algorithm.
- Starting from the goal, iteratively expand search space by enumerating all valid *action operators* that leads to a goal. The action operators are pre-defined in a **planning domain**. Planners also require hand-defined **state estimators**.
- Our method learns regression planner from **video demonstrations** without a planning domain or explicit state estimators.
- By conditioning on the current observation, we can train a regression planner to directly predict a **single path** in the search space that connects the final goal to the current observation.

```
:action pick
:parameters (?a ?o ?p ?g ?t)
:precondition (and (Kin ?a ?o ?p ?g ?q ?t)
                   (AtPose ?o ?p) (HandEmpty ?a)
:effect (and (AtGrasp ?a ?o ?g) (CanMove)
             (not (AtPose ?o ?p)) (not (HandEmpty ?a))
```

A sample *pick* action operator defined in a PDDL planning domain used by classic symbolic planners. RPN learns to plan without a planning domain.

Regression Planning Networks

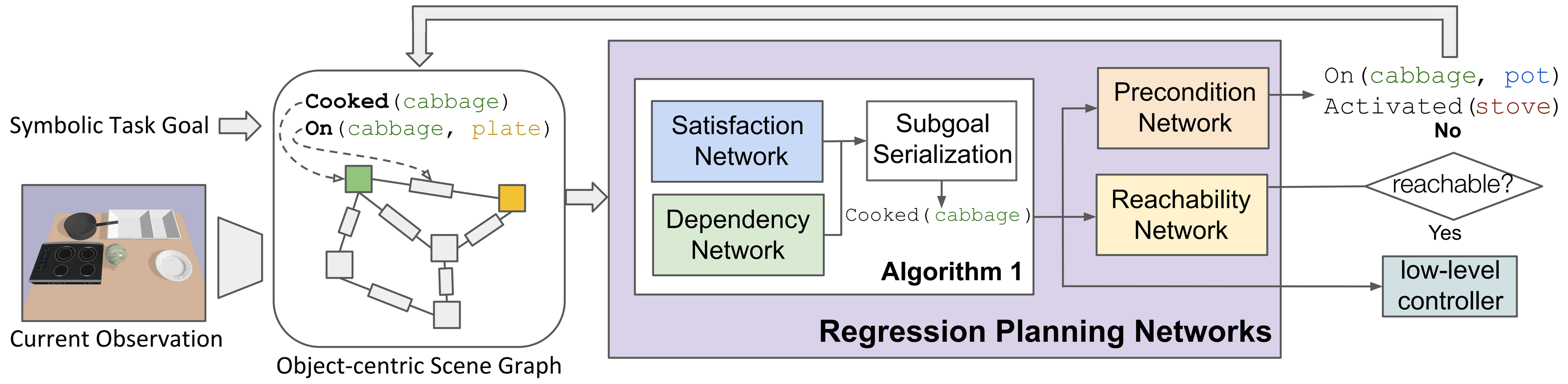
Starting from the final symbolic goal g, RPN iteratively predicts a sequence of intermediate goals conditioning on the current observation o_t until it reaches a goal $g^{(-T)}$ that is reachable from the current state using a low-level controller.



Algorithm 1 REGRESSIONPLANNING

Inputs: Current entity features e_t, final goal g, maximum regression depth M
Outputs: Intermediate goal to be executed by a low-level controller.
$i \leftarrow 0$
while $i < M$ **do**
  $g' \leftarrow \text{SUBGOALSERIALIZATION}(e_t, g)$
  $rec(g') \leftarrow f_{reachable}(e_t, g')$
  if $rec(g') > 0.5$ **then**
    return g'
  end if
  $g \leftarrow f_{precondition}(e_t, g')$
end while

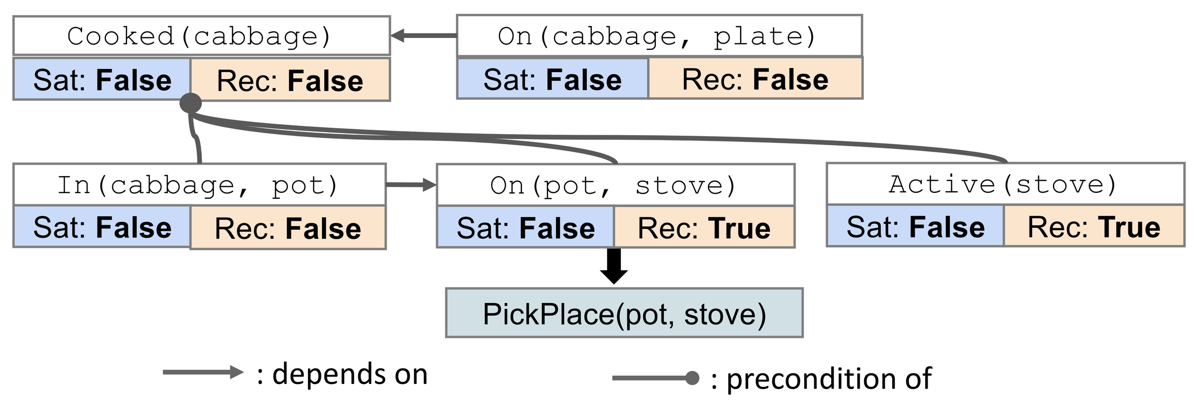
- **Recursive Architecture**: We model regression planning steps with a recursive and modular network architecture.
- **Object-centric representation**: A goal is specified as the **desired** state of an object or a relationship in a **feature scene graph** [2].



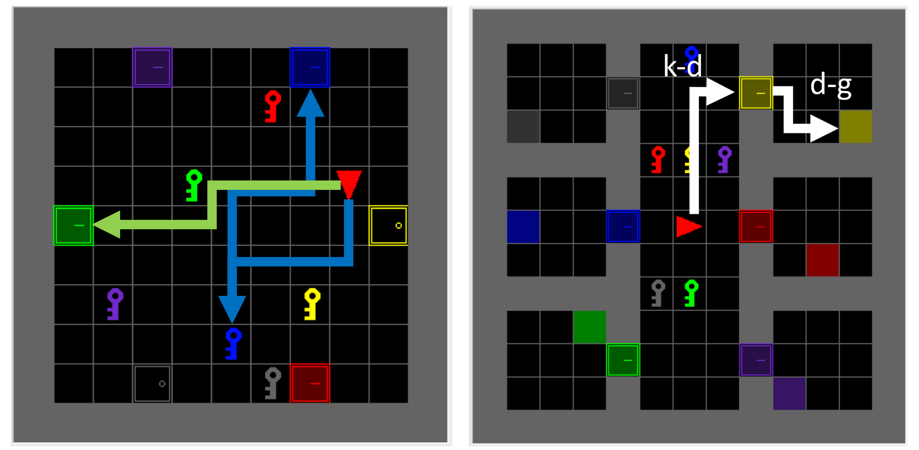
- **Subgoal serialization [3]:** Break a planning goal into sub-goals. We explicitly model their dependencies with a *Dependency Network*.
- **Pre-condition prediction**: Predict the predecessors of a goal that need to be satisfied as pre-conditions with a *Precondition Network*.

Algorithm 1 SUBGOALSERIALIZATION

Inputs: Current entity features e_t, goal g
$v \leftarrow \emptyset$, $w \leftarrow \emptyset$
for all $g_i \in g$ **do**
  $sat(g_i) \leftarrow f_{satisfied}(e_t^{g_i}, g_i)$
  if $sat(g_i) < 0.5$ **then**
    $v \leftarrow v \cup \{g_i\}$, $w \leftarrow w \cup \{e_t^{g_i}\}$
  end if
end for
$depGraph \leftarrow \text{DiGraph}(f_{dependency}(w, v))$
$blockGraph \leftarrow \text{Bron-Kerbosch}(depGraph)$
$blockDAG \leftarrow \text{DAG}(blockGraph)$
$sortedBlocks \leftarrow \text{TopoSort}(blockDAG)$
return $g[sortedBlocks[-1]]$



Experiment: Navigation in Minigrid 2D



Doorkey RoomGoal

Error Type	Network			Environment		
	All Sat	No Prec	Max Iter	Controller	Bad Goal	Max Step
E2E	/	/	/	0.3	92.6	7.1
RP-only	0.0	91.8	0.0	0.2	8.0	0.0
SS-only	0.0	/	/	0.0	78.9	0.0
Ours	0.9	21.3	8.7	0.1	1.4	3.3

In-depth error breakdown of D = 6 task in the DoorKey environment.

Domain	DoorKey			RoomGoal		
	Train	Eval		Train		Eval
Task	D=2	D=4	D=6	k-d	d-g	k-d-g
E2E [34]	81.2	1.2	0.0	100.0	100.0	3.2
RP-only	92.2	18.2	0.0	100.0	100.0	**100.0**
SS-only	99.7	46.0	21.1	99.9	100.0	7.8
RPN	99.1	**91.9**	**64.3**	98.7	99.9	**98.8**

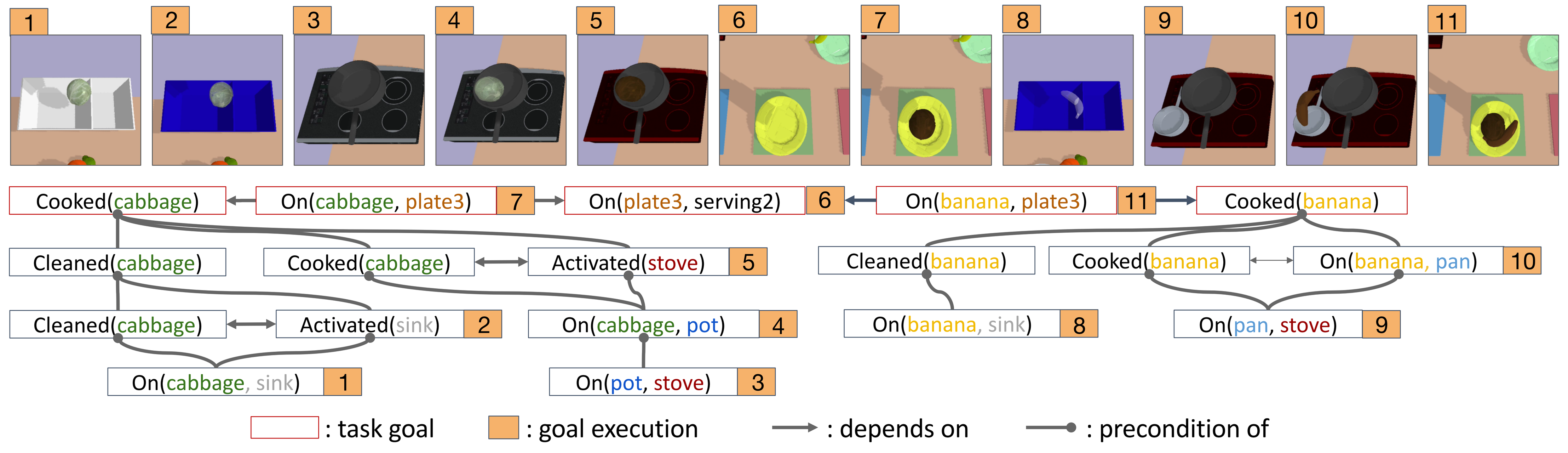
Results on Minigrid 2D

DoorKey
Training: Open D=2 doors with same-colored key.
Evaluation: Open D=[3 … 6] doors.

RoomGoal
Training: *key-door* (k-d) is fetch key and open the locked door to a room.
Training: *door-goal* (d-g) is to open an unlocked door and go to a goal tile.
Evaluation: *key-door-goal* (k-d-g) is to fetch key, open locked door, and get to a goal tile.

Experiment: Cooking in Kitchen 3D

- Prepare a meal with a variable number of dishes, each involving different ingredients and cookwares.
- Each task takes up to ~30 steps. Planner takes Image as input.



Task	Train	Evaluation				
	I=3, D=2	I=2, D=1	I=4, D=1	I=4, D=3	I=6, D=1	I=6, D=3
E2E	5.0 / 8.3	16.4 / 21.2	2.3 / 3.7	0.7 / 3.0	0.0 / <0.1	0.0 / <0.1
RP-only	70.3 / 83.4	67.1 / 77.4	47.0 / 71.7	27.9 / 64.1	<0.1 / 23.9	0.0 / 22.9
SS-only	49.1 / 59.7	59.3 / 61.9	56.6 / 66.2	43.4 / 60.0	42.8 / 69.3	32.7 / 59.7
RPN	**98.5 / 98.8**	**98.6 / 98.7**	**98.2 / 99.2**	**98.4 / 99.2**	**95.3 / 98.9**	**97.2 / 99.4**

Results on Kitchen 3D. Results are reported in average task success rate / average subgoal completion rate. Tasks are categorized by the number of dishes and number of ingredients used. I=3, D=2 means cooking two dishes with three ingredients.

References

[1] Richard Waldinger. Achieving several goals simultaneously. Stanford Research Institute Menlo Park, CA, 1975.
[2] Johnson, Justin, et al. "Image retrieval using scene graphs." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
[3] Richard E Korf. Planning as search: A quantitative approach. Artificial Intelligence, 33(1):65–88, 1987