TATA CONSULTANCY SERVICES
Experience certainty.

TATA

Towards Safe and Reliable Robot Task Planning

AI Safety Workshop, IJCAI 2020.

Presenter:

Snehasis Banerjee

AI Scientist, TCS Research & Innovation

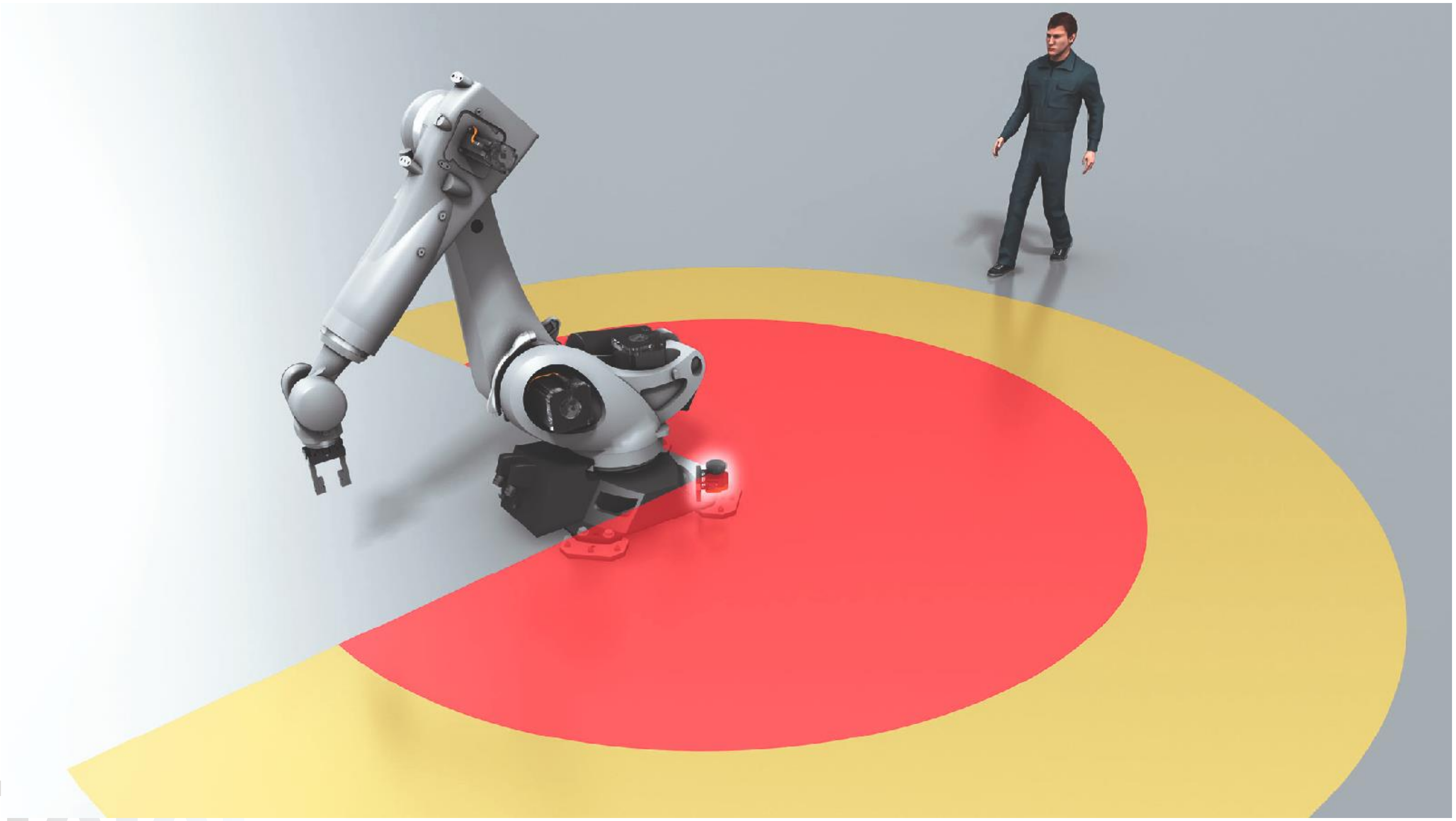
Sr. Member - ACM, CSI, IEEE

Duration: 5 min

Robotics - Safety and Reliability

Reliability is generally determined by the probability of a task to circumvent failures, while safety is related to the consequences of the failures.

For any service robot in human co-occupied space, safety is the #1 priority.



Background

- [Amodei et al. 2016] discusses 5 core research areas related to AI safety. This paper focuses on two areas:
  - Avoiding Negative Side Effects - ensuring the agent actions meet safety constraints
  - Safe Exploration - in dynamic and open environments, how reliable are the strategies to avoid safety hazards.

- Work is ongoing to create an AI Safety Landscape [Espinoza et al. 2019] to nurture a common understanding of AI Safety.

- Data driven ML approaches [Garcıa et al. 2015] [Krakovna et al. 2018] being non-contextual, lacks the richness of semantics in decision making. This work takes a knowledge-guided AI planning approach to tackle the safety aspects for service robotic navigation.

High Level Diagram

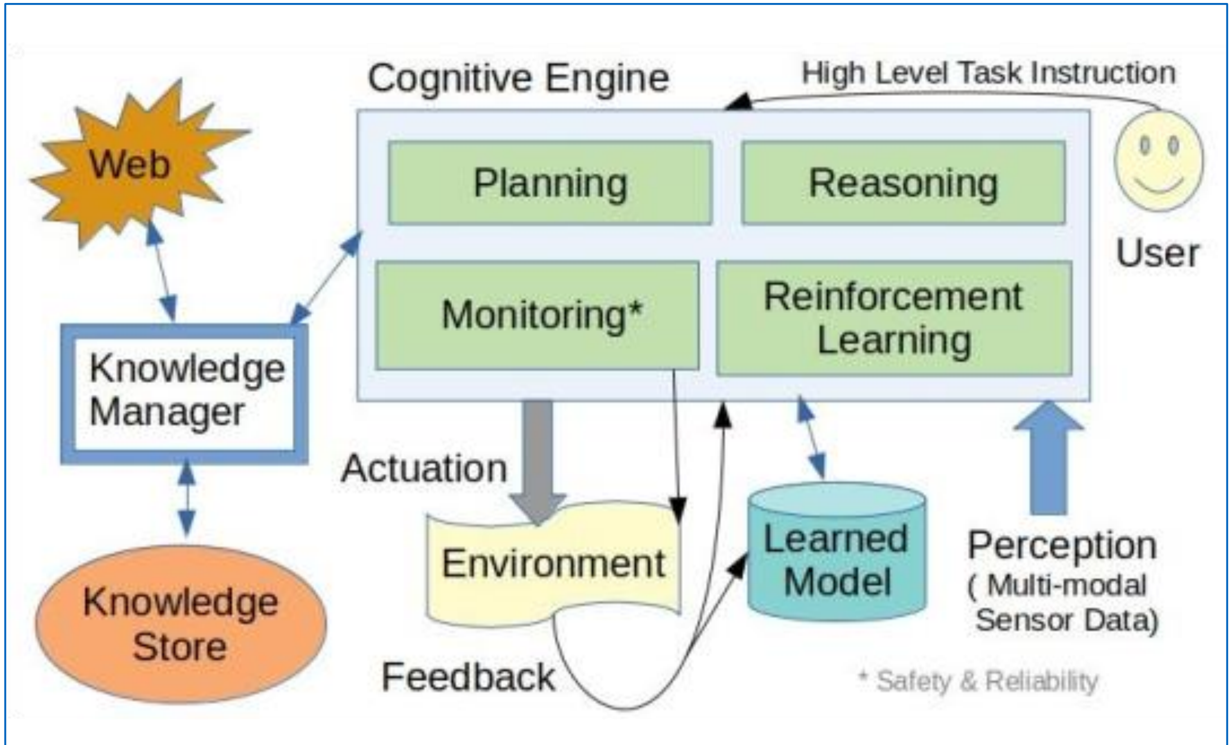


Figure 1: Architecture for Reliable & Safe Task Planning

Stairs hazard for 1-wheeled robot

Dark room - perception issue

Key highlights:

- Knowledge Based decision making
- Flexible values for safety and reliability
- Mix of ML* and KR (extensible) in decision making

Encoding safety and reliability in AI Robotic Planning

- Using Safety and Reliability in Metric PDDL
  - (:metric maximize (total-reliability))
    (:metric maximize (total-safety))
  - :effect (and ... (increase (total-reliability) R)
    (increase (total-safety) S))

- Safety Priorities and Vulnerability Values
  - Class: Robot | Vulnerability - 0.3 | Safety Priority - 0.6
  - Class: Glass Vase | Vulnerability - 0.9 | Safety Priority - 0.4
  - Class: Human | Vulnerability - 0.6 | Safety Priority -1

- State Reliability
  - maximize (w1. world state reliability + w2. individual state reliability
    + w3. task transition reliability + w4. task execution reliability)

Task Reliability Graph

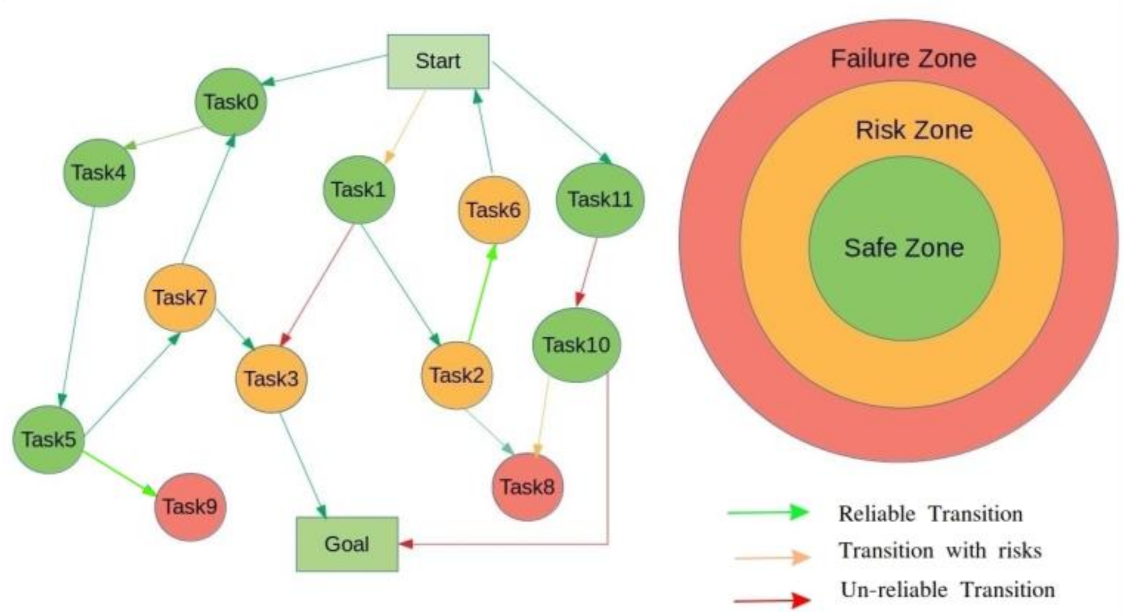


Figure 2: Task Reliability Graph and Reliability Zones

1. Start / Init
2. Goal
3. Task Node
4. Task Reliability
5. Task Transition
6. Task Transition Reliability
7. Safe Zone
8. Risk Zone
9. Failiure Zone

Case 1: If safety is highest priority – non-negotiable: No path exist, until safety is relaxed.
Case 2: If time is highest priority (emergency), Path:
Start □ Task1 □ Task3 □ Goal.
Case 3: If time and safety have relaxed priority, Path:
Start □ Task0 □ Task4 □ Task5 □ Task7 □ Task3 □ Goal.
[Above path has all safe 'green' transitions, but 2 risky tasks]

Task transitions between the state actions of 'turning off lights' and 'moving ahead' are not reliable – as movement in dark is prone to failure without sensors that work in dark.

Algorithm

Algorithm 1: Safe and Reliable Task Planning

```
Result: Task Success or Failure under constraint set C
Parameters:
perception ← sensor input stream;
actuation ← movement or manipulation by agent;
knowledge ← link to semantic knowledge store;
software ← link to software modules and libraries;
goal ← target goal state or final task state to reach;
CS ← current state of agent;
Begin:
plan ← software.plan(perception, knowledge, goal);
TRG ← initialize Task Reliability Graph with priors;
while goal ≠ CS And pre-conditions = satisfied do
    CS ← plan.nextTaskStep();
    if Constraints in CS w.r.t. TRG ⊆ set C then
        TRG.evaluate(perception) for Changes;
        if Changes detected in world state then
            S1 ← software.RL.evaluate(perception) -
              generate next step based on
              reinforcement learned model;
            S2 ← get next step from TRG after
              evaluation of reliability of world state,
              CS, task transition, task execution;
            actuation ← voting (S1 ∩ S2)
        else
            actuation ← voting (TRG.nextActuation()
              ∩ software.RL.nextActuation() );
        end
        TRG.update() - weights of edges and nodes;
        software.RL.update() - update rewards by
          processing current scene and task status;
        knowledge.update() - object state values;
    else
        plan ← software.replan(perception, TRG);
    end
    if goal = reached Or Exit Criterion met then
        actuation ← STOP; Wait for next command;
    end
end
```

TATA CONSULTANCY SERVICES
Experience certainty.

snehasis.banerjee@tcs.com
joysnehasis@gmail.com

Thanks! Questions Please.