Towards Safe and Reliable Robot Task Planning

Snehasis Banerjee

TCS Research & Innovation
Tata Consultancy Services, Kolkata, India
snehasis.banerjee@tcs.com

Abstract

A long-standing goal of AI is to enable robots to plan in dynamic and uncertain environments by managing task failure intelligently. Reliability of a robot task planner has become essential to ensure operational safety. Reliability is generally determined by the probability of a task to circumvent failures, while safety is related to the consequences of the failures. In this paper, methods and an architecture for handling reliability and safety in robotic task planning is presented. The paper takes help of a telepresence navigation scenario to describe some candidate solutions such as Task Reliability Graph and weighted reliability goals.

1 Introduction

Characteristics of a typical robotic AI system are (a) Adaptability (b) Autonomy (c) Interaction. Decision making in AI systems can be fully autonomic or semi-autonomic in case of involvement of external entity (say human operators in telepresence scenario). Recently, 'AI Safety' as a research area has gained prominence due to the gradual infiltration of autonomous agents in the normal human environment. Work is ongoing to create an AI Safety Landscape [Espinoza et al. 2019] to nurture a common understanding of current needs, challenges and state of the art and field practices relevant to safety in AI. [Amodei et al. 2016] discusses five core research areas related to AI safety. This paper focuses on two areas: (a) Avoiding Negative Side Effects - ensuring the agent actions meet safety constraints (b) Safe Exploration - in dynamic and open environments, how reliable are the strategies to avoid safety hazards. A body of work [Garcıa et al. 2015] [Krakovna et al. 2018] has been done on Machine Learning (ML) based approaches to tackle the above areas. However, data driven ML approaches being non-contextual, lacks the richness of semantics in decision making. The paper takes a knowledge-guided AI planning approach to tackle the aforementioned problems, specifically to a telepresence navigation scenario in uncertain environment.

Reliability [Dhillon, B.S. 2015] is closely related to AI Safety. Reliability can be defined as how probable is the agent to successfully complete a task without failures, while safety is determined by the direct or indirect consequences of the failures. In case of a semi-autonomous agent like a telepresence robot, reliability is hampered due to several reasons such as random component and sensor failures, human errors, systematic hardware failures (degradation over time), software errors, dynamic environment. Some related works like [Axelrod, B. 2018] and [Fisac, J.F. 2018] has shown the importance of policy level safety aspects in the case of robot planning in dynamic obstacle scenarios. However, attaching safety aspects to task planning has remained neglected by literature – which is the key highlight of this paper, explained with the help of a telepresence navigation scenario.

AI Task Planning is based on decision making strategies of action sequences, typically for execution by intelligent and autonomous agents, especially robots. In known environments with available models or maps, planning can be done offline. In dynamically changing unknown environments and in open world scenarios, the strategy needs to be revised and adapted while plan is executing. This requires continuous monitoring of state variables (states of objects in environment). An example object discreet state variable is door in open or closed position. However, a continuous state variable is also possible, say the angle of door being open (in semi closed position). Choice of object state variable depends on domain and how much granularity is needed in a solution to accomplish a given goal. Typically, PDDL (Planning Domain Definition Language) [Gerevini et al. 2009] and its variants are used to represent the problem, initial conditions, goal state and action sequences. This is passed on to a suitable planner (based on semantics used in PDDL) to output a suitable sequence of tasks. To take care of different constraints and semantic descriptions, a large number of planning methodology exists such as metric-based planning, spatial planning, temporal planning, probabilistic planning, belief based planning, planning on open world and open domains. However, while the focus of the planning research community has been to come up with planners with faster computing time and richer semantic processing capability, the aspects of safety and reliability in tasks as a feature has remain mostly neglected. The paper describes an architecture and methodology to tackle the safety and reliability aspects as a first step to fill this gap.

2 Applicability in Telepresence Navigation

To illustrate the scenarios in which safety and reliability becomes essential for robotic task planning, help of telepresence use case is taken here. A telepresence robot (type of service robot) helps a human user (called operator) to be virtually present at a remote location. Telepresence robots have started to gain popularity in schools, corporate offices, hospitals, clinics, warehouses and conferences. The default telepresence robot comes with a camera and navigation wheels on top of which more sensors can be added to enhance its perception of the environment. A telepresence robot can operate in fully guided, semi-autonomous as well as autonomous mode as per application needs and sophistication in software and hardware used for that application. However, fully guided mode is discouraged as if user's instructions are strictly followed, the robot may cause harm to itself or others, due to lack of any restrictions. Hence, the desired modes of operation are either autonomous or semi-guided with restrictions (constraints). If a high level instruction is provided by the user say '*Find where the paper shredder is kept*', the robot analyses the instruction and decomposes the task into sequence of most optical sub-tasks to achieve the goal. This requires the robot to get a visual idea about object namely 'paper shredder' it has to find. Also, if the object is not currently in visibility, it needs to explore the environment until it finds the goal object. In semi-guided mode, the successful working of the robot relies heavily on communication network where a remote user is controlling the robot based on the feedback the user is perceiving from the sensors attached to the robot (such as ego view camera). If there is a communication failure, the semi-guided mode may prove to be disastrous especially if communication fails during executing a sub-task. Communication failure may lead to robots colliding with obstacles due to continuation of trajectory in absence of expected interrupt. Also while exploring, if it comes to a zone of continuous moving obstacles (like living beings or other robots), it may go to a stand still (with goal of moving forward) to keep safe distance. Considerable amount of work has been done on collision avoidance algorithms in navigation [Hoy et al. 2015]. PDDL based planning approach [Gayathri et al. 2018] [Ingrand et al. 2017] has been extensively used in robot navigation tasks. However, the algorithm depends on continuous input from sensors that may be faulty and provide erroneous readings or sensed data that eventually leads to collision. Also, there are issues in bridging software and hardware instructions when interacting with physical devices. An example is: if a robot is given a software instruction to rotate X^0 angle, the robots' physical wheel actually rotates $(X+n)^0$ angle, thereby making the robot unsafe in precise navigation. Lack of domain and commonsense knowledge can also have negative consequences in successful task completion. Suppose a robot via its camera can see the target object, however the terrain is not safe for navigation (say staircase in case of wheeled robot), blindly following the instruction will lead the robot to fall and break down. Also there can be adversarial situations that can fool a robot perception thereby leading to task failure. This can be due to target objects appearing in mirror, electronic display screen (say television) or portraits. Another limitation a robot may face is range of a sensor: if the hazard or moving obstacle is not in reception coverage of a sensor, the robot cannot detect whether the selected trajectory is correct. A robot's visual perception may give only an imperfect view of clear path as well as objects due to sensor errors or non-favorable lighting conditions and environmental layout. If the telepresence robot is provided with tactile object manipulation capabilities, then the risk becomes more severe. An example is: if in a pantry environment, the robot lits fire to serve a user instruction, without having a gas and smoke detection sensor. Related safety issues are handling fragile objects without care as the robot lacks the knowledge and experience of holding that object. All this lead to task failures and non-adherence to safety. This demands a dedicated focus on safety and reliability for robotic task planning.

3 Proposed Architecture and Methodology

In a human robot interaction (HRI) set up, the User passes on a high level task instruction (via text, audio, etc.) that is captured by the robotic system. The system needs to decide an optimal sequence of steps – which will be carried out as actuation in the environment, in order to satisfy the goal of the user instruction. A generic architecture (Fig. 1) is presented along with significance of each individual module in aiding optimal task execution. Granular details of each module is skipped to keep the focus on safety and reliability aspects.

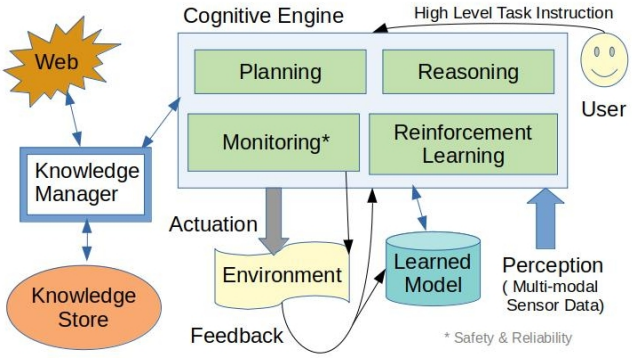

Figure 1: Architecture for Reliable & Safe Task Planning

- *Reasoner*: This module infers semantic facts combining apriori knowledge and perceived realtime facts while operating in the working memory [Mukherjee, D. et al. 2013]. Breaking a given high level instruction into smaller tasks and overcoming dis-ambiguity is taken care by this module. In uncertain environments, decisions can be taken by using standard methodologies in reasoning under uncertainty [Kern-Isberner et al. 2017].
- *Planner*: Planning deals with modeling of the relations between perceptions and actions. Planning uses a world model to estimate deliberation steps as per goal. When problem domain, objects and relevant actions, initial and goal states are properly encoded in a semantic format such as PDDL, a planner can find the best suitable steps to reach goal state. In dynamic and open worlds with

high scope of task failures, dynamic planning becomes essential. In dynamic planning, object state variables are monitored and if a failure state is nearing, re-planning happens from current state condition towards goal.

- *Reinforcement Learning (RL) Module*: As the robot agent explores the world and carries out sub tasks to reach its goal, it continuously learns through its perception capabilities and consequences of actions in the environment. The action and the path (taken by robot) goes as a direct feedback as what to do when confronted with a similar situation - this scenario is modeled by RL with rewards to successfully reach its goal under constraints and penalty (negative reward) in failures. In simple worlds, if only knowledge based decision making is used, RL module can be altogether omitted. However, in complex worlds where granularity of knowledge may become limited in expression, a hybrid approach by mixing semantic decision making with RL policy based actuation decisions will prove fruitful.
- *Knowledge Manager*: When dealing with data from various sensors it is essential to bring all the information in a common semantic knowledge format compliant for reasoning and planning. The purpose of this module is to gather, curate and unify variety of knowledge (like facts, relations, ontologies). It also connects the Semantic Web via SPARQL endpoints for accessing on demand relevant knowledge missing in current working memory.
- *Knowledge Store*: Knowledge being represented in various formats, a requirement is to curate the same in blocks. Due to dynamic environments, the knowledge needs to be updated and over-written based on learned experience or gathering of fresh unseen knowledge.
- *Learned Model*: This serves as a store for updating learned models (safety and reliability weights over a task and a task group) from actuation feedback via either reinforcement learning or the monitoring module of the system or both. Manual entry of initial weights and range thresholds will make system robust to over fitting.
- *Perception Module*: A robot is equipped with multiple sensors which is required for successful completion of a task and its own operation. The perception module is designated with processing raw sensor readings and converting them usable semantic information for the cognitive engine. As an example, a camera image is processed to identify objects and context from robot's ego view.
- *Actuation Module*: Actuation of the robot is based on physical interaction with the environment. This is generally achieved by sending motor movement instructions from the software interaction module of a robot's base system (usually ROS – Robot Operating System) on top of which the cognitive engine is running.
- *Monitoring Module*: This is the most important module (Section 3.5) from the point of safe and reliable task execution. The module oversees (a) monitoring current environment state including robot and objects (b) filtering the perception module for any conflicting sensed information (c) providing realtime input to decision making.

Cognitive Engine's main role is observe–orient–decide–act (OODA). It's main jobs are: (a) Handle failures and errors (b) Learn when things go right and wrong, or in other words learn from negative and positive experience in various task execution scenarios (c) Ensuring safety in operations (d) Predicting likelihood of a task failure or safety risk, given current state. The following methods shed lights on system objectives.

3.1 Using Safety and Reliability in Metric PDDL

Assigning reliability and safety to individual task action sequence in semantic forms like PDDL is straight-forward solution. In the goal objective function, the intricacies of safety and reliability can be included. Optimal path search can be done by using metric supported planners as follows:

(:metric maximize (total-reliability))
(:metric maximize (total-safety))

Task actions will have an effect variable like:

:effect (and ... (increase (total-reliability) R)
(increase (total-safety) S))

A drawback of this technique is that one is required to know beforehand all the safety and reliability values for each task, which might not be possible in a cold start scenario. If probability and uncertainty is involved, P-PDDL (probabilistic PDDL) with unknown states and probabilities will be a more suitable representation. Sensor failures can be modeled based on sensor error modeling techniques [Berk, M. et al. 2019]. But task level failures can be learned by actual task execution in real life or via realistic simulations in robot simulators like Webots, Gazebo, Gibson, AI2Thor.

3.2 Safety Priorities and Vulnerability Values

In the context of AI Safety, it is often observed that total focus is given to human safety. However, in situations where a robot agent is interacting with the environment, it is also essential to give safety priorities to other living beings, valuable objects and the robotic agent itself (it also has value). This is achieved by maintaining a semantic database that contains the relative priorities and properties of the entities that will enable decision making under the specified constraints. Diverse situations and applications will have different priority values. Hence, it is better to have a relative index instead of an absolute one. In case of service robots, the end user should be provided an interface to set custom values, as what humans value most varies from one person to another, which is a technical challenge. As an example, a simple cup might have less material value in terms of money, but if a human memorable event is attached with it, then its relative value becomes higher. A naive strategy is to keep safe distance from high value entities while actuating. A significant aspect to consider is entities in a environment can also be tagged with degrees of vulnerability. As an example, glass made objects if in navigation path of robot, may prove to be vulnerable if robot somehow deviates from its intended route (resulting in collision). However, based on time and emergency constraints, robot may have to execute tasks that carry some risk of failure and safety relaxations. A sample manual entry in tabular format is enlisted:

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

3.3 Task Reliability Graph

This section describes the concept of modeling and handling task failures in terms of Task Reliability Graph (TRG). Some terminology is defined first: (a) Start/Init: this is initial state of the world (b) Goal: it is the expected state of the world after complete plan is executed. (c) Task Node: subtasks representing action sequences that has pre-conditions and effects on the world. This can also be thought as action recipes. (d) Task Reliability: this signifies how much reliable the task is - this value can be a discreet value (such as high, medium, low) It has to be a continuous value in a specified range (say 0 to 1 with normalization) if . This value can be learned by observing task executions in real world or computer simulations. For examples, task of moving straight can be thought as reliable, however rotating on axis may turn out to be unreliable task under precision demanding scenarios (like rotate by exact r^0). (e) Task Transition: this indicates the switching from one task to another following the planner set sequence. When a task completes execution, the control is given back to the cognitive engine from the actuator, which again initiates the next task in planned pipeline. Some task transitions are seamless like between moving straight and rotation. However, 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. Unreliable transitions mean that there is high risk of failure. (f) Task Transition Reliability: it means how reliable is the transition from one task to another. The numeric values around it can also be learned by monitoring task transitions in real world and simulations. (g) Safe zone: the world state where chances of robot doing un-intended action is almost nil (h) Risk zone: the world state where there is a chance of task failure, but recovery or rollback to a safe zone is still possible (e) Failure Zone: the world state where the robot cannot recover and needs external intervention. This world state is more prone to safety violations.

Fig 2. shows an instance of a Task Reliability graph where the objective is to reach Goal node from Start node. The optimal path for given problem is found by looking simultaneously at multiple task nodes (having reliability weights), their connecting edges (having task transition reliability weights), the safety value assigned to the edges and nodes. Depending on the constraints (limited time, no risk) the optimal path will be different. To simplify understanding, reliability weights are not considered in following cases.

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 $\rightarrow$ Task1 $\rightarrow$ Task3 $\rightarrow$ Goal.

Case 3: If time and safety have relaxed priority, Path:
Start $\rightarrow$ Task0 $\rightarrow$ Task4 $\rightarrow$ Task5 $\rightarrow$ Task7 $\rightarrow$ Task3 $\rightarrow$ Goal.
[Above path has all safe 'green' transitions, but 2 risky tasks]

It is important to keep note of visited nodes and where occurrence of failure is more in the graph cycles. Another parameter to consider is cost of rollback - how further back from current node in graph agent needs to travel in order to reach a safe taste. Another point is – if the robot agent do not have the suitable sensors to monitor the state to carry out a vulnerable task, then such a task by default will fall in Risk zone.

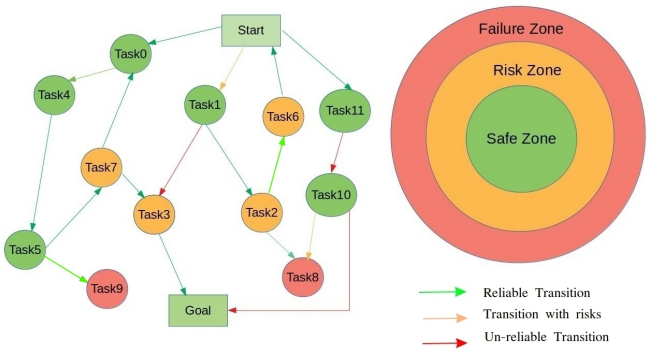

Figure 2: Task Reliability Graph and Reliability Zones

We explain TRG with the help of a simple telepresence scenario. Some of the sub tasks can be: open a door (T1), close a door (T2), turn on lights (T3), turn off lights (T4), move a step (T5 - this can be further subdivided as sub tasks of vector movements), apply pressure to hold an object (T6), lift an object (T7), move an object (T8). Suppose the end user from a remote location issues a command to find an object 'mobile phone' left somewhere at home. In this case, the transition T4 $\rightarrow$ T5 is risky; T6 $\rightarrow$ T8 is also risky if object can not be moved without lifting (here the phone). Also had object being something heavy like portable cupboard, then lifting it may cause the robot to lose balance and damage itself. If the robot has sensors to sense in dark (like infrared camera), then T4 $\rightarrow$ T5 is not risky. So each task transition is characterized by robot's current perception capacity as well as task and environment state. If the current scene set lacks dynamic obstacles, then risk factors get minimized automatically. Hence, reliability values of task and task transitions are based on different scenarios and constraints – accordingly the optimal task path for same goal (user instruction) will vary.

3.4 State Reliability

Joint monitoring the path to goal in solution space as well as state variables is a better approach than relying only on one aspect. As an example, tasks can fail if battery powering the robot (object state) gets drained although there was no issues in the reliability of task at hand and transitions. For each goal, a task path is created that contains states of variables. Simultaneously relevant world state (a subset can be extracted by looking at the problem, goal and possible paths) needs to be monitored and reliability can be assigned to (a) each relevant object state (b) relevant world subset of states representing a possible state. This may not be mapped to a specified task template but can be thought of as an unknown state, but should not be generalized as a single dead state. If there are 'm' state variables with 'n' possible discreet values, there will be much less than maximum possible world states, as co-occurrence of [$m_1(n_1), m_2(n_2)$)] is not possible due to semantic and logical constraints. For example, state variable 'light' in 'off' state will make camera sensed state variables null, as due to darkness there is no visibility. This constraints can be derived from ontological descriptions, common sense knowledge as well specified in PDDL under

the *:constraints* section. This can lead to task reliability prediction as a weighted equation:

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

3.5 Task Monitoring and Decision Making

Any machine learning model is bound to failures as the predictions cannot be 100% certain – a model is as good as its features [Banerjee, S, et al. 2016]. The fundamental aspect of safe and reliable task planning is to minimize levels of uncertainty. So, if a world model is exhaustive, then uncertainty is zero. However, in real life, it is very difficult to have an exhaustive state space description on which an agent can actuate. So a hybrid approach entangling uncertainty handling in the core decision making process is a practical way out:-

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* $\neq$ *CS* ***And*** *pre-conditions = satisfied* **do**
  CS ← plan.nextTaskStep();
  if *Constraints in CS w.r.t. TRG* $\subseteq$ *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

4 Conclusion

The intention of this paper is to ensure that AI systems of the future that uses task planning are not just optimistically safe but robustly, verifiably safe — because it was designed and built with safety and reliability in mind from ground zero. The strategies laid out are generic enough for widespread adoption across safe task planning use cases. Experimentation on multiple use cases on both simulation and real life situations will enable more insights to the proposed approaches. Future work will include mathematical formulation with safety guarantees [Polymenakos, K. et al. 2019] to model agent (robot) actuation in an uncertain world.

5 References

[Amodei et al. 2016] Concrete problems in AI safety, arXiv preprint arXiv:1606.06565.

[Axelrod, Brian et al. 2018] Provably safe robot navigation with obstacle uncertainty. The International Journal of Robotics Research 37.13-14: 1760-1774.

[Banerjee, Snehasis, et al. 2016] Towards wide learning: Experiments in healthcare. arXiv preprint arXiv:1612.05730. ML4Health, NIPS 2016.

[Berk, Mario, et al. 2019] Reliability assessment of safety-critical sensor information: Does one need a reference truth? *IEEE Transactions on Reliability 68.4 (2019)*: 1227-1241.

[Dhillon, B.S. 2015] *Robot System Reliability and Safety: A Modern Approach.* CRC Press.

[Espinoza et al. 2019] Towards an AI Safety Landscape - An Overview. *Source: https://www.ai-safety.org*.

[Fisac, Jaime F., et al. 2018] Probabilistically safe robot planning with confidence-based human predictions, arXiv preprint arXiv:1806.00109.

[Garcıa et al. 2015] A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16.1: 1437-1480.

[Gayathri et al. 2018] Ontology based knowledge representation technique, domain modeling languages and planners for robotic path planning: A survey. *ICT Express* 2018 Jun 1; 4(2):69-74.

[Gerevini et al. 2009] Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners.*AI* 173.5-6: 619-668.

[Hoy et al. 2015] "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey." *Robotica* 33.3: 463-497.

[Ingrand et al. 2017] Deliberation for autonomous robots: A survey. *Artificial Intelligence* 247, 10-44.

[Kern-Isberner, G. et al. 2017]. Many facets of reasoning under uncertainty, inconsistency, vagueness, and preferences: A brief survey. *KI-Künstliche Intelligenz*, 31(1), 9-13.

[Krakovna et al. 2018] Measuring and avoiding side effects using relative reachability. arXiv preprint arXiv:1806.01186.

[Mukherjee, Debnath, et al. 2013] Towards efficient stream reasoning. On the Move to Meaningful Internet Systems, Springer, pp. 735-738.

[Polymenakos, Kyriakos, et al. 2019] Safety guarantees for planning based on iterative Gaussian processes. arXiv preprint arXiv:1912.00071.