Partially Observable Failure Recovery

Gilberto Briscoe-Martinez
Department of Engineering and Applied Sciences
University of Colorado Boulder
Boulder, USA
gilberto.briscoe-martinez@colorado.edu

Stephen Otto
Department of Engineering and Applied Sciences
University of Colorado Boulder
Boulder, USA
stot7670@colorado.edu

James Watson
Department of Engineering and Applied Sciences
University of Colorado Boulder
Boulder, USA
james.watson-2@colorado.edu

***Abstract*—In this experiment we demonstrate how Partially Observable Markov Decision Process and Behavior Trees can be applied to robot manipulation. Using Webots simulator we set up a peg and hole problem with 3 different failures possible. A UR5 robot is used along side a touch sensor in order to model a real life scenario of a robot trying to achieve a peg in hole problem. A Particle Filter is used for state estimation, and Behavior Trees are used for decision making for the simulation.**

***Index Terms*—POMDP, Behavior Trees, UR5 Robot, Webots, Particle Filter**

I. Introduction

II. Related Work

Fault recovery systems are a long studied area of Computer Science. With the complexity and opacity of computer and robotic systems, fault recovery with only a partial understanding of the whole system has become necessary to perform. Early works have shown the promise of using Partially Observable Markov Decision Processes (POMDPs) to recover from faults in complex computer systems [1] [2] and to allow simple mobile robots to recover from navigational errors [3]. POMDPs have also proven to be capable at understanding and solving task execution failures [4]. There have been other approaches to solving task failure in robots that have used specific controllers to enable recovery. [5] We have decided to not use this approach given how the design and tuning of those controllers allow only specific faults and actions to be taken.

Contemporary research on using behavior trees(BTs) combined with a belief system can generate very flexible failure recovery options. [6]

III. Background

A. Behavior Trees

Behavior Trees were first seen used in gaming because of its ability to choose actions in complex environments. State Machines are commonly used in robot manipulation and are easy to integrate into a system, but they have issues with complex environments. Behavior Trees excel over state machines with decision making and goal planning. Behavior Trees applied to robotics is relatively new and robot manipulation rarely uses this framework.

B. POMDP

Partial Observable Markov Decision Process (POMDP) is a form of reinforcement learning finding more applications in robotics. POMDP is a method of machine learning that can make decisions under uncertainty. The PO in POMDP is to say that

C. Particle Filters

Particle Filters are a set of Monte Carlo algorithms used to filter out unlikely sensor data. It consists of trying to estimate the current state of the dynamic system. Particle Filters can be divided into two steps. The prediction step and correction step. Commanding a dynamic system we can estimate what we believe would be the change in state. We can use the sensor data to then have a better estimate of the current state.

IV. System Overview

The experiment was run in Webots simulator. In Webots we use the UR5 robot in the simulated world. Figure 1 shows the object we will be grasping with the UR5 robot. Figure 2 shows the peg we will be trying to insert the object into. These were CAD modeled using SolidWorks and transferred to Webots Simulator by using the STL files.

The problem we are setting up is essentially a peg in hole problem. How it differs is that because of the curved surfaces we need to have a better estimation of where we are in order to find the hole. The robot will have a hard time traversing to the center with the circular pocket around the center area. The robot will have to maneuver by pulling back and then moving in order to achieve its goal. A common solution to peg and hole problems, which can't work because of the circular pocket, is spiraling until the peg finds the objects hole. We will need to develop a better set of actions in order to combat this problem.

For simplicity the robot will have the object already in hand. To give randomness to the experiment the object will touch on a randomized spot on the peg.

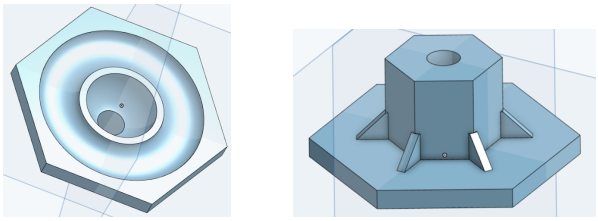

Fig. 1: Top and Bottom View of the Object used to insert the hole

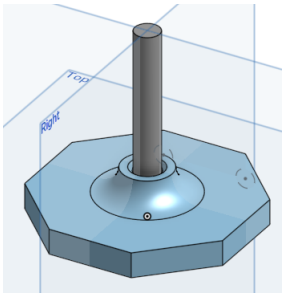

Fig. 2: Top View of the peg

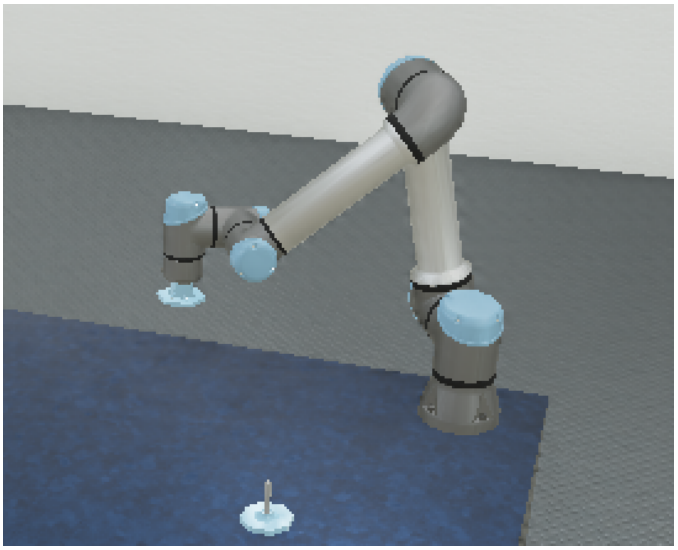

Fig. 3: The experiment in Webots Simulator

V. Problem Setup

At the beginning of each simulation episode, the Object is placed onto the peg such that the bottom surface of the Object is on or near the top of the peg. We consider the Object to be in one of three states according to its relationship to the peg. In State 0, the peg rests in the central concavity of the Object bottom surface. In State 1, the peg rests in the outer annular concavity of the Object bottom surface. In State 2, the peg is beyond the edge of the Object, but we define this "miss" state to be an annular region akin to state 1, for the sake of simplicity.

After contact with the peg, the robot moves the Object along the curve of the bin with the dynamics described in Section VI-B. We call this circular "feeling around" of the bin Spiral Search, similar to but not identical to the Spiral Insertion described in [7]. Note that it is not possible to move between the bin regions of States 0, 1, and 2 using Spiral Search.

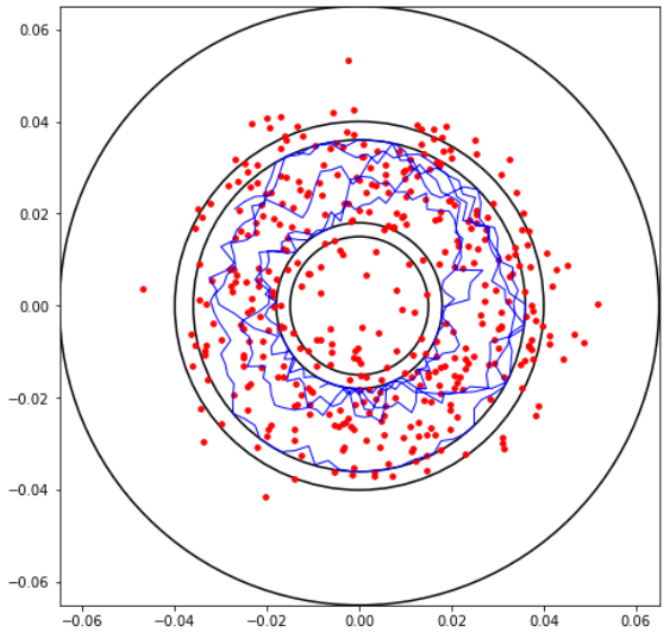

Fig. 5: Particle generative model showing path of peg relative to the Object (blue line) and noisy readings of the same (red dots)

There is an action prescribed for each state. In State 0, the robot should push down and attempt to sink the peg into the central hole of the object following one timestep of Spiral Search. In State 1, the robot should lift the Object slightly and set it back down on the peg. In State 2, the robot should move the Object back to its original resting place, regrasp, and begin Spiral Search again. The latter action is meant to simulate recovery from grasp failure or major misalignment, and it is the most time-consuming action. A table of action time costs can be found in Figure 4.

Action	Code	Time Cost [s]	Spiral Restart?
Continue Spiral	10	0.1	No
Attempt Push	0	0.5	No
Lift & Retry	1	5.0	Yes
Complete Reset	2	20.0	Yes

Fig. 4: Time costs of robot actions

VI. Methods

A. POMDP

The POMDP representation of our problem differs from the usual in three ways. Firstly, the problem has 2 types of state; the 2-dimensional relative position of the peg in the Object frame, and which Object bin the peg currently

rests in (1-dimensional). The former state space relates to the observations of the Object, and the latter relates to the appropriate action Secondly, only the 2-dimensional position particles (See Section VI-B) adhere to the Markov Property. Lastly, while it is true that the system state is not directly observable, the action to take in any bin state is always the same, so the problem is not a true MDP. After the state has been estimated, action selection is trivial. Our aim is to study the time savings of the framework, not to learn recovery actions.

B. Particle Filter State Estimation

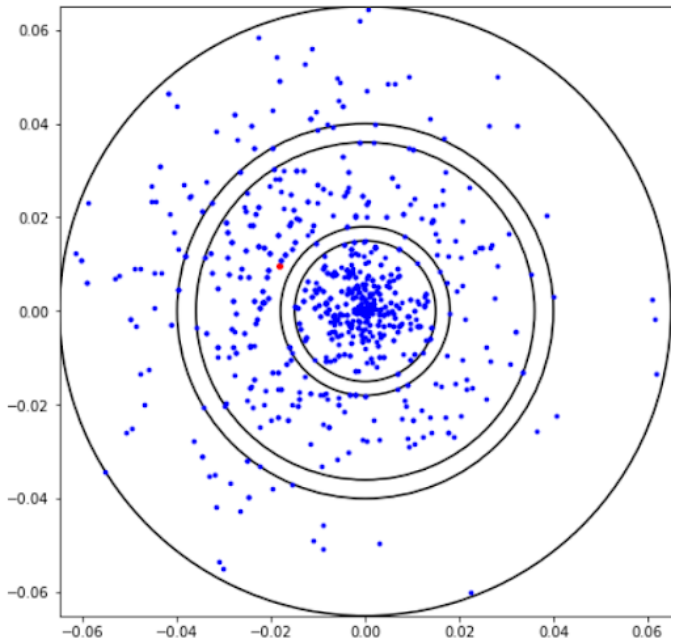

Fig. 6: State Estimation with Particle Filter after the first iteration

The purpose of the implemented particle filter is to estimate a discrete 1-dimensional state from noisy and continuous 2-dimensional observations. The state must be correctly identified in order to avoid inappropriately performing costly long-running actions. Particles represent the possible 2-dimensional positions of the peg relative to the Object. The state of the of a particle is determined by the bounds of the bin. The particle filter uses $N = 1000$ particles to estimate state.

The dynamics of the particles are governed by a generative model (Figure 5). The model assumes that once the peg is in a slot, it stays there until one of the actions that restarts the spiral search (Actions 1 and 2) is taken (Figure 4). Within a slot, a particle proceeds in small angular steps around the curve of the bin ($+\Delta\theta$), counterclockwise when viewed from the bottom of the Object. The radial distance of the particle from the center of the Object bottom plane is perturbed in steps by $\Delta r \sim \mathcal{N}(0, d_b/8)$, where d_b is the radial width of the current bin.

The progression of the particle filter estimation can be seen from Figure 6 to Figure 7. Figure 6 shows the filter after the first resampling. The particles are very spread out, but the particles farthest from the observation (red dot) have already been eliminated. By the tenth iteration (Figure 7), the overwhelming majority of particles reside in true State 1.

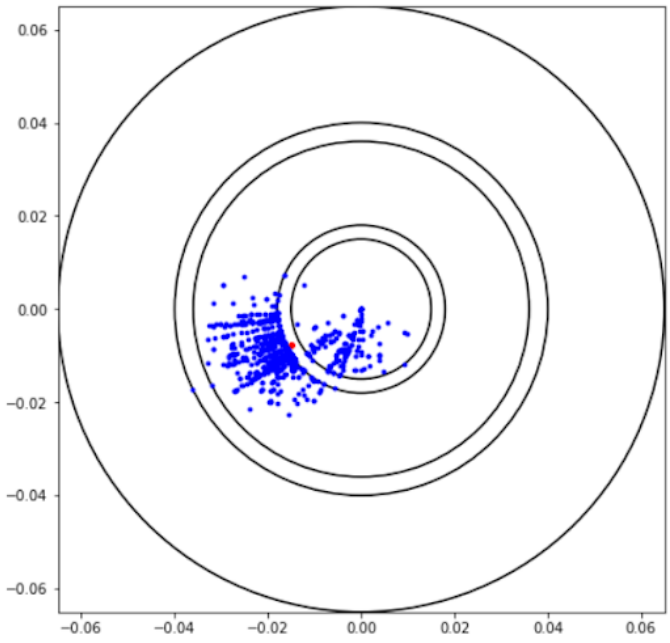

Fig. 7: State Estimation with Particle Filter after the tenth iteration

The simulation provides noisy readings of the peg position relative to the Object, which provide evidence of its ground truth state. The particle filter takes evidence into account via the resampling process. After each observation, a new particle population of size N is built. Particles from the previous iteration are added to the new population with a probability of $\Pr(\text{Add } p_i) = \exp(-(d_{i,o}/s))$, where $d_{i,o}$ is the distance between the current observation and particle i (p_i) and s is a scaling factor. We found $s = 3$ to be a suitable scaling factor by trial and error. We continue to cycle through the old population, adding with $\Pr(\text{Add } p_i)$ until the size of the new population is N. This process creates multiple of sampled states most likely to produce the observation, and deletes the samples least likely to.

Note that we did not find it necessary to repopulate bins starved of particles during resampling, as is often done. This might have been necessary if position readings were noisier.

C. Behavior Trees

The left side of the behavior tree looks for the object and then reads the pose for the object to be picked up. Once done the object will be grasped. The left side of the behavior tree will be skipped and the UR5 robot will start with the object grasped. It is important to show this side of the behavior tree because if a failure goes terribly wrong in another behavior the tree will have the robot reset because of the possibility of a massive grasping issue.

The right side of the behavior tree focuses on observing where the robot is. The robot finds the peg and its pose, and then hovers about it. The robot then lowers its arm til it feels a touch sensor. If there is no touch sensor then a massive failure has happened and the robot should cut its losses and start over. The robot will then move around and give a belief for the current position. If the robot believes its in the middle area it will spiral in a circle and push down. If the robot believes it is in the circular pocket around the center than it will back up move towards the center and repeat the right side of the tree. It will repeat until the robot has finished its goal or found an error worth restarting.

VII. Results & Conclusions

Using the setup described above, we varied the number of particles used by the filter. We ran 100 trials at each N, and tabulated the results in Figure 9. It is difficult to draw conclusions from these results, because the spread of running times is very large. The median running time and standard deviation are lowest at $N = 1000$, so this seems to be the best N of those tested.

N	Median running time t_G [s]	STDDEV(t_G)
50	51.25	60.30
100	39.80	57.75
250	50.95	50.95
500	44.60	61.99
1000	**33.70**	**45.76**

Fig. 9: Variation of Particle Count N, 100 trials

At the chosen $N = 1000$, the acceptance threshold was varied. This is the belief level (fraction of particles) needed to trigger the action associated with the highest state belief. No conclusions can be drawn except that any majority of particles is sufficient to correctly identify the bin state for this problem. The spread of all the running times overlap.

T_A	Median running time t_G [s]	STDDEV(t_G)
0.50	39.25	66.92
0.60	49.50	80.56
0.75	55.65	56.09
0.85	53.95	63.04
0.90	50.05	62.43
0.95	56.15	60.52

Fig. 10: Variation of Acceptance Threshold T_A, $N = 1000$, 100 trials

VIII. Future Work

Future Work will involve bring this experiment to real life application. Grasping the object will also be incorporated. This will allow for a more realistic approach to this problem. The particle filter should also be compared to a hand-designed behavior.

References

[1] G. Shani and C. Meek, 'Improving Existing Fault Recovery Policies', Advances in Neural Information Processing Systems, 2009, 22.

[2] K. R. Joshi, W. H. Sanders, M. A. Hiltunen and R. D. Schlichting, "Automatic Recovery Using Bounded Partially Observable Markov Decision Processes," International Conference on Dependable Systems and Networks (DSN'06), 2006, pp. 445-456, doi: 10.1109/DSN.2006.16.

[3] JOQUÍN L. FERNÁNDEZ , RAFAEL SANZ & AMADOR R. DIÉGUEZ (2004) PROBABILISTIC MODELS FOR MONITORING AND FAULT DIAGNOSIS: APPLICATION AND EVALUATION IN A MOBILE ROBOT, Applied Artificial Intelligence, 18:1, 43-67, DOI: 10.1080/08839510490250097

[4] K. A. Svendsen and M. L. Seto, "Partially Observable Markov Decision Processes for Fault Management in Autonomous Underwater Vehicles," 2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 2020, pp. 1-7, doi: 10.1109/CCECE47787.2020.9255782.

[5] Stefanovski, JD. Novel controller design for fault tolerant tracking with general additive faults and disturbances. Int J Robust Nonlinear Control. 2021; 31: 7724– 7742. doi:10.1002/rnc.5722

[6] R. Wu, S. Kortik and C. H. Santos, "Automated Behavior Tree Error Recovery Framework for Robotic Systems," 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 6898-6904, doi: 10.1109/ICRA48506.2021.9561002.

[7] Watson, James, Austin Miller, and Nikolaus Correll. "Autonomous industrial assembly using force, torque, and RGB-D sensing." Advanced Robotics 34, no. 7-8 (2020): 546-559.

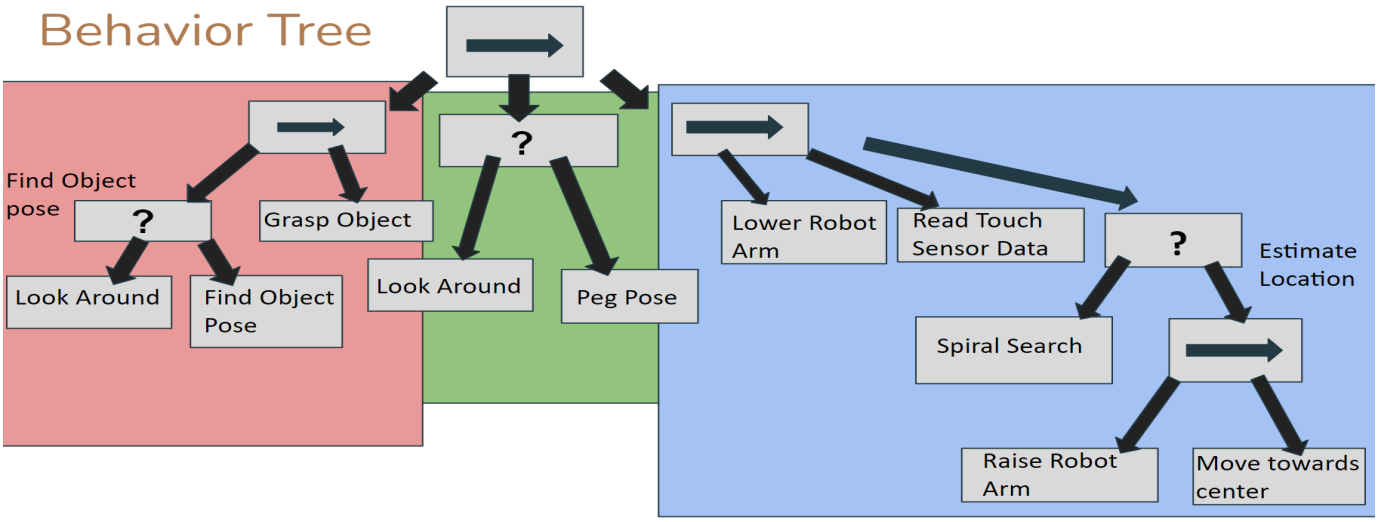

Fig. 8: Behavior Tree Framework