SmartTidyBot

Abhyudit Singh Manhas
dept. of Mechanical Engineering
Stanford University
Stanford, CA, USA
abhyudit@stanford.edu

Parth Rawri
dept. of Mechanical Engineering
Stanford University
Stanford, CA, USA
rawri@stanford.edu

***Abstract*—This paper proposes "SmartTidyBot" which extends the functionality of TidyBot [1] (a real-world robotic mobile manipulation system) for personalized physical assistance. [1] demonstrates TidyBot's capabilities on personalized household cleanup, where the robot moves every object on the floor to its "proper place" i.e. receptacle (such as drawer, closet, recycling bin, and black storage box) via. primitive selection (such as "Pick and Place", and "Pick and Toss"). One of the limitations of such robotic systems is the order in which the environment is cleaned. TidyBot approaches the task of cleaning the floor by relocating itself to the nearest possible object and placing/tossing the object to a receptacle. This approach may not be the most efficient. In this paper, we investigate the advantages associated with integrating an advanced decision-making system into the cleaning strategy of TidyBot. The revised cleaning strategy results in substantial improvements, particularly in navigating through excessively cluttered environments. Unlike the existing approach of always prioritizing the closest object, the enhanced system empowers TidyBot to intelligently assess the most optimal cleaning strategy. Figuratively, SmartTidyBot accomplishes the same tasks faster than TidyBot.**

I. Introduction

Robotic manipulations for carrying out tasks need to be minimal and efficient to carry out the tasks in least time and energy expense, thereby saving operational expense. Specifically, we optimize robotic manipulations using efficient path planning. Inspired by this objective we tackle one of the limitations identified in TidyBot [1] which highlights the need of redefining the cleaning approach, which holds the potential to enhance the overall efficiency of the system, specifically in terms of time consumption and operational costs.

In comparison to traditional path planning algorithms such A*[5], Dijkstra's Algorithm[4], RRT*[3], and Pure pursuit [2], formulating the problem as an MDP allows for stochasticity and efficiently dealing with high dimensional, complex spaces.

In the current setting, TidyBot is employed to assist in tidying up the floor, wherein the robot picks every object on the floor and relocates them to their designated locations. To achieve this task the robot iteratively locates the closest object on the floor, navigates to it, recognizes its category, picks it up, determines the appropriate receptacle for the object, and then either navigates to the receptacle and puts the object inside, or tosses object to the receptacle. The system overview [1] is described in Fig. 1. To map object categories to receptacles as per a user's preference, TidyBot uses the summarization capabilities of a LLM to infer personalized rules for both receptacle selection and manipulation primitive selection [6].

TidyBot uses two overhead cameras for 2D robot pose estimation (x, y, θ) and 2D object localization (x, y). Additionally, the decision making of mapping an object to a receptacle, and the choice of primitive is determined by a Large Language Model (LLM)[1]. For simplification purposes, we assume that TidyBot only relies on the overhead cameras for both localization and object categorization. Therefore, the agent is aware of different object categories and their respective receptacles and primitives before initiating the cleaning process.

The decision-making process is conceptualized as a mathematical model, specifically utilizing a Markov Decision Process (MDP). The objective is to derive the optimal policy through the application of value iteration. Subsequently, a rollout mechanism is employed to determine a sequence of optimal actions from the current state. To assess the efficacy of decision-making intelligence, we implement this methodology in TidyBot. In parallel, an alternative decision-making approach is considered, involving the use of the A* algorithm [5] within the framework of TidyBot. A comparative analysis between the MDP-based system and the A* algorithm-based system is conducted, with the primary evaluation metric being the total time taken for clean up.

II. Method

To address the task of personalized household cleanup, we formulate the problem as a grid world Markov Decision Process (MDP). Given that the state of the agent and objects is constantly known, we opt for an MDP over a Partially Observable MDP (POMDP). The MDP definition is as follows:

State Representation (S): Represents the information required for decision-making.

$$S = \{x, y, (o_1, x_1, y_1, r_1, p_1), \ldots, (o_n, x_n, y_n, r_n, p_n)\}$$

Here, (x, y) are the coordinates of the robot, and $(o_i, x_i, y_i, r_i, p_i)$ represents the information for the i-th object:

- o_i is a discrete number indicating the status of the object.
- (x_i, y_i) are the coordinates of the i-th object on the floor.
- r_i is the receptacle associated with the i-th object.
- p_i is the primitive associated with the i-th object.

For the purpose of this work we limit the space of receptacles and primitives as follows:

$$r_i = \{'drawer', 'closet', 'recyclingbin', 'blackstoragebox'\}$$

$$p_i = \{'pickandplace', 'pickandtoss'\}$$

For objects that are to be tossed, o_i can be either 0 or 1. If $o_i = 1$, the object is still on the floor, and if $o_i = 0$, the object has been tossed. For objects that are to be pick and placed, o_i can be -1, 0, or 1. If $o_i = 1$, the object is still on the floor, if $o_i = 0$, the object is currently being placed by the robot, and if $o_i = -1$, the object has been placed at its receptacle.
Since the robot cannot place two objects simultaneously, o_i can be 0 for only one object (with primitive: pick and place) at a time. Keeping this in mind, the dimensionality of the state space turns out to be:

$$|S| = N \times N \times (n_p + 2) \times 2^{n_{obj}-1}$$

where N is the grid size, n_p is the number of objects with primitive "pick and place", and n_{obj} is the total number of objects in the scene.

Action Space (A): Encompasses movement in eight directions (N, NE, E, SE, S, SW, W, NW), reflecting the decisions SmartTidyBot can make at each step.

$$A = \{\text{"North", "North East", "East", "South East",} \\ \text{"South", "South West", "West", "North West"}\}$$

Transition Probabilities (T): Assumes deterministic robot kinematics, describing how the system transitions from one state to another based on the chosen action. We update the object's coordinates based on the robot's current position. Additionally, if the robot comes in proximity to an object, o_i changes to 0 from 1, and if it comes in proximity to a receptacle (only for objects with primitive: pick and place), o_i changes to -1 from 0.

$$T(s' \mid s, a) = \begin{cases} 1 & \text{If } s' \text{ is the state resulting from taking} \\ & \text{action } a \text{ in state } s \\ 0 & \text{otherwise.} \end{cases}$$

Reward Function (R): Defines immediate benefits or costs associated with actions in a certain state. Two reward functions are introduced to guide object placement, penalizing non-relevant actions and motivating movement towards the appropriate receptacle based on the primitive associated with the last picked object.

If the last picked object has the "pick and place" primitive, a specific reward function is applied to encourage movement towards the corresponding receptacle.

$$R(s,a) = \begin{cases} -10 & \text{if robot moves over objects} \\ -1 & \text{if robot moves over receptacles} \\ -0.01 & \text{for movement} \\ +50 & \text{for reaching the receptacle} \end{cases}$$

For other cases, a general reward function guides the decision-making process.

$$R(s,a) = \begin{cases} -1 & \text{if robot moves over objects or receptacles} \\ -0.01 & \text{for movement} \\ +100 & \text{for reaching objects that are to be tossed} \\ +50 & \text{for reaching objects that are to be placed} \end{cases}$$

Algorithmic approach: Given the relatively small state space, we employ the value iteration technique for optimization. SmartTidyBot dynamically selects between two distinct reward functions based on the primitive associated with the last picked object. If the last primitive is identified as "pick and place," the algorithm employs the corresponding reward function tailored to motivate movement toward the designated receptacle. Conversely, for other primitives, alternative reward functions are integrated into the decision framework.

III. Evaluation

To quantify the enhancement in system efficacy, we conduct a comparative analysis of the time taken by the default baseline approach against the performance of SmartTidyBot, an advanced version of TidyBot empowered with an intelligent decision-making system.

Baseline approach simulation: We emulate the default baseline approach using an A* algorithm, initiating floor cleaning from the nearest object. When encountering an object with the "pick and place" primitive, the robot strategically places it in the corresponding receptacle before re-initiating the cleaning process from the closest object. This sequence continues until all objects on the floor have been successfully picked up. At all times the robot calculates the shortest collision-free path.

Experimental setup: For a fair evaluation, both approaches commence the cleaning process by randomly distributing n_{obj} objects on the floor, each assigned with a receptacle and primitive in a random fashion. Additionally, we position four receptacles at random locations on the floor.

This experimental setup ensures a realistic representation of real-world scenarios, allowing us to capture the intricacies of the cleaning process under varied conditions.

IV. Analysis & Results

As mentioned previously, the baseline approach uses an A* algorithm to plan collision-free paths, and the cleaning process is re-initiated from the closest object. Our approach formulates this problem as an MDP, and solves it using value iteration. Value iteration is run until convergence. It terminates when:

$$max\ |U - U_{prev}| \leq 10^{-3}$$

The optimal policy obtained from value iteration is then used to do a rollout, starting from an initial state (arbitrary x, y), with all the o_i's set to 1. We expect the o_i's to become 0 and -1 for objects that are to be tossed and placed, respectively. That is indeed what happens, and the rollout is stopped at this point. The cleaning is then simulated from this rollout, and

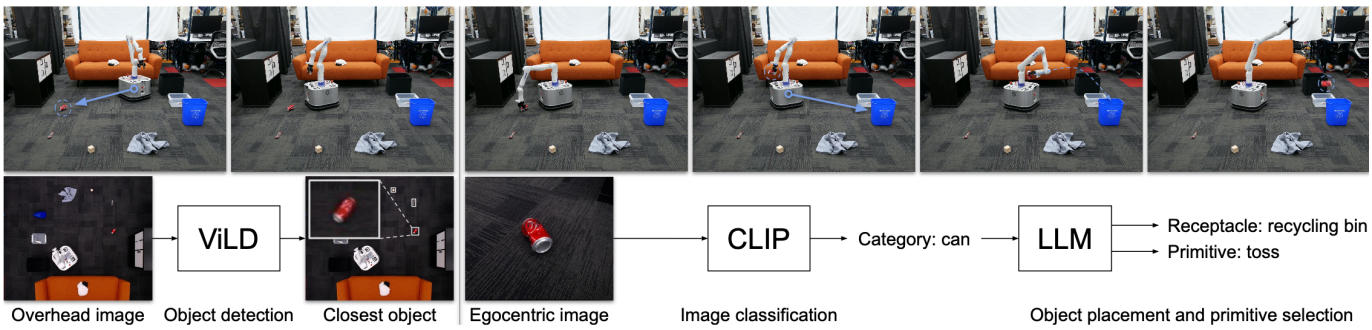

Fig. 1. System overview. Once the user's preferences have been summarized with an LLM, TidyBot will localize the closest object on the floor, move to get a close-up view with its egocentric camera, predict the object's category using CLIP, use the LLM-summarized rules to select a receptacle and manipulation primitive, and then execute the primitive to put the object into the selected receptacle, repeating this entire process until no more objects can be found on the floor [1].

compared with the baseline approach. We compare the total time taken for clean up. This is calculated as follows:

- We assume that the robot accelerates from rest from (x_i, y_i), reaches a peak velocity v_{peak}, and then decelerates to rest, to (x_{i+1}, y_{i+1}). The time taken for this is:

$$t_i = 2\frac{\sqrt{(x_{i+1}-x_i)^2 + (y_{i+1}-y_i)^2}}{v_{peak}}$$

- Hence, the total time taken for traversing the floor cluttered with objects is:

$$t_P = \sum_{(x_i,y_i)\in P} t_i$$

where P is the path.

- Let the time taken for tossing an object be t_t, and time taken for picking (or placing) an object be t_p. Hence, the time taken for cleaning is:

$$t_C = n_t t_t + 2n_p t_p$$

where n_t is the number of objects to be tossed, and n_p is the number of objects to be placed.

- The total time taken for the cleaning process is thus:

$$t_T = t_P + t_C$$

For both approaches, we set:

$$v_{peak} = 0.45\ m/s,\ t_t = 4\ s,\ t_p = 2.5\ s$$

We experimented with different values of N, the grid size, as well as with n_{obj} and n_t (note that $n_p = n_{obj} - n_t$). The time comparisons for some of these cases are summarized below: Based on the total time taken for cleaning, we can observe that our approach performs better, or at worse, the same as the baseline approach.

The cleaning process is animated as well, where objects "disappear" as soon as they are picked up. The snapshots of these animations are shown below for some different values of (N, n_{obj}, n_t).

TABLE I
TIME COMPARISONS FOR DIFFERENT APPROACHES

(N, n_{obj}, n_t)	t_T for Baseline	t_T for MDP policy
$(20, 5, 3)$	250.69 sec	232.47 sec
$(25, 5, 2)$	410.92 sec	400.33 sec
$(25, 6, 3)$	423.87 sec	405.65 sec
$(25, 6, 4)$	328.47 sec	319.12 sec
$(50, 7, 4)$	1067.12 sec	1021.75 sec
$(50, 7, 3)$	1089.23 sec	1091.11 sec

V. Future Work

In our problem statement, we made the assumption that the primitive and receptacle for each object are known through overhead cameras even before initiating the cleaning process. This simplification may not fully capture the challenges posed by real-world scenarios where object recognition might be less straightforward.

Algorithmic Decision Making: The current approach might face computational inefficiencies as the grid size and the number of objects increase. We can see that $|S|$ increases with grid size N, and increases exponentially with number of objects n_{obj}. Exploring alternative decision-making algorithms, particularly online planning methods, could enhance the efficiency and scalability of the system, especially in more complex environments.

Reward Engineering Challenges: While we dedicated significant time to reward engineering, there is acknowledgment that there's room for further improvement. Refinement in the reward structure could potentially lead to a more nuanced and effective learning process, enhancing the overall performance of the system.

VI. Conclusion

In this study, we present the advantages of integrating a decision-making system to enhance the efficiency of robotic manipulations, resulting in significant time and operational cost savings. Our approach involves formulating the problem as a Markov Decision Process (MDP), applying value iteration to solve it, and implementing intelligent navigation for a mobile robotic manipulation system through the use of rollout

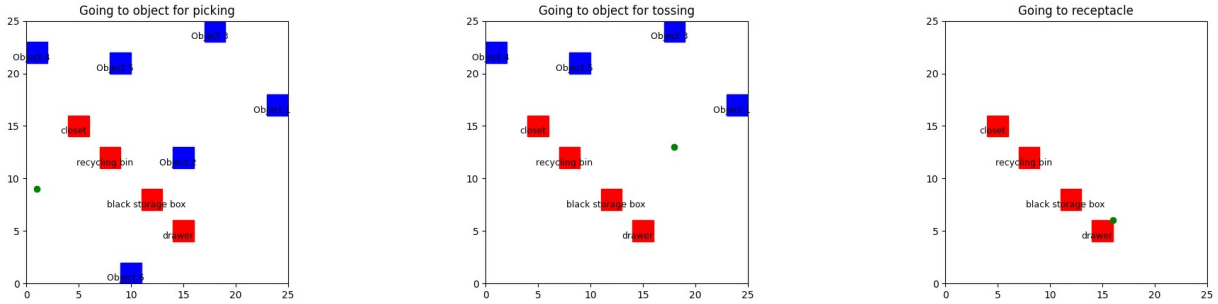

Fig. 2. Different stages of the cleaning process for $(N, n_{obj}, n_t) = (25, 6, 3)$. The three snapshots above are taken at different stages of the cleaning process. The figure on the left is the start of the cleaning process, the center figure is in the middle of the cleaning, and the figure on the right is the end of the process. The objects and receptacles are denoted by blue and red squares, respectively. The green dot represents the robot.

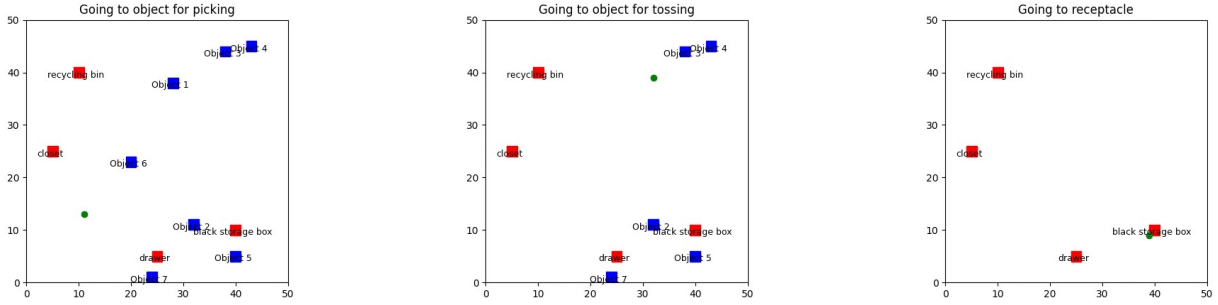

Fig. 3. Different stages of the cleaning process for $(N, n_{obj}, n_t) = (50, 7, 4)$.

techniques. To validate the effectiveness of our proposed system, we conducted a comparative analysis with a robotic system relying on the A* algorithm, a commonly used path-finding technique. Our results demonstrate the superiority of our approach, showcasing enhanced performance and efficiency in robotic manipulations. Furthermore, to substantiate our findings, we present comprehensive time comparison results between our decision-making system and the A* algorithm-based robotic system. These comparisons highlight the tangible benefits of our approach in terms of time savings and operational cost efficiency. In summary, our work not only introduces a decision-making system for optimizing robotic manipulations but also provides empirical evidence of its superiority over a conventional A* algorithm-based approach.

VII. Team Contributions

Parth Rawri primarily contributed to the problem selection, formulating the problem as an MDP, and implementation of the solution using value iteration. Abhyudit Singh Manhas primarily contributed to the development and simulation of both approaches. Additionally, he contributed to testing of the proposed approaches. Both authors contributed to the writing of this paper, brainstorming of the MDP solution, reward engineering, and arriving at the executable scripts.

The code for this project can be found here. The code for A* and value iteration is in `astar.py` and `value_iteration.py` respectively. The simulations and animations can be run in the jupyter notebook `sim_final.ipynb`. The animations for the A* approach and the MDP approach for $(N, n_{obj}, n_t) = (25, 6, 3)$ can be found here.

References

[1] Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, Thomas Funkhouser, "TidyBot: personalized robot assistance with large language models," *Autonomous Robots*, vol. 47, no. 8, pp. 1087–1102, Nov. 2023. DOI: 10.1007/s10514-023-10139-z. ISSN: 1573-7527, Publisher: Springer Science and Business Media LLC.

[2] R. Craig Conlter, "Implementation of the Pure Pursuit Path Tracking Algorithm," *The Robotics Institute, Carnegie Mellon University*, Pittsburgh, Pennsylvania 15213, January 1992.

[3] Sertac Karaman, Matthew R. Walter, Alejandro Perez, Emilio Frazzoli, Seth Teller, "Anytime Motion Planning using the RRT*," *2011 IEEE International Conference on Robotics and Automation*, pp. 1478–1483, 2011. DOI: 10.1109/ICRA.2011.5980479.

[4] Willem Feijen and Guido Schäfer, "Dijkstra's algorithm with predictions to solve the single-source many-targets shortest-path problem," *arXiv preprint arXiv:2112.11927*, 2023.

[5] Daniel Foead, Alifio Ghifari, Marchel Budi Kusuma, Novita Hanafiah, Eric Gunawan, "A Systematic Literature Review of A* Pathfinding," *Procedia Computer Science*, vol. 179, pp. 507–514, 2021. ISSN 1877-0509, DOI: 10.1016/j.procs.2021.01.034.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., "Language Models Are Few-Shot Learners," *Advances in Neural Information Processing Systems*, 2020.