Cooperative Autonomous Search Robots

by

Matt Calhoun

Maryville University

December 2023

As the use of robots has dramatically increased in many areas of industry and life, we can see numerous examples of well-designed solutions to everyday problems. As one example, robotic lawn mowers are beginning to phase out the weekly chore of homeowners spending many hours performing repetitive lawn care. Robotic mowers could be a great thing. In addition to adding useful hours to a homeowner’s summer, they silently replace a weekly fossil fuel-burning ritual with very regular maintenance, that can be less damaging for the grass, as more frequent cuttings lead to a smaller percentage of the blade being cut at any one time.

A similar reduction in the burden of homeownership can be seen inside the house as well. The task of regularly vacuuming at some interval, while dust and debris accumulates in between vacuuming is being phased out and replaced by robotic vacuums. These robots are replacing the chore of vacuuming with regularly scheduled cleanings, typically when the homeowner is out.

Easing the maintenance burdens of homeownership is certainly a worthwhile and profitable goal. However, when the application of robots in these areas shifts from residential focus to commercial focus, the goals would be quite different. For residential robot applications, some of the goals while designing such a system could include (toward the top of the list being higher priority):

* Ease of use by the homeowner
* Reasonably priced
* Very low maintenance
* Performs the task reasonably well
* Quiet
* Performs the task efficiently and quickly
* Consumes very little power

If we were to take either mowing or vacuuming applications and target commercial customers with the same type of products, the priorities could be significantly different. For a commercial setting, the prioritized goals of designing the system could look more like this:

* Performs the task efficiently and quickly
* Performs the task very well
* Very low maintenance
* Uses power efficiently
* Reasonably priced

Notice that in the theoretical priority list for commercial applications, the priority of performing efficiently and quickly has shifted toward the top. Also, given that in a commercial setting, the cost of a technology can often be offset by productivity gains (ability to handle more customers), a different approach to building robotic solutions can be taken. Rather than relying on a single robot to be very good and fast at a certain task, the task should be divided across multiple robots, which can be scaled up as a situation demands.

**Divide and Conquer**

Imagine that a commercial landscaper drives up to the front of your home, opens a trailer door, and 20 or so robotic mowers methodically and quickly perform your regular yardwork in a matter of minutes. That is what I want to make possible. I would like to build a solution that would enable retrofitting of many well-designed 'loner’ robotic systems to take advantage of the presence of multiple robots working on the same problem.

To apply any kind of divide-and-conquer technique to a set of robots, there are a few prerequisites that will be required no matter how we tackle this problem. It is a given that a set of robots needs to be able to physically accomplish their targeted task(s). I am going to presume a set of robots is already capable of doing their task(s) well individually. I will target just the ability to work cooperatively.

|  |  |
| --- | --- |
| Positioning | Robots must be aware of *where* they are at all times. |
| Context | Robots must be aware of where they have been and what they have done. Robots must also be aware of what other robots (and people) have already done in working toward a common solution, as well as where other robots are at any given time. |
| Strategy | Leveraging all the position and context awareness available to them, robots must be able to decide what step to take next at any point in time. |

**Agent vs Robot**

Although I have been referring to the mowers and vacuum cleaners simply as ‘robots,’ from this point forward, I will distinguish between the *host* robot (the mower, etc.) and the system controlling the robot. When referring to the physical robot, I will use the term *robot*, but when referring to the cooperative controlling system, I will use the term *agent*.

**Positionining**

In many cases, GPS may be a workable solution for positioning, but when portions of the sky may be blocked by trees or buildings, it may not be feasible. Also, GPS is usually not going to be an option for agents working indoors. As an example, a couple of residential robotic products, along with a description of their available navigation system is displayed below.

|  |  |
| --- | --- |
| iRobot Vacuum Cleaner (iRobot Corporation, 2022) | Husqvarna Automower® robotic lawn mowers (Husqvarna US, 2023) |
|  | EPOS Home Setup Landing Page Image WEB USE ONLY |
| vSLAM® (Vision Simultaneous Localization and Mapping) technology | Husqvarna EPOS™ (Exact Positioning Operating System) |
| Uses LIDAR to explore and build a map of walls and doorways. Combining the map with live LIDAR data, the robot can estimate position. | Utilizes a permanently fixed transmitter with an unobstructed line of sight to the robot. |

**Decoupling Positioning**

Making an agent that that can be deployed across a wide variety of products will require the solution to be decoupled from the actual positioning system. To accommodate this, I will assume that whatever navigation system is in use can be scaled and mapped to a simple cartesian coordinate system. Within this common coordinate system, all maps will be rectangular-shaped, in order to simplify problems that will need to be solved a little later. In the case of oddly shaped maps (inside a house, for instance), a rectangular boundary will be drawn around all farthest points of the VSLAM-generated map. The walls will be considered obstacles within the map. In the case of Husqvarna’s EPOS system, a rectangular boundary will be imagined that includes the farthest points of all property lines, with untraversable areas (buildings, ponds, etc.) will be considered obstacles.

**Context**

The second requirement for building this cooperative system is context. Since we are building a system that will require a retrofit of some sort anyway, I am going to take the liberty of adding an SBC to the robots, which will serve as the hardware that hosts the agent software. Specifically, our agent will require a Raspberry Pi CM4 (Raspberry Pi Foundation, 2023) to be onboard the robot, which will provide us with Bluetooth, Wi-Fi, and enough computing power to run TensorFlow TFlite models (Google, 2023).

Now that we have a common computing platform with wireless communication abilities, we need a way for an agent to remember its own movements and actions, as well as the ability to see other agents’ movements and actions. One way to accomplish this is to have an onsite Wi-Fi network running a simple REST service that enables connected agents to log their positions and actions, as well as retrieve those of other agents participating in the current task.

The Raspberry Pi CM4 is easily capable of hosting such a REST service, and even a database that can serve as a data store for the service. One or more of the agents will host this REST service and database at all times. When a new group task begins, one of the agents will be selected as the system of record, and from that point until the task is complete, all agents will be logging and retrieving context to and from the selected agent. I will build the required set of REST services using the Django Rest Framework (Django Rest Framework Developers, 2023) as the service layer and Postgres (The PostgreSQL Global Development Group, 2019) as the data layer.

**Strategy**

Devising a strategy for each agent to independently choose what action to take, to cooperatively complete a task is the most challenging aspect of this solution. We do not want agents to overlap one another’s work in most cases. We also do not want the group of agents to use any more combined time or energy than necessary to complete a task. An algorithm could be designed that considers the position and context, generating a reasonable next step for an agent to take. This algorithm would be complicated to make, and given AI capabilities available to us today, building such an algorithm manually is entirely unnecessary. I would prefer to utilize reinforcement learning to generate a model that can predict the best course of action at any time.

**Applying Reinforcement Learning**

When leveraging reinforcement learning techniques to solve this problem, I need to clearly define the problem scope, as well as reframe it slightly. To completely isolate my solution in such a way that I am only solving the cooperative aspect, I am going to decouple the concept of mowing, vacuuming, or any other task from the agent and the system. The agents will simply be performing *a task* over an area, the details of which are left up to the robot to complete. Next, I am going to make some assumptions.

* An agent can travel to any unobstructed point in the *field* (yard, room, etc.).
* An agent can accomplish the selected task within a pre-defined radius of any unobstructed point in the field.
* Either the task or the travel can fail, which must be accounted for.
* When an agent has successfully completed a task at a certain point in the field, the task is considered *complete* within that radius.
* The task *could* be time limited (cover as much area as possible within a given amount of time)

**Training an Agent**

I will be utilizing (Stable Baselines 3 Stable Baselines Developers, 2023) as the reinforcement learning system to train the agent. This will require building simulation of a field, obstructions, actions, movement, and other aspects of the problem. I will build a simulation that does mimic movement through an environment, including some of the unpredictability of that. The simulation will allow for any number of robots. The simulation will also allow for randomly generated maps of varying sizes, with obstructions of varying sizes randomly sprinkled throughout the map. When I have a functional simulation, it will be wrapped within a Gymnasium (Farama Foundation, 2023) environment which enables Stable Baselines 3 to train agents using my simulation.

As a placeholder for the robot-specific task, I will choose *search* and *photograph* as stand-in tasks, which will help agents learn to identify things within a medium-sized and small radius of any point. Both tasks will occasionally fail, helping the agent learn to react to occasional failures. In order to ensure adequate area coverage, my simulation will place targets at random locations throughout the field, and agents’ success will be measured against their ability to always find all targets. Depending on the variety of applications of this system, it is likely that more than one agent type will need to be trained, and the selection of the appropriate agent will be up to the end user, based on their needs.

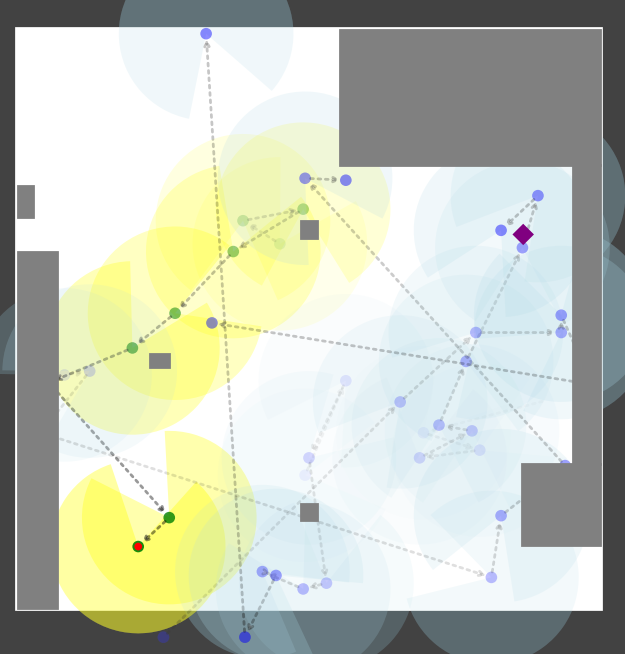
**Action Space**

I will first define a common set of actions that any agent can take. The actions available are as follows:

|  |  |
| --- | --- |
| Look | Check the area for a target item |
| Photograph | Capture a good image of the item |
| Nothing | Rest |
| ReportFound | Report a finding to the Rest service |
| GoForwardShort | Go forward a tiny percentage of the map length |
| GoForwardMedium | Go forward a small percentage of the map length |
| GoForwardFar | Go forward a decent percentage of the map length |
| GoReverseShort | Go reverse a tiny percentage of the map length |
| GoReverseMedium | Go reverse a small percentage of the map length |
| GoReverseFar | Go reverse a decent percentage of the map length |
| RotateLeftSmall | Rotate left by a few degrees |
| RotateLeftMedium | Rotate left by ~10 degrees |
| RotateLeftBig | Rotate left by ~45 degrees |
| RotateRightSmall | Rotate right be a few degrees |
| RotateRightMedium | Rotate right by ~10 degrees |
| RotateRightBig | Rotate right by ~45 degrees |

**Observation Space**

Within reinforcement learning, the observation space refers to the *context* aspect of our problem. In order to provide a single snapshot of context, including past details to both the reinforcement learning process and to the agents in a live setting, I will create an image rendering utility that pulls position and action logs from the current agent and well as other agents (using the Rest service), and generates an image containing all relevant context. The image will be from the current agent’s point of view, even though it will show all agents. An example of the rendered field image is shown below.

In the rendered field image, the current agent is shown as a red dot. Other agents are shown as blue dots. Any searched radius is shown as a partial circle, with an appropriate blind spot removed from it. A time element is added to this image, as events that happened further in the past become more transparent on the image. Yellow vs blue search patterns distinguish between searches done by the current agent and searches done by cooperating agents. Obstacles are marked as gray rectangles. In-bounds areas of the map are marked by white. This image will serve as input to the agent during RL training and during later simulations and actual physical tests.

**Stable Baselines Configuration**

Many combinations of agent types and architectures will be tested, but as a first attempt, I will train a DQN (deep Q network) agent, which will utilize a Convolutional Neural Network (Russell, S., Norvig, P. 2022) as its policy model. During training, the CNN will learn to analyze the rendered field image and identify patterns and behaviors that result in the desired outcome (finding all targets). Training a decent CNN on this image will take quite a while (likely weeks).

**Loner Agent Strategy**

The trained CNN will become the basis for a *loner* agent’s strategy. The agent will get a live field rendering, feed it to the CNN, and receive back a behavior that is predicted to be the best choice. The agent will then perform the selected action. The cycle repeats until the task is complete. Although this agent should be able to complete a task efficiently, it still falls short of the original goal of *cooperative* agents working together.

**Training Cooperative Agents**

I will again run the reinforcement learning process on a fresh/untrained agent. However, this time, the simulation upon which training is done will include *drone* agents. The drone agents are not being trained, but rather independently following the *loner* agent strategy. The drones will be working toward a common goal, but without any awareness of how not to perform redundant actions of other agents. This time, the agent being trained will learn to avoid retracing other agents’ travel paths and actions. The agent-in-training will learn to quickly find any items the other agents have not yet found and to search areas being neglected by other agents. Once this agent has completed training, I will run simulations using exclusively *group-trained* agents, such as this one. I will observe how they perform searches and assess if any follow-up training cycles are needed and what the nature of those cycles would be.

**Deploying Agents/Robots**

Retrofitting an existing robot to utilize this system would involve adding the additional hardware (Pi CM4), adding a Wi-Fi network (a portable router mounted on one of the robots), and mapping appropriate robot behaviors to agent’s action space. The deployed agent will behave and make decisions as if it were still in a simulation, although it will be performing live real-world actions.

**Challenges**

Any number of challenges can be identified in building the system as described above. However, assuming agents can be fully trained, robots can be properly retrofitted, and the system accomplishes tasks, when deployed against real world tasks, there will certainly be unforeseen scenarios, limitations, and edge cases. It is likely that this system will be a good fit for certain applications and not for others, and these limitations may not be immediately obvious.

The system will need to be thoroughly tested in each different area where it is to be applied, searching for these limitations and edge cases. As they are identified, they may be overcome by retraining agents with slightly different goals and architectures, changes to the observation space, or changes to the action space.

**Current Project Status**

I am currently developing this system. All the code is available on my GitHub account (see references). Within my GitHub account, the code relating to high-level control of robots is within the *Pilot* repository and the code relating to simulation and reinforcement learning is contained within the *LVPS Simulation* repository. The project is still highly experimental and under active development.

References

Calhoun, M. (2023). *LVPS Rest Service*. GitHub.com. <https://github.com/mattcalhoun1/LVPS/wiki/LVPS-Rest-Service>

Calhoun, M. (2023b). *Source Code : LVPS Simulation*. GitHub.com. <https://github.com/mattcalhoun1/LVPS_Simulation/tree/main>

Calhoun, M. (2023c). *Source Code: Pilot*. GitHub.com. <https://github.com/mattcalhoun1/Pilot>

(Django Rest Framework Developers, 2023). *Django Rest Framework*. Djangorestframework.com. <https://www.django-rest-framework.org/>

Farama Foundation. (2023). *Gymnasium Documentation*. Gymnasium.farama.org. <https://gymnasium.farama.org/>

Google. (2023). *Object Detection with TensorFlow Lite Model Maker*. TensorFlow. <https://www.tensorflow.org/lite/models/modify/model_maker/object_detection>

Husqvarna US. (2023). *AUTOMOWER® 450XH EPOS*. Husqvarna US. <https://www.husqvarna.com/us/robotic-lawn-mowers/automower-450xh-epos/>

Husqvarna US. (2023). *Husqvarna EPOS*. Husqvarna US. <https://www.husqvarna.com/us/discover/epos/>

iRobot Corporation (2022). *Energy Efficiency and Performance | iRobot*. IRobot.com. <https://about.irobot.com/Corporate-Social-Responsibility/Energy-Efficiency-and-Performance>

Raspberry Pi Foundation (2023). Raspberry Pi Compute Module 4. <https://www.raspberrypi.com/products/compute-module-4/>

Russell, S., Norvig, P. (2022). *Artificial intelligence, A Modern approach*. (4th ed.). Boston: Pearson.

Stable Baselines Developers (2023). *Stable Baselines 3*. Stablebaselines3.com. <https://stable-baselines3.readthedocs.io/en/master/>

The PostgreSQL Global Development Group. (2019). *PostgreSQL: The world’s most advanced open-source database*. Postgresql.org. <https://www.postgresql.org/>