MedBot: an Ambulance Robot Controller

Ibraheem Alhashim
School of Computing Science
Simon Fraser University

April 11, 2009

Abstract

In this paper, we present a simple implementation of an ambulance robot controller. The responsibility of the ambulance in the system is to transport dead robots from their current location to a nearby charging station. This helps in solving the problem of having dead robots in the environment resulting in a more robust and autonomous system. Results from a simulated MedBot controller indicate a slight increase in system performance. The implementation was done using the robot simulator Stage from the Player Project.

1 Introduction

One of the most important aspects of an autonomous mobile robot is its ability to stay functional for long periods. In order to achieve full autonomy with no human intervention, robots must charge their selves when they are low on energy[2]. The simplest approach that humans use for most rechargeable devices is to recharge the device's battery upon reaching a certain level of remaining stored energy. This is usually the case for devices that relay on sources of energy that are only physically available in certain locations in the system's environment.

There are several proposed methods for calculating a good estimate of the energy level reached that would trigger a recharging state[5]. However, finding the best recharging threshold does not guarantee a long lived fully autonomous system. This is especially the case when a robot controller is used on real world robots that operate in unpredictable environments. An example of such environment would be an underground mine in which different sections of the cave are vulnerable to collapses[1]. A mining robot might calculate a recharging threshold based on one of the available routes. If that route no longer exists the robot will run out of energy and will eventually reach a dead state.

Recovering dead robots have not been specifically studied in the literature since it falls under the simple task of picking up objects. This paper describes a simple implementation of such system and present the dynamics involved in using an ambulance robot.

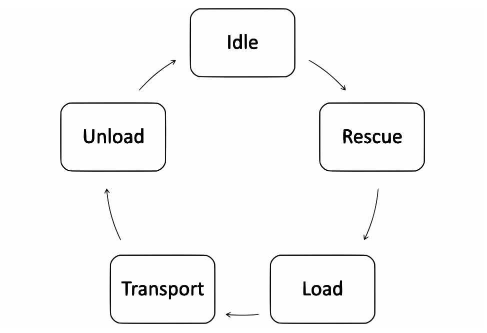


Figure 1: MedBot work cycle

2 Motivation

Most robots are specifically designed to perform certain tasks and would need human intervention for matters relating to maintenance and system failure[6, 3]. For a fully autonomous system, the robots need to relay on themselves to recover from such situations. Designing a robot that performs its tasks along with a recovery mechanism is extremely complicated and in some cases impossible. If a robot loses the ability to control its movement it will not be able to help itself. The only possible solution would be outside interference. Therefore, the need arise for a specialized robot that helps achieve such positive interference.

The ambulance robot described in this paper helps immobile robots reach a destination where it is possible for them to recharge or get fixed. In this paper we are concerned with the case of running out of battery and the need to recharge. A system with such robots allows for the possibility of performing jobs that require energy beyond the threshold specified for a worker robot. The situation is also similar to working in an unpredictable environments where paths may change and the threshold does not work for the modified environment.

The idea of having a system with different robots designed for different tasks relates to real world systems such as hospitals or construction sites. In these systems, different people with different skill sets are assigned to different jobs. All agents in the system are needed to achieve robustness and good performance.

3 Controller Design

The robot controller implemented for the MedBot ambulance is a state based controller. The underlying code is based on the Fasr controller supplied with the Stage 3.1.0 simulator from the Player Project[4]. Most of the navigation aspects of the controller are predefined and are specific to one environment called the Cave map. The ambulance follows five major states in every work cycle shown in Figure 1.

3.1 Idle State

In this state, the robot tries to locate a special charging station that can only supply an ambulance robot with energy. The specialization of charging stations helps reduce the traffic congestion by separating the different types of robots. Once the robot gets close enough to such charging station, it will remain in a parked status. The only movement needed at this state while in the parked status is obstacle avoidance. This is helpful for situations when a moving object approaches the ambulance and the only possible way to avoid a crash is to move away.

For the implementation described in this paper, the MedBot unit monitors the battery level of all worker robots in the environment. If a robot is found to be in a dead state, where the energy level is below a certain threshold, its state would change to the rescue state.

3.2 Rescue State

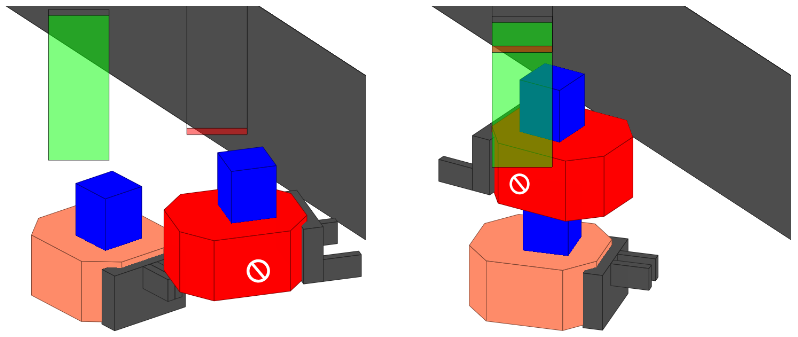
In this state, the MedBot robot tries to navigate itself close enough to a specific dead robot. The location of the dead robot on the map is known to the MedBot. The MedBot uses a predefined map of the environment and estimates a good route from its position to the dead robot area. It is worth mentioning that using good navigation schemes would result in a faster more efficient effort for a rescue job. Once the MedBot reaches a significantly close distance to the dead robot it will change its state into the loading state.

3.3 Loading State

The dead robot needs to be loaded onto the MedBot body for transportation. There are several concepts of the notation of loading. These include lifting the dead robots over the MedBot, dragging it, pushing it, or possibly in an open environment one can design the MedBot to conduct an airborne rescue operation.

The implementation for this paper does not take into account the mechanics of lifting the dead robot and assumes that it is a trivial task. For the sake of simplicity it is assumed in the implementation that the dead robot would be placed on top of the MedBot. Figure 2 shows the loading state during a simulation.

There are several possibilities for a more realistic design. One possible design is to include arms, or forks, similar to the ones used in waste collection vehicles. Another possible design may include a type of arm that extends to the dead robot and hooks it self to its body and drag it (e.g. tow trucks).Once the dead robot is on-board the MedBot controller will change state into the transport state.


(a) A MedBot lifting a dead robot

(b) A dead robot positioned on top of a MedBot

Figure 2: Loading state

3.4 Transport State

The MedBot needs to transport the dead robot to the assigned charging station area. The MedBot can locate the area by following the same navigation plan used in locating its own charging station with different coordinates. An ambulance robot should take into consideration its new form and must deal with issues like its own battery level, power consumption, and obstacle avoidance with a robot on board. In our implementation the only change considered was the threshold considered for a low battery. If a MedBot has a dead robot on board, its threshold would be lower than it is in the idle state.

3.5 Unloading

The optimal option for unloading a dead robot would be for the MedBot to try and place the dead robot in its designated charging station. Care must be taken for this mechanism to work since there are a number of possibilities where it can go wrong. If a robot miscalculates the position of the charging station and leaves a dead robot in such state then the charging station would be nonfunctional since no other robot can use it. Also, if a robot does not anticipate another robot docking into the same charging station while it is unloading then the probability of each robot's sensor not detecting the other would increase and would result in a crash.

The unloading operation implemented for this paper uses a simple solution that works when the dead robot is not functional because of its lack of energy. The solution is easy to implement and takes care of the problems resulting in the unloading process. Once the MedBot reaches the area of a charging station it will try to unload the dead robot close enough to the station but not too close as to cause collisions. Before leaving the dead robot, the MedBot triggers an override mechanism in the dead robot making it change its status from dead to alive. The robot would then be able to fall back into its regular charging routine. For this to work, the dead robot needs to signal its dead status while maintaining a small amount of energy enough to allow it to dock to a charging station nearby.

Once the MedBot finishes the unloading operation it will change its status to idle and the entire cycle repeats again.

4 Experiments

Several experiments have been conducted with, and without, the help of MedBot units. The goal of the experiments is to investigate the impact of having an ambulance robot withing an existing system.

4.1 Setup

The environment in which the experiments were conducted is the Cave environment. This environment is provided within the Stage simulator (version 3.1.0). It encloses an area of 256m^2 with two separated areas consisting of a source and a sink for simulated robot tasks (referred to as flags). The map also includes an area where four charging stations are located along one of the walls.

The experiments were done with two MedBot units located near the flags source area. These units share two of their own special charging station. The location of these charging stations is arbitrary and it was chosen in an area that does not result in much interference with the traffic of worker robots. Figure 3 shows the initial setup of each experiment.

4.2 Autonomous Operation

Several experiments were conducted without the assistance of MedBot units to find a suitable and conservative estimate of the battery threshold for the worker robots. The benchmark for such estimate was set to be 10 hours of autonomous work of collecting flags from the source area to the sink destination. The system is found to be stable around the value 46%. Although this value seem to be large, the experiments have shown that thresholds with lower values result in robots dieing before the end of the 10 hours benchmark.

4.3 MedBot

By experimenting with thresholds lower than 46%, we can test the performance of the system with the aid of our ambulance robots. A simple performance metric of flags collected per hour is used in all of the experiments. Figure 4 shows the rate of flags collected in four different experiments with the rates 46%, 30%, 20% and 10%. The total number of flags collected during the entire simulation time is shown in Figure 5.

In the case of a 10% threshold, we notice a highly unstable rate of collection. This is a result of high dependency on MedBot that in turn results in higher

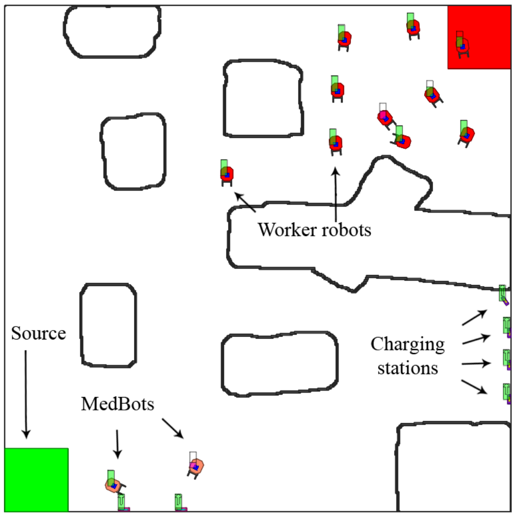


Figure 3: The Cave environment. Worker robots are located close to the sink area in the top right corner. The two MedBot units can be seen at the bottom left corner near the source area.

rates of dead robots, thus, reducing the system's performance. Table 1 shows the average number of flags collected per hour during each 10 hour experiment with the different thresholds. The best performance was achieved when the threshold is at 20%.

Balancing between having a conservative estimate and depending heavily on the MedBot robots results in the best performance. The experiments showed that a 20% threshold, for the current environment, is a good estimate for the worker robots. In the case where such estimate breaks, the MedBot's response compensate for the error and the system returns back to normal.

4.4 Analysis

The results of our experiments showed a slight increase in the system performance due to the inclusion of MedBot robots. The idea of having a backup plan for a low threshold can be thought of as being equivalent to having an extra battery capacity. Another advantage of having a MedBot is its ability to clear traffic congestion caused by dead robots especially in bottlenecks throughout the environment. One last practical advantage worth mentioning is one for robot

Hunger at 46%	30% with MedBot	20% with MedBot	10% with MedBot
294	296	323	301

Table 1: Average flags collected per hour

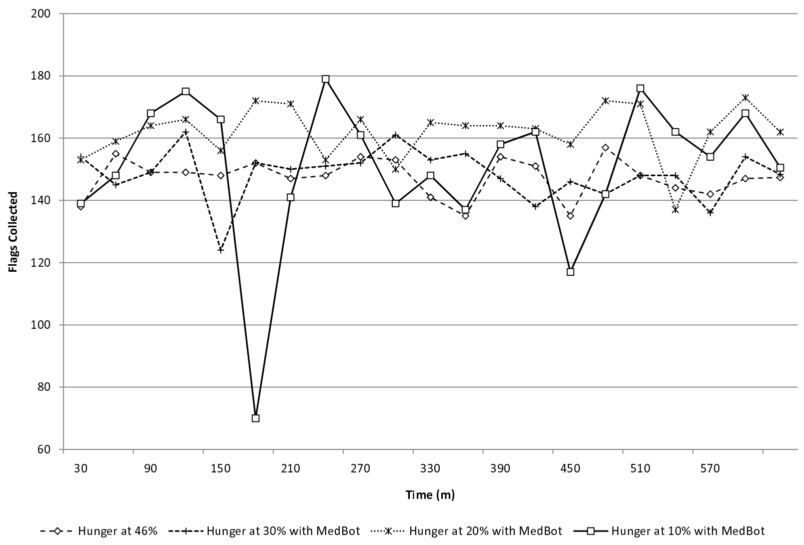


Figure 4: Rate of flags collected

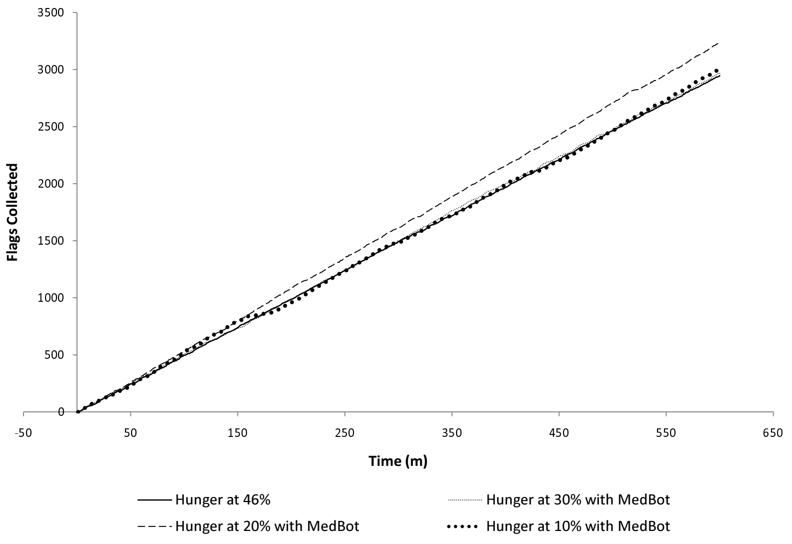


Figure 5: Total number of flags collected in a 10 hours period

designers. Having an ambulance robot allows designers to conduct real life tests of their robots on the field while knowing that a MedBot will return faulty or dead units back to safety.

However, adding agents to a system increases complexity and comes with a number of disadvantages. The presence of MedBots in the system increase traffic which in turn causes more traffic jams. This is especially the case when the MedBot is in the loading state. Also, allowing robots to reach a dead state can bring a system to a halt due to a cascading effect of robots dying. It is also worth mentioning that the energy required to operate the MedBot is more than the energy required for a worker robot to reach a charging station on its own. This is due to the effort needed by a MedBot to travel from its current location to the dead robot and from the charging station back to its idle location.

5 Conclusions and Future Work

In this paper we presented results of a simple ambulance robot controller. Such robots are useful for recovering other robots that are in a dead state. The incorporation of such robots in a system has the potential to increase its productivity. This however, comes with a price of increased energy cost and complexity.

It would be interesting to further investigate the effects of varying the design and mechanics of such ambulance robots on their performance. Methods including active monitoring of dead robots can be further tested for environments that only allow for limited communication. Also, tests on environments with dynamically changing routes can help validate the necessity for such robots.

References

[1] Arabasz W.J. Pankow K.L. Burlacu R. Pechmann, J.C. and M.K. McCarter. Seismological report on the 6 aug 2007 crandall canyon mine collapse in utah. *Seismological Research Letters*, 2008.

[2] M. C. Silverman, D. Nies, B. Jung, and G. S. Sukhatme. Staying alive: a docking station for autonomous robot recharging. In *Proc. IEEE International Conference on Robotics and Automation ICRA '02*, volume 1, pages 1050–1055, 11–15 May 2002.

[3] Illah Nourbakhsh Terrence W. Fong and Kerstin Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 2003.

[4] Richard Vaughan. Massively multi-robot simulation in stage. *Swarm Intelligence*, 2(2):189–208, December 2008.

[5] Jens Wawerla and Richard T. Vaughan. Near-optimal mobile robot recharging with the rate-maximizing forager. 4648:776–785, 2007.

[6] J. Yuh. Design and control of autonomous underwater robots: A survey. *Auton. Robots*, 8(1):7–24, 2000.