Symbiotic-Autonomous Service Robots for User-Requested Tasks in a Multi-Floor Building

Manuela Veloso[1], Joydeep Biswas[2], Brian Coltin[2], Stephanie Rosenthal[1],
Susana Brandao[3,4], Tekin Mericli[5], and Rodrigo Ventura[4]

***Abstract*— Although since the days of the Shakey robot, there have been a rich variety of mobile robots, we realize that there were still no general autonomous, unsupervised mobile robots servicing users in our buildings. In this paper, we contribute the algorithms and results of our successful deployment of a service mobile robot agent, CoBot, in our multi-floor office environment. CoBot accepts requests from users, autonomously navigates between floors of the building, and asks for help when needed in a symbiotic relationship with the humans in its environment. We present the details of such challenging deployment, in particular the effective real-time depth-camera based localization and navigation algorithms, the symbiotic human-robot interaction approach, and the multi-task dynamic planning and scheduling algorithm. We conclude with a comprehensive analysis of the extensive results of the last two weeks of daily CoBot runs for a total of more than 8.7 km, performing a large varied set of user requests.**

I. Introduction

We have been pursuing the goal of deploying multiple autonomous mobile robots capable of performing tasks as requested by users in our multi-floor building. There are several sub-problems to address:

1) Localizing and navigating autonomously and safely
2) Providing an intuitive interface for users to schedule tasks for the robot
3) Scheduling conflict-free task plans for each robot
4) Interacting with humans
5) Overcoming robot limitations to perform tasks

There has been considerable work in the robotics community to solve each of these sub-problems individually as well as combinations of these problems. However, rarely are all of these goals addressed simultaneously on a single platform. The robots Shakey [1], Xavier [2], and museum tour guide robots [3], [4], [5] have addressed some of these problems to varying degrees of success. Additionally, we have been inspired by the contributions of RoboCup@Home [6], a competition for autonomous indoor service robots with a wide scope of human-interaction challenges.

[1]M. Veloso and S. Rosenthal are with the Computer Science Department, Carnegie Mellon University, USA `mmv, srosenth at cs.cmu.edu`

[2]J. Biswas and B. Coltin are with The Robotics Institute, Carnegie Mellon University, USA `bcoltin, joydeepb at cs.cmu.edu`

[3]S. Brandao is with the ECE Department, Carnegie Mellon University, USA `sbrandao at ece.cmu.edu`

[4]S. Brandao and R. Ventura are with the ECE Department, Instituto Superior Tecnico, Lisbon, Portugal `rodrigo.ventura at isr.ist.utl.pt`

[5]T. Mericli is with Department of Computer Engineering, Bogazici University, Istanbul, Turkey `tekin.mericli at boun.edu.tr`

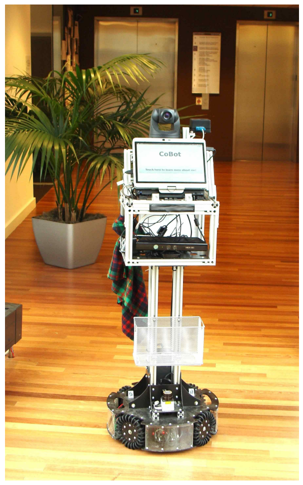


Fig. 1. The deployed CoBot-2 service robot.

In this paper, we contribute a complete system that addresses all of these sub-goals to perform tasks requested by the occupants of an office building. In addition to being effective in work environments, such a system has ready applications to assistive care in hospitals or nursing homes, where it could help overburdened nurses and caregivers to deliver items to bedridden patients.

We have developed two robots, CoBot-1 and CoBot-2. The robots, agile in their navigation due to their omnidirectional bases, purposefully include a modest variety of sensing and computing devices, including a controllable camera, a Microsoft Kinect depth-camera, a small Hokuyo LIDAR, and a touch-screen tablet. The CoBots autonomously localize and navigate in the building using depth-camera and LIDAR based localization and navigation algorithms.

Recently, we have effectively deployed CoBot-2 (Figure 1) to the occupants of our building for several hours each day. Occupants can schedule the CoBot robots to perform four different tasks: (i) go to a room, (ii) deliver a spoken message to a room, (iii) transport an object from one room to another, and (iv) escort a person from an elevator to a room. It has been rewarding to witness the robot continuously moving in our building among four floors without supervision. In this paper, we contribute the underlying technical algorithms, as well as the results of the robot's deployment.

The deployed CoBot-2 follows a *symbiotic* autonomy approach [7] given that it has limitations in its perception, cognition, and action. The robot proactively assesses that it needs help and asks humans to help resolve its limitations, particularly the physical ones. For example, the CoBot robots do not have arms, and therefore they ask for help for manipulating objects and pressing elevator buttons.

CoBot-2 is deployed to perform tasks requested by occupants of the building known as "task solicitors", and gets help from other humans around the robot known as "task helpers". Figure 2 illustrates the multiple components of our symbiotic CoBot robot. Task solicitors can request tasks on an online web server which are then processed by a *scheduler*, which determines their execution time. The *scheduler* sends these tasks to the *task planner and executor* that divides tasks into autonomous actions and symbiotic interactions to request help to ride the elevator and to manipulate objects.

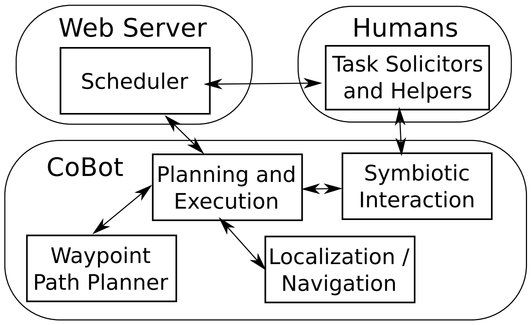


Fig. 2. Connections between web server, CoBot and humans.

In this paper, we present our approach to long-term task-centered robot deployment and extensive results from our own CoBot robots. In particular, we contribute:

1) A web based scheduler that accepts users requests and generates a conflict-free schedule for the robot (Section 2)
2) A task execution framework that allows the robot to overcome its limitations by asking for help from humans (Section 3)
3) A complete autonomous robot system capable of performing its scheduled tasks (Section 4)
4) A set of results from deployment of the robot to the occupants of the building (Section 5)

II. Scheduling User-Requested Tasks

Users request tasks over the web for CoBot-2 to perform. From CoBot-2's website, registered users can book a new task, view a list of their own scheduled tasks and completed tasks, cancel scheduled tasks, and see the current position of the robot.

When scheduling a task, users choose:

- *Task Type*. The task request. The options include 1) go to a room, 2) deliver a spoken message, 3) transport an object from one location to another, and 4) escort a visitor from the elevator to an office.
- *Task Parameters*. Task-specific options. This includes the destination location(s) for each task. Other options could include a spoken message to deliver or the name of an object to transport.
- *Time Constraints*. When the task should be executed. The user can specify "as soon as possible", a specific time, or a window of time.

Each request submitted by the user is sent to the *scheduler*, which may either accept or reject it. If the scheduler rejects a request, it proposes alternative times when the robot is available. The scheduler must fit the requested tasks into a *schedule*, a mapping of tasks to execution times. To plan the schedule, we represent each task request T_i as a tuple $T_i =< s, e, l^s, l^e, d >$. Task T_i must begin execution within the time interval $[s, e]$ where s and e are times in seconds (all time constraints are converted internally to windows of time). The task T_i is expected to take d seconds to complete from start to finish. The task duration d does not include the time taken to travel from the ending location of one task to the starting location of the next, this is estimated by a function $c(l_1, l_2)$ which estimates the time taken to travel between two locations. The task begins at the location $l^s \in L$ and ends at the location $l^e \in L$, where L is the set of all named locations CoBot-2 can travel to, including offices, kitchens and lounges.

A. Estimating Task Times

To effectively produce a schedule of tasks for the robot to execute, we must estimate both how long each task will take and how long it will take CoBot-2 to travel from one task to the next. This is the role of the Task Time Estimator. There are three components of the task duration d to consider:

- t_{nav} - *Navigation Time.* The time spent moving from one location to another. This is computed from the distance planned from CoBot-2's navigation graph and its velocity. It does not include use of the elevators.
- $t_{elevator}$ - *Elevator Transportation Time.* The time spent waiting for and using the elevator.
- t_{task} - *Task Specific Time.* The time specific to the task. For example, when delivering a spoken message, t_{task} includes the time taken to recite the message.

The expected task execution time is the sum

$$d = t_{nav} + t_{elevator} + t_{task} \,.$$

Similarly, the expected time, or cost, to travel between two locations includes the same components, except for t_{task}.

$$c(l_1, l_2) = t_{nav} + t_{elevator}$$

This cost is an estimate, and the actual time to execute a task may be either longer or shorter. CoBot-2 will adjust its schedule accordingly during execution.

B. Forming a Schedule

Based on the estimated execution and travel times, the scheduler fits all of the tasks into a schedule, a mapping of tasks to planned execution times. This is a variant of the Dynamic Vehicle Routing Problem with time windows, in which a fleet of vehicles must visit a set of locations, each within a certain time interval [8]. In the batch problem, the scheduler is given a list T of n task requests that the robot must fulfill by finding a set of starting times t_i so that no two tasks overlap and each task is fulfilled within the requested time window $[s_i, e_i]$. Each task has an estimated duration d_i which does not include travel time between tasks, since this depends on the tasks' ordering. The function c gives the estimated cost to travel between two locations, as computed by the waypoint path planner.

We solve for the variables t_i with a mixed integer program (MIP). Our first set of constraints states that each time t_i must fall within the start and end times of the window: $\forall i \; s_i \leq t_i \leq e_i$. Next, we add constraints to ensure that no two tasks overlap, handling both the case that task i comes before task j and the reverse order. We introduce helper indicator variables $pre_{i,j}$ which indicate whether task i occurs before task j.

$$\forall i, j \; t_i + d_i + c(l_i^e, l_j^s) - t_j \leq |e_i - s_j|(1 - pre_{i,j})$$
$$\forall i, j \; t_j + d_j + c(l_j^e, l_i^s) - t_i \leq |e_j - s_i| pre_{i,j}$$

We minimize the sum of the starting times $\sum_i t_i$ to ensure that user tasks are completed as soon as possible.

Solving an MIP is NP-hard; however, for smaller problem instances it can be done relatively quickly. We generated a thousand problem instances of fifteen tasks with two minutes to a half hour duration, with randomly generated time windows over the course of a four hour period. We expect CoBot to receive similar patterns of requests in the real world. The scheduler solved over 99% of the problems in under two seconds on a laptop computer. In the few cases where a solution is not found in two seconds, we can reject the user's request and ask them to modify their time window.

In practice, the task requests are not processed in a batch, but come in an online-fashion over the web. We reschedule everything whenever a new request is made. If a schedule cannot be found, the user's request is rejected and the user has an opportunity to relax their constraints.

After a schedule is formed, it is sent to the robot's task planner to execute. The task planner provides continuous feedback regarding the state of execution, and informs the scheduler when the task has been completed.

III. Planning and Executing Tasks

CoBot-2 is capable of autonomous localization and navigation, but cannot manipulate objects and has a limited ability to perceive the elevators. However, users can request tasks such as transporting objects, that require these capabilities. Task executor divides each task into autonomous actions and symbiotic interactions to seek help with actions it cannot perform autonomously.

The task executor first plans symbiotic interactions with its task solicitors at the start and end of the task to place and remove objects and confirm task completion. The task solicitors are expected to be willing to help because they requested the task to be performed and therefore want it to succeed. Then, the executor uses waypoint path planning to determine the lowest cost path to travel to rooms autonomously and identifies locations where the robot will need to ask for help to successfully navigate to its destinations. In particular, CoBot-2 needs help pressing the up/down buttons outside the elevator, determining which elevator to enter, and holding the elevator door open before it can navigate into the elevator. Once inside the elevator, it needs help pressing the destination floor button and may sometimes need help determining when it is on the correct destination floor before leaving the elevator. CoBot-2 depends on building occupants who are also taking the elevator to be task helpers. These task helpers are already performing the actions themselves and therefore have low cost to help the robot as well.

A. Symbiotic Interaction with Task Solicitors

While other robots have asked for help from passers-by in the environment [9], [10], CoBot-2 must interact with task solicitors who requested the task and/or occupy the destination locations in order to manipulate objects and confirm task completion. We divide these interactions into those that happen at the *start* and at the *end* of a task.

Transport and Escort tasks both require *start* interactions. In order to transport an object or escort a person, CoBot-2 asks to acquire the object or find the person at the pickup location, saying "Hello, I'm here to take [object/person] to [room]. Press 'Done' when you are ready to go" and displays a button on the user interface. This interaction gives occupants time to find the object and arrange it on the robot or visitors time to finish conversations prior to being escorted. Additionally, the confirmation is a signal to begin navigating to the destination without having to actively sense each potential object or person.

At the *end* of each task, CoBot-2 also interacts with task solicitors. At the end of deliver message tasks, the robot asks "Hello. I have a message from [task solicitor]. Are you ready to hear it?". Then, after CoBot-2 speaks the message, it asks "Would you like me to repeat myself, or can I leave?" and displays a "Repeat" and a "Done" button. For all other tasks, CoBot-2 only confirms it has completed its task, saying "Please press 'Done when I can leave."

B. Waypoint Path Planning for Navigation

CoBot-2 also plans the autonomous and symbiotic interactions to navigate between rooms. Since CoBot-2 navigates in a multi-floor building with many occupants, it takes into account their preferences for traveling by their offices and the need to use the elevator. For each destination, the task executor calls the waypoint path planner to generate a low-cost path. The waypoint path planner keeps a room graph

$$\begin{aligned} &G_R = \langle V_R, E_R \rangle, \\ &V_R = \{v_i = (x_i, y_i)\}_{i=1:|V_R|}, \\ &E_R = \{e_i = (v_{i1}, v_{i2}) : v_{i1}, v_{i2} \in V_R\}_{i=1:|E_R|}. \end{aligned}$$

The vertices $V_i \in V_R$ indicate the locations of offices and other important landmarks such as elevators and kitchens. The edges $e_i = (v_{i1}, v_{i2}) \in E$ indicate that the vertices v_{i1} and v_{i2} are connected by a navigable path. The cost function $c(e_i)$ of the edge is based on the length of the edge, the time to travel the edge, the capability of the robot to navigate there autonomously, and the cost to humans of traversing the edge. For example, the robot may lower the cost of an edge in order to favor particular edges if the building occupants in offices near that stretch of hallway enjoy watching CoBot-

2 go by. The waypoint path planner uses Dijkstra's shortest path algorithm with edge weights $c(e_i)$ and returns a path of waypoint locations

$$\vec{l}_d = \langle l_d^0 = v_{start}, l_d^1, l_d^2, \ldots, l_d^k = v_{end} \rangle$$

for the robot to navigate to get from v_{start} to v_{end}. Note that the path isn't necessarily the shortest distance path, but is rather the lowest cost path.

Given a set of waypoints, the path planner then classifies them into one of two categories: those to *direct* the robot towards more preferable paths and *stops* for help. If CoBot-2 determines that it is not capable of traversing an edge between two consecutive waypoint locations autonomously, it labels the first of the two waypoints in the path as a *stop*. There are three elevator stop locations:

- The location outside the elevator on the starting floor. CoBot-2 needs help pressing the up/down buttons and determining which elevator to enter.
- The location inside the elevator on the starting floor. CoBot-2 needs help pressing the floor number buttons inside the elevator.
- The location inside the elevator on the destination floor. CoBot-2 needs help identifying when it is on the destination floor and can exit the elevator.

All other waypoints in the path are *direct* to indicate that the robot can navigate between them autonomously.

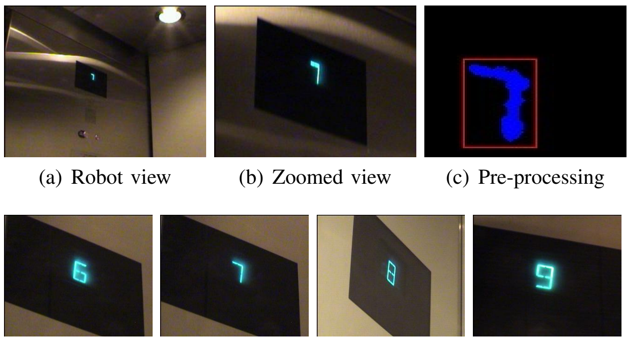
C. Symbiotic Interaction with Task Helpers: Riding the Elevator

One of the most important contributions of the current successful deployment of CoBot-2 is the fact that the robot takes the elevator by itself, as our building has multiple floors. CoBot-2, in its symbiotic-autonomy depends on task helpers who are already using the elevator to help it too.

We view four robot states in riding the elevator, namely *waiting*, *entering*, *inside*, and *exiting* the elevator. Each of these states involves interaction with a human helper and corresponds to one of the three stops in the waypoint path planner's path. In the *waiting* state, the robot is outside of the elevator and asks to "Please push the up/down button" and to identify the arriving elevator as "Which elevator is going up/down" and displays the choices of two possible elevators (see elevators in Figure 1 behind the robot) on a touch screen. Given the human input, before the robot starts moving to position itself in front of the correct elevator, it requests "Please hold the elevator door." and then *enters* the elevator autonomously.

When *inside* the elevator, the robot can asks for help about its destination floor "Can you please push [the destination floor] button and tell me when we get to that floor." At the destination floor, the task executor expects a response to its request and at that time will wait for the doors to open and autonomously navigate out. If a person tells the robot incorrectly that it is on the destination floor, as soon as CoBot-2 exits the elevator, it can localize itself and autonomously detect that it is on the wrong floor. At that time, the task executor replans its path to the destination, asking for help to enter the elevator again.

We have also developed an approach to allow the robot to autonomously read the floor numbers displayed inside the elevator to identify when it is on the destination floor, rather than wait for a response to the request on the correct floor. The floor numbers are in a dedicated LED panel, which the robot can a) *detect* and b) *classify* using its vision camera. The detection algorithm uses the localization information of the robot inside the elevator, controls the camera pan to search for the known location of the LED number display in each elevator, tilts the camera as needed in its search, and zooms on the number. When the robot finds the LED number, it holds its camera to its position, and invokes the classifier. We trained an SVM, using a linear kernel and features given by histograms of gradient on the V channel of the HSV images. We trained one SVM per floor number using a one versus all approach. While the elevator is moving, CoBot-2 sees a small number of images (4-5) of each floor number. By using majority consensus over the last three images, CoBot-2 is able to identify the floor number with high accuracy without running the risk of missing floor transitions. Figure 3 shows examples of different elevator numbers and views obtained from the robot perspective. The numbers are zoomed, pre-processed, and classified.


(a) Robot view (b) Zoomed view (c) Pre-processing

(d) Six (e) Seven (f) Eight (g) Nine

Fig. 3. Elevator floor numbers as processed by CoBot-2.

The detection and classification algorithm falls back to the default behavior of asking a human for help if it reaches preset autonomous number reading thresholds. We tested this autonomous elevator number detection, and it is now being incorporated into the deployed CoBot-2.

IV. Localization and Navigation

Although CoBot-2 is symbiotic and relies on humans for help, it localizes and navigates fully autonomously.

A. Localization

CoBot-2 localizes with the Corrective Gradient Refinement (CGR) localization algorithm [11], using the laser range-finder and the plane filtered point cloud [12] from the Kinect sensor, along with wheel odometry. CGR localizes using a vector map M of the building extracted from the architectural plans. This map consists of a set of lines $l_i \in M$ representing the building walls.

CGR localization uses an observation model where points observed by the laser rangefinder and the Kinect sensor are associated with lines on the map, and the probability of the observation is computed as the joint probability of the observed points arising from the associated lines on the vector map. Let the set of 2D points observed by the laser rangefinder be denoted by $P = \{p_i\}_{i=1:n}$ and the pose of the robot by $x = \{x_l, x_\theta\}$ where x_l is the 2D location of the robot and x_θ its orientation angle. For every point $p_i \in P$, line $l_i \in M$ is found by ray casting such that the ray in the direction of $p_i - x_l$ and originating from x_l intersects l_i before any other line. The perpendicular distance d_i of p_i from the (extended) line l_i is computed. The total (non-normalized) observation likelihood $p(P|x)$ is then given by $p(P|x) = \prod_{i=1}^{n} \exp\left[-\frac{d_i^2}{2f\sigma^2}\right]$. Here, σ is the standard deviation of the distance measurements of a single ray, and f (where $f > 1$) is a discounting factor to discount for the correlation between rays. This observation model is also used to analytically compute the state space gradients of the observation model, which are used for the "refinement" step in CGR.

CGR localization using the Kinect depth camera [12] is based on the assumption that only large planar features observed in the Kinect depth image correspond to lines (walls) in the vector map. Using Fast Sampling Plane Filtering (FSPF) [12], the depth image is used to generate a "plane filtered point cloud" $P = \{p_i, r_i\}_{i=i:n}$ consisting of n points p_i and normals r_i corresponding to planes observed by the robot. Sampled points that did not fit the detected planes are added to the "outlier point cloud". The plane filtered points and their corresponding plane normal estimates are then projected into 2D to yield the set of points $P' = \{p'_i, r'_i\}_{i=1:n'}$ where p'_i are the projected 2D points, and r'_i the corresponding 2D normals. The projected points in P' are then used to compute the observation likelihood using the same observation model as is used for the laser rangefinder. The plane filtered points, as well as the outlier 3D points, are used to compute obstacle avoidance margins for the robot. CGR localization (using the laser rangefinder and Kinect sensors) thus provides the estimated pose (x, y, θ) of CoBot-2 on a particular floor of the building.

To detect when CoBot-2 has entered a new floor (using the elevator), we use the StarGazer sensor. The StarGazer sensor is an off-the-shelf sensor which has a ceiling-facing infrared camera and infrared LEDs. Retroreflective markers, consisting of patterns of dots, reflect the light from the infrared LEDs and are read by the infrared camera. The StarGazer sensor provides the unique identification code of the marker and the robot's pose relative to the marker. Thus, by placing unique StarGazer markers outside the elevator doors on every floor, CoBot-2 can immediately identify which floor it is on after exiting the elevator.

B. Navigation

CoBot-2 uses a graph based navigation planner [13] to plan paths between locations on the same floor of the building. The navigation graph G, distinct from G_R presented previously, is denoted by $G = \langle V, E \rangle$, $V = \{v_i = (x_i, y_i)\}_{i=:n_V}$, $E = \{e_i = (v_{i1}, v_{i2}) : v_{i1}, v_{i2} \in V\}_{i=:n_E}$. The set of vertices V consists of n_V vertices $v_i = (x_i, y_i)$ that represent the location of the ends and intersections of hallways. The set of edges E consists of n_E edges $e_i = (v_{i1}, v_{i2})$ that represent navigable paths between vertices v_{i1} and v_{i2}.

Given a destination location $l_d = (x_d, y_d, \theta_d)$, the navigation planner first finds the projected destination location $l'_d = (x'_d, y'_d, \theta_d)$ that lies on one of the edges in the graph. This projected destination location is then used to compute a topological policy using Dijkstra's algorithm for the entire graph. The navigation planner projects the current location $l = (x, y, \theta)$ onto the graph and then executes the topological policy until the robot reaches the edge on which l'_d lies, and then drives straight to the location l_d. Thus, the navigation planner navigates between start and end rooms $\in G_R$ given by the task executor, irrespective of whether or not they actually lie on the graph.

While executing the navigation plan, CoBot-2 performs obstacle avoidance based on the obstacles detected by the laser rangefinder and Kinect sensors. This is done by computing open path lengths available to the robot for different angular directions. Obstacle checks are performed using the 2D points detected by the laser rangefinder, and the down-projected points in the plane filtered and outlier point clouds generated by FSPF from the latest Kinect depth image.

V. Deployment Results

We deployed CoBot-2 on the upper four floors of an office building for a two week period. CoBot-2 was deployed for two hours every weekday and made available to the building occupants. Occupants were alerted of CoBot-2's availability through email and physical signs posted on bulletin boards and on the robot itself. The deployment times varied each day, and were announced beforehand on CoBot-2's website.

The response to CoBot-2's deployment was positive: over one hundred building occupants registered to use CoBot-2 on the website. Users found creative ways to exploit the robot's capabilities, including, but not limited to:

- Sending messages to friends.
- Reminding occupants of meetings.
- Escorting visitors between offices.
- Delivering printouts, inter-office mail, USB sticks, snacks, owed money, and beverages to other building occupants.

TABLE I
Total number of task requests per task type and the respective number that used the elevator.

Task Type	Total Requests	# Multi-floor
Escort	3	2
GoToRoom	52	22
DeliverMessage	56	20
Transport	29	22

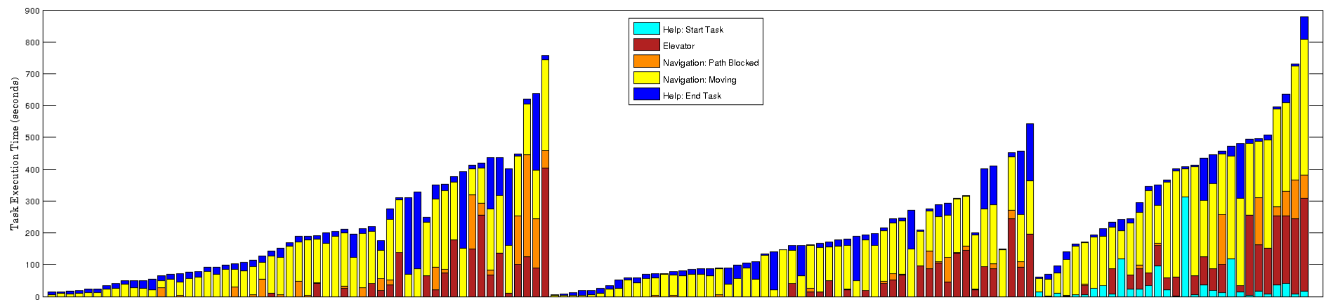


Fig. 4. Execution times for, from left to right, Deliver Message tasks, Go to Room tasks, and Transport tasks. The breakdown includes 1) waiting for help to start the task, 2) riding the elevator, 3) navigating (not including time blocked by obstacles), 4) waiting blocked by an obstacle, and 5) waiting for help to end the task.

Particularly in the first couple days of deployment, we found building occupants following the robot around to see where it was going and how it worked.

We found that occupants scheduled the robot to transport objects between multiple floors of the building more often than they used the multi-floor functionality for other tasks (see Table I). In particular, the transport task saved the task solicitors time because they did not have to travel between floors themselves. However, even the other scheduled tasks utilized the elevator 40% of the time.

In fulfillment of the user requested tasks, CoBot-2 travelled a total of 8.7 km, which covered most of the building. CoBot-2 spent

- 6 hours and 17 minutes navigating this distance,
- 36 minutes with a blocked path waiting for a person to move out of its way,
- 1 hour and 2 minutes waiting for help with the elevator,
- 1 hour and 18 minutes waiting for task solicitor help to complete its tasks.

Figure 4 shows how much time CoBot-2 took to execute each task, and how that time was apportioned. A total of 140 tasks were completed during the two week deployment, which took 9 hours and 13 minutes. Based on these times, we find that task solicitors quickly responded to the robot's request for help at the start and end of tasks. Building occupants (even those that had never scheduled a task) were willing and able to help the robot in and out of the elevator. This finding supports our model of symbiotic autonomy— humans are willing to help a robot complete its tasks so that the robot is available and capable of performing tasks for them as well.

Although CoBot-2 could be required to wait for human help indefinitely, the task execution times are limited. In general, little more than five minutes per task was spent in the elevator. This is because if CoBot-2 spends more than five minutes waiting for human help, it sends an email to our research group asking for assistance. Typically, however, occupants helped CoBot-2 and there was no need to do so. Furthermore, if at the end of a task no human pressed the button to indicate that the task was complete, CoBot-2 marked the task as complete and moved on to the next task.

VI. Conclusion

We have successfully deployed an autonomous robot to service users in a multi-floor building. We have presented how users request tasks dynamically on a web server, and how requests are scheduled via a mixed integer program. Our task planner and executor divides each task into autonomous navigation actions, and interactions with humans which require help. Our robots are symbiotic, helping occupants of the building by fulfilling their requests, while also receiving help from humans to complete tasks and to use the elevators. The robots are also fully autonomous; they navigate and localize on their own. CoBot-2 has traveled over 100km throughout its lifetime, and we present results of 8.7km for public deployment.

References

[1] N. J. Nilsson, "Shakey the robot," SRI International, Tech. Rep. 323, 1984.

[2] R. Simmons, R. Goodwin, K. Z. Haigh, S. Koenig, and J. O'Sullivan, "A layered architecture for office delivery robots," in *Proceedings of the first international conference on Autonomous agents (AGENTS'97)*, 1997, pp. 245–252.

[3] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "Experiences with an interactive museum tour-guide robot," *Artificial Intelligence*, vol. 114, pp. 3–55, October 1999.

[4] T. Fong, I. Nourbakhsh, and K. Dautenhahn, "A survey of socially interactive robots," *Robotics and Autonomous Systems*, vol. 42, pp. 143–166, 2003.

[5] S. Thrun *et al.*, "Probabilistic algorithms and the interactive museum tour-guide robot minerva," *The International Journal of Robotics Research*, vol. 19, no. 11, pp. 972–999, 2000.

[6] U. Visser and H.-D. Burkhard, "RoboCup: 10 years of achievements and future challenges," *AI Magazine*, vol. 28, no. 2, pp. 115–132, Summer 2007.

[7] S. Rosenthal, J. Biswas, and M. Veloso, "An effective personal mobile robot agent through a symbiotic human-robot interaction," in *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2010, pp. 915–922.

[8] A. Larsen and O. Madsen, "The dynamic vehicle routing problem," Ph.D. dissertation, Technical University of Denmark, Department of Informatics and Mathematical Modeling, 2000.

[9] H. Hüttenrauch and S. Eklundh, "To help or not to help a service robot: Bystander intervention as a resource in human-robot collaboration," *Interaction Studies*, vol. 7, no. 3, pp. 455–477, 2006.

[10] A. Weiss, J. Igelsböck, M. Tscheligi, A. Bauer, K. Kühnlenz, D. Wollherr, and M. Buss, "Robots asking for directions: the willingness of passers-by to support robots," in *HRI '10*, 2010, pp. 23–30.

[11] J. Biswas, B. Coltin, and M. Veloso, "Corrective gradient refinement for mobile robot localization," in *Intelligent Robots and Systems (IROS), 2011 IEEE International Conference on*. IEEE, 2011.

[12] J. Biswas and M. Veloso, "Depth camera based indoor mobile robot localization and navigation," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2012.

[13] B. Coltin, J. Biswas, D. Pomerleau, and M. Veloso, "Effective semi-autonomous telepresence," *Proceedings of the RoboCup Symposium*, pp. 289–300, July 2011.