RC25123 (W1103-066) March 10, 2011
Computer Science

IBM Research Report

Towards Data Center Self-Diagnosis Using a Mobile Robot

Jon Lenchner, Canturk Isci, Jeffrey O. Kephart,
Christopher Mansley*, Jonathan Connell, Suzanne McIntosh
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

*Department of Computer Science
Rutgers University
Piscataway, NJ 08854

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Towards Data Center Self-Diagnosis Using a Mobile Robot

Jon Lenchner
IBM Thomas J Watson Research Center
Hawthorne, NY 10532
lenchner@us.ibm.com

Canturk Isci
IBM Thomas J Watson Research Center
Hawthorne, NY 10532
canturk@us.ibm.com

Jeffrey O. Kephart
IBM Thomas J Watson Research Center
Hawthorne, NY 10532
kephart@us.ibm.com

Christopher Mansley
Dept. of Computer Science
Rutgers University,
Piscataway, NJ 08854
cmansley@cs.rutgers.edu

Jonathan Connell
IBM Thomas J Watson Research Center
Hawthorne, NY 10532
jconnell@us.ibm.com

Suzanne McIntosh
IBM Thomas J Watson Research Center
Hawthorne, NY 10532
skranjac@us.ibm.com

ABSTRACT

We describe an inexpensive robot that serves as a physical autonomic element, autonomously navigating, mapping and monitoring data centers—even ones that it has never seen before. Through a series of real experiments and simulations, we establish that the robot is sufficiently accurate, efficient and robust to be of practical benefit in real data center environments. We demonstrate how the robot's integration with Maximo for Energy Optimization, a commercial data center energy management product, supports autonomic behaviors at the level of the data center as a whole, particularly self-diagnosis of emerging thermal problems.

1. INTRODUCTION

Over time, data centers around the world are consuming ever more energy, with those in the US now responsible for an estimated 2% of the nation's electricity budget [1, 2]. Recognizing that cooling is a significant contributor to energy consumption, data center operators are beginning to tolerate higher operating temperatures. While this practice saves substantial amounts of energy, running closer to allowable operating temperature limits increases the risk that temperature problems will result in equipment failures that wipe out the financial benefits of saving energy. Vigilance is needed, and increasingly that vigilance is being provided by data center energy management software that monitors data center temperatures and alerts operators when troublesome hot spots develop.

Previous authors [11] have painted a picture of the data center as an autonomic computing system populated with autonomic elements representing physical components such as power distribution units and air conditioners that interact with one another and with software autonomic elements such as workload managers. In this paper, we describe a truly physical autonomic element—a robot that navigates the data center, mapping its layout and monitoring its temperature and other quantities of interest with little to no human assistance. The robot is integrated into state-of-the art enterprise energy management software in such a way that, as thermal anomalies arise in the data center (such as the occurrence of areas of under and over-provisioned cooling) the robot can automatically be dispatched to investigate these regions to provide evidence regarding suspected causes of the anomalies and facilitate an automatic response.

After reviewing related work in Section 2, we describe the physical and software design principles and architecture in Section 3. In Sections 4 and 5, we describe several basic and enhanced algorithms that support vision, navigation, asset classification, and self-protection, intermingling the algorithms with evidence of their effectiveness from simulation and physical experiments that we have conducted in four different data centers. Section 6 describes the integration of the robot with a commercial data center energy management product, and presents practical scenarios that illustrate how the robot can improve the self-diagnostic capabilities of the data center.

2. RELATED WORK

Intelligent monitoring and automated, adaptive energy and thermal management of data centers has been a highly active research area in recent years. This comes as no surprise in large scale computing as energy and cooling costs loom, the environmental concerns increase, and the emphasis on energy regulation for computing systems and facilities grows stronger [2, 3]. Patel, Sharma et al. present some of the early work in data center monitoring and management [4, 5, 6] identifiying some key innefficiencies in cooling and CRAC configurations. Several studies explore techniques for efficient energy and thermal provisioning in data centers [7, 8, 9, 10]. Some studies employ coordinated schemes based on control or utility models to collectively manage multiple control knobs [11, 12, 13]. Some recent research also incorporates dynamic workload placement as an additional *soft* control knob for data center management [14, 15, 16]. While there is clearly no shortage of work in data center management, most of the prior approaches assume a *perfect* world, where all the required vast level of monitoring is readily available and all the distributed sensing and control knobs are seamlessly integrated. However, the state of reality is nowhere near such level of integration for controllability and especially *observability*. Thus, our mobile monitoring and management approach bridges a major critical gap in data center monitoring and management by first, providing a simple, tractable and dense mon-

ICAC 2011 Karlsruhe, Germany
.

itoring approach and second, demonstrating a true seamless integration of monitoring to data center control.

Some prior studies also consider bridging the gap between monitoring and management by providing distributed integrated monitoring or using controlled mobile sampling. In comparison to these, our technique presents an approach with minimal intrusion to data center operations, and more importantly, brings in *autonomic monitoring* into data centers with a practically *invisible barrier for adoption.* In prior work, Tschudi et al. [17] recommend integrated monitoring schemes for data centers, while this is inarguably of great value, such high-end integrated monitoring has a high cost of entry, and requires periodic maintenence, verification and calibration. Our work reduces this barrier to almost nothing by eliminating any requisite in the existing data center design. Bash et al.[18], also consider efficient data center monitoring by selective sensor sampling and determining the best set of fixed sensor locations, while our technique is based on mobile sensing, which can complement static instrumentation. Hamann et al. present a related mobile measurement technology (MMT) with a mobile sensor cart, which can optionally be compounded with static sensing, for spatially-dense data center profiling [19, 20].Our work also has the same motivation of providing dense monitoring. However, in addition to this, we bring in autonomicity in data center operations, which opens up a whole range of new possibilities for on-demand, control driven, and selective autonomic monitoring and management.

Other researchers have explored robotic monitoring approaches for different environments. Patel et al. [5, 21] argue for robotic monitoring and management for data centers, but do not describe a design or provide any implementation details. Pon et al. [22] describe a cable-based robot for monitoring environmental data in rivers and forests. Mansley et al. [24] give a first example of an autonomous robot for data center mapping and monitoring, which forms the basis of our work. While these studies demonstrate the practical application examples of robotic monitoring and, to some extent, their applicatoin to data center monitoring, in this work we provide quantitative evaluations for effective and accurate mobile monitoring and describe novel techniques that target data center centric optimizations for reliability, accuracy, performance and particularly autonomic management and diagnosis.

3. ROBOT DESIGN

Our robot's most fundamental purpose is to eliminate the human labor required to map and monitor both new and known data centers. This translates into the following set of basic design objectives, all of which must be attained completely autonomously:

- **Full Coverage**. The robot must visit every region of the data center that is contiguous from an arbitrary starting position, with or without prior knowledge of the data center layout.
- **Sensor Map Generation**. The robot must generate a reasonably accurate map of sensor readings. Specifically, it must take sensor readings at sufficiently fine spatial granularity and associate with each reading an accurate spatial location.
- **Layout Generation**. The robot must generate a simple data center layout. At a minimum, the layout should distinguish a few basic types of structures or assets, so that a human operator can recognize the generated layout as reflecting the physical layout of the data center.
- **Robustness**. The robot must carry out its functions under ordinary operating conditions, working around unexpected obstacles, coping with network outages, and protecting itself from harm.
- **Low cost**. The robot should be constructible from inexpensive components.

Figure 1: The data center management robot.

We have been able to meet all of these basic design objectives, as well as satisfy some additional desirable criteria, through judicious physical design, software architecture, and algorithms. The last design objective—low cost—is addressed primarily by the physical design, which is detailed in subsection 3A, while our first four design objectives are achieved primarily through software architecture (described in Section 3B) and algorithms (detailed in Section 4).

3.1 Physical design

The robot, shown in its natural habitat in Figure 1, is based on the *iRobot Create* robotic research platform. The on-board capabilities of the Create include basic locomotion, bump sensors and odometry, and modest compute power to control the robot's motion. We supplemented these capabilities by adding several components:

- An off-the-shelf netbook computer with 1.6 GHz Atom processor and 1 GB of memory, to provide sufficient compute power and programming flexibility;
- A $6\frac{1}{2}$ foot tall aluminum pole mounted vertically to the robot's base, to enable other components to be mounted well above the floor;
- A Logitech C905 USB webcam (which provides 320x240 color images) mounted on the pole approximately 30 inches above the floor and angled down so as to view precisely one floor tile ahead of the present position;
- Seven 40-gauge K-type thermocouples (with 3-second response time to temperature changes) attached to the pole, and spaced at 1 foot intervals from 4 inches to 76 inches above floor height to provide temperature readings at different heights ranging up to the top of typical data center racks;

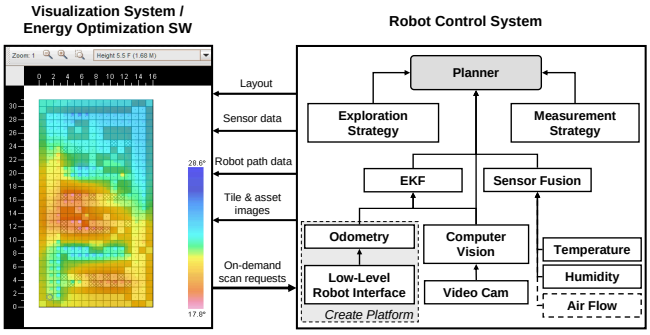


Figure 2: High-level robot architecture showing integration with visualization and energy management software.

- A single capacitive-style relative humidity sensor mounted on the pole approximately 20 inches above the floor; and
- A custom sensor collection interface that collects data (i) from the single humidity sensor by measuring incremental changes in the dielectric constant of the capacitive substrate, and (ii) from the thermocouples by measuring the thermoelectric voltage differentials at the bimetallic sensor junctions in sequence, and converts these measurements to humidity and temperature readings via analog-to-digital conversion, and transmits them back to the netbook computer via RS232.

3.2 Software architecture

Figure 2 provides a high-level overview of the software architecture of the robotic system. The combination of vision, sensing, and navigation components that make up the robot control system is depicted on the right; the visualization and energy management software with which the robot control system is integrated is depicted on the left.

To navigate, the robot takes advantage of the square grid formed by industry standard 2′ x 2′ data center floor tiles[1] to move one floor tile at a time, continuously feeling its way around the data center and keeping track of where it has already been, eventually visiting all unobstructed tiles. Whenever it visits a tile, the robot stops at its center, takes a set of sensor readings at various heights, and analyzes and classifies each tile image. Then, as described it greater detail below, it transmits relevant data over a wireless link to the visualization system.

To assist in the robot's navigation, still images from the onboard webcam are fed to the on-board netbook computer on which the robot control system resides. An image of the tile at which the webcam is currently aimed is analyzed by the Computer Vision component to classify the tile and to estimate the robot's pose (its position and orientation) relative to the center of the current tile. A second estimate of the robot's pose is provided by the Create's on-board odometry system, which uses dead reckoning based upon the wheel rotation and the previously estimated position of the robot. However, odometry alone is known not to be sufficiently accurate for robot navigation [25]. The pose estimates from the Computer Vision and Odometry components are input to an Extended Kalmann Filter (EKF) [26] to generate a more precise pose estimate, which is in turn passed on to the planner, which is responsible for overall navigation.

When the Planner receives a signal from the low-level robot interface indicating that the robot has stopped in a particular orientation at the center of a tile, several data are sent via the netbook's wireless interface to the visualization and energy optimization software, including a still image of the tile ahead, its classification as determined by the Computer Vision component, and an update to the robot's location and path. Additionally, the Planner consults the Measurement Strategy component to determine whether sensor readings are required for the current position. Sensor readings may not be needed if a tile has been visited recently, or if the robot is conducting a quick selective scan of a data center for which the layout is already known. If readings are desired, the Planner triggers readings from the onboard temperature and humidity sensors (and any other types of sensors that may be mounted on the aluminium pole) via a serial port request to the dedicated thermocouple and humidity sensor interface. Data from the various sensors are received by a Sensor Fusion component, which is designed to make the addition of new sensor types as seemless as possible, and sent to the planner, which associates the readings with their known physical locations and sends this tuple data to the visualization and energy optimization software.

Finally, the Planner consults the Exploration Strategy component, which contains logic that governs whether the robot moves forward by one tile or turns 90 degrees (either for viewing or to change the direction of motion). An important factor in exploration decisions is the Computer Vision component's classification of the neighboring tiles with regard to whether the tiles are believed to be visitable or not. The planner maintains a data structure indicating the known layout, what has been visited, and which tiles are believed to be visitable or unvisitable. Using this information, the Exploration Strategy component makes a decision about where to go next, which the planner in turn communicates with the Low-Level Robot Interface, directing it to turn 90 degrees or move forward one tile, as appropriate.

The visualization and energy optimization software with which the robot is integrated is a commercially available asset management product designed specifically for data center energy management, Maximo for Energy Optimization 7.1.1 (MEO), to which we have added a few enhancements. The types of data that the robot sends to MEO include:

- the layout and sensor data as they are acquired;
- the path taken by the robot, from the starting tile up to the present position; and
- a stream of still tile images.

MEO can govern the robot's behavior by sending to its control system on-demand scan requests. The two-way interaction between the robot control system and MEO supports a variety of practical scenarios, including manual or automated dispatch of the robot to a given area to take additional images of a region in which temperatures are elevated, or to perform a fine-grained scan of a particular area of the data center.

4. BASIC VISION AND NAVIGATION ALGORITHMS

In this section we discuss the basic vision and navigation algorithms used by the Robot Control System, and quantify their efficacy through a set of experiments.

4.1 Basic vision algorithms

The purpose of the computer vision system is two-fold. First, in conjunction with input from the robot platform's

[1]or the approximately equivalent 600mm x 600mm standard used in many countries outside of the US.

odometry it helps determine the robot's location and orientation relative to the grid established by the floor tiles. Second, it classifies the tile the robot and its camera are facing. In the current implementation, tiles are classified along two dimensions: visitability and perforation. Specifically, tiles are either *visitable* or *blocked*, and they are either *perforated* or *plain*. Perforated tiles are essential in most air-cooled data centers because they deliver cool air from below the floor to the inlets of IT devices.

Here we provide an overview the vision algorithm; a more detailed description can be found in [24]. Upon receiving a 320x240 color image of the next tile ahead from the webcam, the Computer Vision component processes it in the following sequence of four steps:

- **Grid Line Finding**. Grid lines are identified from the original tile image by a combination of rectangular filtering, Sobel edge finding, and connected component analysis.
- **Robot Pose Determination**. The angles of the detected lines are histogrammed, and the boundaries of the next tile are inferred from peaks in the histogram. The robot's orientation with respect to the grid and its relative offset from the current tile center are then computed.
- **Visitability Checking**. At this point the image of the observed tile is perspective-corrected into a square. Edge detection is again used to identify boundaries. A clearly identified tile boundary indicates a *visitable* tile, while the lack of a clear boundary, such as a missing edge results in a *blocked* classification.
- **Tile Type Detection.** Perforated tiles exhibit light and dark intensity variations that tend to have certain symmetry properties. If the pattern of intensity variations in the transformed tile image is sufficiently symmetric the tile is classified as *perforated*. Otherwise, the tile is classified as *plain*.

The resulting vision system has proven very robust with respect to contrast, variable illumination, and partial occlusion of the grid.

4.2 Basic navigation algorithms

One of our fundamental objectives is to provide a *fully-autonomic* robotic system with a practically *invisible barrier for adoption*. The robot must navigate and produce a data center layout with no prior information.

In our consideration of various navigation algorithms we break the algorithms into two tightly coupled processes: (i) path planning, i.e., deciding on which new data center location to go to; and (ii) exploration, i.e., deciding on how we "uncover" the unknown parts of the data center. The main goal of path planning is to reach every visitable tile as efficiently possible, and the goal of exploration is to find a favorable trade-off between the cost of exploring (turning to look at) unknown tiles vs. exploiting the already uncovered, but unvisited locations.

For path planning we investigated two well-known strategies, and a variant suitable for data center layouts: (i) simple *Depth First Search* (DFS), (ii) a variant of DFS which we have dubbed "DFS-Go-Long," and (iii) Frontier-based A*. DFS goes along a path as long as there are new visitable tiles, and then tracks back to the first visitable or unknown new tile. DFS-Go-Long simply tries to leverage the aisle structure of data centers and aims to minimize turn costs by introducing a directionality preference for going straight forward. Frontier-based A* maintains a frontier of unknown and visitable, but not yet visited tiles, and takes the path to the nearest unknown or visitable frontier at each step. Our evaluations with these approaches show that DFS-based approaches can lead to exploration costs of up to 80% in excess of the number of visitable tiles, while the frontier–based approach typically achieves exploration costs which are in the vicinity of 20% in excess of the number of visitable tiles. While DFS-Go-Long achieves minimum turn costs, the overall navigation cost is still substantially higher than frontier. As a result of these observations we base all subsequent experiments on frontier-based A* navigation.

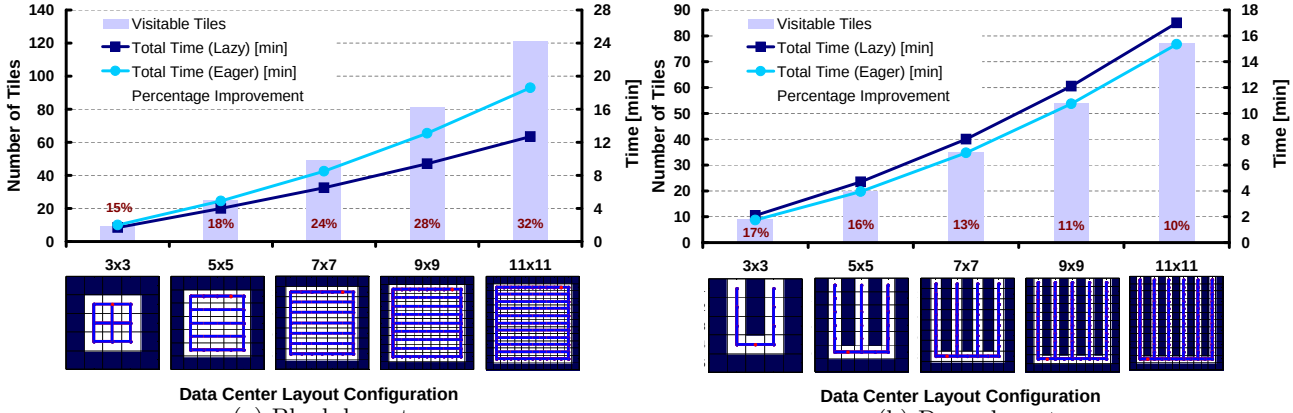
For exploration, we evaluate two approaches that differ in the level of aggressiveness in which they "check" neighboring unknown tiles. The first technique, *Eager Checking*, turns to check all immediate neighbors of the robot at each forward step and decides on the next path planning step with the full frontier information. This leads to a slow and steady exploration pattern that avoids extensive backtracking to pick up omitted neighbors, but entails a large number of turn costs. The second technique, *Lazy Checking*, does not check its neighbors unless it sees no immediate visitable tiles to go to. The exploration pattern in this case is to start quickly, but backtrack in a less informed way to cover omitted neighbors as the exploration progresses.

To study the effectiveness of these exploration strategies and how they depend upon the spatial arrangement of the data center, we developed a navigation simulator. The simulator is provided with accurate estimates of the time required for the robot to turn, move, bump and track back. Since it runs several orders of magnitude faster than the robot, the simulator is ideal for studying alternative navigation algorithms, which are then downloaded to the Exploration Strategy module of the physical robot.

To assess the impact of lazy versus eager checking on overall navigation efficiency we experimented with two simple artificial data centers: (i) *Open*, consisting of a blank data center with no obstacles; and (ii) *Dead-end*, consisting of dense obstacles with parallel aisles of single-width visitable tiles, creating a data center with long, narrow dead-end alleys.

We measured the scan time for each exploration strategy for a series of progressively larger data center layouts of the above types. The results, depicted in Figure 3, reveal that eager checking performs better in dead-end data centers, while lazy checking is superior in open data centers whereing it reduces the number of turns. In both Figures 3a and 3b, the bars represent the number of visitable tiles in each configuration (left vertical axis) while the curves represent the traversal time for each of the two algorithms (right vertical axis). The time advantage of the superior algorithm is provided at the bottom of the bars, and the data center layout configuration is shown underneath the main graph. For the open data center, lazy checking is uniformly faster than eager checking by a percentage that grows with the size of the data center (32% for the 11x11 configuration). In contrast, for the dead-end data center, eager checking is uniformly faster than lazy checking, but the percentage advantage diminishes as a function of the data center size.

Last, Figure4 shows the results for exploration speed (total scan time divided by number of visitable tiles) from actual data center evaluations with the two exploration mechanisms. For each data center the results reflect the averages for 4 different starting points and the error bars represent +1 standard deviation across runs, which is quite insignificant. The actual data center results show that lazy check-



(a) Blank layout.

(b) Dense layout.

Figure 3: Exploration performance with eager and lazy checking for different layout characteristics.

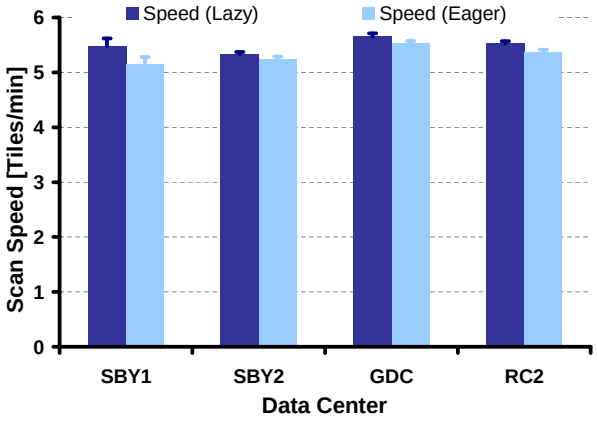


Figure 4: Average exploration speeds with Eager and Lazy checking for different data centers.

ing performs slightly better for actual data center layouts. However, the improvement is marginal, averaging around 3% across all the evaluated data centers.

5. ENHANCEMENTS

In this section, we describe several improvements that we have made to the robot after observing its behavior in several widely different data centers ranging in area from 1000 square feet to 77,000 square feet. These improvements entail algorithmic refinements to the vision, navigation and sensing algorithms that improve the robot's efficiency, coverage, and classification accuracy, as well as its ability to protect itself from hazards it may encounter in the data center, thereby reducing its reliance on human intervention.

5.1 Improved Accuracy and Coverage

Experience with the robot in a number of different data centers has shown the tile classification algorithms to be 97-98% accurate. Misclassification, however, has the following undesirable consequences:

- All misclassifications introduce errors into the automatically derived layout. Layout errors—particularly those in which plain or perforated tiles are misclassified—can result in errors in any modeling or analytics based upon the layout.
- Incorrect classification of a *blocked* tile as *visitable* (which we refer to as a *false positive*), may result in damage to the robot, either by bumping unexpectedly into an obstacle, or by falling into a hole in the floor.
- Incorrect classification of a *visitable* tile as *blocked* (which we refer to as a *false negative*, may create spurious bottlenecks that reduce tile coverage efficiency (by forcing the robot to take a long way around to reach a set of tiles), or tile coverage (by completely isolating some tiles).

In the remainder of this subsection, we focus on mechanisms we have developed to mitigate the impact of navigation errors caused by false positives and false negatives, as these are much more serious in nature than the layout errors. (Moreover, reducing false positives and false negatives also improves layout accuracy.)

5.1.1 *False positives*

We use three mechanisms in tandem to ensure effective mitigation of false positives:

- The built-in *cliff sensor* helps avoid falling into holes (the occasional missing tile);
- The built-in *bump sensor* stops the forward motion of the robot as soon as it makes contact with an obstacle;
- A custom *overhang detection* mechanism uses stereo infrared sensing to avoid obstacles that lie above the field of view of the downward-pointing webcam.

If either of the two built-in sensors is triggered, the robot's navigation logic causes it to back up and return to the center of the previous tile, and the visited tile is now classified as *blocked*.

The third case is more interesting. Even if the vision system correctly identifies that the tile itself is unobstructed at the floor level, the robot may still be placed in jeopardy if it visits that tile because its tall aluminum pole may run into an overhang or other obstacle that lies entirely above the floor, out of range of the vision system. Such a situation is depicted in Figure 5.

Overhangs encountered in data centers include desks, tables, cables, and slide-out keyboard trays mounted in racks. Overhangs are potentially hazardous to the robot's well-being, as they can push the robot's pole backward, thereby stopping the robot in its tracks or even upending it.

Figure 5: The robot getting upended in a futile attempt to scoot under a mobile whiteboard left in the data center.

To avoid such problems, we initially mounted an upward-pointing Sharp, model GP2Y0A02YK0F, infrared proximity sensor at the leading (front-pointing) edge of the robot, tuned so that it could detect obstacles up to the top of the pole. The Sharp proximity sensor consumes 22 mA of current, has an average response time of 33 ms, can measure distances to obstacles from 20-150 cm and can detect the presence of obstacles up to approximately 250 cm. After observing that the infrared sensor was frequently fooled by intense or reflected lighting, we changed the configuration to that shown in Figure 6. Two of the Sharp sensors are installed on the robot base just behind the robot's leading edge and separated by about 15 cm, and the sensors' sensitivity has been reduced to approximately $3\frac{1}{2}$ feet (115 cm) in order to reduce the likelihood of interference from intense lighting. Since the robot will not detect overhangs above $3\frac{1}{2}$ feet, care must be taken prior to any scan to clear the center of overhangs lying between $3\frac{1}{2}$ feet and $6\frac{1}{2}$ feet in height.

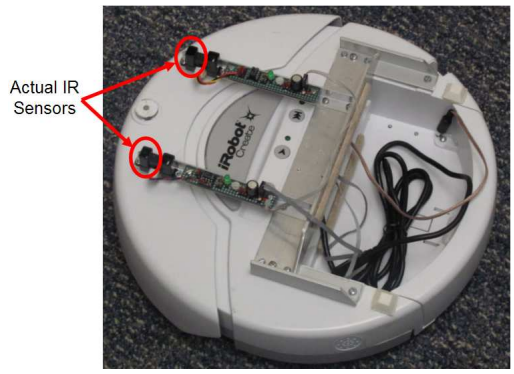


Figure 6: The Create base with mounted dual upward-pointing infrared sensors. The actual sensors point up from just inside the leading (curved) edge of the robot, as indicated.

5.1.2 *False negatives*

False negatives are undesirable because they can reduce both the amount and the efficiency of tile coverage. To reduce their likelihood, we exploit a simple observation. As illustrated in Figure 7, a tile may appear visitable from one angle of approach, but blocked from another. Since, during the course of the robot's exploration, it typically views the same tile from different orientations, we can employ a simple *OR'ing* approach in which we classify a tile as *visitable* if the vision system deems it *visitable* during at least one encounter.

Figure 7: Depending on the angle of approach, the same tile may appear *blocked* (left) or *visitable* (right).

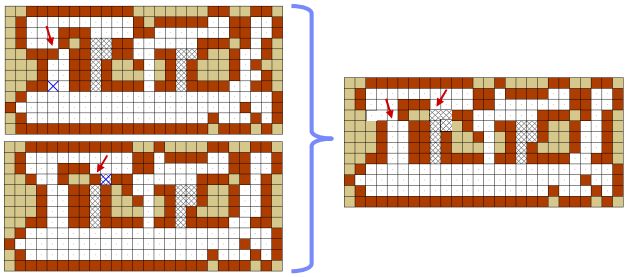


Figure 8: Example application of OR'ing to an actual data center scan. The left layouts show two scan results with misclassifications. In each layout, a key misclassification is highlighted. The right layout shows the achieved layout when OR'ing is enabled.

Figure 8 illustrates how false negatives can reduce tile coverage in a real data center, and how the OR'ing approach can help solve the problem. In a particularly troublesome 1000 square foot data center, a first layout was generated by the robot (top left). Out of 145 visitable tiles in the data center, three or four were misclassified as blocked, one of them resulting in an apparent dead-end that forced the robot to take a very long route to explore tiles that could have been reached easily had it realized that it could traverse the tile. Then, the robot's memory of the layout was deleted, and a second scan was conducted, beginning from a different tile, and resulting in a different layout (bottom left) that was nearly identical to the first, except that a different small set of tiles was misclassified, one of which created a different exploration bottleneck that also reduced the exploration efficiency. In some data centers, the resulting bottleneck can be so severe as to completely isolate several tiles, preventing the robot from visiting them at all.

The right hand plot of Figure 8 shows that, when OR'ing is employed, the spurious obstacles are eliminated, offering the potential for improved coverage and reduced exploration overhead. To understand the potential improvements from that could be realized in practice, we took three scans of the 1000 square foot data center, all of which produced slightly different sets of 3 to 4 misclassified tiles. The three scans were OR'ed together, resulting in a layout that was perfectly correct in terms of visitable/blocked classifications. A worst-case layout was also generated by collecting all of the misclassifications that occurred during any of the three scans. The simulator was then used to determine how much time the robot would take to traverse the OR'ed layout, the

worst-case layout, and the layouts that were generated during the three scans.

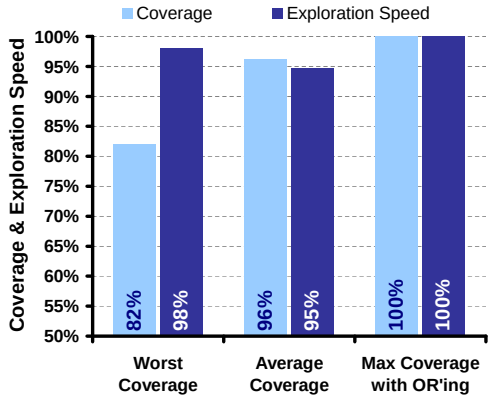


Figure 9: Coverage and exploration speed improvement using vision classification based on OR'ing.

In Figure 9, the light bars represent the average ratio of visited tiles to visitable tiles, while the dark bars represent exploration speed per visited tile. For the worst-case layout, 18% of the visitable data center area cannot be reached, while the exploration speed is only slightly less than for the perfect layout (by just 2%). In the average case, the exploration speed and coverage reduced by about 5% compared to that achieved with the perfect layout. The reason the exploration speed per visitable tile is better for the worst-case layout than it is for the average case is that in the worst-case layout the robot avoids going down narrow alleys that tend to take longer to explore on average.

5.2 Scooching

According to ASHRAE [3], the most temperature sensitive part of the data center is the air inlet areas of racks. Practitioners who utilize the hand-pushed mobile sensing station, or "cart", described in [27], translate this sensitivity into a need to take extra temperatures there and accomplish this by "scooching" the cart as close as possible to the inlet side of racks and taking an extra measurement there. The operator enters the location of these additional readings into a tool.

To succeed in our goal of eliminating the need for human involvement in conducting data center scans, we have added to the robot a scooching mode. The basic idea is to let the robot bump into obstacles that might conceivably be a rack (as detected by either the bump sensor or the overhang detector), pause for a few seconds at the position of contact to allow the pole to stabilize, record temperatures, add those temperatures plus the recorded position at which they were taken to the set of tile-center measurements, and then return to the previously visited tile.

Depending upon how it is implemented, scooching can slow down robot navigation considerably. We have implemented various degrees of aggressiveness:

- **Bumping**. The robot attempts to visit all tiles, whether or not the vision system has classified them as *visitable*. This way, no tile is missed, and moreover the exact position of an obstacle within a tile can be ascertained, but navigation can be extremely slow.
- **Scooching**. The robot employs bumping, and also takes a temperature sample whenever it makes contact with any obstacle.
- **Selective Scooching**. The robot first scans the entire data center using ordinary lazy navigation. Then, likely rack inlet positions are inferred from a combination of layout information (including the location of perforated tiles) and the measured temperature profile. Finally, the robot navigates as efficiently as possible to the rack inlet positions and uses bumping and scooching to obtain the temperatures. This approach can occasionally miss a few inlet locations.

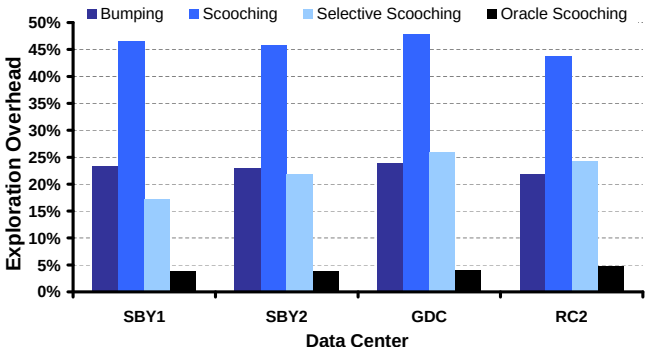


Figure 10: Exploration overheads with bumping, (indiscriminate) scooching, selective scooching, and scooching with oracle knowledge.

To gain some insights into the relative costs of various styles of scooching, we used the simulator to compare the time required to scan four different data centers (all of which have been explored by the robot) using ordinary lazy, vision-based navigation, bumping, scooching, selective scooching, and oracular scooching (which assumes perfect *a priori* knowledge of the inlet positions). Figure 10 displays the results in terms of extra percentage overhead beyond ordinary lazy navigation. The results are quite consistent across the various data center layouts we have tried. Comparing the results for bumping vs. ordinary scooching, we find that the overhead of scooching is about evenly split between navigation and taking temperature samples. Despite the inefficiencies inherent in revisiting positions where the robot has already been, the overhead for selective scooching is just about half that of the naive scooching approach. Moreover, note that most of the overhead for selective scooching is the additional navigation cost for travelling to the inlet locations. Given that the overhead for selective scooching is in the 15%-25% range, compared with about 4% for oracular scooching, it appears that there may be additional room for improvement. It is clear that such an algorithm, if it exists, would have to identify potential rack inlet positions as the robot is doing its initial scan, i.e. it would have to eliminate the need to complete a full scan before traveling back to the inlets to collect their temperatures.

6. SOFTWARE INTEGRATION

IBM Tivoli Maximo for Energy Optimization (MEO) is a commercial product that provides asset management, visualization, and analytics capabilities that are designed specifically for data centers. In its data center viewer window, MEO provides a bird's eye view of the data center layout, displaying in their proper locations various types of assets such as racks, and the individual IT equipment that they contain (including servers and storage devices), power distribution units (PDUs), and CRACs (Computer Room Air Conditioners).

In MEO, one can select a number of layers to superimpose with partial transparency on the asset view, each of which represent two-dimensional slices of the distribution of some measurement of interest, including temperature, hot spots[2], or humidity.

All of these views are based upon interpolations of live sensor readings, which have historically been provided by fixed sensors. At least two distinct types of interpolation are employed. First, if only the live sensor readings are available, interpolation is based on a somewhat sophisticated weighted average of sensor readings in the neighborhood of the point at which the interpolated value is being computed. The weights take into account the distance as measured around thermal barriers. Second, if one has access to temperature readings of fine spatial granularity that were taken at an earlier time, an even more sophisticated and accurate form of physics-informed interpolation can be performed. Historically, the only source of such fine-grained data has been a manual scan by the aforementioned "cart".

Another available layer depicts CRAC zones: regions of the data center for which a particular CRAC provides the primary cooling. While CRAC zones can be computed approximately from certain input data plus live sensor readings, as is the case for temperature interpolation, CRAC zones can be computed more accurately if one has access to fine-grained sensor and layout data collected by the cart.

Referring to Figure 2, the data collected by the robot has multiple valuable uses within MEO. In one mode, the robot can be treated as a replacement for the cart, autonomously exploring a previously unknown data center and populating the asset database with information about the locations of plain and perforated tiles, and other objects, as well as collecting temperature and humidity data that can be used to support more accurate interpolations and analytics such as the CRAC zone calculation. In a second mode, the robot can be run in a periodic or on-demand fashion to collect live sensor readings at locations in which there are no fixed sensors. With the robot, a new type of sensor reading—tile images—becomes available to MEO.

After describing each of these two modes of operation in turn in the next two subsections, we conclude in the final subsection with a few scenarios demonstrating how the robot and MEO can work in tandem to simplify data center energy management.

6.1 The robot as an autonomous cart

When the robot navigates a previously unknown data center, it incrementally transmits layout and sensor data wirelessly to MEO in the same format that was developed to convey data from the MMT cart to MEO. This allows MEO to update its rendering of the layout and the temperature distribution in near real-time, as its knowledge of both continues to grow. As the robot makes its way to new tiles, or views previously unseen tiles, the tile views and visits are recorded and sent to MEO so that a human being, possibly in a remote location, can track the robot's progress. The starting point and current location of the robot are indicated on the map with an "O" and "X" respectively, as illustrated in Figure 11[3].

[2]The hot spot layer is simply a reinterpretation of the interpolated temperature data, in which the pixels are colored according to their offset from a temperature thresholds that is relatively low near the inlet of IT equipment and much higher near the IT equipment exhaust.

[3]In fact information regarding the robot's entire path is sent, though we have found this path to be too cumbersome to present in the GUI.

Currently, tiles are classified as either standard tiles, perforated tiles, or obstacles. Thus the robot does not at present distinguish among different types of assets (e.g. racks, CRACs, PDUs, furniture, or non-assets, such as walls). However, even these coarse distinctions provide a good foundation for the more sophisticated interpolation and other forms of analysis that rely upon some knowledge of data center layout.

6.2 The robot as an on-demand mobile sensor platform

While the robot was originally conceived as a replacement for the MMT cart, the fact that it is fully autonomous opens up a qualitatively new function that is simply not feasible for the cart: it can stay in a data center and serve as a robot-in-residence that runs periodically or in response to a command from administrator or MEO to collect temperature and perhaps other measurements from the entire data center, or some portion of it. In this way, the robot can either supplement or obviate the need for fixed sensors, and can provide temperature maps that are much more fine-grained than can ordinarily be obtained from all but the most densely-monitored data centers. As an example, consider one of the 1000 square foot data centers that has been included in experiments reported in earlier sections of this paper. There are approximately 4 sensors on each rack, and about 20 racks, for a total of about 80 sensors. The robot took 7 temperature readings at each of 145 visitable tiles: 1015 readings in all, more than an order of magnitude more within the same volume of space.

Even more interestingly, the robot can provide additional valuable sensor information that is simply not available from any fixed sensors: streams of still images of tiles as viewed from its onboard video camera. We have modified the user interface of MEO such that an administrator can right-click on a tile and view one or more images of it, possibly from multiple perspectives. This can prove very valuable in diagnosing problems that first surface in MEO as hotspots or other alerts—for example, one can easily see whether a box or piece of equipment has been parked temporarily over a perforated tile, blocking the flow of cool air into the inlet side of racks.

6.3 Usage Scenarios

Here we briefly describe two of the many scenarios that illustrate how the robot can be employed with MEO to facilitate energy management and problem diagnosis in data centers.

Improving spatial resolution of the temperature map: One of the fundamental ways in which the robot's dense temperature scans comes in handy is in improving the quality of the overall interpolation. To get a handle on the extent of this improvement, using data from a robot scan we performed in the Poughkeepsie RC2 data center, we simulated placing four static static sensors on every rack, two on the air inlet side and two on the exhaust side, with one pair placed 1.5 feet from the floor, and another pair placed 4.5 feet from the floor. We then computed how well these static sensors predicted the actual temperatures taken by the robot at each of the seven elevations that it measures (0.5 feet up to 6.5 feet) at the center of each tile. In this data center there were 18 racks so we simulated a data center instrumented with 72 sensors. In contrast, the robot took its 7 sensor readings (not counting humidity) at the center of 251 tiles for a total of 1757 readings. A graph summarizing how well these 72 sensors recovered the 1757 sensors using the inverse distance squared interpolation used in the MEO product is depicted in Figure 12. The overall mean absolute error for all inter-

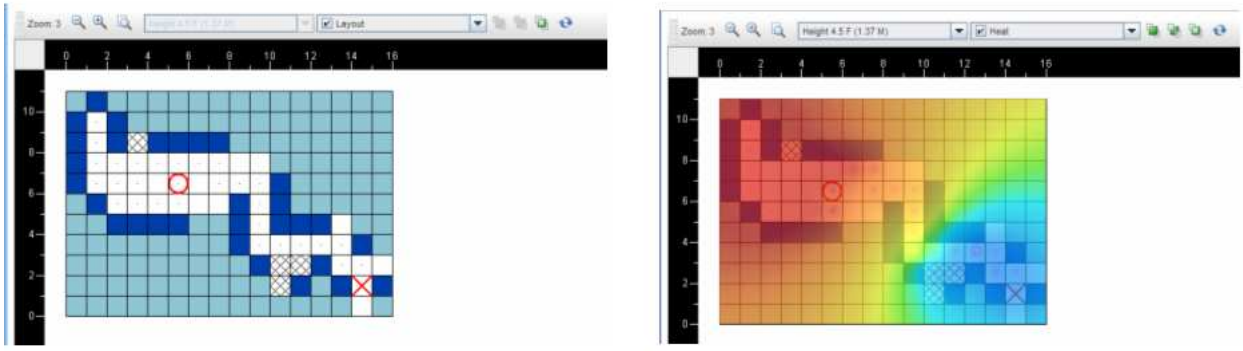


Figure 11: The emerging layout (left) and heatmap (right) as seen from the data center management software.

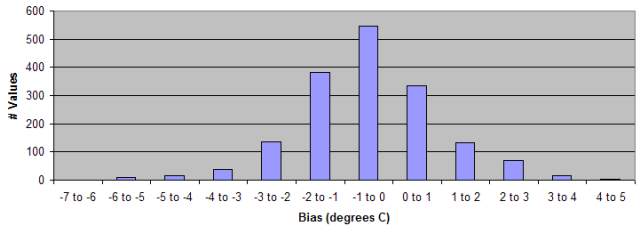


Figure 12: Interpolation bias attributable to 72 static sensors placed at 1.5 feet and 4.5 feet above the floor at both the air inlet and exhaust sides of all racks in the Poughkeepsie RC2 data center.

polated results was 1.18°C. The astute reader will note the net bias towards underestimating temperatures using such a sensor configuration since heat rises and the robot takes relatively more readings above 4.5 feet than it does below 1.5 feet.

Figure 13 further demonstrates the accuracy comparison of the robot's thermal profile to that achievable by different scales of static sensor deployment for the experimented data centers. The figure shows the average error in the interpolated thermal profile with different sensor densities, i.e., the ratio of deployed sensors to the available tiles in the data center. For the range of practical sensor deployments the average error is around $1 - 2^oC$, while the error diminishes to 0 as the density approaches 1, i.e., the number of deployed sensors approaching the number of tiles in the data center.

Automatic or on-demand hotspot investigation: Figure 14 shows the data center viewer with the hotspot layer selected. The temperature readings are interpolated from the 80 fixed sensor readings in the data center. A clear hotspot is developing in the upper left region of the data center.

Using a modified version of the MEO interface, an administrator viewing it could click on the hotspot to see a set of possible actions, one of which includes dispatching the robot to collect temperature and image data within the afflicted region. Using the alerting functions built into MEO, one could also set up a rule that would automatically dispatch the robot. The robot is capable of taking temperature readings confined to a specified region, thereby providing greater spatial resolution that might provide greater insights into the likely cause of the hot spot, and the image data may also help reveal the cause.

7. CONCLUSION

We have described an accurate, efficient, robust, low-cost

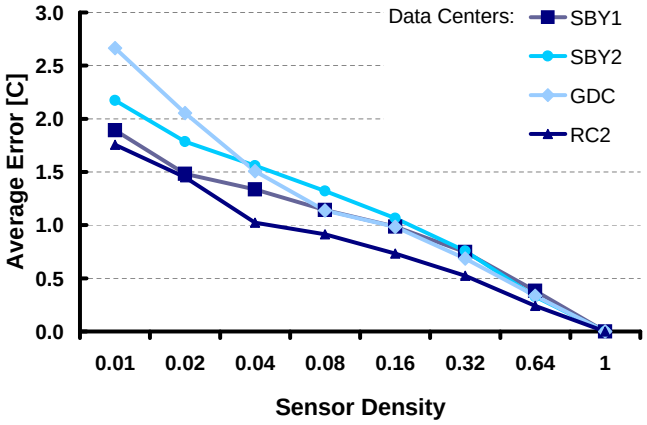


Figure 13: Average error in derived thermal profiles with different number of static sensors for different data centers.

mobile robotic sensor platform, and demonstrated through several experiments that it is practical for mapping and monitoring real data centers, even ones it has never seen before. While much of the existing literature of autonomic computing focuses on ecosystems of software autonomic elements that lead to autonomic or semi-autonomic behavior at the system level, the robot is essentially a physical autonomic element that works alongside its software counterparts (workload managers, etc.) in a data center environment. Like other autonomic elements, the robot manages its own behavior—easily adapting itself to new environments and protecting itself from harm—while contributing to the self-diagnostic and self-optimization capabilities of the data center as a whole.

8. REFERENCES

[1] J. G. Koomey, "Estimating total power consumption by servers in the U.S. and the world," 2007.

[2] U.S. Environmental Protection Agency ENERGY STAR Program, "Report to congress on server and data center energy efficiency," 2007.

[3] ASHRAE Publication, "2008 ASHRAE environment guidelines for datacom equipment: Expanding the recommended environmental envelope," American Society of Heating, Regrigerating and Air-Conditioning Engineers, Inc., Tech. Rep., 2008.

[4] C. Patel, C. Bash, and C. Belady, "Computational fluid dynamics modeling of high compute density data centers to assure system inlet air specifications," *Proc. ASME Int'l*

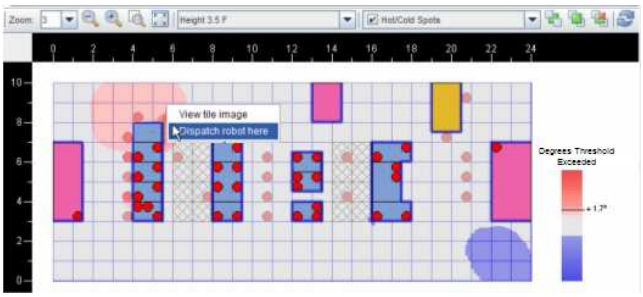


Figure 14: A hot/cold spot map. Hot air is spilling around the sides of the racks at the end of the left hand cold aisle creating a hot spot. The right mouse (context) menu shown is an extension of the MEO interface to allow interaction with the robot and its collected artifacts.

Electronic Packaging Technical Conference and Exhibition, 2001.

[5] C. Patel, C. Bash, R. Sharma, A. Beitelmal, and R. Friedrich, "Smart cooling of datacenters," *Proc. of the PacificRim/ASME Int'l Electronics Packaging Tech. Conference and Exhibition (IPACK)*, July 2003.

[6] R. Sharma, C. Bash, C. Patel, R. Friedrich, and J. Chase, "Balance of power: Dynamic thermal management for internet data centers," *IEEE Internet Computing*, vol. 9, no. 1, pp. 42–49, January 2005.

[7] J. S. Chase, D. C. Anderson, P. N. Thakar, A. N. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in *Proc. 18th Symposium on Operating Systems Principles (SOSP)*, 2001.

[8] P. Ranganathan, P. Leech, D. Irwin, and J. Chase, "Ensemble-level power management for dense blade servers," in *Proc. Thirty-third Annual Int'l Symposium on Computer Architecture (ISCA)*, 2006.

[9] M. Femal and V. Freeh, "Boosting data center performance through non-uniform power allocation," in *2nd Int'l Conference on Autonomic Computing (ICAC)*, 2005.

[10] R. Das, J. Kephart, C. Lefurgy, G. Tesauro, D. Levine, and H. Chan, "Autonomic multi-agent management of power and performance in data centers," *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS), Industry and Applications Track*, pp. 107–114, May 2008.

[11] R. Das, H. Hamann, J. Kephart, and J. Lenchner, "Utility-function-driven energy-efficient cooling in data centers," 2010, pp. 61–70.

[12] T. Boucher, D. Auslander, C. Bash, C. Federspiel, and C. Patel, "Viability of dynamic cooling control in a data center environment," in *Proc. of the 9th Int'l Conf. on Thermal and Thermomechanical Phenomena in Electronics Systems (ITHERM)*, Las Vegas, NV, August 2004, pp. 445–452.

[13] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No power struggles: A unified multi-level power management architecture for the data center," in *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2008.

[14] J. Moore, J. Chase, and P. Ranganathan, "Making scheduling "cool": Temperature-aware workload placement in data centers," in *Proc. 2005 USENIX Annual Technical Conference (USENIX '05)*, 2005.

[15] L. Parolini, B. Sinopoli, and B. H. Krogh, "Reducing data center energy consumption via coordinated cooling and load management," *HotPower 08: Workshop on Power Aware Computing and Systems*, December 2008.

[16] R. Nathuji, C. Isci, and E. Gorbatov, "Exploiting platform heterogeneity for power efficient data centers," in *Proc. Fourth Int'l Conference on Autonomic Computing (ICAC)*, 2007, pp. 5–14.

[17] W. Tschudi, E. Mills, S. Greenberg, and P. Rumsey, "Measuring and managing data-center energy use," *HPAC Engneering*, pp. 45–51, March 2006.

[18] C. E. Bash, C. D. Patel, and R. K. Sharma, "Dynamic thermal management of air cooled data centers," in *Proc. of the 10th Int'l Conf. on Thermal and Thermomechanical Phenomena in Electronics Systems (ITHERM)*, San Diego, CA, May 2006, pp. 445–452.

[19] H. F. Hamann, M. Schappert, M. Iyengar, T. van Kessel, and A. Claassen, "Methods and techniques for measuring and improving data center best practices," in *Proceedings of 11th Intersociety Conference on Thermomechanical Phenomena in Electronic Systems*, May 2008, pp. 1146–1152.

[20] H. Hamann, T. van Kessel, M. Iyengar, J.-Y. Chung, W. Hirt, M. A. Schappert, A. Claassen, J. M. Cook, W. Min, Y. Amemiya, V. Lopez, J. A. Lacey, and M. O'Boyle, "Uncovering energy efficiency opportunities in data centers," *IBM Journal of Research and Development*, vol. 53, no. 3, pp. 10:1–10:12, 2009.

[21] C. Patel, "A vision of energy aware computing from chips to data centers," *Proc. of the International Symposium on Micro-Mechanical Engineering (ISMME)*, Dec 2003.

[22] R. Pon, M. Batalin, J. Gordon, A. Kansal, D. Liu, M. Rahimi, L. Shirachi, Y. Yu, M. Hansen, W. Kaiser, M. Srivastava, G. Sukhatme, and D. Estrin, "Networked infomechanical systems: a mobile embedded networked sensor platform," 2005.

[23] A. Singh, A. Krause, C. Guestrin, W. Kaiser, and M. Batalin, "Efficient planning of informative paths for multiple robots," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, 2007.

[24] C. Mansley, J. Connell, C. Isci, J. Lenchner, J. O. Kephart, S. McIntosh, and M. Schappert, "Robotic mapping and montiroing of data centers," *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[25] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe, "Mobile robot positioning: Sensors and techniques," 1997.

[26] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.

[27] H. F. Hamann, J. Lacey, M. O'Boyle, R. R. Schmidt, , and M. Iyengar, "Rapid three dimensional thermal characterization of large-scale computing facilities," *IEEE Trans. Comp. Pack. Techn.*, vol. 31, no. 2, pp. 444–448, 2008.

[28] J. Hamilton, "An architecture for modular data centers," 2007.