RC25060 (W1010-068) October 19, 2010
Computer Science

IBM Research Report

Geometric Aspects of Robotic Mapping and Monitoring of Data Centers

Chris Mansley
Department of Computer Science
Rutgers University
Piscataway, NJ 08854 USA

Jonathan Connell, Canturk Isci, Jonathan Lenchner, Rajarshi Das, Jeffrey O. Kephart
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

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

Geometric Aspects of Robotic Mapping and Monitoring of Data Centers[1]

Chris Mansley† Jonathan Connell* Canturk Isci* Jonathan Lenchner*
Rajarshi Das* Jeffrey O. Kephart*

Fig. 1. The THOMAS system developed for data center monitoring

***Abstract*— We describe THOMAS, an inexpensive autonomous robot capable of navigating previously unseen data centers and monitoring key environmental parameters. We show the effectiveness of THOMAS with two experimental studies and discuss various practical geometric issues that have arisen during its design and evaluation.**

I. 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 [4], [7]. All of the power required to run IT equipment is ultimately dissipated as heat, and a comparable amount of power may be required by cooling systems employed to remove it. Thus efficient data center cooling is of paramount importance.

A common contributor to data center (DC) cooling inefficiency is over-aggressive cooling, which can result from a lack of understanding of the DC temperature distribution or failure to properly adjust cooling system parameters.

†C. Mansley, Department of Computer Science, Rutgers University, Piscataway, NJ 08854, USA. cmansley@cs.rutgers.edu

*J. Connell, C. Isci, J. Lenchner, R. Das and J. Kephart, IBM T.J. Watson Research Center, Yorktown Heights, NY 10532, USA. {jconnell, canturk, lenchner, rajarshi, kephart}@us.ibm.com

[1]For the full, separately under-submission, version of this abstract see http://www.research.ibm.com/people/l/lenchner/docs/icra11_submitted.pdf.

In order to enable frequent, low-cost DC monitoring, we have developed THOMAS (THermal Observation using Mobile Autonomous Sensing), shown in Fig. 1, a fully autonomous robotic platform for navigating, mapping and sensing key environmental data. In large data centers that it has already mapped, THOMAS can selectively choose locations at which to collect sensor readings, so as to near-optimally recover a temperature profile given a constrained sampling budget. A robotic sensing platform such as THOMAS enables one to easily and cheaply deploy new sensors or sensor types, avoiding expensive installation costs that would be associated with static sensor deployment.

After describing the design and implementation of THOMAS and discussing its successful field deployment in two data centers, we describe and evaluate a selective sampling method used by THOMAS to reduce the time or number of observations required to obtain a sufficiently accurate thermal profile of a data center. Finally, we present several geometric problems that arise in the context of this application, in the hope that they will inspire more formal problem formulations and creative solutions by the computational geometry community.

II. THOMAS: Design and Implementation

The primary design objective of THOMAS is to support autonomous, low-cost navigation and monitoring of key environmental metrics in a data center. While we have limited the first deployment to include temperature sensors only, natural next steps include adding air flow and humidity sensors. Since these quantities typically vary vertically, we provide sensing capabilities at different heights. In addition to monitoring and navigation, THOMAS also provides live, runtime feedback to higher-level management software for dynamic cooling efficiency analysis, data center reconfiguration, hot/cold spot detection and remediation, and adaptive cooling techniques.

THOMAS was developed on top of the iRobot Create robotic research platform. Its robust, low-cost mobile chassis provides a high-level interface for motor control, odometry and on-board sensors. The capabilities of the Create platform were augmented with a webcam-based vision component for navigation and tile detection, an off-the-shelf netbook for on-board processing and a custom thermocouple interface, attached to an aluminum pole for sensing temperature at

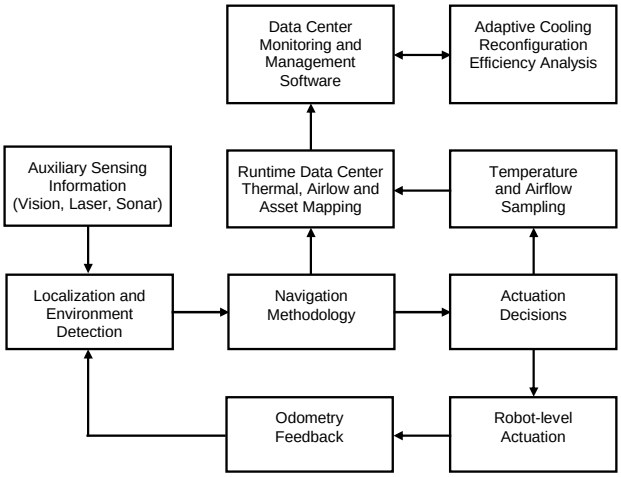


Fig. 2. Mobile DC Monitoring Architecture.

various heights. Figure 2 demonstrates the high-level architectural overview of THOMAS.

The vision system was designed to exploit the grid-like structure of the tiles in a DC. Almost universally, the floor of a DC is comprised of a rectangular grid of industry standard 2 foot by 2 foot tiles, some of which are perforated to allow cool air to pass up through them from a pressurized plenum. THOMAS' inexpensive webcam is mounted approximately 30 inches above the floor to image the tile ahead. From the image the vision system extracts the orientation of the robot relative to the floor grid and its location within the current tile. The vision subsystem is also responsible for differentiating between plain, perforated and blocked tiles.

The basic mode of operation is for THOMAS to traverse the DC, stopping at each tile to take temperature readings. After stopping in a tile it grabs an image of the tile ahead and processes it to determine its actual pose, whether the next tile is *visitable* or *blocked*, and its tile type (i.e., *perforated* or *plain*). Certain challenges associated with the vision process are the extraneous lines introduced by boxes and cabinets, different perforated tile patterns and the occlusion of some tile borders by DC equipment.

The THOMAS vision system is robust with respect to contrast, illumination and partial occlusion of tiles. However, it only provides a differential pose relative to a tile, rather than a full robot pose with respect to the room. We use odometry to keep track of tile crossings and rotations. For global localization, we combine these differential visual observations with odometric feedback using an Extended Kalman Filter (EKF) [6]. The global reference frame is defined by the starting location and orientation.

Given the current tile pose in the global reference frame, the map containing the tile type (perforated or not) and tile obstruction (occluded or not) must be created. In order to fully explore the data center layout, we use a frontier-based exploration system based on A* [3], [8]. As our evaluations show, this exploration algorithm visits, on average, only 16% more tiles than the number of visitable tiles in the DC.

III. Field Deployments

We conducted two sets of experimental deployments of THOMAS in two separate production DCs. The first set of experiments was performed at a research DC in Hawthorne, N.Y. This research DC presented many interesting challenges due to its frequent reconfiguration and most importantly because of its many different types of perforated tiles. It thus became an ideal training DC for THOMAS' tile classifier and we conducted dozens of experimental runs there to tune the vision and localization algorithms.

Our experiments demonstrated the robustness of THOMAS in detecting and navigating through a variety of obstacles that can be encountered in DCs, the reliability of localization with dynamically-varying DC layouts, and the resilience of our vision system in classifying a range of different tile configurations. Overall, this Hawthorne DC comprised 4800 square feet, contained 220 visitable tiles and was completely scanned in 46 minutes.

The second of our deployments was at an enterprise DC in Southbury, CT. This DC was much more static, tiles were relatively uniform, and obstructions were limited in number. Although the tiles in this DC differed substantially from those of the research DC, with different colors, perforation patterns and tile edges, THOMAS successfully scanned this data center on its first run and in several follow-up runs. This DC comprised 960 square feet, contained 115 visitable tiles and our initial scan was completed in 28 minutes.

IV. Selective Sampling

In small DCs such as those used in field deployments, complete scans of every location will complete in a reasonable amount of time. However, in large DCs, complete dense scans could take days to complete, rendering continuous monitoring impractical. A dense scan over a long time period could have inherent temporal variations, reducing the scan's usability. Moreover, in many enterprise DCs, administrative constraints would limit the duration for actively scanning the DC. Thus, in lieu of repeat dense scans, it is desirable to judiciously select a number of informative points that accurately represent the thermal profile of the entire DC.

For selecting a near-optimal subset of sensing locations, we used the mutual information (MI) criterion proposed by Guestrin et al. [2]. Given the complete set of N samples we choose a subset of size K such that we can best recover the temperatures at the remaining $N - K$ locations, using an interpolation method known as Gaussian Process regression [5]. If we let $\mathcal{A}$ denote the set of locations already measured by the robot, our objective is to maximize the change in mutual information by adding a new sensing location, in other words, we wish to maximize:

$$MI(\mathcal{A} \cup y) - MI(\mathcal{A}) = H(y|\mathcal{A}) - H(y|\bar{\mathcal{A}}) \qquad (1)$$

where $H(x|z)$ is the conditional entropy and y is a proposed new sensing location.

We evaluated our MI-based selective sampling approach in comparison to two additional algorithms: *random* and *uniform*. The first algorithm selects k points at random from

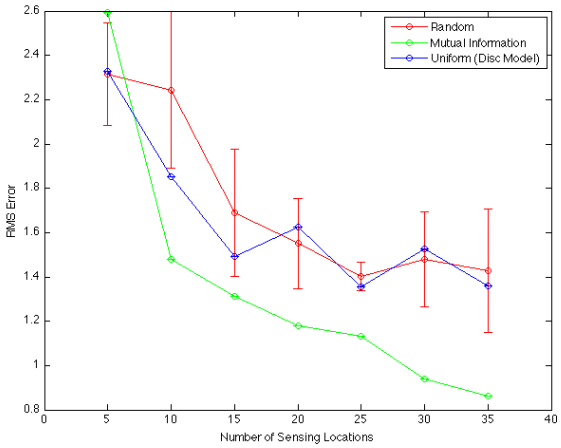
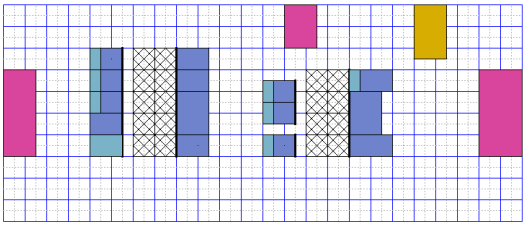


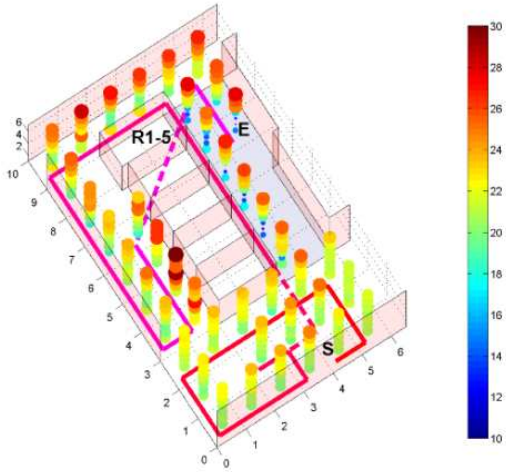
Fig. 3. RMS errors for different sparse sampling strategies and different numbers of sensing locations

all possible sensing locations for a given sample set size of k. The second algorithm picks k "well separated" points from among all sensing locations, in a sense that will be described in the section on geometric considerations (section V-B). An actual medium-size DC in Zurich, Switzerland containing approximately 500 tiles was used for the evaluations with different thermal profiles. For each experiment, a range of selective sample set sizes, k, were used. These k measurements were used to create an approximation of the thermal profile of the DC. The estimated and the actual thermal profile were compared across the whole DC to evaluate the accuracy of each scheme. Fig. 3 shows the root-mean-squared error for each of the three sampling techniques for different values of k. As the figure shows, our MI-based selection is consistently more accurate than both uniform and random sampling for $k \geq 10$ samples.

We also validated our selective sampling strategy with actual physical experiments in the small production DC in Southbury, CT, the layout of which is depicted in Fig. 4(a). We first configured THOMAS to create a dense scan of this DC. A portion of the temperature profile acquired from this initial scan is shown in Fig. 4(b). After the initial scan, we configured THOMAS to operate in "selective sampling mode", where it computed 10 representative sensing locations, and planned and traversed an expedient path among these locations, taking temperature readings at the identified tiles. It then used Gaussian Process regression to approximate the entire DC thermal profile based on the sparse set of samples. Compared to the initial full dense scan, the RMS errors achieved with two separate sparse scans at 10 sensing points were 1.72 and 2.27°C. These results indicate that in an actual enterprise DC setting, THOMAS can effectively approximate an entire DC thermal profile with a relatively small representative sample. The two sets of 10 samples (and ensuing computation using Gaussian Process regression) took THOMAS only 5 minutes each, compared to 28 minutes for the full scan.



(a) Data Center Layout – Blue rectangles are racks of IT equipment, pink rectangles are air conditioning units, cross-hatched squares are perforated tiles and non-cross-hatched squares are plain tiles. The heavy black line segment along one of the edges of each of the blue rectangles indicates the air intake side of the associated rack.



(b) 3D temperature profile and path taken by the robot in a portion of the data center.

Fig. 4. The layout and thermal profile for the production data center.

V. Geometric Considerations

Having introduced THOMAS and described its successful deployment in two data centers, we cite in this section some of the myriad interesting geometric problems that arise in the context of robotic mapping and monitoring of data centers. One broad class of issues pertains to the vision algorithms used for navigation, and for recognizing specific types of objects such as perforated tiles. However, in this section we choose to focus on a different class of geometric issues: those pertaining to the choice of path followed by the robot.

One of the fundamental challenges in our algorithm design is to visit all tiles we need to visit in as little time as possible. The time-wise efficiency of any algorithm is important because we generally need to be sure that measurements of environmental parameters do not vary over the course of the sampling epoch, and also because we must maximize the robot's coverage between battery recharges, which are highly time consuming.

We divide our discussion of path algorithms into two main categories: path exploration and selective sampling and discuss each in turn.

A. Path Planning Issues

To plan our initial path through all tiles with no foreknowledge of the data center layout, we use frontier-based

exploration based on A*. A* is a "best-first" heuristic enhancement to Dijkstra's shortest path tree algorithm due to Hart, Nilsson and Raphael [3], and the frontier-based variant of it we use is due to Yamauchi[8]. When this frontier-based exploration heuristic encounters a Depth First Search (DFS) dead end, it returns to the closest unvisited tile and then continues in DFS fashion. In experiments using layouts of real data centers with n connected visitable tiles, frontier-based A* visited an average of $1.16n$ tiles, while naïve depth-first search visited an average of $1.84n$.

In addition to considering what tile to visit next, an exploratory path algorithm must also govern the direction in which the robot faces, since the robot is mounted with a single camera facing straight ahead. Turning to look at whether neighbors to the north, south, east or west are visitable is a cost, but the knowledge gained from looking in each of these directions can be valuable. In our first deployment, we initially tried a "lazy looking" algorithm, such that, if the tile directly ahead was visitable, we would never look to the sides to see if these neighboring tiles were visitable or not. The rationale for this strategy is that it avoids the time cost of continually turning to check whether neighboring tiles are visitable. Intuitively, this would appear to be an efficient strategy in tours that consist mainly of long straight, narrow paths. The drawback is that less information is available to frontier-based A* when it tries to quickly dart back to the nearest not-yet visited, but known to be visitable, tile, once it reaches a dead end. Therefore, we have switched from "lazy looking" to the opposite strategy, whereby we turn continually to ascertain the visitability of all neighboring tiles. In the initial stages of the search, the robot appears to twitch a lot, but overall its frontier-based A* behavior is more efficient. We feel it is important to characterize this tradeoff more precisely and adopt appropriately adaptive strategies.

B. Selective Sampling Issues

The selective sampling problem as posed in the previous section is over-simplified in several respects. First, we merely find the k best sampling points that most reliably recover the values at the remaining sampling locations. However, it is more meaningful to find the best k sample points such that the recovered temperature surface is as close to the true surface as possible. The catch is that the "true temperature surface" is unknown. However, one can make the simplifying assumption that our interpolation from the initially dense set of samples matches reality exactly. Trying to reproduce the "true" surface in this way would, in general, give more weight to sampling locations in areas where it is more difficult to sample.

Other issues with our algorithm for finding the k most informative sampling points include (i) the time cost is not considered (i.e. we should consider not the total number of samples, but the time required to visit the sample locations and collect the readings); (ii) we attempt to select k samples in a single shot, and never consider altering that set in light of information collected from the initial samples; and (iii) we strive for more samples in regions in which the spatial gradient of temperature is high, but we are not yet considering regions in which the time gradient of temperature is high. Given that our intent is to capture samples that are representative of a snapshot in time, we should sample at these time-sensitive locations especially quickly. In other words, we should scoot around to these time-sensitive locations and then, in more leisurely fashion, sample the remaining points.

Our comparison of sparse sampling based on mutual information to geometrically uniform sampling raises questions about what "geometrically uniform" sampling really means. In data centers that are approximately rectangular, we might divide the data center into equi-sized rectangles and pick a point as close to the center of each piece as possible. However, if the data center is particularly thin or wide, this can result in points that are not well stratified in one of the dimensions, and hence not well-separated. In highly non-rectangular data centers, the problem may be even worse. Are there better methods? We finally chose a different method for finding well-separated points using circle-packing. How good is this method? What optimality guarantees can be developed for it?

We mention one final geometric problem that pertains to large data centers: no matter how efficiently the tiles are traversed, a dense scan may take too long for it to be regarded as an approximation to a snapshot in time and hence serve as a basis for the determination of near-optimal points to sub-sample from. Suppose that the robot is given a sampling budget of just 30 minutes per day. How can it best weave together a large number of 30-minute samples to obtain a a decent approximation to the temperature distribution across the data center at one instant in time?

REFERENCES

[1] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2nd edition, 2001.

[2] C. Guestrin, A. Krause, and A. P. Singh. Near-optimal sensor placements in gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

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

[5] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

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

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

[8] B. Yamauchi. A frontier-based approach for autonomous exploration. In *In Proceedings of the IEEE International Symposium on Computational Intelligence, Robotics and Automation*, 1997.