

# A Dual-Space Approach to Tracking and Sensor Management in Wireless Sensor Networks

Jie Liu, Patrick Cheung  
Palo Alto Research Center  
3333 Coyote Hill Rd.  
Palo Alto, CA 94304  
650-812-4369/4338  
{jlieliu, pcheung}@parc.com

Leonidas Guibas  
Computer Science Department  
Stanford University, CA, 94305  
650-723-0304  
guibas@cs.stanford.edu

Feng Zhao  
Palo Alto Research Center  
3333 Coyote Hill Rd.  
Palo Alto, CA 94304  
650-812-5078  
zhao@parc.com

## ABSTRACT<sup>1</sup>

Wireless *ad hoc* sensor networks have the advantage of spanning a large geographical region and being able to collaboratively detect and track non-local spatio-temporal events. This paper presents a dual-space approach to event tracking and sensor resource management in sensor networks. The dual-space transformation maps a non-local phenomenon, *e.g.*, the edge of a half-plane shadow, to a single point in the dual space, and maps locations of distributed sensor nodes to a set of lines that partitions the dual space. The detection problem becomes finding and tracking the cell that contains the point in the arrangement defined by these lines. This mechanism can be effectively used for power management of the sensor network – nodes that will not be immediately visited by an event can be turned off to save energy required for sensing, processing, and communication. The approach has been successfully demonstrated on a laboratory testbed built using the UC Berkeley motes sensors. An implemented application of detecting and tracking light shadow edges moving over a sensor field is described.

## Categories and Subject Descriptors

C.2.4 [Computer-Communication Networks]: Distributed systems – *distributed applications*.

## General Terms

Algorithms

## Keywords

Sensor networks, geometric duality, arrangements, target tracking, resource management

---

This work is supported in part by the Defense Advanced Research Projects Agency (DARPA) under contract number F30602-00-C-0139 through the Sensor Information Technology Program. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WSNA'02, September 28, 2002, Atlanta, Georgia, USA.  
Copyright 2002 ACM 1-58113-589-0/02/0009...\$5.00.

## 1. INTRODUCTION

Wirelessly distributed, heterogeneous *ad hoc* sensor networks are defined by several unique characteristics unparalleled to those on conventional centralized sensor platforms. A wireless sensor network can cover a large geographical region, and hence can be used to detect and track multiple, simultaneous events and support multiple active user queries. Because of its dense spatial sampling and multi-aspect, multi-modality sensing, the network can assemble information from spatially diverse sources to improve signal/noise ratio. The redundancy in the network can ensure a certain degree of robustness against node failures. The network may be quickly deployed for a particular application, and the ubiquity and low-cost nature of the MEMS micro-sensors can potentially give users unprecedented access to real-time situational information.

One of the key technologies to enable widespread use of *ad hoc* sensor networks in commercial and military applications is *collaborative signal and information processing* (CSIP) [7], [8]. CSIP provides the data representation and control mechanisms to allow sensor nodes in a network to collaboratively process and store sensor information, respond to external events, and report results. While the sensor data is local to each node, the information content to be extracted from the network is global and must be obtained through collaboration among the nodes. For example, in a scenario of tracking a chemical plume, each sensor only has limited information such as whether certain chemical elements exist at the sensing spot, whereas the global information such as the shape of the plume and its motion need to be determined collaboratively by many sensors. In addition, because of limited node energy reserves (*i.e.*, battery power), such processing and communication must be achieved in an energy efficient way.

This paper addresses a key problem in supporting scalable CSIP for sensor networks, *the use of physical constraints to dynamically define sensor collaboration regions*. The implication of this is twofold:

- This capability enables a network to track non-local spatio-temporal events or objects by aggregating information from multiple nodes.
- It provides a principled mechanism to manage the sensing, communication, and power usage within the network, by using the physical constraints from the sensor layout and a model of the physical event so as

to activate and control only those sensors that can obtain information relevant to the task at hand.

As a surrogate for studying large-scale tracking problems, we have developed a laboratory testbed of a 2-dimensional, wirelessly connected sensor field using the Berkeley motes sensors. We consider an edge detection and tracking problem for a 2-D continuous shadow over the sensor field. If the sensor array is dense, then, locally, the edge of the shadow may be approximated as a straight line segment. We develop a dual-space representation to map the non-local line segment into a local phenomenon in an appropriately parameterized configuration space. We then show how motion constraints from the target shape and dynamics can be exploited to activate only those sensors relevant to the current configuration. The dual-space algorithm has been implemented on the testbed of 16 motes and tested on tracking a moving half-plane shadow.

One important goal of this work is to study the effect of active sensor management on the energy usage. Hence we have initially focused on developing an efficient and practical dual-space algorithm, while making simplifying assumptions on the shape of the object being tracked, the signal processing of sensor events, and the implementation of the algorithm on the testbed. Carrying the result of tracking a half-plane shadow over to objects with arbitrary geometric shapes requires employing more sophisticated modeling algorithms such as those commonly used in computer vision, as discussed in the Discussion section. While our implementation deals with the important problem of sensor noise in detections, the paper will gloss over how we set up the detection threshold in the experiment for space reasons. As an initial implementation, the processing of sensor data is all carried out centrally. A distributed implementation is currently under the way. Despite these simplifying assumptions for the initial implementation, our experiment has allowed us to assess the impact of sensor management on the network energy use.

Activating only relevant sensor nodes has significant savings on the energy use of a sensor network. Modern wireless sensor hardware platforms usually have low-power mode, in which the processor, sensors, and the wireless transmission circuits are put into deep sleep to preserve power. Sensor nodes can be turned back to active mode by receiving wakeup packets. For example, in Berkeley MICA motes [3], one second of sleeping mode can save enough power for sending more than 70 packets, or performing 70K operations. In our experiment, we have observed that only 28% sensors on the average are awake at any given time, out of a total of 16 nodes.

In the remainder of the paper, we will describe the dual-space representation and algorithm (Sections 2 and 3), present experimental results from the testbed (Section 4), and discuss possible extensions of the current work (Section 5).

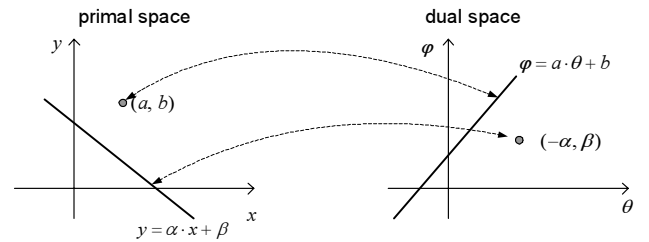
## 2. DUAL-SPACE PRINCIPLE

The approach we take to estimate the edge of the shadow is based on the dual space principle in computational geometry, (see e.g. [1], [5]). In the following, we assume the shadow is a half plane, bounded by a line. We exploit the fact that both a sensor location and the location (line equation) of the edge shadow can be described by two parameters.

What does a dual-space representation buy us? The geometric duality described below allows us to map a seemingly non-local phenomenon (the position of the shadow edge), into a local attribute in the dual space. This allows the sensor nodes to be ordered according to how “close” they are relative to the frontier of the object motion and simplifies the sensor activation procedure. If the sensor activation algorithm were implemented in the primal space, without using the dual-space transformation, each sensor node will have to reason about its distance to the object edge relative to other sensor nodes and the motion of the object, a fairly complex geometric problem to solve.

### 2.1 Dual-Space Transformation

Let us consider a line in a 2-dimensional space (called the *primal space*):  $y = \alpha \cdot x + \beta$ , which is uniquely defined by two parameters  $\alpha$  and  $\beta$ . To represent this line through this pair of parameters, we can simply use the point  $(-\alpha, \beta)$  in another 2-dimensional space (called the *dual space*)<sup>2</sup>. Similarly, a point in the primal space  $(a, b)$  uniquely defines a line in the dual space:  $\varphi = a \cdot \theta + b$ . This 1-to-1 mapping, as shown in Figure 1, is one form of *dual-space transformation*<sup>3</sup>.



**Figure 1. The mapping between the primal space and the dual space.**

This dual-space transform has several useful properties, which follow immediately from the definition:

- If, in the primal space, a point  $(a, b)$  is on a line  $y = \alpha \cdot x + \beta$ , then, in the dual space, the corresponding line  $\varphi = a \cdot \theta + b$  goes through the corresponding point  $(-\alpha, \beta)$ , and vice versa.
- If, in the primal space, a point  $(a, b)$  is *above* a line  $y = \alpha \cdot x + \beta$ , i.e.,  $b > \alpha \cdot a + \beta$ , then in the dual space, the corresponding line  $\varphi = a \cdot \theta + b$  is *above* the corresponding point  $(-\alpha, \beta)$ , i.e.  $\beta < -\alpha \cdot a + b$ . Similar results hold for the *below* relation.
- If, in the primal space, a line  $y = \alpha \cdot x + \beta$  performs a continuous motion, including rotation and translation, the corresponding point  $(-\alpha, \beta)$  performs a continuous motion in the dual space.

<sup>2</sup> We use  $-\alpha$  instead of  $\alpha$  in the dual space so that the properties are easy to derive later on.

<sup>3</sup> It is also called *Hough Transformation* in some literatures.

Now, consider a set of points  $\{P_1, \dots, P_4\}$  and one line  $L$ , in the primal space, as shown in Figure 2, whose corresponding dual-space representations,  $\{p_1, \dots, p_4\}$  and  $l$ , are shown in Figure 5.

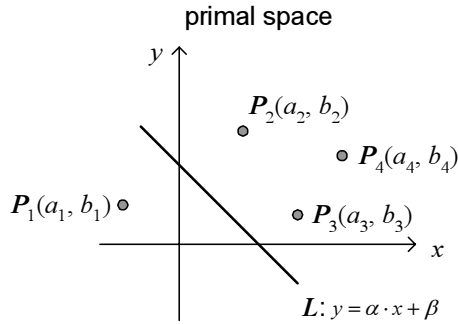


Figure 2. A set of points and a line in the primal space.

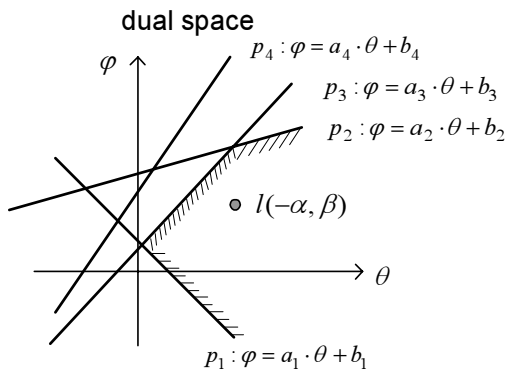


Figure 3. The dual space representation of the points and the line in Figure 2.

In Figure 3, the lines  $\{p_1, \dots, p_4\}$  define a *line arrangement* that partitions the dual space into a set of convex polygons, called *cells* [1], [5]. The boundaries of these cells are line segments lying on the lines  $\{p_1, \dots, p_4\}$ . Obviously, some cells are bounded, while others extend to infinity. The dual of a primal line  $L$  is a point  $l$  that must be contained in one of the cells (in this example, the shaded cell in Figure 3), unless it is on a cell boundary. Let us use  $l < p$  to indicate that the point  $l$  is below the line  $p$  in the dual space; then, the shaded cell in Figure 3 contains all points  $l$  satisfying:

$$\begin{aligned} l &> p_1 \\ l &< p_2 \\ l &< p_3 \end{aligned} \quad (1)$$

When the line  $L$  in the primal space moves,  $l$  moves in the dual space. As long as  $L$  does not rotate across the vertical direction or intersect any point in the primal space,  $l$  will stay in the cell defined by (1) in the dual space. Furthermore, in the dual space,  $l$  can enter other cells only if it crosses one of the cell boundaries, including, conceptually, a boundary at infinity. In particular, as shown in Figure 3,  $l$  cannot intersect  $p_4$ , before it crosses one of the current cell boundaries. This observation is the key for our power management scheme: if  $\{P_1, \dots, P_4\}$  are the positions of four sensors and  $L$  is the boundary of the half plane shadow, then  $P_4$  can be turned off as long as none of  $P_1, P_2$  and  $P_3$  senses a transition.

## 2.2 Line Arrangements

The arrangement of lines in the dual space and the cells they create can be computed by using the *topological sweep* algorithm of Edelsbrunner and Guibas [2], as modified by Rafalim, Souvaine and Streinu to deal with degeneracies [6]. The details of topological sweep algorithm are beyond the scope of this paper; we only describe how the sweep results are used.

A typical topological sweep algorithm computes the segments created by the intersections of the set of lines, and their relative locations in terms of adjacency and direction. For example, Figure 4 shows a subset of the line arrangement. Segment A is on line  $p_1$ , and is uniquely defined by the intersection of A and B (or C), and the intersection of A and D (or E). Furthermore, the relative locations of A with its adjacent cells can be given by:

- B is on the *left up* of A;
- C is on the *left down* of A;
- D is on the *right up* of A; and
- E is on the *right down* of A.

Of course, if a segment extends to infinity, such as G and H among many others in Figure 4, then two of these four adjacent segments may not exist. Also note that the adjacent segments at the same endpoint may not belong to the same line. This may happen when multiple lines intersect at the same point. Given these relations, the cells that have the segment A as their boundary can be determined by “walking” through the relations. For example, the cell that is above A must have A’s left-up segment (B) and right-up segment (D) on its boundary, and so on.

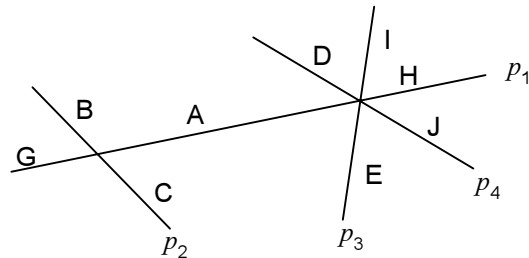


Figure 4. A subset of the line arrangement.

Thus, for an arrangement of lines, we can use the following data structure to efficiently store the segments:

```
Segment = { id;
            Line; left end; right end;
            left-up segment;
            left-down segment;
            right-up segment;
            right-down segment;
            }
```

A table of all these segments in the plane gives sufficient information to construct all the cells and their relative locations.

### 3. HALF-PLANE SHADOW LOCATING AND SENSOR MANAGEMENT

Assume that the shadow is a half plane. By using the dual space transform, we can easily determine the set of sensors at the “frontier”, i.e., the ones that may detect a transition next. We use light sensors to facilitate the discussion in this section. Obviously, the mechanism applies to any sensors that give binary readings.

Let  $0$  represent a *dark* reading at a sensor, and  $1$  represent a *light* reading. Then at any time, the sensor field gives a vector of readings consists  $0$ 's and  $1$ 's. The goal is to identify the set of sensors that the shadow may cross next, and thus estimate the edge of the shadow and turn off the nodes that are irrelevant at this time.

Using the dual space transform, each sensor defines a line in the dual space, and the edge of the shadow is a point. Thus, the problem is converted to determining the cells that are consistent with current sensor reading; these are the cells that may contain the dual of the shadow edge. The sensors corresponding to the boundaries of the cell are the sensors at the “frontier,” so other nodes can be safely turned off. In addition, the corners of the cell, corresponding to several lines in the primal space, define the extreme positions that the edge of the shadow could be.

Note that the constraints in the dual space are in forms of *above* and *below* relations. For the same vector of sensor reading, there may be two possible answers for the location of the shadow; i.e. the shadow is above its boundary or the shadow is below its boundary. For example, the two shadow locations, shown in Figure 5 (a) and (b), yield the same sensor readings,  $[0, 1, 1, 1]$  on  $\{P_1, \dots, P_4\}$ . However, the constraints are different:

- In (a), the constraints are:

$$\begin{aligned} l > p_1; \quad l < p_2; \\ l < p_3; \quad l < p_4. \end{aligned} \quad (2)$$

- In (b) the constraints are:

$$\begin{aligned} l < p_1; \quad l > p_2; \\ l > p_3; \quad l > p_4. \end{aligned} \quad (3)$$

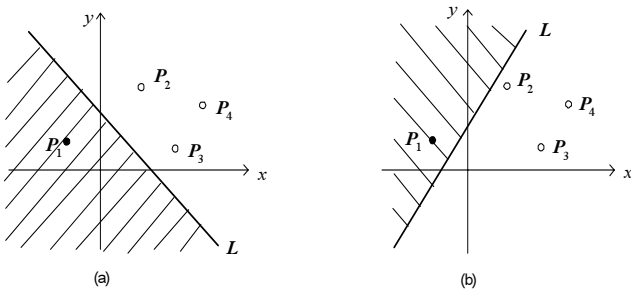


Figure 5. Two different configurations that yield the same sensor reading.

Moreover, in the dual space, situations in Figure 5 (a) and (b) have different representations. The representation of (a) is exactly the same as in Figure 3, while the representation of (b) is illustrated in Figure 6.

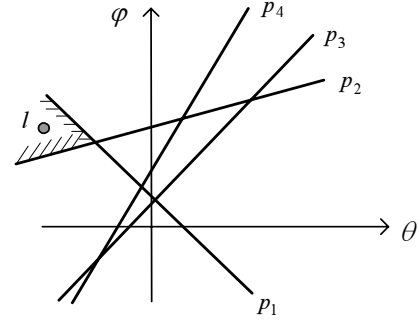


Figure 6. The dual space representation for the situation shown in Figure 5 (b).

In general, the sensors that contribute to the boundaries of the cells in the two situations may not be the same, and both of them should be taken care of. The algorithm that finds the cells is similar to linear programming:

First, assume that the shadow is *below* its boundary  $L$ , then a  $0$  reading at sensor  $P_i$  indicates a constraint:

$$l > p_i,$$

and an  $1$  reading at sensor  $P_j$  indicates a constraint:

$$l < p_j.$$

Next, solve the set of all inequality constraints to find a point in the cell boundary (in fact, a corner of the cell). This can be done through standard linear programming techniques. Finally, the cell can be found by walking along the boundaries of the cell following the constraints and the line arrangements structure. A similar process can be applied for the case when the shadow is *above*  $L$ , after flipping all constraints. Of course, sometimes, only one of the two cases yields a solution, i.e. only one cell satisfies all constraints.

The method described above is static, and could be applied without knowing the history of the motion of the shadow. However, if we take advantage of the fact that the motion of a shadow is continuous, so that the dual of its edge can only move from one cell to an adjacent cell, then the linear programming part of the computation does not have to be performed after the algorithm is started. For example, if the cell  $\{B, A, D\}$  in Figure 4 contained the dual of the shadow edge, and sensor  $p_1$  just flipped its reading, then it is clear that  $\{C, A, E\}$  is the new cell. We can immediately “walk” out the cell by starting at the intersection of A and B (or C), flipping the constraint on line  $p_1$ , and keeping all other constraints unchanged.

Note again that only those sensors corresponding to lines bounding the cell(s) that may contain  $l$  need be kept active: one of these sensors must detect an event before any other sensor in the field can. By tracking the dual cell containing  $l$ , we are effectively tracking this active subset of the sensors. Furthermore, in any line arrangement the expected number of lines bounding an ‘average’ cell is at most four [1] (in the model where all cells are equally likely), independent of the overall number of sensors present. Thus we can expect the number of sensors that need to be active at any one time to be very small. In a large sensor field this may lead to substantial energy savings over time.

## 4. EXPERIMENT

We have built an experiment testbed to validate the shadow tracking algorithms and to demonstrate the benefits of sensor power management using a network of “motes” with light sensors.

### 4.1 Rene Motes

Motes [3] are designed at UC Berkeley to advance the understanding of system issues in wireless networked sensors. Figure 7 shows a Rene mote (one mote design) that we used in our experiment. The CPU on a Rene mote is an 8-bit Atmel AT90LS8535 processor running at 4 MHz. Software can be programmed on to 8Kbytes of flash memory on board. Each mote has four programmable power output pins and seven multiplexed 10-bit Analog-to-Digital converter input pins. One of the A/D channels is connected to a light sensor. The software on motes is developed using TinyOS [3], a small footprint event-driven operating system. The sensor readings are triggered by a timer of 64 Hz. RF communications between motes is carried on the 916 MHz band. Data is passed in a 30-byte packet at up to 10K bits per second.

Our first-cut implementation takes a centralized approach to dual-space computation. In our experiment, motes are considered locally collaborative. A mote configured as a *master* collects data from each *soldier* mote. The soldier motes are configured to broadcast their readings, whenever they detect a significant change in light level. In this small area experiment, the master mote can hear directly from every soldier. However, network routing approaches such as *directed diffusion* [4] may be used for larger-scale testbed. Except for the mote ID, which is used to identify sensor information during the computation, all soldier motes run an identical program. The master mote serves as a base station for the soldier motes and is connected to a PC for processing needs. In the current implementation, all dual-space-based computations are processed and displayed on a PC.

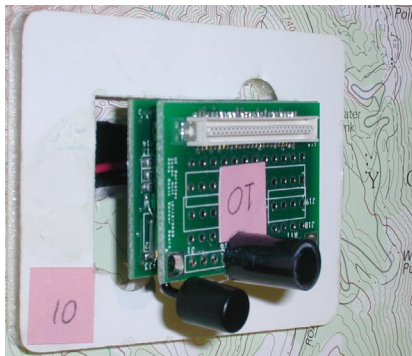


Figure 7. A picture of Berkeley wireless sensor platform, known as a *mote*. The processor, as well as the RF module, is soldered on the lower board; the upper sensor board allows for additional sensor circuitry. A photodiode is provided onboard and is housed inside a collimator in our experiment. The black cylinder at the corner is the antenna.

### 4.2 Sensor Field Board

The experiment is performed on a vertical 6 foot by 6 foot board to allow an overhead viewgraph projector to illuminate the entire

platform. Sixteen motes are mounted on the board at randomized, pre-determined cross-points on a 3.5-inch grid. Figure 8 shows the photograph of the board and the motes, which are numbered from 1 to 16 to facilitate the explanation of how the experiment is performed, mounted on a geographical map.

Although the shadow can enter and leave the sensor field from any direction, without loss of generality, a case will be shown where the shadow comes down from the top left corner.

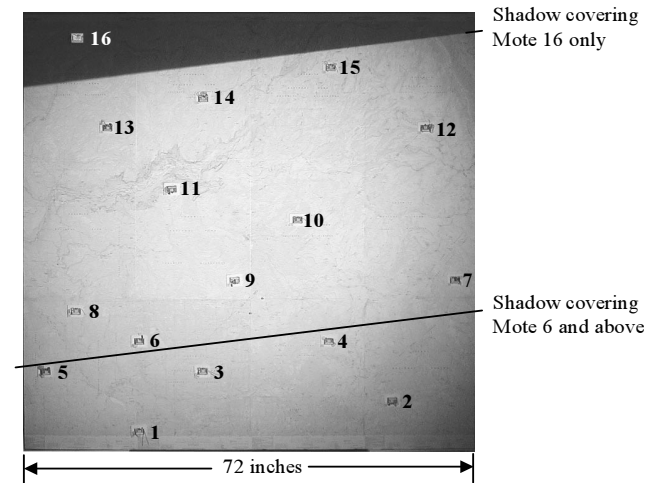


Figure 8. The layout of 16 motes on a vertical board.

### 4.3 Experiment

Before the shadow appears, all of the motes are being illuminated and the shadow can arrive from any direction. Therefore, all the motes on the convex hull are “on-guard” and are depicted as darker nodes in the screen shot in Figure 9. The lines that connect these nodes are the possible extreme positions of the shadow; the shadow itself must be out of that boundary.

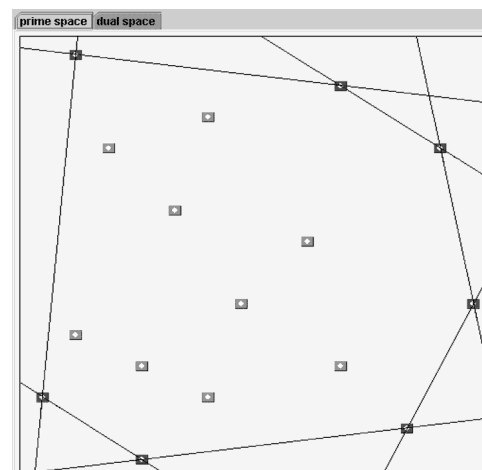
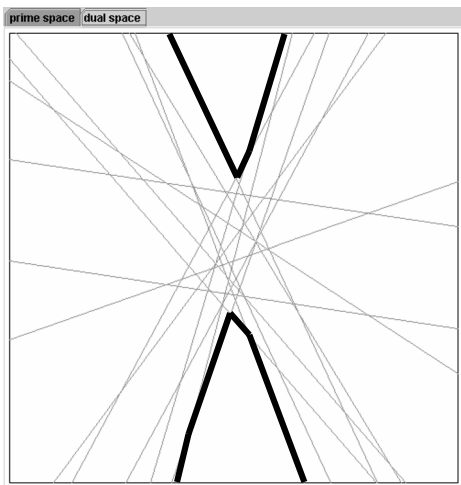


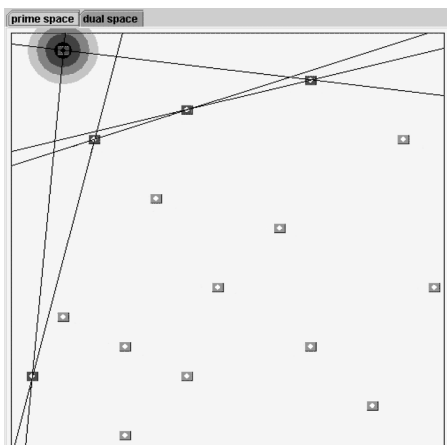
Figure 9. The screenshot when no sensor node is covered by the shadow. The “darker” sensors linked by the lines at the boundary are the sensors on-guard.

Figure 10 shows the screen shot of the dual space representation for the configuration in Figure 9. All possible half-plane lines lying outside the mote perimeter are conceptually points inside the top and bottom troughs outlined in bold. These two cells in the dual space indicate that both “above” and “below” assumptions yield a valid solution. In fact, in the primal space, the shadow can come from either above or below.

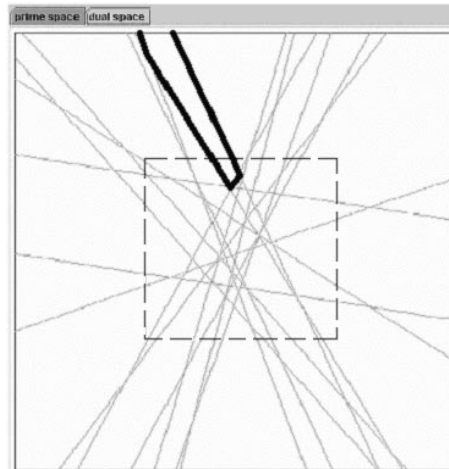
As the shadow moves down to cover mote 16, the mote will detect a change in illumination level and therefore sends out a RF packet to the master mote. The states of the motes are updated as shown in Figure 11. Mote 16 is marked with a big black dot to identify in the GUI as overcast. Four other motes are marked in dark gray, meaning that they are possibly the next ones to be visited by the shadow. The lines connecting the dark gray motes depict position and orientation of the edge of the shadow. In other words, the shadow’s edge must lie within the bounds of all the five lines shown in Figure 11. The corresponding dual space representation of the shadow’s edge is bounded within the cell (not completely shown) marked by bold lines in Figure 12. Note that the “below” cell in Figure 10 is no longer valid, indicating that the “below” assumption yields no solution.



**Figure 10. The screenshot of dual space representation when no sensor node is covered.**

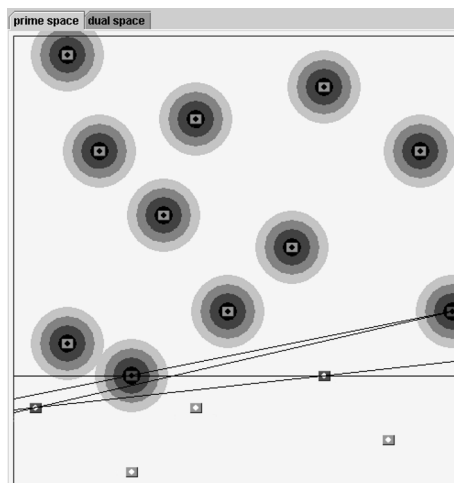


**Figure 11. Mote 16 is covered by the shadow. Motes 16, 15, 14, 13, and 5 are the now frontiers for detection.**

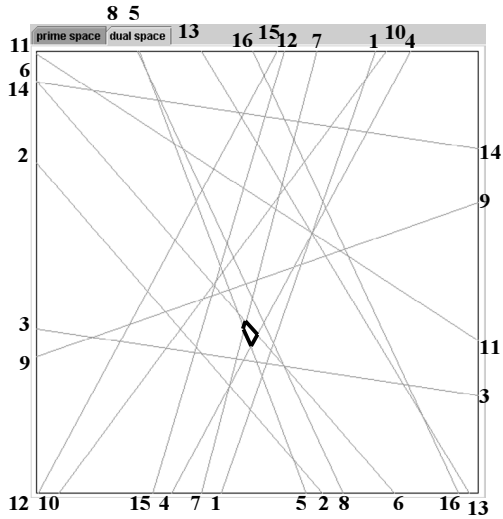


**Figure 12. Dual space representation of shadow covering mote 16 only.**

As the shadow moves further down, more motes are being covered. The dual space representation moves from one cell to an adjacent cell. In each step, the boundary line it crosses corresponds exactly to the mote that is just covered. Figure 13 shows the primal space when all motes with ID greater than 6 are covered. At this time, motes that are far behind of the shadow frontier can also be turned off. The half-plane hence lays within the lines formed amongst motes 4, 5, 6, and 7. Figure 14 shows the corresponding dual space location of the shadow’s edge. When the shadow covers every mote on the board, the shadow’s edge position is undetectable by this set of motes, and the representation resembles that of Figure 9 again.

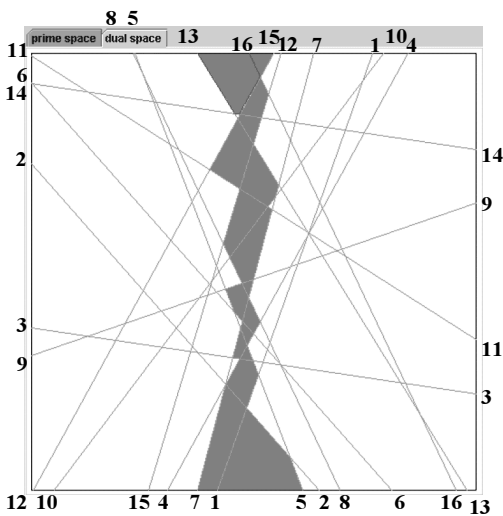


**Figure 13. Motes with ID bigger than 6 are covered by the shadow.**



**Figure 14:** Cell outlined in bold indicates the location of the shadow after it covers mote 6. Numbering of the lines refers to the corresponding mote number. The range of this figure is the box indicates by the zoom-in area in Figures 12.

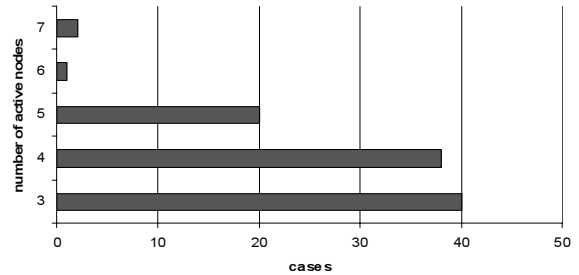
Figure 15 summarizes all the cells the shadow has traversed in the dual space after the shadow has covered the last mote. Since the shadow's motion in this example is mainly a translation from top to bottom without too much rotation, the footprint of the edge in the dual space moves vertically downwards to indicate that there has been mainly a change in the y-intercept but the slope of the half-plane remains largely unchanged. Given a trace like in Figure 15, it is possible to reconstruct the trajectory of the shadow in some optimal means, say with minimal rotation, so that the reconstructed trajectory produces the same sensor readings (i.e. crossing the same cell boundaries) as the original shadow.



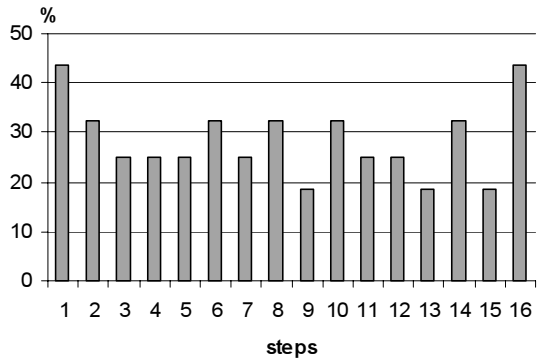
**Figure 15:** The trace of the shadow, in terms of the cells that its edge has traversed in the dual space.

In the dual space, the 16 lines create a total of 102 cells, which cover all possible positions of the shadow edge. The number of boundaries for each cell indicates the number of nodes that need

to be activated when the dual of the shadow edge falls in that cell. Figure 16 shows the distribution of this number for all 102 possibilities. In almost all cases (>97%), only 3 to 5 out of the 16 nodes need to be active at the same time. Figure 17 shows the number of active motes during the previous experiment. On average, less than 30% motes are active. The rest can be put into sleep to preserve power.



**Figure 16:** The number of active nodes in the 16 mote configuration.



**Figure 17:** The percentage of active motes during the experiment.

## 5. DISCUSSION

Based on the ideas of the dual-space transform and half-plane shadow detection, ongoing work continues on developing distributed detection algorithms and the tracking of shadows of arbitrary shape.

### 5.1 Distributed Detection

In the current implementation, all dual-space transforms and boundary detections are computed centrally on a PC. This is not completely unrealistic, considering that 1) the computational power, in terms of both speed and memory size, on the motes is very limited, and 2) it is reasonable to have a heterogeneous sensor network that consists of a few powerful nodes together with numerous less powerful and cheaper sensors such as the motes. However, it is also interesting to develop distributed algorithms so that homogeneous mote nets may work autonomously.

There are several possibilities for distributed algorithms, depending on the goal of the application. First, note that, for a finite number of sensors in the field, there are only a finite (although quadratic) number of sensor-reading configurations.

We may use a group of, say 16, sensor nodes as a cluster and pre-compute all the cell arrangements into a table and store it on the sensor nodes. These clusters can then serve as tiles in a large scale system. Within a cluster, turning off and waking up sensor nodes becomes a simple table lookup. A higher-level system can then connect the edge estimates from each cluster to form the global shape. An alternate is to have each mote only know those dual space cells that are incident on the line representing it. These cells correspond to the concept of the *zone* [1] of a line in a line arrangement, and it is known that the storage required will only be linear in the total number of lines (motes). A third alternate is to perform the dual computation on-line. A simple approach is for each node to compute its own constraints intersecting with a partial result, which is passed from mote to mote. For example, the first mote computes a half plane, and passes the information to the second mote. Then the second mote adds in its own constraint and obtains a wedge, and so on. After passing through all the motes, a cell or two cells are found. This process can be used to initialize the sensor field, or performed occasionally to collaboratively diagnose sensor failures.

## 5.2 Edge Refinement Using Mobile Sensor Network

The size of a cell in the dual space represents the “freedom” that the edge of the shadow has in the primal space. The smaller the cell is, the more accurate our estimate is of the edge of the shadow. If we suppose that the sensor nodes actually have mobility, then we can further refine the shadow detection accuracy by moving the sensor nodes or adding more sensors.

Given a cell in the dual space that contains the edge of the shadow, and assuming that the motion of the shadow is relatively slow, one of the sensor nodes corresponding to one line segment that forms the cell can move, so that the size of the cell shrinks. In particular, we can optimally choose the sensor node and its motion, so that the size of the cell is shrunk the fastest. When this sensor changes its reading, we know that the edge of the shadow is right on that node. Then, another sensor node can move to further shrink the cell. If the second sensor changes reading, then the edge of the shadow is the line that connects the two sensors. We can easily control the motion of the second sensor node in the dual space representation so that the second node does not overlap with the first one.

## 5.3 Tracking More Complicated Shadows

More complicated shadow detection and tracking can be achieved using half-plane detection as a building block. If the density of sensors is large enough comparing to the size of the shadow, then a (convex) shadow can be considered as the intersection of several half-plane shadows. As shown in Figure 18, sensors surrounding the edges of the polygon shadow can be dynamically clustered, and the same half-plane shadow detection discussed previously can be applied for that cluster. The intersection of these half-planes will give the boundary of the polygon shape. Of course, this detection is always an approximation. For example, in Figure 18, the detected shape may have four edges, while the original shadow has five. Nevertheless, the detected shape will be consistent with sensor readings.

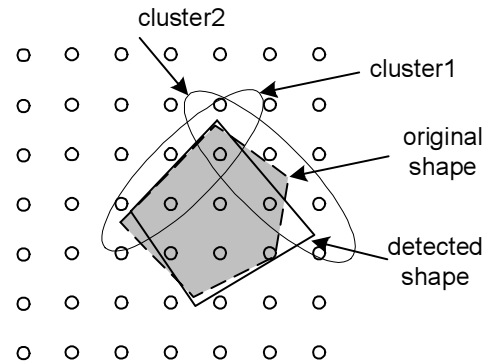


Figure 18. Detecting convex shadows through sensor node clustering.

## 6. CONCLUSION

This paper presents a shadow edge detection and power management scheme in sensor networks using a dual-space transformation. The approach converts non-local phenomena into localized representations and solves the problem in an appropriate configuration space. A testbed of distributed motes was implemented to demonstrate this approach. A fully distributed detection of shadows of more complicated shapes is under development.

## 7. ACKNOWLEDGEMENT

The authors would like to thank Olaf A. Hall-Holt for helping on the topological sweep software, and thank Jim Reich and Juan Liu for inspiring discussions during this work.

## 8. REFERENCES

- [1] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, Berlin, 1997.
- [2] H. Edelsbrunner and Leonidas J. Guibas, “Topologically sweeping an arrangement,” *J. Comput. Syst. Sci.*, vol. 38, 1989, pp. 165-194.
- [3] Jason Hill, Robert Szweczyk, Alec Woo, Seth Hollar, David Culler, Kristofer Pister. “System architecture directions for network sensors,” in Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX) Cambridge, MA, Nov. 2000.
- [4] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin, “Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks,” In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networks (MobiCOM 2000), August 2000, Boston, Massachusetts.
- [5] J. O’Rourke, *Computational Geometry in C*, 2<sup>nd</sup> Ed. Cambridge University Press, 1998
- [6] Eynat Rafalin, Diane Souvaine, and Ileana Streinu, “Topological sweep in degenerate cases,” in Proceedings of the 4th Workshop on Algorithm Engineering and

- Experiments (ALENEX'02), San Francisco, CA, January, 2002.
- [7] Workshop on Collaborative Signal and Information Processing, <http://www.parc.com/cosense/csp.html>, Xerox Palo Alto Research Center, January 2001.
- [8] *IEEE Signal Processing Magazine* special issue on Collaborative Signal and Information Processing for Microsensor Networks, S. Kumar, F. Zhao, D. Shepherd (eds.), vol. 19, no. 2, March 2002.