

Apply Geometric Duality to Energy Efficient Non-Local Phenomenon Awareness using Sensor Networks

Jie Liu, Patrick Cheung, Leonidas Guibas and Feng Zhao[†]

ABSTRACT

Wireless 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 the application of geometric duality in sensor selection and non-local phenomena tracking. Using the dual-space transformation, which 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, one can turn off majority of the sensors to achieve resource preservation without losing the detection and tracking accuracy. To scale up the system, we propose a hierarchical architecture that consists of a small number of computationally powerful nodes and a massive number of ad hoc resource constrained nodes. By taking advantage of the continuity of physical phenomena and the duality principle, we can greatly reduce power consumption in non-local phenomena tracking and extend the life time of the network.

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 non-local phenomena which cannot be captured by any individual sensor. 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.

[†] Jie Liu, Patrick Cheung, and Feng Zhao are with Palo Alto Research Center (PARC), 3333 Coyote Hill Rd, Palo Alto, CA 94304. {jieliu, pcheung, zhao}@parc.com.
Leonidas Guibas is with Department of Computer Science, Stanford University, Stanford, CA, 94305. guibas@cs.stanford.edu

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. Let us consider a scenario of tracking chemical plumes using ad hoc, just-in-time deployment of sensor nets:

The Valley Authority just declared a region-wide emergency: A large-scale hazardous chemical gas leak occurred ten minutes ago near the town of XYZ. The National Guard has been activated to evacuate nearby towns, and to close roads and bridges. To get a real-time situational assessment of the extent and movement of the gas release and aid the evacuation, a SWAT Team is called in. Three unmanned aerial vehicles (UAVs) are immediately launched from an open field 15 miles south of the accident site, each carrying 1,000 tiny wireless chemical sensing nodes (see Figure 1). Upon flying over the vicinity of the accident site, the sensor nodes are released. The nodes self-organize into an ad hoc network, while airborne, and relay the tracking result back to the UAVs nearby: Where is the plume? How big is it? What is the shape? How fast is it moving?



Figure 1. Tracking chemical plumes using ad hoc distributed sensors.

In this example, 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 article addresses a key problem in supporting scalable and power-efficient information processing for sensor networks, *the use of physical constraints to dynamically define sensor collaboration groups*. 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.

We consider a dense sensor network so that the edge of a non-local phenomenon can be piece-wise approximated by straight lines, and study an edge detection and tracking problem for a 2-D continuous shadow over the sensor field. We develop a dual-space representation to map the non-local line segments 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. This algorithm can serve as a building block in a scalable hierarchical architecture that overcomes the communication and computation limitations.

One important goal of this work is to study the effect of active sensor management on the energy usage. 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 sleeping mode, in which the processor, sensors, and the wireless transmission circuits are put into deep sleep to preserve power. For example, in Berkeley MICA motes², one second of sleeping mode can save enough power for sending more than 70 packets, or performing ~70K operations. Sensor nodes can be turned back to active mode by receiving wakeup packets using, for example, carrier detect circuit. Thus, the name of the game is to selectively put sensors into sleep without losing the performance of the application. This has traditionally been tackled by adjusting the sampling and communication rate of the sensor nodes. In this article, we introduce a different approach, which is to use application-specific physical constraints to select nodes to be activated. In our experiment of a shadow tracking using 16 motes, we have observed that only 28% sensors on the average are awake at any given time.

2. DUAL-SPACE-BASED EDGE DETECTION

The approach we take to estimate the edge of a shadow is based on the dual space principle in computational geometry, (see e.g. [1], [4]). In the following, we first assume a half plane shadow, bounded by a line. We exploit the fact that both a sensor location and the location (line equation) of the shadow edge can be described by two parameters.

² Available from Crossbow Technology Inc., <http://www.xbow.com>

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 2D space (called the *primal space*): $y = \alpha \cdot x + \beta$, which is uniquely defined by two parameters α and β . To represent this line through this pair of parameters, we can simply use the point $(-\alpha, \beta)$ in another 2D space (called the *dual space*)³. 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 2, is one form of a *dual-space transformation*⁴.

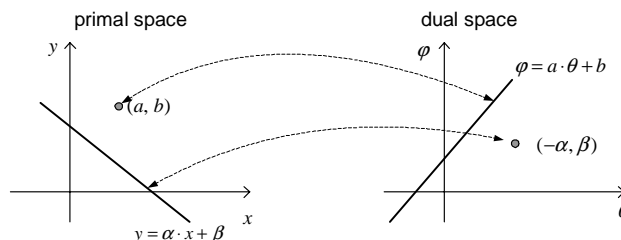


Figure 2. The mapping between the primal space and the dual space.

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

- A. 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.
- B. 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.

³ We use $-\alpha$ instead of α in the dual space so that some properties are easy to derive later on.

⁴ It is also called *Hough Transformation* in some literatures.

C. 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.

For example, consider a set of points $\{P_1, \dots, P_4\}$ and one line L , in the primal space, as shown in Figure 3(a), whose corresponding dual-space representations, $\{p_1, \dots, p_4\}$ and l , are shown in Figure 3(b).

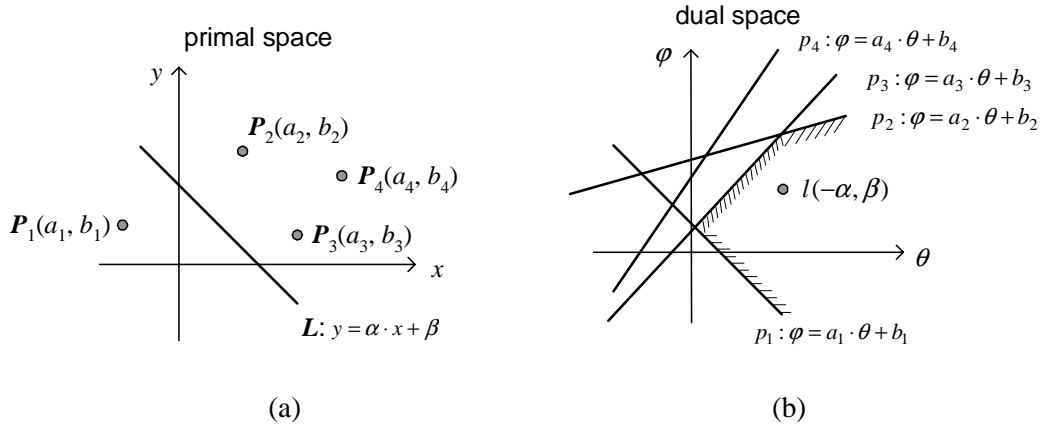


Figure 3. A set of points and a line in the primal space and their representations in the dual space.

In Figure 3(b), 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], [4]. The boundaries of these cells are line segments lying on the lines $\{p_1, \dots, p_4\}$. Obviously, some cells are completely 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(b)), unless it is on a cell boundary. By abusing notation, let us use $l < p$ to denote that the point l is below the line p in the dual space; then, the shaded cell in Figure 3(b) contains all points l satisfying:

$$\begin{aligned}
 l &> p_1 \\
 l &< p_2 \\
 l &< p_3
 \end{aligned} \tag{1}$$

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(b), 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 safely turned off as long as none of P_1 , P_2 and P_3 senses a transition.

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 Rafalin, Souvaine and Streinu to deal with degeneracies [5]. The details of topological sweep algorithms are beyond the scope of this paper, but please note that it is a centralized computation that requires the location of all points.

2.2 Shadow Edge Estimation and Sensor Selection

Assume that the shadow is a half plane. By using the dual space transformation, we can estimate the edge of the shadow by solving the set of constraints imposed by particular sensor readings. Using that information, we can further determine the set of sensors at the “frontier”, i.e., the ones that may detect a transition next when the shadow moves. For ease of discussion, we use light sensors as a metaphor for the sensing model. Obviously, the mechanism applies to any sensing models that give binary readings through quantization. Let $\mathbf{0}$ represent a *dark* reading at a sensor, and $\mathbf{1}$ represent a *light* reading. Then at any time, the sensor field gives a vector of readings consists $\mathbf{0}$'s and $\mathbf{1}$'s. The goal is to identify the set of sensors that bounds the shadow edge, and thus estimate the shadow location 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. Note that the constraints in the dual space are in forms of *above* and *below* relations. The same vector of sensor reading may yield two possible answers for the location of the shadow; i.e. the shadow is above its edge or the shadow is below its edge. For example, the two shadow locations, shown in Figure 4 (a) and (b), yield the same sensor readings, $[\mathbf{0}, \mathbf{1}, \mathbf{1}, \mathbf{1}]$ on $\{P_1, \dots, P_4\}$. However, the constraints are different:

- In (a), the constraints are:

$$\begin{aligned} l &> p_1; & l < p_2; \\ l &< p_3; & l < p_4. \end{aligned} \tag{2}$$

- In (b) the constraints are:

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

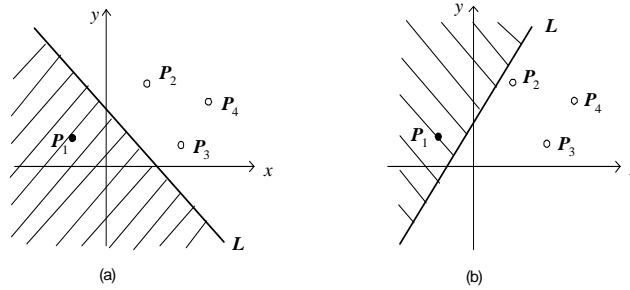


Figure 4. Two different configurations that yield the same sensor reading.

Moreover, in the dual space, situations in Figure 4 (a) and (b) have different representations. The representation of (a) is exactly the same as in Figure 3(b), while the representation of (b) is illustrated in Figure 5.

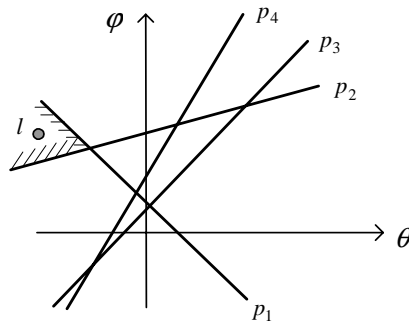


Figure 5. The dual space representation for the situation shown in Figure 4 (b).

The cells that are consistent with the set of sensor readings can be computed via linear programming over the results of a topological sweep. A 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 6 shows a line arrangement. Segment C is on line p_1 , and is uniquely defined by the intersection of B and C and the intersection of C and D. Furthermore, the relative locations of C with its adjacent cells can be given by:

- B is on the *left up* of C;
- G is on the *left down* of C;

- D is on the *right up* of C; and
- H is on the *right down* of C.

Of course, if a segment extends to infinity, such as A and B among many others in Figure 6, 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 C as their boundary can be determined by “walking” through the relations. For example, the cell that is above C must have C’s left-up segment (B) and right-up segment (D) on its boundary, and so on.

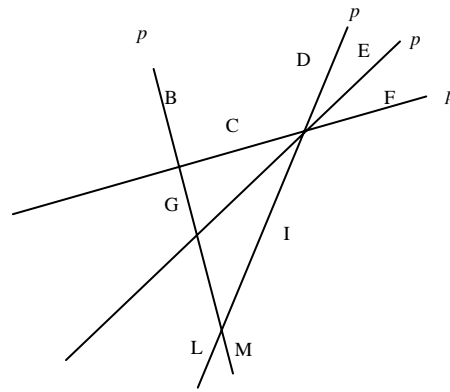


Figure 6. A line arrangement of four lines.

Given the line arrangement information, the cells that satisfy all constraints can be found using linear programming. First, assume that the shadow is *below* its boundary L , thus a $\mathbf{0}$ reading at sensor P_i indicates a constraint $l > p_i$, and a $\mathbf{1}$ reading at sensor P_j indicates a constraint $l < p_j$. Next, using standard linear programming techniques, we can solve the set of all inequality constraints to find a point at the cell boundary (in fact, the point is at a corner of the cell). 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.

Once we find the cells that satisfy the sensor reading constraints, the corners of the cells, corresponding to several lines in the primal space, define the extreme positions that the edge of the shadow could be in. Each pair of lines, intersecting at a corner point, together with the corresponding contains on the lines, gives a wedge in the primal space. For example, in Figure 3(b), the corner of the intersection of p_1 and p_3 together with the relations that the cell is above p_1 and below p_3

defines a wedge that contains all lines that are above P_1 and below P_3 in the primal space. Similarly, the intersection of p_2 and p_3 and the fact that the cell is below p_2 and below p_3 , gives a wedge that contains all lines that are below P_2 and P_3 . For each cell, the intersection of these wedges represents the estimate of the shadow edge. That is, the edge of the shadow must be within that wedge under a certain assumption, *e.g.* dark means below. Of course, if there are two consistent cells in the dual space, the union of each shadow edge estimate gives the overall answer. For example, mapping back the cells in Figure 3(b) and Figure 6, we get a wedge that is shown in Figure 7. In general, the size of the cells in the dual space dictates the freedom of the edge in the primal space. Thus, the denser the sensor field is, the smaller the cells are, and the more accurate shadow edge estimation one can get.

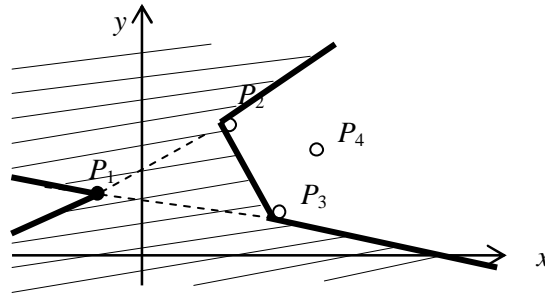


Figure 7. An estimation of the location of the shadow edge, subject to the resolution of the sensor field.

Furthermore, at any time, the dual of the edge of a shadow can at most be in one of two cells. The sensors corresponding to the boundaries of these cells are the sensors at the “frontier,” in the sense that no matter how the shadow moves, it must cross one (or more) of these sensors before crossing any other sensors. So, only those sensors corresponding to the lines bounding the cell(s) need be kept active, therefore brings the energy savings. It can be shown that 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 dense sensor field this may lead to substantial energy savings over time.

2.3 Distributed Sensor Management

The method described above on finding the cells in the dual space 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 system is properly initialized. For example, if the cell $\{C, G, H\}$ in Figure 6 contained the dual of the shadow edge, and sensor p_2 just flipped its reading, then it is clear that $\{H, K, I\}$ is the new cell. We can immediately “walk” out the cell by starting at the intersection of G and H (or C and H), flipping the constraint on line p_2 , and keeping all other constraints unchanged. If the line arrangements are precomputed and stored in the sensor nodes, then finding the new cell is simply a table lookup.

This process is so simple that it is amendable for a distributed implementation on tiny sensors with very limited memory and processing power. After giving each cell a unique ID, each sensor node only needs to remember 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 (nodes). The network can be initialized by having all sensors agree on the same cell ID. The sensor nodes that know nothing about that cell can go to the sleep mode. When one of the sensors on guard notices a flip of sensor reading, it wakes up sensor nodes that are new in the new cell, and broadcasts to all awakened sensors with the new cell ID. The sensor nodes forming the old cell but are no longer in the new cell can turn itself into the sleep mode. For example, in Figure 6, if the current active cell is $\{C, G, H\}$, and the sensor at P_2 just observe a transition of sensor reading, then sensor P_2 can wake up sensor P_3 , and announce the new cell $\{H, K, I\}$. By receiving this information, sensor P_1 can safely go to the sleep mode.

Notice that there are still two potential difficulties for this sensor management mechanism: 1) obtaining the cell configuration is computationally intensive and the algorithm is centralized; and 2) sensor nodes that forming a particular cell in the dual space may be far away in the primal space. There may not be a direct connection to wake up a sensor node if all not-on-guard nodes are sleeping. We will present later a two-tier architecture that over comes these limitations.

3. A LAB EXPERIMENT

We have built an experiment testbed to validate the shadow tracking algorithms and to demonstrate the benefits of sensor management using a network of Rene motes with light sensors. Figure 8 shows a Rene mote (designed by UC Berkeley) 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 rate of

sensor reading is programmed to be 8Hz. RF communications between motes are carried on the 916 MHz band. Data are communicated at up to 10K bits per second.



Figure 8. A picture of Berkeley wireless sensor platform, known as a *mote*. The processor, as well as the RF module, is 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 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 but known location. Figure 9 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.

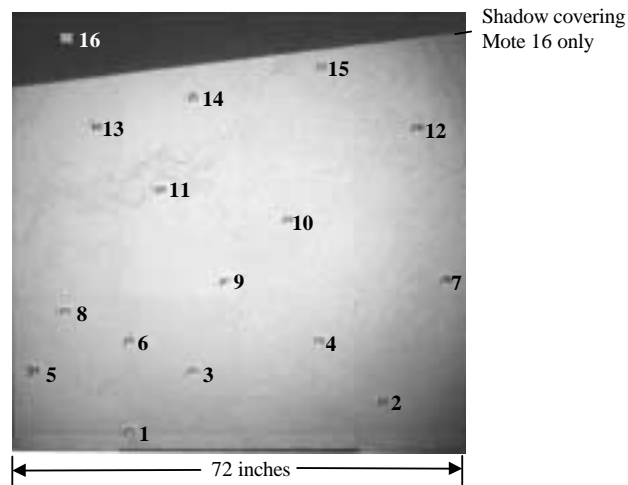
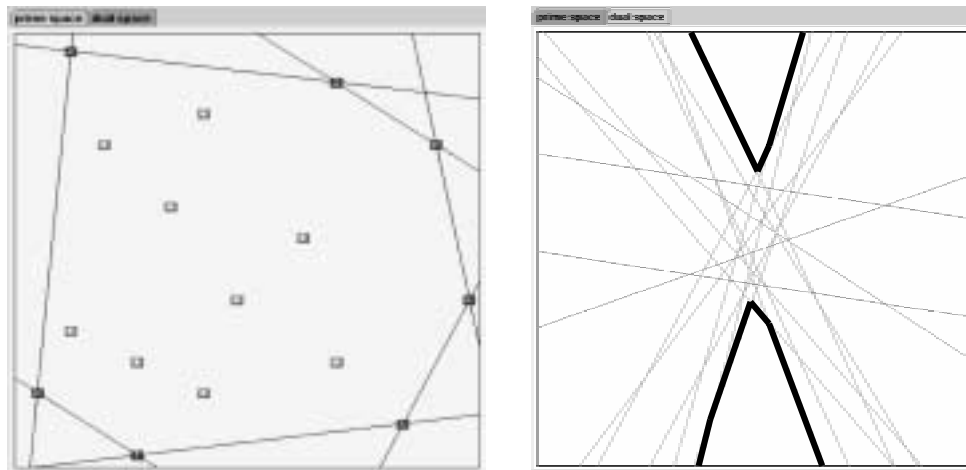


Figure 9. The layout of 16 motes on a vertical board.

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 should be “on-guard.” They are depicted as darker nodes in the primal space

screenshot in Figure 10(a). The lines that connect these nodes are the possible extreme positions of the shadow; the shadow itself must be out of that boundary.



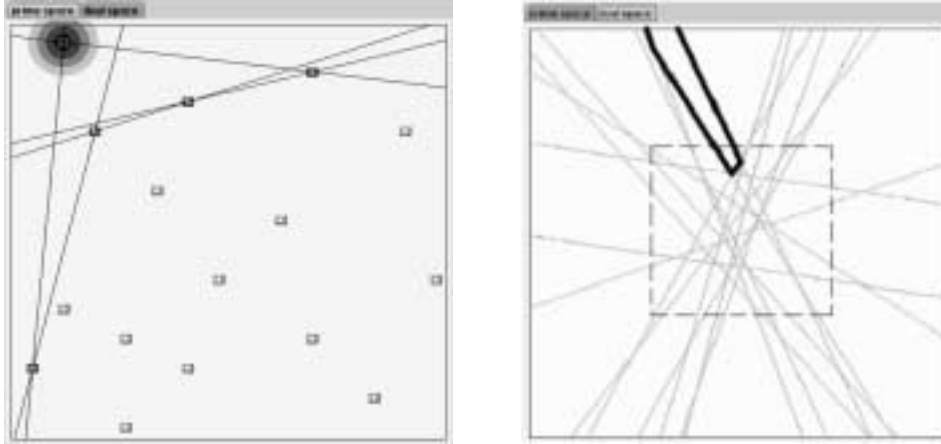
(a) primal space

(b) dual space

Figure 10. 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.

This convex hull can easily be obtained from the dual space representation. Figure 10(b) shows the screenshot of the dual space cells for the configuration in Figure 10(a). 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. The states of the motes are updated as shown in Figure 11(a). 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(a). The corresponding dual space representation of the shadow’s edge is bounded within the cell (not completely shown) marked by bold lines in Figure 11(b). Note that the “below” cell in Figure 10(b) is no longer valid, indicating that the “below” assumption yields no solution.



(a) primal space

(b) dual space

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

Figure 12 summarizes all the cells the shadow has traversed in the dual space after the shadow moving vertically down and 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 12, 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.

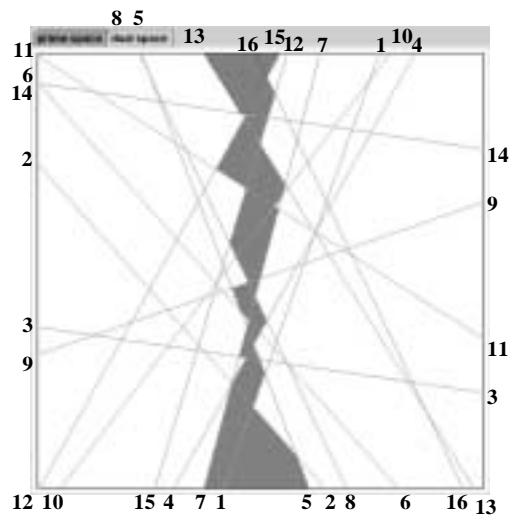


Figure 12. The trace of the shadow, in terms of the cells that it's edge has traversed in the dual space.

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 13 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 14 shows the number of active motes during the previous experiment. On average, less than 30% motes are active at any time. The rest can be put into sleep to preserve power.

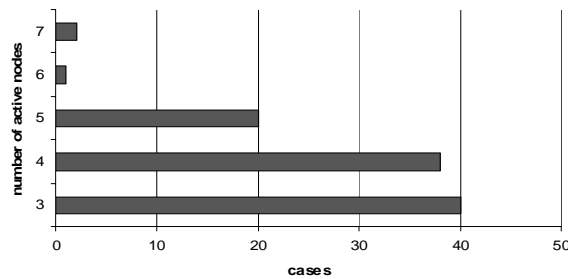


Figure 13. The number of active nodes in the 16 mote configuration.

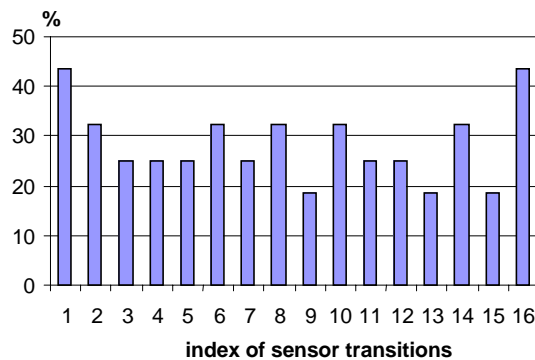


Figure 14. The percentage of active motes during the experiment.

4. A TWO-TIER ARCHITECTURE

To apply the dual space principle and the sensor management mechanisms to scalable non-local phenomena detection and tracking, we propose a hierarchical architecture that consists of a large number of feather-weight sensor motes and a small number of more powerful (in terms of computation) nodes serving as cluster heads⁵. In order to maximize the life time of the network, both motes and the cluster head may have a deep sleep mode that consumes almost no power. They can also

be waked up wirelessly. As shown in Figure 15, the sensor field is virtually divided into a regular grid. Each grid square (whose shape may not be exactly square) has one cluster head (shown as a star) and a set of motes (shown as the circles) deployed in an *ad hoc* fashion. Both cluster heads and motes have wireless communication capabilities. In addition to communicating with the motes in its square, a cluster head can also communicate with other cluster heads in adjacent squares to create a mesh network topology (shown as the hashed lines in Figure 15). The size of the grid squares is small enough that a broadcast from one mote can be heard by all the motes (and the cluster head) in the square, and it is big enough to minimize the total number of squares in the field. The cluster head, which may not be equipped with any sensor, can be placed arbitrarily in the grid. We also assume that all motes are localized, presumably with the help of the cluster heads.

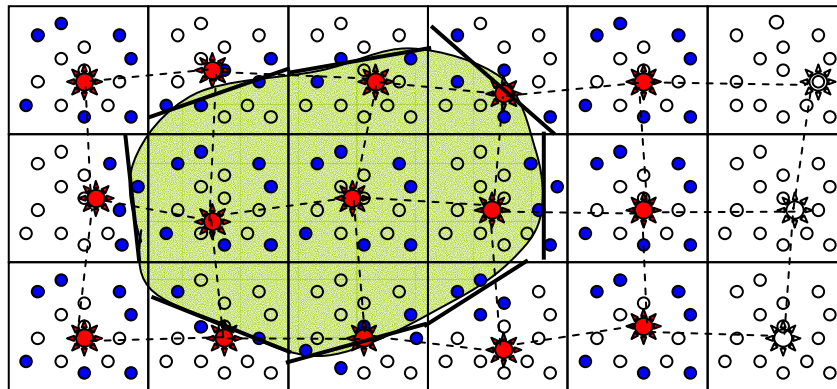


Figure 15. Using a hierarchical architecture to track non-local phenomena.

Suppose that the boundary of the physical phenomena to be detected and tracked, for example, a chemical or biologically cloud, is smooth such that it can be approximated as straight lines in each grid square. Then, a dual-space-based detection, tracking, and sensor management scheme can be applied in each square:

- Once deployed, the motes can send the cluster head their locations. Using that information, the cluster head can perform a topological sweep algorithm and compute all cells in the dual space. This information is then sent back to the motes in the form of a table.
- Upon initialization, the sensor motes also send their sensor readings to the cluster head. A grid square is called *covered* by the shadow if all its sensors has positive reading; it is called *uncovered* if all sensors have negative

⁵ In fact, with sufficient computational power, the sensing nodes and the cluster heads can be of the same kind, and their roles can be dynamically assigned

readings; and it is called *partially covered* otherwise. A grid square is *active* if at least one of its sensor motes is in the active monitoring mode. A cluster head communicates its coverage property to its direct neighbors and activates its own cluster by the following rules:

- A partially covered square is always active.
 - A covered square is active only if at least one of its neighbors is partially covered or uncovered.
 - An uncovered square is active only if at least one of its neighbors is partially covered or covered.
 - In case of a perimeter protection, a grid square is also active if it is at the boundary of the sensor field.
- Using the initial sensor readings, a cluster head can compute the cell in the dual space that contains the shadow edge in that grid square. The initial cell ID is broadcasted to all motes in the corresponding grid square. Motes who know nothing about that cell can switch itself into the sleep mode. After initialization, the cluster head can switch into the sleep mode, too.
 - A partially covered square performs a tracking and sensor selection scheme using the algorithm described in section 2.3. Motes that are not at the frontier of tracking go to the sleep mode.
 - An active but fully covered or fully uncovered grid square performs a detection and sensor selection scheme with the following additional constraints:
 - If the grid square is covered and the square above (below) it is uncovered, then only consider the *below* (*above*) relation in finding the cell in the dual space.
 - If the grid square is uncovered and the square above (below) it is covered, then only consider the *above* (*below*) relation in finding the cell in the dual space.

These constraints help further reduce the number of active motes.

- Once the coverage property of a grid square changes, its cluster head is waked up by a mote and it tells all direct neighboring cluster heads about its new coverage property.

By using this scheme, only the sensors that are absolutely necessary in detecting and tracking the non-local phenomena are active.

One of the assumptions we made in this system is that the edge of the shadow can be piece-wise approximated as straight lines in each grid square. While the straight line approximation assumption is in general true, the boundaries of the lines

may not be aligned with the boundaries of the grid squares. A future work is to dynamically creating the clusters to adapt to complex shapes. As shown in Figure 16, 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 16, the detected shape may have four edges, while the original shadow has five. Nevertheless, the detected shape will be consistent with sensor readings.

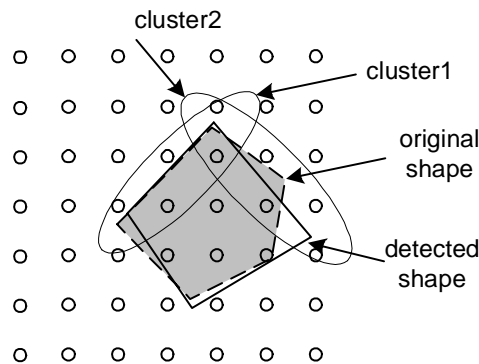


Figure 16. Detecting convex shadows through sensor node clustering.

5. CONCLUSION

Using application specific properties can achieve, in maximum, energy savings for sensor networks. This paper presents a shadow edge detection and power management scheme using a dual-space transformation and a hierarchical architecture. By converting non-local phenomena into localized representations and solves the problem in an appropriate configuration space, the sensor nodes are frontier can be easily identified. Thus other nodes can be safely switched to a power saving mode. A small scale testbed using Berkeley motes has demonstrated the effectiveness of our schemes.

ACKNOWLEDGEMENT

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 authors would also 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.

6. 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 Szewczyk, 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] J. O'Rourke, *Computational Geometry in C*, 2nd Ed. Cambridge University Press, 1998
- [5] 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.