Distributed Localization and Clustering Using Data Correlation and the Occam's Razor Principle

Pankaj K. Agarwal*, Alon Efrat†, Chris Gniady†, Joseph S. B. Mitchell‡,
Valentin Polishchuk¶, Girishkumar R. Sabhnani§
*Computer Science Department, Duke University
†Computer Science Department, The University of Arizona
‡Department of Applied Mathematics and Statistics, Stony Brook University
¶Helsinki Institute for Information Technology, CS Dept, University of Helsinki
§Metron Aviation Inc.

***Abstract*—We present a distributed algorithm for computing a combined solution to three problems in sensor networks: localization, clustering, and sensor suspension. Assuming that initially only a rough approximation of the sensor positions is known, we show how one can use sensor measurements to refine the set of possible sensor locations, to group the sensors into clusters with linearly correlated measurements, and to decide which sensors may suspend transmission without jeopardizing the consistency of the collected data. Our algorithm applies the "Occam's razor principle" by computing a "simplest" explanation for the data gathered from the network. We also present centralized algorithms, as well as efficient heuristics.**

I. Introduction

Consider the following scenario. A set S of n sensors is deployed in a physical environment. Exact locations of the sensors are unknown due to the randomness in the deployment process. Rather, for each sensor $s = 1, \ldots, n$, some region $R_s \subseteq \mathbb{R}^2$, containing s, is known. We call R_s the *feasibility region* of s. For simplicity, assume that each R_s is an axis-aligned rectangle. One might consider sensors deployed rapidly, for example dropped from an airplane, so that the approximate location of the airplane at each instance indicates feasible region where the sensor might lie.

The sensors measure (sense) some environmental characteristic, z. The measurements are taken at D distinct instances of time (D "days"); we let z_s^d denote the measurement of sensor s on day d. The measurements are subject to errors: the true value of z on day d at the location of sensor s lies in the interval $[z_s^d - \varepsilon_s, z_s^d + \varepsilon_s]$ where ε_s, for $s = 1, \ldots, n$, are given numbers known from the specification of the sensors.

In this paper, we suggest a novel technique for simultaneously estimating the locations of the sensors and clustering them according to their measurements. Figure 1 demonstrates our ideas on a simple one-dimensional problem instance. The sensors are deployed on the x-axis; the feasibility regions are the blue intervals. The sensor's order along the line is not specified. The measurement errors are indicated with vertical bars. There are 6 sensors. Solid disks (within the blue intervals) indicate one choice of possible sensor locations. Note that for this localization, any piecewise-linear function fitting the measurements requires at least 5 linear pieces (one such function is shown dashed). At the same time, a slight perturbation of the locations (swapping the pairs s_1 and s_2, s_3 and s_4, and s_5 and s_6) allows one to fit a (single) linear function, which provides a much simpler explanation of the obtained measurements. Restricting the sensors to the positions that make such a simple explanation possible shrinks the set of feasible sensor locations, thereby improving the localization. Figure 2 shows how the localization can be further improved using measurements from two (or more) days. On each day, a function can be fit to the measurements. The constraint that sensor positions be consistent with a linear fit of the data on each day yields an improved localization, narrowing further the set of possible sensor locations.

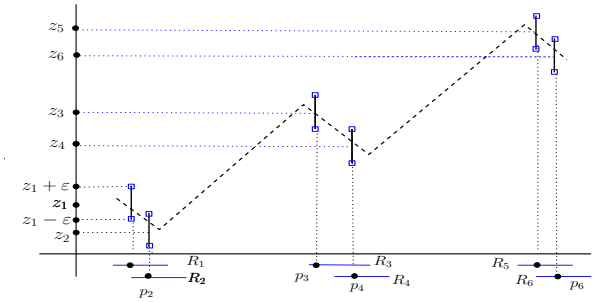

Fig. 1. A one-dimensional example: Sensors $s_1, \ldots, s_6$ (with measurements $z_1, \ldots, z_6$) are known to lie within the blue intervals, R_i ($s_i \in R_i$). If the sensors are positioned at solid disks, then any polyline fitting the data requires at least 5 segments (dashed). However, a simpler, single line, fits the data if the sensor positions are appropriately moved within their intervals (e.g., by swapping the positions of s_1 and s_2, s_3 and s_4, and s_5 and s_6).

Our method is based on exploring the space of bivariate functions that interpolate z over the days. Of course, there exist many possible ways to choose such functions. However, some functions are more likely to provide correct interpolations than others. We employ a philosophical principle of *Occam's razor* as stated by medieval English philosopher and Franciscan friar William of Ockham (ca. 1285-1349):

> *One should not increase, beyond what is necessary, the number of entities required to explain anything.*

In other words, *"given several explanations for the same phenomena, the simplest one is likely to be the correct one."* This principle suggests that a collection of functions is *"correct"* if it is in some sense "simple". We measure the simplicity of a

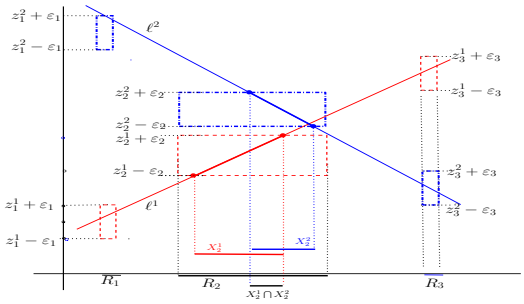

Fig. 2. Data from three sensors s_1, s_2, s_3 is plotted for two time slots ("days"), 1 and 2, rendered in red and blue colors, resp. Sensor s_i is confined to lie in interval R_i. The measurement of sensor s_i at time d lies in $(z_i^d-\varepsilon_i, z_i^d+\varepsilon_i)$. If at each day there is a linear correlation of the sensor measurements, then the line ℓ^d is a possible interpolation for time-slot d; this requires that s_2 lies in the interval X_2^d that is the projection on the x-axis of the intersection of ℓ^d and the box $R_2 \times [z_2^d-\varepsilon_2, z_2^d+\varepsilon_2]$, for $d=1,2$. Hence s_2 must lie in $X_2^1 \cap X_2^2$. Note that ℓ^d could be perturbed a bit in this example.

collection of functions by the number of linear pieces needed to "assemble" the functions in the collection; we formalize the measure in the next section.

For a justification of the application of the principle in a physical setting, imagine that the sensors are deployed in "rooms" in a building or in several "climate zones", and assume, e.g., that z is temperature. Then, one can expect that within the same region the measurements of the sensors are spatially correlated, so that a simple (e.g., linear) function can be fit to the measurements within each room. Segmenting the sensor field into a small number of regions is more likely to reveal such simple relationships.

We propose a combined solution to 3 fundamental problems defined on the sensor field:

Localization. Determine the set of possible locations for the sensors.

Clustering. Group the sensors whose measurements are correlated.

Sensor suspension. Suspend transmission for some sensors, while continuing to receive consistent measurements.

The output of our algorithms consists of (i) a subdivision of the plane into (few) regions such that within each region the measurements are spatially and temporarily correlated, (ii) functions (per region) describing this spatio-temporal correlation, and (iii) a geometric description of feasible placements for each sensor, consistent with the measurement data. We consider mainly the case in which the functions are linear or piecewise-linear. We note, however, that our results (e.g., the algorithm of Section V) extend to computing consistency with respect to more general functions. In particular, it suffices that the problem can be formulated as an *abstract LP-type* problem [22]. For example, we can solve the case in which functions are given by paraboloids or by piecewise-parabolic functions.

II. Preliminaries

The input to our problem is specified by the feasibility regions of the sensors (assumed here to be axis-aligned rectangles), the sensors' measurements, and the measurements' error bounds. Specifically, the input is a triple $(\mathcal{R}, \mathcal{Z}, \mathcal{E})$, where $\mathcal{R} = \{R_s\}_{s=1}^n$ is a set of rectangles, $\mathcal{Z} = \{z_s^d\}_{s=1,d=1}^{n,D}$ are the measurements[1], and $\mathcal{E} = \{\varepsilon_s\}_{s=1}^n$ are the errors. Viewed in the xyz-space, the input is a collection of sets of boxes — one set of boxes per day, one box per sensor in a set.

A *localization* $P = (p_1, \ldots, p_n)$ is an assignment of each sensor s to a point $p_s \in R_s$. In what follows, we consider bivariate functions mapping a point in the xy-plane to a value z. We will identify functions with their graphs (terrains in xyz-space).

A. The Linear Consistency Problem

Let $H = (h^1, \ldots, h^D)$ be a collection of linear functions, $h^d(x,y)$ (planes in xyz), where h^d is associated with day d.

Definition 1. The localization $P = (p_1, \ldots, p_n)$ and the collection H are *linearly consistent* if $\left|z_s^d - h^d(p_s)\right| \leq \varepsilon_s$ for all $s = 1, \ldots, n, d = 1, \ldots, D$.

We define the *linear consistency problem* (***LCP***) as follows: *Given $(\mathcal{R}, \mathcal{Z}, \mathcal{E})$, find a localization P and a collection of linear functions H such that P and H are linearly consistent.*

Suppose that we found a solution to the LCP. This implies that if the sensors are located appropriately, then the spatial correlation of measurements on any day can be explained by a linear function, within the given errors bounds. Of course, the function may be different for each day.

While LCP is a very basic problem, our solution to it is used as an oracle in our algorithms for the Piecewise_LCP (PLCP) problem, which is more general and is described below. Note that LCP asks for just *one* location for each sensor consistent with *one* collection of functions. At the same time, it is of interest to describe the set of "all possible" locations for the sensors that allow for linear consistency. Specifically, we are interested in computing the *localization region of sensor* s, denoted F_s, defined as the set of all locations $p_s \in R_s$ for which there exists a localization $(p_1, \ldots, p_{s-1}, p_{s+1}, \ldots, p_n)$ of the remaining sensors and a collection H of D linear functions such that the localization $P = (p_1, \ldots, p_{s-1}, p_s, p_{s+1}, \ldots, p_n)$ is consistent with H. Formally $F_s = \{p_s \in R_s : \exists p_1 \in R_1, \ldots, p_{s-1} \in R_{s-1}, p_{s+1} \in R_{s+1}, \ldots, p_n \in R_n, h^1, \ldots, h^D : \forall s, d, \left|z_s^d - h^d(p_s)\right| \leq \varepsilon_s\}$.

B. The Piecewise-Linear Consistency Problem

Without loss of generality we may assume that all sensors live in a large bounding triangle, which we will call the *sensor field*. Let g be a bivariate function (e.g. temperature) whose domain is the sensor field. Let $\mathcal{T}$ be a triangulation of the sensor field, i.e., a subdivision of the field into triangles (the triangles' vertices may be arbitrary, they do not have to coincide with sensors). For a triangle Δ from $\mathcal{T}$, denote by $g|_\Delta$ the restriction of g to Δ. We say that g is a *piecewise-linear function with subdomains in* $\mathcal{T}$ if for any triangle Δ from $\mathcal{T}$, $g|_\Delta$ is linear. Note that we do not require a piecewise-linear function to be continuous.

[1]Note that in this paper, unless otherwise specified, superscripts are used to denote the time (day) of measurements, and are not used as exponents.

Let $G = (g^1, \ldots, g^D)$ be a collection of piecewise-linear functions with subdomains in $\mathcal{T}$. Consider a triangle Δ from $\mathcal{T}$. Let $G|_\Delta = (g^1|_\Delta, \ldots, g^D|_\Delta)$ be the restrictions of the functions to Δ. Let $S_\Delta = \{s | R_s \subset \Delta\}$ be the sensors whose feasibility regions lie fully within Δ. Let P_Δ be the localization P restricted to the sensors in S_Δ. We say that localization P and the collection G are *piecewise-linearly consistent* if for any triangle Δ from $\mathcal{T}$, P_Δ is linearly consistent with $G|_\Delta$. That is, P and G are piecewise-linearly consistent if $\left|z_s^d - g^d(p_s)\right| \leq \varepsilon_s$ for every $\Delta \in \mathcal{T}$ and $s \in S_\Delta$. We also say that the sensors in S_Δ are *linearly consistent*; with this, the (decision version of) LCP becomes checking whether S is linearly consistent.

Given an arbitrary localization P, it is easy to build, in a redundant way, a collection of functions consistent with P: A trivial solution can be obtained for example if each sensor is encapsulated in its own triangle. However, the Occam's razor principle (and the common sense) would suggest that (in the absence of other information), a localization that allows one to find a triangulation with fewer triangles is probably a better choice. Hence in this paper, we study the *piecewise-linear consistency-problem* ***(PLCP)*** defined as follows: *Given $(\mathcal{R}, \mathcal{Z}, \mathcal{E})$, find a triangulation $\mathcal{T}$ with the minimum number of triangles, a collection G of piecewise-linear functions with subdomains in $\mathcal{T}$, and a localization P such that P and G are piecewise-linearly consistent.* The LCP is a special case of the PLCP, asking for a triangulation $\mathcal{T}$ with just one triangle.

C. Clustering and Suspension

We emphasize that the triangulation $\mathcal{T}$ is common to all functions in G, i.e., the triangulation does not change over the days. This way, $\mathcal{T}$ provides a natural clustering of the sensors: for any triangle Δ from $\mathcal{T}$, the sensors in S_Δ can be grouped together since their measurements are linearly correlated on any of the D days. Clustering also allows one to do sensor "suspension": some of the sensors within a cluster can be turned off since their measurements can be inferred from the measurements of the others. Interestingly, our distributed algorithm from Section V has a "built-in" method for deciding which sensors transmit their data and which do not.

III. Contributions and Roadmap

Our main contribution is a distributed algorithm for PLCP presented in Section VI. It finds a subdivision of the sensor field into a small number of cells such that inside each cell there is a localization of the sensors that linearly correlates the measurements on each day; the localization and the corresponding linear functions are also output by the algorithm. In other words, the algorithm finds large cells, so that the sensors inside each cell are linearly consistent.

For the main subroutine in our algorithm we developed oracles called EXACT-LCP and APPROX-LCP that determine (exactly and approximately, resp.) whether a set of sensors is linearly consistent, as well as find the corresponding localization and the linear functions. EXACT-LCP is centralized and runs in $D^{O(1)}n^{3D+1}$ time. APPROX-LCP is a distributed algorithm that requires each node to send $O(\log^2 n)$ constant-size messages.

We also provide combinatorial upper and lower bounds on the complexity — the number of connected components, edges and vertices — of the localization regions. These bounds shed light on the complexity of the LCP and are the basis for our efficient approximate solutions.

IV. Related Work

Locating sensors after they have been deployed is important in interpreting and understanding data reported by the sensor. Location conveys information where a particular event occurred, allowing an appropriate action to be taken either in explaining anomalies in sensed data, sending rescue teams for certain military applications or understanding movement in wildlife monitoring. While sensor localization may seem like an easy task by including high-end GPS hardware [26], it may not be applicable for many reasons, such as cost effectiveness, energy efficiency or environmental constraints. As a result, many techniques that trade off accuracy, energy consumption, and cost optimizations have been proposed and evaluated. See for example the survey by Patwari *et al.* [29]. Localization schemes can be grouped into three categories: range-based [4], [8], [11], [16], [21], [24], [27], [28], [30], [31] – which depend on the ranging hardware localization in each sensor; range-free [6], [9], [15], [23], [25], [32] – which listen for radio signals to estimate neighbor locations; and data-based approaches – which use sensed data to locate the sensors. Detailed summary and taxonomy of localization techniques are presented by Amundson et all. [3]. Because our algorithm falls into the latter class, we elaborate on the details of the previous work in the area.

The data-based localization approach, which does not require additional hardware or network communication, relies on correlating data/events to sensor locations [5], [13]. Simultaneous localization and tracking (SLAT) [13], [33] uses the images from the camera to correlate adjacent cameras to the observed object transitions. This technique provides information about the location of each sensor with respect to others and also the direction of the camera on the sensor. The intuition here is that if the sensors observe the same data they are most likely in close proximity. A more generalized approach, similar to the one presented in this paper, was proposed by Baryshnikov and Tan [5]. They first analyze the data, fitting it into some contour that can explain measured data, and the sensor locations are obtained from contour maps where each measured data value lies within some region that best fits the global contours.

Besides addressing sensor localization, our paper expands existing work on *Terrain Simplification* problem [1], [2], [12], [14], [19]. Typically, the input to the problem is a terrain (a graph of a bivariate function) and an error bound; the problem is to find a "simpler" terrain staying within ε from the given one. Agarwal and Suri [1] showed NP-hardness of the problem, and gave an approximation algorithm; a faster approximation algorithm (but with a weaker approximation guarantee) was presented by Agarwal and Desikan [2]. Terrain

simplification has numerous applications in computer graphics, GIS, shape analysis and other fields. Practical heuristics are commonly based either on "edge contraction" – starting from a complex terrain repeatedly reduce the number of patches), or on "triangulation refinement" which proceeds in the opposite way [20], [34]. The problem studied in this paper can be seen as a generalization of terrain simplification: we assume that not only the vertical coordinates of the terrain vertices are not precise, but also that the vertices' xy-coordinates are not known precisely in advance.

Finally, we mention that our optimization problem has similarities to the dictionary design/learning in signal processing/machine learning [18].

V. Solving LCP

A. Bounds and exact algorithm for LCP

It is significant (and somewhat surprising) that the localization region F_s of a sensor s may have multiple connected components (and, consequently, is not convex):

Lemma V.1. *The localization region F_s of sensor s can have $D-1$ connected components.*

Proof: An *hourglass* H in this context is a set of lines for which there exist two convex chains of points $U(H)$ and $L(H)$ such that H consists of the set of lines that are above the points in $L(H)$ and below the points of $U(H)$. See Fig. 3.

For a sensor s, let $B_s^d = R_s \times [z_s^d - \varepsilon_s, z_s^d - \varepsilon_s]$ be the *box of s on day d*; let $\mathcal{B}^d = \{B_1^d, \ldots, B_n^d\}$. We first argue that for any collection of hourglasses $\mathcal{H} = (H^1, \ldots, H^D)$, there exists an instance of our problem such that for the boxes $\mathcal{B}^d$ (the boxes of day d) from the instance, we have $\overline{H_{\mathcal{B}^d}} = H^d$, where $\overline{H_{\mathcal{B}^d}}$ is the set of all lines determined as the intersection of the xy-plane with all linear functions that are linearly consistent with the constrains of day d (i.e., intersect the boxes of day d). To see this, take an hourglass H^d from $\mathcal{H}$. Place a dummy vertex on the first and a dummy vertex on the last edge of $L(H^d)$. Now for each vertex v of $L(H^d)$ (including the dummy vertices) we will have one sensor. The measurement z_s^d of the sensor will be 0. The measurement error of the sensor will be 0. The feasibility region will be the infinite rectangle with one corner at v and the opposite corner at $(-\infty, +\infty)$. Similarly, we will have one sensor per every vertex v of $U(H^d)$; the measurement of the sensor will be 0, the error will be 0, and the feasibility region will have one vertex at v and the opposite vertex at $(+\infty, -\infty)$. Refer to Fig. 3.

We repeat the above construction for each hourglass from $\mathcal{H}$. If a sensor s appears during the construction for hourglass H^d, the measurement error of s on any day $d' \neq d$ is infinitely large; this way the box of s on day d' is surely intersected by any plane from H^d. That is, the measurement errors of s are different on different days: the error on day d is 0, the error on the other days is infinite. This completes the construction of the instance.

We now introduce a sensor s_0 whose allowable region is the whole xy-plane and whose measurement error is infinite. This way $F_{s_0} = \bigcap_d H^d$.

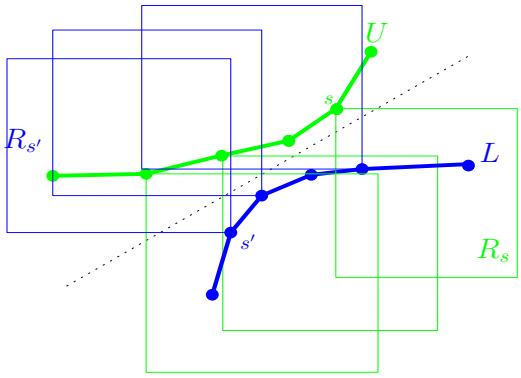

Fig. 3. An hourglass. Lower and upper chains are vertices of allowable regions.

What remains to show is that there exists a set of D hourglasses whose intersection has $D-1$ connected components. To prove this we use geometric duality and the following fact [17]: there exists a collection of disjoint convex polygons $\mathcal{C} = (C^1, \ldots, C^D)$ and a set of lines $\mathcal{L} = \{\ell_1, \ldots, \ell_{D-1}\}$ such that the order in which different lines from $\mathcal{L}$ meet the polygons from $\mathcal{C}$ is different. Let $\mathcal{C}^* = ({C^1}^*, \ldots, {C^D}^*)$ be the hourglasses – duals of the sets in $\mathcal{C}$; let $F = \bigcap_d {C^d}^*$. Let $\mathcal{L}^* = (\ell_1^*, \ldots, \ell_{D-1}^*)$ be the points dual to the lines in $\mathcal{L}$. Since every line in $\mathcal{L}$ intersects all sets in $\mathcal{C}$, every point in $\mathcal{L}^*$ belongs to every hourglass in $\mathcal{C}^*$, and hence – to F. Since a line cannot meet disjoint polygons in two different orders, it is impossible to continuously move from a line ℓ_i to a line $\ell_j, j \neq i$ while always intersecting all sets in $\mathcal{C}$; translated to the dual this means that ℓ_i^* and ℓ_j^* are in different connected components of F. ∎

Lemma V.2. *The total complexity of the localization regions (summed over all sensors) is $D^{O(1)}n^{3D}$. The localization regions can be computed in time $D^{O(1)}n^{3D+1}$.*

Proof: Associate a point $H^* = (a^1, b^1, c^1, \ldots, a^D, b^D, c^D) \in \mathbb{R}^{3D}$ with the set of planes $H = \{h^1, \ldots, h^D\}$, where $h^d = \{(x, y, z) | z = a^d x + b^d y - c^d\}$. (We ignore planes orthogonal to the (x, y)-plane.) Define the set H_s^* as the union of all points H^* for which there is a placement $p_s \in R_s$ for sensor s with $|h^d(p_s) - z_s^d| \leq \varepsilon_s$ for every $1 \leq d \leq D$. Our first step is to compute $\bigcap_s H_s^*$.

Let ∂H_s^* denote the boundary of H_s^*. Every family of $3D$ boundaries meet at $D^{O(1)}$ vertices; hence, the total number of vertices in the arrangement of the boundaries is $D^{O(1)}n^{3D}$. This bounds the complexity of $\bigcap_s H_s^*$. For each vertex of $\bigcap_s H_s^*$, we also have a corresponding set $H = \{h^1, \ldots, h^D\}$ of D linear functions. For a fixed H, the sensor placements do not depend on each other, so each sensor is checked for possible placement, giving a running time of $D^{O(1)}n^{3D+1}$. ∎

B. Approximation and LP formalization of LCP

Since the exact solution to LCP is both centralized and computationally expensive, we present another algorithm, Approx-LCP, overcoming both of these difficulties, at the expense of providing only approximate solutions. Assume each linear

function is parametrized as $h = \{(x, y, z)|z = ax + by - c\}$. Then, solving the LCP is equivalent to finding a set of locations $\{(x_s, y_s)\}_{s=1}^n$ and a collection of coefficients $\{(a^d, b^d, c^d)\}_{d=1}^D$ of the linear functions in H such that $(x_s, y_s) \in R_s$, and $|z_s^d - (a^d x_s + b^d y_s - c^d)| \leq \varepsilon_s$. Unfortunately, the left-hand side of the latter inequality is not linear, since a^d, b^d, x_s and y_s are all variables. Moreover, Lemma V.1 above indicates that F_s may have multiple connected components, suggesting that convex optimization techniques are probably not useful here. To overcome this difficulty, and to enable a practical algorithm, we need to discretize the space of solutions. A naive approach would allow the values a_s^d, b_s^d to be selected from a discrete set Ψ of values, and check feasibility for each assignment of values. This would mean that $|\Psi|^{2D}$ subproblems have to be solved. Instead, we show that a more careful parametrization leads to a solution at which only $|\Psi|^D$ subproblems need to be solved. Each subproblem can be expressed as a linear program (LP), hence leading to an efficient distributed algorithm. Moreover, our experiments (Section VII) indicate that Ψ could be quite small — $|\Psi| = 16$ seems to be sufficient. One should also note that, although the running time is exponential in the number of days D, the experiments also indicate that after very few days the localization errors tend to stabilize, and the benefit from using data from more days is marginal.

Re-paramtrization: We introduce the variables $\alpha^d = a^d/b^d$, $\beta^d = 1/b^d$, and $\gamma^d = c^d/b^d$. Our algorithm iterates over a discrete pre-defined set of possible values of α^d, for $1 \leq d \leq D$. For fixed α^d's, we can write the constraints as an LP:

$$\begin{array}{ll} \textit{Find } (x_s, y_s)_{s=1}^n, (\beta^d, \gamma^d)_{d=1}^D \textit{ s.t} & \\ y_s \geq \alpha^d x_s + \gamma^d - \beta^d(z_s^d + \varepsilon_s), & \forall d, s \\ y_s \leq \alpha^d x_s + \gamma^d - \beta^d(z_s^d - \varepsilon_s), & \forall d, s \\ (x_s, y_s) \in R_s & \forall s \end{array} \quad (1)$$

Geometrically, fixing α^d means fixing the orientation of the line ℓ at which the plane $h = \{(x_s, y_s, z_s)|z = a^d x_s + b^d y_s - c^d\}$ meets the plane $z = 0$.

C. Solving the LP with low communication cost

We assume a multi-hops communication model. Of course, a possible solution is for all sensors to send their measurements to a basestation, and execute the algorithm of Section V-B; however, this requires a large number of transmissions. For example, if sensor s can communicate only with its neighboring sensors, $s-1$ and $s+1$, then the total number of messages sent for collecting the data from all sensors might be as large as $\Omega(Dn)$. We improve this bound by presenting next an algorithm that requires only $O(D \log^2 n)$ messages per node. We assume that an ad-hoc communication network between the sensors is established already. Our bounds could be improved if the diameter of the communication graph is significantly smaller than n. This improvement is straightforward, and is omitted.

Recall that the set of all solutions to (1) is a polytope in $\mathbb{R}^{2D+2n}$. Let $\bar{Q}_s \subseteq \mathbb{R}^{2D+2n}$ denote the set of all points that are solutions to the constraints corresponding to sensor s. Let $Q_s \subseteq \mathbb{R}^{2D}$ denote the set of all 2D-tuples $\vec{\Gamma} = (\beta^1, \gamma^1, \ldots, \beta^D, \gamma^D)$ for which the inequality corresponding to s hold, for some appropriately chosen (x_s, y_s). Note that Q_s is an orthogonal projection of $\bar{Q}_s$ from $\mathbb{R}^{2D+2n}$ onto $\mathbb{R}^{2D}$. Hence Q_s is a polytope as well, and hence can be expressed as intersection of halfspaces in $\mathbb{R}^{2D}$.

Let $\mathcal{Q} = \bigcap_s Q_s$. Note that $\mathcal{Q}$ is non-empty if and only if there is a solution to LCP with the orientations $\alpha_1, \ldots \alpha_D$ fixed. The last observation implies that $\mathcal{Q}$ is also a convex polytope in $\mathbb{R}^{2D}$, and each point on its boundary can be found using the following LP with $2D$ variables:

$$\begin{array}{ll} \max \vec{c} \cdot \vec{\Gamma} & \\ s.t. \quad M \cdot \vec{\Gamma} \geq \vec{b}, & \end{array} \quad (2)$$

where $\vec{\Gamma} = (\beta^1, \gamma^1, \ldots, \beta^D, \gamma^D)$, M is a matrix representation with a row for each constraint of $\mathcal{Q}$, and $\vec{c}, \vec{b}$ are the appropriate vectors. Each sensor s then can find possible placements (there may be more than one) constrained to have the functions determined by $\vec{\Gamma}$. Based on the discussion above, this placement is guaranteed to lie in F_s.

We next describe the details of the algorithm APPROX-LCP for solving (2). Each sensor s is assigned a value, its *weight* w_s, initially set to 1. $\vec{\Gamma}$ is an arbitrary set of $2D$ values. We assume that one sensor serves as the "basestation". An initial solution $\vec{\Gamma}$ is sent from the basestation to all sensors.

During the course of the algorithm, let W denote the total weight of all sensors. We repeat the following procedure, called a *round*, $O(D \log n)$ times.

1) A random subset S' of the sensors send their sets Q_s to the basestation. The decision of which sensors should send is done in a distributed fashion, with each sensor making a random choice to send by "flipping a biased coin", and sending Q_s with probability $9D^2 w_s/W$.
2) The basestation solves the linear programming (1), for (only) the constraints Q_s just received. Let $\vec{\Gamma}$ denote the new partial solution. $\vec{\Gamma}$ is sent to all sensors. (This requires $O(D)$ fixed-size messages from each sensor.)
3) Each sensor s checks if there is a placement p_s of s satisfying all of the constraints posed by s and $\vec{\Gamma}$. If the answer is positive, the algorithm halts — a solution was found. Otherwise, the sensors compute the total weight W' of the sensors s for which there is no placement of s that is consistent with $\vec{\Gamma}$; we call such sensors *violators*. The computation of W' takes 2 messages per sensor.
4) If $W' \leq (1/4D)W$, we call this round a *successful round*, and each violator doubles its weight. (Otherwise, no weights are doubled.)

Following the same argument as in [10], we can prove that the algorithm terminates after $O(D \log n)$ rounds (and if after $O(D \log n)$ iterations no solution is found, then with high probability no linearly consistent localization exists). Indeed, the total weight of constraints, $\sum_s W_s$, increases by a factor of at most $(1 + 1/4D)$ after each successful round. Let $\vec{\Gamma}^*$ be the set of constraints that define the optimal partial solution. If the set of violators is non-empty in a round, then it contains at least one of the constraints of $\vec{\Gamma}^*$. Hence, the weight of at least

one constraint in $\vec{\Gamma}$ is doubled after each successful iteration. After kD successful iterations, the total weight is at most $n(1+1/4D)^k d \leq ne^{k/4}$, and there is a constraint in $\vec{\Gamma}$ whose weight is at least 2^k. Since this quantity is bounded by the total weight of the constraints, we can deduce that $k = O(\log n)$. In each round, the expected weight of the sensors sending their constraints is at most $9D^2/W$, so a standard random-sampling argument [10] shows that the probability of a round being successful is at least $1/2$.

VI. Solving PLCP

Either of the algorithms for the Exact-LCP or Approx-LCP can serve as oracles for a PCLP algorithm. While the discussion in Section IV already shows that finding an exact solution is NP-Hard, we first show a centralized algorithm that provides approximation guarantees.

A. A quality-guaranteed approximation for PLCP

Agarwal and Suri [2] and Agarwal and Desikan [1] presented $O(\log OPT)$-approximation algorithms for terrain simplification, where OPT is the complexity of an optimum (i.e., minimum-complexity) terrain. The main subroutine in the algorithms is an oracle for checking whether there is a plane that intersects a set of vertical line segments. Our algorithm for exact or approximate solution to the LCP provides such an oracle; when plugged into the algorithms from [1], [2], it gives $O(\log OPT)$-approximation algorithms for the PLCP.

B. Quadtree-subdivision in a distributed setting

We describe a quadtree-based distributed algorithm; The algorithm is recursive. Consider a node of the quadtree corresponding to square/quadrant Q. (Initially, Q corresponds to the entire sensor field.)

1) Each sensor s knows its own feasibility region R_s.
2) A leader sensor s_0 is elected, using any standard method of leader election.
3) Sensor s_0 executes Approx-LCP to check linear consistency within Q for sensors whose feasibility regions lie fully inside Q. Only these sensors reply and/or forward the message to other sensors. (We assume that the sensors fully inside Q always form connected components.) As argued in the previous section, this requires sending $O(D \log^2 n')$ messages, where n' is the number of sensors in Q.
4) If Approx-LCP fails to find linear consistency within Q, then s_0 announces it, and announces the splitting of Q into its four quadrants. A new leader is chosen in each subquadrant, and the process repeats.

VII. Experiments

All experiments were performed on a 1.86 GHz Intel Core 2 Duo processor machine with 2 GB RAM, running Debian linux. The code is written in C++, and GLPK (GNU Linear Programming Kit) is used to solve the linear programs. Original sensor locations (x_s, y_s) are generated randomly (uniformly) with $x_s, y_s \in (0, 1000)$. The initial deployment region for each sensor is a random rectangle, centered at the original sensor location, of width/height uniform in $(loc_{err}, 2 \cdot loc_{err})$, for a parameter loc_{err}. The measured values z_s^d for each day d are generated randomly such that the true measurement at location (x_s, y_s) is given by a linear function, $z^d = a^d x_s + b^d y_s - c^d$, and satisfies $0 \leq z^d \leq 1000$. The error bound, ε, in the sensor measurements is assumed to be given, and remains the same for all days and all sensors. The LP (1), as described in Section V, is implemented after discretizing α^d, such that $\alpha^d = \text{cotan}(\theta^d)$, and θ^d iterates over the values in the set of $k_0 + 1$ orientations $\Psi = \{0, \frac{2\pi}{k_0}, 2\frac{2\pi}{k_0}, \ldots, 2\pi\}$, for k_0 to be determined later.

The efficiency of our algorithm depends on the number k_0 of orientations in Ψ. We show experimentally that k_0 need not be too large: We show that if at least one feasible solution exists, then using $k_0 = 32$ is enough to be able to find it, using the $k_0 + 1$ orientations of Ψ. Below, we first study Approx-LCP; then, in Section VII-B, we study the performance of our Quadtree-based algorithm for PLCP.

A. Approx-LCP

Single day ($D = 1$): We fix $k_0 = 32$, and let $orient(feasible)$ be the number of orientations in Ψ for which there exists a feasible solution to the LP of a single day. In the experimental results plotted in Figure 4 we show the average value of $orient(feasible)$ as a function of the number of sensors. Here, the box sizes were fixed, using the choice of parameters $loc_{err} = 50$ and $\varepsilon = 50$. We see that with $k_0 = 32$ there were always enough orientations for multiple feasible solutions and that increasing the number of sensors has little effect on the number of feasible solutions, once there are at least 100 sensors. Then, in Figure 5, we show how the average value of $orient(feasible)$ varies with the box-size parameters loc_{err} and ε (with $loc_{err} = \varepsilon$), for a fixed number, 50, of sensors.

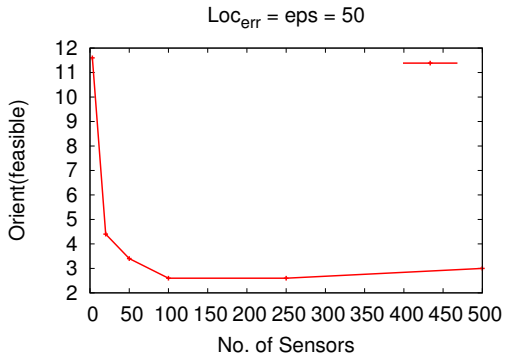

Fig. 4. $orient(feasible)$ vs. the number of sensors. Here, $loc_{err} = \varepsilon = 50$.

Multiple days ($D > 1$): Next, we study how localization improves with the number, D, of days over which data is collected. Figure 6 shows experimentally how the average area of the localization regions varies with D. As expected, as D increases, the areas of localization regions decrease. We note that the decreases occur in steps.

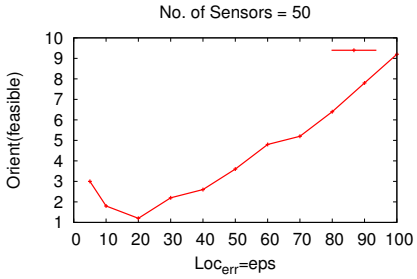

Fig. 5. $orient(feasible)$ vs. box size parameters $loc_{err} = \varepsilon$.

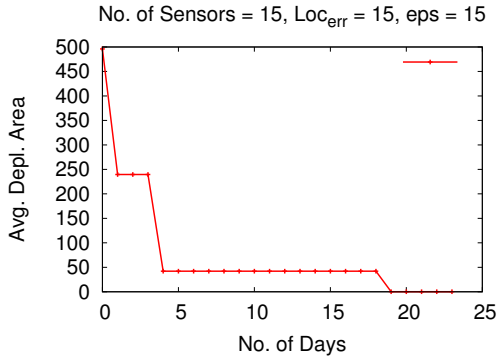

Fig. 6. Average area of localization regions vs. number of days.

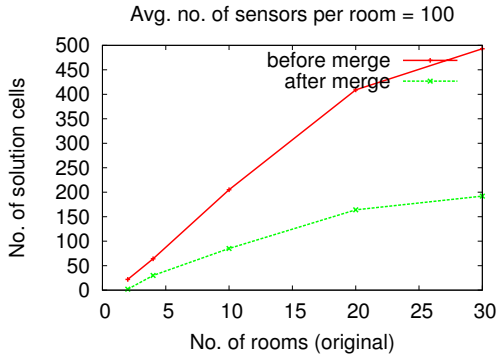

Fig. 7. No. of solution cells vs. no. of rooms.

B. Multiple rooms: Quadtree approach

Here we study the behavior of the PLCP algorithm, used in APPROX-LCP as an oracle. We generate random rectangular rooms via a Binary Space Partition (BSP) of the 1000×1000 sensor region. For each room, the sensed measurement field is assumed to be linear for each day (independent and possibly different from other rooms), and comes from a set of 16 discrete major orientations, for simplicity of formulating the LP. After computing the quadtree solution, we post-process the solution by combining pairs of quadtree cells. Only neighboring cells that merge to form a rectangle are considered, and merging takes place if and only if there exists a deployment of each sensor lying within the rectangle that satisfies a linear measurement field for each day. We experiment with different numbers of rooms, while keeping the average number of sensors the same (100) for each room. Figure 7 shows the variation of the number of cells in the solution with the increasing number of rooms. The number of cells considerably decreases after post-processing. For this experiment, $loc_{err} = \varepsilon = 5$ is kept small, in order that we can analyze also the quality of assignment of sensors to the cells.

We quantify the quality of sensor assignment (or classification) using the following four metrics: the fraction of sensor pairs that are *(typeA)* assigned to the same cell and also share the same room, *(typeB)* assigned to the same cell but do not share the same room, *(typeC)* assigned to different cells, which shared the same room, *(typeD)* assigned to different cells, which also did not share the same room.

For good assignment/classification, *typeA* and *typeD* should be close to 1, while *typeB* and *typeC* should be close to 0. Figure 8 shows the four metrics for the quadtree cell subdivision. All four of the metrics look reasonable for up to 30 rooms. The fraction of sensors that were not classified are also shown. These are the ones whose initial deployment region overlapped cell boundaries. The left plot shows the results for the quadtree subdivision alone; the right plot shows the results after the merging of neighboring cells done in the post-processing. Figure 9 shows a 4-room example illustrating the result of the quadtree subdivision/classification as well as the result after the post-processing merging of quadtree cells, which yields a reasonably good approximation to the original set of rooms.

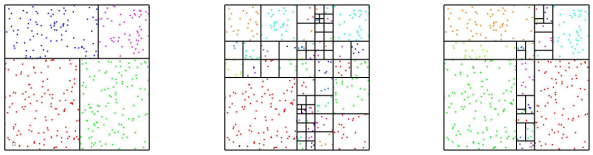

Fig. 9. Examples of results. From left to right: Original 4 random (BSP) rooms; resulting subdivision/classification; the post-processed subdivision/classification after merging neighboring cells.

VIII. CONCLUSION

We have given a distributed data-driven solution to localization, clustering, and sensor suspension problems. While we have assumed here that the sensor measurements come from a linear or piecewise-linear field (function), our methods extend to the piecewise-algebraic case as well. Theoretically, we obtained a $\log OPT$-approximation centralized algorithm for the *Piecewise linear consistency problem* (***LCP***), which is NP-hard even for $D = 1$, matching the best bounds known for the related terrain approximation problem [1], [2] (a special case of our problem).

The basic procedure of our terrain simplification approach—checking for linear consistency—can be plugged into any of the existing practical means of performing terrain simplification

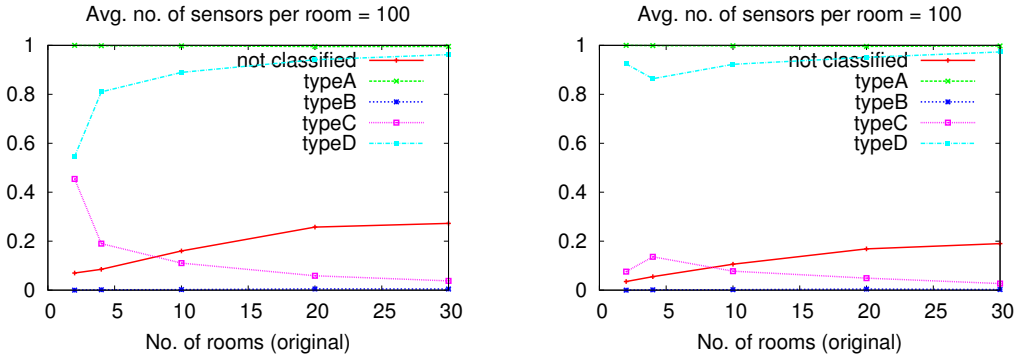

Fig. 8. Classification error vs. no. of rooms. Top: Quadtree subdivision. Bottom: After merging neighboring cells.

(e.g., edge contraction, incremental insertion, etc). Here, we used a quadtree-based approach for simplicity.

One direction for future research is to devise an incremental version of our algorithms, so that updates to the localization can be done as each new day's measurements are obtained.

ACKNOWLEDGEMENTS

We wish to thank Swaminathan Sankararaman for his help in preparing this manuscript. We thank Jie Gao and Danny Halperin for helpful discussions. We also thank anonymous reviewers for providing productive comments and suggestions. P. K. Agarwal is supported by NSF under grants CNS-05-40347, CCF-06 -35000, IIS-07-13498, and CCF-09-40671, by ARO grants W911NF-07-1-0376 and W911NF-08-1-0452, by an NIH grant 1P50-GM-08183-01, and by a grant from the U.S.–Israel Binational Science Foundation. A. Efrat is supported by the NSF by NSF CAREER grant 0348000 and NSF grant CNS-1017714. J. Mitchell is partially supported by the National Science Foundation (CCF-0729019,CCF-1018388) and by NASA Ames. V. Polishchuk is supported by the Academy of Finland grant 138520.

REFERENCES

[1] P. K. Agarwal and P. K. Desikan, "An efficient algorithm for terrain simplification," in *SODA*, 1997.

[2] P. K. Agarwal and S. Suri, "Surface approximation and geometric partitions," *SIAM J. Computing* 27(4):1016–1035, 1998.

[3] I. Amundson and X. Koutsoukos, "A Survey on Localization for Mobile Wireless Sensor Networks" *2nd International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments (MELT)* 2009.

[4] P. Bahl and V. N. Padmanabhan, "Radar: an in-building rf-based user location and tracking system," vol. 2, 2000, pp. 775–784.

[5] Y. M. Baryshnikov and J. Tan, "Localization for anchoritic sensor networks," in *DCOSS*, 2007.

[6] A. Basu, J. Gao, J. S. B. Mitchell, and G. Sabhnani, "Distributed localization using noisy distance and angle information," in *MobiHoc*, 2006, pp. 262–273.

[7] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, Springer, Berlin, 2008.

[8] M. Broxton, J. Lifton, and J. Paradiso, "Localizing a sensor network via collaborative processing of global stimuli," in *Proc. European Conference on Wireless Sensor Networks*, 2005.

[9] N. Bulusu, J. Heidemann, and D. Estrin, "Gps-less low-cost outdoor localization for very small devices," *Personal Communications, IEEE* vol. 7, no. 5, pp. 28–34, 2000.

[10] K. Clarkson, "Las Vegas algorithms for linear and integer programming when the dimension is small" *J. ACM* vol. 42, 1995, 488–499.

[11] P. Corke, R. Peterson, and D. Rus, "Networked robots: Flying robot navigation using a sensor net," 2005, pp. 234–243.

[12] W. Evans, D. Kirkpatrick, and G. Townsend, "Right-triangulated irregular networks," *Algorithmica*, vol. 30, pp. 264–286, 2001.

[13] S. Funiak, C. Guestrin, M. Paskin, and R. Sukthankar, "Distributed localization of networked cameras," in *IPSN*, 2006.

[14] C. Gray, M. Löffler, and R. I. Silveira, "Smoothing imprecise 1.5D terrains," in *Workshp on Approximation and Online Algorithms*, 2008.

[15] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks," in *MobiCom*, 2003, pp. 81–95.

[16] L. Hu and D. Evans, "Localization for mobile sensor networks," in *MobiCom*, 2004, pp. 45–57.

[17] M. Katchalski, T. Lewis, and J. Zaks, "Geometric permutations for convex sets," *Discrete Math.*, vol. 54, pp. 271–284, 1985.

[18] A. Krause and V Cevher, "Submodular Dictionary Selection for Sparse Representation," in *Proc. ICML*, 2010.

[19] M. van Kreveld and R. I. Silveira, "Embedding rivers in polyhedral terrains," in *Proc. SoCG*, 2009.

[20] J. Li, S. Hu, X. Ye, and Z. Li, "A hybrid simplification algorithm for large scale terrain," in *8th Intl. Conf. on Signal Processing*, 2006.

[21] K. Lorincz and M. Welsh, "Motetrack: a robust, decentralized approach to rf-based location tracking," *Personal Ubiquitous Comput.*, 2007.

[22] J. Matoušek, M. Sharir, and E. Welzl. "A subexponential bound for linear programming," *Algorithmica*, 16:498–516, 1996.

[23] R. Nagpal, H. Shrobe, and J. Bachrach, "Organizing a global coordinate system from local information on a sensor network," *IPSN*, 2003.

[24] D. Niculescu and B. Nath, "Ad hoc positioning system (APS) using AOA," vol. 3, 2003, pp. 1734–1743 vol.3.

[25] D. Niculescu and B. Nath, "Dv based positioning in ad hoc networks," *Journal of Telecommunication Systems*, 2003.

[26] B. Parkinson and J. Spilker, "Global Positioning System: theory and applications," *Progress in Aeronautics and Astronautics*, vol. 163, 1996.

[27] P. N. Pathirana, H. Balakrishnan, E. Demaine, and S. Teller, "Mobile-assisted topology generation for auto-localization in sensor networks," in *Infocom*, 2005.

[28] P. N. Pathirana, N. Bulusu, A. V. Savkin, and S. Jha, "Node localization using mobile robots in delay-tolerant sensor networks," *Mobile Computing, IEEE Trans. on*, vol. 4, no. 3, pp. 285–296, 2005.

[29] N. Patwari, J.N. Ash, S. Kyperountas,, III Hero, R.L. Moses and N.S. Correal, Locating the nodes: cooperative localization in wireless sensor networks, *IEEE Signal Processing Magazine* 22:(2005) 54-69.

[30] A. Savvides, C.-C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *MobiCom*, 2001.

[31] M. Sichitiu, V. Ramadurai, E. Demaine, and S. Teller, "Localization of wireless sensor networks with a mobile beacon," in *Proc. MASS*, 2004.

[32] R. Stoleru, T. He, J. A. Stankovic, and D. Luebke, "A high-accuracy, low-cost localization system for WSNs," in *SenSys*, 2005, pp. 13–26.

[33] C. Taylor, A. Rahimi, J. Bachrach, H. Shrobe and A. Grue, Simultaneous Localization, Calibration, and Tracking in an ad Hoc Sensor Network *IPSN* 2006.

[34] A. Xu, S. Sun, and K. Xu, "Texture information driven triangle mesh simplification," in *Computer Graphics and Imaging*, 2005.