Resolution-Directed Optimization-based Distributed Sensing

Richard J. Vaccaro* Petros Boufounos† Mouhacine Benosman‡

Abstract

In this paper we study the problem of optimal zone coverage, using distributed sensing, i.e. a group of collaborating sensors. We formulate the problem as an optimization problem with time-varying cost function. We examine the case where a group of elevated imaging sensors look down to and form the map of a 2-dimensional environment at a pre-specified resolution. The sensors solve an optimization problem that attempts to optimize a time-varying cost function. The cost at any time instance measures the distance between the desired resolution function and the achieved resolution until the previous time instant. We discuss the numerical implementation challenges of this approach and demonstrate its performance on a numerical example.

1 Introduction

As distributed sensing systems become increasingly capable and prevalent, distributed control, positioning and path planning become increasingly important. For such control to be successful it is important that it takes the physical properties of sensor appropriately into account, both in terms of mobility and in terms of sensing properties. Furthermore, it is important that the positioning and path planning of the sensors is matched to the application goals.

In this paper we examine the use of mobile sensors for mapping the environment at a pre-defined resolution. The goal of the sensors is to cover a region, taking snapshots as they move around. The desired resolution is achieved by combining multiple snapshots, possibly taken by different sensors, each with different resolution characteristics. In addition to it path, at each step each sensor optimizes the parameters of its imaging system, which determine the area covered and the resulting resolution. For each snapshot, the sensors may acquire a larger area at a low resolution or a smaller area at a high resolution, according to the physics and capabilities of their imaging systems.

Distributed algorithms for configuring mobile sensors to cover a specified region have been developed over the past decade, e.g. [1–4]. In earlier work sensors were assumed to have infinite range or to be bounded range isotropic sensors [5]. In more recent work, anisotropic sensors with a bounded footprint are considered [6–8]. The approach taken in these references is to model a 2-dimensional environment as a polygon, possibly containing polygonal obstacles. A fixed objective function is defined and optimized, which is the joint probability of detection of interesting objects. The objective function is formulated using an a priori fixed density function that represents the importance of each point in the environment. The gradient of the objective function with respect to the parameters of the sensors provides a control direction to move the sensors towards important regions. These algorithms are mostly useful in applications where the sensors need to monitor a region of interest, especially areas where targets might be present, but not necessarily map the whole area.

In terms of the formulation, this paper differs significantly from previous work in several ways. First, the sensor model is an elevated imaging sensor looking down to a 2-dimensional environment. Second, the objective is to map the environment, i.e., to provide image resolution over the environment that achieves a specified value at each point in the environment. We use a subadditive function to model the resolution when images of overlapping sensors are combined. The third, and most important difference, is that the objective function is time varying. At each time index t, the objective function is a measure of the difference between the desired resolution and the resolution achieved by all the sensors up to time $t-1$. This model enables the mapping functionality we desire, instead of the monitoring functionality addressed in the existing literature. Although our problem formulation allows for the presence of polygonal obstacles as in [9], this paper considers only an obstacle-free environment.

While the sensor model we describe is motivated

*University of Rhode Island, Electrical, Computer, and Biomedical Engineering, E-mail: vaccaro@ele.uri.edu

†Mitsubishi Electric Research Laboratories, 201 Broadway, Cambridge, MA 02139, USA, E-mail:petrosb@merl.com

‡Mitsubishi Electric Research Laboratories, 201 Broadway, Cambridge, MA 02139, USA, E-mail: m_benosman@ieee.org

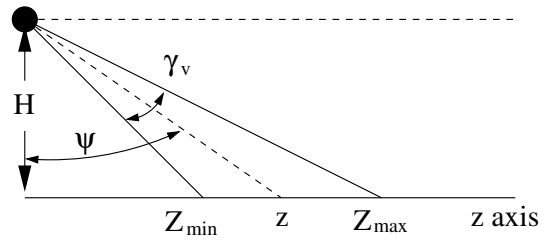

Figure 1: Side view in sensor coordinates. The point z is located on the line bisecting the angle γ_v.

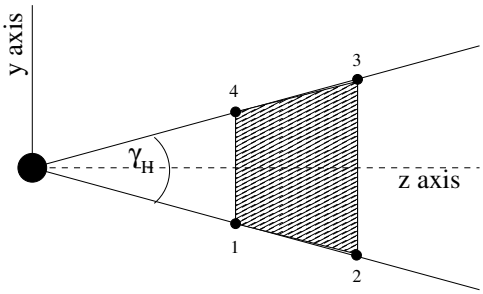

Figure 2: Sensor footprint in sensor coordinates (top view). The footprint is the shaded region, which extends from Z_{min} to Z_{max}. The footprint is the polygon defined by the four labeled vertices.

by optical sensors, very similar models apply to radar, lidar, and Synthetic Aperture Radar (SAR) models. In general, these sensors have a fixed angular resolution and field of view. This, in turn, determines the spatial resolution and the spatial coverage of the sensor as a function of the distance of the scene the sensor is acquiring. Multiple snapshots of a scene can be fused using well-established approaches, which we don't explore here. For examples, see [10] and references within. We only assume that the resolution of the fused image is a sublinear combination of the resolution achievable with individual snapshots.

In the next section we provide a precise formulation of the sensing problem. The corresponding optimization problem is presented in Sec. 3. The implementation of the proposed optimization-based sensing is presented in Sec. 4. Section 5 is dedicated to our numerical results, and, finally, the paper ends with a discussion in Sec 6.

2 Problem Formulation

2.1 Sensor Model In this paper we consider sensors sensing and mapping an environment Q of polygonal shape, which is a subset of the xy plane. We assume the environment is discretized and use $q \in Q$ to denote

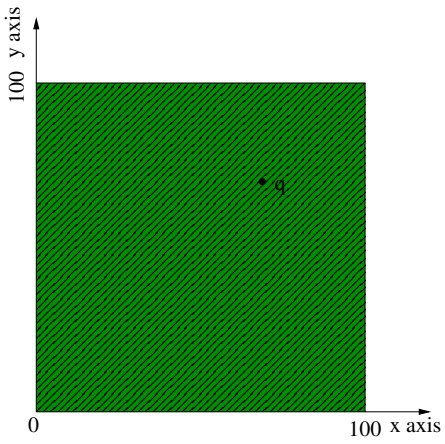

Figure 3: An example environment polygon, Q, of square shape and dimensions 100×100, also used for our experiments. An arbitrary point in the environment is labeled $q \in Q$.

points. For the examples in this paper, Q is the 100×100 square region shown in Fig.3.

We assume the sensors are elevated and roaming above the environment, imaging the ground. Each sensor plans its own path and decides on its orientation and its imaging parameters. Specifically, at each step the sensor selects its coordinates (c_x, c_y) in the xy plane, its azimuth angle, θ, i.e., the rotation of its imaging system with respect to the environment's x axis, and the vertical orientation of its imaging system, ψ, with respect to the axis normal to the environment. The imaging system has a fixed angular field-of-view (FOV) in the horizontal and vertical direction, denoted using γ_h and γ_v, respectively. This is fixed by the imaging system hardware. Furthermore, the sensor is position at height H above the ground, which is also fixed throughout the path planning, even though variable sensor height can be incorporated in the model. We assume there is no roll in the sensor flight, although it can also be integrated into the model in a straightforward manner.

Figures 1 and 2 illustrate a side and top view of the sensor geometry, respectively. Given the sensor parameters, the sensor acquires snapshots of its FOV footprint on the environment, indicated using the shaded area in Fig. 2, lying between Z_{min} an Z_{max} in Fig. 1. All the snapshots from all the sensors are then combined to form the final map of the environment.

The full set of sensor parameters, together with indicative values used in simulations in this paper, is described in Table 1. For the simulations in this paper we assume all sensors have the same non-tunable sensing parameters, i.e., same angular field of view, angular

Variable	Description	Value
H	Height of sensor	30[1]
γ_h	Horizontal angular width	20°
γ_v	Vertical angular width	2°
(c_x, c_y)	(x, y) location of sensor	arbitrary[2]
ψ	90 - declination angle	arbitrary
θ	Azimuth angle of sensor	arbitrary

Table 1: Definition of variables.

sensing resolution and height. The path planning problem each sensor solves involves determining its location (c_x, c_y), azimuth angle θ and vertical orientation ψ.

It is often convenient to describe the environment in coordinates relative to the sensor orientation, i.e., in the sensor's reference frame. In this case, the environment lies in the zy plane, which is a translated and rotated version of the xy plane according to the sensor position and orientation. When $\theta = 0$, the z-axis of the sensor is aligned with the x-axis of the global coordinate system. As described in Fig. 1, a given sensor is located at height H above the origin of the zy plane. The angle ψ, capturing the vertical orientation of the (FOV) of the sensor, is measured with respect to the normal to the zy plane.

Thus, by tuning the vertical orientation of the sensor, it is possible to image areas of different size in the environment, as depicted in Figs. 1 and 2. In particular, the labeled ranges in Fig. 1 are equal to

$$\begin{aligned} Z_{max} &= H\tan(\psi + \gamma_v/2) \\ Z_{min} &= H\tan(\psi - \gamma_v/2) \\ z &= H\tan(\psi) \end{aligned} \tag{2.1}$$

In sensor coordinates, the footprint vertices (z_k, y_k) corresponding to vertex k in Fig. 2 are given using:

$$\begin{bmatrix} z_1 & y_1 \\ z_2 & y_2 \\ z_3 & y_3 \\ z_4 & y_4 \end{bmatrix} = \begin{bmatrix} Z_{min} & -Z_{min}\sin(\gamma_h/2) \\ Z_{max} & -Z_{max}\sin(\gamma_h/2) \\ Z_{max} & Z_{max}\sin(\gamma_h/2) \\ Z_{min} & Z_{min}\sin(\gamma_h/2) \end{bmatrix} \tag{2.2}$$

We use $\mathbf{S}(\theta)$ to denote the rotation matrix

$$\mathbf{S}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}. \tag{2.3}$$

Thus, the global coordinates (x, y) of a point (z, y) in the sensor footprint are obtained by rotating the point by the angle θ and translating it by the camera location (c_x, c_y):

$$\begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{S}(\theta)\begin{bmatrix} z \\ y \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}. \tag{2.4}$$

Using (2.4), the four vertices in (2.2) defining the sensor footprint may be mapped into global coordinates.

In our path planning problem, the sensor footprint is determined by four variables: (c_x, c_y, θ, ψ). The first two, c_x, c_y, describe the sensor position above the plane of the environment and are determined by the motion of the sensor. The last two parameters, θ, ψ, describe the horizontal and vertical angular orientation of the imaging system on the sensor. Thus, it is assumed that the location parameters will be updated on a "slow" time scale because they correspond to the physical motion the sensor. On the other hand, it is assumed that the angular variables will be updated on a "fast" time scale because the sensor can change these angles very quickly simply by moving its imaging system around its axis. In the remainder of this paper, all variables associated with the ith sensor will have a superscript of i.

2.2 Resolution of Overlapping Snapshots Most imaging systems can be accurately described as having a fixed angular resolution or, equivalently, a fixed number of pixels to cover its field of view. Thus, we define the resolution of a sensor as the total number of sensor pixels divided by the footprint area.

Assuming that the sensor has fixed horizontal and vertical angular FOV, as defined in Table 1, the resolution R_i of sensor i is inversely proportional to the squared horizontal range z^2 (see Fig. 1):

$$R_i = \frac{K_i}{z^2} \tag{2.5}$$

where K_i is a constant that depends on the physical characteristics of sensor i. In this paper we assume that all sensors have the same value of K_i. We also make the approximation that the resolution at *all* points in the footprint of the sensor i is the value given by (2.5). This approximation is valid when the angular FOV of the sensor is small, i.e., when $\sin(\gamma) \approx \gamma$.

At every time step each sensor is taking a snapshot of its FOV, according to its parameters. All these snapshots are eventually fused to generate a large map of the environment at high resolution, using one of a number of possible fusion and super-resolution algorithms, e.g., see [10] and references within.

In order to be able to direct the sensor parameter optimization, we need a resolution model for the resulting map. Our model explicitly models the resolution

[1] The units are arbitrary. The same units are used to describe the environment polygon.

[2] The sensor-location coordinates are given in the same units as the sensor height.

achieved at each point in the environment, assuming the point has been acquired by n snapshots—i.e., has been part of their footprint—each with resolution R_i pixels per unit area.

When fusing multiple snapshots the resulting image may not have lower resolution than the maximum resolution of any snapshot. Otherwise, we could just use the snapshot with the higher resolution and obtain a better result without fusion. Thus, this is a lower bound on the achievable resolution of the fused image. Ideally, the resolution of the fused image would be the sum of the resolutions of each individual snapshot. However, due to uncertainties and imperfections of the sensor fusion process, this is unfortunately unrealistic. Still, it does provide an upper bound on the achievable resolution.

The actual overall resolution will be somewhere between these bounds. That is, if

$$\mathbf{R} = \begin{bmatrix} R_1 & R_2 & \cdots & R_N \end{bmatrix} \tag{2.6}$$

is a vector of the resolutions achieved by N sensors, the overall resolution res($\mathbf{R}$) obtained at points in the intersection of the sensor footprints must satisfy the following inequalities

$$\max_i R_i \leq \text{res}(\mathbf{R}) \leq R_1 + \cdots + R_N. \tag{2.7}$$

One function that satisfies this property is the l_p norm of the vector $\mathbf{R}$, $1 < p < \infty$,

$$\|\mathbf{R}\|_p \stackrel{\text{def}}{=} \left(R_1^p + \cdots R_N^p\right)^{1/p}. \tag{2.8}$$

When $p = 1$, the l_p norm equals the upper bound in (2.7). When $p = \infty$, the l_p norm equals the lower bound in (2.7). Thus, in this paper, we use the l_p norm, $1 < p < \infty$, of the vector of individual resolutions, as a subadditive model for the resolution obtained by fusing the snapshots of overlapping sensors. Note, however, that this is not a critical assumption in our development. Other subadditive smooth functions can be substituted.

3 Objective Function and Optimization

In order to direct the sensing process, we define a desired resolution function, $\phi_d(q)$, for each point $q \in Q$ in the environment. This is an operator-defined global function that the sensors try to achieve through the distributed optimization of their parameters.

To formulate the optimization it is often convenient to aggregate the system variables using vector notation. Thus, we use $\mathbf{x}_t$ to denote the vector of the location variables of all of the sensors at time t, and $\boldsymbol{\psi}_t$ and $\boldsymbol{\theta}_t$ the vectors of the vertical and horizontal angular orientations, respectively, of the sensors at time t. We use R^i to denote the resolution provided by the ith sensor at all points in its footprint F_i, which is uniquely determined by the corresponding sensor variables $(cx^i, cy^i, \theta^i, \psi^i)$.

The resolution is inversely proportional to the area of the imaged scene. Thus, it is equal to

$$R^i(cx^i, cy^i, \theta^i, \psi^i, q) = \begin{cases} \dfrac{K}{H^2[1+\tan^2(\psi^i)]}, & q \in F_i(cx^i, cy^i, \theta^i, \psi^i) \\ 0, & \text{o.w.} \end{cases} \tag{3.9}$$

where K is a sensor constant that depends on the total number of pixels captured by the sensor. It is assumed that all of the sensors have the same value of K, in which case its value is unimportant for the optimization that follows.

At any time t the objective function we wish to minimize is a measure of the difference between the desired resolution and the achieved resolution up to time t. This is measured by the following function:

$$G_t(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\psi}) = \int_Q f\left(\phi_d(q) - [\phi_{t-1}^p(q) + \textstyle\sum_i \left(R^i(cx^i, cy^i, \theta^i, \psi^i, q)\right)^p]^{1/p}\right)^2 \mathrm{d}q, \tag{3.10}$$

where $\phi_{t-1}(q)$ is the resolution achieved by the sensors up to time $t-1$, p defines the norm used to model a subadditive combination of overlapping sensors (see Section 2.2), and $f(\cdot)$ is a loss function such as

$$f(x) = |x|^\gamma \tag{3.11}$$

for some $\gamma \geq 1$. If the value $\gamma = 2$ is used, the square of the difference between the desired and achieved resolutions in (3.10) will tend to ignore small errors and greatly emphasize large errors. On the other hand, if $\gamma = 1$ is used, small errors will have a greater influence than with $\gamma = 2$. In the simulations that follow we use $\gamma = 1.75$. By definition, the initial achieved resolution map $\phi_0(q)$ is identically zero.

A gradient-based optimization approach is described by the following initialization and iteration. At each time instant, a complete gradient-based minimization with respect to the angle parameters of the sensors is performed. However, sensor positions are updated using only a single gradient step. The reason is that once the sensors have moved and acquired new data, the objective function has changed.

Initialization Given the desired resolution map $\phi_d(q)$, and $\mathbf{x}_0$, a vector of initial sensor locations, find the initial sensor angles by the following optimization

$$\boldsymbol{\theta}_0, \boldsymbol{\psi}_0 = \arg\min_{\boldsymbol{\theta}, \boldsymbol{\psi}} G_0(\mathbf{x}_0, \boldsymbol{\theta}, \boldsymbol{\psi}). \tag{3.12}$$

Calculate the initial position gradient, $\mathbf{g}_0$, which is the gradient with respect to $\mathbf{x}$ of $G_0(\mathbf{x}, \boldsymbol{\theta}_0, \boldsymbol{\psi}_0)$ evaluated at $\mathbf{x}_0$. This gradient is used in Step 2 of the iteration shown below.

Iteration, $t = 1, 2, \cdots$

1. Acquire images from all sensors and update the achieved resolution map: $\phi_j(q, \mathbf{x}_{t-1}, \boldsymbol{\theta}_{t-1}, \boldsymbol{\psi}_{t-1})$ is equal to
(3.13)
$\{\phi^p_{t-1}(q) + [\sum_i (R^i(cx^i_{t-1}, cy^i_{t-1}, \theta^i_{t-1}, \psi^i_{t-1}, q))^p]\}^{1/p}$

2. Take a position step. That is, take a single step in the direction of the negative position gradient

$$\mathbf{x}_t = \mathbf{x}_{t-1} - \alpha \mathbf{g}_{t-1}, \tag{3.14}$$

where α is a step size, and $\mathbf{g}_{t-1}$ is the position gradient from the previous step evaluated at $\mathbf{x}_{t-1}$. The value of α is found by a line search.

3. Update the sensor angular parameters and the position gradient:

$$\begin{aligned} \boldsymbol{\theta}_t, \boldsymbol{\psi}_t &= \arg\min_{\boldsymbol{\theta}, \boldsymbol{\psi}} G_t(\mathbf{x}_t, \boldsymbol{\theta}, \boldsymbol{\psi}) \\ \mathbf{g}_t &= \nabla_{\mathbf{x}} G_t(\mathbf{x}, \boldsymbol{\theta}_t, \boldsymbol{\psi}_t) \ \text{ evaluated at } \mathbf{x}_t. \end{aligned} \tag{3.15}$$

4 Implementation

The numerical implementation of the algorithm uses tools initially developed for computer graphics. The environment is modeled as a polygon in a 2-dimensional plane, as are the sensor footprints. Obstacles may also be modeled as polygons. The data structure for a polygon is simply a list of its vertices in counterclockwise order. A public-domain Matlab class, Polygon.m, has been developed for representing manipulating polygons [11]. This class includes a function to find the grid points that are inside the footprints of the sensors as well as a function to find the grid points in the intersection of two or more polygons.

The function $G_j(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\psi})$ defined in (3.10) is an integral over the environment. The value of this integral is computed numerically at a finite grid of points. The grid spacing in the x and y directions is called δ. Fig. 4 shows two sensor footprints, each with a vertical angle $\psi = 25°$. The footprint in the lower left shows grid points with $\delta = 0.5$ while the footprint in the upper right shows grid points with $\delta = 0.25$. The smaller value of δ was used to generate the simulation results.

4.1 Gradient Calculation The objective function G defined in (3.10) is the integral of a function which is discontinuous with respect to q as well as with respect to the sensor parameters. The discontinuities with respect to the sensor parameters are due to the fact that the achieved resolution of each sensor is constant within the sensor footprint and zero outside the footprint. When any sensor parameter changes, the footprint changes. Points that were outside the original footprint with a function value of zero may now be inside the footprint with a positive function value, and vice versa. If the integrand of G were continuous, it would be advantageous to move the derivative operator inside the integral and calculate the derivative analytically. With a discontinuous integrand, moving the derivative inside the integral requires a special calculation for the points of discontinuity [7]. Using the existing polygon tools and a sampled grid of points, it would be difficult to identify the set of discontinuity points, especially for the case of overlapping sensor footprints considered here. Thus, we use a completely numerical approach to computing the gradient by calculating the change in the value of G divided by an incremental change in a single parameter. Because the function G is evaluated on a sampled grid of points, it is necessary to use parameter increments that cause a change in the function value.

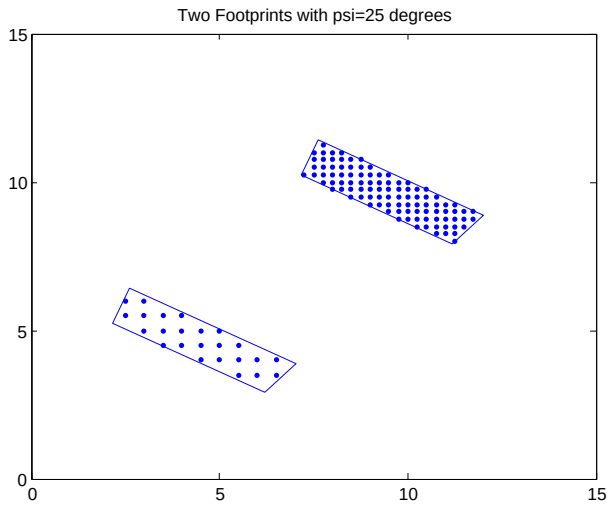

Figure 4: Sensor footprints with $\psi = 25°$ showing grid points with two different values of grid spacing δ. The footprint in the lower left shows grid points with $\delta = 0.5$ while the footprint in the upper right shows grid points with $\delta = 0.25$.

4.2 Parameter Increments To numerically calculate the derivative with respect to the angle ψ^i of the ith sensor, we must choose a small increment, $\mathrm{d}\psi^i$ to make the sector midpoint, z, change by an amount that

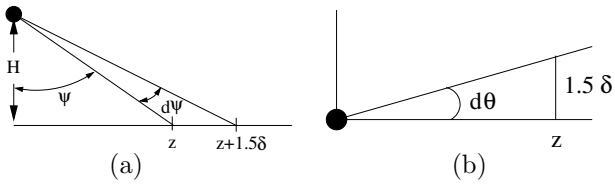

Figure 5: (a) $d\psi$ is the smallest change in ψ that can be used in the gradient calculation. It results in the point z moving by an amount 1.5δ, where δ is the grid spacing.(b) $d\theta$ is the smallest change in θ that can be used in the gradient calculation. It results in the point z moving by an amount 1.5δ, where δ is the grid spacing.

is large enough to move the sensor footprint at least one grid point. We choose the value 1.5δ, as shown in Fig. 5(a). The result is

$$d\psi^i = \tan^{-1}\left(\frac{1.5\delta}{H} + \tan\psi^i\right) - \psi^i \tag{4.16}$$

The partial derivative with respect to the vertical angle of the ith sensor is then computed as

$$\frac{\partial G}{\partial \psi^i} \approx \frac{G(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\psi} + d\boldsymbol{\psi}) - G(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\psi})}{d\psi^i}, \tag{4.17}$$

where $d\boldsymbol{\psi}$ is zero except element i, which equals $d\psi^i$.

A similar calculation is done for the azimuth angles θ^i. Specifically, the increment $d\theta^i$ is calculated to move the point z by an amount 1.5δ, as shown in Fig. 5(b).

Recall that $z = H\tan\psi$. Then the result is

$$d\theta^i = \tan^{-1}\left(\frac{1.5\delta}{H\tan\psi^i}\right). \tag{4.18}$$

Note that both angle increments, $d\psi^i$ and $d\theta^i$, are functions of δ as well as of the current value of ψ^i.

5 Simulation Results

In our simulations we consider a scenario in which four cooperating sensors are dispatched to observe an environment and provide the desired resolution shown in Fig. 6(a). The high-resolution region in the center has a value of 5.5, which is the resolution provided by a single sensor when $\psi = 25°$. The desired background resolution has a value of 2.4, which is the resolution provided by a single sensor when $\psi = 54.8°$. Thus, in principle, the whole region of interest can be mapped using non-overlapping snapshots, appropriately directed. The algorithm presented in Section 3 was initialized and run for 100 time steps. The initial footprints of the four sensors are shown in Fig. 6(b).

The resolution values for overlapping sensor footprints were combined using the l_2 norm; i.e. $p = 2$ in (2.8). The achieved resolution at $t = 100$ is shown in Fig. 6(c), which is very close to the desired resolution shown on Fig. 6(a). A plot of the evolution of the cost function G as a function of time is shown in Fig. 6(d). It can be seen that the algorithm converges to a nonzero limiting value at $t = 100$.

A simple calculation, using nonoverlapping sensor footprints, can be used to bound the minimum number of time steps needed to achieve a desired resolution map. using nonoverlapping sensor footprints. This number can be compared to that used by the proposed algorithm. Suppose that the desired resolution map consists of two constant-resolution regions, as in the simulation example. Let A_1 be the area of the low-resolution region of Fig. 6(a) and let A_2 be the area of the high resolution region. If the ψ angle of a given sensor is chosen to be 54.8°, the resolution of the resulting footprint will be 2.36, the desired background resolution in Fig. 6(a). With $\psi = 54.8°$, the area of the sensor footprint is A_1. Similarly, the area of the sensor footprint whose resolution matches the high-resolution region in Fig. 6(a) is A_2. A lower bound on the number of time steps needed by $n = 4$ sensors to achieve the desired resolution map with nonoverlapping sensor footprints is

$$N = \frac{A_1}{na_1} + \frac{A_2}{na_2}. \tag{5.19}$$

Note that this is a lower bound, that is not necessarily achievable due to the shape of the sensor footprints. Even if the footprints allow for perfect non-overlapping coverage, achieving this bound requires central planning of the sensors' path and orientation. Still, the simulations show that we get close to this value. Specifically, the value of N for the simulation example is $N = 80$. Thus, the use of 100 time steps to approximately achieve the desired resolution with distributed planning has very small compared to the bound.

6 Discussion and Conclusion

We have studied the problem of optimal zone coverage for mapping using distributed sensing. We formulated the problem as an optimization problem with time-varying const function and shown that a simple gradient-descent optimization algorithm can lead to very efficient zone coverage, in the case of slow sensor motion. Our optimization exploits our ability to separate the fast varying angular variables from the slowly varying position variables and formulates the optimal problem as a sequential optimization with respect to the two separate sets of variables. Our experimental re-

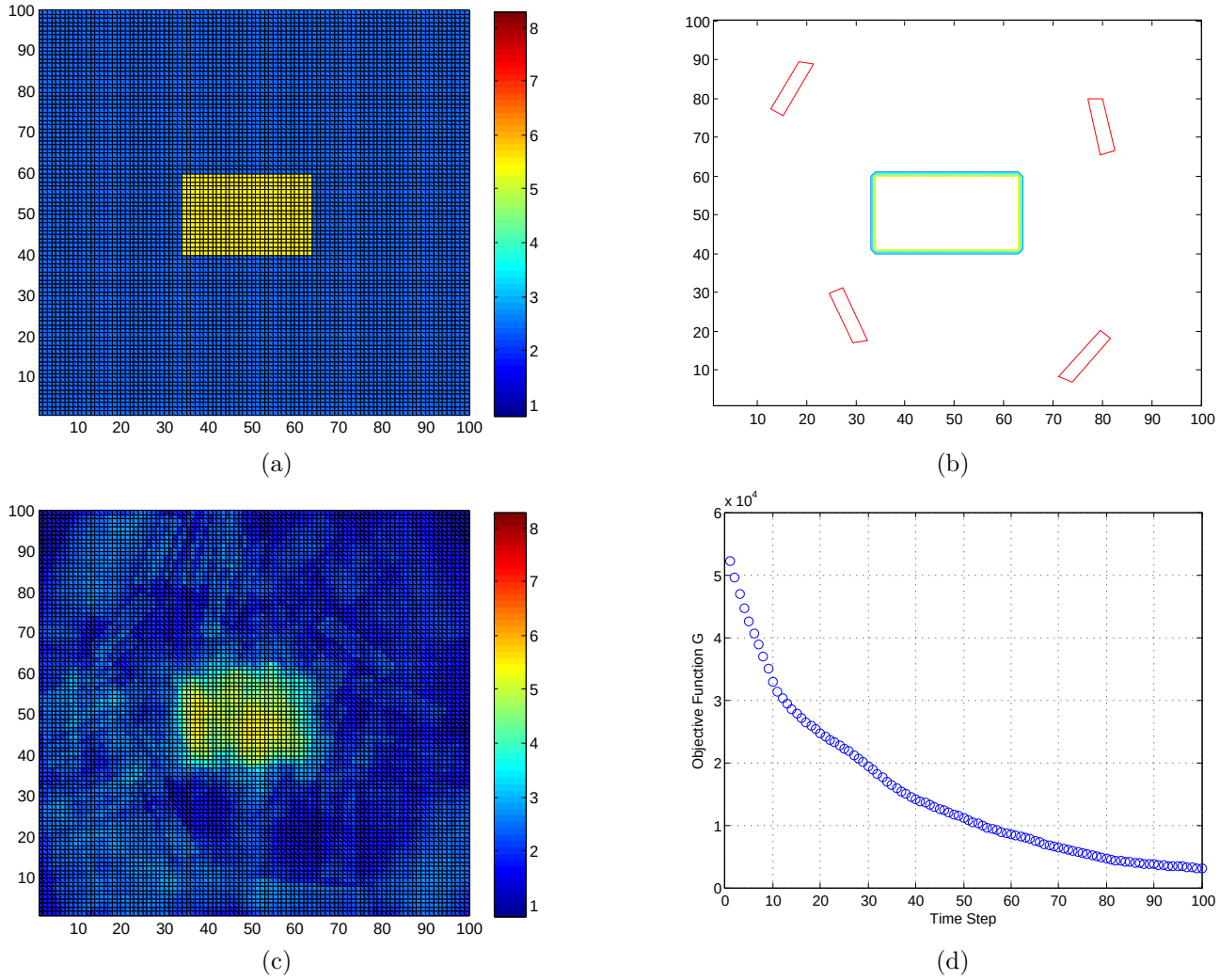

Figure 6: (a) The desired resolution function $\phi_d(q)$ used in our experiments. The environment contains a region which we desire to map with high resolution (yellow), while the rest of the environment should be mapped in lower resolution (blue). (b) Initial sensor footprints. The rectangle in the center represents the region of high desired resolution shown in (a). (c) The achieved resolution at $t = 100$, which is close to the desired resolution. (d) The evolution of the cost function $G_t(\mathbf{x}, \boldsymbol{\theta}_t, \boldsymbol{\psi}_t)$ as a function of time t. The loss function in the definition of G was $f(x) = |x|^{1.75}$. Note that the loss function has a norm structure, but the norm can be different than the norm used in modeling the resolution of the fused images.

sults demonstrate the effectiveness of our approach in achieving the desired map resolution.

A more rigorous analysis of the proposed algorithm in terms of convergence and optimality is necessary and will be presented in a longer journal version of this work. Furthermore, a complete distributed implementation of the algorithm, using some techniques from distributed optimization, e.g. [12], is also under investigation; the corresponding results will be reported in future publications.

References

[1] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Trans. on Robotics and Automation*, 20(2):243–255, 2004.

[2] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. Saluja. Sensor deployment strategy for target detection. In *Proc. of 1st ACM Intl. Workshop on Wireless Sensor Networks and Applications*, pages 42–48, 2002.

[3] F. Lian and R. Murray. Real-time trajectory generation for the cooperative path planning of multi-vehicle systems. In *Proc. of 41st IEEE Conf. on Decision and Control*, pages 3766–3769, 2002.

[4] L. Mihaylova, T. Lefebvre, H. Bruyninckx, and K. Gadeyne. Active sensing for robotics-a survey. in proc. of the 5th intl. conf. on numerical methods and applications. In *Proc. of the 5th Intl. Conf. on Numerical Methods and Applications*, pages 316–324, 2002.

[5] Christos G Cassandras and Wei Li. Sensor networks and cooperative control. *European Journal of Control*, 11(4):436–463, 2005.

[6] Minyi Zhong and Christos G Cassandras. Distributed coverage control and data collection with mobile sensor networks. In *49th IEEE Conference on Decision and Control (CDC)*, pages 5604–5609, 2010.

[7] Bruno Hexsel, Nilanjan Chakraborty, and Katia Sycara. Distributed coverage control for mobile anisotropic sensor networks. Technical Report Tech. Report CMU-RI-TR-13-01, Robotics Institute, Carnegie Mellon University, January 2013.

[8] Azwirman Gusrialdi, Takeshi Hatanaka, and Masayuki Fujita. Coverage control for mobile networks with limited-range anisotropic sensors. In *47th IEEE Conference on Decision and Control (CDC)*, pages 4263–4268, 2008.

[9] Minyi Zhong and Christos G Cassandras. Distributed coverage control in sensor network environments with polygonal obstacles. In *17th IFAC World Congress*, 2008.

[10] Jianchao Yang and Thomas Huang. Image super-resolution: Historical overview and future challenges. *Super-resolution imaging*, pages 20–34, 2010.

[11] Peter Corke. code.google.com/p/matlab-toolboxes-robotics-vision/source/browse/matlab/common/Polygon.m.

[12] Robin L. Raffard, Claire J. Tomlin, and Stephen P. Boyd. Distributed optimization for cooperative agents: Application to formation flight. In *Proc. of 43st IEEE Conf. on Decision and Control*, pages 2453–2459, 2004.