Swarm Interpolation Using an Approximate Chebyshev Distribution

Joshua Kirby[1], Marco A. Montes de Oca[2], Steven Senger[2], Louis F. Rossi[2], and Chien-Chung Shen[1]

[1] Department of Computer and Information Sciences, University of Delaware, Newark, USA
[2] Department of Mathematical Sciences, University of Delaware, Newark, USA
jothki@udel.edu, {mmontes,senger,rossi}@math.udel.edu, cshen@mail.eecis.udel.edu

Abstract. In this paper, we describe a novel swarming framework that guides autonomous mobile sensors into a flexible arrangement to interpolate values of a field in an unknown region. The algorithm is devised so that the sensor distribution will behave like a Chebyshev distribution, which can be optimal for certain ideal geometries. The framework is designed to dynamically adjust to changes in the region of interest, and operates well with very little a priori knowledge of the given region.

For comparison, we interpolate a variety of nontrivial fields using a standard swarming algorithm that produces a uniform distribution and our new algorithm. We find that our new algorithm interpolates fields with greater accuracy.

1 Introduction

The capability for a swarm of robots for tracking the location of a contamination or other hazard has been well understood for some time [6] [4], but once a primary body has been identified, or if its location is obvious from the start as for a large oil spill, mapping out the distribution of the field, the *swarm interpolation problem* is a different matter. Bertozzi et. al. presented a system for edge tracking, using a linked chain of robots that shape themselves to the outside contours of the region [2]. This method is sufficient for gathering information about the shape of a contaminated region, but not about the distribution of contaminants within it. Turduev et. al. developed a system for coordinating movement towards areas of higher concentration, but is designed more for identifing locations of maximum concentrations than for complete coverage of the region [10]. Cortes et. al. put forth a system for coverage via managing the configuration of Voronoi partitions, but it is optimized to detect events rather than gather data [3] . Finally, Krause et. al. presented an algorithm using the concept of mutual information to optimize placement, but assumes a fixed network rather than a mobile swarm [8] .

Kalentar et. al. proposed a solution involving dividing the robots present into two mutually exclusive groups [7] . One group acts to orient itself with the edge

M. Dorigo et al. (Eds.): ANTS 2012, LNCS 7461, pp. 324–331, 2012.

of the region, in a manner similar to Bertozzi's work. The second group acts to fill out the middle of the group, using a more conventional swarming algorithm to maintain a uniform distribution. The goal of this paper is to demonstrate a technique for improvement upon the distribution of robots within such a region.

When interpolating fields with a large number of measurements, the distribution of interpolating nodes is crucial for minimizing error. An effective distribution for this is based on the roots of a Chebyshev polynomial. (See [1], [9], and the references contained therein for a general discussion.) The following system yields the roots of the desired Chebyshev polynomial.

$$T_0(x) = 1, \qquad T_1(x) = x, \qquad T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \quad (n \geq 1) \tag{1}$$

Restricting the domain to [-1,1], for instance, Chebyshev polynomials can be specified by

$$T_n(x) = \cos(n \cos^{-1} x) \quad (n \geq 0) \tag{2}$$

where n is the desired number of roots for the polynomial. Using the positions of the roots of a Chebyshev polynomial as interpolation nodes for a 1D field leads to an error formula of

$$|f(x) - p(x)| \leq \frac{1}{2^n(n+1)!} \max_{|t| \leq 1} |f^{(n+1)}(t)|, \tag{3}$$

where $f(x)$ is the function being interpolated and $p(x)$ is the interpolation polynomial based on the Chebyshev roots. This distribution is optimal for polynomial interpolation. The distribution can be further extended from 1D into a 2D circle by applying the Chebyshev distribution along the radial axis while distributing points uniformly along the angular axis.

Aligning a swarm to a grid is a difficult problem when the area to be covered is not known in advance, but generating a similar but meshless distribution is a simpler matter. Typical swarming algorithms will produce meshless uniform distributions of robots, so we finesse a standard swarming algorithm by altering the distances measured between robots. If the perceived positions of the robots are transformed such that a Chebyshev distribution appears to the robots to be a uniform distribution, then the robots will naturally settle into an arrangement which is extremely close to an appropriate Chebyshev distribution as they swarm. In this paper, we present a model for achieving this distribution.

2 Force-Based Swarming Model

The simulation is carried out by a modified version of the Qualnet simulation platform, which handles actions and communications as discrete events, and simulates delay and signal loss in communications.

The algorithm utilizes a force-based model, where each robot has attractive or repulsive forces exerted on it by other nearby robots. In order to simulate a more realistic environment with limited communication ranges, and to limit the amount of computation required, a cutoff based on physical distance is applied,

with robots that fall outside that distance being ignored by each other when forces are calculated. The overall force for a robot is given by

$$\boldsymbol{F_n} = \sum_{j \in r_{near}} \text{Force}(n, j) \tag{4}$$

where F_n is the total force vector for robot n, r_{near} is the set of nearby robots, and $\text{Force}(n, j)$ is a function giving the force vector felt between two robots n and j.

In order to allow the robots to rapidly spread across a region while being constrained by its edge, we used an algorithm that alters the behavior of robots based on the difference between their current sensor readings and a set field strength, with the force felt by the robots directly proportional to that difference. Robots within the region thus feel repulsive forces and robots outside the region feel attractice forces, which approach zero as robots approach the edges of the region. In addition, a small field-independent repulsive force is included. This serves both to prevent robots on the outside from overly converging, and to prevent robots directly on the edge from becoming completely locked into position. The equation for the forces is

$$\begin{aligned}\text{Force}(n, j) = (&\text{Fscale}_n e^{-F_{\text{factor}}\sqrt{(x_n - x_j)^2}} + R_{\text{scale}} e^{-R_{\text{factor}}\sqrt{(x_n - x_j)^2}}) \\ &\times (\cos(\arg(x_n - x_j)), \sin(\arg(x_n - x_j)))\end{aligned} \tag{5}$$

where F_{factor}, R_{scale}, and R_{factor} are scaling factors, and Fscale_n is given by

$$\text{Fscale}_n = a(\phi_n - \text{edgevalue}) \tag{6}$$

where a is a scaling factor, ϕ_n is the value sensed by robot n at its current position, and edgevalue is the value that is sensed on the boundary of the region.

Unlike for physical forces, the net force does not indicate an acceleration, but rather a target velocity. The equation for acceleration is

$$\boldsymbol{A_n} = \kappa(\boldsymbol{F_n} - \boldsymbol{V_n}) \tag{7}$$

where $\boldsymbol{V_n}$ is the current velocity of robot n as measured at the time of computation and κ is a factor determining the rate at which acceleration occurs.

This model will yield a uniform spread across the region of interest. The modifications necessary to produce a Chebyshev distribution are described in the next section.

3 Applying Chebyshev Distribution

In order to move from a uniform distribution to a Chebyshev distribution, the coordinates can be remapped in such a way that they appear uniform when the nodes are properly distributed. The equation for forces then becomes

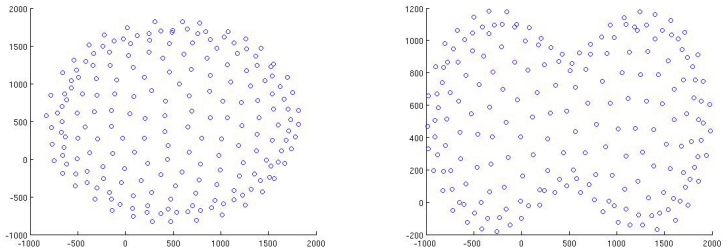
$$\begin{aligned}\text{Force}(n, j) = (&\text{Fscale}_n e^{-F_{\text{factor}}\sqrt{(vx_n - vx_j)^2}} + R_{\text{scale}} e^{-R_{\text{factor}}\sqrt{(x_n - x_j)^2}}) \\ &\times (\cos(\arg(x_n - x_j)), \sin(\arg(x_n - x_j)))\end{aligned} \tag{8}$$

while the formula for $\overline{vx}$, assuming that the points lie along the x axis, is

$$\overline{vx} = \frac{x_{max} - x_{min}}{2}(\pi - \cos^{-1}(\overline{x} - \frac{x_{max} + x_{min}}{2})) \tag{9}$$

where x_{max} is the high endpoint of the region, x_{min} is the low endpoint of the region, $\overline{x}$ is the set of true positions of the robots, and $\overline{vx}$ is the set of virtual positions of the robots, which will be used for generating forces.

The concept of a Chebyshev distribution can be extended to a circular region, with a dense outer edge and a sparse middle. An example of such a distribution is given in Figure 1. Extending a Chebyshev distribution in such a manner requires a shift in the way coordinates are handled, above and beyond simply adding an additional dimension. The same basic density distribution is present, but rather than simply remapping both the x and y axes, the coordinates need to be remapped along every line passing through the midpoint of the region.



(a) Circular Chebyshev Distribution (b) Starlike Chebyshev Distribution

Fig. 1. 2D Chebyshev Distributions

This can be accomplished by converting the coordinates of the robots from Cartesian to polar, centered on the midpoint of the region. Once this is done, the necessary coordinate shifts will all be parallel to the radial axis, and the magnitude of the shifts will be based solely on the radial positions. The equations for the shifts are

$$\overline{r} = \sqrt{(\overline{x} - x_{mid})^2 + (\overline{y} - y_{mid})^2} \tag{10}$$

$$\overline{\theta} = \tan^{-1}((\overline{y} - y_{mid}), (\overline{x} - x_{mid})) \tag{11}$$

$$\overline{vr} = r_{edge} \cos^{-1}(\tfrac{\overline{r}}{r_{edge}}) \tag{12}$$

$$\overline{v\theta} = \overline{\theta}, \overline{vx} = \overline{vr} \cos(\overline{v\theta}), \overline{vy} = \overline{vr} \sin(\overline{v\theta}) \tag{13}$$

where x_{mid} is the x coordinate of the midpoint of the region, y_{mid} is the y coordinate of the midpoint of the region, and r_{edge} is the radius of the region.

Perfectly circular regions are unlikely to exist under realistic conditions, but the concept of a Chebyshev-like distribution can be extended by allowing the

value of $\overline{r_{\text{edge}}}$ to vary across the region. As a result, each robot will have its own idea of how its distribution should work based on its angular position. The formula for this is

$$\overline{vr} = \overline{r_{\text{localedge}}} \cos^{-1}\left(\frac{\overline{r}}{\overline{r_{\text{localedge}}}}\right) \tag{14}$$

where $\overline{r_{\text{localedge}}}$ is an array containing the local edge distances for each node. An example of such a distribution is given in Figure 1.

Ideally, $\overline{r_{\text{localedge}}}$ would contain the exact values for the edge distances, but in this algorithm, the only information available is the reported positions and sensor readings of the other nodes. In order to approximate the true distance to the nearest edge, the nodes on the outside of the region are self-selected to act as representatives for a section of the edge, based on whether there is at least a 90 degree arc between any of the node's neighbors. Nodes on the inside look for the edge representative with the closest angular distance, and base their value for $\overline{r_{\text{localedge}}}$ on the distance between the representative node and the swarm, while nodes on the outside adopt their own distance, canceling out any shift in position.

4 Experiment Design

Four sets of experiments were performed with the algorithm, each based on a different sensed field. The equations for the four fields are given below, with (15) generating a circular level set, (16) generating a square, (17) generating a perturbed circle, and (18) generating a concave level set.

$$\phi(x, y) = e^{-8((x-.5)^2+(y-.5)^2)} \tag{15}$$

$$\phi(x, y) = e^{(-8\max(|x-.5|,|y-.5|)^2)} \tag{16}$$

$$\begin{aligned}\phi(x, y) = {} & (.05 * (\sin(15(x - .5)) + \sin(15(y - .5)))) \\ & \times e^{(-8((x-.5)^2+(y-.5)^2))}\end{aligned} \tag{17}$$

$$\begin{aligned}\phi(x, y) = {} & e^{(-8((x-.15)^2+(y-.5)^2))} \\ & + e^{(-8((x-.85)^2+(y-.5)^2))}\end{aligned} \tag{18}$$

Each set consisted of multiple experiments, across which the number of nodes varied, with each experiment run using 50, 100, 200, or 400. In addition, the same configurations were used for a version of the algorithm with the virtual coordinate remapping, yielding uniform distributions of robots across the region, with the same exterior edge but different interior node density. Each set therefore contained four Chebyshev runs and four corresponding uniform runs.

For all of the runs, the robots were initially placed in a uniform rectangular grid spanning from the coordinates [0,0] to [1000,1000], though they flowed beyond those boundaries during the runs. The parameters used for the swarming algorithm were $F_{\text{factor}} = .01$, $a = 200$, $R_{\text{scale}} = .02$, and $R_{\text{factor}} = 100$. The target edge strength for all fields was edgevalue = .5. The scaling factor for acceleration was $\kappa = 1$.

5 Interpolation

In the kinds of applications we are envisioning, all the data we will have at our disposal are measurements at the robots' locations. Thus, our input is a set $\{(\boldsymbol{x}_1, \phi_1), (\boldsymbol{x}_2, \phi_2)), \ldots, (\boldsymbol{x}_N, \phi_N))\}$, where N is the number of robots, $\boldsymbol{x}_n \in \mathbb{R}^2$ represents the location of the nth robot, and $\phi_n = f(\boldsymbol{x}_n)$ is the nth robot's measurement of the variable of interest (represented by the evaluation of the function f, whose definition is not known). Our goal is to find a function g such that $g(\boldsymbol{x}) = \phi$ and that the difference between g and f at locations different from $\boldsymbol{x}_n$, $n = 1, \ldots, N$ is as small as possible. This problem is known as *scattered data interpolation* [5].

In this paper, we tackle this problem using radial basis function interpolation. The goal is to find the values of the coefficients c_k, $k = 1, \ldots, N$ such that

$$g(\boldsymbol{x}) = \sum_{i=1}^{N} c_i \varphi(||\boldsymbol{x} - \boldsymbol{x}_i||_2)\,, \tag{19}$$

where φ is a radial basis function, and $||\cdot||_2$ is the Euclidean norm. The radial basis functions used in our experiments are Gaussians of the form

$$\varphi(r) = e^{-(ar)^2}\,, \tag{20}$$

where a is a parameter called *shape parameter*. By enforcing the condition $g(\boldsymbol{x}_i) = y_i$, the coefficients c_i, $i = 1, \ldots, N$ can be found by solving the linear system $A\boldsymbol{c} = \boldsymbol{\phi}$ where the entries A_{jk} of the matrix A are equal to $\varphi(||\boldsymbol{x}_j - \boldsymbol{x}_k||_2)$, $j, k = 1, \ldots, N$, $\boldsymbol{c} = [c_1, c_2 \ldots, c_N]^T$, and $\boldsymbol{\phi} = [\phi_1, \phi_2 \ldots, \phi_N]^T$.

We use two error measures. The first measure is the root-mean-square error (RMS-error) and is computed as follows

$$\text{RMS-error} = \sqrt{\frac{1}{M} \sum_{j=1}^{M} \left(g(\boldsymbol{\mathcal{E}}_j) - f(\boldsymbol{\mathcal{E}}_j)\right)^2}\,, \tag{21}$$

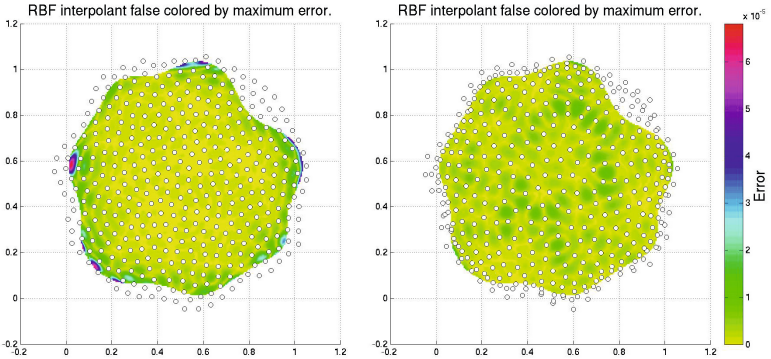
where $\boldsymbol{\mathcal{E}}_j$, $j = 1, \ldots, M$ are the evaluation points. The second measure is the maximum error (MAX-error) and is given by

$$\text{MAX-error} = \max\{|g(\boldsymbol{\mathcal{E}}_j) - f(\boldsymbol{\mathcal{E}}_j)|\}\,. \tag{22}$$

6 Results and Conclusions

We generated a rectangular grid of 500×500 points to sample a function f, which represents the fields described in Section 4, in the region $[-0.2, 1.2]^2$ and selected the points where $\boldsymbol{x} \in [-0.2, 1.2]^2$ to compute the RMS and MAX errors.

The positions of the robots were rescaled so that the boundary of the swarm matched the level curves corresponding to $f(\boldsymbol{x}) = 0.5$ in all the tested fields, resulting in a slightly smaller evaluation domain. The shape parameter of the radial



Spatial error with 400 robots. Uniform (left), Chebyshev (right).

Fig. 2. Spatial distribution of the error on the irregular field (Eq. 17)

Table 1. Error Ratios

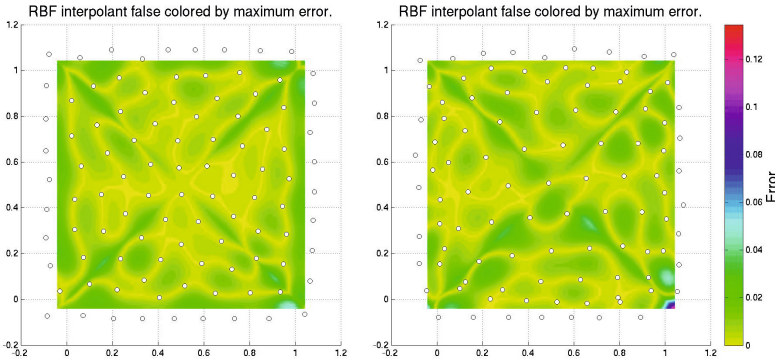
Robots	RMS Error				MAX Error			
	Circular field	Rectilinear field	Perturbed field	Concave field	Circular field	Rectilinear field	Perturbed field	Concave field
50	6.28e−01	8.35e−01	7.78e−01	7.20e−01	7.36e−01	5.68e−01	**1.15e+00**	**1.17e+00**
100	**1.21e+00**	9.65e−01	**1.29e+00**	9.81e−01	**1.47e+00**	3.76e−01	**2.30e+00**	**2.04e+00**
200	7.00e−01	5.79e−01	**1.08e+00**	**1.38e+00**	8.59e−01	5.39e−01	**2.36e+00**	**3.82e+00**
400	**1.11e+00**	8.37e−01	**2.36e+00**	**2.31e+00**	**2.06e+00**	5.80e−01	**2.98e+00**	**3.60e+00**

basis function used in our experiments was set to 7. Videos and results of the experiments can be found at `http://degas.cis.udel.edu/SwarmInterpolation/`.

Fig. 2, shows the spatial distributions of the error on the perturbed field with 400 robots. While the uniform distribution shows high error regions close to the edges of the evaluation domain, which confirms the observation made in Section 1 about the tendency of the error to grow near the boundaries of the domain, the Chebyshev-like distribution of robots results in the error being more evenly distributed across the domain.

In Table 1, we report the ratio of the errors obtained with the uniform distribution to the Chebyshev-like distribution using RMS and maximum metics. A ratio greater than one (highlighted in boldface) means that the error obtained with the uniform distribution is greater than the error obtained with the Chebyshev-like distribution.

In the majority of test cases, the RMS and MAX errors were greater than one, meaning that the Chebyshev-like distribution outperformed the uniform distribution. This effect was particularly pronounced for the more complex perturbed and concave fields, and with greater numbers of robots within the fields. The primary exception to this was the rectilinear field, which we assume is due to issues our swarming algorithm has with reaching the corners of the level curves.



Spatial error with 100 robots. Uniform (left), Chebyshev (right).

Fig. 3. Spatial distribution of the error on the rectilinear field

Acknowledgments. The authors thank Sherry Vaughan for her contributions to this project. This material is based upon work supported by the National Science Foundation under grant CCF-0916035.

References

1. Battles, Z., Trefethen, L.N.: An extension of matlab to continuous functions and operators. SIAM J. Sci. Comput. 25(5) (May 2004), http://dx.doi.org/10.1137/S1064827503430126
2. Bertozzi, A., Kemp, M., Marthaler, D.: Determining environmental boundaries: Asynchronous communication and physical scales. LNCIS, vol. 309, pp. 403–405 (2005)
3. Cortes, J., Martinez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks. IEEE Trans. on Robotics and Automation 20(2), 243–255 (2004)
4. Cui, X., Hardin, T., Ragade, R.K., Elmaghraby, A.S.: A swarm-based fuzzy logic control mobile sensor network for hazardous contaminants localization. In: 2004 IEEE Int. Conf. on Mobile Ad-hoc and Sensor Systems, pp. 194–203 (October 2004)
5. Fasshauer, G.E.: Meshfree Approximation Methods with MATLAB. Interdisciplinary Math. Sci., vol. 6. World Scientific Publishing, Singapore (2007)
6. Kadrovach, B.A., Lamont, G.B.: A particle swarm model for swarm-based networked sensor systems. ACM, New York (2002)
7. Kalantar, S., Zimmer, U.: Distributed shape control of homogeneous swarms of autonomous underwater vehicles. Autonomous Robots 22(1), 37–53 (2007)
8. Krause, A., Singh, A., Guestrin, C.: Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. J. Mach. Learn. Res. 9, 235–284 (2008), http://dl.acm.org/citation.cfm?id=1390681.1390689
9. Trefethen, L.: Spectral Methods in MATLAB. SIAM, Philadelphia (2000)
10. Turduev, M., Atas, Y., Sousa, P., Gazi, V., Marques, L.: Cooperative chemical concentration map building using decentralized asynchronous particle swarm optimization based search by mobile robots. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4175–4180 (October 2010)