The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Environmental Field Estimation of Mobile Sensor Networks Using Support Vector Regression

Bowen Lu, Dongbing Gu and Huosheng Hu

Abstract— **This paper presents a distributed algorithm for mobile sensor networks to monitor the environment. With this algorithm, multiple mobile sensor nodes can collectively sample the environmental field and recover the environmental field function via machine learning approaches. The mobile sensor nodes are able to self-organise so that the distribution of mobile sensor nodes matches to the estimated environmental field function. In this way, it is possible to make the next-step sampling more accurate and efficient. The machine learning approach used for function regression is support vector regression (SVR) algorithm. A distributed SVR learning algorithm is used for on-line learning. The self-organised algorithm used for deployment is based on locational optimisation techniques. In particular, Lloyd's algorithm for optimising centroidal Voronoi tessellations (CVT) is used to spread mobile sensor nodes over the monitored environment. The environmental field function is simulated in static and dynamic settings and the demonstration on the simulated environments shows the proposed algorithm is effective.**

I. Introduction

Sensor networks can answer queries about environment by sampling the environment over a large region. A network of mobile sensor nodes spreading out over the highly interested area is able to model the environmental field function. Spreading out a network of mobile sensor nodes is so-called coverage control of mobile sensor networks. The environmental field function or sensory distribution function is known a prior in research work of coverage control in [1], [2] where the locational optimisation techniques, particularly Lloyd's algorithm for optimising centroidal Voronoi tessellations (CVT) are used. Recent research in this area concentrates on simultaneously environmental function learning and coverage control. For example, the environmental field function is represented by using a radial basis function (RBF) network and the coverage control is implemented by using Lloyd's algorithm in [3]. The environmental field function is represented by using a RBF network and the coverage control is implemented by using flocking algorithms in [4]. The environmental field function is approximated by using an inverse distance weighting interpolation method and updated by using a Kalman filter in [5]. The environmental field function is modelled by using a spatial temporal Gaussian process and a flocking algorithm is used for the coverage control in [6]. The environmental field function is modelled by a spatial-temporal model and then estimated by a Kriged Kalman filter in [7].

The authors are with School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, Colchester, UK. blv@esssex.ac.uk,dgu@essex.ac.uk, hhu@essex.ac.uk

Support vector regression (SVR) is an approach to solve problems of function regression. As SVR learning is formulated as a constrained quadratic optimisation problem, it can find a global minimum. Using kernel methods for function regression over sensor networks has recently been studied [8]. It was reported in [9] where a Gaussian kernel function is adopted and the minimisation problem of kernel methods is converted into a linear equation problem. The distributed solution lies in the use of a bump function to glue local estimates to approximate the global estimate. In [10], the minimisation problem of kernel methods is converted into a linear equation problem and the distributed solution builds on a distributed application of Gaussian elimination. Specially, a message passing algorithm with the running intersection property is used for Gaussian elimination. In [11], the learning algorithm of kernel methods is viewed as an application of successive orthogonal projection algorithms, in which sensor nodes perform a local computation in sequential order. In [12], the minimisation problem of kernel methods is solved by using an incremental sub-gradient method due to its additive structure. A two-step message passing process is required to visit every sensor in the network.

In this paper we propose to use SVR for function regression over sensor networks. As the core of SVR is a constrained quadratic optimisation problem with additive structure, the incremental sub-gradient method [12] can be used. Alternatively, least square SVR [13] can be used to convert the minimisation problem into a linear equation problem and Gaussian elimination [10] can be applied to solve the linear equations. However, both of them ask for constructing a path in the sensor networks for message passing. This paper proposes to use a gradient based approach to SVR for distributed computation purpose. The gradient based approach uses the kernel mapping to map nonlinear data in low dimensional space into linear features in high dimensional space [14], [15]. Although the gradient based approach is proposed working in sequential, it can also work in parallel. However, it is not possible to apply this algorithm directly in sensor networks due to the fact that the iteration in the algorithm requires that each node obtain training data from all other nodes. In addition, sensory measurements from adjacent places are often highly precise and remote places are often not, we adopt a kernel function with finite support in the gradient based approach to implement distributed computation. As long as the communication range between neighbour nodes is large than the finite support of the kernel function, the iteration in the gradient based approach can be computed in a distributed way.

Mobile sensor networks are required to be able to self-organise based on the estimated environmental field function. The purpose of self-organisation is to maximise the possibility of monitoring the environment more accurately and efficiently. The locational optimisation is an efficient means to distribute the mobile sensor nodes so that more sensor nodes can be located in the area where the environmental field function has higher values. Higher values from an environmental field function usually indicates the region of interest, particularly, higher pollution concentration in a pollution monitoring case. Lloyd's algorithm used for optimising (CVT) was used in [1] for coverage control. In this paper, we also use Lloyd's algorithm as an adaptive controller for deployment of mobile sensor networks.

In following sections, Section II presents the gradient based distributed SVR algorithm. Section III applies CVT algorithm for coverage control of mobile sensor networks. Section IV provides simulation results including a static and a dynamic environmental field function. Finally, our conclusion is given in Section V.

II. Distributed Support Vector Regression

A. ε-SVR

$Q \subset \mathbb{R}^2$ is a $2D$ convex working environment for an N-sensor network. An arbitrary point in it is denoted by q. The coordinates of ith sensor in this network is denoted by $q_i = [x_i, y_i]^T$, and $z_i(t)_{(i=1,\ldots,N)}$ denotes the measured value from ith sensor at time t. Sample set of sensory measurement is defined as $S = ({q_i}^T, z_i)_{i=1}^N$. Finding a function $f(q) = w^T\phi(q)+b$, which can give a fitting result to the sample set S with a limited error ε, is the goal of ε-SVR [16]. $\phi(q)$ is a feature space function, which maps q from $\mathbb{R}^2$ to a higher-dimensional space. b is a biased constant.

Weight parameter w can be found by solving the following constrained convex optimising problem [17]:

$$\min_{w,\xi_i,\xi_i^*} \left\{ \frac{1}{2} w^T w + C\Big(\sum_{i=1}^{N} \xi_i + \sum_{i=1}^{N} \xi_i^* \Big) \right\} \tag{1}$$

subject to

$$\begin{cases} z_i - \langle w, \phi(q_i)\rangle - b \le \varepsilon + \xi_i \\ \langle w, \phi(q_i)\rangle + b - z_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases}$$

where $\langle \bullet, \bullet \rangle$ denote inner products. ξ_i and ξ_i^* are slack variables, constant $C > 0$ determines the tradeoff between a better fitting result or higher successful fitting rate. For solving this problem with inequality constraints, a dual optimisation problem should be solved [17]:

$$\begin{aligned} \min_{\alpha_i,\alpha_i^*} J =& \frac{1}{2} \sum_{i,j=1}^{N} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle \phi(q_i), \phi(q_j)\rangle \\ &+ \varepsilon \sum_{i=1}^{N} (\alpha_i + \alpha_i^*) - \sum_{i=1}^{N} z_i(\alpha_i - \alpha_i^*) \end{aligned} \tag{2}$$

subject to

$$\begin{cases} \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) = 0 \\ 0 \le \alpha_i, \alpha_i^* \le C \end{cases}$$

where α_i and α_i^* are non-negative Lagrange multipliers, and weight parameter w can be obtained from equation (3):

$$w = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*)\phi(q_i) \tag{3}$$

With equation (3), learning function $f(q) = w^T\phi(q) + b$ can be reformulated as equation (4):

$$f(q) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*)\langle \phi(q_i), \phi(q)\rangle + b \tag{4}$$

It was suggested that an augmenting factor λ can be used to simplify equation (4) and the following equation is obtained [14]:

$$f(q) = \langle \bar{w}, \bar{\phi}(q)\rangle \tag{5}$$

where

$$\begin{cases} \bar{w} = \{w^T, b/\lambda\} \\ \bar{\phi}(q) = \{\phi(q), \lambda\} \end{cases}$$

Eventually, the regression function is turned into:

$$f(q) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*)[\langle \phi(q_i), \phi(q)\rangle + \lambda^2] \tag{6}$$

Now the function regression problem comes down to finding a suitable mapping function $\phi(q)$ and learning parameters α_i, α_i^*.

For acquiring $\langle \phi(q_i), \phi(q)\rangle$ in equation (6), SVR maps an input data q from low dimensional data space $\mathbb{R}^2$ into a feature vector in high dimensional feature space $\mathcal{F}$ via a feature mapping $\phi(q)$. "Kernel trick" in SVR hides $\phi(q)$ into kernel function $K(q_i, q_j) = \langle \phi(q_i), \phi(q_j)\rangle$. The algorithm does not need to know $\phi(q)$ explicitly, but the inner product of feature vectors, i.e. kernel function $K(q_i, q_j) = \langle \phi(q_i), \phi(q_j)\rangle$. With this kernel function definition, equation (6) can be expressed as below:

$$f(q) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*)[K(q_i, q) + \lambda^2] \tag{7}$$

Building a suitable kernel function $K(q_i, q_j)$ needs to follow Mercer's condition:

1) $K(q_i, q_j)$ is continuous;
2) $K(q_i, q_j)$ is symmetrical $K(q_i, q_j) = K(q_j, q_i)$;
3) $K(q_i, q_j)$ is semi-positive definite.

B. Distributed SVR

Selecting a suitable kernel function is important for a distribute network. The following bump function with finite support B meets the above conditions and can be used as a kernel function:

$$K(q_i, q_j) = \begin{cases} \frac{1}{2}\left[1 + \cos(\frac{\pi \|q_i - q_j\|}{B})\right], & \|q_i - q_j\| \in [0, B] \\ 0, & otherwise \end{cases}$$

The main feature of such a kernel function lies in its finite support. This is one of the key points in our implementation of distributed SVR. As it is known, "kernel trick" hides explicit use of feature vectors. The above or Gaussian like kernel functions hides the explicit use of data q_i. It simply uses the Euclidean distance $||q_i - q_j||$ and z_i. This simplicity is very useful in sensor networks where $||q_i - q_j||$ is geometric distance between nodes i and j. Most sensor networks have ability to observe the geometric distance $||q_i - q_j||$ via local range finders. Due to limited communication and sensory range, it is cost-expensive to obtain all the geometric distances for every node. This kernel function with finite support B leads to a local version of computing the kernel matrix $K(q_i, q_j)$.

A sensor node i has a limited wireless communication range. Any other sensor nodes within this range are defined as its neighbour set N_i. Distributed SVR algorithm only sums up the measured values from the neighbours of each, instead of the measured values from whole network. Distributed SVR algorithm is listed in algorithm 1. More details on this algorithm can be found from our previous work [18].

Algorithm 1 Distributed SVR Algorithm

Initialise $\alpha_i = 0, \alpha_i^* = 0, t = 0$
Loops until the terminal condition is met:
  Each sensor node obtains the distance $||q_i - q_j||$ from its neighbour set N_i
  Each sensor node calculates the kernel function $K(q_i, q_j)$
  $E_i = z_i - \sum_{j \in N_i}(\alpha_i - \alpha_i^*)[K(q_i, q_j) + \lambda^2]$
  $\delta\alpha_i = \min\{\max[\gamma(E_i - \varepsilon), -\alpha_i], C - \alpha_i\}$
  $\delta\alpha_i^* = \min\{\max[\gamma(-E_i - \varepsilon), -\alpha_i^*], C - \alpha_i^*\}$
  Update $\alpha_i = \alpha_i + \delta\alpha_i$
  Update $\alpha_i^* = \alpha_i^* + \delta\alpha_i^*$
  $t \leftarrow t + 1$
end

III. Distributed Coverage Control

Our algorithm proposed in this paper includes two major steps in each loop. The first step is to implement the function regression with SVR as discussed in previous section. The second step is to implement the locational optimisation by using Lloyd's algorithm. This section introduces a distributed Lloyd's algorithm for coverage control based on the estimated environmental field function.

A. CVT

The definition of a Voronoi tessellations is showed below:

$$V_i = \{q \in Q \mid \|q - q_i\| \leq \|q - q_j\|, \forall i \neq j\} \tag{8}$$

V_i from above represents the Voronoi region of ith sensor, q is an arbitrary point in Q, and q_i is the generating point (sensor node position) of a Voronoi region. CVT is a special Voronoi tessellation, which requires each generating point move toward to the centre of mass of each Voronoi cell. For computing the mass centre of each V_i, we use the following definitions:

$$M_{V_i} = \int_{V_i} f(q)dq$$
$$L_{V_i} = \int_{V_i} qf(q)dq$$
$$C_{V_i} = \frac{L_{V_i}}{M_{V_i}}$$

where $f(q)$ is the environmental field function estimated from SVR algorithm.

The cost function of the locational optimisation problem is defined as:

$$H(q_1, \ldots, p_N) = \sum_{i=1}^{N} \int_{V_i} \frac{1}{2} \|q - q_j\|^2 f(q)dq \tag{9}$$

The gradient of the cost function with respect to sensor node position q_i is:

$$\begin{aligned} \frac{\partial H}{\partial q_i} &= -\int_{v_i} (q - q_j) f(q)dq \\ &= -M_{V_i}(C_{V_i} - q_i) \end{aligned} \tag{10}$$

B. Distributed CVT

Let R_i denote the maximum communication range of sensor node i. In order to calculate Voronoi region V_i for sensor node i, it is necessary to evaluate $f(q)$ in V_i. However, the distributed SVR algorithm only provides possibility of evaluating $f(q)$ within a circle region Ω_i of radius B for sensor node i, where $\Omega_i = \{q \in Q | \ \|q_i - q\| \leq B\}$. When $R_i \geq 2B$, the function evaluation is in a distributed form:

$$f(q) = \sum_{j=1}^{N_i} (\alpha_i - \alpha_i^*)[K(q_j, q) + \lambda^2] \tag{11}$$

Therefore a sensor node can only calculate a range-limited Voronoi region W_i:

$$W_i = \{q \in Q \mid \Omega_i \cap V_i\}$$

The corresponding mass centre of each W_i should be calculated in the following way:

$$M_{W_i} = \int_{W_i} f(q)dq$$
$$L_{W_i} = \int_{W_i} qf(q)dq$$
$$C_{W_i} = \frac{L_{W_i}}{M_{W_i}}$$

The mobile sensor node is modelled as a linear point model:

$$\dot{q}_i = u_i$$

where u_i is speed input to mobile sensor node i. With a step length β. The controller is:

$$u_i = -\beta \frac{\partial H}{\partial q_i} = \beta M_{W_i}(C_{W_i} - q_i) \quad (12)$$

This locational optimisation process is named as CVT algorithm and summarised below:

Algorithm 2 Distributed CVT algorithm

Initialise $M_{W_i} = 0, L_{W_i} = 0$
Sensors q_i is randomly located at a certain region
Loops until the terminal condition is met:
  Sample environmental variable z_i.
  Execute the distributed SVR algorithm.
  Obtain the updated $f(q)$ from SVR
  $M_{W_i} = \int_{W_i} f(q)dq$
  $L_{W_i} = \int_{W_i} qf(q)dq$
  Calculate the new mass centre $C_{W_i} = L_{W_i}/M_{W_i}$
  Move q_i towards C_{W_i} with a certain speed
end

Each CVT loop contains a SVR learning algorithm with multiple learning steps.

IV. Simulations

The simulation was conducted on an 1×1 square environment and included two sections: one is static simulation where the environmental field is a scalar static $2D$ function and another is dynamic simulation where the environmental field is a scalar time-variant $2D$ function. In both cases, a sensor network with $N = 30$ nodes was used. The communication range of each sensor of this network was set as $R = 0.2$. Number of loops for SVR algorithm was selected as 10. The number of loops for CVT algorithm was selected to be 50 for the static environment field, and 100 for the dynamic one. In both of these two simulations, sensor readings are assumed perfect (without noises).

A. Static Simulation

In the static simulation, sensor nodes were randomly placed in the centre of 1×1 simulation area. Considering most of real polluted environments, the concentration of pollution will not be evenly distributed and the changing is following continuous environmental field function. We use three Gaussian shape distributions denote the concentration, in which higher value implies more serious polluted. This field function was chosen as equation (13):

$$f(q) = 0.2e^{-\frac{(x-0.6)^2+(y-0.4)^2}{\sigma^2}} + 0.3e^{-\frac{(x-0.4)^2+(y-0.5)^2}{\sigma^2}} + 0.4e^{-\frac{(x-0.8)^2+(y-0.6)^2}{\sigma^2}} \quad (13)$$

where $\sigma^2 = 0.005$.

Fig 1 shows the true environmental field function. Fig 2 presents the estimated result of static simulation after 50 CVT loops. Comparing with true function in Fig 1, it is clearly showed that three peaks of underlying function are captured by the sensor network with proposed algorithms.

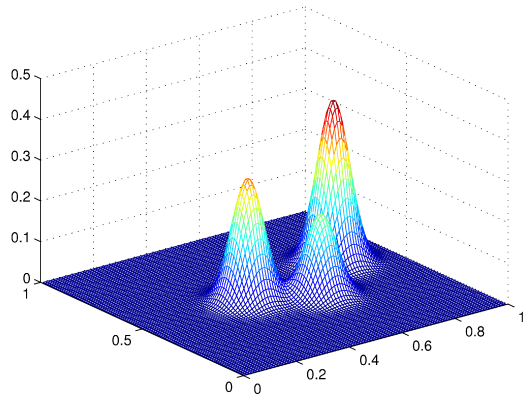

Fig. 1. True static environmental field function

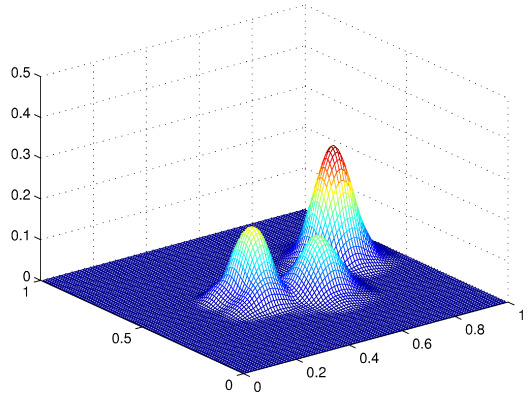

Fig. 2. Estimated static environmental field function

Error sum of true and estimated static environmental field functions was calculated after each loop and is showed in Fig 3 where the number of loops is denoted in x-axis. This figure clearly shows the error sum kept reducing but maintained with a stable static error after 30 loops. The static error is due to the sparseness of samples produced from the sensor network. This static error can be reduced by increasing the

sensor's communication range and/or increasing the density of sensors.

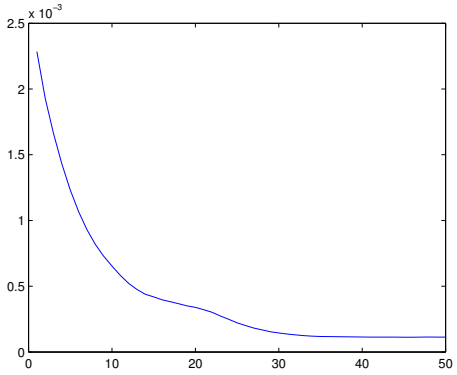

Fig. 3. Error curve of static simulation

The locational optimising procedure of CVT algorithm can be observed in Fig 4, which shows how mobile sensors moved and spread out by tracking the estimated environmental field function. The final Voronoi tessellation is also showed in Fig 4 for a clear demonstration.

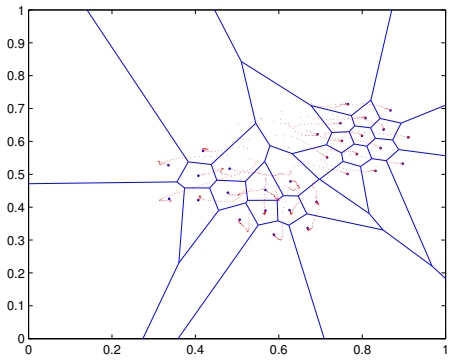

Fig. 4. Top view of locational optimised trajectories

B. Dynamic Simulation

In the dynamic simulation, the environmental field function was changing with time. It tries to test if the proposed algorithms can track a moving environmental field function. Still the underlying function had three peaks in this simulation. Initially, all the three peaks were located at the same position $(0.3, 0.8)$. Amplitude parameter and σ of each Gaussian function were selected the same as previous static simulation. The environmental field function was made changing with time by moving the three peak locations with constant speeds. The moving speeds of three peak components were selected as $(5 \times 10^{-4}, -5 \times 10^{-3}), (5 \times 10^{-3}, -5 \times 10^{-3})$, and $(5 \times 10^{-3}, -1 \times 10^{-3})$ per loop.

Fig 5 to Fig9 show the dynamic simulation process where (a) of each figure shows the true function, and (b) shows the estimated function. Time interval between two figures is 20 loops.

Fig 5 shows the three peaks of underlying function was moving at loop 20, but still very close to each other. The estimated function by the mobile sensor network had the similar pattern. After about 40 loops (Fig 6 to Fig 9), the three peaks gradually separated from each other due to their different velocity. The sensor network self-organised itself according to the changed underlying function and the estimated function was able to match to the truth.

Fig 10 gives the error sum between the true and estimated environmental field functions in each loop. The error sum was reduced because the mobile sensor network was gradually distributed close to the true distribution. A stable static error existed just the same as in the static simulation due to the sparse use of sensors.

The top view of sensor's moving trajectory in Fig 11 in the dynamic simulation shows how the mobile sensors tracked the underlying function.

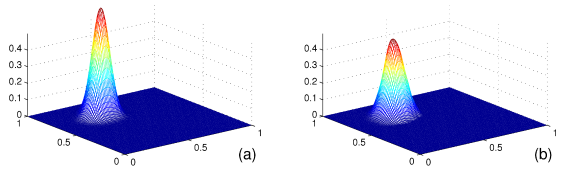

Fig. 5. Dynamic simulation result at loop = 20

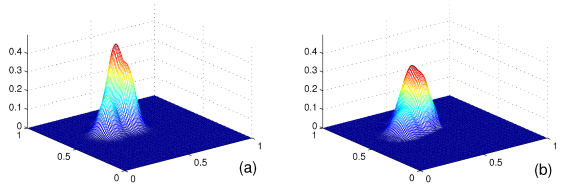

Fig. 6. Dynamic simulation result at loop = 40

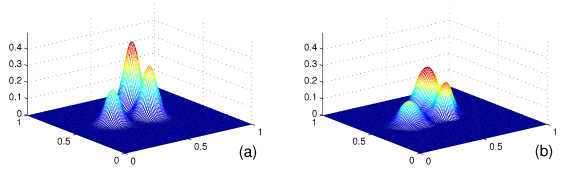

Fig. 7. Dynamic simulation result at loop = 60

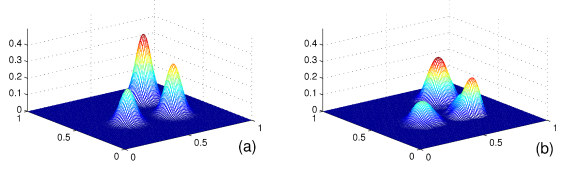

Fig. 8. Dynamic simulation result at loop = 80

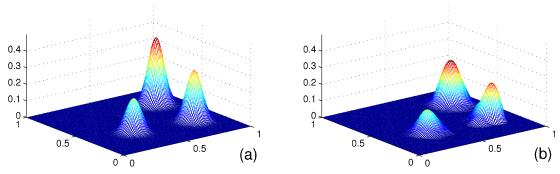

Fig. 9. Dynamic simulation result at loop = 100

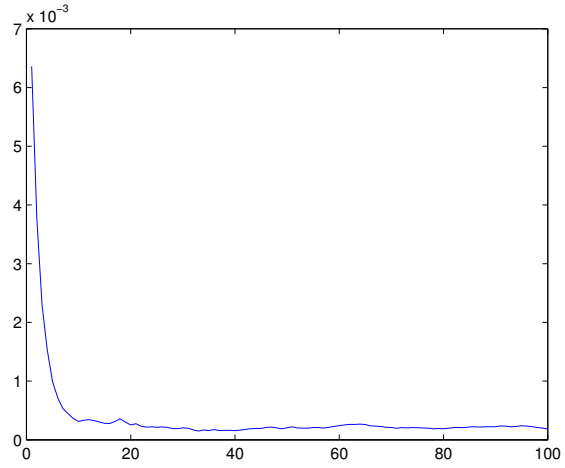

Fig. 10. Error curve of dynamic simulation

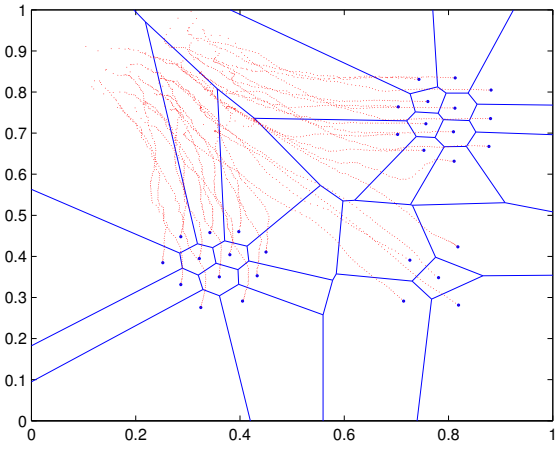

Fig. 11. Top view of sensor's trajectories

V. Conclusions

A distributed environment monitoring algorithm for mobile sensor networks is investigated in this paper. Distributed SVR algorithm is an effective means to estimate environmental field functions. Distributed SVR algorithm combined with locational optimisation CVT algorithm provides a solution to environment monitoring problem with mobile sensor networks. Static and slow-changed environmental field functions are simulated to verify the effectiveness of proposed algorithm. The distributed nature of the proposed algorithm improves the robustness of environment monitoring algorithm.

Our further research will consider the addition of exploration behaviour to the algorithm so that the entire network is able to escape from local minima. Also, various of improved algorithms which can adapt for different environments (e.g. gas, water, and liquid) will be investigated, and eventually these algorithms should be applied on SHOAL project.

Acknowledgement

This research has been financially supported by European Union FP7 program, ICT-231646, SHOAL: Search and monitoring of Harmful contaminants, Other pollutants And Leaks in vessels in port using a swarm of robotic fish.

References

[1] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. on Robotics and Autonamous*, vol. 20, no. 2, pp. 243–255, 2004.

[2] L. C. A. Pimenta, V. Kumar, R. C. Mesquita, and G. A. S. Pereira, "Sensing and coverage for a network of heterogeneous robots," in *Proc. of the IEEE Conf. on Decision and Control*, Cancun, Mexica, 2008.

[3] M. Schwager, D. Rus, and J. Slotine, "Decentralized, adaptive coverage control for networked robots," *Int. J. of Robotics Research*, vol. 28, no. 3, pp. 357–375, 2009.

[4] K. M. Lynch, I. B. Schwartz, P. Yang, and R. A. Freeman, "Decentralized environmental modelling by mobile sensor networks," *IEEE Trans. on Robotics*, vol. 24, no. 3, pp. 710–724, 2008.

[5] S. Martinez, "Distributed interpolation schemes for field estimation by mobile sensor networks," *IEEE Trans. on Control Systems Technology*, vol. 18, no. 2, p. to appear, 2010.

[6] J. Choi, J. Lee, and S. Oh, "Swarm intelligence for achieving the global maximum using spatio-temporal Gaussian processes," in *Proc. of the American Control Conference*, Seattle, Washington, 2008.

[7] J. Cortes, "Distributed Kriged Kalman filter for spatial estimation," *IEEE Trans. on Automatic Control*, vol. 54, no. 12, pp. 2816–2827, 2009.

[8] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Singal Processing Magazine*, vol. 23, no. 4, pp. 56–69, 2006.

[9] S. Simic, "A learning theory approach to sensor networks," *IEEE Pervasive Computating*, vol. 10, pp. 44–49, 2003.

[10] C. Guestrin, R. Thibaux, P. Bodik, M. A. Paskin, and S. Madden, "Distributed regression: an efficient framework for modeling sensor network data," in *Information Processing in Sensor Networks (IPSN '04), Berkeley*, 2004.

[11] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Distributed kernel regression: an algorithm for training collaboratively," in *Proc. of 2006 IEEE information Theory Workshop*, Punta del Este, Uruguay, 2006.

[12] M. G. Rabbat and R. D. Nowak, "Quantized incremental algorithms for distributed optimization," *IEEE J. Special Areas Comunication*, vol. 23, no. 4, pp. 798–808, 2006.

[13] J.A.K.Suykens and J. Vandewalle, "Least square support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[14] S. Vijayakumar and S. Wu, "Sequential support vector classifiers and regression," in *Proc. of Int. Conf. on Soft Computing*, Genoa, Italy, 1999, pp. 610–619.

[15] T. Frieb, N. Cristianini, and C. Campbell, "The kernel-adatron algorithm: a fast and simple learning procedure for support vector machines," in *Proc. of the 15th Int. Conf. on Machine Learning*, 1998, pp. 188–196.

[16] V. Vapnik, *The nature of statistical learning theory*. Springer-Verlag, New York, 1995.

[17] A. J. Smola and B. Schlkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199–222, 2004.

[18] D. Gu and Z. Wang, "Distributed regression over sensor networks: an support vector machine approach," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS08)*, Nice, France, Sept. 2008.