Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Reformulating Kalman Filter Based Optimal Dynamic Coverage Control

Lingji Chen and Raman K. Mehra

Scientific Systems Company Inc, 500 West Cummings Park, Suite 3000, Woburn, MA, 01801, USA (Tel: 781-933-5355; email: Lingji.Chen,Raman.Mehra@ssci.com).

Abstract: Our objective in this paper is to examine the problem formulation of one particular published result, make logical extensions, and consider the question of whether we should obtain solutions under such a formulation, or pursue alternative formulations. The problem considered in "A Kalman Filter-Based Control Strategy for Dynamic Coverage Control" by I. I. Hussein is for a network of mobile sensors with limited range to traverse and estimate a spatially-decoupled scalar field. In that paper the optimal trajectories are generated by an online procedure that minimizes the trace of the instantaneous covariance of the estimation error obtained from Kalman Filtering, using a finite set of admissible control inputs. We extend the formulation by observing that the procedure can be performed offline, that the cost function can be defined over a finite horizon, and that the set of control inputs can be a continuum. We illustrate, with simple examples, the kind of solutions that can be obtained using dynamic programming, and ask the question "Is this the type of trajectories that we want?" Alternative formulations are suggested and are left for future work.

1. INTRODUCTION

Networks of mobile agents equipped with limited-range sensors have been used to perform various tasks, and they present interesting cooperative control problems; see Hussein [2007], Cortes et al. [2004], Cassandras and Li [2005], Chung et al. [2004], A.Tiwari et al. [2005] and the references therein for reviews and real world applications. One such task is the dynamic coverage of a spatially-decoupled scalar field, and the objective is to design trajectories of the agents so as to maximize the quality of the field estimate. In Hussein [2007] the optimal trajectories are generated by an online procedure that minimizes the trace of the instantaneous covariance of the estimation error obtained from Kalman Filtering, using a finite set of admissible control inputs.

Our interest in this paper is to examine the above formulation, make logical extensions, and investigate the nature of the solutions obtained. More specifically, We first observe that the procedure can be performed offline, that the cost function can be defined over a finite horizon, and that the set of control inputs can be a continuum. We then use some simple examples to illustrate the kind of solutions that can be obtained. These solutions have the appearance of "getting stuck in local minima," while in fact they are globally optimal solutions under the "minimum trace" criterion. Hence, if such trajectories are not desirable, it is an indication that perhaps alternative problem formulations should be further investigated. In a way, this paper raises more questions than it answers, and should be understood as exploratory rather than prescribing a practical solution.

To begin, we first describe the problem set up as in Hussein [2007]. The field to be measured is represented by a set of scalar values r_k at a set of pre-determined locations (which we will call "targets") p_k on a plane, $k = 1, 2, \ldots, N$. The field is assumed to be static but corrupted by noise, so we have

$$r_k(t+1) = r_k(t) + d_k(t),$$

where $d_k(t)$ is a zero mean Gaussian noise with variance σ_k^2.

There are M sensors moving on the plane according to

$$x_j(t+1) = x_j(t) + u_j(t), j = 1, 2, \ldots, M,$$

where $x_j(t)$ is the location of the j-th sensor, and the control input $u_j(t)$ is constrained by, without loss of generality,

$$||u_j(t)|| \le 1, \forall t \le 0. \tag{1}$$

Each sensor has a finite range b_j, and thus its measurement of the k-th target takes the form

$$z_k^j(t) = \mathcal{I}_{\{x_\ell:||x_\ell - p_k||<b_j\}}(x_j) r_k(t) + w_k^j(t),$$

where $\mathcal{I}_S(s)$ is the indicator function of the set S, i.e., $\mathcal{I}_S(s)$ equals 1 if $s \in S$ and 0 otherwise. The variance of w_k^j is denoted by W_k^j and modeled by

$$W_k^j = \begin{cases} \dfrac{\alpha||x_j - p_k||^2}{b_j^2} + \eta^2 & \text{if } ||x_j - p_k|| < b_j \\ \alpha + \eta^2 & \text{otherwise,} \end{cases} \tag{2}$$

where α and η are parameters.

If we define

$$r(t) = [r_1(t), r_2(t), \ldots, r_N(t)]^T,$$
$$d(t) = [d_1(t), d_2(t), \ldots, d_N(t)]^T,$$
$$x(t) = [x_1^T(t), x_2^T(t), \ldots, x_M^T(t)]^T,$$
$$u(t) = [u_1^T(t), u_2^T(t), \ldots, u_M^T(t)]^T,$$

$$z(t) = [z_1^1(t), \ldots, z_N^1(t), z_1^2(t), \ldots, z_N^2(t), \ldots, z_1^M(t), \ldots, z_N^M(t)]^T,$$

and

$$w(t) = [w_1^1(t), \dots, w_N^1(t), w_1^2(t), \dots, \\ w_N^2(t), \dots, w_1^M(t), \dots, w_N^M(t)]^T,$$

then we can write the equations in matrix form:

$$r(t+1) = r(t) + d(t), \tag{3}$$

$$x(t+1) = x(t) + u(t), \tag{4}$$

$$z(t) = H(t)r(t) + w(t), \tag{5}$$

where $H(t)$ can be defined in a straightforward way. Let the covariances of $d(t)$ and $w(t)$ be $D(t)$ and $W(t)$ respectively. We also assume that noises in different processes or at different time instants are independent.

Given positions of the sensors at time t, the matrix $H(t)$ is determined, whose entries are zeros and ones depending on whether the targets are in the range or out of range of the sensors. Kalman Filtering can then be performed to estimate the signal $r(t)$ with an error covariance $P(t)$.

The coverage control problem is to define the trajectories of the sensors optimally in some sense. The basic algorithm proposed in Hussein [2007] can be qualitatively summarized as follows:

- At each instant, choose for each sensor a candidate control input from a finite set.
- For every candidate move $u(t)$ for the entire sensor suite, calculate the resulting candidate position $x(t+1)$ and subsequently the candidate error covariance $P(t+1)$.
- Pick the $u(t)$ that minimizes the trace of $P(t+1)$.

There are modifications to the basic algorithm in Hussein [2007] to deal with the problem of local minima; this will be commented upon later.

In the next section we will examine this problem formulation and make logical extension of it. Then we will describe how to solve the newly posed problem using dynamic programming. In Section 4 we will illustrate the kind of "optimal" trajectories that will be obtained under the defined optimality criterion, and ask the question of whether this is the kind of trajectories that we want for the coverage control problem. Discussion will be given in Section 5 and conclusions drawn in Section 6.

2. EXTENDED PROBLEM FORMULATION

We make the following observations regarding the problem formulation and the basic algorithm described in the previous section.

Optimization can be performed offline: The estimate $\hat{r}(t)$ of the field has to be obtained online while the sensors are traversing it, since $\hat{r}(t)$ depends on the measurement signal $z(t)$ which in turn depends on the realization of the noise signal $d(t)$. However, the nature of Kalman Filtering is such that, given a trajectory $x(t)$ of the sensor suite, the evolution of the error covariance matrix $P(t)$ can be determined offline, *independent* of the measurement signal $z(t)$ received online. Since performance index is defined in terms of the trace of $P(t)$, essentially the optimization procedure can be performed offline.

The implication of this observation is threefold.

First, we can compute the solution offline using as much computational power as needed, and store the solution for online use.

Second, if the environment is time-varying and requires online re-computation, then we can think about ways to devise approximate solutions that can be computed fast, and ways to use the offline solutions as good initialization values for online procedures.

Third, and most pertinent to the objective of this paper, we can investigate suitability of various performance indices based on the error covariances, separately from the constraints of online computation. Simply speaking, a performance index is a tool that helps us to arrive at a solution for a practical problem; it can and should be subject to modification in design iterations. A fighter-aircraft feedback control law that is optimal under a particular Linear Quadratic Gaussian criterion may not be optimal after all if the pilot says "this does not feel right," and the designer may want to change for example the weighting matrices and have the corresponding solutions verified again.

Thus, if some performance index is more computationally intensive, but leads to a "better" result, then at the very least we know that we can obtain the offline solution and make use of it in some fashion online. Conversely, if the full-blown offline solution under some performance index is not satisfactory, then at least we know that it is not the online approximating scheme (subject to the limited computational time and resources) that is to blame.

Looking ahead multiple steps should be better than one step: To convey the idea, imagine that there is only one sensor and one target. The sensor has a limited range and can move some maximum distance in one time step. Assume that in one step the target will still be out of the range of the sensor, but in two steps it will be in the range. If the performance index is the trace of the error covariance matrix at only the next time step, then the sensor would not have any preference of the direction it chooses to go next, since in one step it cannot see the target and thus the error covariance would be the same in all directions. But if we define the performance index such that we look ahead two or more steps, then the optimal solution would apparently be for the sensor to go towards the target.

The set of allowable control inputs do not have to contain only extreme values: More often than not we would have to discretize a continuous variable to perform certain computations. Nevertheless, the choice for the control input to the sensor does not have to be limited to only extreme values such as "no movement" or "one maximum step east."

Imagine again the example of one target and one sensor, and the target is only half a maximum step away from the sensor. When we formulate the problem, we should allow the sensor to get to the target by going forward only half a maximum step, otherwise it would always overshoot and oscillate around the target.

Based upon these observations, we "naturally" extend the problem formulation as follows: Given the field and sensor equations (3), (4), (5), the constraint (1), and standard Kalman filtering procedure that yields an error covariance matrix $P(t)$, find optimal trajectories of the sensors such that the following performance index is minimized:

$$J(x_0, P_0) = \sum_{i=0}^{T} \text{trace}(P(i)). \tag{6}$$

where the number of steps T is a parameter chosen a priori, x_0 is the initial position of the sensors, and P_0 is the prior covariance matrix of the targets.

3. SOLUTION BY DYNAMIC PROGRAMMING

In this section we describe how to solved the newly posed optimization problem, using dynamic programming.

We start by giving the evolution of the error covariance in Kalman Filtering. By an abuse of notation we will put the time dependence in the subscript (which consequently no longer denotes individual sensors or targets), and put other dependence in the pair of parentheses following a variable.

$$\begin{aligned}
P^-_{t+1} &= P_t + D_t,\\
K_{t+1} &= P^-_{t+1} H^T_{t+1}(x_{t+1})\\
&\quad (H_{t+1}(x_{t+1}) P^-_{t+1} H^T_{t+1}(x_{t+1}) + W_{t+1}(x_{t+1}))^{-1},\\
P_{t+1} &= (I - K_{t+1} H_{t+1}(x_{t+1})) P^-_{t+1}.
\end{aligned}$$

Recall that the sensor movement equation is

$$x_{t+1} = x_t + u_t, \tag{7}$$

thus we define a function $f(\cdot)$ suitably so that we can write in a compact form

$$P_{t+1} = f(x_t, P_t, u_t). \tag{8}$$

Now if we rewrite J as

$$J(x_0, P_0) = \text{trace}(P_T) + \sum_{i=0}^{T-1} \text{trace}(P_i), \tag{9}$$

then system equations (7) and (8), cost function (9), together with constraints (1), constitute the optimization problem that can be solved using dynamic programming. Our objective is to obtain an optimal feedback law

$$u^*_t = \mu^*_t(x_t, P_t)$$

such that $[u^*_0, u^*_1, \ldots, u^*_{T-1}]$ minimizes $J(x_0, P_0)$.

To proceed, we first discretize both x and P to obtain a finite "state space," and discretize u to obtain a finite set of control inputs. More specifically, we put a bounded grid on (x, P), and consider only input values that can move the sensor from one grid point to another.

Now we solve the dynamic programming problem recursively as follows (Bertsekas [2005]):

- Start with
$$J_T(x_T, P_T) = \text{trace}(P_T).$$
- For $i = T-1, T-2, \ldots, 0$, solve
$$J_i(x_i, P_i) = \text{trace}(P_i) + \min_{u_i} J_{i+1}(x_i + u_i, f(x_i, P_i, u_i)).$$
- Then $J_0(x_0, P_0)$ is the optimal cost, and the minimizing u_i in the recursion above form the optimal sequence.

To understand the form of the solution, suppose there is one sensor and two targets, and we look ahead 5 steps ($T = 5$). Then the optimal solution is stored as a collection of 5 atlases, with each atlas containing maps indexed by two values corresponding to the two error variances of the targets. Each map specifies that at a given location of the field, where the sensor should move to at the next step. Examples of such maps will be given in the next section.

It is well known that dynamic programming suffers from the curse of dimensionality, and we can see from the above example that even with one sensor and two targets, the storage space needed for the solution is quite large. There are ways to alleviate this problem, for example, the solution can be (approximately) represented by an interpolating function instead of a table (Bertsekas and Tsitsiklis [1996]).

However, as stated earlier, the main objective of this paper is to examine different problem formulations, rather than proposing one particular formulation and solution. As such, we would not focus on how to solve the above optimization problem more efficiently. Instead, we will look at what type of optimizing trajectories we can get out of the problem formulation, which is illustrated in the next section.

4. "OPTIMAL" TRAJECTORIES ILLUSTRATED

Let us now consider two simple examples.

Example 1: There is one sensor and one target. The target is located at the origin, and the parameters for the sensor, used in (2), are

$$b = 1.5, \alpha = 1, \eta = 1.$$

T is chosen to be 3.

Intuitively, the sensor should go towards the target as quickly as possible, so the optimal feedback law should be

$$u^*_t(x_t, P_t) = \begin{cases} -\dfrac{x_t}{||x_t||} & \text{if } ||x_t|| > 1, \\ -x & \text{otherwise.} \end{cases} \tag{10}$$

The optimizing feedback law takes the form of 3 atlases each indexed by the error variance P. We show the 3 maps corresponding to $P = 1$ in Figure 1, as velocity plots. – In a velocity plot, the vectors are scaled for better display, so we can tell the relative but not absolute amplitudes of the optimizing $u(t)$. Thus These maps show the best direction to go at any point in the field (when the error variance is 1) for 1-step-to-go, 2-step-to-go and 3-step-to-go respectively.

We can observe the following from Figure 1:

- When there is only one step to go (top map), the optimal direction depends on how far away the sensor is to the target. If the sensor is within a circle of radius 2 centered at the origin (where the target is), then it knows that in one step the target will be within its range and that it can reduce the variance of the estimation error. Therefore it decides to go towards the target. If the sensor is outside that circle, then it cannot reduce the estimation error in one step by going in any direction. The direction shown in the

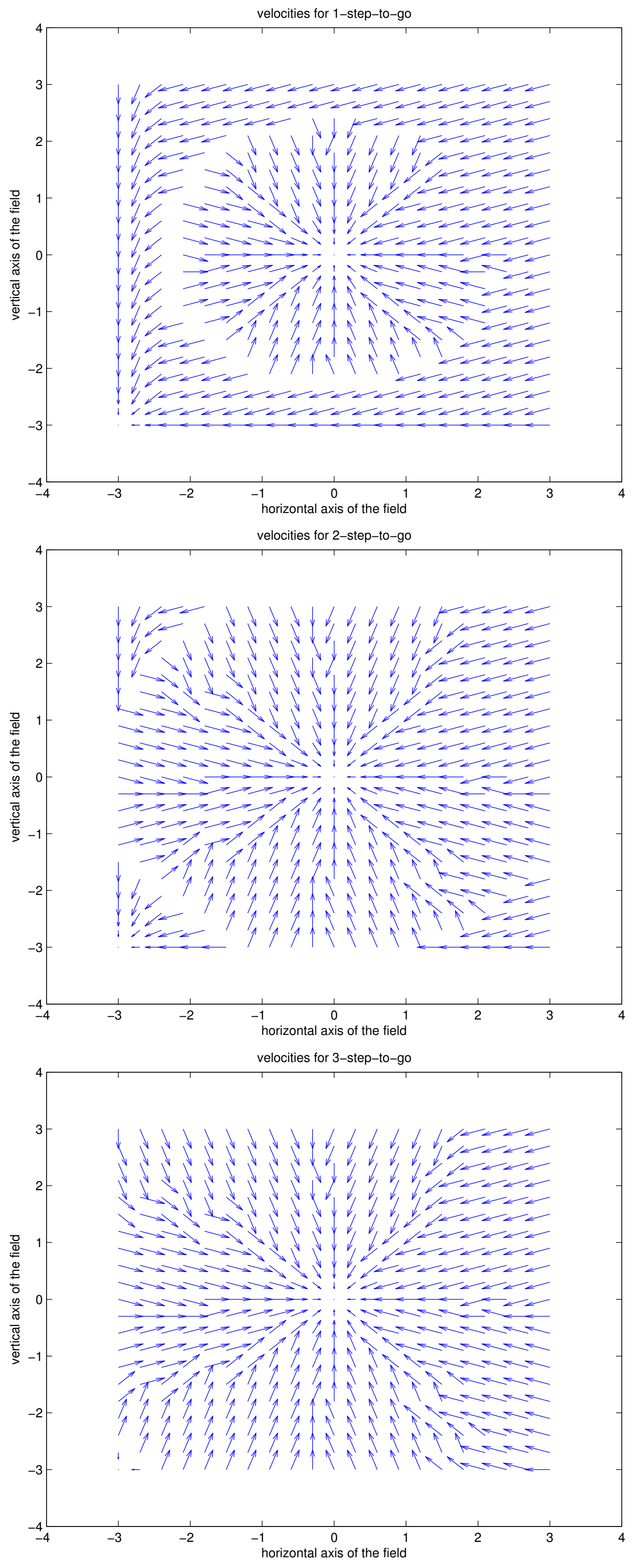

Fig. 1. Optimal feedback law shown as velocity plots for 1-step-to-go, 2-step-to-go and 3-step-to-go. There is one sensor and the target is at the origin.

map merely reflects the preference in tie breaking (lower index of an array, where values are negative).

- Where there are two steps to go (middle map), the radius of the circle becomes 3.
- When there are 3 steps to go (bottom map), the optimal feedback law resembles that specified by (10). The mismatch is mostly likely due to the discretization of the state space and control input.
- An optimal trajectory starting from (x_0, P_0) is read from the atlases as follows: First consult the 3-step-to-go atlas. Find the map that matches the discretized value of P_0, and look at the position on that map that matches the discretized value of x_0. Apply the specified optimal control input u_0, which brings the sensor and estimate to (x_1, P_1). Then consult the 2-step-to-go atlas, and repeat the process.
- We can thus see that the 3-step-to-go map subsumes the 2-step-to-go and 1-step-to-go maps, i.e., after we have consulted with the 3-step-to-go map and when we need to consult the 2-step-to-go map, we can read out the same information still from the 3-step-to-go map, and similarly for the 1-step-to-go map. It seems most likely that this can be generalized to an arbitrary number of sensors and targets, as long as the number of steps T is large enough to cover the entire field, but a rigorous proof is beyond the scope of this paper.

Example 2: Let us now consider another example where there are two targets and one sensor. The targets are located at $(-1, 0)$ and $(1, 0)$ respectively.

First we let the range of the sensor be chosen as $b = 1.5$, and other parameters the same as in Example 1. Now the atlas is indexed by two values from the diagonal of the covariance matrix P (i.e., the error variances of the two target estimates). Two 5-step-to-go maps are shown in Figure 2, for the case of $P_0 = \text{diag}(1, 1)$ and $P_0 = \text{diag}(1, 0.1)$ respectively.

In the top map, initial estimates of the two targets have the same accuracy and hence the configuration is symmetric. Since $b = 1.5$, at the origin the sensor can see both targets one unit distance away, and therefore it “settles down” at the origin, in the sense that if it moves away from the origin towards one target, then the accuracy of the estimate for that target improves while the accuracy for the other target deteriorates, and the net effect is a loss in the sum of variances.

In the bottom map, the initial estimate for the left target is coarser than that of the right target, and therefore the “settling down” point is closer to the left target, at the border where it can still see the right target.

Now we change the range of the sensor to $b = 0.5$, and the two maps are shown in Figure 3.

This time the sensor cannot see both targets simultaneously at any location, so it “settles down” to one of them.

Note that we are only showing two maps from the atlas, and therefore the details of any optimal trajectory is not evident by inspection of these two maps. One particular trajectory is shown in Figure 4.

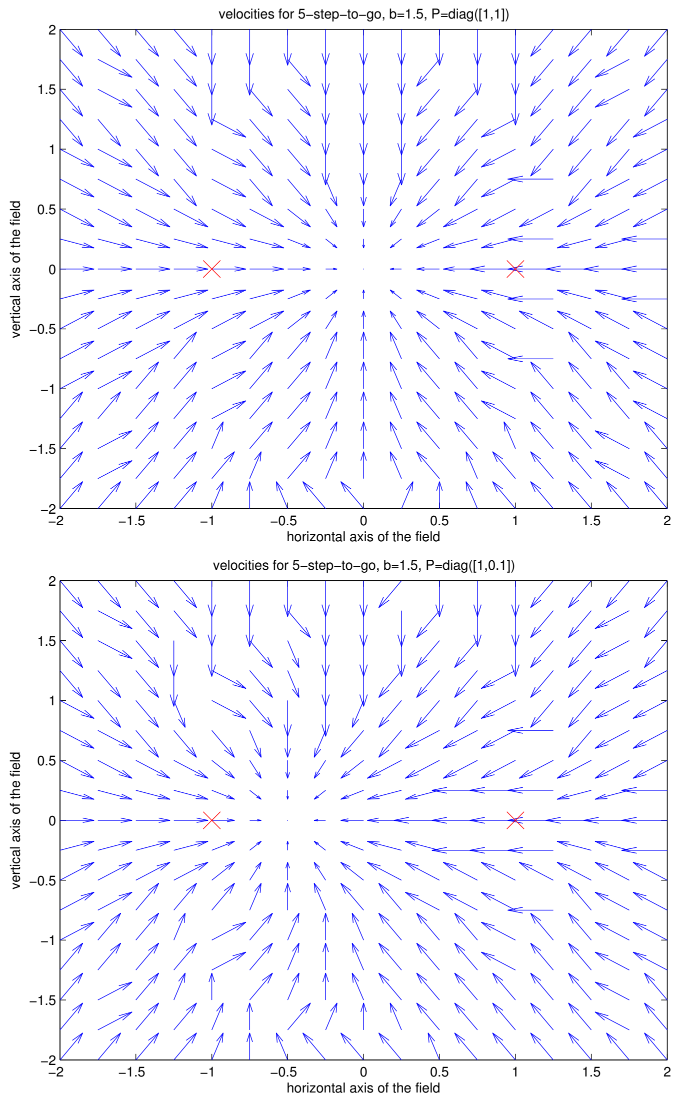

Fig. 2. The 5-step-to-go map for $P_0 = \text{diag}(1, 1)$ and $P_0 = \text{diag}(1, 0.1)$ respectively. Sensor range is $b = 1.5$ and the two targets are marked by (red) crosses.

5. DISCUSSIONS AND ALTERNATIVE PROBLEM FORMULATIONS

Looking at the maps shown in Figure 2 and Figure 3, and the "optimal" trajectory shown in Figure 4, we may wonder whether we have gotten stuck in some "local minima."

But keep in mind that the dynamic programming procedure gives us globally optimal solutions.

If (6) *is* what we want to minimize, then the trajectory in Figure 4 *is* globally optimal.

We may have in mind a trajectory that simply visits the two targets sequentially, as shown in Figure 5.

This may be a better thing to do for a particular coverage problem, e.g., when we do not care about a target any more after enough accuracy has been achieved *some time in the past,* so that we can move away from it and get closer to some other target.

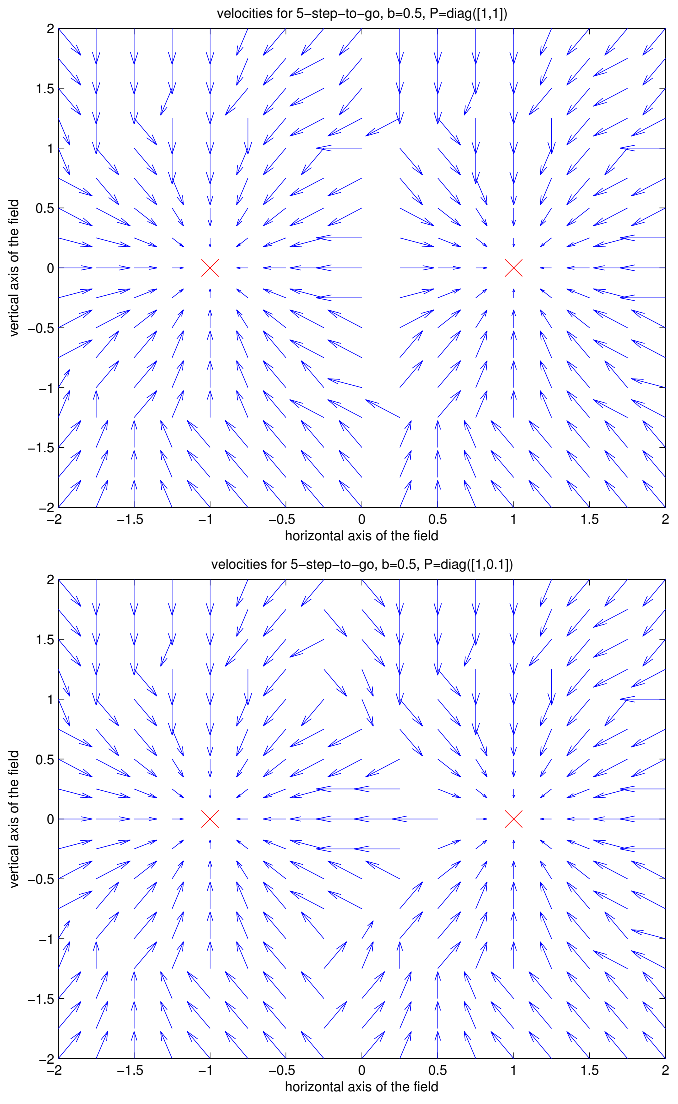

Fig. 3. The 5-step-to-go map for $P_0 = \text{diag}(1, 1)$ and $P_0 = \text{diag}(1, 0.1)$ respectively. Sensor range is $b = 0.5$ and the two targets are marked by (red) crosses.

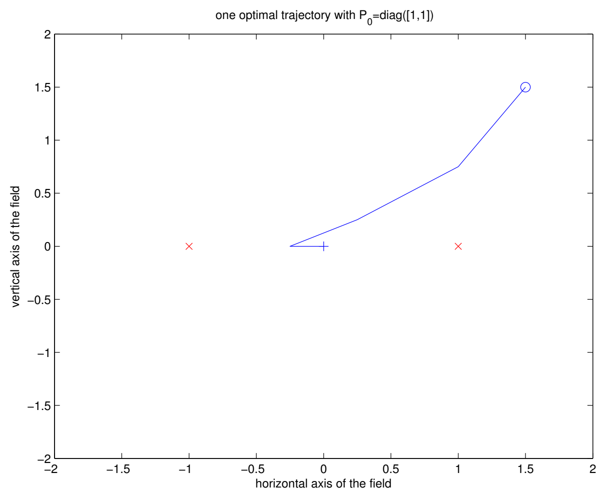

Fig. 4. One optimal trajectory that corresponds to the control law shown in Figure 2.

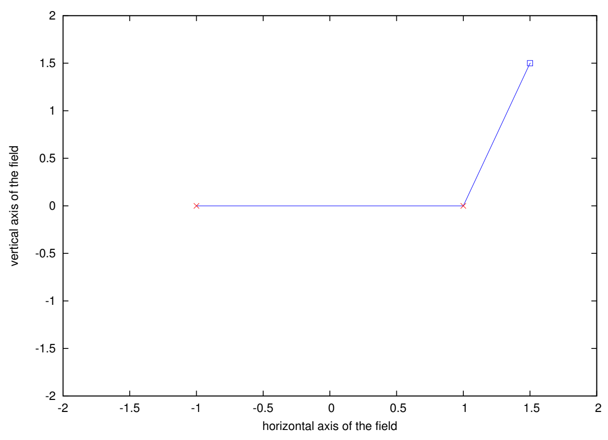

Fig. 5. A simple trajectory that visits the targets sequentially.

In Hussein [2007] modification to the basic algorithm is made such that, if a target has not been visited by a sensor as a result of the optimizing procedure, then additional control logic is activated to command the closest sensor to go and visit this target.

One way to produce such effects, while still using (6) as the performance index, is as follows. We start with N targets, and we follow the optimal trajectory from the N-target-atlas. When the variance of one target drops below a threshold, we remove it from the target list and follow the corresponding $N-1$-target-atlas.

Thus we would have to solve many optimization problems and keep lots of atlases. The optimal trajectories are still produced offline, and we can think of ways to represent them in a more succinct manner.

We may also want to think about directly defining different performance indices. To avoid the burden of notations, we will describe in words possible candidates. Suppose we define that, when the accuracy of the estimate of a target exceeds certain threshold, this target becomes a "treasure" that has just been collected (and thus cannot be re-collected in the future). Then we can try to maximize

$$J \triangleq \text{total treasures collected in a given time},$$

or maximize

$$J \triangleq \text{average treasures collected per time step},$$

or minimize

$$J \triangleq \text{time elapsed to collect all treasures},$$

or minimize

$$J \triangleq \text{distance traveled to collect all treasures},$$

or possibly some other performance indices.

So far we have represented the field to be surveyed by a finite set of "targets." This is of course only a discretization strategy. The ultimate objective is the estimation of the field itself. Thus, when we formulate an optimal control problem under a performance index, we may also want to take into account how well the formulation can be adapted to the situation with a finer or coarser discretization, or with a "weighting" by a probability distribution (to denote non-uniform interest in knowing the field).

6. CONCLUSIONS

Our objective in this paper is to examine the problem formulation in Hussein [2007], which deals with the problem of how a network of mobile sensors with limited range should optimally traverse and estimate a spatially-decoupled scalar field. The published result proposed that the optimal trajectories be generated by an online procedure that minimizes the trace of the instantaneous covariance of the estimation error obtained from Kalman Filtering, using a finite set of admissible control inputs. We extend the formulation by observing that the procedure can be performed offline, that the cost function can be defined over a finite horizon, and that the set of control inputs can be a continuum. Solution to the extended problem are obtained by dynamic programming, and we use two simple examples to illustrate the nature of the optimal trajectories thus obtained. We point out that these globally optimal trajectories may seem to be stuck in "local minima," which points to the need of questioning the suitability of the performance index used. If a practical coverage control problem dictates that the accuracy of the estimation of a target need not be maintained all the time once it has reached certain level, then alternative performance indices can be devised, and further studies based upon them should be conducted.

ACKNOWLEDGEMENTS

The first author would like to thank Professor I. I. Hussein for helpful discussions. We would also like to thank the reviewers for constructive suggestions that enhance the paper.

REFERENCES

A.Tiwari, M. Jun, D. E. Jeffcoat, and R.M. Murray. Analysis of dynamic coverage problem using kalman filters for estimation. In *Proceedings of the 16th IFAC World Congress*, 2005.

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2005.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

C.G. Cassandras and W. Li. Sensor networks and cooperative control. *European Journal of Control*, 11(4-5): 436–463, 2005.

T.H. Chung, V. Gupta, J.W. Burdick, and R.M. Murray. On a decentralized active sensing strategy using mobile sensor platforms in a network. In *IEEE Conference on Decision and Control*, volume 2, pages 1914 – 1919, 2004.

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

I. I. Hussein. A Kalman filter-based control strategy for dynamic coverage control. In *Proceedings of the 2007 American Control Conference*, pages 3271–3276, 2007.