2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Parallel Grid-based Recursive Bayesian Estimation Using GPU for Real-time Autonomous Navigation

Tomonari Furukawa, Benjamin Lavis and Hugh F. Durrant-Whyte

Abstract— **This paper presents the parallelization of grid-based recursive Bayesian estimation (RBE) using a graphics processing unit (GPU) for real-time control of autonomous vehicles. Although the grid-based method has been effectively used for autonomous search due to its ability to represent search space explicitly, heavy computational load has been a bottleneck for real-time application similarly to other non-Gaussian RBE techniques. The proposed RBE, which parallelizes grid-wise computations using GPU upon the analysis of mathematical operations, removes sequential processes and accelerates RBE significantly. Numerical examples have first demonstrated the validation of the proposed RBE and investigated its performance through parametric studies. The proposed RBE was then applied to the cooperative search by autonomous unmanned ground vehicles (UGVs), and its real-time capability has been demonstrated.**

I. Introduction

Search and tracking are two fundamental tasks for sensor platforms engaging on objects of interest [4]. When objects are not detected, sensor platforms must search for objects. When a object is found, tracking, broadly including staying for a static object, takes place to engage on the object. Since the actions of the platforms are based on uncertain prior and empirical information, it is adequate to estimate the state of the object within the framework of RBE [11]. Due to the difference in operation, RBE techniques for search and tracking however used different numerical techniques in the past.

Techniques effective for the search operation include the grid-based method [1], the Gaussian quadrature method [12], the Monte Carlo method [6] and their variants, although the Gauss quadrature method is only applicable to one-dimensional space. These methods locate nodes for numerical integration called grid points, Gauss points and particles, respectively, in a different manner, but the nodes are commonly spread over the object space to maintain the configurations of the space to search and the non-Gaussian belief on each object over the search space.

The techniques suitable for tracking, such as the extended Kalman Filter (EKF) [10], the sequential Monte Carlo (SMC) methods also known as the particle filter methods [7], the sequential Quasi-Monte Carlo (SQMC) methods [5] and their variants, result from their computational efficiency in describing the belief of the observable object. As a result, the areas considered in the object space are limited to those where objects are observable.

While most of the past work treated search and tracking independently, continuous work by the authors has shown the unified formulation of search and tracking in the RBE framework [2], [3], [4]. In their work, an observation likelihood is formulated in a unified manner such that the belief can be maintained regardless of whether the object is detectable, but the use of the grid-based method, which is inevitable due to the need for maintaining the configuration of target space for search, makes the computational time extremely expensive and has not allowed its application to real-world problems. One may represent the belief more coarsely to solve this problem, but it is not adequate in tracking since the belief is often represented sharply in a small region.

This paper presents the parallelization of grid-based RBE using a GPU for real-time control of autonomous vehicles. Since the grid-based method involves grid-wise computation for the core processes of update and prediction, it is expected that the parallelized grid-based RBE performs fast and possibly allows real-time RBE even for practical problems requiring a large number of grid cells. This paper describes not only the parallelization but also the analytical estimation of computational speedup as one may need to find parameters for the grid-based method *a priori* such that the resulting parallel RBE can be achieved in real time.

The paper is organized as follows. The following section reviews the RBE that predicts and updates the belief of each target as well as the grid-based method. Section III presents the proposed parallel RBE using GPU, which enables real-time RBE in real-world problems. Section IV demonstrates the efficacy of the proposed RBE through numerical examples and conclusions are summarized in the final section.

II. Recursive Bayesian Estimation

RBE forms a basis to the maintenance of belief on a dynamically moving object. Given prior belief as well as knowledge on its motion and observations, RBE updates and maintains the belief using two recursive processes, prediction and correction. Let the prior belief on the object be $p\left(\tilde{\mathbf{x}}_0^o\right)$ where $p\left(\cdot\right)$ is a probability density function. Prediction uses knowledge on motion written in the form of a probabilistic Markov motion model $p\left(\mathbf{x}_{k+1}^o|\mathbf{x}_k^o\right)$ and updates the belief

This work was supported by US RDECOM/DARPA grant (FA23896-08-1-4129) and ARC Centre of Excellence for Autonomous Systems

T. Furukawa is with Department of Mechanical Engineering, Virginia Tech, Blacksburg, VA tomonari@vt.edu

B. Lavis is with Department of Mechanical Engineering, University of California at Berkeley, Berkeley, CA b.lavis@berkeley.edu

H.F. Durrant-Whyte is with Australian Centre for Field Robotics, The University of Sydney, Sydney 2006 Australia hugh@acfr.usyd.edu.au

in time by using Chapman-Kolmogorov equation:

$$p\left(\mathbf{x}_k^o|^{s_i}\tilde{\mathbf{z}}_{1:k-1},\tilde{\mathbf{x}}_{1:k-1}^{s_i}\right) = \int_{\mathcal{X}^o} p\left(\mathbf{x}_k^o|\mathbf{x}_{k-1}^o\right)\, p\left(\mathbf{x}_{k-1}^o|^{s_i}\tilde{\mathbf{z}}_{1:k-1},\tilde{\mathbf{x}}_{1:k-1}^{s_i}\right)\, d\mathbf{x}_{k-1}^o, \tag{1}$$

where $p\left(\mathbf{x}_{k-1}^o|^{s_i}\tilde{\mathbf{z}}_{1:k-1},\tilde{\mathbf{x}}_{1:k-1}^{s_i}\right) = p\left(\tilde{\mathbf{x}}_0^o\right)$ when $k = 0$, and $\tilde{\mathbf{x}}_{1:k}^{s_i} \equiv \left\{\tilde{\mathbf{x}}_\kappa^{s_i}|\forall\kappa\in\{1,...,k\}\right\}$ and ${}^{s_i}\tilde{\mathbf{z}}_{1:k} \equiv \left\{{}^{s_i}\tilde{\mathbf{z}}_\kappa|\forall\kappa\in\{1,...,k\}\right\}$ represent a sequence of states of the sensor platform s_i and a sequence of observations by the sensor platform from time step 1 to time step k respectively. Note here that $\tilde{(\cdot)}$ represents an instance of variable $(\cdot)$.

Correction, on the other hand, constructs an observation likelihood $l^{s_i}\left(\mathbf{x}_k^o|^{s_i}\tilde{\mathbf{z}}_k,\tilde{\mathbf{x}}_k^{s_i}\right)$ from observation ${}^{s_i}\tilde{\mathbf{z}}_k$ and updates the belief in measurement using Bayes theorem:

$$p\left(\mathbf{x}_k^o|^{s_i}\tilde{\mathbf{z}}_{1:k},\tilde{\mathbf{x}}_{1:k}^{s_i}\right) = \frac{l^{s_i}\left(\mathbf{x}_k^o|^{s_i}\tilde{\mathbf{z}}_k,\tilde{\mathbf{x}}_k^{s_i}\right)p\left(\mathbf{x}_k^o|^{s_i}\tilde{\mathbf{z}}_{1:k-1},\tilde{\mathbf{x}}_{1:k}^{s_i}\right)}{\int_{\mathcal{X}^o} l^{s_i}\left(\mathbf{x}_k^o|^{s_i}\tilde{\mathbf{z}}_k,\tilde{\mathbf{x}}_k^{s_i}\right)p\left(\mathbf{x}_k^o|^{s_i}\tilde{\mathbf{z}}_{1:k-1},\tilde{\mathbf{x}}_{1:k}^{s_i}\right)\, d\mathbf{x}_{k-1}^o}, \tag{2}$$

or

$$p\left(\mathbf{x}_k^o|^{s_i}\tilde{\mathbf{z}}_{1:k},\tilde{\mathbf{x}}_{1:k}^{s_i}\right) = \frac{g_c\left(\mathbf{x}_k^o|^{s_i}\tilde{\mathbf{z}}_{1:k},\tilde{\mathbf{x}}_{1:k}^{s_i}\right)}{\int_{\mathcal{X}^o} g_c\left(\mathbf{x}_k^o|^{s_i}\tilde{\mathbf{z}}_{1:k},\tilde{\mathbf{x}}_{1:k}^{s_i}\right)d\mathbf{x}_{k-1}^o}, \tag{3}$$

where

$$g_c\left(\mathbf{x}_k^o|^{s_i}\tilde{\mathbf{z}}_{1:k},\tilde{\mathbf{x}}_{1:k}^{s_i}\right) = l^{s_i}\left(\mathbf{x}_k^o|^{s_i}\tilde{\mathbf{z}}_k,\tilde{\mathbf{x}}_k^{s_i}\right)p\left(\mathbf{x}_k^o|^{s_i}\tilde{\mathbf{z}}_{1:k-1},\tilde{\mathbf{x}}_{1:k}^{s_i}\right). \tag{4}$$

Note that the likelihood takes two different forms depending on the detectability of the object and is given by

$$l^{s_i}\left(\mathbf{x}_k^o|^{s_i}\tilde{\mathbf{z}}_k^o,\tilde{\mathbf{x}}_k^{s_i}\right) = \begin{cases} 1 - P_D\left(\mathbf{x}_k^o|\tilde{\mathbf{x}}_k^{s_i}\right) & \nexists\, {}^{s_i}\tilde{\mathbf{z}}_k^o \in {}^{s_i}\mathcal{X}_D^o \\ p\left(\mathbf{x}_k^o|^{s_i}\tilde{\mathbf{z}}_k^o,\tilde{\mathbf{x}}_k^{s_i}\right) & \exists\, {}^{s_i}\tilde{\mathbf{z}}_k^o \in {}^{s_i}\mathcal{X}_D^o \end{cases} \tag{5}$$

where ${}^{s_i}\mathcal{X}_D^o$ is the "detectable region" of the sensor platform s_i, which describes the region within which the sensor confidently finds the object o, i.e., ${}^{s_i}\mathcal{X}_D^o = \left\{\mathbf{x}_k^o|\epsilon^o < P_D\left(\mathbf{x}_k^o|\mathbf{x}_k^{s_i}\right) \le 1\right\} \subset {}^{s_i}\mathcal{X}_O^o$, where ϵ^o is a positive threshold value which determines the detection of the object. Depending on whether there exists an observed object within the detectable region, the upper and lower formulas provide likelihoods useful for search and tracking, respectively. If the observed object state is not within the detectable region, the observation is insignificant. As a result, the likelihood is defined in terms of the negation of the POD. If the observation is within the detectable region, the likelihood can be defined as a probability density having its peak around the observed object state.

III. Parallel Recursive Bayesian Estimator

A. Data Parallelization

The approach chosen to improve the computational efficiency of the grid-based RBE is the form of parallelization known as data parallelization. Data parallelization is the process of converting processes which operate on each piece of data individually in a sequence, to one that operates on groups of data at once, which can result in a dramatic reduction in computation time. The RBE, described in Section II, has multiple processes which lend themselves to computational speed-up by parallelization. However, the numerical implementation should be analyzed to determine which of these are likely to produce meaningful improvements to the computational speed.

The potential computational speedup produced by parallelizing operations in a process can be estimated using Amdahl's law, which states that the maximum speedup achievable by parallelization is:

$$S = \frac{t_p}{t_o} = \frac{1}{(1-P) + \frac{P}{N}} \tag{6}$$

where S is the computational speedup, t_p is the iteration time for the parallelized algorithm, t_o is the iteration time for the original algorithm, P is the proportion of the process which is parallelizable and N is the number of parallel processors available to perform the parallelization. This formula is the theoretical maximum of the computational speedup and does not take into account the bottlenecking effect of communications between large numbers of threads; however it gives a good estimate of the effectiveness of parallelizing each operation in the RBE.

To enable this estimation, each operation in the RBE must be examined to estimate the number of floating point operations. These results can be used to estimate the proportion of the algorithm that can be parallelized and allow the maximum potential speedup to be calculated using Equation (6).

B. Parallelization of Recursive Bayesian Estimation

1) Update: The correction operation requires the computation of Equation (2). Given the predicted belief $p_{\mathbf{x}_k^o}^{i_x,i_y}\left({}^{s_i}\tilde{\mathbf{z}}_{1:k-1},\tilde{\mathbf{x}}_{1:k}^{s_i}\right)$ and the observation likelihood $l_{\mathbf{x}_k^t}^{i_x,i_y}\left({}^{s_i}\tilde{\mathbf{z}}_k,\tilde{\mathbf{x}}_k^{s_i}\right)$, the belief of the grid cell $[i_x,i_y]$ can be corrected as

$$p_{\mathbf{x}_k^o}^{i_x,i_y}\left({}^{s_i}\tilde{\mathbf{z}}_{1:k},\tilde{\mathbf{x}}_{1:k}^{s_i}\right) = \frac{q_{\mathbf{x}_k^o}^{i_x,i_y}(\cdot)}{A_c\sum_{\alpha=1}^{n_x}\sum_{\beta=1}^{n_y} q_{\mathbf{x}_k^o}^{\alpha,\beta}(\cdot)}, \tag{7}$$

where A_c is the area of a grid cell and

$$q_{\mathbf{x}_k^o}^{i_x,i_y}\left({}^{s_i}\tilde{\mathbf{z}}_{1:k},\tilde{\mathbf{x}}_{1:k}^{s_i}\right) = l_{\mathbf{x}_k^o}^{i_x,i_y}\left({}^{s_i}\tilde{\mathbf{z}}_k,\tilde{\mathbf{x}}_k^s\right)p_{\mathbf{x}_k^o}^{i_x,i_y}\left({}^{s_i}\tilde{\mathbf{z}}_{1:k-1},\tilde{\mathbf{x}}_{1:k}^{s_i}\right). \tag{8}$$

In the numerical implementation, the correction operation can be broken down into three steps:

1) Calculate $q_{\mathbf{x}_k^o}^{i_x,i_y}\left({}^{s_i}\tilde{\mathbf{z}}_{1:k},\tilde{\mathbf{x}}_{1:k}^{s_i}\right)$ by multiplying the predicted belief $p_{\mathbf{x}_k^o}^{i_x,i_y}\left({}^{s_i}\tilde{\mathbf{z}}_{1:k-1},\tilde{\mathbf{x}}_{1:k}^{s_i}\right)$ by the observation likelihood $l_{\mathbf{x}_k^o}^{i_x,i_y}\left({}^{s_i}\tilde{\mathbf{z}}_k,\tilde{\mathbf{x}}_k^{s_i}\right)$;
2) Sum $\sum_{\alpha=1}^{n_x}\sum_{\beta=1}^{n_y} q_{\mathbf{x}_k^o}^{\alpha,\beta}\left({}^{s_i}\tilde{\mathbf{z}}_{1:k},\tilde{\mathbf{x}}_{1:k}^{s_i}\right)$ and multiply the sum by A_c; and
3) Calculate $p_{\mathbf{x}_k^o}^{i_x,i_y}\left({}^{s_i}\tilde{\mathbf{z}}_{1:k},\tilde{\mathbf{x}}_{1:k}^{s_i}\right)$ by dividing $q_{\mathbf{x}_k^o}^{i_x,i_y}\left({}^{s_i}\tilde{\mathbf{z}}_{1:k},\tilde{\mathbf{x}}_{1:k}^{s_i}\right)$ by $\sum_{\alpha=1}^{n_x}\sum_{\beta=1}^{n_y} q_{\mathbf{x}_k^o}^{\alpha,\beta}\left({}^{s_i}\tilde{\mathbf{z}}_{1:k},\tilde{\mathbf{x}}_{1:k}^{s_i}\right)$.

Out of the three steps, Steps 1 and 3 are the operations whose calculations can be conducted independently for every grid cell. Step 1 requires a multiplication operation for each grid cell in the target space, thus for the search space with n_g cells, there are n_g floating point operations per iteration. Steps 2 and 3 require additional floating operations through summation and division, which each totals n_g at most. Thus, the maximum number of floating point operations is estimated as $3n_g$.

2) Prediction: The prediction operation requires the numerical evaluation of the Chapman-Kolmogorov equation described in Equation (1). Given the current belief $p^{i_x,i_y}_{\mathbf{x}^o_k}\left({}^{s_i}\tilde{\mathbf{z}}_{1:k-1}, \tilde{\mathbf{x}}^{s_i}_{1:k-1}\right)$ as well as the Malkov motion model $p^{i_x,i_y}_{\mathbf{x}^o_k|\mathbf{x}^o_{k-1}}$ constructed in the matrix of size $I_x \times I_y$ as the convolution kernel, the belief can be predicted as

$$\begin{aligned} & p^{i_x,i_y}_{\mathbf{x}^o_k}\left({}^{s_i}\tilde{\mathbf{z}}_{1:k-1}, \tilde{\mathbf{x}}^{s_i}_{1:k-1}\right) \\ &= p^{i_x,i_y}_{\mathbf{x}^o_{k-1}}\left({}^{s_i}\tilde{\mathbf{z}}_{1:k-1}, \tilde{\mathbf{x}}^{s_i}_{1:k-1}\right) \otimes p^{i_x,i_y}_{\mathbf{x}^o_k|\mathbf{x}^o_{k-1}} \\ &= \sum_{\alpha=0}^{I_x}\sum_{\beta=0}^{I_y} p^{\alpha,\beta}_{\mathbf{x}^o_k|\mathbf{x}^o_{k-1}} p^{i_x-\alpha,i_y-\beta}_{\mathbf{x}^o_{k-1}}\left({}^{s_i}\tilde{\mathbf{z}}_{1:k-1}, \tilde{\mathbf{x}}^{s_i}_{1:k-1}\right), \end{aligned} \quad (9)$$

where $\otimes$ indicates the convolution of the current belief with the Markov motion model. The equation first shows that the prediction operation at each grid cell can be conducted independently. This means that the prediction operation is completely parallelizable. However, this equation also shows that the computation time for prediction is largely dominated by the size of the convolution kernel. The convolution kernel must be able to capture the motion of the object but needs to be small to perform the prediction operation fast.

Figure 1 shows how the number of prediction steps varies with kernel radii and grid size where the kernel radius is defined as the number of grid cells the kernel matrix extends from the center cell to the outer row or column, not including the center cell itself. For a suitable kernel radius r_k, the number of multiplications required to calculate each grid cell is $(2r_k + 1)^2$. For each grid cell, these multiplications are summed together, and thus there are a further $(2r_k+1)^2$ floating point operations per cell. In total, the number of floating point operations in the Prediction step is $n_g(2(2r_k+1)^2)$. The figure shows clearly the exponential increase in operations with kernel radii. There are also other factors, which will determine the kernel size in the grid-based representation, including the allowable memory size and the transfer rates between different processors and memory locations.

C. Estimated Computational Speedup

Table I summarizes the approximate number of floating point operations in each major operation of the RBE. For a given number of grid cells, kernel radius and number of parallel processors available on the GPU the maximum attainable speedup may now be estimated with Equation (6) which has been plotted in Figure 2. Note that the number of grid cells is a factor of each of the total number of floating point operations in these processes, thus the speedup which calculates the proportion of time spent in each process,

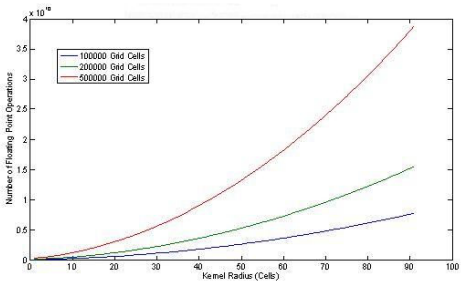


Fig. 1. Number of floating point operations in prediction

will be theoretically independent of grid size. For example, for a grid with 1,000,000 cells, a kernel radius of 8 and using an NVidia G80 chipset GPU with 32 coprocessors, the maximum speedup attainable is approximately 27.58 for the prediction and 1.00 for the update step.

TABLE I
COMPUTATIONAL REQUIREMENT FOR GRID-BASED METHOD

Operation	Number of floating point operations
Update	$3n_g$
Prediction	$n_g\left\{2\left(2r_k+1\right)^2\right\}$

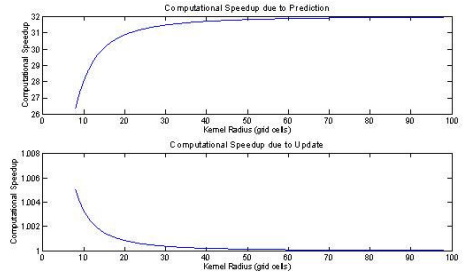


Fig. 2. Computational Speedup for Grid-based Representation

When computed linearly using a CPU, the computational time required for an iteration of RBE at 4740 grid cells was 1.27 seconds, which would mean that, by parallelizing the prediction step alone, the iteration time could be reduced to approximately 0.046 seconds with a 96.4% reduction in computation time. Although this is a theoretical limit, the parallelization of the prediction step with the grid-based method has very low overhead due to memory access issues. This is because the data is stored coherently in a matrix in the grid-based method, which means that the actual improvement in computation time should be close to this estimated limit.

IV. NUMERICAL EXAMPLES

This section presents results of a series of qualitative and quantitative tests carried out to investigate real-time capability of the system implementing the proposed parallel RBE. The experimental setup used in the tests is a parallel

RBE system which utilizes an NVidia GPU to achieve real-time performance. The setup specifications which have been chosen is shown in Table II.

TABLE II
TEST COMPUTER SYSTEM SPECIFICATIONS

Processor	Intel Core2Duo, 2.4GHz
Memory	2.0GB RAM
OS	Windows XP Pro, Service Pack 2
GPU	NVidia 8800GTS, 640MB onboard RAM, stream processors

A. Validation

1) Test 1: Validation of parallelized RBE: Test 1 was aimed at validating the proposed parallelized RBE by comparing it with results from the sequential RBE which runs on a CPU only. As RBE is recursive, any errors in an iteration of the parallel RBE will rapidly accumulate. Both the RBEs were commenced with the same target belief. The target grid size was varied to remove its effect on the results.

The results for the mean squared error are shown in Figure 3. These indicate that there is a small random error between the parallel and sequential RBEs, with a mean value of 1.0×10^{-6}. This is well within the inherent noise of the models being used in the iteration, and is of the correct order of magnitude to suggest that it is due to the use of single precision floating point operations on the GPU as opposed to the double precision floating point operations on the CPU.

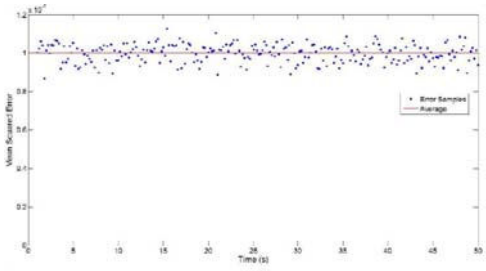
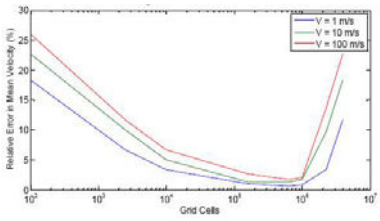


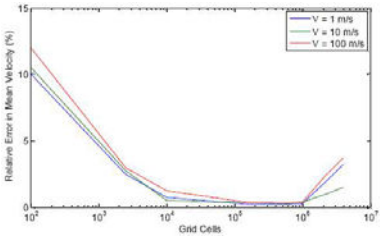
Fig. 3. Mean Squared Error for Parallel Recursive Bayesian Estimation

2) Test 2: Validation of Target Motion Model: The accuracy of the RBE was examined to determine the parameters required for its real-time computation. The size of this convolution kernel will affect both the accuracy of the representation of the targets motion as well as the real-time performance of the system itself. Realistic limits on the size of the convolution kernel need to be established to ensure that this accuracy is within the noise of the system to be implemented. In this test, the velocity characteristics of the motion model of the target were varied as well as the kernel size used to represent them.

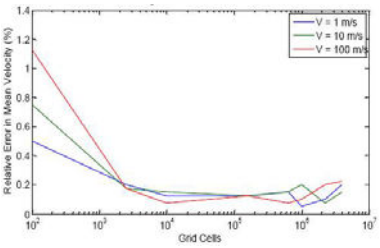
Figure 4 shows the results of the parallel RBE in tests performed at different grid sizes and convolution kernel sizes with different input target motions models. The belief at the end of each time step was analyzed by calculating the weighted mean position of the peak. The results well show that the relative error is well below 1% across grid sizes from 10,000 to 1,000,000 grid cells when the kernel size is large or an appropriate velocity is used. However, the error is large if the kernel size is small and the target velocity is large. Note that as the number of grid cells increase to above 1,000,000 the convolution kernels become less accurate as the number of grid cells between the center of the kernel and the peak in the kernel becomes larger.



(a) kernel radius 8



(b) kernel radius 32



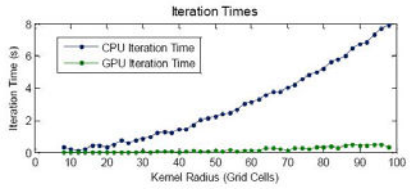
(c) kernel radius 96

Fig. 4. Parallel recursive Bayesian estimation

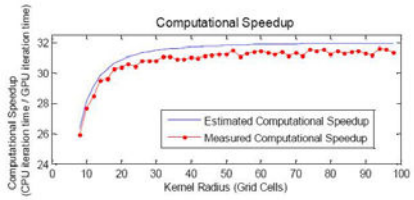
B. Computational Speedup by Parallelization

Since the parallelization achieves a speedup in computation time, the iteration time for the parallelized and the sequential RBEs was measured at different kernel radii and is shown in Figure 5(a) whereas the computational speedup for the experimental iteration time is shown along with the estimation of the maximum speedup in Figure 5(b). The resulting improvement in the iteration rate appears to be fairly close to the theoretical limit to which data parallelization could achieve with the NVidia GPU used. On average across the range of convolution kernels tested, the proposed parallel RBE has been found to achieve over 95% of the estimated maximum improvement in computational speed.

This reconfirms that the high arithmetic intensity of the prediction step has resulted in the considerable improvement, and that the additional overhead due to memory access and idle threads created for coalesced data reads was small enough not to affect the result significantly.



(a) Iteration time vs kernel radius



(b) Computational speedup vs kernel radius

Fig. 5. Iteration time and computational speedup

C. Real-time Performance

The objective of this paper is to develop a technique of performing RBE at an appropriate rate to be considered real-time so that it can be utilized in real search and tracking operations. Parametric studies were performed to identify the real-time performance of the developed parallel RBE system.

1) Search Space Grid Size: One of the most important parameter in determining the real-time performance of the system is the size of the grid space itself. As the number of grid cells increases, the number of floating point and memory operations, and in turn the iteration time, increases linearly. The iteration time for the parallel RBE system was measured at different sizes of grid spaces and the results are shown in Figure 6. The system shows reasonable real-time performance over this range of grid cells. Although the iteration time increases linearly with total number of grid cells, even at the largest grid size tested, 1,000,000, the iteration time is in the order of 0.1 seconds.

2) Number of Sensor Platforms and Sensor Range: Previous tests of the real-time performance were carried out with a single sensor platform with a typical search range, however the impact with which the characteristics and quantities searching sensor platforms have on the real-time performance should also be analyzed. Figure 7 shows the resulting iteration time of the parallel RBE system for different numbers of sensor vehicles and with sensor ranges of 5m, 50m and 100m. The results indicate that the increase in time with respect to the number of sensors is linear and

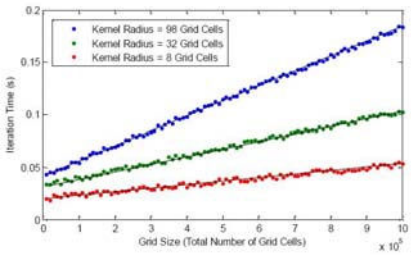


Fig. 6. Iteration time vs grid size

not significant. Even for sensor ranges as large as 100m, the iteration time for 100 such sensor platforms is only 20% longer than for a single sensor platform. These results indicate that the real-time RBE system will still be able to iterate at real-time speeds for problems involving a number of sensor platforms with considerable sensor ranges.

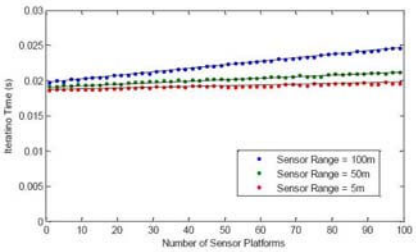


Fig. 7. Real-time performance with multiple sensor platforms

D. Autonomous Cooperative Search by UGVs

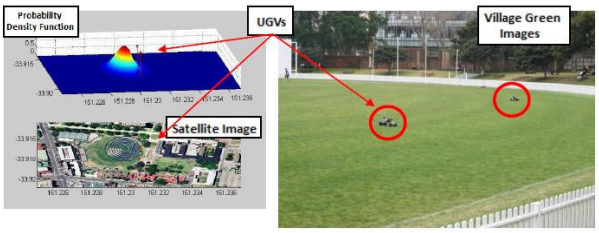
The parallel RBE system was finally used with physical autonomous UGVs for cooperative search. The UGVs were designed to transmit all observations so that the computer at the base station can perform RBE with a GPU and transmit control actions to the UGVs.

The sequence of images in Figure 8 present the target belief updated using the parallel RBE system. The figure also shows a two-dimensional contour plot of the belief overlayed on the satellite image to illustrate correlation between the UGV positions in the search space and its position in the real world. The results illustrate the ability of the system to operate not only real-time RBE by receiving UGV positions and but also perform real-time cooperative control by transmitting waypoints to the autopilots on the UGVs.

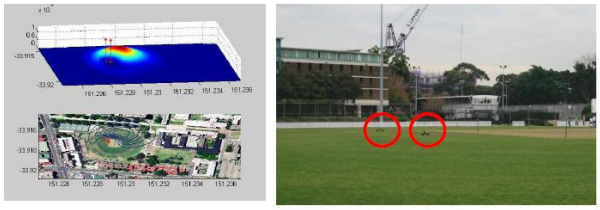
Figure 9(a) shows the iteration times for the parallel RBE as well as that for communication and the total iteration time during experiments whereas the ratio of the iteration time for the parallel RBE to the total iteration time is plotted in Figure 9(b). Even at the larger grid size of 1,000,000 cells, the parallel RBE takes less than 35% of the total iteration time and has been found to allow real-time RBE without much influence.

V. Conclusion and Future Work

The parallelization of grid-based RBE using a GPU for real-time control of autonomous vehicles has been presented.



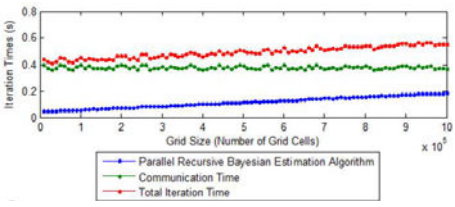
(a) 0 second



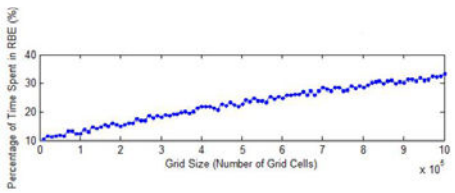
(b) 60 seconds

(c) 120 seconds

Fig. 8. Cooperative search by UGVs



(a) Iteration time vs grid size



(b) Percentage vs grid size

Fig. 9. Iteration time and percentage

Both the update and prediction process have been parallelized, and because of the heavy computational load of prediction when the kernel radius is large, the computational speedup has been estimated analytically. Numerical examples have first validated the performance of the proposed RBE and further investigated its effectiveness through parametric studies. The proposed RBE was then applied to the cooperative search by autonomous UGVs, and its real-time capability has been demonstrated.

The current study is merely the first step for the application of the parallel RBE, and various extensive studies are ongoing. One of the ongoing theoretical studies is the incorporation of a frontier-based exploration technique that changes the configuration of the target space based on the belief. To further demonstrate its validity and efficacy in real world applications, the parallel RBE is currently being applied to the cooperative control of multiple UGVs and UAVs.

REFERENCES

[1] Bergman, N. *Recursive Bayesian Estimation Navigation and Tracking Applications*, Ph.D Dissertation, Linkopings University, 1999.

[2] Bourgault, F., Goktogan, A., Furukawa, T. and Durrant-Whyte, H. F., "Coordinated Search of a Lost Target in a Bayesian World," *Journal of Advanced Robotics*, pp. 187-195, 2004.

[3] Bourgault, T., Furukawa, T. and Durrant-Whyte, H. F., "Optimal Search for a Lost Target in a Bayesian World," Eds. S. Yuta and H. Asama, Field and Service Robots IV, Springer Tracts in Advanced Robotics (STAR), Springer-Verlag, Vol. 24, pp. 209-222, 2006.

[4] Furukawa, T., Bourgault, F., Lavis, B. and Durrant-Whyte, H.F., "Bayesian Search-and-Tracking Using Coordinated UAVs for Lost Targets," *2006 IEEE International Conference on Robotics and Automation*, Orlando, May 14-18, 2006, pp. 2521-2526, 2006.

[5] Guo, D. and Wang, X., "Quasi-Monte Carlo Filtering in Nonlinear Dynamic Systems," *IEEE Transactions on Signal Processing*, 54(6), pp. 2087-2098, 2006.

[6] Hammersley, J. M. "Monte Carlo Methods for Solving Multivariable Problems," *Ann. New York Acad. Sci.*, 86, pp. 844-874, 1960.

[7] Liu, J.S. and Chen, R., "Sequential Monte Carlo Methods for Dynamic Systems," *Journal of the American Statistical Association*, 93(443), pp. 1032-1044, 1998.

[8] Peraire, J., Vahdati, M., Morgan, K., Zienkiewicz, O.C., "Adaptive remeshing for compressible flow computations," *J. Comp. Phys.*, 72(2), pp. 449-466, 1987.

[9] Sato, M., "Theory of hyperfunctions," *Int. J. Fac. Sci.*, Univ. Tokyo, Sec. I, 8(1) 139-193, 1959.

[10] Sorenson, H.W. Ed., *Kalman Filtering: Theory and Application*, New York: IEEE, 1985.

[11] Tarantola, A., *Inverse Problem Theory and Methods for Model Parameter Estimation*, Society for Industrial and Applied Mathematics, Philadelphia, 2005.

[12] Whittaker, E. T. and Robinson, G. "Gauss's Formula of Numerical Integration," *The Calculus of Observations: A Treatise on Numerical Mathematics*, 4th ed., New York: Dover, pp. 152-163, 1967.

[13] Zienkiewicz, O. C. and Taylor, R. L., *The Finite Element Method: Its Basis and Fundamentals*, Sixth Edition, Elsevier Butterworth-Heinemann, Oxford, 2005.