Parallel Implementation of a Bacterial Potential Field Algorithm for Robot Path Planning

Ulises Orozco-Rosas[1,2], Oscar Montiel[2], Roberto Sepúlveda[2], Antonio S. Montemayor[3], and Juan J. Pantrigo[3]
ulises.orozco@cetys.mx, {oross, rsepulvedac}@ipn.mx, {antonio.sanz, juanjose.pantrigo}@urjc.es
[1]Centro de Enseñanza Técnica y Superior (Mexico), [2]Instituto Politécnico Nacional (Mexico), [3]Universidad Rey Juan Carlos (Spain)

Introduction

In this work, a parallel implementation on GPU of the bacterial potential field (BPF) algorithm for robot path planning is presented. The BPF algorithm proposed in [1] is employed to ensure a feasible, smooth, and safe path for robot navigation.

The BPF algorithm employs concepts from the artificial potential field (APF) method, mathematical programming, and meta-heuristic to solve efficiently the robot path planning problem.

The path planning problem in mobile robots is one of the most computationally intensive tasks, in that sense, heterogeneous computing helps to gain performance. By means of GPUs is possible to process data-intensive tasks (evaluation of solutions) efficiently.

Algorithm

The BPF algorithm makes use of the APF method proposed by Khatib [2] with a bacterial evolutionary algorithm (BEA) to obtain an enhanced flexible path planner.

The BEA introduces two operators inspired by the microbial evolution phenomenon, bacterial mutation and genetic transfer. The labor of the **bacterial mutation** operator is to optimize the chromosome of a single bacterium, and the **genetic transfer** operator provides the transfer of information between the bacteria in the population [3].

The BPF algorithm uses the start, goal and obstacle positions as features to obtain a sequence of objective points (path) that the robot must attain. Hence, the BPF algorithm achieves the task of path planning generation, with the particular characteristic that it provides an optimal or nearly optimal reachable set of configurations (path) if it exists.

The flowchart (right) shows the BPF algorithm and its parallel process implemented on GPU (green blocks). The process starts with a creation of a random population of bacteria $P(t)$, each bacterium (possible solution) is codified with the values of the proportional gains, attraction k_a and repulsion k_r required to generate a feasible path. For the parallel evaluation on GPU, where each bacterium is evaluated. First, the **potential field** $U(q)$ computation is performed,

$$U(q) = \frac{1}{2}\left[k_a(q-q_f)^2 + k_r\left(\frac{1}{\rho} - \frac{1}{\rho_0}\right)^2\right]$$

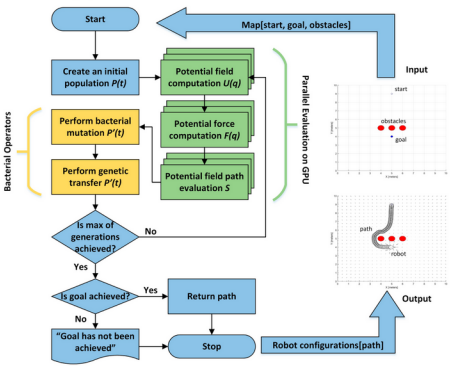
Then, the generalized **potential force** $F(q)$ which is used to drive the robot is obtained by the negative gradient of the potential field $U(q)$,

$$F(q) = -\nabla U(q)$$

Last, for the parallel evaluation on GPU, the potential field **path evaluation** S is performed,

$$S = \sum_{i=0}^{n-1} \|q_{i+1} - q_i\|$$

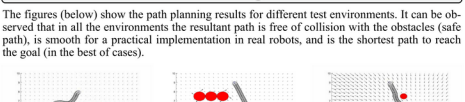
After the parallel evaluation, the bacterial operators are applied to evolve the bacteria in the population $P(t)$. All the process is repeated over and over again until the maximum number of generations has been achieved. If the resultant path successfully achieves the goal in a safe and efficient manner, the path is returned as a result, otherwise a message of "goal has not been achieved" is displayed.



For the parallel implementation on GPU, we have chosen MATLAB-CUDA as a platform to implement the BPF algorithm. We use CUDA kernel integration in the MATLAB application, a kernel (code written in CUDA) called from MATLAB is executed on the GPU to accelerate the evaluation process (potential field computation, potential force computation, and potential field path evaluation) of the BPF algorithm.

Path Planning Results

The figures (below) show the path planning results for different test environments. It can be observed that in all the environments the resultant path is free of collision with the obstacles (safe path), is smooth for a practical implementation in real robots, and is the shortest path to reach the goal (in the best of cases).



Test environment 1 | Test environment 2 | Test environment 3

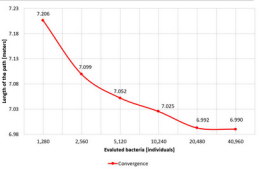
Performance Results

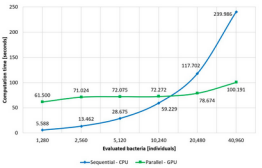
The experiments were achieved using:

- Software: NVIDIA CUDA 7.5; MATLAB R2015A; Ubuntu Trusty distribution of Linux.
- Hardware: Intel i7-5820K CPU@3.30 GHz; NVIDIA GeForce GTX TITAN X.

To evaluate the performance of the parallel BPF implementation on GPU versus the sequential implementation on CPU, we carried out independently thirty tests for each total population evaluated (bacteria).

The convergence plot (red), the average computation time in seconds for the parallel implementation (green) and for the sequential implementation (blue) are shown in the next graphs.





Conclusion

- In this work, we have presented the parallel implementation on GPU of a bacterial potential field (BPF) algorithm for robot path planning.
- The parallel BPF has demonstrated its capability to perform path planning for different environments and population sizes.
- The performance results show that the parallel implementation on GPU accelerates the evaluation process by a factor of 2.4x for the bigger population tested.
- The performance results show that a small population size could guide the BPF algorithm to reasonable solutions (paths) and that a large population size could make the BPF algorithm to spend more computation time to find best solutions.
- Making a compromise between solution quality and computation time, we have found that the best performance on GPU is obtained when the total population evaluated is of 15,000 (bacteria evaluated) or bigger.
- Furthermore, the path planning results demonstrates the efficiency of the parallel BPF algorithm to solve the path planning problem in different scenarios.

References & Acknowledgments

[1]O. Montiel, U. Orozco-Rosas, and R. Sepúlveda.: Path planning for mobile robots using Bacterial Potential Field for avoiding static and dynamic obstacles. Expert Systems with Applications, 42(12), pp. 5177–5191, 2015.
[2]O. Khatib.: Real-time obstacle avoidance for manipulators and mobile robots. The International Journal of Robotics Research, pp. 90-98, 1986.
[3]N.E. Nawa, and T. Furuhashi.: Fuzzy system parameters discovery by bacterial evolutionary algorithm. IEEE Transactions on Fuzzy Systems, 7(5), pp. 608-616, 1999.

We thank to the Mexican National Council of Science and Technology (CONACYT) for supporting our research activities.