memEAPF for Mobile Robot Path Planning on Jetson Platform

Ulises Orozco-Rosas[1], Kenia Picos[1], Antonio S. Montemayor[2], Juan J. Pantrigo[2], and Alfredo Cuesta-Infante[2]
{ulises.orozco, kenia.picos}@cetys.mx, {antonio.sanz, juanjose.pantrigo, alfredo.cuesta}@urjc.es
[1]**CETYS Universidad,** Centro de Excelencia en Innovación y Diseño (CEID), Mexico,
[2]**Universidad Rey Juan Carlos,** Departamento de Informática, Spain

Introduction

In this work, a parallel implementation on the NVIDIA Jetson TX2 of the membrane evolutionary artificial potential field (memEAPF) algorithm for mobile robot path planning is presented. The memEAPF algorithm proposed in [1] is employed to achieve feasible paths, considering minimum path length, safety, and smoothness.

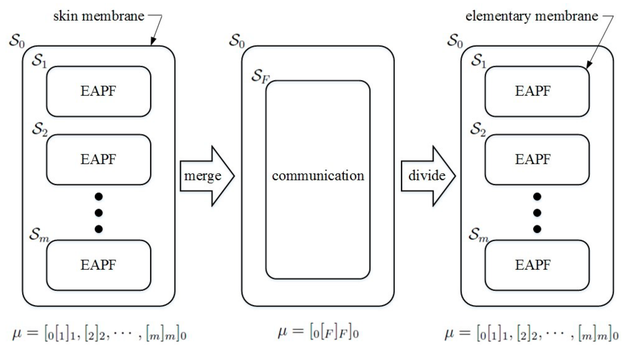
The memEAPF algorithm combines membrane computing [2] with a genetic algorithm, in specific a membrane-inspired evolutionary algorithm with a one-level membrane structure and the artificial potential field method [3] to find the parameters to solve efficiently the robot path planning problem.

The path planning problem in mobile robots is one of the most computationally intensive tasks, in that sense, heterogeneous computing helps to gain performance. Employing GPUs is possible to process data-intensive tasks efficiently, in this work the evaluation of solution paths.

memEAPF algorithm

The memEAPF algorithm for path planning consists of a cell-like P system that evolves a set of parameters required for the artificial potential field (APF) method. These parameters are the attractive proportional gain k_a and the repulsive proportional gain k_r.

The memEAPF algorithm employs a dynamic structure μ with active membranes and rules, such as membrane merger and division, as can be seen in the below figure. The membrane merger is helpful to enhance the information communication among individuals (set of parameters, k_a and k_r), and the membrane division is beneficial to improve the search capability.



The computational process outlined by the above figure consists of three steps: First, each elementary membrane S_i, composed of an evolutionary artificial potential field (EAPF) evolves the individuals. Each individual is codified with a set of parameters. The purpose of this first stage is to find the best individual in each elementary membrane S_i.

Second, all the elementary membranes S_i, merge into one membrane S_F, containing all the individuals. Communication rules are applied, which first separate the best individual of each elementary membrane with the purpose of finding the global best individual (k_a and k_r), and further send out into the skin membrane S_0 a copy of this global best individual to preserve the current global best solution.

The communication continues in the merged membrane S_F to exchange information among the elementary membranes S_i that will be formed in the next step. During the merge process, each subpopulation is maintained, and the worst individuals (a portion of the subpopulation) will be replaced by a copy of the best individuals to improve the subpopulation in each elementary membrane.

Third, the process is repeated to refine the sets of parameters to be able to perform an optimal or close-to-optimal path planning. Inside of each elementary membrane S_i the evolution process is performed by the EAPF. The EAPF starts with the creation of a random population of individuals *P(t)*, each individual (solution) is codified with the values of the proportional gains, attraction k_a and repulsion k_r required to generate the path.

For the parallel evaluation on GPU, where each path is evaluated. First, the total potential field $U_{total}(q)$ computation is performed,

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

Then, the potential force $F_{total}(q)$ which is employed to drive the mobile robot is computed,

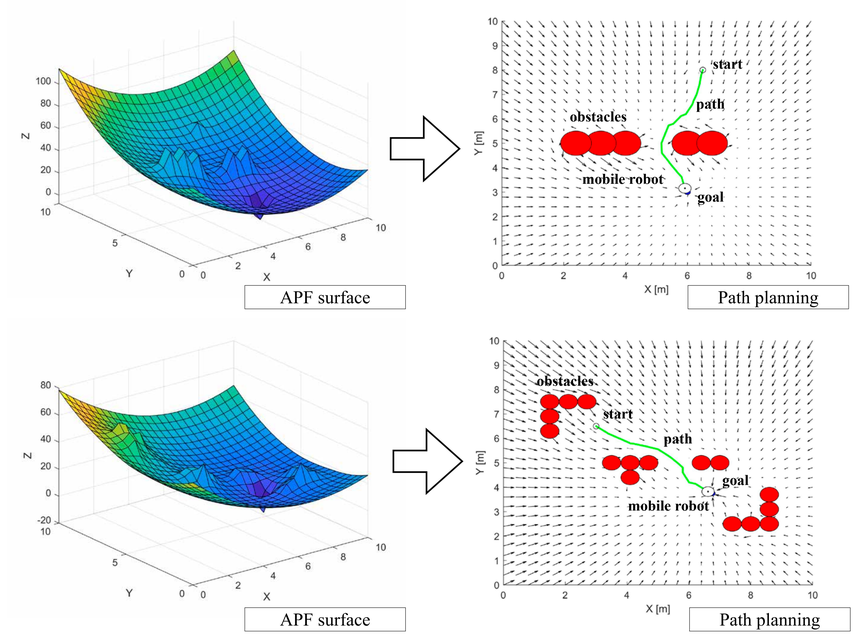
$$F_{total}(q) = -\nabla U_{total}(q)$$

Last, for the parallel path evaluation on GPU, each path is evaluated using

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

After the parallel path evaluation on GPU, the selection, crossover and mutation operators are applied to evolve the individuals in *P(t)*. All the path planning is enclosed in an iterative process until the maximum number of generations has been achieved.

Path Planning Results

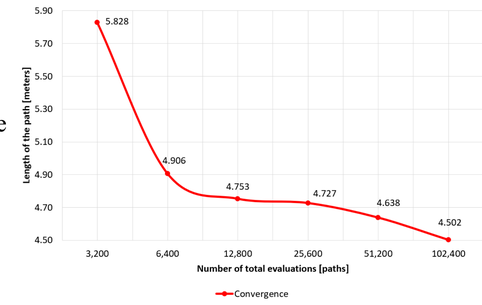


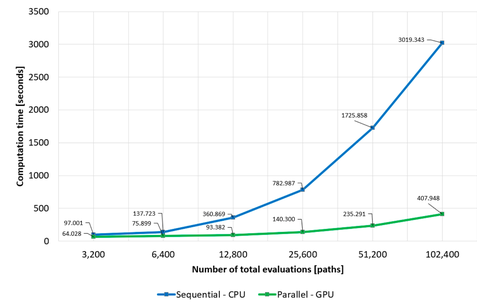
Performance Results

The experiments were achieved using the NVIDIA Jetson TX2 that is an embedded system-on-module (SoM) with dual-core NVIDIA Denver2 + quad-core ARM Cortex-A57, 8GB 128-bit LPDDR4 and integrated 256-core Pascal GPU.

The software installed is Ubuntu 18.04, JetPack 4.2.2 with CUDA 10.0.

To evaluate the performance of the parallel path computation on GPU versus the sequential path computation on CPU, we carried out independently thirty tests for each number of total evaluations.





Conclusions

- In this work, we have presented the parallel path computation on GPU for mobile robot navigation using the memEAPF programmed in C++/CUDA.
- The performance results show that the parallel path computation on GPU accelerates the process by a factor of **7.4x** for the biggest population tested in comparison with sequential path evaluation on CPU.
- We can see the advantage of using the parallel path computation on GPU, as well as we can see that this advantage applies to onboard computers like the Jetson TX2.
- The memEAPF algorithm is highly scalable and can be implemented on other Jetson Platforms like AGX Xaxier Series or the compact Jetson Nano.
- The memEAPF algorithm could be useful to many applications in mobile robots for global and local path planning, including industrial and domestic mobile robots, unmanned aerial vehicles, autonomous underwater vehicles, exploration vehicles, and self-driving cars.

[1] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, "Mobile robot path planning using membrane evolutionary artificial potential field," Applied Soft Computing, vol. 77, pp. 236 – 251, 2019. https://doi.org/10.1016/j.asoc.2019.01.036
[2] G. Păun, "Computing with membranes," Journal of Computer and System Sciences, vol. 61, no. 1, pp. 108–143, 2000. https://doi.org/10.1006/jcss.1999.1693
[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," The International Journal of Robotics Research, vol. 5, no. 1, pp. 90–98, 1986. https://doi.org/10.1177/027836498600500106

This work was supported in part by the Coordinación Institucional de Investigación of CETYS Universidad, and by the Consejo Nacional de Ciencia y Tecnología (CONACYT).