GPU-ACCELERATED MOBILE ROBOT LOCALIZATION

Maxim N. Rud

Tomsk Polytechnic University, Russia, Tomsk
E-mail: rudmax13@gmail.com

Introduction

The problem of localization is one of the most important problems that has to be solved for autonomous robot control. One of the most effective methods of robot localization is particle – based Monte – Carlo method. Given the map of the environment, this algorithm estimates the position and orientation of robot as it moves and senses the environment. In Monte – Carlo method system state is represented by set of particles. Each particle is a data object containing one of the hypothetical robot states from the distribution and a «weight» value. To maintain an accurate representation of the state distribution, we must have a large number of particles, which makes the particle filter computationally intensive. However, the hardest steps in the particle filter are done independently on each particle, so it is inherently suitable for parallel processing on graphical processing unit (GPU). CUDA [1] is a parallel computing platform running on NVIDIA GPUs It is one of the popular GPGPU (General-Purpose computing on GPU) platforms.

Monte-Carlo localization

The idea of Monte – Carlo localization is representation of possible robot states by set of particles. These particles can be seen as virtual copies of robot's existence. Particles move simultaneously with robot according to robot's motion model and have several sensor beams representing robot's sensor model.

Our robot operates in the laboratory of robotics and turns with differential steering, so its state includes three variables – x, y coordinates for robot's location and θ – yaw angle for robot's orientation.

There are main steps of the algorithm are presented below:

1. Generate a set of particle and initialize them with random positions and weights.
2. Apply robot's motion model to particles to predict the next robot's state.
3. Sense the environment and compare actual measurements with predicted ones.

4. Compute particle weights according on how well each particle predicted the actual robot's state.
5. Resample the distribution. Probability of choosing each particle is proportional to its weight value.
6. Back to step 2.

Results

On fig. 1, you can see comparison of algorithm implementations using CPU and GPU. We tested a plenty of particle numbers in order to collect an accurate statistics.

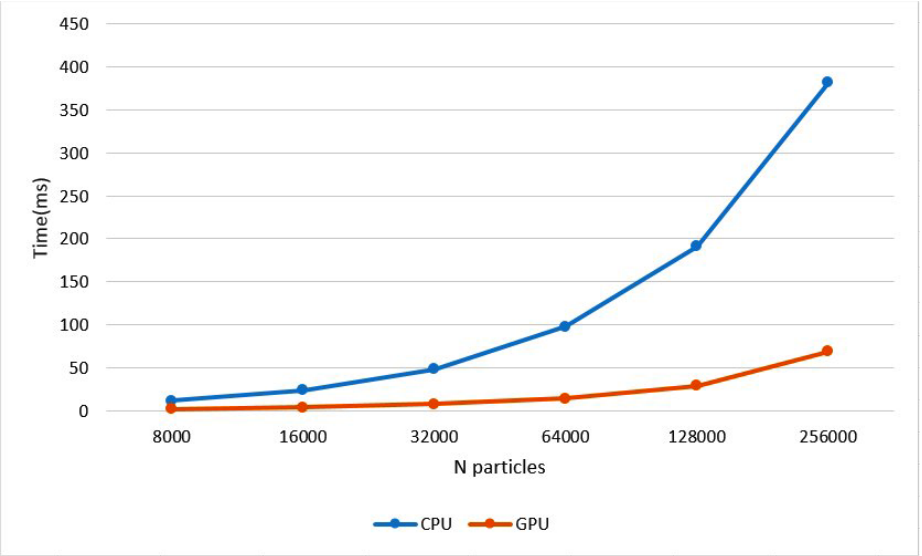

Fig. 1. Performance tests

We can see that using CUDA in Monte – Carlo localization implementation gives us a great increase in performance. As it was predicted, the weight calculation step is the most time – consuming step in the algorithm. After porting it onto CUDA, we obtained approximately 15 times speed – up. Algorithm was tested in specially created simulator. Now it is ready for using on real hardware.

References

1. Thrun S. Probabilistic robotics // Wolfram Burgard, 2000.
2. Wilt N. CUDA Handbook: A comprehensive guide to GPU programming // Addison-Wesley, 2013.
3. Falahati S. OpenNI Cookbook // PACKT Publishing, 2011.
4. NVIDIA Jetson TK1. – URL : http://www.nvidia.ru/object/jetson-tk1-embedded-dev-kit-ru.html