IMPROVEMENT OF 3D MONTE CARLO LOCALIZATION USING A DEPTH CAMERA AND TERRESTRIAL LASER SCANNER

S. Kanai [a,*], R. Hatakeyama [a], H. Date [a]

[a] Graduate School of Information Science and Technology, Hokkaido University, Kita-ku, Sapporo 060-0814, Japan - {kanai, hdate}@ssi.ist.hokudai.ac.jp, r_hatakeyama@sdm.ssi.ist.hokudai.ac.jp

Commission IV/WG7 & V/4

KEY WORDS: Monte Carlo Localization, Depth Camera, Terrestrial Laser Scanner, GPGPU, IMU, Scene Simulation

ABSTRACT:

Effective and accurate localization method in three-dimensional indoor environments is a key requirement for indoor navigation and lifelong robotic assistance. So far, Monte Carlo Localization (MCL) has given one of the promising solutions for the indoor localization methods. Previous work of MCL has been mostly limited to 2D motion estimation in a planar map, and a few 3D MCL approaches have been recently proposed. However, their localization accuracy and efficiency still remain at an unsatisfactory level (a few hundreds millimetre error at up to a few FPS) or is not fully verified with the precise ground truth. Therefore, the purpose of this study is to improve an accuracy and efficiency of 6DOF motion estimation in 3D MCL for indoor localization. Firstly, a terrestrial laser scanner is used for creating a precise 3D mesh model as an environment map, and a professional-level depth camera is installed as an outer sensor. GPU scene simulation is also introduced to upgrade the speed of prediction phase in MCL. Moreover, for further improvement, GPGPU programming is implemented to realize further speed up of the likelihood estimation phase, and anisotropic particle propagation is introduced into MCL based on the observations from an inertia sensor. Improvements in the localization accuracy and efficiency are verified by the comparison with a previous MCL method. As a result, it was confirmed that GPGPU-based algorithm was effective in increasing the computational efficiency to 10-50 FPS when the number of particles remain below a few hundreds. On the other hand, inertia sensor-based algorithm reduced the localization error to a median of 47mm even with less number of particles. The results showed that our proposed 3D MCL method outperforms the previous one in accuracy and efficiency.

1. INTRODUCTION

With recent interest in indoor navigation and lifelong robotic assistance for human life support, there is an increased need for more effective and accurate localization method in three dimensional indoor environments. The *localization* is the ability to determine the robot's position and orientation in the environment. So far, three typical methods have been proposed for the indoor localization; (1) based only on internal sensors such as an odometry or an inertial navigation, (2) utilizing observations of the environment from outer sensors in an a priori or previously learned map such as Monte Carlo Localization (MCL), and (3) relying on infrastructures previously-installed in the environments such as distinct landmarks such as bar-codes, WiFi access points or surveillance camera networks (Borenstein et al., 1997).

Among them, MCL gives one of the promising solutions for the localization when previously created environment map is available and the system installation should be realized at a low cost. The MCL is a kind of probabilistic state estimation methods (Thrun et al., 2005) which can provide a comprehensive and real-time solution to the localization problem. However, previous work of MCL has been mostly limited to 2D motion estimation in a planar map using 2D laser scanners (Dellaert et al., 1999; Thrun et al., 2001). Recently, a few 3D MCL approaches have been proposed where rough 3D models and consumer-level depth cameras are used as the environment maps and outer sensors (Fallon et al., 2012; Hornung et al., 2014; Jeong et al., 2013). However, their localization accuracy and efficiency still remain at an unsatisfactory level (a few hundreds millimetre error at up to a few FPS) (Fallon et al., 2012; Hornung et al., 2014), or the accuracy is not fully verified using the precise ground truth (Jeong et al., 2013).

Therefore, the purpose of this study is to improve an accuracy and efficiency of 6DOF motion estimation in 3D Monte Carlo Localization (MCL) for indoor localization. To this end, firstly, a terrestrial laser scanner is used for creating a precise 3D mesh model as an environment map, and a professional-level depth camera is installed in the system as an outer sensor. GPU scene simulation is also introduced to upgrade the speed of prediction phase in MCL. Moreover, GPGPU programming is implemented to realize further speed up of the likelihood estimation phase, and anisotropic particle propagation is introduced based on the observations from an inertia sensor in MCL. The improvements in the localization accuracy and efficiency are verified by the comparison with a previous 3D MCL method (Fallon et al., 2012).

2. 3D MONTE CARLO LOCALIZATION

Monte Carlo Localization (MCL) is one of probabilistic state estimation methods (Thrun et al., 2005) using observation from outer sensor. The position and orientation of a system to be estimated is expressed by a state variable. A probability density function of the state variables is represented approximately by a finite set of "particles" each of which expresses a discrete instance of the state variables, and a progress of the probability

*Corresponding author:
Satoshi Kanai (kanai@ssi.ist.hokudai.ac.jp)

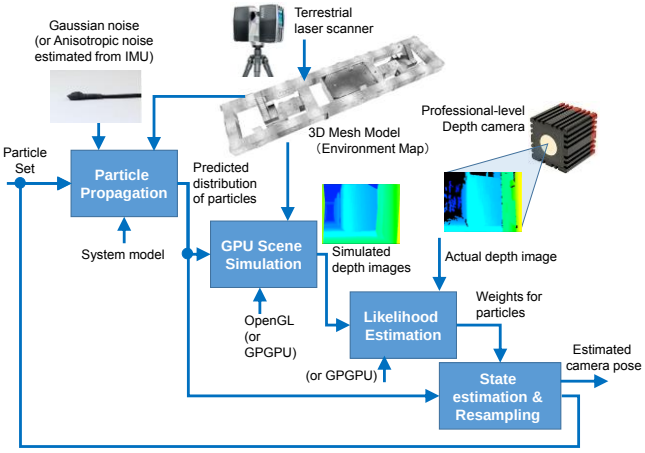


Figure 1. Our 3D MCL baseline algorithm

distribution is estimated by repeating propagation, likelihood evaluation and weighting based on the observation, and resampling of the particles.

The 3D MCL algorithms of our study are extensions of the previous 3D MCL approach which made use of GPU scene simulation (Fallon et al., 2012). As our extensions, in order to increase the accuracy of a map and an outer sensor, a terrestrial laser scanner is introduced for creating a precise 3D mesh model as an environment map, and a professional-level depth camera is installed as an outer sensor. Moreover, in our second extensions, a GPGPU programming is implemented to realize a speed up of the localization process, and an anisotropic particle propagation based on the observations from an inertia sensor is introduced in order to increase the localization accuracy.

Figure 1 shows a flow of our baseline 3D MCL algorithm. The state variable $\boldsymbol{x}_t = [x_t, y_t, z_t, \varphi_t, \theta_t, \psi_t]$ denotes a 6-DOF pose of the depth camera at a time step t, and a probability distribution of $\boldsymbol{x}_t$ is expressed by two different set of particles $\mathcal{X}_{t|t-1}$and $\mathcal{X}_{t|t}$ as Eqs (1) and (2).

$$\mathcal{X}_{t|t-1} \equiv \left\{ \boldsymbol{x}_{t|t-1}^{(i)} \right\} \tag{1}$$

$$\mathcal{X}_{t|t} \equiv \left\{ \boldsymbol{x}_{t|t}^{(i)} \right\} \tag{2}$$

where, $\mathcal{X}_{t|t-1}$ is called a predicted distribution, $\mathcal{X}_{t|t}$ is called a filter distribution, and $\boldsymbol{x}_{t|t-1}^{(i)}$ and $\boldsymbol{x}_{t|t}^{(i)}$ represent an *i*-th particle of each distribution respectively.

The depth camera pose $\boldsymbol{x}_t$ can be estimated by the following steps;

1) **Initialization**: An initial filter distribution $\mathcal{X}_{0|0}$ at t =0 is created by assigning a same state $\boldsymbol{x}_{0|0}$ into all of the particles in the distribution as Eq(3).

$$\mathcal{X}_{0|0} = \{\boldsymbol{x}_{0|0}, \boldsymbol{x}_{0|0}, \ldots\ldots, \boldsymbol{x}_{0|0}\} \tag{3}$$

2) **Propagation**: A predicted distribution $\mathcal{X}_{t+1|t}$ of the next time step $t+1$ is generated from a current filtered distribution $\mathcal{X}_{t|t}$ by applying a system model of 6-DOF motion of the depth camera. In the baseline algorithm, the system model is given by a Gaussian noise model as Eq(4).

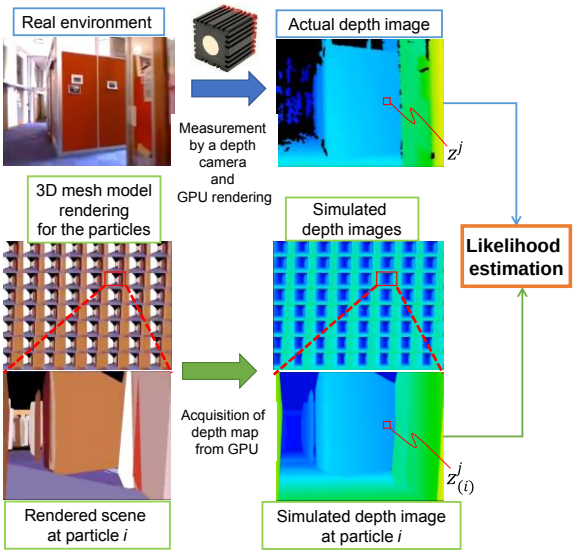


Figure 2. GPU scene simulation

$$\boldsymbol{x}_{t+1|t}^{(i)} = \boldsymbol{x}_{t|t}^{(i)} + \aleph(\boldsymbol{0}, \boldsymbol{\sigma}^2) \tag{4}$$

where, $\aleph(\boldsymbol{0}, \boldsymbol{\sigma}^2)$ is a 6D normal random numbers with variances $\boldsymbol{\sigma}^2 = [\sigma_x^2, \sigma_y^2, \sigma_z^2, \sigma_\varphi^2, \sigma_\theta^2, \sigma_\psi^2]$. By substituting $\boldsymbol{x}_{t|t-1}^{(i)}$ with $\boldsymbol{x}_{t+1|t}^{(i)}$ in $\mathcal{X}_{t|t-1}$, a predicated distribution at the next time step is updated to $\mathcal{X}_{t+1|t}$.

3) **GPU scene simulation**: As shown in Figure 2, a set of simulated depth images is generated in a premade a 3D mesh model of an environment map. Each simulated depth image $Z_{(i)}^G$ is easily obtained by GPU rendering whose viewpoint coincides with a camera pose expressed by an *i*-th particle $\boldsymbol{x}_{t+1|t}^{(i)} \in \mathcal{X}_{t+1|t}$ in the predicted distribution. In this study, an OpenGL function *glReadPixels*() is used to quickly obtain an two-dimennsional array of normalized depth values of $Z_{(i)}^G$ from a depth buffer of GPU in one go.

4) **Likelihood estimation and weighting**: A raw depth image at time step $t+1$ is captured from the depth camera, and then a simple moving average filter among consecutive frames is applied to the image to suppress noises of depth values. This filtered depth image is also rendered by GPU to generate a rendered actual depth image Z^D which has a same image format as a simulated image $Z_{(i)}^G$. Then Z^D is compared with each of the simulated image $Z_{(i)}^G$. As a result of the comparison, a likelihood of *i*-th particle $l_{(i)} \in [0, 1]$ is evaluated by Eqs(5) and (6).

$$\tilde{l}_{(i)} = \sum_{j=1}^{M} 1 - \frac{\left| z_D^j - z_{(i)}^j \right|}{0.5 + \left| z_D^j - z_{(i)}^j \right|} \tag{5}$$

$$l_{(i)} = \frac{\tilde{l}_{(i)} - \tilde{l}_{min}}{\tilde{l}_{max} - \tilde{l}_{min}} \tag{6}$$

where, z_D^j is a depth value at a pixel *j* in the rendered actual depth image Z^D, $z_{(i)}^j$ is a depth value at a pixel *j* in the simulated image $Z_{(i)}^G$ corresponding to an *i*-th particle,

and M is the number of pixels in a depth image. $\tilde{l}_{max}$ and $\tilde{l}_{min}$ are the maximum and minimum likelihood values respectively among elements of a set $\{\tilde{l}_{(i)}\}$ each of which is evaluated by Eq(5) corresponding to an i-th particle. Using Eq(6), a normalized likelihood value $l_{(i)}(\in [0,1])$ for an i-th particle can be obtained.

Once a normalized likelihood value $l_{(i)}$ is obtained, a normalized weight $w_{(i)}(\in [0,1])$ is assigned to the i-th particle in the predicted distribution $\mathcal{X}_{t+1|t}$ based on $l_{(i)}$ as Eqs(7) and (8).

$$\tilde{w}_{(i)} = \exp\left[-\frac{(\tilde{l}_{(i)}-1)^2}{0.05^2}\right] \quad (7)$$

$$w_{(i)} = \frac{\tilde{w}_{(i)}}{\sum_{i=1}^{N} \tilde{w}_{(i)}} \quad (8)$$

where, N is the number of particles.

5) **State estimation and resampling**: The most probable estimate of the depth camera pose $\boldsymbol{x}_{t+1}$ is obtained as a weighted sum of the particles in $\mathcal{X}_{t+1|t}$ as Eq(9).

$$\boldsymbol{x}_{t+1} = \sum_{\boldsymbol{x}_{t+1|t}^{(i)} \in \mathcal{X}_{t+1|t}} w_{(i)} \boldsymbol{x}_{t+1|t}^{(i)} \quad (9)$$

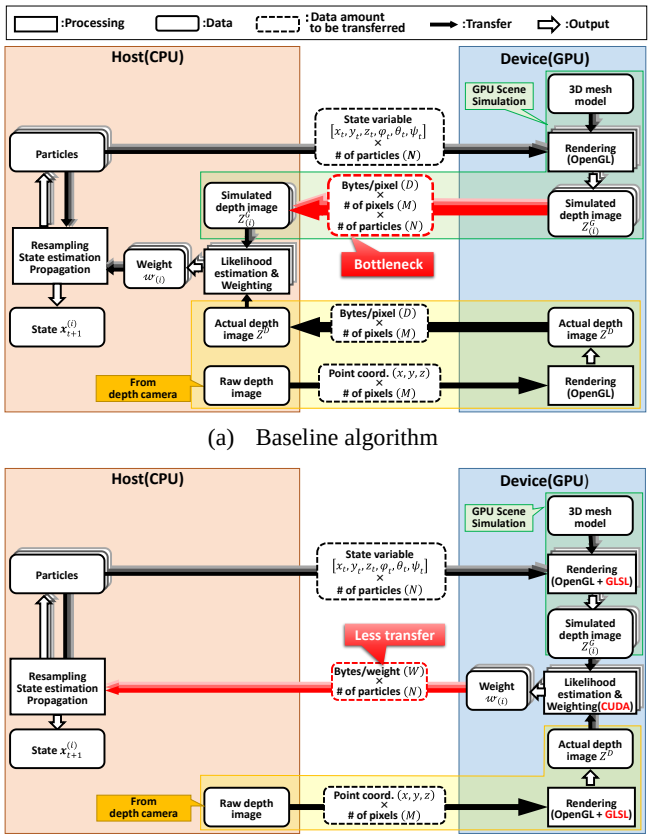
Finally, a new filtered distribution $\mathcal{X}_{t+1|t+1}$ is recreated by resampling particles in $\mathcal{X}_{t+1|t}$ so that an existence probability of i-th particle approximately equals to $w_{(i)}$. Roulette wheel selection method (Doucet et al, 2001) is used for the resampling.

By repeating the above from step 2) to 5), the particle distribution can be updated according to the observation from the depth camera, and state estimation of the camera pose $\boldsymbol{x}_t$ is sequentially updated. GPU rendering is introduced in step 2) for generating N simulated depth images in real time.

3. EFFICIENCY IMPROVEMENT OF THE LOCALIZAITON BASED ON GPGPU PROGRAMMING

As shown in Figure 3-(a), in our baseline MCL algorithm described in section 2, processing other than GPU scene simulation (step 2)) are executed by CPU, and every simulated depth image $Z_{(i)}^G$ which is rendered in GPU has to be uploaded directly to CPU right after the rendering using an OpenGL function *glReadPixels*().As a result, $(N \times M \times D)$ byte data has to be uploaded in all from GPU to CPU in every time step, where N means the number of total particles, M is the number of pixels in a depth image, and D is the number of bytes of one pixel data. In our setting, a professional-level depth camera (SR4000) is used, and it generates a depth image of 176×144 pixels in every frame. When a float-type variable (4 bytes) is used per pixel and 100 particles are included in a distribution, around 10Mbyte data has to be uploaded from GPU to CPU in every frame. Due to relatively slow execution speed of *glReadPixels*(), its transmission delay is not negligible, and it was observed that the delay caused a considerable bottleneck of the baseline MCL algorithm.

To reduce this delay, General Purpose computing on GPU (GPGPU) (Eberly, 2014) is introduced in the algorithm so that the GPU takes both GPU scene simulation (Step 2)) and likelihood estimation (Step 3)) as shown in Figure 3-(b). The



(a) Baseline algorithm

(b) GPGPU-based algorithm

Figure 3. Difference in MCL between the baseline and GPGPU-based algorithm

parallel processing of likelihood estimations for all particles can be handily implemented in GPGPU. In the GPGPU implementation, the simulated depth image $Z_{(i)}^G$ is directly rendered using GLSL (OpenGL Shading Language). And using CUDA (Sanders et al., 2010), a resulting rendered image can be compared with the rendered actual depth image Z^D, and likelihood estimation and weighting can be processed on the GPU without transferring any simulated depth image data to the CPU. Finally, only a set of weights of the particles $\{w_{(i)}\}$ only has to be transferred from GPU to CPU. Moreover, a likelihood value at a pixel j in the right hand side of Eq.(5) can also be evaluated independently of the other pixel. So we parallelized this pixel-wise likelihood calculation using CUDA.

As a result of this implementation, a set of weights $\{w_{(i)}\}$ whose data amount is around $(N \times W)$ byte (W: a number of bytes per one weight) in total only has to be uploaded from GPU to CPU per every frame. This can significantly reduce the data amount to be uploaded to about 400Byte. The effect of the implementation on our localization performance is explained in 5.2.

4. ACCURACY IMPROVEMENT OF THE LOCALIZAITON WITH THE AID OF AN INERTIA SENSOR

As the other approach for the accuracy improvement in the localization, a small 6-DOF inertial measurement unit (IMU) is installed in the system. With the aid of this IMU, the

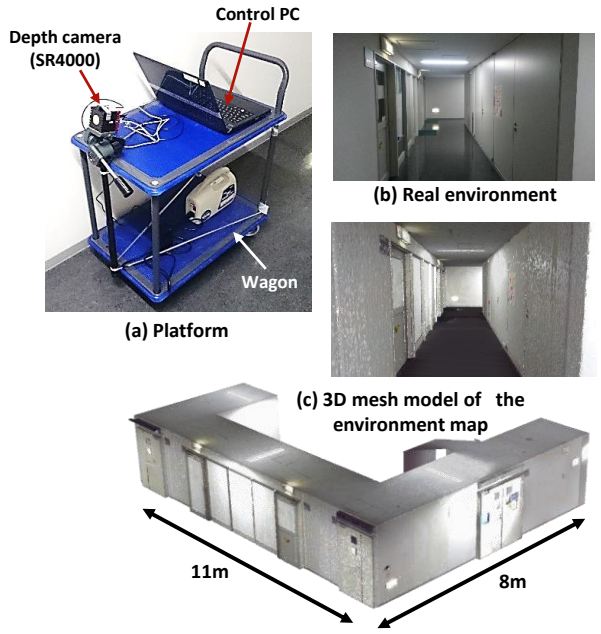


Figure 4. The platform and the environment map

anisotropic particle propagation based on an actual observation from the IMU can be introduced in the propagation step (Step1)) in MCL.

Different from Gaussian-based isotropic particle propagation in the baseline MCL, more particles are allocated adaptively along the direction of the measured acceleration and angular acceleration given from the IMU. In the propagation step in the MCL, this anisotropic particle propagation is easily implemented by replacing the original propagation equation Eq(4) with Eq(10).

$$\boldsymbol{x}_{t+1}^{(i)} = \boldsymbol{x}_{t}^{(i)} + \aleph\left(\boldsymbol{d}_t, \hat{\boldsymbol{\sigma}}^2\right) \qquad (10)$$

where, $\boldsymbol{d}_t = \left[\, d_{xt}, d_{yt}, d_{zt}, d_{\varphi t}, d_{\theta t}, d_{\psi t} \,\right]$ is a 6D estimated positional and rotational displacements derived from the numerical integration of the measured acceleration and angular acceleration of IMU at time step t. We also assume from the observation that a standard deviation of each displacement in the system model is comparable to the estimated displacement $\boldsymbol{d}_t$, and therefore set as $\hat{\boldsymbol{\sigma}}^{\boldsymbol{2}} = \left[\, d_{xt}^2, d_{yt}^2, d_{zt}^2, d_{\varphi t}^2, d_{\theta t}^2, d_{\psi t}^2 \,\right]$.

5. EXPERIMENTS

5.1 Setup

The improvement in efficiency and accuracy is verified in localization experiments in an indoor environment. Figure 4 shows our experimental setup. As shown in Figure 4-(a), a professional-level depth camera (SR4000, TOF camera, Pixel resolution: 176×144) and an IMU (ZMP IMU-Z Tiny Cube) were attached on a top surface of a small wagon. A laptop PC (Windows-7, Corei7-3.7GHz, and GeForce GT-650M) on the wagon recorded a time sequence both of raw depth image from the depth camera and acceleration data from the IMU on site. On the other hand, the localization calculation was done on the other desktop PC (Windows-7, Corei7-2.93GHz, and GeForce GTX-770).

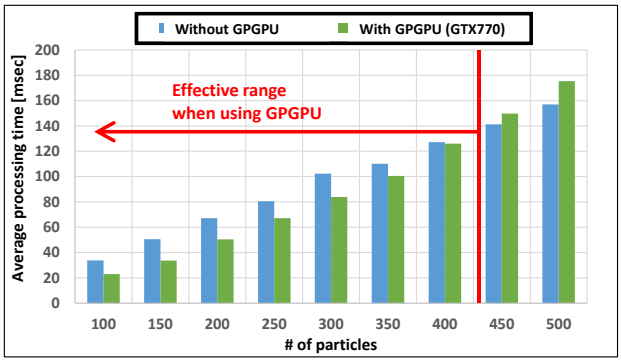


Figure 5. Processing time with and without GPGPU

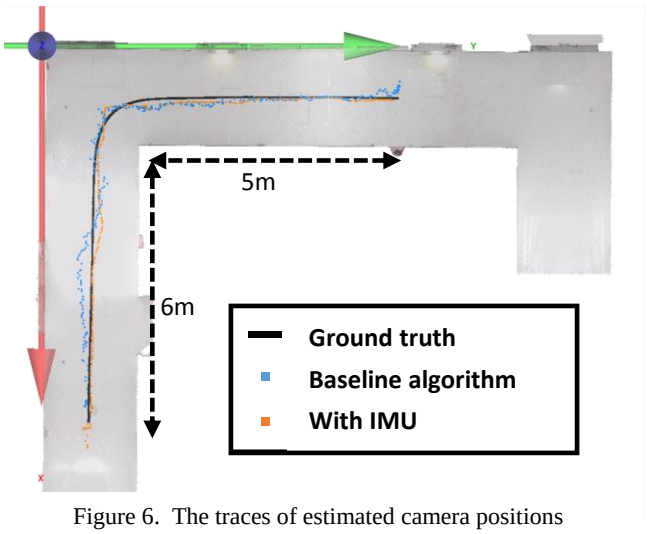


Figure 6. The traces of estimated camera positions with and without IMU

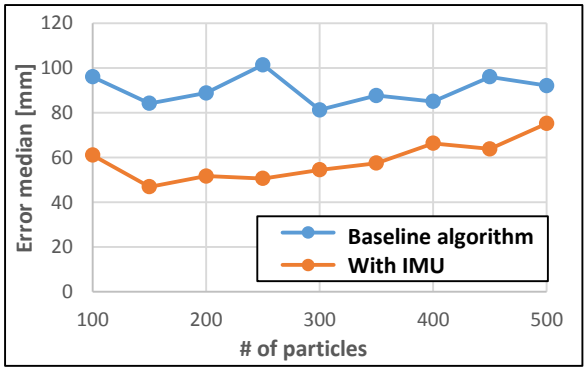
As a preparation, 3D point clouds of corridor space (9× 11 × 2.4m) in a university building shown in Figure 4-(b) was first measured by a terrestrial laser scanner (FARO-Focus 3D), and a precise 3D mesh model with 141,899 triangles shown in Figure 4-(c) was created as an precise environmental map of the space using a commercial point cloud processing software. GeForce GTX-770 was used when investigating the effect of GPGPU programming on an efficiency of the localization.

5.2 Results

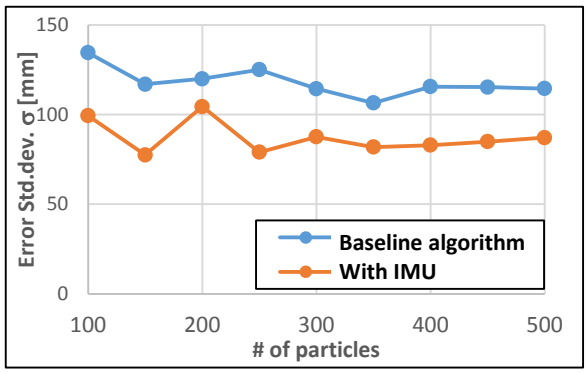
Efficiency improvement by GPGPU programming: Figure 5 compares the averaged processing time for single time step of MCL. 6-DOF depth camera pose is estimated in the experiment. From this figure, it is clearly shown that the proposed GPGPU-based algorithm is effective in reducing the time for localization when the number of particles remains below a few hundreds which is a general setting of this MCL.

However, the improvement in the processing speed was not so significant even when the GPGPU implementation is applied as shown in Figure 5. The reason for this behaviour is that GPGPU coding generally requires a sophisticated knowledge of parallel processing, and there is still a room for more efficient parallelization coding in likelihood estimation and weighting processing in GPU.

Accuracy improvement by IMU: Figure 6 compared the estimated camera positions using the baseline and proposed IMU-based algorithm in case of 200 particles. In this



(a) Median of error



(b) Standard deviation of error

Figure 7. Localization error in different particle settings

experiment, 6-DOF depth camera pose was estimated, but the accuracy verification is verified only in a planer position $[x, y]$ of 2-DOF. The ground truth trace of the wagon was precisely collected from physical marker recording attached under the wagon using a terrestrial laser scanner. The estimated positions with the aid of the IMU locate much closer to the ground truth than those without IMU.

Figure 7 shows the median and standard deviation of localization errors in different settings of the number of particles. In all settings, the IMU-based algorithm outperformed the baseline one, and the averaged median of the localization errors from the ground truth was reduced by 34%. And as shown in Figure 8, no significant decrease in the performance was observed when introducing the IMU.

Table 1 summarizes the performances between the proposed MCL method with IMU and the previous 3D MCL study (Fallon et al., 2012). The table shows that our MCL method realized much smaller localization error (16%) even with 57% less particles.

6. SUMMARY

Several methods were proposed to improve the accuracy and efficiency of 3D Monte Carlo Localization (MCL) for indoor localization. For the baseline algorithm, a terrestrial laser scanner was used for creating a precise 3D mesh model as an environment map, and a professional-level depth camera was installed as an outer sensor. Moreover, two original approaches were proposed for the improvement. As a result, it was confirmed that GPGPU-based algorithm was effective in increasing the computational efficiency to 10-50 FPS when the

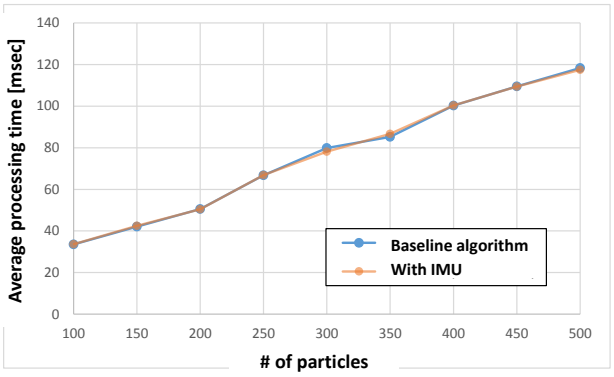


Figure 8. Processing time per frame in different particle settings

	# of particles	Speed [FPS]	Localization Error		
			Median[mm]	σ [mm]	3σ %
Proposed method with IMU	150	23.5	47	78	100
[Fallon et al., 2012]	350	7-8	300	N/A	90

Table 1. Comparison of the performances between the proposed method and [Fallon et al., 2012]

number of particles remain below a few hundreds. On the other hand, inertia sensor-based algorithm reduced the localization error to a median of 47mm even with less number of particles. The results showed that our proposed 3D MCL method outperforms the previous one in accuracy and efficiency aspects.

References

Borenstein, J., Everett, H.R., Feng, L., and Wehe, D., 1997, Mobile robot positioning: Sensors and techniques. *Journal of Robotic Systems*, 14(4), pp.231–249.

Dellaert, F., Fox, D., Burgard, W., and Thrun, S., 1999. Monte Carlo Localization for mobile robots. *IEEE International Conference on Robotics and Automation (ICRA)*, pp.1322–1328.

Doucet, A, Freitas, N., and Gordon, N., 2001. *Statistics Sequential Monte Carlo Methods in Practice*, Springer, New York, pp.437-439.

Eberly, D.H., 2014. *GPGPU Programming for Games and Science*, CRC Press, Boca Raton.

Fallon, M.F., Johannsson, H., and Leonard, J.J., 2012. Efficient scene simulation for robust monte carlo localization using an RGB-D camera. *IEEE International Conference on Robotics and Automation (ICRA)*, pp.1663–1670.

Hornung, A., Oswald S., Maier D., and Bennewitz, M., 2014. Monte carlo localization for humanoid robot navigation in complex indoor environments. *International Journal of Humanoid Robotics*, 11(2), pp. 1441002-1–1441002-27.

Jeong, Y., Kurazume, R., Iwashita Y., and Hasegawa T., 2013, Global Localization for Mobile Ro bot using Large-scale 3D

Environmental Map and RGB-D Camera. *Journal of the Robotics Society of Japan*, 31(9), pp.896–906.

Sanders, J., and Kandrot, E., 2010. *CUDA by Example: An Introduction to General-Purpose GPU Programming*, Addison-Wesley Professional, Boston.

Thrun, S., Fox, D., Burgard, W., and Dellaert, F., 2001. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1), pp.99–141.

Thrun, S., Burgard, W., and Fox, D., 2005. *Probabilistic Robotics*, MIT Press, Cambridge, pp.189–276.