Journal of KIIT. Vol. 22, No. 2, pp. 71-82, Feb. 28, 2024. pISSN 1598-8619, eISSN 2093-7571 71
http://dx.doi.org/10.14801/jkiit.2024.22.2.71

효율적 동적 객체 위치 추정을 위한 GPU 기반 병렬적 PSO 파티클 필터

김나연*, 이헌철**

Parallelized PSO-based Particle Filters with GPU for Efficient Dynamic Object Localization

Nayeon Kim*, Heoncheol Lee**

이 연구는 금오공과대학교 학술연구비로 지원되었음(2021년)

요 약

본 연구는 로봇 시스템에서 깊이 센서를 이용한 동적 객체 위치 추정 문제를 다룬다. 동적 객체에 대한 깊이 센서 데이터는 오차를 포함하고 있기 때문에 파티클 필터와 같은 확률적 필터를 사용해야 한다. 하지만 기존 파티클 필터는 파티클 공핍(Depletion) 문제로 인해 추정 성능이 저하된다. 본 논문은 입자군집최적화 기법을 파티클 필터에 적용함으로써 재추출(Resampling) 과정 이후 수렴 성능을 높임으로써 추정 성능을 향상시킴과 동시에, 파티클 개수를 유지하면서도 수행시간을 단축시키기 위한 GPU(Graphics Processing Unit)기반 병렬 처리 기법을 제안한다. 제안된 방법은 다양한 실험을 통해 추정 오차의 평균 및 표준편차를 감소시킴과 동시에 전체 연산 시간은 기존 방법과 유사하게 유지할 수 있음이 입증되었다.

Abstract

This paper addresses the problem of estimating dynamic object location estimation using a depth sensor in robotic systems. Because depth sensor data for dynamic objects contains errors, a probabilistic filter such as a particle filter has been widely used. However, the estimation performance of the conventional particle filter degenerates as time goes due to the particle depletion problem. This paper proposes an improved method to not only increase the estimation performance with particle swarm optimization(PSO) but also accelerate it by parallelizing the parts with excessive computation using a GPU(Graphics Processing Unit) without reducing the number particles. The proposed method was tested by real experiments, and its improved performance was verified by reducing the mean and standard deviation of estimation errors while showing similar computation times with the conventional method.

Keywords
dynamic object tracking, sensor, particle filter, particle swarm optimization, GPU parallelization

* 금오공과대학교 전자공학부 학사과정
- ORCID: https://orcid.org/0009-0003-5000-582X
** 금오공과대학교 IT융복합공학과 부교수(교신저자)
- ORCID: https://orcid.org/0000-0001-8776-5185

· Received: Jan. 12, 2024, Revised: Feb. 07, 2024, Accepted: Feb. 10, 2024
· Corresponding Author: Heoncheol Lee
Dept. of IT Convergence Engineering, School of Electronic Engineering
Kumoh National Institute of Technology, Korea
Tel.: +82-54-478-7476, Email: hclee@kumoh.ac.kr

Ⅰ. 서 론

1.1 연구 배경

전장 환경은 알 수 없는 환경과 위협의 불확실성, 정보 수집의 제한을 갖는 특수한 환경이다[1]. 이러한 환경에서 국방 과학기술의 발달로 사람을 대신할 수 있는 무인로봇은 다양한 목적으로 활용되고 있다. 전장 환경에서 실시간으로 객체를 인식하고 객체의 정보를 이용하는 것은 중요하다. 그중 객체의 위치 정보를 활용하기 위해선 여러 센서와 데이터 처리 능력이 필요하다[2]. 하지만 전장 환경에서의 대부분 로봇은 무거운 배터리를 사용하기 어렵고, 실시간 처리를 위한 에너지 자원의 한계가 있다. 따라서 탑재할 수 있는 센서의 수가 제한적이고, 배터리를 이용하여 동작하는 로봇은 소모 전력이 제한적이기 때문에 에너지 효율은 중요하게 고려되어야 할 사항이다. 또한 제한된 계산 자원으로 데이터를 처리해야 하기에 최적화 과정이 필요하다. 일반적으로 객체의 위치를 추정하기 위해서 비전 센서와 LiDAR 센서를 결합하여 정교한 객체 추정을 수행하지만, 실시간 로봇 시스템에서 객체 탐지 및 위치 추정을 수행하기 위해선 저성능의 깊이 센서를 이용한다. 저성능의 깊이 센서는 측정 범위와 해상도 및 각도에 제한받기 때문에 다소 부정확한 데이터를 수집하게 된다[3]. 부정확한 데이터를 이용한 객체 추정은 정확성과 신뢰성의 저하로 이어지게 된다.

1.2 관련 연구

이러한 한계를 극복하기 위한 연구는 다양하게 시도됐다[4]-[10]. 객체 추정을 위해 칼만 필터 알고리즘[4], 파티클 필터 알고리즘[5] 등을 활용한 연구[6]가 계속 진행되었다. 파티클 필터의 리샘플링 과정을 변형하거나[7], 객체의 주변 정보를 추가로 사용한다[8][9]. 또는 여러 센서를 융합하여 객체의 위치를 정확하게 추정하지만, 노이즈가 포함된 정보가 들어올 때 부분적으로 불안정한 추정 결과가 도출되기도 한다[10]. 파티클 필터를 사용함으로써 불확실한 정보를 처리할 수 있고 실제 환경의 특징인 비선형성에 적용할 수 있다. 하지만 파티클 필터를 사용해도 알려지지 않은 객체에 대한 불확실성으로 인해 예측이 어렵고 입자들의 다양성이 퇴화하는 파티클 필터의 특징 때문에 안정적인 추정이 어렵다. 이는 PSO 알고리즘을 파티클 필터와 결합하여 입자 결핍 문제 및 안정적인 결과를 내는 연구들을 찾을 수 있다[11]-[14]. 또한 파티클 필터의 성능과 연관 있는 입자 개수와 관측 모델의 정교성으로 인해 큰 수행시간을 요구한다. 이는 실시간성이 저하하는 원인이 된다. 실시간성을 견고하게 하기 위해 여러 방법으로 수행시간을 감소시킬 수 있는데, 성능의 저하가 일어나지 않기 위해선 입자 개수의 손실 또한 없어야 한다. 이렇게 입자 개수의 손실 없이 수행시간을 효과적으로 줄이기 위해서 GPU를 이용한 병렬화 연산이 필요하다. 이러한 이유로 확률적 연산의 수행시간을 줄이기 위한 GPU 기반의 병렬화를 사용하여 다양한 분야에 활용하는 연구가 진행되고 있다[15]-[19]. 객체의 위치 추정하기 위한 이러한 문제점을 극복하기 위해 본 논문에서는 PSO를 도입하여 PF의 안정적인 수렴을 도출하고, 파티클 개수를 유지하면서도 수행시간을 단축하기 위한 GPU를 이용한 병렬화 방법을 제안한다.

표 1. 파티클 필터 기반의 객체를 추정 연구들의 비교
Table 1. Comparison of particle filter-based object estimation studies

Related studies	Types of objects	Estimation methods	GPU-based parallelization
[6], [8]	Dynamic	PF	X
[5], [9]	Dynamic	PF	X
[7]	Dynamic	PF	O
[11], [12]	Dynamic	PSO-PF	X
[16], [18], [19]	Dynamic	PF	O
[15], [17]	Dynamic	PSO	O
Proposed	Dynamic	PSO-PF	O

II. 기존 연구와 문제점

2.1 심층학습 기반 객체 인식

심층학습 기반의 객체 인식은 주로 YOLO(You Only Look Once)를 사용하여 실시간 객체 인식을 진행한다. YOLO를 사용하여 RGB 이미지를 기반으로 객체를 분류하고 Bounding Box(BBox)를 이용하여 객체의 추정을 진행한다[20]. 하지만 2D 객체 인식이기 때문에 이를 실제 공간인 3차원 환경에서 위치와 방향을 알기 어렵다는 단점이 있다.

2.2 깊이 센서 기반 로봇 시스템 및 동적 객체 위치 간 좌표계 구성 및 객체 위치 추정 방법

심층학습 기반으로 객체를 인식하는 YOLO를 객체의 BBox는 2D 픽셀 좌표(u, v)이며, 이를 실제 환경에 투영하기 위해 깊이 센서의 정보를 사용하여 로봇과 동적 객체 사이의 실제 거리를 파악할 수 있다.

깊이 센서 기반의 객체 위치 추정은 YOLO를 이용하여 추정한 객체의 BBox 가운데 픽셀 정보를 사용하고, 이 정보를 3차원 공간상의 좌표(x, y, z)로 변환하여 로봇 시스템과 객체의 위치를 같은 지도상에 표시하여 실제 위치를 표현한다. 3차원 환경으로 변환된 좌표는 센서 기반의 측정값이기 때문에 노이즈가 포함된 불확실한 좌표다. 또한 정적 객체의 경우 그림 1에서 로봇의 위치, 깊이 센서를 기반으로 측정된 원시데이터(■), PSO-PF를 거쳐서 추정된 결과 입자들의 평균(■)을 같은 지도상에 표시하고 있다. 여기서 원시데이터는 RGB 이미지를 한 종류의 객체로 추정한 BBox의 가운데 픽셀을 깊이 센서의 3차원 좌표계로 변환한 값이다. 그림 1을 통해 정적 객체의 추정이 안정적으로 이루어짐을 확인할 수 있다. 하지만 동적 객체의 경우 이동을 예측할 수 없고 거리가 멀어질수록 센서의 오차가 더욱 증가하기 때문에 측정값에 의존해서 객체 위치를 추정하기에 한계가 있다. 이를 해결하기 위해 파티클 필터를 도입하여 객체의 위치를 안정적으로 추정한다.

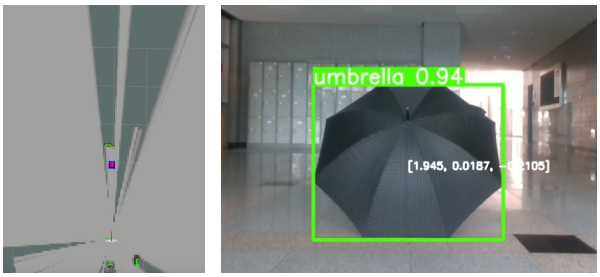


그림 1. YOLO와 깊이 카메라 센서를 기반으로 측정된 정적 객체 인식 및 위치 추정 결과
(정적 객체 인식 : ■, 위치 추정 : ■)
Fig. 1. Static object detection and position estimation results based on YOLO and depth camera sensor
(Static object detection : ■, Position estimation : ■)

2.3 기존 파티클 필터의 문제점

파티클 필터는 객체의 움직임이 비선형적이고 불확실한 상황에서 안정적인 위치 추정을 가능하게 한다. 하지만 입자의 개수에 따라 성능이 달라지고, 추정하고자 하는 객체가 비선형 형태의 움직임을 보인다면 입자의 수렴성이 낮아지는 경향이 있다. 따라서 상당한 입자 개수가 연산에 필요하지만, 연산량이 비례하여 증가하기 때문에 수행시간이 많이 소요된다. 많은 연산량은 실시간으로 수행하는 소형 로봇에서는 성능을 하락시키는 치명적인 요소가 된다. 효율적인 소형 로봇의 실시간 추정을 위해서는 입자의 연산 수행시간을 줄이는 방법이 필요하다.

III. 제안하는 기법

3.1 전체 구조

제안하는 방법의 전체적인 구조는 그림 2와 같이 깊이 센서 기반 객체 위치 추정, 파티클 필터, 입자 군집 최적화 단계로 구성되어있다. 깊이 센서 기반 객체 위치 추정은 심층학습 기반으로 얻은 객체 BBox 정보와 깊이 센서 정보를 이용하여 원시데이터를 얻는다. 이렇게 얻은 원시데이터를 입력으로 받아 필터링 동작을 한다. 필터링의 순서는 그림 3에서 순서도로 나타냈다. 파티클 필터는 샘플링, 측정 업데이트, 가중치 업데이트, 리샘플링 단계로 진행된다. 먼저 입자들의 샘플링을 진행하여 예측 모델을 이용한 초기 입자들을 생성한다.

이후 측정값을 업데이트 한 후, 예측된 입자들과 측정값의 거리를 구한다. 구해진 거리를 기반으로 가중치를 할당하는데, 가우시안 분포를 사용하여 거리가 작을수록 더 높은 가중치가 부여한다. 이렇게 부여된 가중치 기반의 리샘플링을 진행하여 가중치가 높은 입자들이 더 많이 선택되도록 한다. 이를 통해 노이즈로 인한 불확실성이 최소화된 입자들을 얻는다. 이후 파티클 필터의 결과 입자들을 입자 군집 최적화 알고리즘의 입력으로 연결하여 가중치가 높은 입자에 대한 수렴도를 높이고 입자들의 다양성을 유지하는 결과를 얻는다.

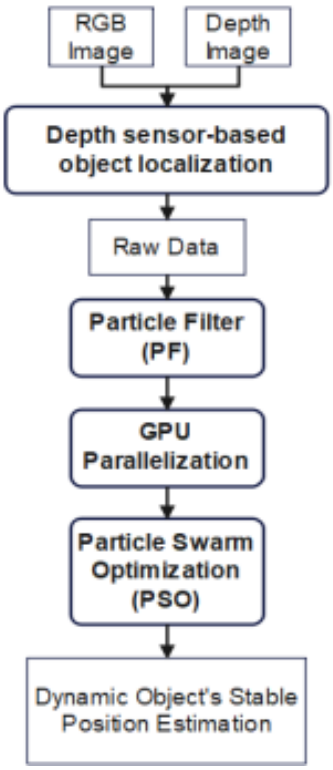


그림 2. 제안하는 기법의 구조도
Fig. 2. Structure diagram of the proposed

3.2 동적 객체 위치추정을 위한 모션 모델

본 연구에서는 동적 객체의 운동학성 특성, 즉 어떤 방향으로, 얼마만큼 움직일지 알 수 없다고 가정하였다. 따라서 동적 객체가 어떤 패턴으로 움직이는지 알 수 없는 불확실성으로 인해 예측이 어렵다. 따라서 객체의 이전 위치를 이용하여 다음 단계의 위치를 예측할 수 있도록 모션 모델(Motion model)을 설계했다.

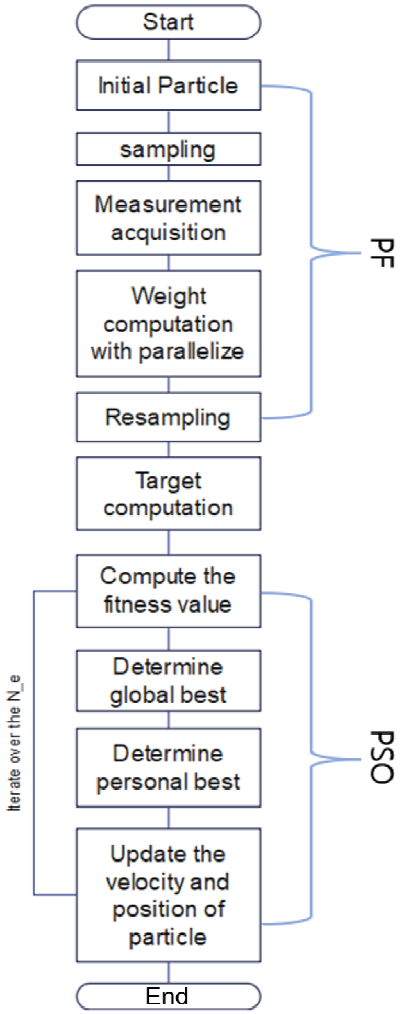


그림 3. 병렬적 연산의 PSO-PF 순서도
Fig. 3. Flowchart of parallelized PSO-based particle filters

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} + \Delta \boldsymbol{x}_t + \boldsymbol{\epsilon}_t \tag{1}$$

식 (1)에서 x는 객체의 위치, t는 시간, 객체의 위치 변화 Δx_t, 오차 벡터 ϵ_t를 나타낸다. Δx_t는 $\Delta \boldsymbol{x}_t = [\boldsymbol{x}_{t-1} - \boldsymbol{x}_{t-2}]^T$으로 t-2에서 t-1까지 객체가 얼마만큼, 어느 방향으로 움직였는지 나타낸다. 수식 (1)에서 객체의 이전 위치 x_{t-1}에 객체 위치 변화량 Δx_t을 더하고 오차 행렬을 추가함으로써 사간 t에서의 객체 위치 x_t를 나타낸다.

3.3 관측 모델(Observation model)

본 논문의 관측 모델은 깊이 센서로 측정된 객체의 위치 좌표를 사용한다.

먼저 YOLO를 이용하여 얻은 BBox의 가운데 픽셀 좌표인 $(u,\ v)$와 깊이 센서 카메라의 principal point $(x_p,\ y_p)$, 깊이 센서 카메라의 focal length $(f_x,\ f_y)$, 마지막으로 깊이 센서로부터 픽셀까지의 거리값인 $depth$를 식 (2)~(4)를 이용하여 3차원의 좌표로 변환한다.

$$\boldsymbol{x}_d = (u - x_p) \cdot \frac{depth}{f_x} \tag{2}$$

$$\boldsymbol{y}_d = (v - y_p) \cdot \frac{depth}{f_y} \tag{3}$$

$$\boldsymbol{z}_d = depth \tag{4}$$

이후 카메라 기준으로 구성된 $x_d,\ y_d$를 로봇의 기준으로 변환하기 위해 다음의 식 (5)~(7)을 사용한다.

$$\boldsymbol{l}_x = \boldsymbol{x}_d \tag{5}$$

$$\boldsymbol{l}_y = -\boldsymbol{y}_d \tag{6}$$

$$\boldsymbol{l}_z = -\boldsymbol{z}_d \tag{7}$$

위의 좌표계 변환 과정을 통하여 얻은 $(l_x,\ l_y,\ l_z)$ 는 로봇 중심 좌표이기 때문에 로봇과의 상대적인 거리를 표현한다. 이는 같은 지도상에 표시하기 어렵기 때문에, 다시 식 (8)~(10)을 이용하여 전역 좌표계로 변환한다[20].

$$\boldsymbol{g}_x = f_x + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{8}$$

$$\boldsymbol{g}_y = f_y + \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{9}$$

$$\boldsymbol{g}_z = f_z + \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

이렇게 변환된 $(g_x,\ g_y,\ g_z)$는 측정할 때의 노이즈가 포함되기 때문에 동일한 위치의 객체임에도 불구하고 모두 다른 객체로 추정된다. 따라서 일정 범위에 존재하면 동일한 객체의 위치 좌표로 간주하는 k-nearest neighbour test를 이용하여, 보다 안정적이고 정확한 객체의 위치 $\boldsymbol{k}$로 사용한다[21]. 이후 난수 δ을 더하여 관측값에 불확실성을 표현한 식 (11)과 같이 observation model로 사용한다.

$$\boldsymbol{z}_t = \boldsymbol{k} + \delta_t \tag{11}$$

3.4 PSO 기반 추정 성능 향상

입자 군집 최적화 알고리즘(PSO)은 각 입자들을 현재 위치와 속도를 갖고 있는 개체로 정의하고 각 입자가 자신의 경로와 경험을 공유하며 최적해를 찾는 알고리즘이다. 파티클 필터의 낮은 수렴성을 보완하기 위해 입자 군집 최적화를 도입하여 입자들의 공유성을 이용함으로써 안정적인 추정과 입자들의 수렴성을 향상시켰다. 본 논문에 사용된 알고리즘은 파티클 필터에서 추정된 입자들의 분포를 다시 입자 군집 최적화의 초기 위치로 사용하고 입자 군집 최적화 알고리즘의 목표물을 파티클 필터 결과 입자들의 평균으로 설정하여 파티클 필터의 단계 이후에 입자 군집 최적화를 진행한다. 우선 전역 최적해를 찾기 위해 식 (12)를 사용한다.

$$\boldsymbol{p}_g = \arg\min_{\boldsymbol{x}_i(s)} \| \boldsymbol{x}_i(s) - \boldsymbol{m}_{\boldsymbol{x}} \|, \tag{12}$$
$$\boldsymbol{m}_{\boldsymbol{x}} = \frac{1}{N_p} \cdot \sum_{i=1}^{N_p} \boldsymbol{x}_i(1)$$

전역 최적해는 현재의 주변 입자들 중에 가장 좋은 성능의 입자의 위치이다. 그 다음 개인 최적해를 찾는다. 식 (13)을 이용하여 각 입자가 지금까지 발견한 가장 좋은 위치를 업데이트한다.

$$\boldsymbol{p}_{\boldsymbol{p}} = \arg\min_{\boldsymbol{x}_i(s)} \| \boldsymbol{x}_i(s) - \boldsymbol{m}_{\boldsymbol{x}} \| \tag{13}$$

모든 입자는 전역 최적해와 개인 최적해를 공유하기 때문에 두 최적해를 이용하여 입자의 가속도 업데이트 할 수 있다. 입자의 가속도는 다음과 같이 정의한다.

$$\begin{aligned} \boldsymbol{a}_i(s+1) &= \omega \boldsymbol{v}_i(s) \\ &+ c_1\varepsilon_1(\boldsymbol{P}_{\boldsymbol{p}} - \boldsymbol{x}_i(s)) + c_2\varepsilon_2(\boldsymbol{P}_{\boldsymbol{g}} - \boldsymbol{x}_i(s)) \end{aligned} \tag{14}$$

ω는 관성 가중치로 값이 높을수록 새로운 탐색을 지향하지만 수렴은 지연된다. c_1, c_2는 전역 최적해와 개인 최적해의 영향을 결정하는 상수로 사용된다. 또한 $\varepsilon_1, \varepsilon_2$는 0와 1사이의 난수를 사용하여 다양한 위치를 탐색할 수 있도록 한다. 식 (14)를 사용하여 다음 반복의 가중치를 예측하고, 이는 다음 반복의 속도 예측에 사용된다.

$$\boldsymbol{v}(s+1) = \boldsymbol{v}_i(s) + \boldsymbol{a}_i(s+1) \tag{15}$$

$$\boldsymbol{x}_i(s+1) = \boldsymbol{x}_i + \boldsymbol{v}_i(s+1) \tag{16}$$

식 (15), (16)을 사용하여 입자의 속도와 위치를 예측한다. 이 과정을 반복하여 입자들은 최적의 위치로 근접하게 된다.

3.5 GPU 기반 병렬화

PSO-PF의 좋은 성능을 도출하기 위해선 많은 입자의 개수를 사용해야 한다. 하지만 그에 비례해서 연산량이 증가하기 때문에 GPU를 이용한 병렬화를 통해 연산 수행시간을 감소시켰다. GPU는 대량의 Core와 SIMT(Single Instruction, Multiple Threads) 구조를 이용하여 하나의 명령어로 동시에 여러 데이터를 처리할 수 있다. 본 논문에서는 CUDA를 이용하여 GPU의 병렬적 연산 처리를 하였다. CUDA에서는 각 스레드가 데이터를 처리하고 이를 모든 스레드는 같은 커널을 공유하여 동일한 연산을 수행한다. 스레드는 크게 두가지 계층으로 나눌 수 있는데, 하나는 스레드의 집합인 블록, 다른 하나는 블록의 집합인 그리드로 정의한다. 또한 커널에서 각 데이터에 접근하기 위해 인덱스를 사용하는데, 블록과 그리드 변수를 사용하여 인덱스를 지정한다.

그림 4는 입자 개수에 따른 PSO 파티클 필터의 수행시간 프로파일링 결과이다. 이를 통해 입자의 가중치를 계산하는 부분이 비교적 높은 수행시간으로 측정되는 것을 확인했다. 따라서 그림 5와 같이 가중치 연산을 GPU를 이용하여 병렬화로 진행한다.

가중치 계산에 필요한 메모리를 결정하기 위해, 스레드의 크기를 지정하고, 블록 크기를 식 (17)과 같이 입자 개수 N에 따라 동적으로 계산한다.

$$Block = (N + blockDim.x - 1)/blockDim.x \tag{17}$$

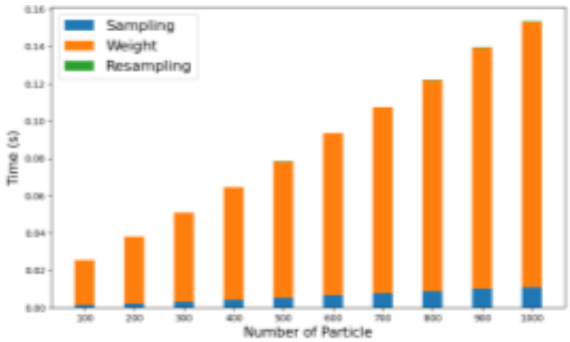


그림 4. 파티클 개수에 따른 PSO 파티클 필터 수행시간 프로파일링
Fig. 4. Profiling PSO particle filter execution time based on the number of particles

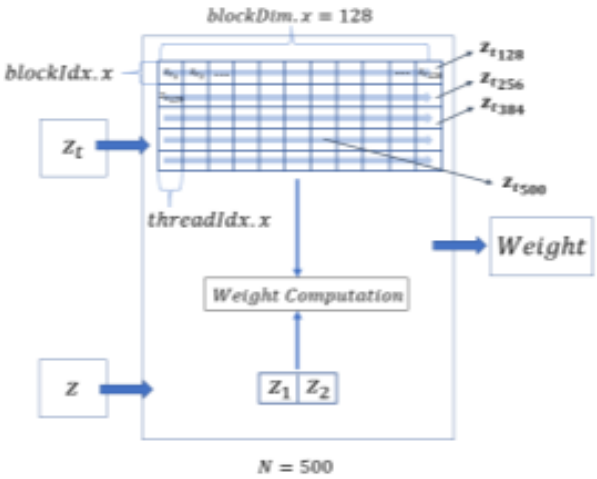


그림 5. 가중치 연산의 병렬화를 위한 커널
Fig. 5. Kernel for parallelizing weight computation

이후 가중치 연산 커널에 필요한 데이터를 GPU로 복사하고, 인덱스를 사용하여 각 스레드에 대한 연산을 동시에 수행하여 병렬로 처리한다. 커널의 연산은 기존 CPU의 계산과 동일하게 입자와 측정치의 차이를 계산하고 이를 기반으로 가우시안 분포인 확률 밀도 함수를 따라 가중치를 할당한다. 수식 (18)은 확률 밀도 함수다. 이후 각 입자의 가중치를 다시 CPU에 복사하여 다음 리샘플링 연산에 사용할 수 있도록 한다.

$$f(x, y \mid \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right) \tag{18}$$

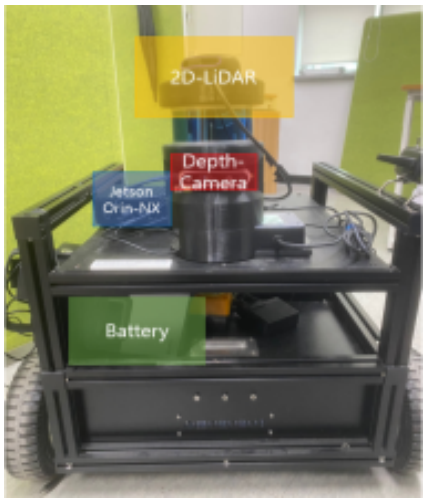


그림 6 실험 환경
Fig. 6. System

Ⅳ. 실험 결과

4.1 실험 환경 및 로봇 시스템 구성

본 연구는 그림 6의 환경으로 실험을 진행했다. 로봇에 Intel Realsense D435i 깊이 센서를 탑재하여 움직이면서 객체를 측정할 수 있도록 하고, 2D-LiDAR와의 정보를 함께 사용하여 객체 주변의 정보를 이용한다. 실험에서 사용된 Intel Realsense D435i는 깊이 해상도 최대 1280 x 720을 출력하고 사용하기에 이상적인 범위는 0.3m ~ 3m로 정해진 깊이 센서다. 깊이 센서의 정확도는 2m 이내에서 2%의 정확도와 최소 깊이 거리가 대략 28cm로 근거리일 때, 객체가 센서 중앙에 위치할 때 가장 높은 정확도를 갖는 센서이다. 객체의 위치는 SLAM(Simultaneous localization and mapping)을 활용하여 실제 지도상에 표시하였다. 로봇에 탑재된 Jetson Orin-NX는 NVIDIA의 임베디드 시스템으로, 배터리로 구동하는 로봇에서 제어 및 실시간 처리를 가능하게 해준다. 또한 고성능의 GPU를 제공하기 때문에 병렬적 연산 및 다중 센서 처리를 가능하게 해준다. 센서 및 프로세싱 시스템 구성과 관련하여, 깊이 센서와 동일한 종류의 데이터를 보다 높은 정확도로 얻기 위해서는 고성능 영상 센서 데이터와 LiDAR 센서 데이터를 융합해야 한다. 하지만 이 경우 시스템의 복잡도가 높아지게 되고 다중 센서 융합을 위한 추가적인 기법이 필요하게 되므로 알고리즘의 복잡도 역시 높아지게 됩니다. 따라서 데이터의 정확도가 다소 떨어지더라도 Depth 센서를 사용하게 되었다. 그리고 저성능 센서를 사용함에 따라 발생하는 정확도 문제를 해결하기 위해 알고리즘 성능을 향상시킬 필요가 있었고, 그에 따른 수행시간 단축을 위해 GPU가 포함된 프로세서를 사용하게 되었다.

4.2 객체 인식 및 위치추정 결과

본 논문에서는 SLAM과 깊이 센서를 활용하여 로봇의 위치와 원시데이터를 얻고, 추정된 객체 위치를 같은 지도상에 표시했다. 로봇이 정적 혹은 동적인 객체를 인식하면, 카메라 화면에 인식된 객체의 클래스와 로봇과 객체 간의 거리 좌표가 표시된다.

실험은 총 3가지 경우로 진행했고, 모두 동일하게 입자 개수 100개, 반복횟수 7번으로 설정했다. 첫 번째 실험은 정적인 객체(cover)를 로봇이 앞으로 움직이면서 위치를 실시간으로 추정하는 실험이다.

(a) 로봇과 객체의 초기 위치
(a) Initial positions of the robot and the object

(b) 로봇이 객체에 가까이 접근
(b) Robot approaching the object closely

그림 7. 정적 객체의 깊이 센서 기반 원시데이터 획득 및 위치 추정 결과
(센서 데이터 : ■, 위치 추정 : ■)
Fig. 7. Results of depth sensor-based raw data acquisition and position estimation for static objects
(Raw data : ■, Position estimation : ■)

그림 7의 (a)는 깊이 센서를 통해 얻은 원시데이터와 로봇의 초기 위치의 객체 위치 추정 결과를 보여준다. 그림 7의 지도에서 로봇의 시작점은 사각형 테두리로 표시하였고, 깊이 센서를 통해 얻은 원시데이터(■), PSO-PF의 결과 입자들(■)을 표시했다. 여기서 정적 객체는 상자(Cover)를 사용하였다.

그림 7의 (b)는 로봇이 객체에 다가갔을 때를 보여주고 왼쪽은 위의 그림 7의 설명과 동일하게 객체 원시데이터, 추정 위치, 로봇의 위치를 표현했다. 또한 로봇 시작점을 기준으로 로봇의 이동 정도를 직선으로 확인할 수 있고, 이는 타원 형태로 표시했다.

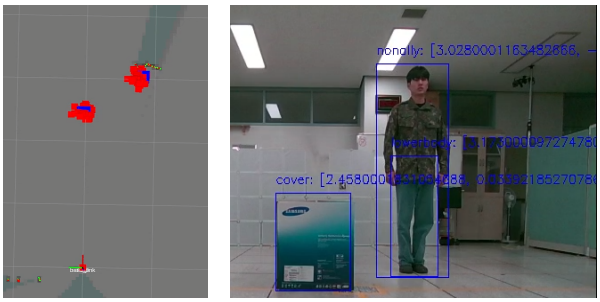


그림 8. 다중 객체의 깊이 센서 기반 원시데이터 획득 및 위치 추정 결과
(센서 데이터 : ■, 위치 추정 : ■)
Fig. 8. Results of depth sensor-based raw data acquisition and position estimation for static and dynamic objects
(Raw data : ■, Position estimation : ■)

두 번째 실험은 다중 객체를 동시에 추정하는 실험이다. 실험 1과 마찬가지로 로봇의 위치, 원시데이터, PSO-PF 결과 입자를 표시하였고, 객체는 상자(Cover), 사람(Nonally)를 이용하였다. 그림 8을 통해 다중 객체의 추정 결과를 확인할 수 있다.

세 번째 실험은 지정한 경로를 따라 움직이는 동적 객체를 추정하였다. 지도에 표시된 직사각형은 사전에 지정한 경로를 의미하고, 원시데이터(■), PSO-PF의 결과 입자들의 평균(■)을 표시하고 있다. 그림 9의 왼쪽을 통해 객체가 코너를 돌 때처럼 이동 상태의 변화가 생겼을 때, 제안된 병렬적 연산의 PSO-PF의 위치 추정 결과가 실제 위치와 가깝게 나온 것을 확인할 수 있다.

그림 9. 실험 3에서의 깊이 센서 기반 원시데이터 획득 및 위치 추정 결과
(원시데이터 : ■, 위치 추정 : ■)
Fig. 9. Results of depth sensor-based 원시데이터 acquisition and position estimation in experiment 3
(Raw data : ■, Position estimation : ■)

4.3 추정 안정성(standard deviation) 비교 및 분석

본 논문에서 제안한 병렬적 PSO 파티클 필터의 추정 결과이다. 그림 10~12의 그래프는 객체의 이동을 직선의 경로(▬)로 나타내고, 깊이 센서로 얻은 원시데이터(▬), 제안된 병렬적 PSO 파티클 필터의 위치 추정 결과 입자들을 평균으로 나타낸 분포(▬)를 표시했다. 제안된 필터의 결과 입자들의 결과는 원시데이터보다 안정적이고 Truth 위치에 더 수렴한 그래프다.

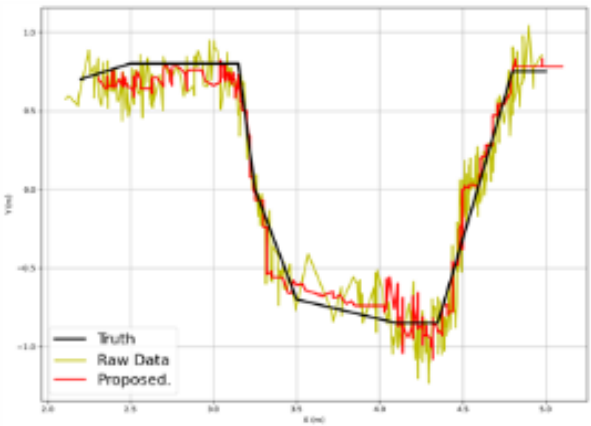


그림 10. 실험 1의 위치 추정 결과
(실제데이터 : ▬, 원시데이터 : ▬, 제안된 기법 : ▬)
Fig. 10. Object Position estimation results of Case 1
(Truth : ▬, Raw data : ▬, Proposed : ▬)

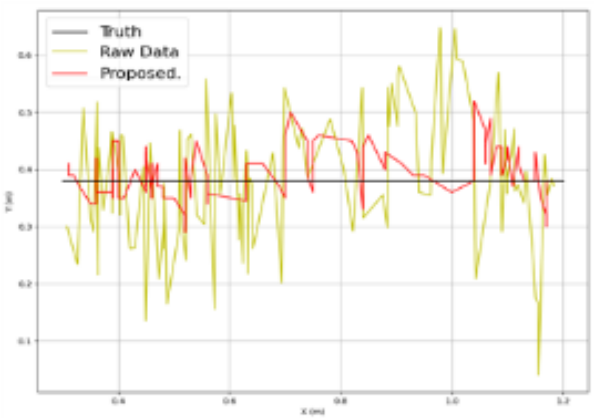


그림 11. 실험 2의 위치 추정 결과
(실제데이터 : ▬, 원시데이터 : ▬, 제안된 기법 : ▬)
Fig. 11. Object position estimation results of Case 2
(Truth : ▬, Raw data : ▬, Proposed : ▬)

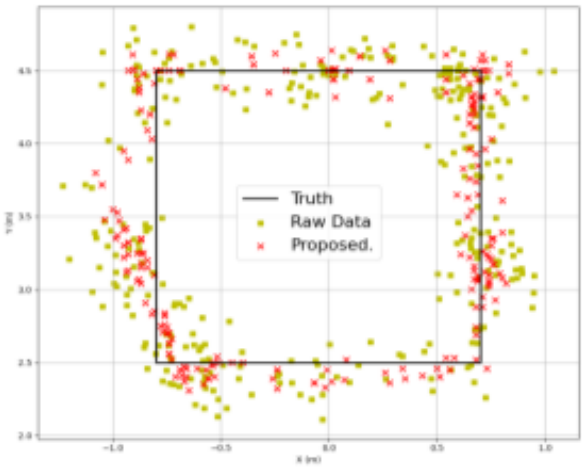


그림 12. 실험 3의 위치 추정 결과
(실제데이터 : ▬, 원시데이터 : ▬, 제안된 기법 : ×)
Fig. 12. Object position estimation results of case 3
(Truth : ▬, Raw data : ▬, Proposed : ×)

실험 1의 결과인 그림 10은 객체가 불규칙한 패턴의 경로를 움직였을 때 추정한 결과를 보여준다. 제안된 필터의 위치 추정 결과 입자들이 시간의 흐름에 따라 위치를 비교한 결과, 실제 위치의 움직임과 유사한 모습을 보인다.

실험 2의 결과인 그림 11은 객체가 x축으로 일정하게 움직였을 때 위치 추정을 보여준다. 그 결과 깊이 센서 기반의 원시데이터보다 입자들의 움직임의 폭이 작고, 이를 통해 안정적인 위치 추정이 된 것을 확인할 수 있다.

실험 3은 직사각형의 경로로 움직이는 객체를 추정하였다. 그림 12를 통해 원시데이터와 위치 추정된 입자들을 비교한 결과, 위치 추정된 입자들의 분포가 실제 경로의 분포와 비슷한 것을 확인했다. 또한 원시데이터의 결과보다 오차 범위가 작아 더 안정적인 추정을 했다고 볼 수 있다.

표 2. 실험 1~3의 RMSE 결과 비교
Table 2. Comparison of RMSE results for experiments 1 to 3

	Case 1 (m)	Case 2 (m)	Case 3 (m)
Raw data	0.581	0.476	0.614
Proposed	0.352	0.218	0.496

표 2는 원시데이터와 제안한 병렬적 PSO 파티클 필터의 RMSE(Root Mean Square Error)를 비교했다. 제안된 필터의 결과가 원시데이터보다 RMES의 수치가 더 작은 것으로 나왔다. 평균적으로 제안된 필터는 0.355 (m)의 오차를 갖고, 원시데이터는 평균 0.557 (m)를 갖는다. 이를 통해 실제 위치에 대해 더 안정적이고 정확한 위치를 추정하는 것을 보여준다.

4.4 수행시간 비교 및 분석

본 논문에서는 Jetson Orin-NX를 사용하여 기존 CPU의 연산보다 많은 코어 수를 이용했고, 모든 알고리즘은 입자 개수 500개, 6번의 반복을 사용했다.

먼저 그림 13에서는 가중치 업데이트 연산의 수행시간을 기존의 수행시간과 병렬적 연산을 통해 처리한 수행시간을 비교하였다. 그래프에서 Conventional은 기존 수행시간을 의미하고, Proposed는 제안하는 병렬적 연산의 수행시간을 의미한다. 결과를 통해 입자 개수에 따라 8배~16배 차이를 확인할 수 있고, 입자 개수가 증가함에 따라 CPU 수행시간은 비례하여 증가하지만, GPU 수행시간의 증가 폭은 크지 않은 것을 확인했다. 그림 14에서는 PF, PSO-PF, 제안된 병렬적 PSO 파티클 필터의 연산 시간을 비교했다. PSO의 알고리즘은 반복적인 수행으로 인해 같은 입자 수를 사용하더라도 PF보다 더 많은 시간을 소요하기 때문에 PF의 수행시간은 3가지 필터 중에서 가장 빠른 것으로 확인하였다. 하지만 PSO-PF와 병렬적 연산의 PSO-PF의 수행속도 차이는 1.67배 차이가 나는 것을 알 수 있다.

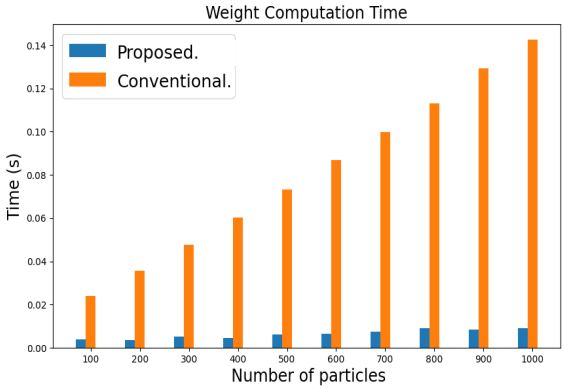


그림 13. 가중치 업데이트 수행시간 비교
Fig. 13. Comparison of weight update computation time

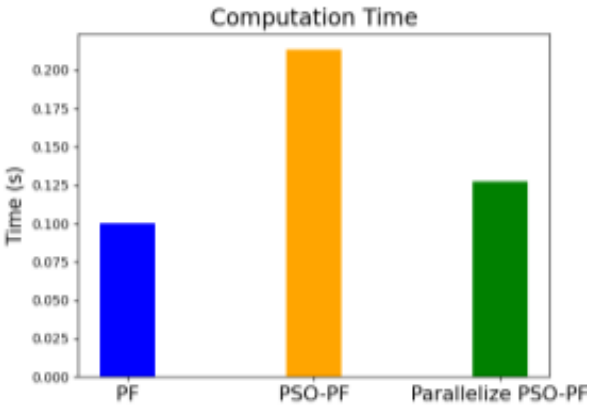


그림 14. 다양한 확률적 필터의 수행시간 비교
Fig. 14. Comparison of execution times for various stochastic filters

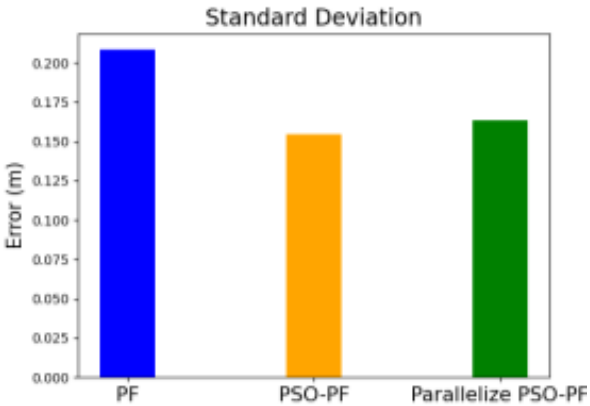


그림 15. 다양한 확률적 필터 결과 입자들의 표준 편차
Fig. 15. Comparison of standard deviations for particle results from various stochastic filters

그림 15를 통해 필터들의 위치 추정 결과에 대한 표준 편차를 확인할 수 있다. 표준 편차를 통해 PSO 알고리즘의 빠른 수렴 속도와 높은 수렴도를 보여주고, 이는 기존 파티클 필터의 사용보다 안정적인 추정 결과로 이어진다. PSO-PF와 병렬적 PSO 파티클 필터는 같은 연산 과정을 거치지만 표준 편차에 다소 차이가 발생하였다. 그 이유는 동일한 변수에 대해 CPU에서는 64bits가 사용되고, GPU에서는 32bits가 사용되기 때문이다. 즉, GPU에서 병렬 연산이 수행될 때 미세하지만 소수점 데이터의 일부가 반영되지 않았다. 다만, 그 차이가 아주 작았기 때문에 전체적인 추정 정확도는 차이가 거의 없었고, 표준 편차에서만 다소 차이가 있었다.

V. 결 론

본 연구는 실시간 로봇 시스템에서 제한된 센서를 이용하여 동적 객체를 실시간으로 추정하기 위해 PSO 기반의 파티클 필터를 제안한다. 파티클 필터를 이용하여 저성능의 센서가 포함하는 노이즈를 제거하여 안정적인 객체의 위치 추정을 가능하게 한다. 하지만 동적 객체의 움직임이 비선형적이고 미리 예측하기 어려울수록, 많은 입자가 필요하다. 또한, 파티클 필터의 낮은 수렴성으로 인해 입자의 위치 추정 결과가 나빠질 수 있다. 따라서 PSO 알고리즘을 도입하여 입자들의 협력을 통해 높은 수렴성과 입자들의 다양성을 유지하여 안정적인 추정을 가능하게 한다. 한편, PSO의 도입으로 수행시간이 증가하여 실시간으로 동적 객체를 추정하기 어려워지고, 로봇의 높은 계산 부담으로 이어진다. 이를 해결하기 위해서 GPU 기반의 병렬적 연산을 사용한다. 결과를 통해 기본 PF, PSO-PF, 논문에서 제안하는 병렬적 연산의 PSO-PF, 총 3가지의 필터를 비교하여 제안한 필터의 결과가 객체의 안정적인 추정 및 향상된 연산 수행시간을 확인할 수 있었다. 향후 연구에는 다중 로봇 시스템 환경에서 다중 객체의 위치를 효율적으로 추정하고, GPU 병렬화 계산을 이용한 연산을 확장하여 더 효과적인 수행시간의 감소를 위해 연구할 예정이다.

References

[1] K. W. Kim and S. K. Cho, "Military security threats and countermeasures of future defense

robots", The Journal of Military Robotics Society, Vol. 1, No. 1, pp. 15-21, Apr. 2022.

[2] J. P. Lee, J. Y. Kim, J. H. Ko, K. M. Go, and Y. W. Park, "Fast Adaptive Snake Technique for Realtime Object Segmentation and Recognition in Combat Field", Joint Conference on Institute of Control, Robotics and Systems, The 7th Conference on National Defense Technology, pp. 830-837, Jul. 2011.

[3] I. S. Jang, H. W. Kim, and H. C. Lee, "Object-Centered Spectral Matching for Efficient Environmental Information Fusion in Multiple Small Robot Systems", Korean Institute of Information Technology, Vol. 21, No. 8, pp. 27-38, Aug. 2023. http://dx.doi.org/10.14801/jkiit.2023.21.8.27.

[4] P. R. Gunjal, B. R. Gunjal, H. A. Shinde, S. M. Vanam, and S. S. Aher, "Moving Object Tracking Using Kalman Filter", 2018 International Conference On Advances in Communication and Computing Technology (ICACCT), Sangamner, India, pp. 544-547, Nov. 2018. https://doi.org/10.1109/icacct.2018.8529402.

[5] J. U. Cho, S. H. Jin, X. D. Pham, J. W. Jeon, J. E. Byun, and H. Kang, "A Real-Time Object Tracking System Using a Particle Filter", 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, pp. 2822-2827, Oct. 2006. https://doi.org/10.1109/iros.2006.282066.

[6] A. Ellouze, M. Ksantini, F. Delmotte, and M. Karray, "Single Object Tracking Applied to an Aircraft", 2018 15th International Multi-Conference on Systems, Signals & Devices (SSD), Yasmine Hammamet, Tunisia,pp. 1441-1446, Apr. 2018. https://doi.org/10.1109/ssd.2018.8570663

[7] K. W. Hong, Y. J. Kim, and H. C. Bang, "GPU-Accelerated Particle Filter for Multi-Sensor Multi-Target Tracking", Institute of Control, Robotics and System, Vol. 23, No. 3, pp. 152-156, Mar. 2017. https://doi.org/10.5302/J.ICROS.2017.16.0193.

[8] P. Tian, "A particle filter object tracking based on feature and location fusion", 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, pp. 762-765, Sep. 2015. https://doi.org/10.1109/icsess.2015.7339168.

[9] A. Mukhtar and L. Xia, "Target tracking using color based particle filter", 2014 5th International Conference on Intelligent and Advanced Systems (ICIAS), Kuala Lumpur, Malaysia, pp. 1-6, Jun. 2014. https://doi.org/10.1109/icias.2014.6869447.

[10] J. Kim, S. Nam, G. Oh, S. Kim, S. Lee, and H. C. Lee, "Implementation of a Mobile Multi-Target Search System with 3D SLAM and Object Localization in Indoor Environments", 2021 21st International Conference on Control, Automation and Systems (ICCAS), Jeju, Korea, pp. 2083-2085, Oct. 2021. https://doi.org/10.23919/iccas52745.2021.9650063.

[11] Y. Zheng and Y. Meng, "Swarming particles with multi-feature model for free-selected object tracking", 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, pp. 2553-2558, Sep. 2008. https://doi.org/10.1109/iros.2008.4651004.

[12] Z. Hao, X. Zhang, H. Li, and J. Li, "Video object tracking based on swarm optimized particle filter", 2010 The 2nd International Conference on Industrial Mechatronics and Automation, Wuhan, China, pp. 702-706, May 2010. https://doi.org/10.1109/icindma.2010.5538211.

[13] H. C. Lee, S. K. Park, J. S. Choi, and B. H. Lee, "PSO-FastSLAM: An improved FastSLAM framework using particle swarm optimization", 2009 IEEE International Conference on Systems, Man and Cybernetics, San Antonio, TX, USA, pp. 2763-2768, Oct. 2009. https://doi.org/10.1109/icsmc.2009.5346572.

[14] G. Tong, Z. Fang, and X. Xu, "A Particle Swarm Optimized Particle Filter for Nonlinear System State Estimation", 2006 IEEE International

Conference on Evolutionary Computation, Vancouver, BC, pp. 438-442, Jul. 2006. https://doi.org/10.1109/cec.2006.1688342.

[15] Y. H. Han, H. C. Lee, H. H. Gwon, W. S. Choi, and B. R. Jeong, "Parallelized Particle Swarm Optimization with GPU for Real-Time Ballistic Target Tracking", IEMEK Journal of Embedded Systems and Applications, Vol. 17, No. 6, pp. 355-365, Dec. 2022. https://doi.org/10.14372/IEMEK.2022.17.6.355.

[16] J. W. Kim and H. C. Lee, "GPU-based Acceleration of Particle Filters for Real-Time Target Localization", KSAS 2021 Fall Conference, pp. 1439-1440, Nov. 2021.

[17] J. H. Park, H. C. Lee, H. H. Kwon, Y. J. Hwang, and W. S. Choi, "Parallelized Particle Swarm Optimization on FPGA for Realtime Ballistic Target Tracking", Sensors, Vol. 23, No. 20, pp. 1-23, Oct. 2023. https://doi.org/10.3390/s23208456.

[18] S. S. Kim, J. H. Cho, and D. J. Park, "GPU-based Acceleration of Particle Filter Signal Processing for Efficient Moving-target Position Estimation", IEMEK Journal of Embedded Systems and Applications, Vol. 12, No. 5, pp. 267-275, Oct. 2017. https://doi.org/10.14372/IEMEK.2017.12.5.267.

[19] A. Varsi, J. Taylor, L. Kekempanos, E. P. Knapp, and S. Maskell, "A Fast Parallel Particle Filter for Shared Memory Systems", IEEE Signal Processing Letters, Vol. 27, pp. 1570-1574, Aug. 2020. https://doi.org/10.1109/lsp.2020.3014035.

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", 2016 IEEE Conference on Computer Vision and Pattern Recognition(CVPR), Las Vegas, NV, USA, pp. 779-788, Jun. 2016. https://doi.org/10.1109/cvpr.2016.91.

[21] D. Y. Kim and H. C. Lee, "Depth Sensor-based Detection and Localization of Rescue-Needed Targets in Small Robot Systems", Korean Artificial Intelligence Association & Naver Fall Conference, Nov. 2022.

[21] P. Cunningham and S. J. Delany, "k-Nearest Neighbour Classifiers - A Tutorial", ACM Computing Surveys, Vol. 54, No. 6, pp. 1-25, Jul. 2021. https://doi.org/10.1145/3459665.

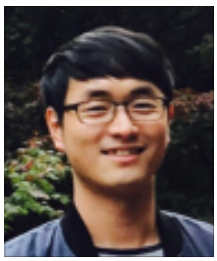
저자소개

김 나 연 (Nayeon Kim)

2019년 2월 ~ 현재 : 금오공과대학교 전자공학부

관심분야 : SLAM, Algorithm acceleration with GPU and FPGA, embedded system

이 헌 철 (Heoncheol Lee)



2006년 8월 : 경북대학교 전자전기컴퓨터학부(공학사)

2008년 8월 : 서울대학교 전기컴퓨터공학과(공학석사)

2013년 8월 : 서울대학교 전기컴퓨터공학과(공학박사)

2013년 9월 ~ 2019년 2월 : 국방과학연구소 선임연구원

2019년 3월 ~ 2023년 8월 : 금오공과대학교 전자공학부 IT융복합공학과 조교수

2023년 9월 ~ 현재 : 금오공과대학교 전자공학부 IT융복합공학과 부교수

관심분야 : SLAM, 자율주행, 인공지능, 알고리즘 가속화