Camera Pose Optimization with Genetic Algorithm in Digital Twin

1st Kwanho Kim and 2nd Kanghyun Jo
dept. of Electrical, Electronic and Computer Engineering
University of Ulsan
Ulsan, South Korea
aarony12@ulsan.ac.kr[1], acejo@ulsan.ac.kr[2]

***Abstract*—With the increase of interest in digital twin, the way to link dynamic objects into digital twin is also important task as well as static environment. However, there are many situations that the developer can not know all about real environment. This paper focus on finding camera extrinsic parameters(6-DOF camera pose) to fix misalignment between real and digital twin images. Due to the absent of knowledge about function of image $I(\theta)$ given camera parameter θ, this work can't use gradient descent algorithm known as a powerful optimization method in deep learning. So this paper propose how to apply GA(Genetic Algorithm) for optimizing camera parameters. Fitness function, one of the most important property in GA, is composed of similarity between two images. This paper propose novel approach to calculate similarity using classical feature extraction and matching methods. Even though this work use classic feature extraction method, errors are estimated within $1.42m$ for camera position and 1.22° for rotation in all experiments(Averagely $0.744m$ and 0.903°). The code of the algorithm can be found in the following link: https://github.com/Kwan-Ho-Kim/Camera-pose-optimization.git.**

***Index Terms*—Digital twin, computer vision, optimization**

I. Introduction

A digital twin is the environment generated by imitating a real environment. Recently, there have been many uses for digital twins in the industry such as smart factory, autonomous driving, monitoring system. In the building of digital twin, linking dynamic objects(such as vehicle, pedestrian, etc.) between digital twin and real one is an important task as well as static environment(building, road, etc.). Dynamic objects are linked with digital twin by observing real sensor data. But sometimes, developer should know exact sensor pose or parameters to make reliable digital twin. This paper focus on finding the pose of a camera using image captured on virtual environment.

Fig. 1 represents the motivation of this work. The misalignment between images are caused by the errors between the camera parameters. This misalignment will inevitably cause serious errors when building real-time digital twin. Before starting this work, it was very hard to manually find camera pose by comparing two images. In other words, even though the developers know a intrinsic parameters of a camera, images are not exactly matched if they do not know the extrinsic parameters of the camera. This paper proposes the method to find camera pose using Genetic Algorithm(GA) and experiment in digital twin.

Fig. 1. Misalignment problem of images from digital twin and real. Blue bounding boxes on the left-bottom image from real environment are generated by YOLOv8 [1] trained with VisDrone [2] dataset. Bounding boxes on the right-bottom image from virtual environment are generated by just mapping those from real image to twin image. The image above is made by overlap two bottom images. Red ellipses show the misalignment between the images captured on real camera and digital twin camera. Because of this problem, the vehicles in the green circle are placed in the wrong position, compared with vehicles in left-bottom image.

II. Related work

A. Camera pose estimations

As the increasing of performance of deep learning networks, many researchers have been trying to utilize neural network for camera pose estimation [3]–[5]. PoseNet [3], one of the most popular camera pose estimation network, used GoogLeNet [6] as a base model and simple Euclidean distance as a loss function to regress 6D pose (3D position and orientation). Nevertheless, they estimate the camera pose within approximately $2m$ and 6° for outdoor scene. However, supervised deep learning algorithm based methods need to gather datasets. In many real-world industrial scenarios, it is often not feasible to collect datasets, and even if collected, it can be time-consuming.

Camera extrinsic parameters can be found by just mathematical method [7]–[10] rather than deep learning based algorithms. These approaches tried to solve Perspective n Points(PnP) algorithms usually used for Simultaneously

Localization And Mapping(SLAM). Solving Perspective-3-Points(P3P) problem [8] utilizes 3 points while Direct Linear Transform(DLT) which is the most classical approach uses at least 6 points to find camera extrinsic parameters. Although these methods don't need to collect datasets in advance, there are still challenges to know corresponding points between two image. To automatically find matching points, feature extracting algorithms [11]–[13] have been utilized so far, but there are still issues about accuracy that depends on feature extracting performance. In section III, this paper proposes the method to relieve the problem of mismatched points from feature extraction algorithms by fitness function of GA.

B. Optimization methods

Gradient descent(GD) is the basic optimization algorithm at deep learning that apply GD to the weights of neural networks [1], [3]–[6]. GD updates the arguments to the direction of decreasing loss. It can also be used as optimization for extrinsic camera parameters if we know the function of an image given camera parameters. But that function cannot be defined in the case of this work. Meta-heuristic algorithms such as GA, Particle Swarm Optimization(PSO) [14] find optimal value without any function between image and camera parameters. GA used at this paper is one of Evolutionary Algorithms [14], [15] that mimic selection of nature. GAVO [16] and its variations [17], [18] utilize GA and PSO to find proper velocity of a robot even though velocities are not the target of meta-heuristic algorithms, generally. Similar with GAVOs [16]–[18], in this paper, 6-DOF camera pose is used as population of GA and found by using novel techniques.

III. Properties of Genetic Algorithm

A. Population

The first step of this work is to manually find similar camera pose as shown in Fig. 1(above image). GA is applied after the this step. To optimize camera extrinsic parameters, 6-DOF camera pose(3D position & rotation) is chosen as population. Population is randomly selected from pre-determined range expressed with noise n_i in (1).

$$\theta_i = \theta_s + n_i \tag{1}$$

In (1), θ represents 6-DOF camera pose used as population. Subscript i represents the chromosome in population and s represents the starting pose determined from manual searching.

B. Fitness

Fitness function is the most important property in GA. What this work should consider is how to calculate similarity between two different view images. However, original methods usually used at template matching to compare two images are not suitable for this work because they can't consider pixel position but only pixel value. To consider pixel position, this paper propose to calculate fitness with feature extraction and matching(This work use SIFT [11] and BFMatcher).

Firstly, features are extracted from two images to be compared. Then, features are matched by matching algorithm such as BFMatcher. To save the memories and remove low-confidence matching, in this paper, only 10% of matching points are remained. After feature extraction and matching, Euclidean distance is calculated for each matching points. Fig. 2 shows the concepts of calculating fitness function.

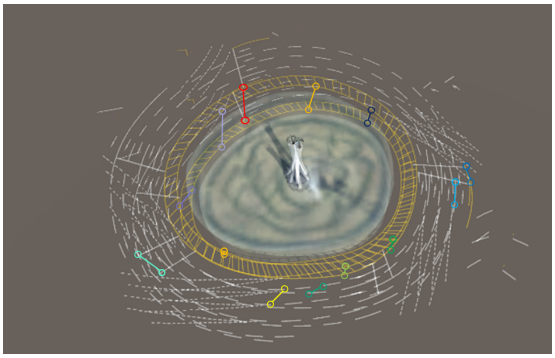

Fig. 2. Similarity between two different view images. Circles with same color and connected line show matching point from feature extraction. Average distance of the lines is utilized as fitness function

$$S = -\frac{1}{N}\sum_{i=1}^{N} ||P_i^C - P_i^T||_2 \tag{2}$$

In (2), S represents the fitness component of similarity between images from target and one of chromosome. 2D vector P_i means the position of i_{th} matching points in an image. N means the number of matching points(Superscript C for each chromosome, T for target). Minus symbol is added to make the equation as a fitness from loss function.

Let us consider one of the offspring watch totally weird space so the features can not be extracted. Nevertheless, sometimes it will keep high fitness due to the low number of matching points and mismatched points. To alleviate this problem, this work introduce the other fitness component K, the number of features, and compound two fitness component using the parameter α. Total fitness function using two components is shown in (3). α is set to 0.1 in this paper. Equation (3) represents full fitness function combining S and K.

$$f = (1-\alpha)S + \alpha K \tag{3}$$

C. Crossover

Crossover method proposed in this paper is inspired by those of GAVOs [16]–[18]. 3D position and rotation of offspring are calculated by linear interpolation(Lerp). To let each populations to explore more space, after Lerp, this work add random noise n^t in predetermined range. Fig. 4 shows how to get offspring with (4).

$$\theta^{t+1} = Lerp(\theta_i^t, \theta_j^t) + n^t \tag{4}$$

Parents are selected from PMF(Probability Mass Function) made by fitness while noise parameters n_i, n^t in (1), (4) from uniform distribution. Probabilities for each chromosome to be selected as a parent are calculated by normalizing the

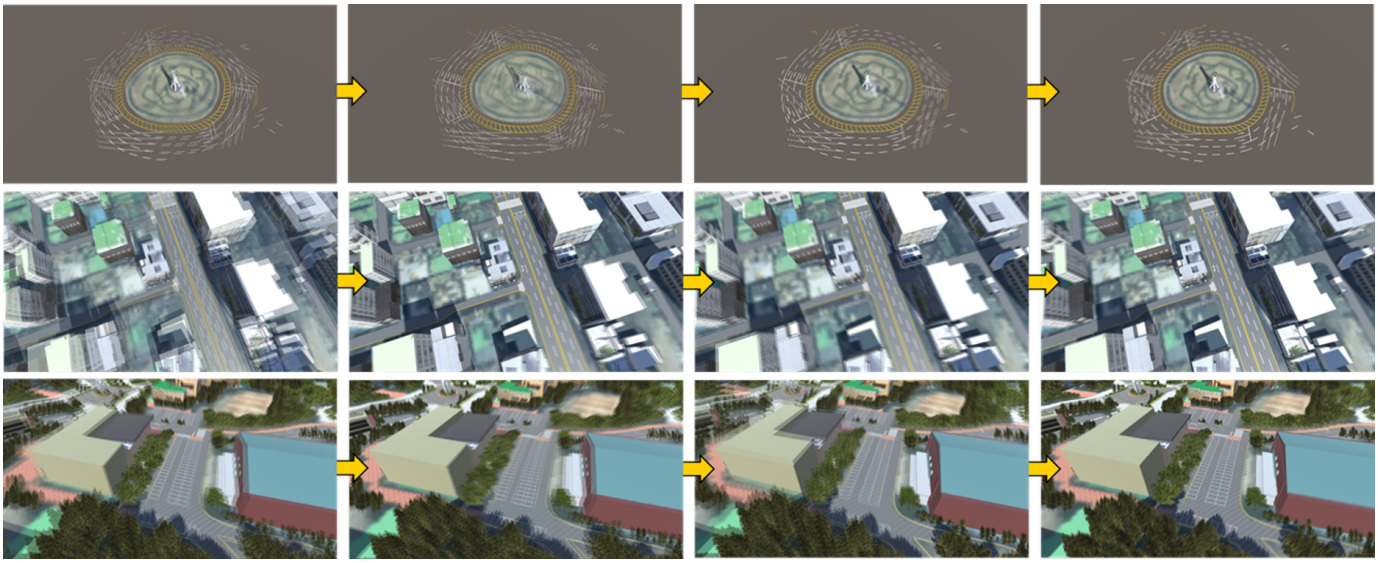

Fig. 3. Visualization of change while optimizing. Each images are made by overlap based on target camera. The leftmost images represent the first step and the rightmost images represent the optimal step(not last step). The images in between represent random steps between two images. The steps to reach best fitness for Rotary, FlatRoad, UOU are 861, 846, 86, respectively.

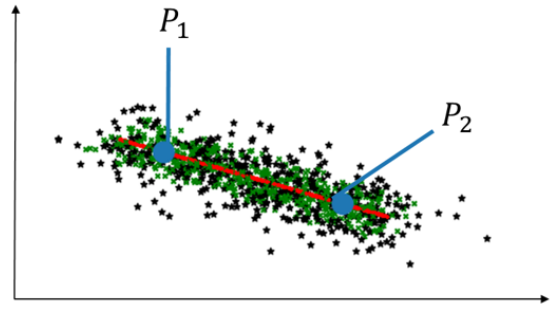

Fig. 4. The range of offspring. This figure shows how the noise let the populations to have more explorations. P_1 and P_2 represent parents expressed as **blue points**. **Red points**(expressed as +) represent the offspring without noise while **green points**(expressed as x) represent those with noise. Linear interpolation can be used for not only the position, but also the rotation, using quaternion. **Black points**(expressed as *) represent mutated offspring.

fitness. In order to prevent overfit, if the variance of fitness in population becomes less than threshold(20 in this paper), iteration restart at the global best pose.

D. Mutation

In this paper, mutation probability is set to 0.25 for each genes(e.g. 3D position & rotation). If a gene is selected as a mutation, it is randomly regenerated by adding noise of same range as n_i in (1). In other words, all genes explore more wide solution space with a probability of 0.25 as shown in Fig. 4.

IV. Experiments

A. Environment

This paper experiment in three environment to validate the performance: Rotary, Flat Road, UOU(University of Ulsan). The environments are shown in Fig. 3 with the visualization of similarity of two images during optimization. Adjustable parameters are shown in Table I. To ensure the method of this work is also suitable for general case, parameters are set to same values for all environment except to Fitness Compound Ratio which thoroughly depends on environments. As shown in Fig. 3, Rotary has small amount of features while Flat Road and UOU has a lot.

TABLE I
Adjustable parameters and the values used

Parameters	Environment		
	Rotary	***Flat road***	***UOU***
Fitness Compound Ratio α	0.1	0.01	0.01
Feature Filtering Ratio	0.1		
Overfit Threshold	20		
Num. of Population	20		
Num. of Exchange	18		
Pos. Random Range	3		
Rot. Random Range	4		
Pos. Expand Ratio	1.25		
Rot. Expand Ratio	1.25		
Mutation Probability	0.25		
Iteration	1000		

B. results

Fig. 3 shows the changes of images overlaped on the target until algorithm find optimal pose. Even though some images seem to have lower performance than previous one, the final images show the algorithm of this work can figure out optimal camera pose.

Fig. 5 and Table II shows the changes during optimization. Similarity S and Fitness f are calculated by (2), (3), respectively. Position error means distance between positions of the target and chromosome. Rotation error means angle(°)

TABLE II
CHANGES OF METRIC DURING OPTIMIZATION.

		1	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
Rotary	***f(Fitness)***	32.97	143.06	160.49	174.82	174.82	174.82	174.82	174.82	174.82	181.97	196.733
	Pos. Error [m]	3.14	0.82	1.12	0.257	0.257	0.257	0.257	0.257	0.257	0.204	0.332
	Rot. Error [°]	10.46	1.84	1.80	1.18	1.18	1.18	1.18	1.18	1.18	0.58	1.22
Flat Road	***f(Fitness)***	-53.01	28.545	33.57	33.57	33.57	33.83	33.83	33.83	33.83	34.01	34.01
	Pos. Error [m]	3.215	0.339	0.780	0.780	0.780	0.356	0.356	0.356	0.356	0.48	0.48
	Rot. Error [°]	6.80	0.87	0.98	0.98	0.98	0.59	0.59	0.59	0.59	0.79	0.79
UOU	***f(Fitness)***	92.27	85.27	80.87	111.75	116.269	107.94	114.81	85.60	93.32	110.72	84.66
	Pos. Error [m]	4.449	4.87	4.68	3.955	3.56	1.98	3.60	5.287	7.57	2.73	5.72
	Rot. Error [°]	0.88	3.17	1.46	1.39	0.79	0.67	0.48	3.20	1.33	1.70	3.00

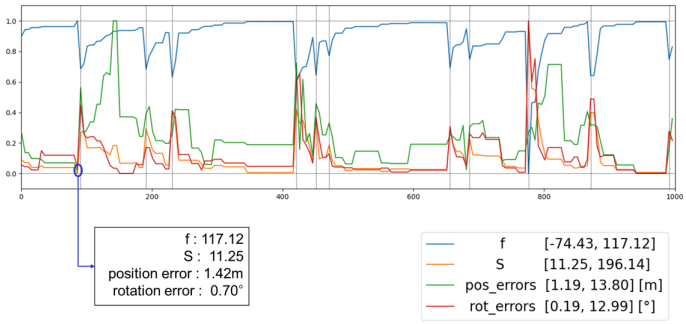

Fig. 5. Changes of metric during optimization in UOU environment. All of the evaluation metrics are normalized to compare changes easier. Ranges in the legend shows the maximum and minimum value that correspond to horizontal dashed-lines in the graph. Vertical lines show the restart points caused by preventing overfit. Note that the large the fitness(f), the better. Other metrics should be small(S is inversely proportional to Similarity in (2)).

between directions of the target and chromosome. An interval between restart points is defined as the episode in this paper. Although optimal pose were found at the first episode in this case, by exploring more solution space, graph shows the algorithm try to find whether there is another optimal value or not. The fact that fitness can not reach global best in all of the episodes implicitly tell us that convergence to the optimal point within an episode depends on the restarting point. Also, the fact that fitness reach global best at some episode means convergence to the optimal point with total iteration does not depend on starting point.

In Table II, changes of metrics for this work are quantitatively represented. In environments of Rotary and Flat Road, optimal poses are obtained at final step. However, in the case of UOU environment, optimal poses are obtained at 86_{th} step and the metrics in the table are sampled every 100 steps. So the values of metrics are represented in Fig. 5. As a results, this work estimate camera pose with errors within $1.42m$ for position, $1.22°$ for rotation in all experiments(Averagely $0.744m$ and $0.903°$).

ACKNOWLEDGMENT

This result was supported by "Regional Innovation Strategy (RIS)" through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(MOE)(2021RIS-003)

REFERENCES

[1] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics
[2] P. Zhu, L. Wen, X. Bian, H. Ling, and Q. Hu, "Vision meets drones: A challenge," 2018.
[3] A. Kendall, M. Grimes, and R. Cipolla, "Convolutional networks for real-time 6-dof camera relocalization," *CoRR*, vol. abs/1505.07427, 2015. [Online]. Available: http://arxiv.org/abs/1505.07427
[4] H. Hu, A. Wang, M. Sons, and M. Lauer, "Vipnet: An end-to-end 6d visual camera pose regression network," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–7.
[5] H. Chen, P. Wang, F. Wang, W. Tian, L. Xiong, and H. Li, "Epro-pnp: Generalized end-to-end probabilistic perspective-n-points for monocular object pose estimation," 2022.
[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842
[7] S. Urban, J. Leitloff, and S. Hinz, "Mlpnp - A real-time maximum likelihood solution to the perspective-n-point problem," *CoRR*, vol. abs/1607.08112, 2016. [Online]. Available: http://arxiv.org/abs/1607.08112
[8] Z. Zhang, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 1330 – 1334, 12 2000.
[9] R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.
[10] E.-S. Kim and S.-Y. Park, "Extrinsic calibration of a camera-lidar multi sensor system using a planar chessboard," in *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*, 2019, pp. 89–91.
[11] T. Lindeberg, *Scale Invariant Feature Transform*, 05 2012, vol. 7.
[12] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417.
[13] X. Wang, Z. Liu, Y. Hu, W. Xi, W. Yu, and D. Zou, "Featurebooster: Boosting feature descriptors with a lightweight neural network," 2023.
[14] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4. IEEE, 1995, pp. 1942–1948.
[15] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," 2017.
[16] Z. Gyenes, L. Bölöni, and E. G. Szádeczky-Kardoss, "Can genetic algorithms be used for real-time obstacle avoidance for lidar-equipped mobile robots?" *Sensors*, vol. 23, no. 6, p. 3039, 2023.
[17] J. Kim, K. Kim, and K. Jo, "Obstacle avoidance in dynamic environment using particle swarm optimization and kalman filter," in *2023 23rd International Conference on Control, Automation and Systems (ICCAS)*, 2023, pp. 1199–1203.
[18] ——, "Genetic algorithm based obstacle avoidance for 4-wheeled robot," in *2023 62nd Annual Conference of the Society of Instrument and Control Engineers (SICE)*, 2023, pp. 297–300.