International Journal of Control, Automation and Systems 17(9) (2019) 2284-2296
http://dx.doi.org/10.1007/s12555-018-0134-6
ISSN:1598-6446 eISSN:2005-4092
http://www.springer.com/12555

Convolutional Neural Network for Monocular Vision-based Multi-target Tracking

Sang-Hyeon Kim and Han-Lim Choi*

Abstract: This paper addresses multi-target tracking using a monocular vision sensor. To overcome the fundamental observability issue of the monocular vision, a convolutional neural network (CNN)-based method is proposed. The method combines a CNN-based multi-target detection into a model-based multi-target tracking framework. While previous CNN applications to image-based object recognition and tracking focused on prediction of region of interest (RoI), the proposed method allows for prediction of the three-dimensional position information of the moving objects of interest. This is achieved by appropriately construct a network tailored to the moving object tracking problems with potentially occluded objects. In addition, the cubature Kalman filter integrated with a data association scheme is adopted for effective tracking of nonlinear motion of the objects with the measurements information from the learned network. A virtual simulator that generates the trajectories of the target motions and a sequence of images of the scene has been developed and used to test and verify the proposed CNN scheme. Simulation case studies demonstrate that the proposed CNN improves the position accuracy in the depth direction substantially.

Keywords: CNN-based multi-target detection, indoor quadrotor tracking, monocular vision, multi-target tracking, three-dimensional indoor quadrotor simulator.

1. INTRODUCTION

Tracking multiple moving targets is an important task in many application domains such as automated surveillance, traffic monitoring, image-based navigation, and multi-agent missions [1–6]. Object detection that extracts the target information from the sensor read is a core part of the multi-target tracking system. While specific theories and algorithms for multi-target tracking strongly depend on the choice of sensors, one popular choice due to its cost-effectiveness and availability is to use vision sensors [9–13]. However, the use of a vision sensor suffers a fundamental limitation in extracting three-dimensional information from the target of interest, as the image is a two-dimensional projection of the three-dimensional world. Stereo processing is one standard and typical solution to produce three-dimensional information, i.e., bearing and depth, but, the compatibility of such stereo system depends on the overall system requirement and availability. Therefore, monocular vision is often adopted as a sensor system for the sake of cost and weight.

However, due to the aforementioned fundamental limitation, the use of monocular vision, which is of the present work's interest, requires some other mechanism to overcome this limitation; often this is achieved by moving the sensors and taking information from different perspectives. As such, monocular vision-based tracking has often been interested in keeping track of target locations within the two-dimensional image plane rather than tracking the three-dimensional position coordinates.

On the other hand, there have been significant advances in deep learning-based image detection and tracking. In particular, the convolutional neural network (CNN) [7] has been established as an effective class of models for the image interpretation; it has been shown to achieve a state-of-art result on image recognition and object region proposal [8]. There have been noticeable applications of CNNs to multi-object detection, including R-CNN [14], Fast R-CNN [15], Faster R-CNN [16], YOLO [17, 18], and SSD [19]. Faster R-CNN [16] is composed of two modules which are a deep fully convolutional network that proposes regions and detector that uses the proposed regions. YOLO [17, 18] has a single convolutional neural network which simultaneously detects objects and classi-

Manuscript received March 3, 2018; revised September 7, 2018 and March 7, 2019; accepted March 20, 2019. Recommended by Associate Editor Vu Nguyen under the direction of Editor Won-jong Kim. This work was supported in part by ICT R&D program of the Ministry of Science & ICT via Institute for ICT Planning and Evaluation (#R-20150223-000167), and in part by Defense Acquisition Program Administration via High-speed Vehicle Research Center and Agency for Defense Development (#UD170018CD).

Sang-Hyeon Kim and Han-Lim Choi are with the Department of Aerospace Engineering & KI for Robotics, KAIST, 291 Daehak-ro, Yuseong-gu, Daejeon 34141, Korea (e-mails: {k3special, hanlimc}@kaist.ac.kr).
* Corresponding author.

fies the detected objects. SSD [19] also has a single convolutional neural network like YOLO, but the neural network architecture is different from YOLO. YOLO and SSD are known to be more effective than Faster R-CNN in computing speed.

It should be noted that these conventional CNN-based object recognition techniques are designed to predict the *Region of Interest* (RoI) and the object class defined by VOC [23] or COCO [24] data set. This is sufficient if the goal for tracking is to obtain the target position information on the image plane. However, if the objective of tracking is to have knowledge of the motion of the target, more specifically, the center of mass of the target, keeping track of RoI within the image plane may not be sufficient. For this, three-dimensional position information is needed either by direct sensing or by inference.

For this reason, this paper proposes to extend the existing CNN architecture to enable inference of depth information as well as the bearing information. The proposed CNN-based multi-target detection method recognizes the three-dimensional position of the target based on the entire image of the target. The proposed method features a combination of the CNN-based multi-target detection and the conventional multi-target tracking algorithm.

Fig. 1 shows the CNN for multi-target detection. The proposed CNN only uses a single image to generate the three-dimensional positions of targets. These three-dimensional position information is used as the measurement information of the target tracking algorithm.

The three-dimensional position information obtained using the proposed CNN model is input as measurements of the nonlinear tracking filter that tracks quadrotors. The method used in this paper as a nonlinear tracking filter is Cubature Kalman Filter (CKF) [20], which is based on the same principle as the Unscented Kalman Filter (UKF) [21] but is known to be computationally more efficient than the UKF. In order to track multiple targets accurately,

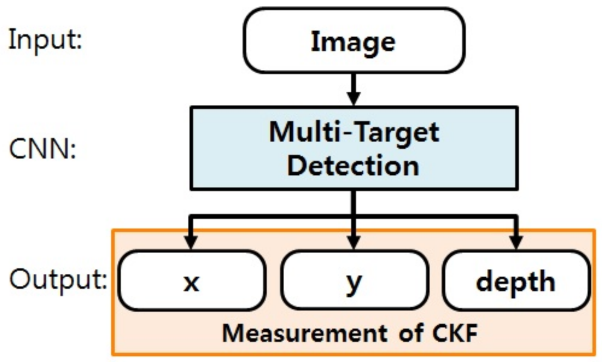

Fig. 1. CNN for multi-target detection. The proposed CNN model recognizes the three-dimensional position (x, y, depth) based on the single image. (x, y) is the center of mass position on the image.

the measurements obtained through CNN should be appropriately assigned to each target. The nearest neighbor method is used for this data association procedure. The nearest neighbor method is a method that assigns the closest measure to the current estimate of the target.

Three-dimensional indoor quadrotor simulator is developed for generating the training data set which contains images of the single/multi-quadrotor and the ground truth position information of the single/multi-quadrotor, and for testing the proposed framework. We used Python language and OpenGL module [22] to develop this simulator. Experiments are conducted to demonstrate the single/multi-target tracking performances of the proposed framework.

The contributions of this paper are as follows:

- A CNN model is proposed that can recognize the three-dimensional positions of multiple targets only using a monocular vision sensor. For this purpose, the CNN model can obtain the distance information from the sensor to the target based on the shape information of the target.
- A multi-target tracking framework is proposed which is a combination of a conventional multi-target tracking method and the CNN-based multi-target detection method for sustainable target tracking.
- A 3-D simulator is developed and applied to the performance analysis of the proposed technique. Using this simulator, a virtual indoor flight environment and a quadrotor model are implemented and images for learning the proposed CNN model are generated.

The organization of this paper is as follows: Section 2 presents the proposed CNN architecture for detecting multiple targets and recognizing the three-dimensional position. Section 3 describes the multi-target tracking algorithm using CKF. Section 4 introduces the 3-D indoor quadrotor simulator for training data generation and performance evaluation of the proposed multi-target tracking framework. Section 5 shows experiment results for demonstrating the single or multiple targets tracking performances of the proposed approach.

2. CONVOLUTIONAL NEURAL NETWORK MODEL FOR THREE-DIMENSIONAL POSITION RECOGNITION

In this section, our CNN-based three-dimensional position recognition method is presented. The overall CNN architecture is inspired by YOLO [17, 18]. The key difference between the proposed CNN and YOLO is that the proposed CNN obtains depth measurement with the center coordinates while YOLO obtains Region of Interest (RoI) of the object.

2.1. Proposed CNN model architecture

The proposed CNN can detect multiple targets using single neural network like YOLO [17, 18]. By using our CNN, the three-dimensional position is predicted based on the features obtained from the entire input image. The architecture of proposed CNN model is shown in Fig. 2. The size of the input image is (480,480,3) and the final output is (6,6,8) tensor. The CNN model consists of 8 convolutional layers and 1 fully connected layer.

The CNN-based three-dimensional position recognition method divides the input image into 6×6 grids for recognizing the centroid position of the target and the distance to the target (depth). Each grid cell carries four parameters – the centroid position of target, (X_p, Y_p), the distance to the target (depth), Z_p, and the objectiveness, $O \in [0,1]$, which indicates the probability of being an object. To handle an occluded target, an additional layer with the aforementioned four parameters is added. Therefore, the size of output tensor is $((6 \times 6) \times 4) \times 2$. More detailed explanation is as follows.

2.2. CNN model training

The framework of CNN model training is illustrated in Fig. 3. The ground-truth three-dimensional position in the inertial frame, whose center is at the camera and z-direction is directed to the object, is converted into the image frame. In other words, the inertial position coordinates is converted to the pixel position and the depth (X_p, Y_p, Z_p):

$$\begin{aligned} X_p &= \frac{W \tan^{-1}(X/Z)}{\alpha} + \frac{W}{2}, \\ Y_p &= \frac{H^2 \tan^{-1}(Y/Z)}{W\alpha} + \frac{H}{2}, \\ Z_p &= \frac{Z}{L_{norm}}, \end{aligned} \tag{1}$$

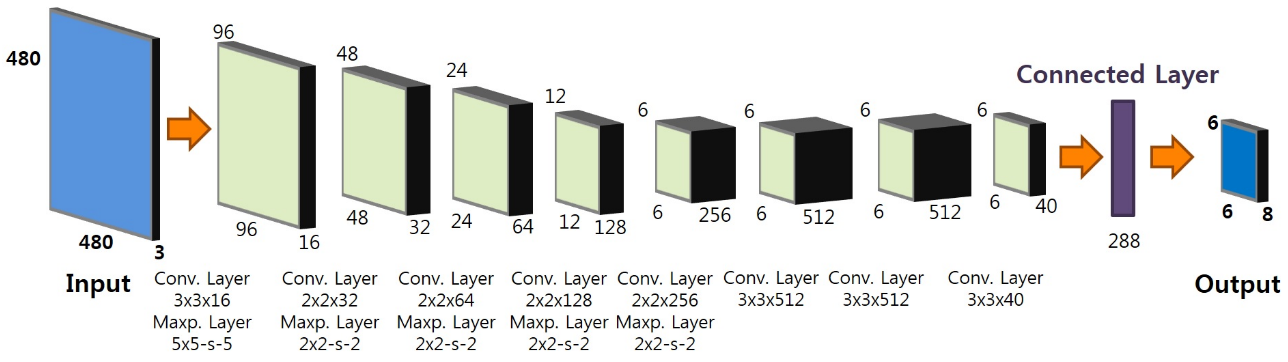

Fig. 2. The proposed CNN model architecture consisting of 8 convolutional layers (five of which is combined with max-pooling layers) and 1 fully connected layer at the back. The size of input image is (480,480,3) and the final output is (6,6,8) tensor. The number of total trainable parameters is 4,314,344.

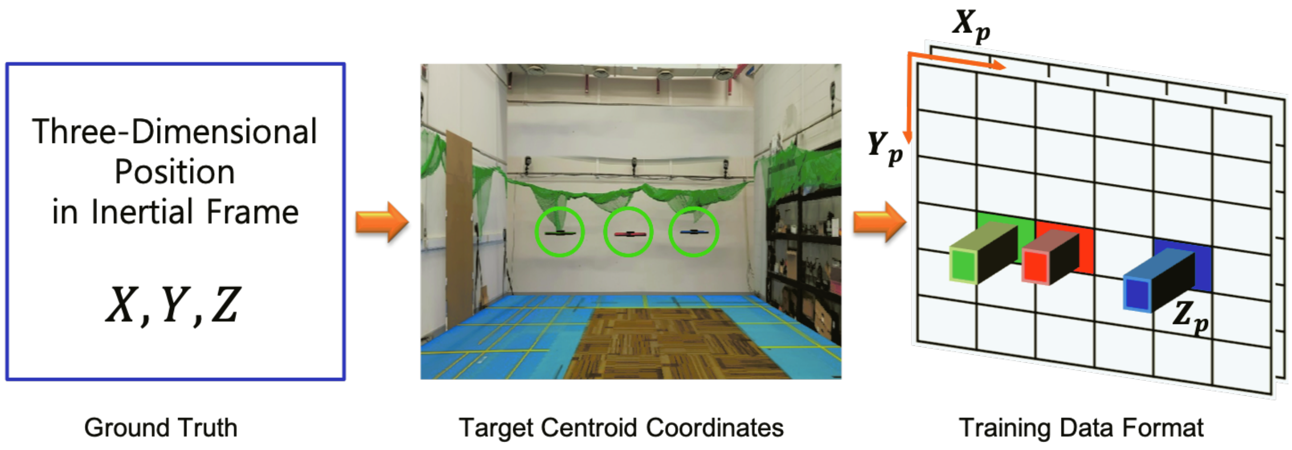
Three-Dimensional Position in Inertial Frame
X, Y, Z
Ground Truth
Target Centroid Coordinates
X_p
Y_p
Z_p
Training Data Format

Fig. 3. Framework of CNN model training. To generate training data of CNN model, ground-truth three-dimensional position in inertial frame is converted into the image frame. The training data contains the centroid position (X_p, Y_p) and depth measurement Z_p.

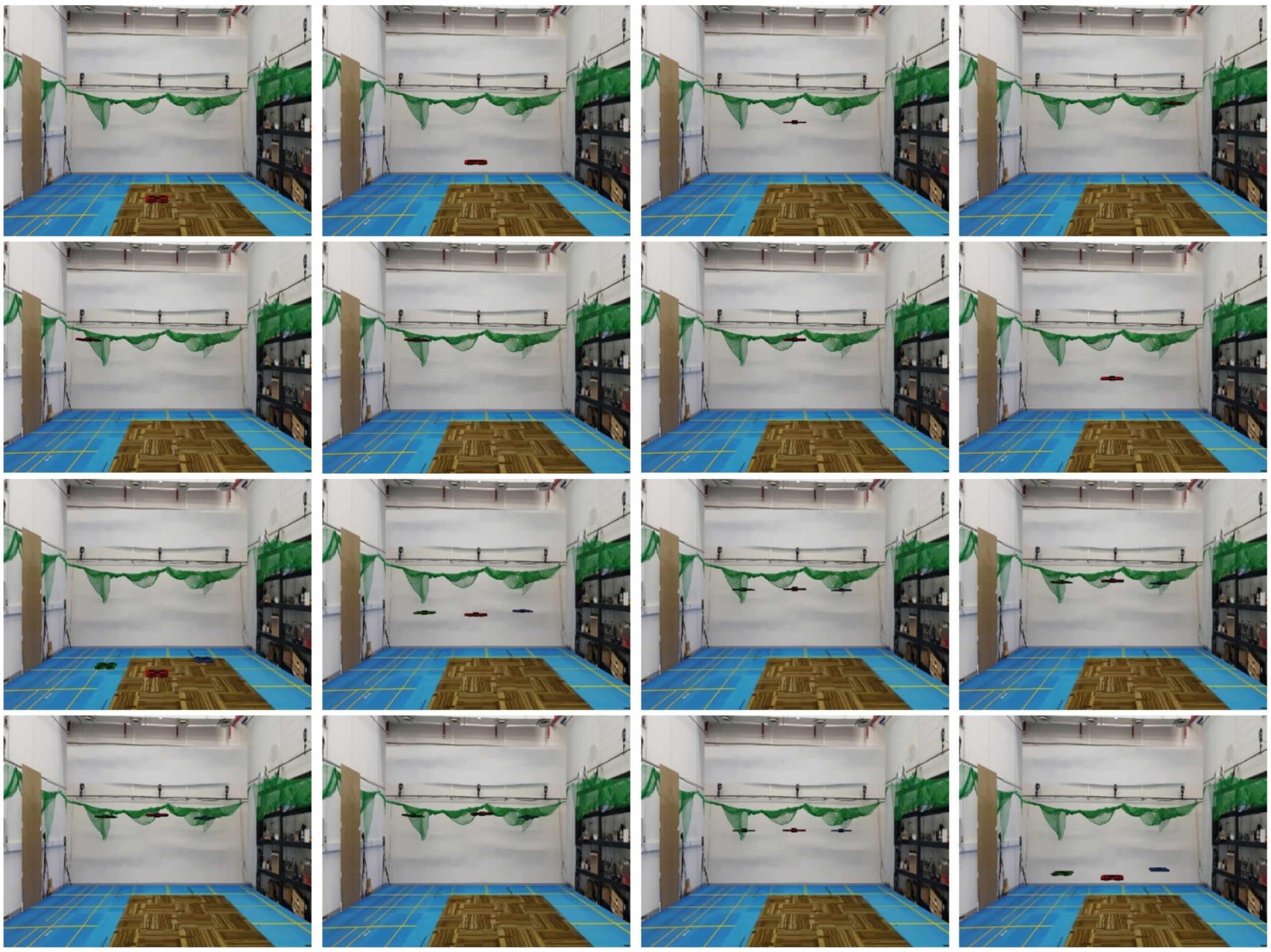
Fig. 4. Example of training image set.

where W is the image width, H is the image height, and $W > H$. α is the camera's field of view (FoV) and L_{norm} is the normalization parameter. Note that the following equivalently holds:

$$\begin{aligned} X &= Z\tan\left(\frac{\alpha(X_p - 0.5W)}{W}\right), \\ Y &= Z\tan\left(\frac{\alpha(Y_p - 0.5H)}{H^2/W}\right), \\ Z &= L_{norm}Z_p. \end{aligned} \tag{2}$$

If the centroid coordinates (X_p, Y_p) of a target is within a certain grid cell, that grid cell is responsible for recognition of the position of the target predicting X_p, Y_p, Z_p, and O of the target. Some tie-breaking rule can be applied if the centroid is one the boundary.

Training data for the proposed CNN can be obtained by either real experiments and/or virtual simulated data; this work particularly takes advantage of the virtual simulation whose details will be explained in Section 4. In essence, this simulator generates images of the motion of target vehicles – quadrotor drones; some reference trajectories are given to these vehicles and random perturbation is added to diversify the motion characteristics. Examples of images used in the training are depicted in Fig. 4. The images used for the training CNN model are shown in Fig. 4.

The training process optimizes the following custom loss function:

$$\begin{aligned} \text{Loss} =& \lambda_1 \sum_{i=1}^{m} O_i[(X_{p,i} - \hat{X}_{p,i})^2 + (Y_{p,i} - \hat{Y}_{p,i})^2] \\ &+ \lambda_2 \sum_{i=1}^{m} O_i \left(\sqrt{Z_{p,i}} - \sqrt{\hat{Z}_{p,i}}\right)^2 \\ &+ \sum_{i=1}^{m} (O_i - \hat{O}_i)^2, \end{aligned} \tag{3}$$

where the subscript i indicates the i-th entry of the output tensor considering the grid size and the overlapping, $m = (6 \times 6) \times 2$ is the total number of these entries, λ_1, λ_2 are loss weight parameters, and $\hat{(\cdot)}$ refers to the predicted value while the plain variable refers to the truth. Note that this loss function penalizes the prediction error only if the object center is present in the grid cell. A gradient descent method with adaptive learning rates, the Adam (Adaptive moment estimation) optimizer [28] is used in the process of optimization.n addition to storing an exponentially decaying average of past squared gradients like Adadelta [29] and RMSprop [30], Adam also keeps an exponentially decaying average of past gradients.

In order to apply the CNN model to an environment with disturbance, training data should be generated through several iterative experiments. When the training data is simply generated repeatedly, the size of the training data increases in proportion to the number of iterations. This causes training time increasing and wastes memory resources for storing training data.

This paper suggests the concept of "selective training" to manage the increase in the training data size in an environment with disturbances and noises. The selective training initially generates training data for the entire time domain and trains the CNN model. Then, a set of multi-target tracking scenarios are simulated using this trained CNN model. Due to the disturbance, the target position would vary in this simulation step, which causes large tracking errors. In the second round of training, training data is generated only when the predicted position has a large error. Then, the first training data and the second selective training data are used for further training of the network. This process is repeated until the training data is no longer needed to successfully accomplish the multi-target tracking mission. This selective procedure is found to reduce the total size of the training data for a given tracking mission.

3. MULTI-TARGET TRACKING ALGORITHM

Fig. 5 illustrates the overall architecture of the multi-target tracking algorithm that estimates the target state

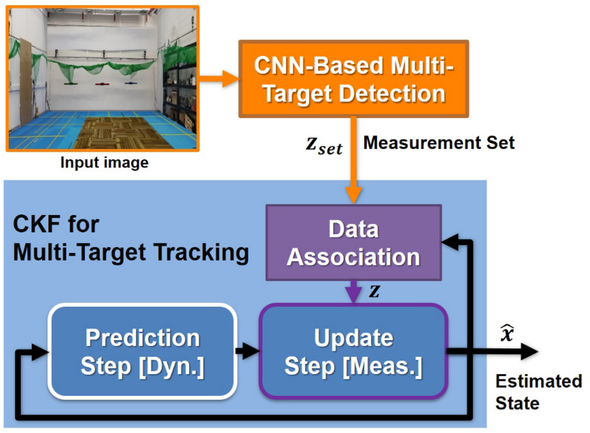

Fig. 5. Multi-target tracking framework. Multi-target tracking algorithm uses CKF [20] and nearest neighbor data association method. CNN is used for generating measurements of targets.

variables, i.e., position, velocity, and attitude, using the position measurements from the CNN. A data association technique is also adopted to prevent intermixing of the tracks.

To track state variables of multiple targets, this paper adopts the cubature Kalman filter (CKF) [20] with the nearest neighbor data association scheme. Cubature rules are used in order to calculate the statistical moments of a random variable approximately which is transformed through a nonlinear function [20]. The moments are expressed as multidimensional integrals that cannot generally be solved in a closed form; some type of numerical approximation is often used. The cubature rule approximates a multi-dimensional integral as a weighted sum of integrands evaluated at specified set of nodes. Consider an integral $I(f) = \int_D f(\mathbf{x})w(\mathbf{x})d\mathbf{x}$ where $f(\cdot)$ is an arbitrary nonlinear function (dynamic or measurement equation) of the state vector $\mathbf{x} \in \mathbf{R}^n$ and $w(\mathbf{x})$ is the associated weight. $D \subseteq \mathbf{R}^n$ is the region of integration and $w(\mathbf{x}) \geq 0$. The main idea is that the integral $I(f)$ can be approximated using $2n$ cubature points $\mathbf{x}_q$, $q = 1, \ldots, 2n$, and the associated weights, w_q, such that:

$$I(f) \approx \sum_{q=1}^{2n} w_q f(\mathbf{x}_q). \tag{4}$$

Assuming that a weighted sum of Gaussian density functions is again a Gaussian,

$$I(f) = \int_{\mathbf{R}^n} f(\mathbf{x})\mathcal{N}(\mathbf{x};0,I)d\mathbf{x} \approx \sum_{q=1}^{2n} w_q f(\xi_q), \tag{5}$$

where $\mathcal{N}(\cdot;\cdot)$ represents the probability density function of a Gaussian distribution, and

$$\xi_q = (-1)^q \mathbf{e}_q, \tag{6}$$

$$w_q = \frac{1}{2n}, \quad q = 1, 2, \ldots, 2n, \tag{7}$$

where $\mathbf{e}_q$ is the q-th unit vector whose q-th entry is one and others are all zero.

Suppose that the target motion is expressed as the following system dynamics:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) + v_k \tag{8}$$

with some nonlinear function $f(\cdot)$, where $v_k \in \mathbf{R}^n$ is the zero-mean white Gaussian process noise with covariance $Q_k \succeq 0$. The measurement equation is given by

$$\mathbf{y}_k = h(\mathbf{x}_k) + n_k \tag{9}$$

with output $\mathbf{y}_k \in \mathbf{R}^l$ and a nonlinear observation function $g(\cdot)$. $n_k \in \mathbf{R}^l$ is the zero-mean Gaussian sensing noise with covariance R_k. The noise covariance is assumed to be positive definite, $R_k \succ 0$.

Like the other variants of Kalman filters, the CKF features a recursive procedure of initialization, prediction, and update steps. The recursion begins with the initial condition on the state estimate $\hat{\mathbf{x}}_0$ and the estimation error covariance $\mathbf{P}_0$:

$$\hat{\mathbf{x}}_0 \triangleq \mathrm{E}[\mathbf{x}_0], \quad \mathbf{P}_0 \triangleq \mathrm{E}[(\mathbf{x}_0-\hat{\mathbf{x}}_0)(\mathbf{x}_0-\hat{\mathbf{x}}_0)^T], \tag{10}$$

which are often provided as a priori knowledge or initialized in a non-informative way.

The prediction step can be written as follows:

- Draw cubature points ξ_q.
- Propagate the cubature points. The matrix square root is the lower triangular cholesky factor.

$$\mathcal{X}_{q,k-1|k-1} = \sqrt{\mathbf{P}_{k-1|k-1}}\xi_q + \hat{\mathbf{x}}_{k-1|k-1}. \tag{11}$$

- Evaluate the cubature points with the dynamic model function.

$$\mathcal{X}_{q,k|k-1} = f(\mathcal{X}_{q,k-1|k-1}). \tag{12}$$

- Estimate the predicted mean and the error covariance.

$$\hat{\mathbf{x}}_{k|k-1} = \frac{1}{2n}\sum_{q=1}^{2n}\mathcal{X}_{q,k|k-1}, \tag{13}$$

$$\mathbf{P}_{k|k-1} = \frac{1}{2n}\sum_{q=1}^{2n}(\mathcal{X}_{q,k|k-1}-\hat{\mathbf{x}}_{k|k-1}) \times (\mathcal{X}_{q,k|k-1}-\hat{\mathbf{x}}_{k|k-1})^T + \mathbf{Q}_{k-1}. \tag{14}$$

The update step can be written as follows:

- Evaluate the cubature points with the measurement model function.

$$\mathcal{Y}_{q,k|k-1} = h(\mathcal{X}_{q,k|k-1}). \tag{15}$$

- Estimate the predicted measurement.

$$\hat{\mathbf{y}}_{k|k-1} = \sum_{q=1}^{2n}\mathcal{Y}_{q,k|k-1}. \tag{16}$$

- Estimate the innovation covariance matrix.

$$\mathbf{S}_{k|k-1} = \frac{1}{2n}\sum_{q=1}^{2n}(\mathcal{Y}_{q,k|k-1}-\hat{\mathbf{y}}_{k|k-1}) \times (\mathcal{Y}_{q,k|k-1}-\hat{\mathbf{y}}_{k|k-1})^T + \mathbf{R}_k. \tag{17}$$

- Estimate the cross-covariance matrix.

$$\mathbf{P}_{xy,k|k-1} = \frac{1}{2n}\sum_{q=1}^{2n}(\mathcal{X}_{q,k|k-1}-\hat{\mathbf{x}}_{k|k-1}) \times (\mathcal{Y}_{q,k|k-1}-\hat{\mathbf{y}}_{k|k-1})^T. \tag{18}$$

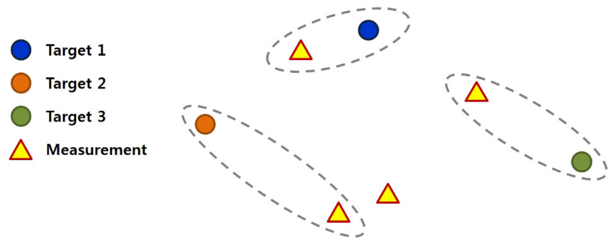

Fig. 6. Nearest neighbor method for data association.

- Calculate the Kalman gain.

$$\mathbf{K}_k = \mathbf{P}_{xy,k|k-1}\mathbf{S}_{k|k-1}^{-1}. \tag{19}$$

- Estimate the updated state and the error covariance.

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}_{k|k-1}), \tag{20}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_{k|k-1}\mathbf{K}_k^T. \tag{21}$$

For accurate tracking of multiple targets, data association that determines to which target a measurement is corresponding is crucial. While many different methods have been proposed for data association, this work takes advantage of a simple nearest neighbor method [31] . The narest neighbor method assigns the closest measurement to the current estimate of the target as depicted in Fig. 6. For example, target 2 will be assigned the next closest measure since the closest measure is assigned to target 1.

4. THREE-DIMENSIONAL INDOOR QUADROTOR SIMULATOR

4.1. Simulation engine: 6-DoF quadrotor dynamics and controller

The quadrotor considered herein consists of four fixed-pitch-angle blades. The quadrotor kinematics model is illustrated in Fig. 7. Each rotor produces thrust and torque

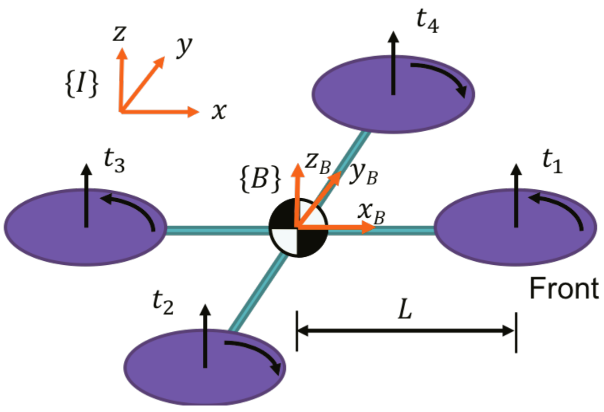

Fig. 7. Quadrotor kinematics with $\{B\}$ and $\{I\}$ representing the body frame and inertial frame.

whose combination generates the main thrust, the roll, pitch, and yaw torque acting on the quadrotor [25–27].

In this paper, the quaternion-based 6-DoF dynamics equation is used; the state vector for quadrotor motion is defined as:

$$\mathbf{x} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \boldsymbol{\omega}^T & \mathbf{q}^T \end{bmatrix}^T, \tag{22}$$

$$\boldsymbol{\omega} = \begin{bmatrix} p & q & r \end{bmatrix}^T, \tag{23}$$

$$\mathbf{q} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T, \tag{24}$$

where (x, y, z) are three-dimensional position coordinate in the inertial frame [1], (p, q, r) are angular velocity components in the body frame. $\mathbf{q}$ is the quaternion representing attitude of the quadrotor and its norm satisfies $||\mathbf{q}||_2 = 1$ [32] [2]. By applying both the Euler-Lagrange and the Newton-Euler approaches, the quadrotor 6 degree-of-freedom (DoF) dynamics can be represented as

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \frac{1}{m}\mathbf{R}_{DCM}\mathbf{t}_B + \mathbf{f}_D, \tag{25}$$

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}(\boldsymbol{\tau}_B - \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega})), \tag{26}$$

$$\dot{\mathbf{q}} = \frac{1}{2}\begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix}\boldsymbol{\omega}, \tag{27}$$

where g is the gravitational acceleration, m is the total mass of the quadrotor, $\mathbf{J}$ is the moments of inertia matrix, $\mathbf{R}_{DCM}$ is the direction cosine matrix (DCM) to represent the coordinate transformation from the body frame to the inertial frame, $\mathbf{t}_B$ is the total thrust acting on the quadrotor body, τ_B is the torque command acting on the quadrotor and $\mathbf{f}_D$ is the aerodynamic drag vector.

A standard way of designing a position controller for the quadrotor is to use the proportional-derivative (PD) control method. When the desired position (x_d, y_d, z_d) and the yaw angle ψ_d are determined, the rotational speed (u_1, u_2, u_3, u_4) of each motor is calculated. z_d is related to the altitude of the quadrotor. The motor rotational speed input value, u_d, required to maintain the altitude z_d by compensating the gravity is calculated as follows:

$$u_d = P_z(z_d - z) - D_z\dot{z} + \sqrt{\frac{mg}{4k}}, \tag{28}$$

where P_z and D_z are the proportional and the derivative control gains, and k is the thrust coefficient. Regarding the attitude, the quaternion and the attitude angles are related as

$$\phi = \tan^{-1}\left(\frac{2(q_0q_1 + q_2q_3)}{1 - 2(q_1^2 + q_2^2)}\right), \tag{29}$$

$$\theta = \sin^{-1}\left(2(q_0q_2 - q_3q_1)\right), \tag{30}$$

$$\psi = \tan^{-1}\left(\frac{2(q_0q_3 + q_1q_2)}{1 - 2(q_2^2 + q_3^2)}\right). \tag{31}$$

The motor rotational speed input value, τ_ψ for ψ_d is calculated as follows:

$$\tau_\psi = P_\psi(\psi_d - \psi) - D_\psi r, \tag{32}$$

where P_ψ and D_ψ are the proportional gain and the derivative gain, respectively. To move the quadrotor to the desired location, the pitch angle and the roll angle need to be changed accordingly. Thus, the desired pitch, θ_d and roll, ϕ_d angles can be calculated in terms of the desired location:

$$\begin{bmatrix} \theta_d \\ \phi_d \end{bmatrix} = \begin{bmatrix} P_x & 0 \\ 0 & P_y \end{bmatrix}\mathbf{R}_\psi\begin{bmatrix} x_d - x \\ y_d - y \end{bmatrix} + \begin{bmatrix} D_x & 0 \\ 0 & D_y \end{bmatrix}\mathbf{R}_\psi\begin{bmatrix} -\dot{x} \\ -\dot{y} \end{bmatrix} \tag{33}$$

with appropriate gains P_x, P_y, D_x, and D_y. $\mathbf{R}_\psi$ is the rotational matrix for the yaw angle, ψ and is defined as:

$$\mathbf{R}_\psi = \begin{bmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{bmatrix}. \tag{34}$$

The motor rotational speed values, (τ_θ, τ_ϕ) for (θ_d, ϕ_d) are calculated as follows:

$$\begin{bmatrix} \tau_\theta \\ \tau_\phi \end{bmatrix} = \begin{bmatrix} P_\theta & 0 \\ 0 & P_\phi \end{bmatrix}\begin{bmatrix} \theta_d - \theta \\ \phi_d - \phi \end{bmatrix} + \begin{bmatrix} D_\theta & 0 \\ 0 & D_\phi \end{bmatrix}\begin{bmatrix} -q \\ -p \end{bmatrix} \tag{35}$$

with control gains P_θ, P_ϕ, D_θ, and D_ϕ. The motor rotational speed values obtained in the previous procedure are assigned to the corresponding motors as follows:

$$u_1 = -\tau_\theta + \tau_\psi + u_d, \tag{36}$$

$$u_2 = -\tau_\phi - \tau_\psi + u_d, \tag{37}$$

$$u_3 = \tau_\theta + \tau_\psi + u_d, \tag{38}$$

$$u_4 = \tau_\phi - \tau_\psi + u_d. \tag{39}$$

The total thrust, $\mathbf{t}_B$ is calculated using the motor rotation values, (u_1, u_2, u_3, u_4) as follows:

$$\mathbf{t}_B = \begin{bmatrix} 0 \\ 0 \\ k(u_1^2 + u_2^2 + u_3^2 + u_4^2) \end{bmatrix}, \tag{40}$$

where k is the thrust coefficient. The torque command, τ_B acting on the quadrotor is calculated as follows:

$$\tau_B = \begin{bmatrix} Lk(u_4^2 - u_2^2) \\ Lk(u_3^2 - u_1^2) \\ b(u_1^2 - u_2^2 + u_3^2 - u_4^2) \end{bmatrix}, \tag{41}$$

where b is the drag coefficient acting on rotor blades, and the moment arm length L is shown in Fig. 7.

The parameter values for the quadrotor dynamics and control in this work are as follows: $m = 0.5$ (kg), $L = 0.3$ (m), $k = 2 \times 10^{-6}$, $b = 2 \times 10^{-6}$, $\mathbf{I} = \mathtt{diag}(0.08, 0.08, 0.16)$ (kg·m^2), $P_z = 300$, $D_z = 0$, $P_\psi = 200$, $D_\psi = 200$, $P_x = 0.1$, $P_y = 0.1$, $D_x = 1$, $D_y = 1$, $P_\theta = 600$, $P_\phi = 600$, $D_\theta = 0.2$, and $D_\phi = 0.2$.

[1]This may not be identical to the inertial frame in Section 2, but appropriate coordinate transformation can easily be applied.

[2]To ensure this norm condition all the time even in the presence of numerical errors, a normalization process is often applied in the numerical integration of dynamics.

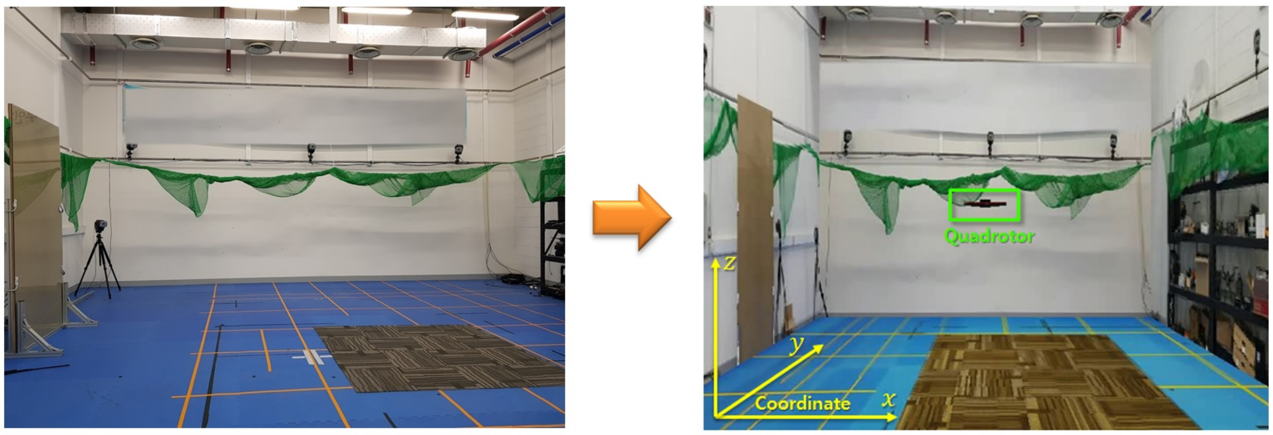

Fig. 8. Virtual simulator: real image of environment (left), simulated environment (right).

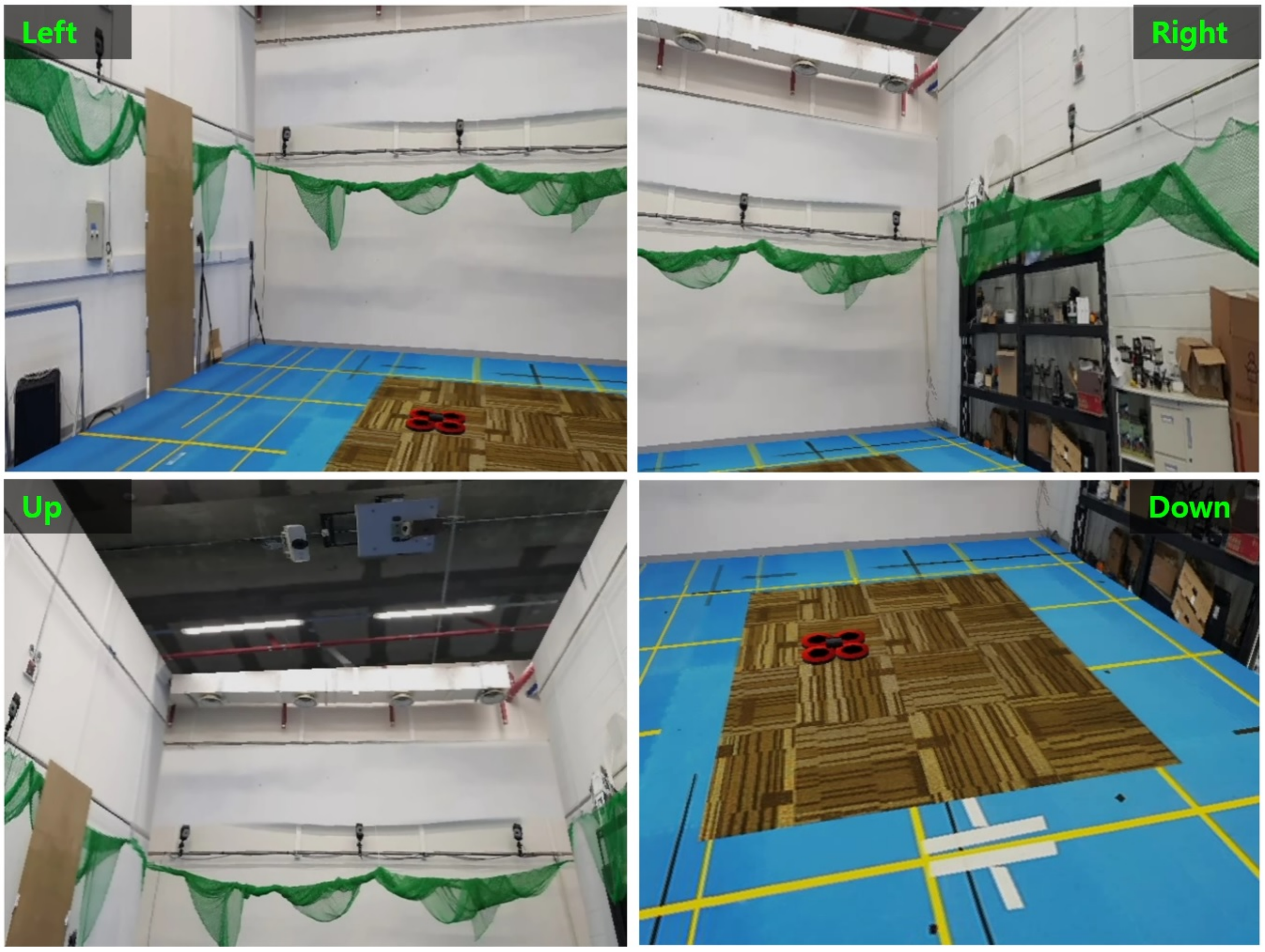

Fig. 9. Look-arounds of virtual environment.

4.2. Three-dimensional model

Three-dimensional indoor quadrotor simulator is developed to train the proposed CNN and to verify the performance of the multi-target tracking algorithm. The simulator is written in the Python language and uses OpenGL graphical tools [22] to implement three-dimensional virtual space and quadrotors. An exemplary simulation view is shown in Fig. 8. Real images are used to improve the reality of the environment. Fig. 9 illustrates some look-around views of the virtual test room reconstructed from real images. An image of the virtual environment and the target vehicles in it is regarded as one taken by some virtual camera; this is used as a training and/or test data for the CNN.

5. SIMULATION CASE STUDIES

To verify the performance of the proposed algorithm, tracking scenarios are created in the virtual simulation environment. The root mean square error (RMSE) is calculated as the metric of tracking performance:

$$RMSE_x = \sqrt{\frac{1}{N}\sum_{i=1}^{N}((x-\hat{x})^2)}, \tag{42}$$

where x is the true position, $\hat{x}$ is the estimated position, and N is the total number of data.

Tracking of three quadrotors is considered in the virtual simulation environment. The quadrotors move to follow the waypoints listed in Table 1, and random input disturbance is added to the motor control input of the quadrotor. Note that quadrotors pass over themselves when following the waypoints; this is intended to test the performance of the proposed algorithm in case of occlusion.

The true position, predicted position, and estimated position of each quadrotor are shown in Figs. 10 to 12, respectively. It can be seen that the trajectories of the three quadrotors are properly detected and tracked. Comparison of the errors in the CNN prediction and the CKF estimate

Table 1. Waypoints of the multiple quadrotors.

Target 1	Start	Waypoint 1	Waypoint 2	Waypoint 3	Waypoint 4	End
	(0,0,0) m	(0,0,1.8) m	(1,0,1.8) m	(-1,0,1.8) m	(0,0,1.8) m	(0,0,0) m
Target 2	Start	Waypoint 1	Waypoint 2	Waypoint 3	Waypoint 4	End
	(1,1,0) m	(1,1,1.8) m	(-1,1,1.8) m	(0,1,1.8) m	(1,1,1.8) m	(1,1,0) m
Target 3	Start	Waypoint 1	Waypoint 2	Waypoint 3	Waypoint 4	End
	(-1,0.5,0) m	(-1,0.5,1.8) m	(0,0.5,1.8) m	(1,0.5,1.8) m	(-1,0.5,1.8) m	(-1,0.5,0) m

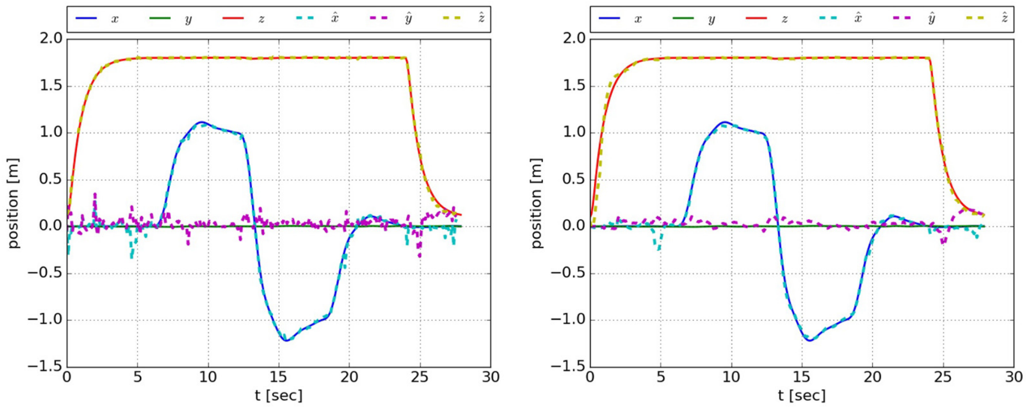

Fig. 10. Quadrotor 1's position history: CNN prediction (left), CKF estimate (right).

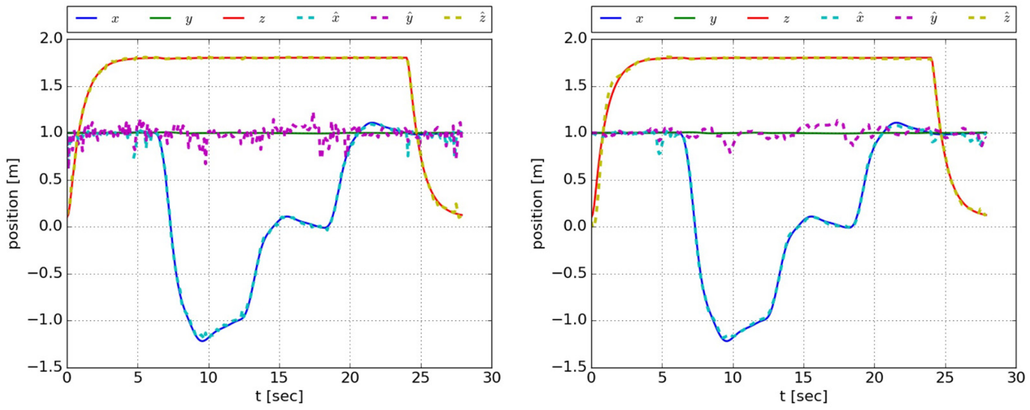

Fig. 11. Quadrotor 2's position history: CNN prediction (left), CKF estimate (right).

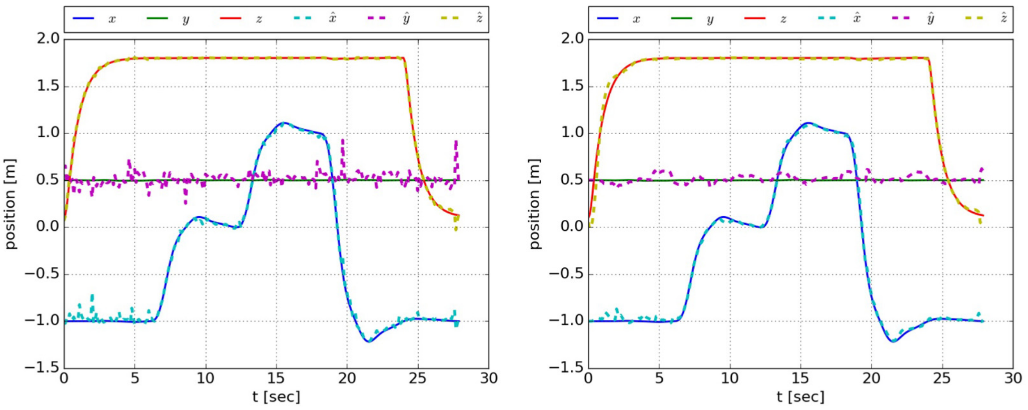

Fig. 12. Quadrotor 3's position history: CNN prediction (left), CKF estimate (right).

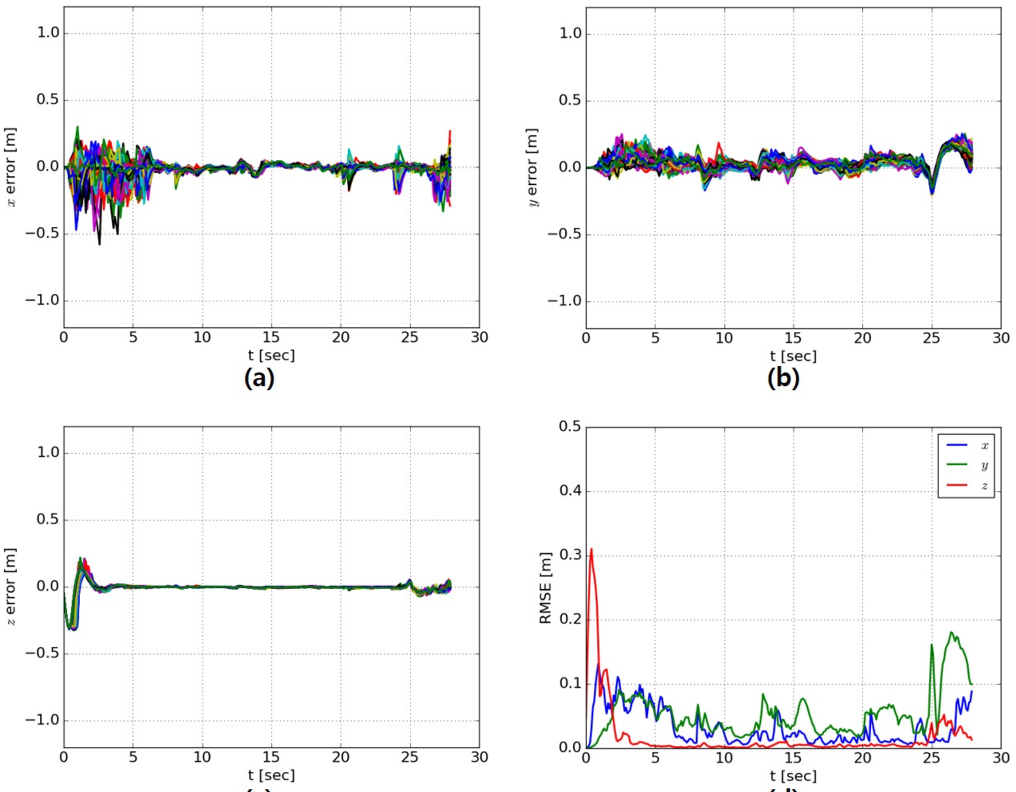

Fig. 13. Quadrotor 1's position error history: (a) x position error, (b) y position error, (c) z position error, (d) RMSE.

suggest that the error is further reduced by CKF that can take advantage of the knowledge of the dynamics of the targets.

Monte-Carlo results on the tracking errors with 100 scenarios are illustrated in Figs. 13 to 15. It can be seen that the position error is around 0.5m while the error in the altitude is noticeably smaller than the errors in the other directions. Note also that the error of the y-coordinate is less than 0.5m all the time, indicating that the targets are well separated without confusion.

Several snapshots from the whole scenario are depicted in Fig. 16. The green, red, and blue crosses indicate true position of the first, second, and third quadrotor, respectively; the yellow cross indicates a predicted position.

6. CONCLUSIONS

A convolutional neural network (CNN)-based multi-target tracking scheme has been proposed for the problem of multi-target tracking using a monocular vision system. The scheme overcomes the lack of depth measurement from the monocular vision by learning the CNN so

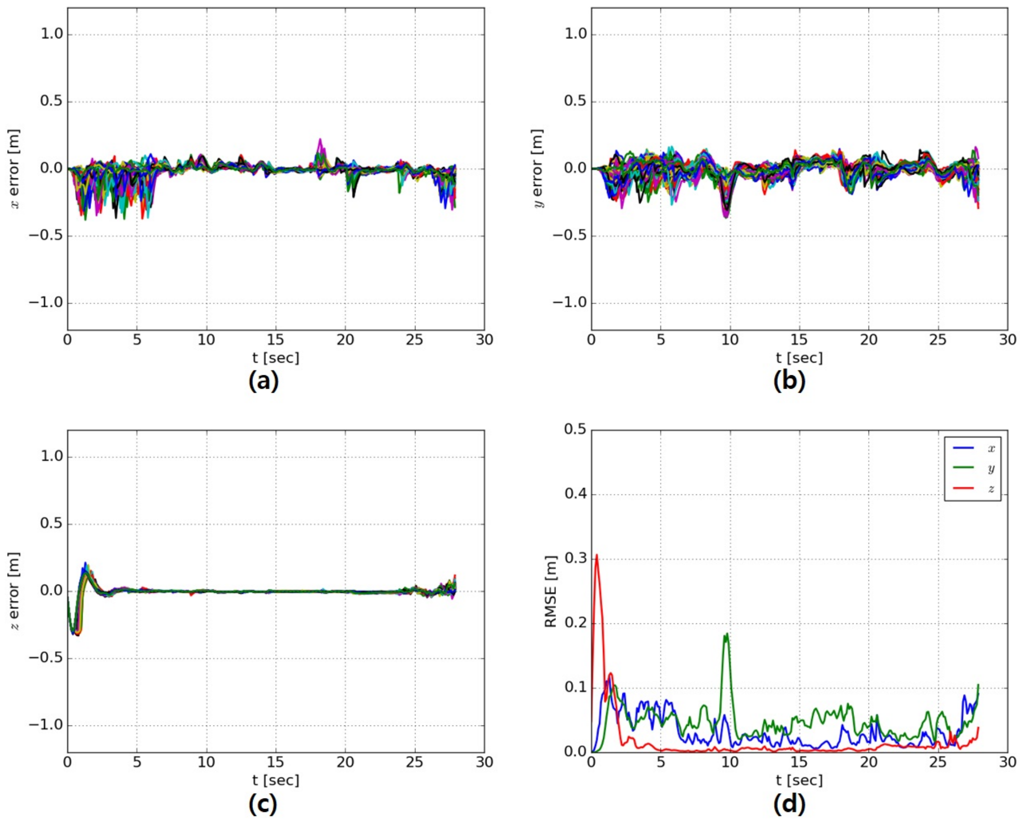

Fig. 14. Q

d) RMSE.

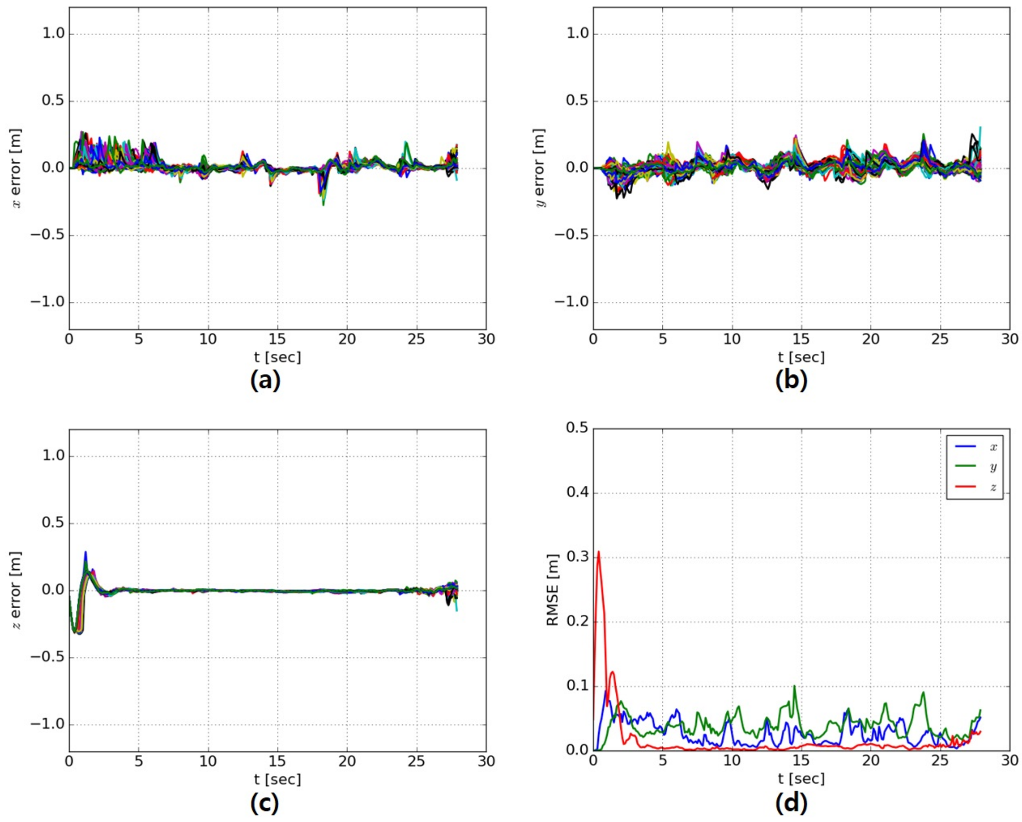

Fig. 15. Quadrotor 3's position error history: (a) *x* position error, (b) *y* position error, (c) *z* position error, (d) RMSE.

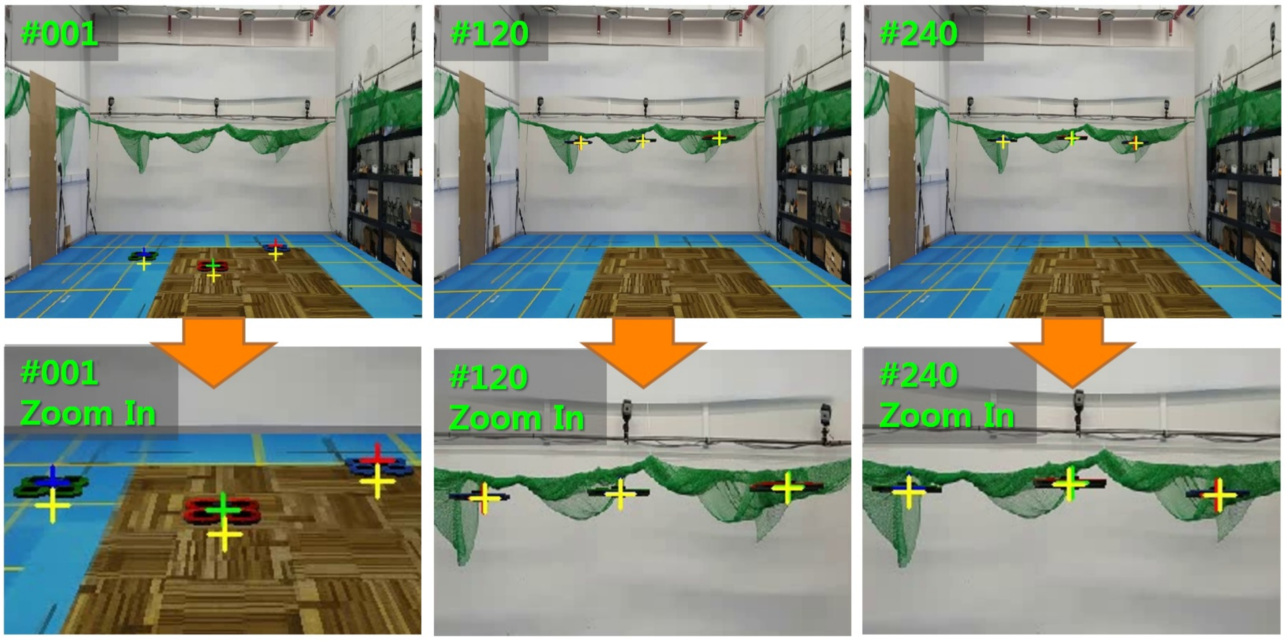

Fig. 16. Snapshots of target detection and tracking.

that it can produce the three-dimensional target information rather than the region of interest. A virtual simulation environment has been set up in order to generate the training data for the CNN and also to verify the proposed multi-target tracking scheme. Numerical case studies have shown that the position accuracy in the depth direction can be significantly improved using the proposed CNN architecture over a state-of-the-art CNN method.

REFERENCES

[1] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: a survey," *Acm Computing Surveys*, vol. 38, no. 4, 2006.

[2] K. Seo, J. Shin, W. Kim, and J. Lee, "Real-time object tracking and segmentation using adaptive color snake model," *International Journal of Control, Automation, and Systems*, vol. 4, no. 2, pp. 236-246, 2006.

[3] A. Chosh, B. N. Subudhi, and S. Chosh, "Object detection from videos captured by moving camera by fuzzy edge incorporated Markov random field and local histogram matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 8, pp. 1127-1135, 2012.

[4] T. Moranduzzo and F. Melgani, "A SIFT-SVM method for detecting cars in UAV images," *Proc. of Geoscience and Remote Sensing Symposium*, pp. 6868-6871, 2012.

[5] J. Yang, H. Ji, and Z. Fan, "Probability hypothesis density filter based on strong tracking MIE for multiple maneuvering target tracking," *International Journal of Control, Automation, and Systems*, vol. 11, no. 2, pp. 306-316, 2013.

[6] J. Yang, P. Li, L. Yang, and H. Ge, "An improved ET-GM-PHD filter for multiple closely-spaced extended target tracking," *International Journal of Control, Automation, and Systems*, vol. 15, no. 1, pp. 468-472, 2017.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffener, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

[8] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural network," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1725-1732, 2014.

[9] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," *Proc. of 6th IEEE International Conference on Computer Vision*, pp. 555-562, 1998.

[10] D. G. Lowe, "Object recognition from local scale-invariant features," *The Proceedings of the 7th IEEE International Conference on Computer Vision*, pp. 1150-1157, 1999.

[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, vol. 1, pp. 886-893, 2005.

[12] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "DeCAF: a deep convolutional activation feature for generic visual recognition," *Proc. of International Conference on Machine Learning*, vol. 32, no. 1, pp. 647-655, 2014.

[13] H.-H. Kim, J.-K. Park, J.-H. Oh, and D.-J. Kang, "Multitast convolutional neural network system for license plate recognition," *International Journal of Control, Automation, and Systems*, vol. 15, no. 6, pp. 29422949, 2017.

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580-587, 2014.

[15] R. Girshick, "Fast R-CNN," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440-1448, 2015.

[16] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *Advances in Neural Information Proceeding Systems*, pp. 91-99, 2015.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779-788, 2016.

[18] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *arXiv preprint arXiv:1612.08242*, 2016.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single Shot Multibox Detector," *Proc. of European Conference on Computer Vision*, pp. 21-37, 2016.

[20] I. Arasaratnam and S. Haykin, "Cubature Kalman filters," *IEEE Transaction on Automatic Control*, vo. 54, no. 6, pp. 1254-1269, June 2009.

[21] S. J. Julier and J. K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," *Proc. of Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, 1997.

[22] D. Shreiner, G. Sellers, J. Kessenich, and B. Licea-Kane, *OpenGL Programming Guide: The Official Guide to Learning OpenGL, version 4.3*, Addison-Wesley, 2013.

[23] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303-338, 2010.

[24] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, and C. L. Zitnick, "Microsoft COCO: common objects in context," *Proc. of European Conference on Computer Vision*, pp. 740-755, 2014.

[25] S. Bouabdallah, A. Noth, and R. Siegwart, "PID vs LQ control techniques applied to an indoor micro quadrotor," *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2451-2456, 2004.

[26] L. R. G. Carrillo, A. E. D. López, R. Lozano, and C. Pégard, "Modeling the quad-rotor mini-rotorcraft, *Quad Rotorcraft Control*, Advances in Industrial Control, Springer, London, pp. 23-34, 2013.

[27] F. Rinaldi, S. Chiesa, and F. Quagliotti, "Linear quadratic control for quadrotors UAVs dynamics and formation flight," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1-4, pp. 203-220, April 2013.

[28] D. Kingma and J. Ba, "Adam: a method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] M. D. Jeiler, "ADADELTA: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[30] T. Tieleman and G. Hinton, "Lecture 6.5-RMSprop: divide the gradient by a running average of its recent magnitude," *Coursera: Neural Networks for Machine Learning*, pp. 3919-3924, 2012.

[31] L. Y. Pao and R. M. Powers, "A comparison of several different approaches for target tracking with clutter," *Proc. of the American Control Conference*, vol. 5, 2003.

[32] B. L. Stevens, F. L. Lewis, and E. N. Johnson, *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*, John Wiley and Sons.

Sang-Hyeon Kim is a senior researcher in Samsung Electronics. He received the Ph.D. decree in Aerospace Engineering at KAIST (Korea Advanced Institute of Science and Technology) in 2018. Prior to this, he received the B.S. degree in Aerospace Engineering from the Inha University, Incheon, Korea, in 2011 and the M.S. degree in Aerospace Engineering from KAIST, Daejeon, Korea in 2013. His research interests include vision-based estimation and control for autonomous systems and deep learning techniques.

Han-Lim Choi is an Associate Professor of Aerospace Engineering at KAIST (Korea Advanced Institute of Science and Technology). He received the B.S. and M.S. degrees in aerospace engineering from KAIST, Daejeon, Korea, in 2000 and 2002, respectively, and the Ph.D. degree in aeronautics and astronautics from MIT (Massachusetts Institute of Technology), Cambridge, in 2009. His research interests include decision making for multi-agent systems and machine learning methods for dynamic systems.