Received November 13, 2020, accepted December 17, 2020, date of publication December 23, 2020, date of current version January 6, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3046730

Tracking In-Cabin Astronauts Using Deep Learning and Head Motion Clues

RUI ZHANG[1], YULIN ZHANG[2], AND XUEYANG ZHANG[1]
[1]School of Space Command, Space Engineering University, Beijing 101416, China
[2]School of Aerospace Engineering, Tsinghua University, Beijing 100084, China

Corresponding author: Rui Zhang (zhangrui07820@126.com)

ABSTRACT A person-following robot is under development for astronaut assistance on the Chinese Space Station. Real-time astronaut detection and tracking are the most important prerequisites for in-cabin flying assistant robots so that they can follow a specific astronaut and offer him/her assistance. In the limited space in the space station cabin, astronauts stand close to each other when working collaboratively; thus, large regions of their bodies tend to overlap in the image. In addition, because astronauts wear the same clothes most of the time, it is difficult to distinguish an individual astronaut using human body features. In this paper, we distinguish the astronauts by tracking their heads in the image. A deep learning model trained using big data is proposed for effective head detection. In addition, a motion model based on spatial clues is combined with the head detection results to track astronauts in the scene. A complete pipeline of the algorithm has been implemented and run efficiently on the Tegra X2 embedded AI microprocessor. A set of experiments were carried out and successfully validated the effectiveness of the proposed tracking algorithm. This algorithm is a step toward the implementation of robot assistants, especially in resource-limited environments.

INDEX TERMS Person-following robot, astronaut assistance, space station, deep learning, Tegra X2.

I. INTRODUCTION

Operating on near-Earth orbit for more than ten years, the Chine Space Station will support the development and testing of a number of advanced and core technologies in the field of space science. The implementation of complex scientific experiments and the long-term care of the space station will mainly depend on astronauts. However, astronaut crew time on the Space Station is a precious commodity. Improvements in astronaut work efficiency are particularly important for space missions. To assist the astronauts and thus improving their work efficiency, several in-cabin assistant robots have been proposed for or even sent to the International Space Station (ISS). Typical examples are PSA [1] and Astrobee [2], made by NASA's Ames Research Center, SPHERES [3] and smart SPHERES [4], made by MIT, Int-Ball [5], made by JAXA, CIMON, made by Airbus and IBM, and AAR [6], [7], made by the Chinese Academy of Sciences. Among them, Int-Ball and Cimon have been sent to the ISS for verification. Int-Ball was maneuvered by controllers and researchers on the ground, taking over photography duties from astronauts. CIMON can freely move and rotate using internal fans, interacting face to face with astronauts via voice.

The associate editor coordinating the review of this manuscript and approving it for publication was Hiram Ponce.

The Chinese Space Station will be launched around 2020. It will serve as a microgravity research laboratory and support the testing of spacecraft systems that are required for China's manned missions to the Moon and Mars. An intelligent astronaut assistant robot called the Intelligent Formation Personal Satellite (IFPS) [8] was proposed for assisting astronauts on the Chinese Space Station. In contrast to the robots mentioned above, the IFPS will focus more on astronaut following and intelligent human-robot interaction. It can not only fly independently in the cabin to guarantee safe, but also can follow the astronaut it is assigned to serve to assist with daily tasks.

Astronaut visual tracking is an important prerequisite to enable the IFPS to follow an astronaut and provide immediate assistance. The estimated number of crew members on the Chinese Space Station will be three, and our goal is to track the astronaut that is currently being assisted by the IFPS robot. In our previous work [9], [10], an astronaut visual tracking algorithm that is based on human body detection and motion prediction was proposed and implemented. In this algorithm, if the tracked astronaut is lost for a period of time because of occlusion, the tracking process will fail. This algorithm is more suitable for tracking an astronaut who is walking in the cabin with arbitrary postures or performing single-person missions. When working collaboratively in the limited space of the Chinese Space Station cabin, astronauts will often stand close to each other, which means that much of their bodies will overlap in the robot's field of view. In addition, the astronauts almost always wear the same clothes. It is difficult to distinguish an individual astronaut through human body features in such a scenario. However, their heads appear in the field of view most of the time without overlapping. The head occupies a smaller region in the image, making it easier to locate accurate than the body. Through spatial clues and motion prediction, heads can be accurately tracked even when body occlusion occurs. Thus, in this paper, for the multi-person collaborative work scenario on the Chinese Space Station, we focus on tracking the robot-assisted astronaut through head detection and tracking. A lightweight deep convolutional neural network (DCNN) was designed and trained to detect head. To use the head detection results, a motion prediction model based on spatial clues is proposed to track the robot-assisted astronaut. The complete pipeline of the algorithm was implemented and run in real time on an embedded AI microprocessor called the Tegra X2. A set of experiments were carried out and successfully validated the effectiveness of the proposed algorithm.

The reminder of this paper is organized as follows. Section II reviews the related work. Section III introduces the main pipeline of our proposed astronaut-tracking algorithm. The DCNN for head detection is presented in Section IV. Section V introduces the motion prediction based head-tracking algorithm. The experiments and results are discussed in Section VI. Lastly, we present the conclusion in Section VII.

II. RELATED WORK

A. DEEP-LEARNING-BASED OBJECT DETECTION AND TRACKING

In recent years, DCNNs have been found to have significant advantages for several kinds of object detection tasks [11], [12]. Deep-learning-based object detection networks mainly consist of two approaches, which are respectively based on the region proposal method and regression method. Region proposal method-based networks consist of a two-stage process. The first stage generates rough candidate regions, and then the second stage performs object classification prediction and fine regression to determine object locations. Representative networks of this type are R-CNN [13], Fast R-CNN [14], Faster R-CNN[15], R-FCN [16], FPN [17] and Cascade R-CNN [18]. Networks based on regression integrate the steps of candidate region generation, object classification, and location regression into one network. The networks improve inference speed while ensuring accuracy. Regression-method-based networks include the YOLO series [19]–[22] and SSD series [23]–[25]. Among these object-detection networks, the YOLO series and the original SSD networks are currently the state-of-the-art with respect to the trade-off between speed and accuracy. In addition, deep-learning-based methods have also attracted considerable interest in the visual tracking community as robust visual trackers. According to their architectures, state-of-the-art deep-learning-based visual tracking methods are categorized as convolutional neural networks (CNNs) [26]–[28], Siamese neural networks (SNNs) [29], [30], recurrent neural networks (RNNs) [31], [32], generative adversarial networks (GANs) [33], [34]. Although these state-of-the-art methods have achieved significant progress, they are still not reliable for real-world applications mainly because they lack the intelligence for scene understanding [35]. On the one hand, the computational demand of such a network is huge, and this computational complexity makes it impossible for real-time embedded applications. On the other hand, given the ill-posed nature of the problem, a customized dataset is necessary for each specific task. As far as we know, there is no large-scale benchmark dataset for astronaut tracking on space stations. Thus, to build a solution for the astronaut tracking task on the Chinese Space Station, we decided to use a combination of a DCNN-based head detector and head motion prediction to track the heads of the astronauts [36]. Although we also do not have large-scale dataset for astronaut head detection, making use of transfer learning and the sharing of a pre-trained model, our proposed deep-learning-based astronaut head detection model can achieve good results. Astronaut head tracking is achieved using head detection results and head motion prediction. This algorithm is more likely to be deployable in real time in the computationally limited embedded processors of the robot.

B. LIGHTWEIGHT DCNN DESIGN AND DEPLOYMENT

Deep-learning-based models have made important breakthroughs in accuracy and robustness. However, their vast computational demands and high number of parameters make it difficult to deploy them in real time on computationally limited embedded platforms. Not satisfied by simply obtaining smaller networks by shrinking, factorizing, or compressing pre-trained networks, researchers have dedicated themselves to building very small, low latency models that can be easily matched to the design requirements of mobile and embedded vision applications. Several efficient networks, such as SqueezeNet [37], ShuffleNet [38], MobileNet [39], and MobileNetV2 [40], have been proposed. Moreover, increasingly more model compression methods [41], [42] and inference acceleration software [43], [44] or engines [45] have been proposed for deploying deep learning models on mobile devices. Given their improvements in lightweight design and inference acceleration, the regression-method-based networks could be a better choice for real-time deployment in mobile and embedded devices [46].

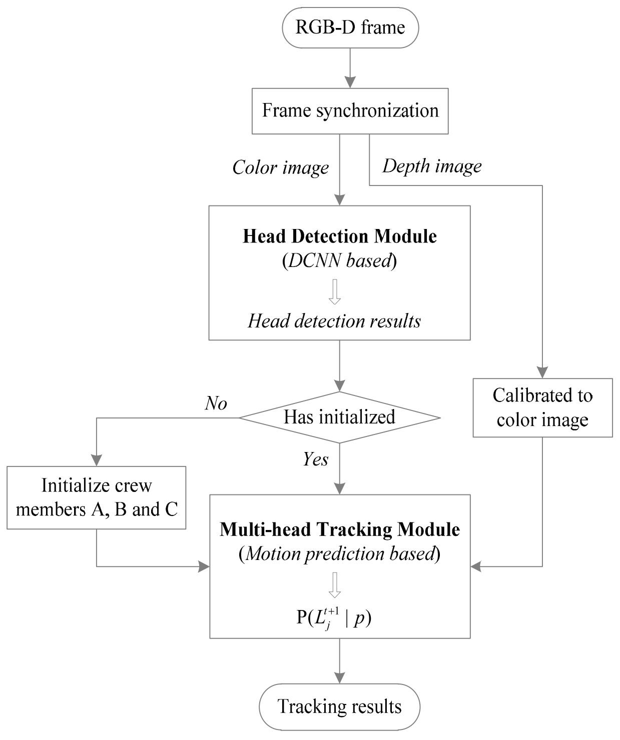


FIGURE 1. Astronaut tracking in the collaborative work scenario.

III. FORMULATION OF ASTRONAUT TRACKING IN COLLABORATIVE WORK SCENARIO

In a collaborative work scenario, the in-cabin astronauts stand close to each other. Large regions of their bodies overlap in the image and they wear the same clothes most of the time. We track the astronauts through head detection and tracking in this environment. A tracking-by-detection method is proposed, and **FIGURE 1** shows its work flow.

The astronaut head detection and tracking uses only RGB-D images. The images are acquired with an Orbbec Astra-Mini RGB-D camera. The head detection module first detects all heads in the color image. After initializing the detection results with the crew members' identities, the multi-head tracking module begins tracking the astronauts.

Multi-head tracking is achieved by a probability model, expressed as $P(L_j^{t+1}|p)$, which represents the probability that member p moves to L_j^{t+1}. Here, subscript j indexes the m head detection results, where $j = 1, 2, \ldots, m$, and p denotes the identity of the crew member, which is A, B, or C. Further, L_j^{t+1} denotes the spatial position of the jth detection result at $t+1$, which is obtained from both the detection result and the calibrated RGB-D image.

In the following two sections, the DCNN based head detection module and the motion prediction based head tracking module are described.

IV. DCNN BASED HEAD DETECTION MODULE

A. HEAD DETECTION DCNN DESIGN

A lightweight DCNN, which is an improved version of FaceBoxes [47] is proposed for astronaut head detection. The architecture of the network is shown in **FIGURE 2**. The network is a regression-method-based object detection model and is a lightweight network. As shown in **FIGURE 2**, the network consists of a data layer (white part), feature extraction layers (orange part), and multi-scale prediction layers (green, blue and yellow parts). We mainly made adaptive improvements to the multi-scale prediction layers according to the characteristics of the astronaut head detection problem:

1) Adaptive design of the prediction layer parameters. The input image is limited to 640×480 pixels. The default bounding boxes are designed on three different layers, as shown in IV-B. A larger layer size, leads to a higher density of the designed default bounding boxes. These bounding boxes enable the prediction of human heads from 20 pixels to 320 pixels in size in the input image. This ensures that the robot can detect astronaut heads at different distances in the space station cabin. The inner space of this cabin is limited, and the robot and astronauts work cooperatively at a distance of about 1 to 2m most of the time, as shown in **FIGURE 3**. The astronauts' heads in the image range from about 20×20 to 80×80 pixels in size. The corresponding default bounding boxes of 32×32, 64×64 and 128×128 pixels are all designed on the 32×32 Inception3 layer. This ensures that the heads within the size range can be predicted with sufficient density in the image. Therefore, the robot can effectively detect human heads in most common situations.

2) In the proposed network, a densification strategy is not used on the default bounding boxes. The FaceBoxes network pays more attention to the recall rate, and thus a densification strategy is used on its default bounding boxes. Here, for the astronaut head detection and tracking task, we prefer to obtain accurate envelopes for the human heads. Detection results with overlapping envelopes are not suitable for accurately measuring human head positions and this lack of accuracy adversely affects the head tracking task at a later stage. Thus, the densification strategy is not used on the default bounding boxes in proposed head detector DCNN. Hence, the parameters of the location (green part) and confidence (blue part) prediction layers were modified accordingly.

B. DCNN TRAINING AND VALIDATION

The astronaut head detection DCNN training consists of two steps. We first pre-trained the network using the CrowdHuman [48] dataset, which provides large-scale training and test samples for human head detection. When the pre-training finished, the network gained a well pre-trained model. Then, the network was further fine-trained using our newly built astronaut head detection dataset. We collected 4,684 pictures in total, of which 3,513 pictures were used as training dataset while the other 1,171 pictures were used as testing dataset. In the dataset, more than half of the pictures contain three persons. The training dataset contains about 7,900 head images, whereas the test dataset contains about 2,600 head images. The pictures in the two datasets are evenly distributed. Both datasets take into account a variety of possible situations,

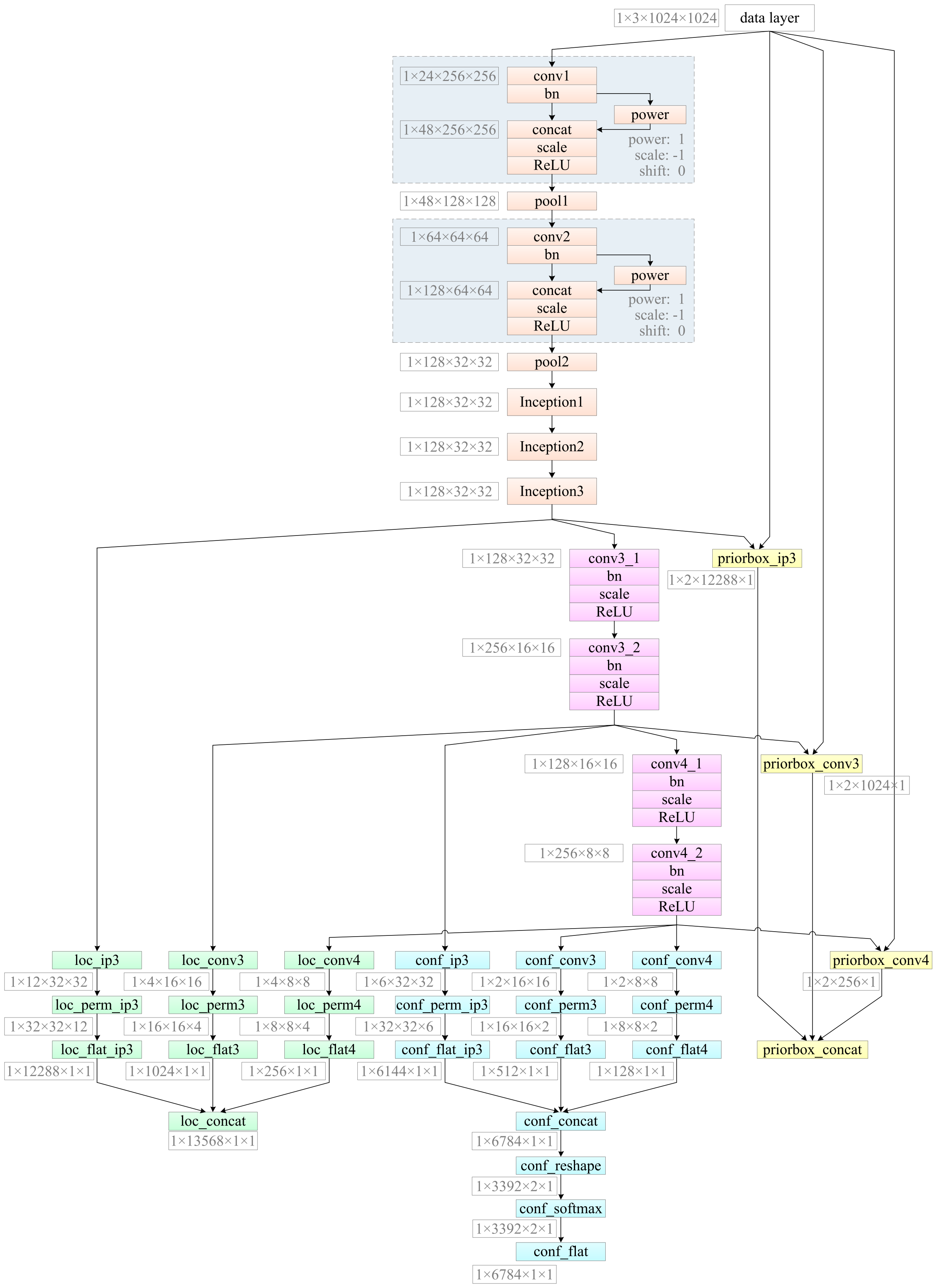


FIGURE 2. FaceBoxes network-based astronaut head detection DCNN.

TABLE 1. Parameters of the default boxes for head detection.

Parameters \ Layers	Inception3	conv3_2	conv4_2
Feature map size	32×32	16×16	8×8
$L^{k}_{\min}$	32, 64, 128	256	512
step	32	64	128
aspect ratio	1	1	1
clip	false		
offset	0.5		
variance	{0.1, 0.1, 0.2, 0.2}		
Number of default boxes	12288	1024	256
Receptive field in the input image	20×15, 40×30, 80×60	160×120	320×240

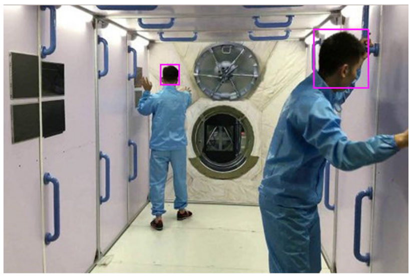


FIGURE 3. **Range of the most common sizes of astronaut heads in the input RGB-D image.**

FIGURE 4. **Example images in our dataset.**

such as a variable number of humans, different human postures, and occlusions between human bodies. **FIGURE 4** shows some example pictures in our astronaut head detection dataset.

Astronaut head detection is actually a localization and binary classification task. The training loss is defined as a weighted sum of the localization loss and the confidence loss. A two-class softmax loss was adopted for classification and the smooth L1 loss [14] was used for regression. Considering that the proportion of human heads in the image is relatively small, we set the number of negative samples (no person in image) in each training image to seven times the number of positive samples.

The stochastic gradient descent method was used for fine-tuning with an initial learning rate of 0.0005, a momentum of 0.9, and a weight decay of 0.0005. The fine-tuning procedure consisted of 100,000 iterations, and the learning rate decreased by 10 at 20,000, 40,000, 60,000, and 80,000 iterations. The training procedure was implemented on a high-performance parallel computing unit, the NVIDIA TITAN Xp, using the Caffe engine.

FIGURE 5(a) shows how the training loss changes during the fine-tuning procedure. The pre-trained model was used to initialize the network when fine-tuning began; thus, the initial value of the training loss was only 3.78. The training loss gradually decreased as the number of iterations increased, reaching approximately 0.7 after 100,000 training iterations. **FIGURE 5(b)** shows the average test precision curve during the fine-tuning procedure. At the beginning of fine-tuning procedure, the network already had a high average precision rate of 88.57%. As the number of iterations increased, the average precision rate gradually increased, reaching 90.89% by the end of the fine-tuning procedure.

FIGURE 6 shows the precision recall curves and average precision over different intersection over unions (IOUs) in the testing dataset. **FIGURE 6(a)** shows that the network has achieved high recall rates for different IoU thresholds. The precision rate decreases slightly when the recall rate exceeds 70%. Larger IoU thresholds decrease the precision. However, even when the IoU is 0.8 and the recall is 90%, the network precision is still more than 97.1%. **FIGURE 6(b)** shows that the network has achieved high average precision under different IoU thresholds. As the IoU threshold increases and exceeds 0.65 the average precision rate decreases slightly. However, even if the IoU threshold is as high as 0.8, the average precision rate is still higher than 90%. In addition, we tested the generalization ability of the network by detecting heads in images that were not a part of the training dataset. **FIGURE 7** shows some of the test results. The data above the

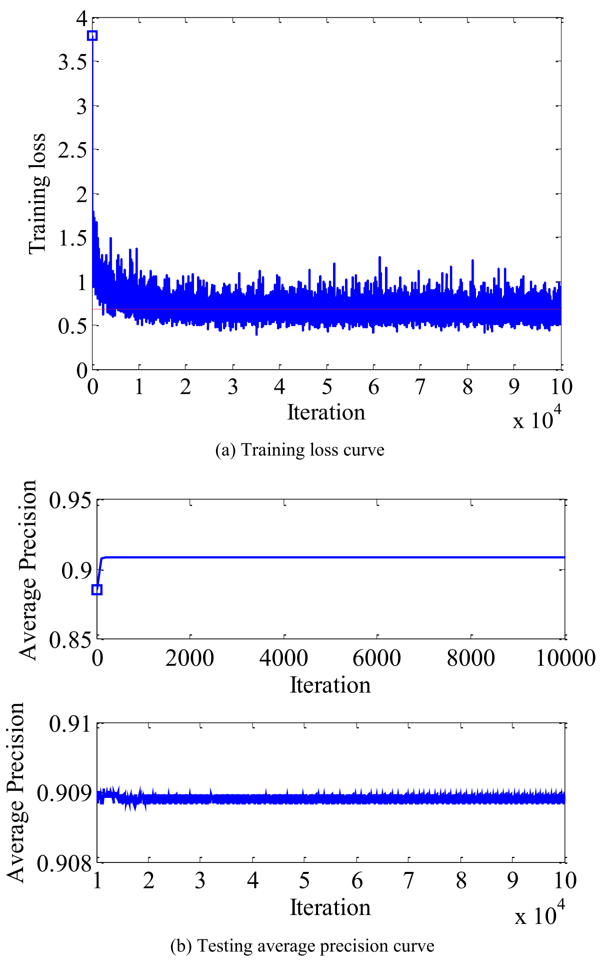


FIGURE 5. **Training loss and average test precision during the fine-tuning procedure.**

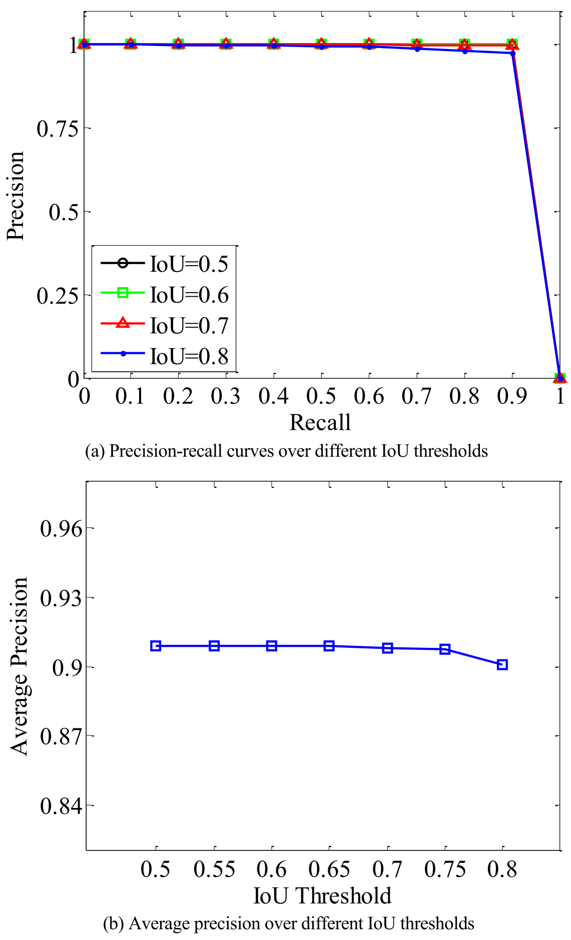


FIGURE 6. **Precision recall curves and average precision over different IoU thresholds.**

bounding box is the confidence of the detection result. The participants walked or squatted in the cabin. The network can detect human heads of various sizes and at different orientations in the image. These results demonstrate that the head detection network has good generalization ability, obtaining stable and accurate head detection results.

V. MOTION PREDICTION BASED MULTI-HEAD TRACKING MODULE

After the heads have been detected and their spatial positions identified, multiple astronauts can be tracked by predicting their head movement. This section describes a multi-target motion tracking algorithm to predict and track all three astronauts in the cabin.

A. HEAD POSITIONING BASED ON RGB-D IMAGE

Before tracking, we first obtain the spatial position of the human heads relative to the robot camera. In Section IV, we describe how the human heads in each frame of the color image are located using rectangles. If we know the position of the head in the image coordinate system and its depth information, we can use the camera model to calculate the spatial position of the head relative to the camera. Here, we treat the center point (u^0, v^0) of the bounding box as the head's position in the image coordinate system, as shown in **FIGURE 8(a)**. As for the depth information of each head, it can be obtained easily from the fused color image and depth image. If the depth value corresponding to the center point (u^0, v^0) in the color image is not zero, the depth value is the depth information of the head. Otherwise, the center point is used as the starting point, and multiple non-zero depth values are obtained by searching increasing distances in neighborhood around it. The average of these depth values is used as the head depth value. **FIGURE 9** shows the flow of this algorithm. In contrast to the conventional solution, which is to filter the depth image and then obtain the depth value of the center point, this method requires much less calculation, but still obtains high-precision depth values. This improves the speed of human head position measurement. The method is simple and easy to implement in hardware.

B. HEAD TRACKING ALGORITHM

The head tracking is realized by constructing a probability prediction model related to head movement. During a normal

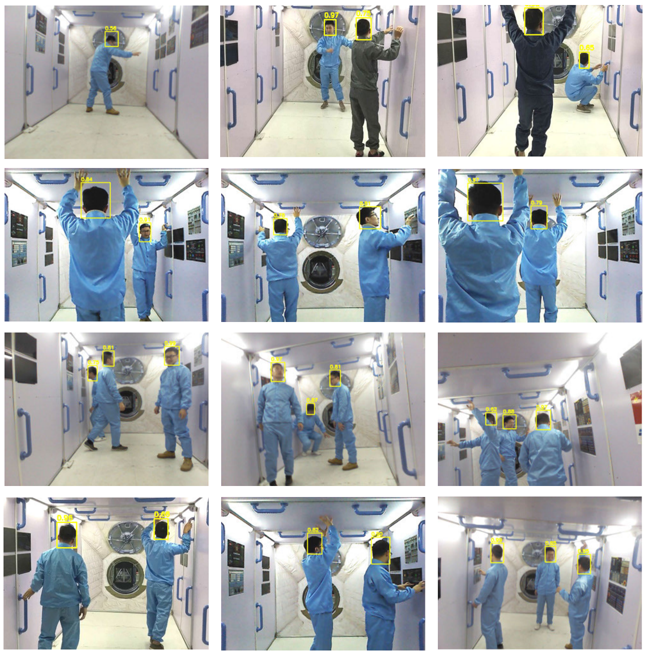
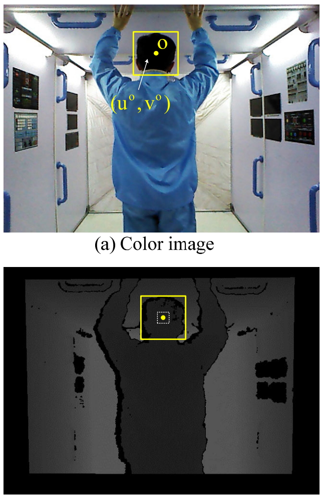


FIGURE 7. Detection results for images not included in the training and testing datasets.



(a) Color image

(b) Depth image

FIGURE 8. Head positioning method.

long-term flight, three crew members will work in the space station cabin. Here, we assume that the robot follows A for task assistance, so our final goal is to continuously track A by predicting the positions of all three persons. As shown in II, red, green, and blue circles indicate the head positions of crew members A, B, and C respectively. Their spatial positions at time t are respectively denoted as L_1^t, L_2^t, and L_3^t. The grey circles represent the spatial position of m head detection results at time $t+1$. The task of tracking is to select the correct detection result from the m head detection results at time $t+1$ and determine whether its identity is crew member A, B, or C.

For the m detection results at time $t+1$, the probability that the jth result is the identity of A, B or C is as:

$$\mathrm{P}(L_j^{t+1}|p) = \sum_i \left(\mathrm{P}(L_i^t|p) \cdot h_{ji}^{t+1} \right), \tag{1}$$

$$\sum_i \mathrm{P}(L_i^t|p) = 1, \tag{2}$$

where $p \in$ S and S $=$ {A,B,C}. As noted in Section III, subscript j indexes the m head detection results, where $j = 1, 2, \ldots, m$. Subscript i indexes the identified crew member A, B, or C at time t, where $i = 1, 2, 3$, respectively. The probability of astronaut p ($p \in$ {A,B,C}) appearing at location L_j^{t+1} at time $t+1$ depends only on the probability distribution of astronaut p in the last frame t and the motion assumption that is made. Here, $\mathrm{P}(L_i^t|p)$ is the probability distribution of astronaut p in the last frame t. Moreover, h_{ji} is the transition probability for the motion assumption. It is the probability that a person from location L_i^t at time t moves to location L_j^{t+1} at time $t+1$. We use the following simple Gaussian probability distribution for the transition probability:

$$h_{ji}^{t+1} = \frac{1}{Z} e^{-\mathrm{k} \cdot \|\Delta L_{ji}\|^2}, \tag{3}$$

$$\Delta L_{ji} = \left\| L_j^{t+1} - L_i^t \right\|, \tag{4}$$

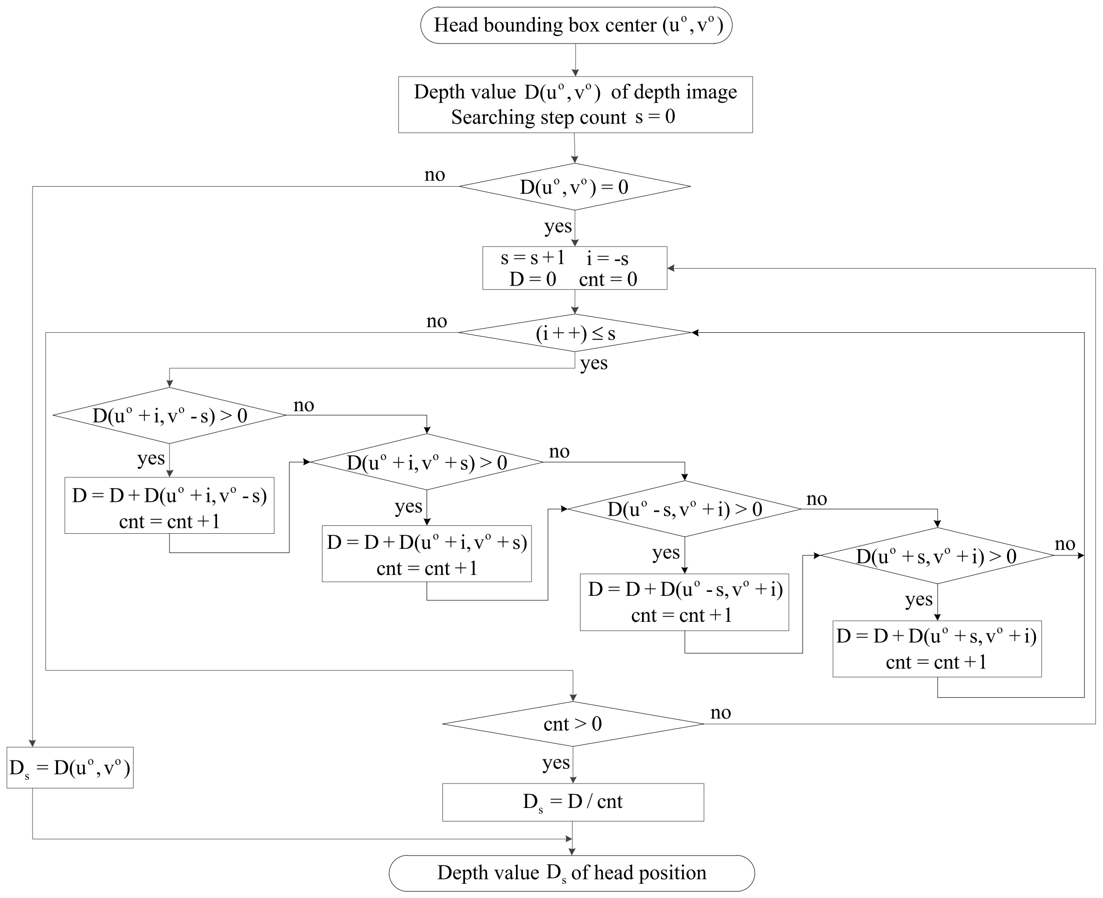
where ΔL_{ji} is the distance between location L_i^t at time t and location L_j^{t+1} at time $t+1$. Parameter Z is a normalization factor whereas k is constant parameter. Here, we set the values of Z and k to 1. After calculating the identity prediction probability of the all detection results at $t+1$, we determine the identity and update the probability as follows:

$$\mathrm{P}^{t+1}(L_j^{t+1}|p) = \begin{cases} 1 & \begin{array}{l} \mathrm{P}^{t+1}(L_j^{t+1}|p) > \mathrm{T_{conf}}\ \textit{and} \\ \mathrm{P}^{t+1}(L_j^{t+1}|p) > \mathrm{P}^{t+1}(L_j^{t+1}|\mathrm{C}_S p) \end{array} \\ 0 & \text{others,} \end{cases} \tag{5}$$

where $p \in$ S and S $=$ {A,B,C}. Here, $\mathrm{C}_S p$ represents all the people in set S except for p, $\mathrm{T_{conf}}$ is the probability threshold. A tradeoff must be considered when selecting its value. If the parameter value is too small, the tracking procedure may be easily affected by the resulting missed detections. If this value is too big, incorrect tracking may occur when multiple astronauts are positioned close together. Here, we set $\mathrm{T_{conf}}$ to be 0.65 after weighing the above two factors.

The aim of the proposed probabilistic prediction model for multi-target tracking is to simultaneously track three astronauts in the cabin. In practical applications, the number of human heads in the robot's field of view will change because of human body movements. The tracking algorithm should take into account which crew have entered and exited the camera's field of view, and promptly update the probability and position of each target in the probability prediction model. **FIGURE 11** shows the flow chart of the multi-target tracking module. The tracking module consists of the following main steps:

1) At the beginning of each tracking process, we determine the two parameters D^t and M^t. Here, D^t is the



Head bounding box center (u^o, v^o)
Depth value $D(u^o, v^o)$ of depth image
Searching step count $s = 0$
$D(u^o, v^o) = 0$
no
yes
$s = s+1 \quad i = -s$
$D = 0 \quad cnt = 0$
$(i++) \le s$
$D(u^o+i, v^o-s) > 0$
$D = D + D(u^o+i, v^o-s)$
cnt = cnt + 1
$D(u^o+i, v^o+s) > 0$
$D = D + D(u^o+i, v^o+s)$
$D(u^o-s, v^o+i) > 0$
$D = D + D(u^o-s, v^o+i)$
$D(u^o+s, v^o+i) > 0$
$D = D + D(u^o+s, v^o+i)$
cnt > 0
$D_s = D(u^o, v^o)$
$D_s = D / cnt$
Depth value D_s of head position

FIGURE 9. Algorithm for obtaining the depth value of the head position.

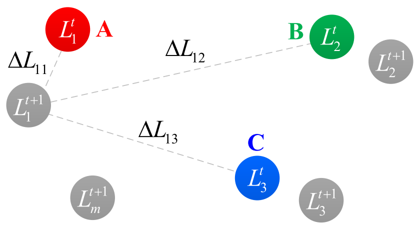


FIGURE 10. Multi-target head motion prediction model.

number of detected human heads in the current frame of the video, and M^t is the largest number of people who have appeared in the camera's field of view so far, where $M^t \le 3$. To avoid the influence of false detections, the six consecutive frames of video closest to the current moment are used to determine M^t.

2) The algorithm detects whether the position of each crew member has been initialized (**FIGURE 11**, yellow part). If $M^t = 1$, the robot just follows crew member A, and location of A is initialized with the position of the detected result. Crew members B and C are initialized with virtual positions (0, 5, 0) and (0, −5, 0), respectively. A virtual position is a distant position in front of or behind the robot. Similarly, if $M^t = 2$, crew members A and B will be initialized with the positions of the detected results whereas crew member C is given virtual position (0, −5, 0). If $M^t = 3$, all three crew members are initialized with the positions of their detected results.

3) Human entries are detected (**FIGURE 11**, blue part) according to different M^t values. If $M^t = 2$ and the number of astronauts is less than two (Bout = 1), the new entrant is given the identity of crew member B. His/her spatial position is set using the method described in Section V-A. If $M^t = 3$ and the number of astronauts is less than three (Bout = 1 or Cout = 1), each new entrant is given the identity of crew member B and crew member C in that order.

4) The probability is calculated and updated (**FIGURE 11**, purple part). The probability that member p moves to L_j^{t+1} is calculated and updated, as described in Eqs. (1) to (5).

5) Human exits are detected (**FIGURE 11**, grey part) according to different M^t values. As noted above, our main goal is to track the astronaut the robot is assisting (crew member A). Therefore, the algorithm stops if crew member A exits the robot's field of view. If crew

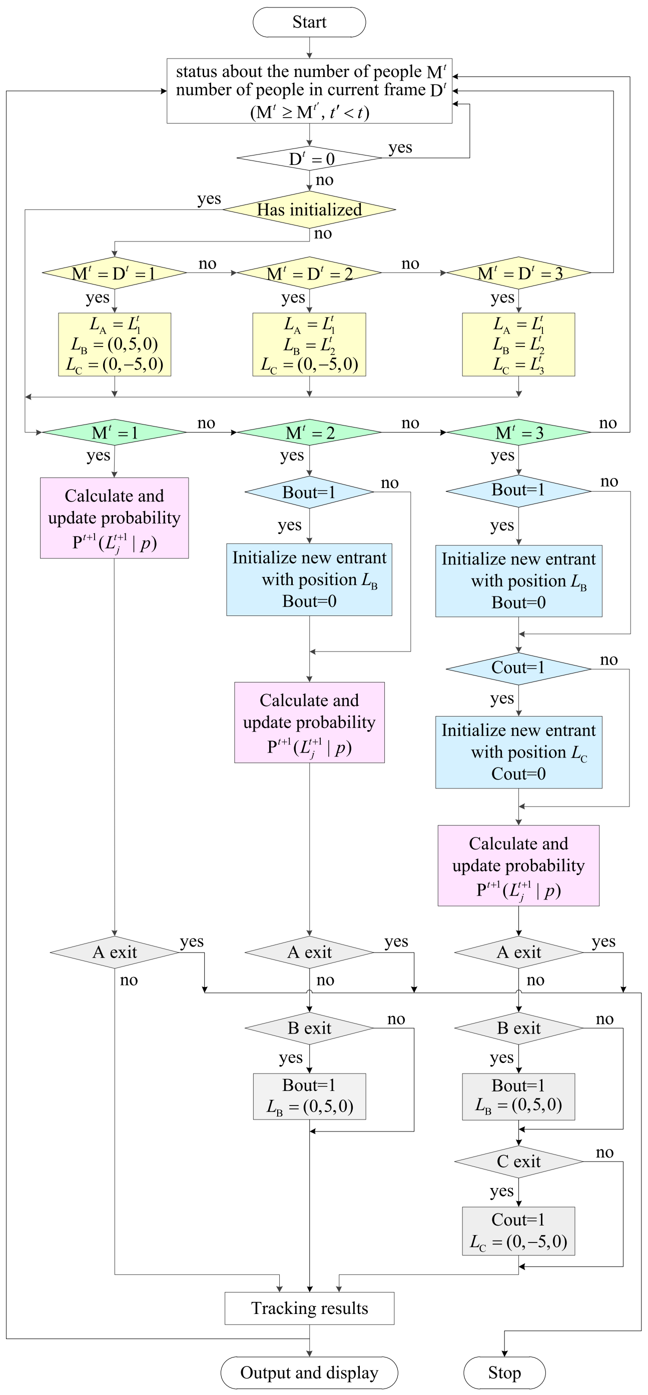

Start
status about the number of people M^{t}
number of people in current frame D^{t}
$(M^{t} \geq M^{t'}, t' < t)$
$D^{t} = 0$
Has initialized
$M^{t} = D^{t} = 1$
$M^{t} = D^{t} = 2$
$M^{t} = D^{t} = 3$
$L_A = L_1^t$, $L_B = (0,5,0)$, $L_C = (0,-5,0)$
$L_A = L_1^t$, $L_B = L_2^t$, $L_C = (0,-5,0)$
$L_A = L_1^t$, $L_B = L_2^t$, $L_C = L_3^t$
$M^{t} = 1$
$M^{t} = 2$
$M^{t} = 3$
Calculate and update probability $P^{t+1}(L_j^{t+1} \mid p)$
Bout=1
Initialize new entrant with position L_B Bout=0
Cout=1
Initialize new entrant with position L_C Cout=0
A exit
B exit
Bout=1 $L_B = (0,5,0)$
C exit
Cout=1 $L_C = (0,-5,0)$
Tracking results
Output and display
Stop
yes
no

FIGURE 11. Multi-target tracking module.

member B or C exits the camera's field of view (Bout = 1 or Cout = 1), their positions are set to the virtual positions (0, 5, 0) and (0, −5, 0), respectively.

6) The tracking results are output and displayed. The tracking results of the current frame are used as the initial conditions for the next tracking loop.

The algorithm determines when an astronaut enters or exits the field of view according to certain conditions. The following two conditions for identifying crew member exits must be met:

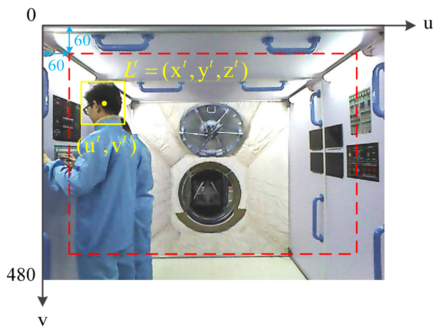


FIGURE 12. Determination of when an astronaut exits or enters the camera's field of view.

1) The position (u^{t}, v^{t}) of the crew member's head in the color image moves outside the area of the image that has valid depth information. The red dashed box in **FIGURE 12** indicates this area. Because no valid depth image information is available outside this area (as shown in **FIGURE 8**), the spatial position $L^{t} = (x^{t}, y^{t}, z^{t})$ of the crew member cannot be updated.

2) There is no obvious movement of the crew member over six consecutive frames of the video. That is, the distance moved by the crew member during this period does not exceed a certain threshold. Similarly, two conditions must be met before the algorithm determines that a crew member has entered the field of view: 1) The position (u^{t}, v^{t}) of the crew member's head in the color image is inside the valid depth information area. 2) The stable and valid spatial position $L^{t} = (x^{t}, y^{t}, z^{t})$ of the crew member can be measured.

VI. EXPERIMENTS AND RESULTS

In the experiments described in this section, the overall pipeline of astronaut tracking algorithm was implemented and transferred to the Tegra X2 embedded AI platform for online application. The experimental platform for evaluating the tracking algorithm was the same as that developed in [10].

A. ALGORITHM DEPLOYMENT ON AN EMBEDDED PLATFORM

We deployed the head tracking algorithm in the IFPS's embedded microprocessor, the Tegra X2. The codes of the algorithm were developed in C++ on Linux with Clion. CUDA 8.0 and TensorRT3 were required for high efficient parallel computing, and openNI2 and OpenCV3 were required for image acquisition and preprocessing.

We first transfer the head-detection DCNN on Tegra X2. The network inference is based on TensorRT engine [45] instead of Caffe. The inference procedure of the head-detection DCNN includes two steps: the engine build and network inference, as shown in Fig. 12 of [10]. The reshape layer, flatten layer, concat layer (axis = 2), and softmax layer (axis = 2) in the head detection DCNN cannot be directly supported by TensorRT. Hence, they must be implemented with the user-defined layer interface PluginFactory of TensorRT.[1]

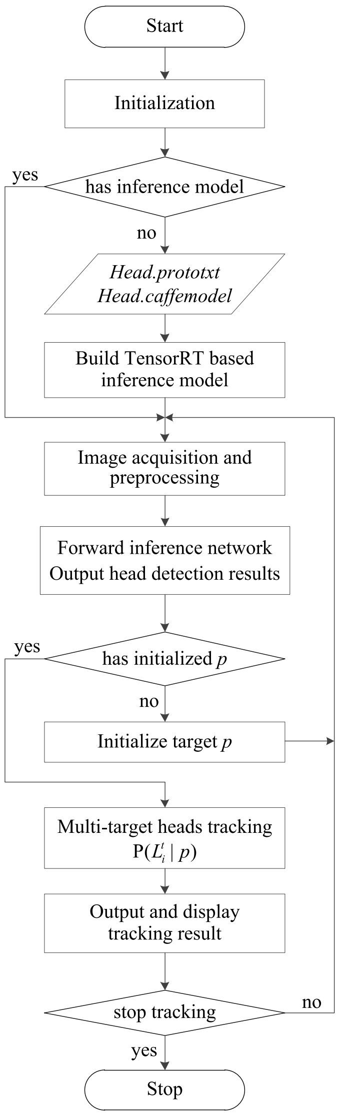

Start
Initialization
has inference model
yes
no
Head.prototxt
Head.caffemodel
Build TensorRT based inference model
Image acquisition and preprocessing
Forward inference network
Output head detection results
has initialized p
yes
no
Initialize target p
Multi-target heads tracking
P(L_i^t | p)
Output and display tracking result
stop tracking
no
yes
Stop

FIGURE 13. **Online head tracking algorithm.**

FIGURE 13 shows the overall flow of the online head-tracking algorithm deployed on the Tegra X2. After initialization, the algorithm checks whether the TensorRT inference engine of the head detection DCNN has been built. If it has, image acquisition and preprocessing are performed. Next, the forward inference of the head detection DCNN is performed to detect the heads in the color image. Subsequently, the assisted astronaut (crew member A) is specified and the multi-target tracking algorithm for tracking heads is called. Finally, tracking results are output and displayed.

[1]Codes of implementing custom layers using the user-defined layer interface PluginFactory of TensorRT are available at: https://github.com/richeal722/user_defined_TRT_layers_for_head_detection

B. ONLINE TRACKING TESTS AND RESULTS

This section describes the astronaut motion tracking experiments under single and multi-person conditions. Each experiment considers crew member exiting and entering situations as well as the robot's movements. Similar to [10], the platform was servo-controlled by vision. It can maintain a certain distance from the tracked participant (crew member A), and is controlled by a servo to keep the participant in the middle of the color image. The yellow rectangles are used to locate the head detection results. The red rectangle is used to display the location of the human head of crew member A. The green and blue rectangles are used to locate the human heads of crew member B or crew member C.

FIGURE 14 shows some screenshots of the single person tracking experiment result. The participant (crew member A) entered and walked around in the cabin. During this period, his head was continuously detected and tracked by the tracking algorithm. The tracked results were located using red rectangles. The experimental platform was servo controlled and moved to make the tracked head in the middle of the color image. When the participant (crew member A) accelerated and exited the robot's field of view, the tracking algorithm stopped.

FIGURE 15 shows some screenshots of the two-person tracking experiment results. At the beginning, the robot tracked the first participant it was assisting (crew member A). Then, another participant entered the robot's field of view. He was detected and his location was displayed using a yellow rectangle (top row, second graph). Subsequently, he was tracked and his location displayed using a green rectangle. In the first test, the new person walked behind crew member A. He then walked around crew member A and finally exited the robot's field of vision in front of A. In the second test, the new person walked in front of crew member A. He then walked around crew member A and finally exited the robot's field of vision from the behind of A. Throughout the experiment, both participants were accurately identified. Stable and accurate tracking of the person (crew member A) assisted by the robot was realized.

FIGURE 16 shows some screenshots of the three-person tracking experiment result. At the beginning, the robot successfully tracks the first participant, whom it is assisting (crew member A, indicated by the red rectangle). Then, the second participant entered and walked around crew member A (top row and first two images of the second row). The second participant is indicated using a green rectangle. During this period, the algorithm maintained stable and accurate tracking of both participants. Subsequently, the third participant entered the robot's field of view (second row, third image). He walked around crew member A. After a while, the second participant exited (third row, first two images). Immediately afterwards, the third participant exited and re-appeared after several seconds. He was detected and re-tracked. His position was indicated using a green rectangle instead of the blue rectangle used before (third row,

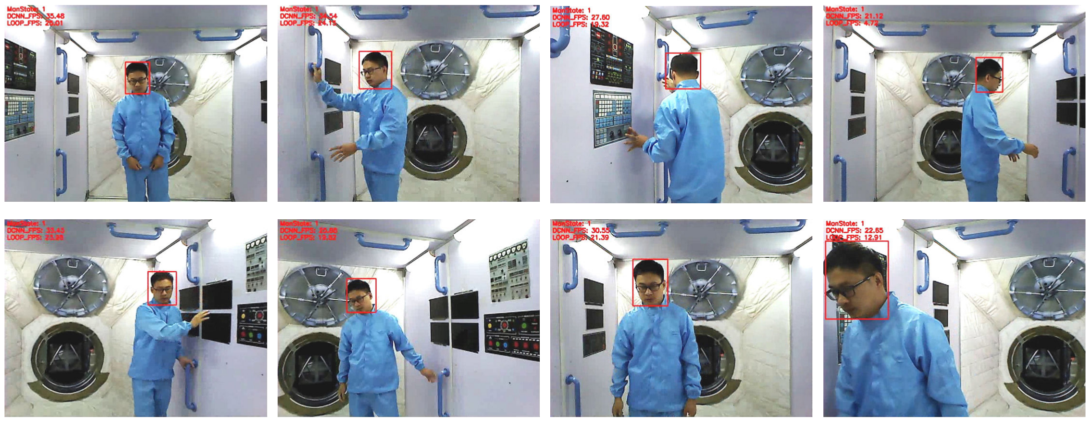


FIGURE 14. Qualitative experimental results of single-person tracking.

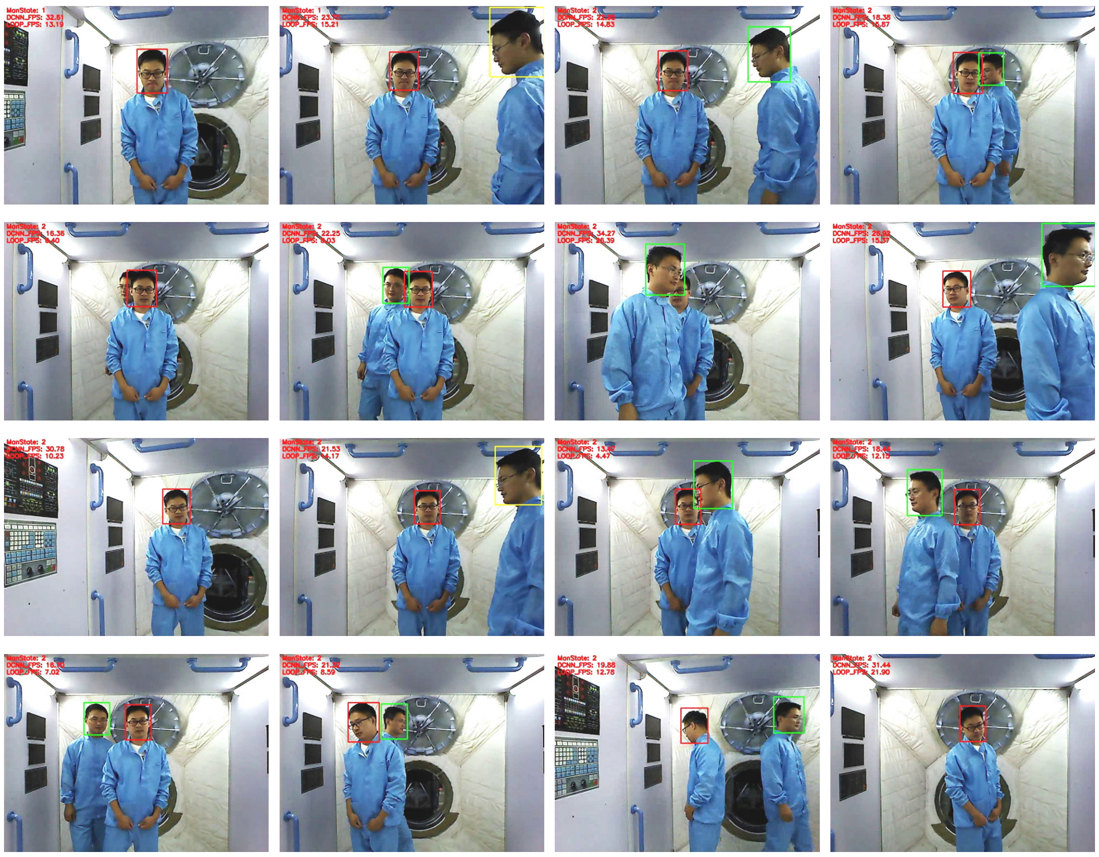


FIGURE 15. Qualitative experimental results of two-person tracking.

last image). Because our main purpose is to track the astronaut that robot is assisting (crew member A), we do not pay attention to the identity of the other two people (crew member B or C). The location of the first entering person (crew member B or C) was indicated using a green rectangle, whereas the person entering later was indicated using a blue rectangle. Finally, the remaining two participants walked around in the cabin, and the participant marked by the green rectangle exited. Throughout the experiment, stable and accurate tracking of the target (crew member A) was realized.

Because of the optimizations in the lightweight network architecture design and inference acceleration implemented with the TensorRT engine, the head-detection DCNN runs faster than 30 frames per second on the Tegra X2. However, because of the time spent on image fusion during image

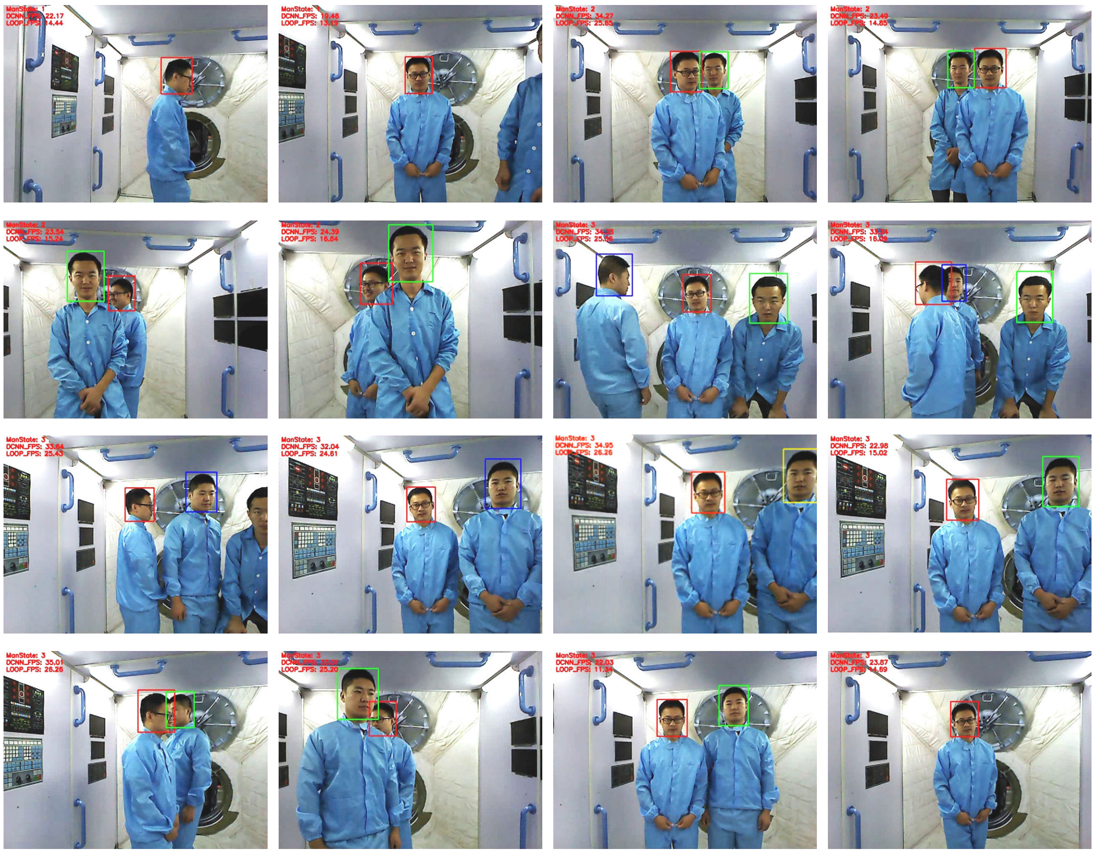


FIGURE 16. Qualitative experimental results of three-person tracking.

preprocessing, the average speed of the tracking algorithm is limited to about 15 frames per second.

To ensure the accuracy of the color and depth image registration, the field of view of an RGB-D camera is usually small. The field of view of the RGB-D camera used in this paper is about $58.4^\circ \times 45.5^\circ$. For astronaut assistance applications in the cabin, the robot is usually closer to the astronaut it assists. If the maneuverability of the robot is weak and it cannot follow a moving astronaut in real time, the astronaut could exit the field of view of the robot. As a result, the tracking would fail and stop. Therefore, the algorithm is more suitable for tracking an astronaut in collaborative working scenarios in the cabin. In such scenarios, the astronauts are almost always standing still or moving at a moderate speed. As a result, the tracking is more stable.

VII. CONCLUSION

In this paper, we tracked a robot-assisted astronaut in several in-cabin collaborative working scenarios using human head features. A highly reliable head detection DCNN was designed and trained. Based on the head detection results and the fused color and depth images, a multi-target tracking algorithm for head tracking was proposed. The tracking algorithm was then deployed on the robot's Tegra X2 microprocessor. Online application experiments were carried out, and the experimental results verified the effectiveness of the algorithm. This algorithm provides a good solution to astronaut-tracking problem in in-cabin collaborative working scenarios. In future work, we will focus on the engineering implementation of the tracking algorithm on the robot. In addition to the astronaut tracking program, there are many other programs that need to be run on the robot's embedded computing platform. To ensure the efficiency and stability of the tracking algorithm, further optimization of the allocation of the embedded CPU and GPU computing resources should be considered.

REFERENCES

[1] G. A. Dorais and Y. Gawdiak, "The personal satellite assistant: An internal spacecraft autonomous mobile monitor," in *Proc. IEEE Aerosp. Conf.*, Big Sky, MT, USA, Mar. 2003, pp. 333–348.

[2] T. Smith, J. Barlow, M. Bualat, T. Fong, C. Provencher, H. Sanchez, and E. Smith, "Astrobee: A new Platform for Free-Flying Robotics on the International Space Station," in *Proc. 13th Int. Symp. Artif. Intell., Robot., Automat. Space (i-SAIRAS)*, Beijing, China, 2016, pp. 83–86.

[3] D. Miller, A. Saenz-Otero, J. Wertz, A. Chen, G. Berkowski, C. Brodel, S. Carlson, D. Carpenter, S. Chen, S. Cheng, and D. Feller, "SPHERES: A testbed for long duration satellite formation flying in micro-gravity conditions," in *Proc. AAS/AIAA Space Flight Mech. Meeting*, Clearwater, FL, USA, 2000, pp. 167–179.

[4] M. Micire, T. Fong, T. Morse, E. Park, C. Provencher, E. Smith, V. To, R. J. Torres, D. W. Wheeler, and D. Mittman, "Smart SPHERES: A telerobotic free-flyer for intravehicular activities in space," in *Proc. AIAA SPACE Conf. Expo.*, San Diego, CA, USA, Sep. 2013, pp. 5338–5352.

[5] S. Mitani, M. Goto, R. Konomura, Y. Shoji, K. Hagiwara, S. Shigeto, and N. Tanishima, "Int-Ball: Crew-supportive autonomous mobile camera robot on ISS/JEM," presented at the IEEE Aerosp. Conf., Big Sky, MT, USA, Mar. 2019.

[6] J. Liu, Q. Gao, Z. Liu, and Y. Li, "Attitude control for astronaut assisted robot in the space station," *Int. J. Control, Autom. Syst.*, vol. 14, no. 4, pp. 1082–1095, Aug. 2016.

[7] Q. Gao, J. Liu, T. Tian, and Y. Li, "Free-flying dynamics and control of an astronaut assistant robot based on fuzzy sliding mode algorithm," *Acta Astronautica*, vol. 138, pp. 462–474, Sep. 2017.

[8] R. Zhang and Z. K. Wang, "An intelligent nano-satellite for astronaut assistance," presented at the 69th Int. Astron. Congr. (IAC), Bremen, Germany, Oct. 2018.

[9] R. Zhang, Z. Wang, and Y. Zhang, "Astronaut visual tracking of flying assistant robot in space station based on deep learning and probabilistic model," *Int. J. Aerosp. Eng.*, vol. 2018, pp. 1–17, Jul. 2018.

[10] Z. Rui, W. Zhaokui, and Z. Yulin, "A person-following nanosatellite for in-cabin astronaut assistance: System design and deep-learning-based astronaut visual tracking implementation," *Acta Astronautica*, vol. 162, pp. 121–134, Sep. 2019.

[11] X. Sun, P. Wu, and S. C. H. Hoi, "Face detection using deep learning: An improved faster RCNN approach," *Neurocomputing*, vol. 299, pp. 42–50, Jul. 2018.

[12] Y. Ren, C. Zhu, and S. Xiao, "Small object detection in optical remote sensing images via modified faster R-CNN," *Appl. Sci.*, vol. 8, no. 5, p. 813, May 2018.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA, Jun. 2014, pp. 580–587.

[14] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 1440–1448.

[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Montréal, QC, Canada, 2015, pp. 91–99.

[16] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Barcelona, Spain, 2016, pp. 379–387.

[17] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 2117–2125.

[18] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into high quality object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 6154–6162.

[19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 779–788.

[20] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 6517–6525.

[21] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*. [Online]. Available: http://arxiv.org/abs/1804.02767

[22] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*. [Online]. Available: http://arxiv.org/abs/2004.10934

[23] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, Amsterdam, The Netherlands, 2016, pp. 21–37.

[24] J. Jeong, H. Park, and N. Kwak, "Enhancement of SSD by concatenating feature maps for object detection," 2017, *arXiv:1705.09587*. [Online]. Available: http://arxiv.org/abs/1705.09587

[25] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD : Deconvolutional single shot detector," 2017, *arXiv:1701.06659*. [Online]. Available: http://arxiv.org/abs/1701.06659

[26] Y. Zha, T. Ku, Y. Li, and P. Zhang, "Deep position-sensitive tracking," *IEEE Trans. Multimedia*, vol. 22, no. 1, pp. 96–107, Jan. 2020.

[27] Q. Zhou, B. Zhong, Y. Zhang, J. Li, and Y. Fu, "Deep alignment network based multi-person tracking with occlusion and motion reasoning," *IEEE Trans. Multimedia*, vol. 21, no. 5, pp. 1183–1194, May 2019.

[28] Q. Zhou, B. Zhong, X. Lan, G. Sun, Y. Zhang, B. Zhang, and R. Ji, "Fine-grained spatial alignment model for person re-identification with focal triplet loss," *IEEE Trans. Image Process.*, vol. 29, pp. 7578–7589, 2020.

[29] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," 2018, *arXiv:1808.06048*. [Online]. Available: http://arxiv.org/abs/1808.06048

[30] X. Li, C. Ma, B. Wu, Z. He, and M. H. Yang, "Target-aware deep tracking," 2018, *arXiv:1808.06048v1*. [Online]. Available: http://arxiv.org/abs/1808.06048v1

[31] T. Yang and A. B. Chan, "Recurrent filter learning for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 2010–2019.

[32] D. Ma, W. Bu, and X. Wu, "Multi-scale recurrent tracking via pyramid recurrent network and optical flow," in *Proc. 29th Brit. Mach. Vis. Conf. (BMVC)*, Newcastle, U.K., 2018, p. 242.

[33] F. Zhao, J. Wang, Y. Wu, and M. Tang, "Adversarial deep tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 7, pp. 1998–2011, Jul. 2019.

[34] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. H. Lau, and M.-H. Yang, "VITAL: VIsual tracking via adversarial learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, Jun. 2018, pp. 8990–8999.

[35] S. M. Marvasti-Zadeh, L. Cheng, H. Ghanei-Yakhdan, and S. Kasaei, "Deep learning for visual tracking: A comprehensive survey," 2019, *arXiv:1912.00535*. [Online]. Available: http://arxiv.org/abs/1912.00535

[36] S. You, H. Zhu, M. Li, and Y. Li, "A review of visual trackers and analysis of its application to mobile robot," 2019, *arXiv:1910.09761*. [Online]. Available: http://arxiv.org/abs/1910.09761

[37] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," 2016, *arXiv:1602.07360*. [Online]. Available: http://arxiv.org/abs/1602.07360

[38] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," 2017, *arXiv:1707.01083*. [Online]. Available: http://arxiv.org/abs/1707.01083

[39] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: http://arxiv.org/abs/1704.04861

[40] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 4510–4520.

[41] P. Meloni, A. Capotondi, G. Deriu, M. Brian, F. Conti, D. Rossi, L. Raffo, and L. Benini, "NEURAghe: Exploiting CPU-FPGA synergies for efficient and flexible CNN inference acceleration on Zynq SoCs," 2017, *arXiv:1712.00994*. [Online]. Available: http://arxiv.org/abs/1712.00994

[42] P. Colangelo, N. Nasiri, A. Mishra, E. Nurvitadhi, M. Margala, and K. Nealis, "Exploration of low numeric precision deep learning inference using intel FPGAs," 2018, *arXiv:1806.11547*. [Online]. Available: http://arxiv.org/abs/1806.11547

[43] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, L. Jiao, L. Qendro, and F. Kawsar, "DeepX: A software accelerator for low-power deep learning inference on mobile devices," in *Proc. 15th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Vienna, Austria, Apr. 2016, pp. 23–34.

[44] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "AMC: AutoML for model compression and acceleration on mobile devices," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Munich, Germany, Sep. 2018, pp. 784–800.

[45] *TensorRT 3.0 Developer Guide*. Accessed: Oct. 2, 2018. [Online]. Available: https://docs.nvidia.com/deeplearning/sdk/tensorrt-archived/tensorrt_304/tensorrt-developer-guide/index.html

[46] Á. Arcos-García, J. A. Álvarez-García, and L. M. Soria-Morillo, "Evaluation of deep neural networks for traffic sign detection systems," *Neurocomputing*, vol. 316, pp. 332–344, Nov. 2018.

[47] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li, "FaceBoxes: A CPU real-time face detector with high accuracy," presented at the IEEE Int. Joint Conf. Biometrics, Rio de Janeiro, Brazil, Jul. 2018.

[48] S. Shao, Z. Zhao, B. Li, T. Xiao, G. Yu, X. Zhang, and J. Sun, "CrowdHuman: A benchmark for detecting human in a crowd," 2018, *arXiv:1805.00123*. [Online]. Available: http://arxiv.org/abs/1805.00123

RUI ZHANG was born in Sichuan, China, in 1989. He received the B.S. degree in automation from the University of Electronic Science and Technology of China, Chengdu, China, in 2012, and the M.S. and Ph.D. degrees in aerospace science and technology from the National University of Defense Technology, Changsha, China, in 2014 and 2019, respectively.

From 2015 to 2019, he studied with Tsinghua University as a Joint Training Student. Since December 2019, he has been working as a Lecturer with Space Engineering University, Beijing, China. He is the first author of four articles and one invention. His research interests include distributed space systems and artificial intelligence.

YULIN ZHANG was born in Shanxi, China, in 1958. He received the B.S. and M.S. degrees from the National University of Defense Technology, Changsha, China, in 1982 and 1984, respectively, and the Ph.D. degree from the Department of Industrial Automation of Zhejiang University, Hangzhou, China, in 1988.

He is currently a Professor with Tsinghua University, Beijing, China. He is the author of 13 books, more than 200 articles, and more than 20 inventions. His research interests include space system dynamics and control, distributed spacecraft and fast response space systems, earth-moon space architecture, and intelligent systems. He was a recipient of the Manlina Badge for Excellence in the conference of the 34th International Astronautical Federation.

XUEYANG ZHANG was born in Hefei, China, in 1988. He received the B.S. degree in mathematics from the University of Science and Technology of China, Hefei, in 2010, and the M.S. degree in system science and the Ph.D. degree in aerospace science and technology from the National University of Defense Technology, Changsha, China, in 2012 and 2017, respectively.

Since December 2017, he has been working as a Lecturer with Space Engineering University, Beijing, China. He is the first author of 11 articles and three inventions. His research interests include information analysis and artificial intelligence.