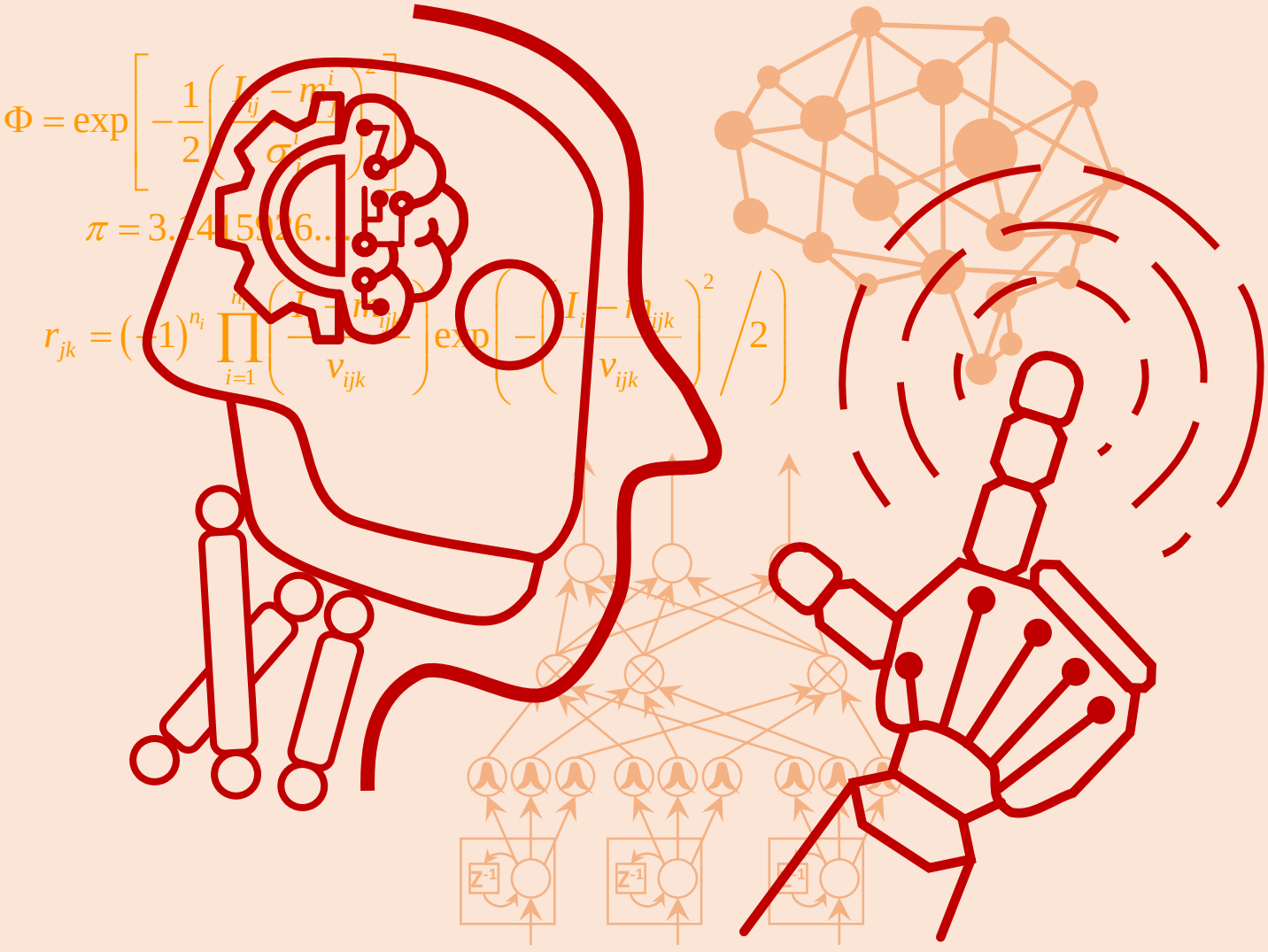
ISSN 2616-8170

Robotics Society of Taiwan

iRobotics

VOLUME 2
NUMBER 2
JUNE 2019



PUBLISHED BY THE ROBOTICS SOCIETY OF TAIWAN

*i*Robotics

PUBLISHER

Robotics Society of TAIWAN (RST)
Society President: Ching-Chih Tsai

Department of Electrical Engineering, National Chung Hsing University
Taichung, Taiwan

Tel: +886-4-2285-1549#601
URL: http://www.rst.org.tw

The ***i*Robotics** is published quarterly each year by the Robotics Society of Taiwan (RST). Institutional rate: US$140 annually; individual annual subscription rate: US$50 for nonmembers, US$25 for members (including postage). Note that another US$100 is needed if the express is required.

Lane Detection Approaches: RANSAC and Deep Convolutional Image Segmentation

Asheber Techane Wagshum, Anjana Kumar, Yu-Cheng Kuo and Chung-Hsien Kuo*

***Abstract*—Development of end-to-end advanced driver assistance system (ADAS) becomes handy with current paradigm shift in computer vision and deep learning based image processing. Such advanced system can provide a safety precaution for drivers during driving. Lane detection is a key aspect of developing driving safety system. In this paper, two approaches including random sample consensus (RANSAC) and deep convolutional image segmentation were discussed. The RANSAC approach used the video streams collected from PAPAGO GoSafe 530G camera to detect the lane appearing on the front scene of the vehicle on highway. In addition, a deep learning based semantic segmentation architecture called SegNet was used for land detection based on the benchmark CamVid dataset. The SegNet model was trained for 11 classes. On validation, it was found that the mean intersection over union (mIOU) is around 60.1 and the global average of the model is 90.40.**

Index Terms—lane detection, random sample consensus, deep convolutional neural network, image segmentation.

I. Introduction

THE current, advance in driver assist system (DAS) has significantly improved the vehicle safety records [1]. Many scholars have been researching the development of driver assistance system [2] in quest for new algorithms which can have the better understanding of the road scene. According to a survey, 59% of all road accidents were caused by lane departure [3]. Hence, in an efficient DAS system detecting the road lanes accurately is one sought requisite. This research aims at developing lane detection system to improve the traffic safety and avoid accidents due to human error. Moreover, a camera is used in vision-based systems [4] as sensing device due to its comparatively low cost than radar or laser technology and relatively well developed base knowledge on computer vision algorithms.

Fast growing prospect of self-deriving cars, regardless of philosophical question on safety issues surrounding it, has introduced advanced deriving technologies. On the other hand, the race in development of autonomous vehicle(AV) is showing remarkable results [5]. Nevertheless, the effort to make human-controlled driving safe has continued and its showing promising results. The juncture, sometimes with overlapping results, of these three line of researches is a promise for emergency of new technologies in driving experience.

Usually, highways and freeway roads will have well marked multiple numbers of lanes. It's through this line that drivers sharing common road communicate for safe drive. It is obvious that the car needs to move either within a lane or change its track to another lane. Lanes are the curvature and boundaries of every road segment. Each lane is considered as the reference line [6] for driver assistant vehicle to maintain the driving path and safe distance from other vehicles. Hence, it is essential to detect the lane quickly and more accurately. Additional lane information under different lightning environment is required to make the rapid decision and necessary action.

Computer vision is essentially integrating itself into our daily life such as object recognition, healthcare automation, robotics control, autonomous vehicle and driver assist system [7]. Computer vision system is used to capture the road scene images from the front view camera and recognize objects in scene and other important road information such as road signs and markings [8]. Under the structured road condition, it is assumed that the road is flat and lanes are in parallel with each other. Furthermore, same interval is maintained by the lanes. The major problem in lane detection is the lanes are intermittent and insufficient for detection. As a result, no lanes are detected which is a very serious challenge in driver assistant system. Therefore, region of interest (ROI) estimation, and a deep learning based lane detection mechanism are proposed in this paper.

Deep learning is specific form of machine learning which learn data representation without explicitly dictated features [9]. Deep learning uses several hidden layers and neurons as compared to traditional neural network [10]. Essentially deep learning is extracting features of dataset it introduced to such that it able to classify or predict information embed in the data. Various deep learning methods such as deep neural networks, convolutional neural network (CNN), deep recurrent

This work was partially supported by the National Science Council, Taiwan, R.O.C. under Grants NSC MOST MOST-106-2221-E-011-004-MY3 and the "Center for Cyber-physical System Innovation" from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan.

A. T. Wagshum, A. Kumar, Y. C. Kuo and C. H. Kuo are with the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, 106 Taiwan (corresponding author e-mail: chkuo@mail.ntust.edu.tw;).

networks (RNN), etc. have been proposed which can be applied in computer vision, image recognition, and driver assistant system. CNN [11] model can effectively be used in lane detection. Hence, a CNN model is used in our proposed work, to detect the lane segments from the road image taken by the camera present in the car.

The rest of the paper is organized as follows. Existing works on lane detection, computer vision, and deep learning are discussed in section 2. In section 3, a RANSAC approach for lane detection is proposed. In section 4, a deep learning based semantic segmentation architecture with SegNet for lane detection is discussed. Experimental setup and results of two approaches are given in section 5 and the concluding remarks are made in section 6.

II. Related Work

Many lane detection mechanisms have been proposed in the recent past. In [12], a drivable path detection algorithm was proposed for the autonomous vehicles. The lanes of the road were detected in real-time by analysing each frame of video through edge detection and Hough transform techniques for safe navigation. Similarly, [11] has developed end-to-end lane position estimating deep neural network model that inputs images from laterally-mounted down-facing cameras. In [13], a lane detection method was introduced by integrating lane shape for online vehicle position calculation. A lane detection and departure warning system were integrated and investigated by [14]. However, long processing time and high computation are required by Hough transformation.

In [15], a parallel processing model was used for Hough transformation and image analysis for faster lane detection. A reliable lane detection method was proposed in [16] based on spatiotemporal images. The images were generated by accumulating the consecutive scan line pixels to improve the accuracy of detection. In [17], a computer vision based multi-vehicle detection mechanism was proposed by considering vehicle location in a single CNN model. However, different lighting effect is not considered in the existing works. Hence, we plan to incorporate unlike lighting effect in our work to improve the detection efficiency.

In [18], a deep neural network-based car-following model was proposed by considering velocity and position difference as inputs. By using deep learning, the feature extraction is more accurate to describe complicated human actions. However, the lane position is not considered in this work which also equally important in a driver assistant system. In [19], a deep fully convolutional neural network architecture known as SegNet was proposed for semantic pixel-wise segmentation images. An encoder and corresponding decoder network were designed for pixel-wise classification and feature map production. The performance of SegNet is better in the segmentation process. Hence, we plan to use the SegNet model as our base CNN model.

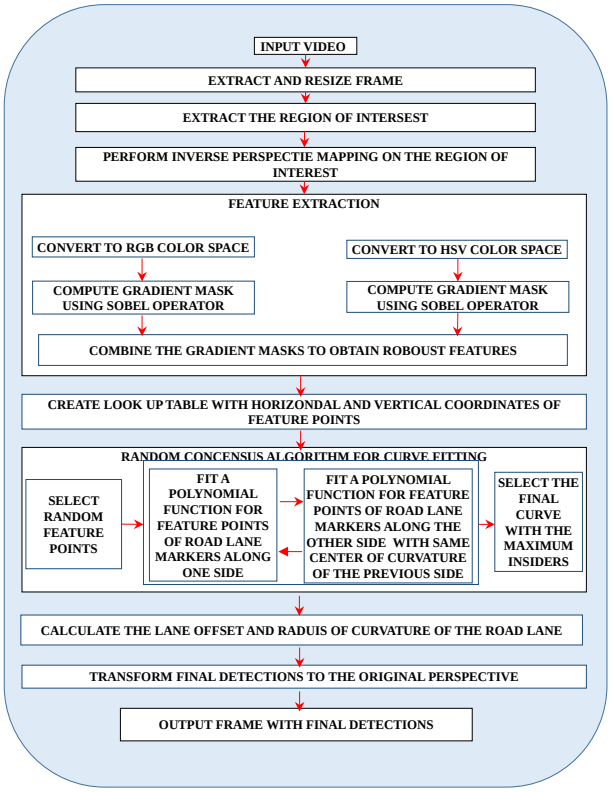


Fig. 1. Flow char of the proposed RANSAC lane detection appatroach.

III. Method I: RANSAC

In this section, a RANSAC approach is proposed to detect the lane on the highway. The flow chart of the proposed approach is shown in Figure 1.

For road lane detection, the road lane markers which are white or yellow in color are used as reference to obtain the boundary of the road lane in which the vehicle is present. Each frame from the input video data is extracted and resized. The region of interest (ROI) which represents the lane in which the vehicle is present is extracted and the inverse perspective transform is computed to obtain the top view. Utilizing the image obtained in HSL color space and in gray scale image, gradient mask to extract the features corresponding to the yellow and white road lane markers is computed. The gradient mask computed for each color space is combined to obtain final feature points corresponding to the road lane markers. A look up table is created to record the horizontal and vertical coordinates of the feature points extracted. Random sample consensus algorithm is used for fitting a curve for the extracted feature points.

The feature points are extracted for road lane markers along the right and left side of the vehicle. Primarily, the feature points of the road marker along left side is utilized. A random set of points are taken and a curve fitting function is used to compute the parameters of the polynomial and the number of feature points that are well-fitting is computed. After iterating over all the subset sets of feature points that are

randomly selected, the regression parameters which can fit the maximum number of feature points are taken as the final parameter. These well-fitting feature points are known as insiders.

For the feature points along the right side, similar process is used to obtain the final regression parameters. But, to compute the final regression parameter, an additional condition is used. The center of curvature of the curve obtained through the curve fitting function for the feature points on the right side of the lane must be same as that of the left side. Hence both the curves obtained are parallel to one another. Similar process is done taking the feature points on the right side of the lane as reference. From the two results obtained, the curve which has the maximum number of insiders is taken as the final estimation.

The radius of curvature of the lane and the lane offset are computed. To represent the final prediction, the lane boundary prediction is converted to the original perspective and is plotted on the original input frame for visualization.

A. Extraction of Region of Interest (ROI) and Perspective Transformation

The road lane is present in the lower half of the input image in Figure 2. The region of interest (ROI) represents the road lane in which the vehicle is present. Every frame from the input video is extracted and resized to 1280 X 720 resolution. This region of interest is a trapezoidal region which is extracted from the resized image. Perspective transformation for this region of interest (ROI) is computed to obtain the top view of the road lane. Perspective transformation or Inverse perspective mapping is computed to avoid the perspective effect.

B. Feature Extraction

To estimate the boundary of the road lane in which the vehicle is present, the road lane markers which are either yellow and white in color are used as reference. The perspective transformed image is converted to gray scale and in HSL color space. Using the Sobel operator, gradient mask is computed for both gray scale image and for the saturation channel of the HSL color space. The Sobel operator uses a 3X3 kernels which is convolved with the original image to calculate approximations of the derivatives. There are two different kernels each for computing the approximation of derivatives along the horizontal and vertical direction.

Let **I** be the original image and $\mathbf{G}_x$ and $\mathbf{G}_y$ are images representing the horizontal and vertical derivative approximation. The calculation is represented as (1) – (2),

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * I \quad (1)$$

$$G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I \quad (2)$$

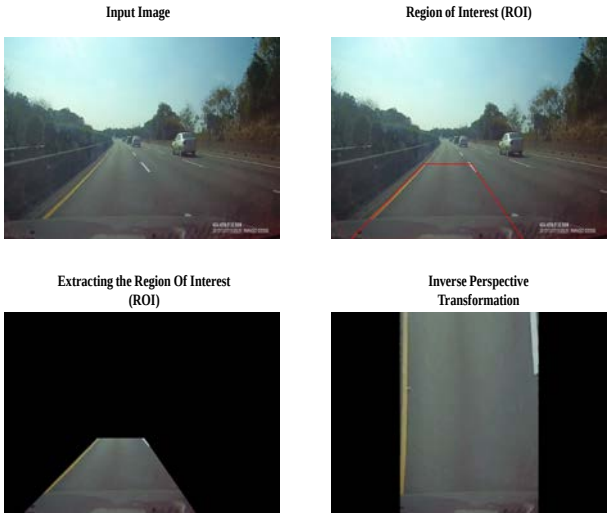


Fig. 2. Region of interest extraction and inverse perspective transform.

where, *denotes the 2-dimensional signal processing convolution operation.

At each point in the image **I**, the resulting gradient approximations $\mathbf{G}_x$ and $\mathbf{G}_y$ can be combined to obtain the gradient magnitude **G** as (3); the gradient direction can be computed as (4).

$$G = \sqrt{G_x^2 + G_y^2} \quad (3)$$

$$\theta = \text{atan}\left(\frac{G_y}{G_x}\right) \quad (4)$$

After computing the gradient mask for gray scale and the saturation channel of HSL color space, the image is divided into two and the feature points corresponding to the road lane markers along the left and the right side are detected as shown in Figure 3. A look up table is created which contains the x and y coordinates of the feature points which represents the road lane markers.

C. RANSAC Based Curve Fitting

Random sample consensus algorithm (RANSAC) is an iterative method to estimate various parameters of a mathematical model. This algorithm iterates over a set of data which contains data points that can fit well as well as outliers. Outliers are data points which are distant from other observations. The presence of these outliers do not affect the estimates of parameters of the mathematical model. Hence, this algorithm can detect outliers from the given data points.

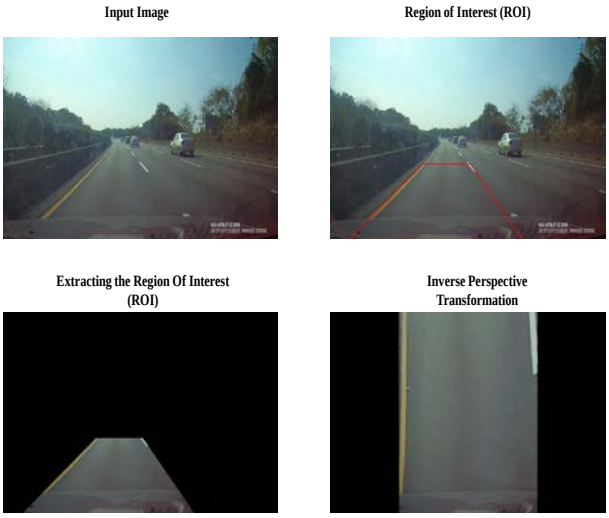


Fig. 3. Feature points extracted from grayscale image and from HLS color space.

Basically, a set of data points can be classified into two namely inliers and outliers. Inliers are data points which can fit well in a mathematical model whereas outliers are data points which are distant from the mathematical model. The main advantage of using RANSAC is that the presence of outliers in the input data points do not influence the estimation of parameters for a mathematical model. The RANSAC algorithm consists of two steps which are executed iteratively,

Step 1: A subset of data points from the given data set is randomly selected. The parameters for the curve fitting model is computed using this subset.

Step 2: The algorithm checks if the input data point fits the in the model with the estimated parameters from the first step with other data points in the given data set apart from data points which were randomly picked in the step 1. Data points which do not fit well in the model with estimated parameters within the given threshold are called outliers.

Through the above steps, set of data points which are well-fitting with the model is obtained. These data points are known as inliers. The RANSAC algorithm iterates over the input data points until sufficient amount of inliers are obtained. This set of inliers is known as consensus set.

The input to the RANSAC algorithm are input data points, the model for which data points must be fit, maximum number of iterations and threshold to determine if the data points can be classified as inlier. For road lane detection system, input data points are taken from the look up table containing the features corresponding to road lane markers. Linear regression is used to estimate parameters of the curve fitting function. The overall algorithm for RANSAC based curve fitting is shown in Table I, and their results are shown in Figures 4 - 5. The RANSAC algorithm is shown in Table II.

Table I Overall algorithm for RANSAC curve fitting

Algorithm 1 RANSAC curve fitting	
Step 1	Get input feature points of left lane markers (x_1, y_1) and right lane markers (x_2, y_2) and maximum number of iterations. Check for consistency of data points.
Step 2	Fit the feature points of left lane markers (x_1, y_1) using RANSAC linear regression for estimation of parameters for second degree curve equation.
Step 3	Accumulate inliers estimated by for every iteration of RANSAC linear regressions and select the best fit based on the maximum number of inliers estimated from each iteration.
Step 4	Fit the feature points of right lane markers (x_2, y_2) using RANSAC linear regression for estimation of parameters for second degree curve equation.
Step 5	Accumulate inliers estimated by for every iteration of RANSAC linear regressions and select the best fit by using the curve which has the same center of curvature as that of the fit selected for the feature points of left road lane markers.
Step 6	Similarly, fit the feature points of right lane markers (x_2, y_2) using RANSAC linear regression for estimation of parameters for second degree curve equation.
Step 7	Accumulate inliers estimated by for every iteration of RANSAC linear regressions and select the best fit based on the maximum number of inliers estimated from each iteration.
Step 8	Fit the feature points of left lane markers (x_1, y_1) using RANSAC linear regression for estimation of parameters for second degree curve equation.
Step 9	Accumulate inliers estimated by for every iteration of RANSAC linear regressions and select the best fit by using the curve which has the same center of curvature as that of the fit selected for the feature points of right road lane markers.
Step 10	Select the estimated parameters for the pair of fit which has the maximum number of feature points are classified as inliers.

Table II Overall algorithm for RANSAC curve fitting

Algorithm 2 Random sample consensus algorithm	
Step 1	*Ransac_function* (x, y, maximum_iteration, validity_bounds): *for* (i <= maximum_iteration) (a, b) = Randomly select data points from (x, y) to form a subset Solve linear regression. Check for other data points which fit well in the mathematical model for the estimated parameters. Check for number of inliers obtained for the estimated parameters
Step 2	Select estimated parameters for which maximum number of inliers are obtained as the final fit for the given data points

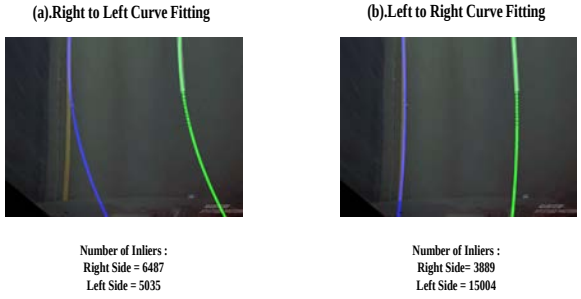


Fig. 4. Results of RANSAC curve fitting.

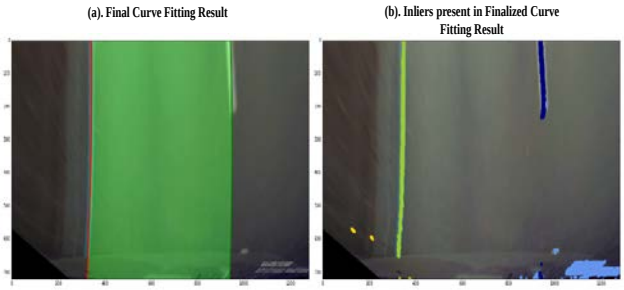


Fig. 5. Results of final curve fitting.

D. Transformation to Original Perspective and Calculation of Curvature and Lane Offset

After estimating the boundary of the road lane in which the vehicle is present using the top view image, the final detection is transformed back to the original perspective. The curvature of the lane and the lane offset are calculated using the linear regression parameters of the final detection. For a second order polynomial of form as (5), the radius of curvature R can be calculated as (6). Where A, B, C are parameters to represent the polynomial curve fitting equitation. The final lane detection experiments were done via Taiwan's No. #1 highway, and the results are shown in Figures 6 - 7.

$$Ax^2 + Bx + C = 0 \tag{5}$$

$$R = \frac{\left(1+\left(2Ax+B^2\right)\right)^{3/2}}{|2A|} \tag{6}$$

Fig. 6. Transformation to the original perspective and calculation of lane curvature and lane offset.

IV. Method II: Image Segmentation

In addition to RANAC algorithm, this paper also discuss a deep learning based image segmentation approach. The step-by-step procedure of the proposed method is shown in Figure 8. The video of the road scene is taken as input and each frame is extracted. SegNet [19] which is a deep learning based semantic segmentation algorithm is used to segment the pixel corresponding to the road surface marking in the image. Afterward, the perspective transformation is computed on the extracted feature to obtain the inverse perspective mapping or the bird's eye view. The histogram for the bottom half of the image is computed to determine the location of the feature corresponding to the road surface marking. Besides, a sliding window based search is used across the entire image to detect required features. Finally, a polynomial curve fitting function is used to fit the feature points.

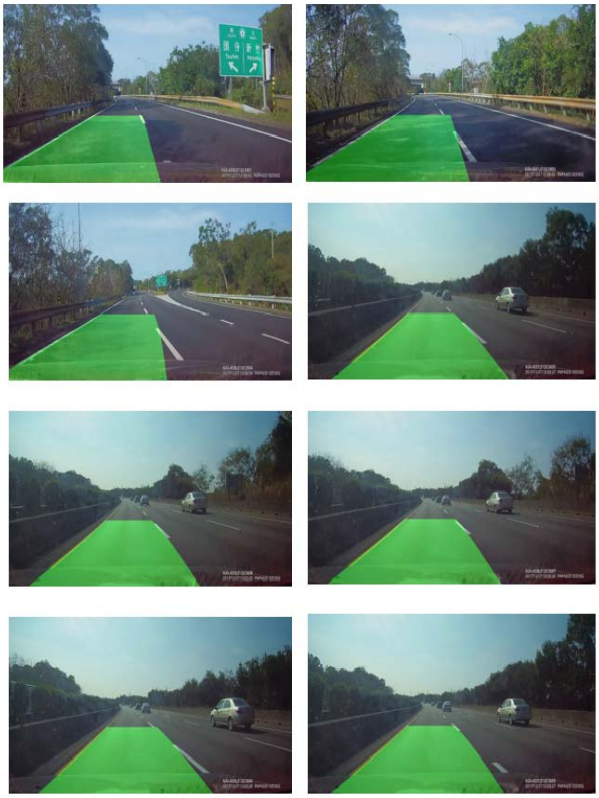


Fig. 7. Road lane detection using RANSAC curve fitting.

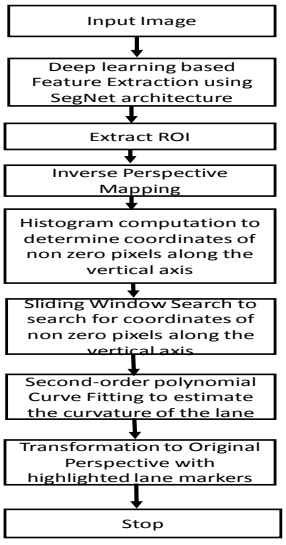


Fig. 8. Flowchart of the proposed image segmentation approach.

A. Deep Learning Based Feature Extraction

Extracting accurate features of the road marking and signs is necessary to build a robust navigation system. In this paper, a deep learning based semantic segmentation architecture called SegNet is used. The SegNet architecture is shown in

Figure 9. It is deep fully convolutional neural network architecture to perform pixel-wise semantic segmentation. It consists of an encoder network, a decoder network followed by a classification layer to enable pixel-wise classification.

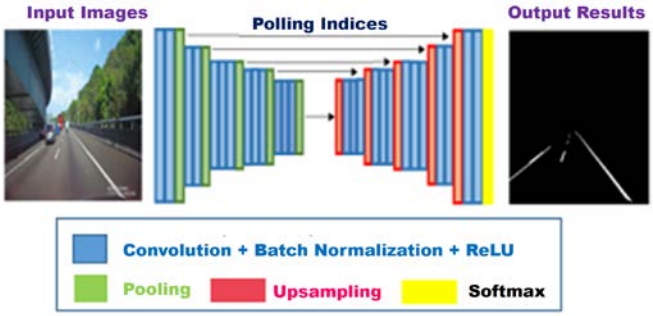


Fig. 9. SegNet: a deep convolutional encoder-decoder architecture for image segmentation.

The encoder network consists of 13 convolutional layers similar to VGG16 network [20] architecture. Therefore, pre-trained network parameters on large image dataset can be used to initialized the weights during training. The main purpose of the decoder is to map low-resolution feature maps to input resolution feature maps. The decoder uses the pooling indices calculated in the max-pooling layer in the encoder to perform upsampling thus eliminating learning to upsample during training. The upsampling performed by the decoder is non-linear by nature. In SegNet unlike the other decoder, the output decoder (decoder corresponding to first encoder) produces multi-channel feature maps regardless of the fact that the input to first encoder is RGB image [11]. The classification layer is used to classify every pixel in the image into its corresponding class.

The main advantage of the SegNet architecture is that the number of trainable parameters are less compared to other semantic segmentation architectures and the network can be end to end using stochastic gradient descent. This architecture is computation and memory efficient.

B. Inverse Perspective Mapping, Histogram Computation and Sliding Window Search

The feature extracted using SegNet semantic segmentation algorithm described above is plotted as a binary image. The region of interest (ROI), which span size of a lane, is selected to separate the feature representing the lane corresponding to the user's vehicle. This ROI is the constant trapezoidal area. Since the lane marker features are captured as oblique lines in the front view, the perspective effect must be removed for easier computation. To solve this issue, the perspective transformation to compute the Bird's eye view or top view of the image is performed as similarly to Figures 2 – 3 that were previously stated in *Subsection A* of Section III.

This transformed image is converted to gray scale and in HSL color space. Using the know Sobel operator, which can be a filter learned through CNN, gradient mask is computed for both gray scale image and for the saturation channel of the HSL color space.

To obtain the coordinates of the non-zero pixels along the horizontal axis, the histogram is drawn for the bottom half of the image. A sliding window search is used to scan along the vertical axis to obtain the position of non-zero pixels which represent the feature corresponding to the road surface marking. The width of the sliding window is set to 40 pixels. The centroid of the window can be adjusted to re-center the window.

D. Polynomial Curve Fitting and Transformation to Original Perspective

Transformation back to the original perspective view is necessary to compute curvature and lane offset. A second-degree polynomial curve fitting function is used to fit the coordinates obtained from the previous step. This is used to determine the shape of the area in between detected road markers and to understand the curvature of the road. Thus, after computing the location of the road surface marking, the image is transformed back to the original perspective position and the detected region is highlighted.

E. Experiment Results and Discussion

In this study, the benchmark dataset, CamVid dataset [20] is used to validate our lane detection method. The pre-trained weights for the deep learning based semantic segmentation model are used to initialize parameters at the start of the training. This model is trained for 3433 training images obtained for a combination of datasets [21] and [22]. The SegNet model is trained for 11 classes. On validation, it is found that the mIOU is around 60.1 and the global average of the model is 90.40 [18].

Threshold is set to separate the feature with respect to the road lane marker from the segmented image as shown in Figure 10. Deep learning based segmentation model is observed to be better than color and edge detection method as these conventional methods as shown in Figure 10 may not perform well under different lighting conditions. The region of interest is then separated as shown in Figure 10 and then this region is converted to top view or bird's eye view.

The histogram of the bottom half of the perspective transformed image is computed to locate the position of non-zero pixels along the x-axis. A sliding window based search is used to extract the coordinates of the non-zero pixels along the y-axis as shown in Figure 10. The centroid of the window is adjusted automatically to detect the non-zero pixels. A second-degree polynomial curve fitting function is used to fit the extracted coordinates. The image is then transformed to the original perspective and the detected lane is highlighted. This image is combined with the original image to provide the entire road scene with the detected lane. The entire process involves computation time of 0.355 per frame.

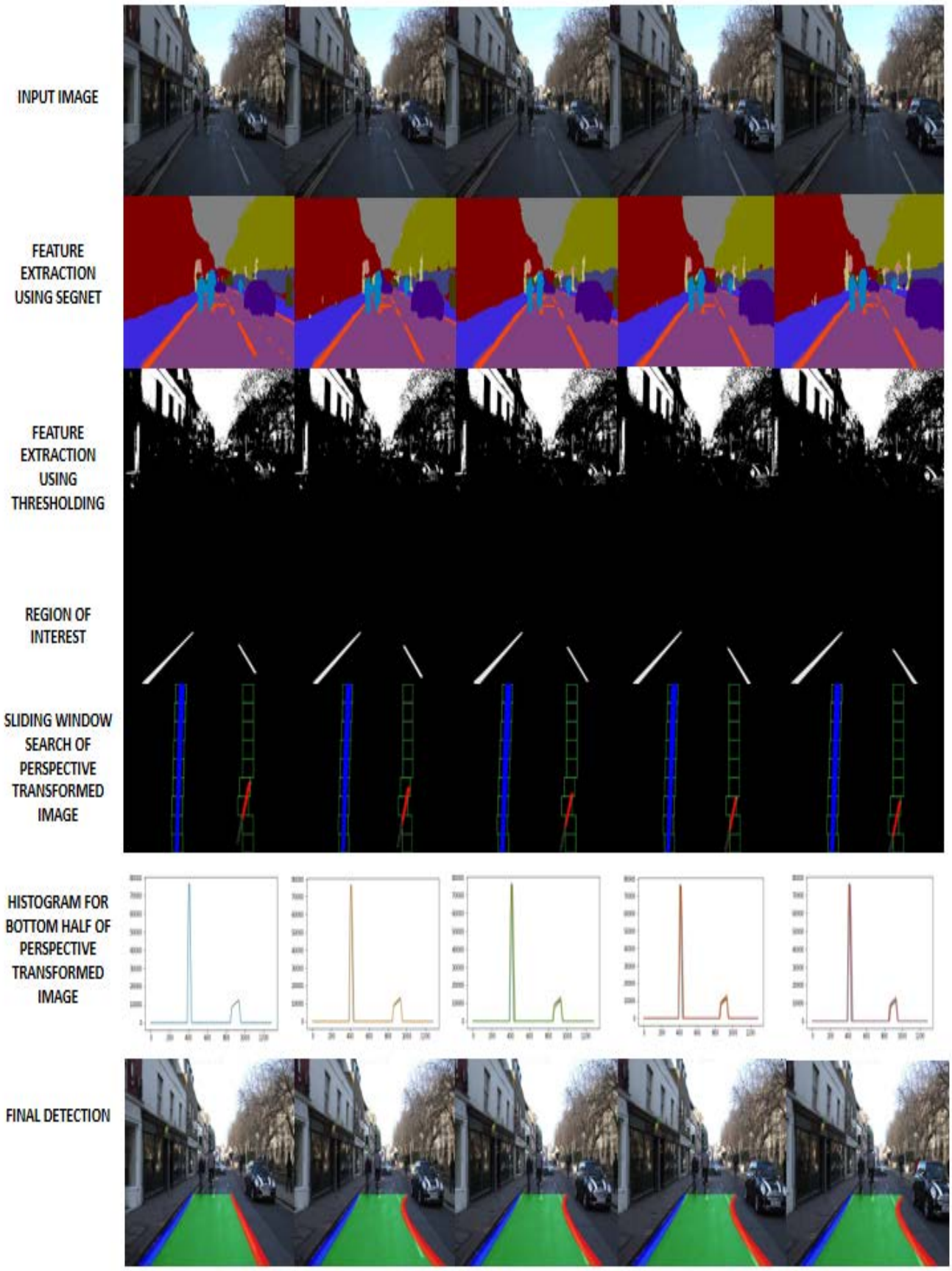


Fig. 10. Experimental Results.

V. Conclusion and Future Works

In this paper, we have presented a RANSAC and a deep learning based lane detection approaches for advanced driver assistant system. The RANSAC used the algorithms proposed in Tables I and II to perform robust lane detections in the highway. In addition, by incorporating deep learning based semantic segmentation algorithm for feature extraction, the lane detection system is more accurate and robust with 0.355s processing time.

In the future, this work can be extended to improve the detection speed to identify the lane and a warning system can be designed for the driver during overtaking of the vehicle in a freeway environment.

References

[1] J. Dai, L. Wu, H. Lin, and W. Tai, "A driving assistance system with vision based vehicle detection techniques," in *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2016, pp. 1-9.

[2] L. Fletcher, L. Petersson, and A. Zelinsky, "Driver assistance systems based on vision in and out of vehicles," in *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No.03TH8683)*, 2003, pp. 322-327.

[3] F. Zhang, H. Stähle, C. Chen, C. Buckl, and A. Knoll, "A lane marking extraction approach based on Random Finite Set Statistics," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 1143-1148.

[4] M. Jeong, B. C. Ko, and J. Nam, "Early Detection of Sudden Pedestrian Crossing for Safe Driving During Summer Nights," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 6, pp. 1368-1380, 2017.

[5] B. Huval *et al.*, *An Empirical Evaluation of Deep Learning on Highway Driving*. 2015.

[6] Jim *et al.*, "Improving the Lane Reference Detection for Autonomous Road Vehicle Control %J Journal of Sensors," vol. 2016, p. 13, 2016, Art. no. 9497524.

[7] M. Bolaños, M. Dimiccoli, and P. Radeva, "Toward Storytelling From Visual Lifelogging: An Overview," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 1, pp. 77-90, 2017.

[8] J. Yan, G. Feng, and X. Guoyan, "Computer vision-based multiple-lane detection on straight road and in a curve," in *2010 International Conference on Image Analysis and Signal Processing*, 2010, pp. 114-117.

[9] Soniya, S. Paul, and L. Singh, "A review on advances in deep learning," in *2015 IEEE Workshop on Computational Intelligence: Theories, Applications and Future Directions (WCI)*, 2015, pp. 1-6.

[10] D. Ravì *et al.*, "Deep Learning for Health Informatics," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 1, pp. 4-21, 2017.

[11] A. Gurghian, T. Koduri, S. V. Bailur, K. J. Carey, and V. N. Murali, "DeepLanes: End-To-End Lane Position Estimation Using Deep Neural Networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2016, pp. 38-45.

[12] J. Wadhwa, G. Kalra, and B. V. Kranthi, *Real Time Lane Detection in Autonomous Vehicles Using Image Processing*. 2015, pp. 429-433.

[13] H. Cai, Z. Hu, G. Huang, and D. Zhu, *Robust road lane detection from shape and color feature fusion for vehicle self-localization*. 2017, pp. 1009-1014.

[14] S. P. Narote, P. N. Bhujbal, A. S. Narote, and D. M. Dhane, "A review of recent advances in lane detection and departure warning system," *Pattern Recognition*, vol. 73, pp. 216-234, 2018/01/01/ 2018.

[15] C. Premachandra, R. Gohara, and K. Kato, "Fast lane boundary recognition by a parallel image processor," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2016, pp. 000947-000952.

[16] S. Jung, J. Youn, and S. Sull, "Efficient Lane Detection Based on Spatiotemporal Images," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, pp. 289-295, 2016.

[17] Y. Yao, B. Tian, and F. Wang, "Coupled Multivehicle Detection and Classification With Prior Objectness Measure," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 3, pp. 1975-1984, 2017.

[18] X. Wang, R. Jiang, L. Li, Y. Lin, X. Zheng, and F. Wang, "Capturing Car-Following Behaviors by Deep Learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 910-920, 2018.

[19] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481-2495, 2017.

[20] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014.

[21] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88-97, / 2009.

[22] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354-3361.

Asheber Techane Wagshum

Anjana Kumar

Yu-Cheng Kuo received the Ph.D. degree in electrical engineering from National Taiwan University of Science and Technology, Taipei, Taiwan, in 2017. He is currently a postdoctoral fellow at the Department of Electrical Engineering, National Taiwan University of Science and Technology. His research interests include humanoid robots, human-machine interface and smart automation.

Chung-Hsien Kuo received the Ph.D. degree in mechanical engineering from National Taiwan University in 1999. Since 2007 he has been with the Department of Electrical Engineering, National Taiwan University of Science and Technology (Taiwan TECH) and is currently a professor of Taiwan TECH. His research interests include humanoid robots, autonomous mobile robots, smart automation, medical instrumentation and brain-computer interface.

Autonomous Navigation of an Indoor Mecanum-Wheeled Omnidirectional Robot Using SegNet

Po-An Wei, Ching-Chih Tsai, *Fellow, IEEE,* and Feng-Chun Tai

***Abstract*-This paper proposes an autonomous navigation control structure using SegNet for autonomous navigation of an indoor Mecanum-wheeled omnidirectional robot (MWOR) in an indoor environment. The SegNet is used to achieve environment recognition and obstacle avoidance. The navigation control architecture is composed of one FastSLAM 2.0 module, one global path planning module using Dijstra algorithm, one obstacle avoidance module by fusing the outputs of the existing DWA method and SegNet, and one motion control module for the MWOR. The autonomous navigation experimental system is equipped with one Jetson TX2 module from Nvidia, one LiDAR, one OpenCR, one Intel RealSense D435i, and one MWOR. Localization and mapping of the working environment are done by the known FastSLAM 2.0 algorithm along with the MWOR's odometry and LiDAR scanning data, where the LiDAR data are also employed to avoid any collisions from any static, or dynamic, or unexpected moving objects. Intel RealSense D435i along with SegNet is used to detect environmental objects and obstacles in the environment. Experimental results are conducted to show the effectiveness and merits of the proposed autonomous navigation method.**

Index Terms—Deep learning, Omnidirectional mobile robot, autonomous navigation, SegNet.

I. INTRODUCTION

DEEP learning is very popular in recent years. Deep learning models have sometimes achieved increasing success due to the availability of massive datasets and extenting model depth and parameterisation. Nevertheless, practical factors licluding memory and computational time during training and testing are important factors to consider while choosing a model from a large bank of models. Hence,the time of training turns into a major consideration particularly while the performance gain is not commensurate with increased training time as shown in our experiments.Test time memory and computational load are important to deploy models on specialised embedded devices.From an overall efficiency viewpoint, less attention has been paid to smaller and more memory, time efficient models for immediate applications such as road scene understanding.It was the primary motivation behind the proposal of SegNet, which is significantly than other competing architectures. SegNet has been shown efficient for tasks such as road scene understanding.

Autonomous navigation of mobile robots or vehicles is expected to provide various services within living environments of humans [1]. Such a robotic technology has been ready to show its practical use in industry, but, so far, the robots for industry simply follow a given motion by humans.Therefore, we will conduct as a means to allow elderly and physically impaired people to travel to destinations within public facilities such as airports. Nowadays, image segmentation is more popular such that it can be used to identify the regions of interest in a scene or annotate the data [2]. In image recognition by deep learning, areas within the image are recognized and classified as, for example, a person, road, sidewalk, or building. With this recognition method, the vehicle can estimate its position and direction of travel with great reliability. The authors in [3] used deep learning to achieve image recognition and extraction of the travelable area by proceessing the images acquired from monocular camera images mounted on autonomous cars. The authors in [4] showed that an electric mobile robot autonomously travels through a passage, and its own position and direction were estimated using deep learning.In particular, this system in [4] operated by using a camera in a smart phone to obtain an input image in order to make it as simple as possible.

Mecanum-wheeled omnidirectional robots (MWORs) have been widely used for our living life and industrial material handling, such as omnidirectional wheelchairs, automatic guided vehicles, and etc. There are two kinds of MWORs built by using 45-degree and 90-degree Mecanum wheels. Unlike conventional differential driving, MWORs have the superior flexibility to move towards any position and orientation. MWORs can be made using different wheel configurations including three wheels, four wheels, car-like four wheels, and etc.

Motivated by [1-4], this paper aims to develop and verify an autonmous navigation system for an indoor car-like MWOR that has varous applications in industry and our daliy life. The proposed techniques would provide references for professionals working in this area, especially for researchers and engineers working for personal care robots.

Po-An Wei, Ching-Chih Tsai, and Feng-Chun Tai are with the Department of Electrical Engineering, National Chung Hsing University, Taichung 40227, Taiwan.
(Corresponding author Ching-Chih Tsai, email: cctsai@nchu.edu.tw)
(email: g106064501@mail.nchu.edu.tw, fctai@nchu.edu.tw)

The authors gratefully acknowledge financial support from the Ministry of Science and Technology, Taiwan, the R.O.C., under contract MOST 107-2221-E-005 -073-MY2.

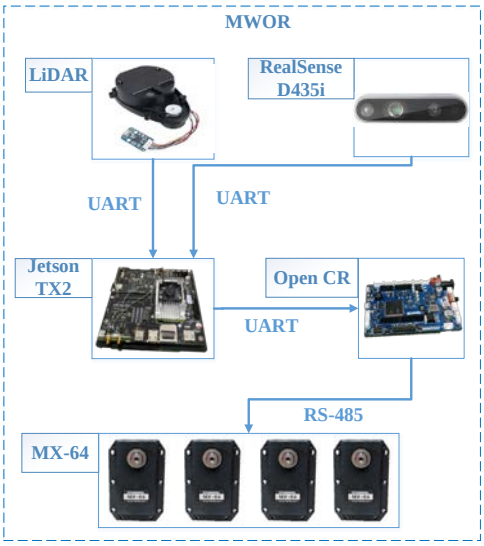


Figure 1. System structure and configuration of the experimental MWOR.

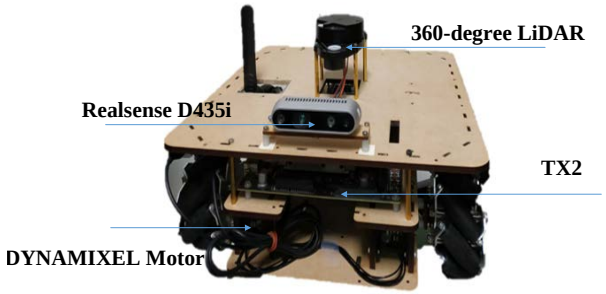


Figure 2. Picture of the experimental MWOR.

II. System Description of the Experimental MWOR

A. System Structure

Figure 1 shows the system configuration of the experimental Mecanum-wheeled omnidirectional mobile robot (MWOR). The autonomous navigation system is equipped with one Jetson TX2 module from Nvidia, one OpenCR, one Laser scanner (LiDAR), one Intel RealSense D435i, and one MWOR. The well-known FastSLAM2.0 method is adopted to address the localization and mapping problem by using the MWOR's odometry and LiDAR scanning data. The laser scanner is also employed to avoid any collisions from any static, or dynamic, or unexpected moving objects. The Intel RealSense D435i uses to environment and people recognition and obstacle avodidance for the working environment.

Figure 2 displays the physical configuration of the experimental MWOR, and Figure 3 shows the pose definitions of the laboratory-built MWOR. Moreover, Figure 4 depicts the four main modules of a general navigation control system, whereas Figure 5 illustrates the proposed autonomous navigation system of the experimental MWOR. As a special case of the general navigation control system, the proposed navigation control architecture includes the existing FastSLAM2.0 method, global path planning, environment and object recognition, 2D and 3D obstacle avoidance, and motion control of the experimental MWOR.

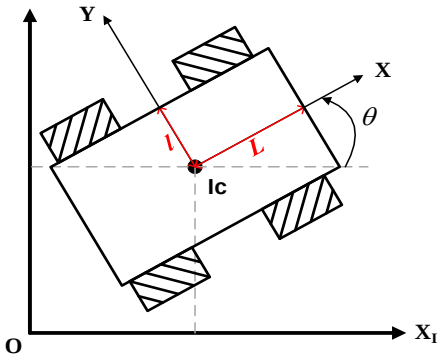


Figure 3. Pose definition of the laboratory-built MWOR.

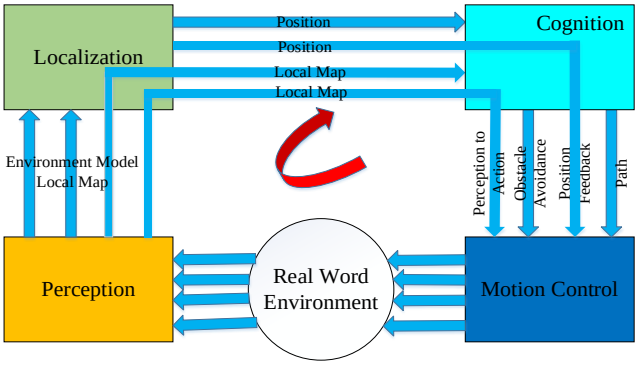


Figure 4. Flowchart of a general navigation control system.

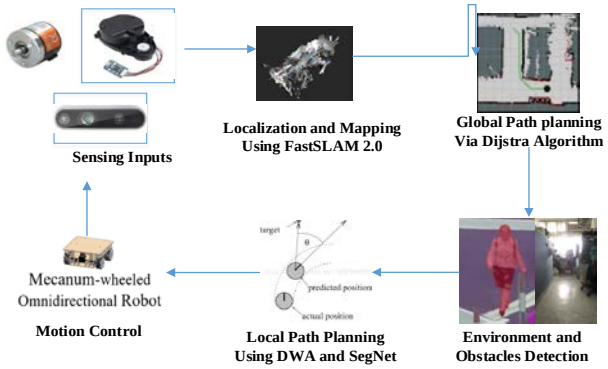


Figure 5. Flowchart of the proposed autonomous navigation system.

B. Kinematic Model of the Experimental MWOR

The subsection will briefly describe the kinematic model, both forward and inverse kinematic models of the experimental MWOR are formulated in the global frame, where the pose vector $\boldsymbol{x}=[x\ y\ \theta]^T$ denotes the position and orientation of the robot in the world frame as shown in Figure 3. The forward kinematics of the experimental mobile robot is then described by

$$\begin{bmatrix} \dot{x}_w \\ \dot{y}_w \\ \dot{\theta} \end{bmatrix} = \mathbf{J}(\theta)\boldsymbol{\upsilon}_w \tag{1}$$

where $\boldsymbol{\upsilon}_w = [\upsilon_{1w}\ \upsilon_{2w}\ \upsilon_{3w}\ \upsilon_{4w}]^T$ represents the velocity vector of the four wheels and $\upsilon_{iw}, i=1,...,4$, denotes the speed of the ith wheel. Moreover, $\mathbf{J}(\theta)$ is given by

$$\boldsymbol{J}(\theta)=\frac{1}{4}\begin{bmatrix}\sqrt{2}\sin(\theta_1) & \sqrt{2}\cos(\theta_1) & \sqrt{2}\cos(\theta_1) & \sqrt{2}\sin(\theta_1)\\ -\sqrt{2}\cos(\theta_1) & \sqrt{2}\sin(\theta_1) & \sqrt{2}\sin(\theta_1) & -\sqrt{2}\cos(\theta_1)\\ -\frac{1}{L+l} & \frac{1}{L+l} & -\frac{1}{L+l} & \frac{1}{L+l}\end{bmatrix} \quad (2)$$

where $\theta_1=\theta+\pi/4$, L and l respectively denote the length shown in Figure 3. By using the pseudo inverse matrix, $\mathbf{J}^+(\theta)$, of the matrix $\mathbf{J}(\theta)$ where $\mathbf{J}(\theta)\mathbf{J}^+(\theta)=\mathrm{I}_3$, one expresses the inverse kinematics model of the robot by

$$\boldsymbol{\upsilon}_w=\mathbf{J}^+(\theta)\begin{bmatrix}\dot{x}_w\\ \dot{y}_w\\ \dot{\theta}\end{bmatrix} \quad (3)$$

where

$$\mathbf{J}^+(\theta)=\begin{bmatrix}\sqrt{2}\sin(\theta_1) & -\sqrt{2}\cos(\theta_1) & -(l+L)\\ \sqrt{2}\cos(\theta_1) & \sqrt{2}\sin(\theta_1) & (l+L)\\ \sqrt{2}\cos(\theta_1) & \sqrt{2}\sin(\theta_1) & -(l+L)\\ \sqrt{2}\sin(\theta_1) & -\sqrt{2}\cos(\theta_1) & (l+L)\end{bmatrix}$$

The forward model (1) of the experimental MWOR is used to accomplish odometry for the existing FastSLAM2.0 method, while the inverse kinematic model is exploited to achieve motion control.

C. Kinematic Motion Control

This subsection will recall the kinematic control method of the MWOR for tracking any smooth differentiable trajectory $\begin{bmatrix}x_d(t) & y_d(t) & \theta_d(t)\end{bmatrix}^T\in C^1$. To this end, define the following tracking error vector

$$\begin{bmatrix}x_e(t)\\ y_e(t)\\ \theta_e(t)\end{bmatrix}=\begin{bmatrix}x_w(t)\\ y_w(t)\\ \theta(t)\end{bmatrix}-\begin{bmatrix}x_d(t)\\ y_d(t)\\ \theta_d(t)\end{bmatrix} \quad (4)$$

Taking the time derivative of (4) and using (2) yield obtains

$$\begin{bmatrix}\dot{x}_e(t)\\ \dot{y}_e(t)\\ \dot{\theta}_e(t)\end{bmatrix}=\begin{bmatrix}\dot{x}(t)\\ \dot{y}(t)\\ \dot{\theta}(t)\end{bmatrix}-\begin{bmatrix}\dot{x}_d(t)\\ \dot{y}_d(t)\\ \dot{\theta}_d(t)\end{bmatrix}=\mathbf{J}(\theta(t))\begin{bmatrix}r\omega_1(t)\\ r\omega_2(t)\\ r\omega_3(t)\\ r\omega_4(t)\end{bmatrix}-\begin{bmatrix}\dot{x}_d(t)\\ \dot{y}_d(t)\\ \dot{\theta}_d(t)\end{bmatrix} \quad (5)$$

Hence, the kinematic motion control law is proposed as below.

$$\begin{bmatrix}\omega_1(t)\\ \omega_2(t)\\ \omega_3(t)\\ \omega_4(t)\end{bmatrix}=\frac{1}{r}\boldsymbol{J}^+(\theta(t))\left(-\boldsymbol{K_p}\begin{bmatrix}x_e(t)\\ y_e(t)\\ \theta_e(t)\end{bmatrix}-\boldsymbol{K_I}\begin{bmatrix}\int_0^t x_e(\tau)d\tau\\ \int_0^t y_e(\tau)d\tau\\ \int_0^t \theta_e(\tau)d\tau\end{bmatrix}+\begin{bmatrix}\dot{x}_d(t)\\ \dot{y}_d(t)\\ \dot{\theta}_d(t)\end{bmatrix}\right) \quad (6)$$

where the two gain matrices, K_P and K_I, are symmetric and positive-definite. Substituting (6) into (5) and using the identity $\mathbf{JJ}^+=\mathbf{I}_3$ leads to the succeeding closed-loop error system

$$\begin{aligned}\begin{bmatrix}\dot{x}_e(t)\\ \dot{y}_e(t)\\ \dot{\theta}_e(t)\end{bmatrix}&=\mathbf{J}\left(\theta(t)\right)r\begin{bmatrix}\omega_1(t)\\ \omega_2(t)\\ \omega_3(t)\\ \omega_4(t)\end{bmatrix}-\begin{bmatrix}\dot{x}_d\\ \dot{y}_d\\ \dot{\theta}_d\end{bmatrix}\\ &=-\mathbf{K_p}\begin{bmatrix}x_e(t)\\ y_e(t)\\ \theta_e(t)\end{bmatrix}-\mathbf{K_I}\begin{bmatrix}\int_0^t x_e(\tau)d\tau\\ \int_0^t y_e(\tau)d\tau\\ \int_0^t \theta_e(\tau)d\tau\end{bmatrix}\end{aligned} \quad (7)$$

where the globally asymptotical stability of the closed-loop error system can be easily proven by selecting the subsequent quadratic Lyapunov function

$$\begin{aligned}V_3(t)&=\frac{1}{2}\begin{bmatrix}x_e(t) & y_e(t) & \theta_e(t)\end{bmatrix}\begin{bmatrix}x_e(t)\\ y_e(t)\\ \theta_e(t)\end{bmatrix}\\ &+\frac{1}{2}\begin{bmatrix}\int_0^t x_e(\tau)d\tau & \int_0^t y_e(\tau)d\tau & \int_0^t \theta_e(\tau)d\tau\end{bmatrix}\boldsymbol{K_I}\begin{bmatrix}\int_0^t x_e(\tau)d\tau\\ \int_0^t y_e(\tau)d\tau\\ \int_0^t \theta_e(\tau)d\tau\end{bmatrix}\end{aligned} \quad (8)$$

III. FastSLAM 2.0

This section briefly recalls how to localize the MWOR and construct its 2D map by using FastSLAM2.0 with a 360-degree Lidar sensor mounted atop the MWOR. This approach is used to sample the path via particle filters and include the kinematic model of the MWOR. Each particle will be attached to its own map, consisting of N extended Kalman filters. Figure 6 depicts the working principle of FastSLAM 2.0 that uses a particle filter (PF) to sample the robot trajectory, and Figure 7 illustrates the basic flowchart of FastSLAM algorithm for each particle, where the procedure includes the retrieval of a robot pose from the previous particle set, new pose prediction, updating of new observed features, and calculation of the important weights from new particles. Worthy of mention is that the accuracy of the existing FastSLAM2.0 is improved by incorporating with the kinematic model of the experimental MWOR with a two-dimensional LiDAR (light detection and ranging) and a camera. Although there still exist accumulating errors caused by the used odometry method even with the MWOR kinematic model, pose estimation errors of the FastSLAM 2.0 algorithm are substantially reduced by fusing the readings from the LiDAR.

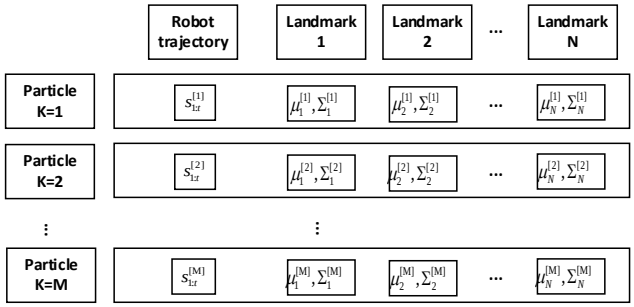


Figure 6. PFs in FastSLAM is used to compute the robot path, where the EKF is used to compute mean and covariance of each landmark.

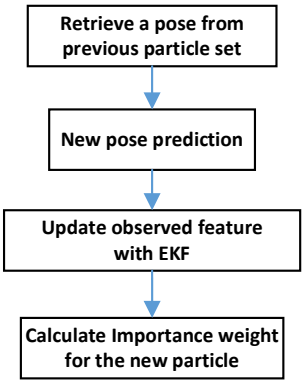


Figure7. Flowchart of the used FastSLAM 2.0 algorithm for each particle.

TABLE I
THE PSEUDO CODE OF DIJKSTRA'S ALGORITHM.

```
1. Function Dijkstra (Graph, source):
2.     for each vertex in Graph:
3.         dist[v] := infinity;
4.         previous[v] := undefined
5.     end for
6.     dist[source] := 0;
7.     Q:=the set of all nodes in Graph
8.     while Q is not empty:
9.         u := vertex in Q with smallest distance in dist[];
10.        remove u from Q;
11.        if dist[u]=infinity;
12.            Break
13.        end if
14.        for each neighbor v of u
15.            alt := dist[u]+dist_between (u,v);
16.            if alt < dist[v]:
17.                dist[v] := alt;
18.                previous[v] := u;
19.                decrease-key v in Q;
20.            end if
21.        end for
22.    end while
23.    return dist
24. End Function
```

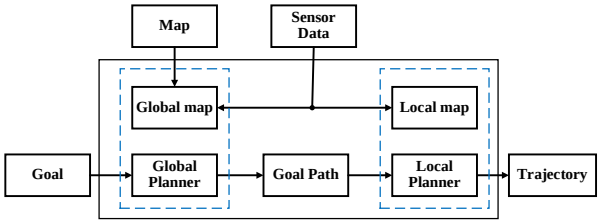


Figure 8. Flowchart of the proposed path planning method.

IV. GLOBAL PATH PLANNING

A. Dijkstra Algorithm

Using Dijkstra algorithm as a global path planning method will enable the MWOR to carry out global path planning easily. Once the map of the working space around the MWOR has been obtained, the map can be utilized to search for the best optimal path. Table 1 illustrates the pseudocode of the existing Dijkstra algorithm. When the goal is commanded, the global planner based on the Dijkstra algorithm will be used to produce a global path trajectory which will not hit any obstacles in the global cost map based on the shortest path. Furthermore, Figure 8 shows the flowchart of the proposed path planning method. As can be seen in Figure 8, once the global path has been found, the local path will be done by the local path planner as discussed in the sequel.

V. ENVIRONMENT RECOGNITION AND OBSTALE AVOIDANCE USING DYNAMIC WINDOW APPROACH AND SEGNET

A. Introduction to SegNet

SegNet has an encoder network and also has a corresponding decoder network, followed by a final pixelwise classification layer. The architecture is illustrated in Figure 9 The encoder network makes up of 13 convolutional layers which correspond to the first 13 convolutional layers in the VGG16 [5] network designed for object classification.Therefore, we can initialize the training process fromweights trained for classification on large datasets. Also we can discard the entirly connected layers in favour of retaining higher resolution feature maps at the deepest encoder output. This also reduces the number of parameters in the SegNet encoder network significantly as compared to other recent architectures [6]. Every encoder layer has a corresponding decoder layer. Hence, the decoder network has 13 layers. The last decoder output is fed to a multi-class soft-max classifier in order to produce class probabilities for every pixel independently.

Every encoder in each encoder network shows convolution with a filter bank to produce a set of feature maps. These are then batch normalized [7-8].Then an element-wise rectifiedlinear non-linearity (ReLU) max is applied. Following that, max-pooling with a 2 × 2 window and stride 2 is performed and the resulting output is sub-sampled by a factor of 2. Max-pooling is used to achieve translation invariance over small spatial shifts in the input image. Sub-sampling results in a large input image context for each pixel in the feature map. While several layers of max-pooling and

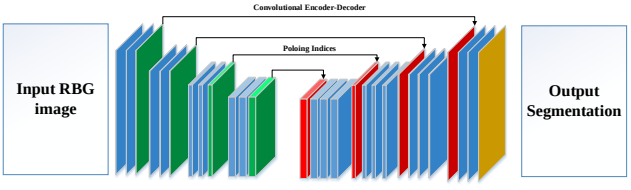


Figure 9. Architecture of SegNet.

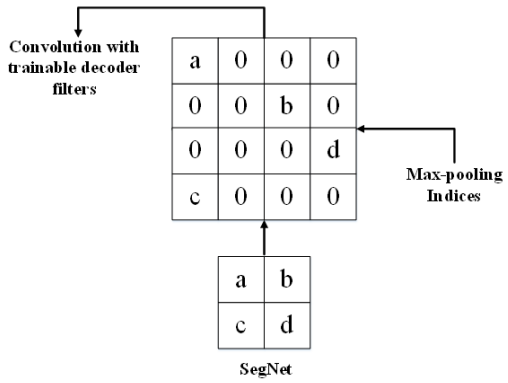


Figure 10. Decoders of the SegNet.

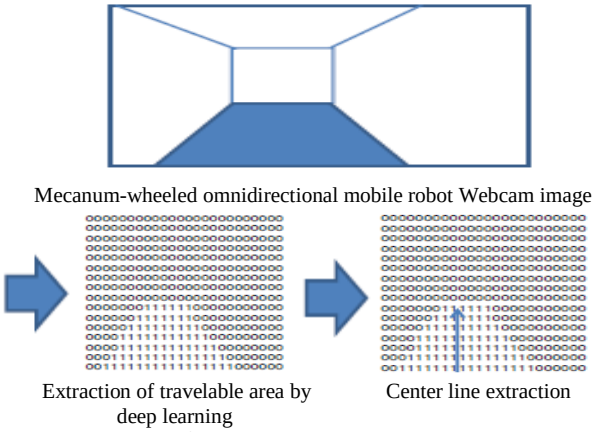


Figure 11. Flowchart of the image recognition.

sub-sampling can achieve more translation invariance for robust classification correspondingly. There is a loss of spatial resolution of the feature maps. This increasingly lossy image representation is not beneficial for segmentation where boundary delineation is vital. Hence, it is necessary to catch and store boundary information in the encoder feature maps before sub-sampling is expressed. If memory during inference is not constrained, all the encoder feature maps can be stored well-done. The map is sometime not the case in practical applications. Hence, we propose a efficient way to store this information. It involves storing only the max-pooling indices,i.e, the locations of the maximum feature value in each pooling window is memorized for each encoder feature map. In principle,the map can be done using 2 bits for each 2×2 pooling window.Therefore, it is more efficient to store as compared to memorizing feature maps in float precision. This can lower memory storage results in a slight loss of accuracy.However, it is still suitable for practical applications.

This appropriate decoder in the decoder network upsamples its input feature maps using the memorized max-pooling indices from the corresponding encoder feature maps. The step produces sparse feature maps [9]. The SegNet decoding technique is illustrated in Figure 10. These feature maps are then convolved with a trainable decoder filter bank to produce dense feature maps.A batch normalization step is then applied to each of these maps.Note that the decoder corresponding to the first encoder produces a multi-channel feature map, although its encoder input has 3 channels (RGB). It is not like the other decoders in the network which produce feature maps with the same number of size and channels as their encoder inputs.The huge dimensional feature representation at the output of the final decoder is fed to a trainable soft-max classifier. The soft-max classifies every pixel independently. The output of the soft-max classifier is a K channel image of probabilities where K is the number of classes. The predicted segmentation corresponds to the class with maximum probability at every pixel [10].

B. Environment Recognition

Semantic segmentation is based on image recognition, except the classifications occur at the pixel level as opposed to classifying entire images as with image recognition. This is accomplished by convolutionalizing a pre-trained image recognition model (like Alexnet), which turns it into a fully-convolutional segmentation model capable of per-pixel labelling. Useful for environmental sensing and collision avoidance, segmentation yields dense per-pixel classification of many different potential objects per scene, including scene foregrounds and backgrounds.

C. Obtacle Avoidance Using Dynamic Window Approach

After the global path has been planned by the global planner, the local planner will start up obstacle avoidance at any time based on the sensing data. The DWA (Dynamic Window Approach) method based on the LiDAR data is used for the local planner. The basic idea of DWA is delineated in the following steps.

1. Discretely sample robot's control space (dx,dy,dtheta).
2. For each sampled velocity, perform forward simulation for the robot to see the result of the sampled velocity applied for a period of time.
3. Evaluate each trajectory from the result of forward simulation, using a metric that includes properties such as: proximity to obstacles, proximity to goal, and proximity to the global path and discard those collide with obstacles. The cost function is defined as follows:

$$\begin{aligned} cost = {} & path\ distance\ bias * \\ & (distance\ to\ path\ \ from\ the\ endpoint\ of\ \ the\ trajectory) \\ & + goal\ distance\ bias * \\ & (distance\ to\ local\ goal\ \ from\ the\ endpoint\ of\ \ hte\ trajectory) \\ & + occdist\ scale * \\ & (The\ closer\ to\ the\ obstacle\ the\ higher\ of\ \ this\ cost) \end{aligned}$$

4. Pick the highest-score velocity and send it to the robot.
5. Rinse and repeat.

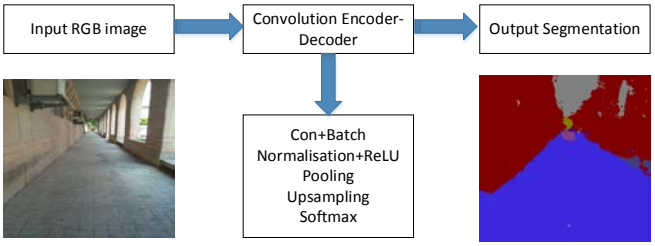
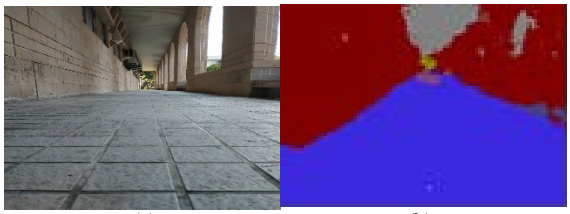


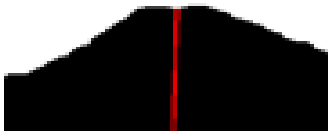
Figure 12. Image segmentation process of SegNet.

(1) Find the center point in the width direction in the travelable area
(2) Find a straight line with the least error from the center point. (Least squares method)

Figure 13. Centerline calculation.



(a) (b)



(c)

Figure 14. Segmentation results using SegNet and centerline detection. (a) An original picture. (b) Segmentation results using SegNet. (c) Detected the centerline of travelling area.

D. Obstacle Avoidance Using SegNet

This subsection will briefly describe the approach to avoiding any static and dynamic obstacles by using SegNet. The self-driving procedure to prevent any collisions from obstacles is described in the following six steps. First, the image recognition is done by using the flowchart shown in Figure11. As depicted in Figure 12, the webcam images are processed and extracted by SegNet via the Jetson TX2 board, in order to find a travelable area. Second, calculate the determined centerline. Figure13 shows how to find the centerline (locations represented as a "1" in Figure 13). The centerline is extracted from the travelable area using the least squares method by finding the centers of the widths of the travelable area; afterwards, a straight line is then obtained along these centers with the least error, as shown in Figure 14(c). Third, calculate the orientation deviation between the centerline and the current orientation of the mobile robot. Forth, generate the motion commands. After the positions of the tip of the centerline and the front of the MWOR are mapped to two-dimensional space, the deviation between the centerline and the direction in which the mobile robot is facing is calculated by the Jetson TX2 board, and send the feedback commands to mobile robot so as to reduce the deviation from the centerline. Fifth, calculates the current position of the robot via the Intel RealSense D435i camera and location system and LiDAR. Sixth, generates motion control commands to activate the four servomotors for autonomous navigation.

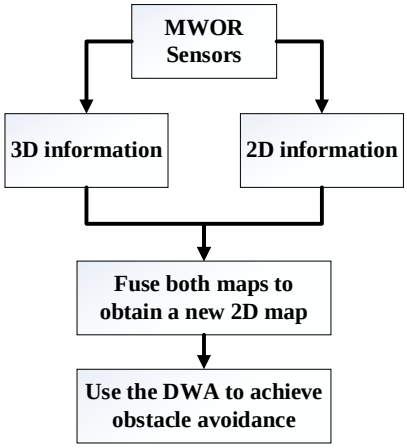


Figure 15. Flowchart of the fused obstacle avoidance.

E. Fused Obstacle Avoidance

Since the MWOR uses both on-board sensors to create 2D and 3D map, both 2D and 3D environmental mapping can be combined together to acquire a new 2D map. Based on the newly modified map, the DWA approach is exploited to achieve obstacle avoidance. Figure 15 shows the conceptual idea of the fused obstacle avoidance scheme.

VI. Experimental Results and Discussion

This section will show experimental results to show the effectiveness of the proposed navigation method. The first experiment is aimed to test the experimental MWOR under ROS 1.0. Figure 16 displays the experimental environment, and Figure17 shows the environmental map created by using the FastSLAM 2.0 algorithm, showing the success of the system integration for the experimental MWOR.

The second experiment is performed to examine the effectiveness of the proposed autonomous navigation method. Figure 18 depicts the experimental results of the proposed navigation method without any unexpected obstacles, showing the feasibility of the proposed method. Figure 19 shows the experimental results of the steering MWOR using DWA and SegNet avoidance in the environment, where an unexpected people was regarded as a static obstacle. The result in Figure 10 indicated that the fused obstacle avoidance method worked well. Figure 20 displays the experimental results of the environmental objects by using SegNet. The results in Figure 20 reveals that the SegNet successfully recognized different objects of the working space. Worthy of mention is that the SegNet has to work in an environment with good illumination.

Figure 16. Experimental environment.

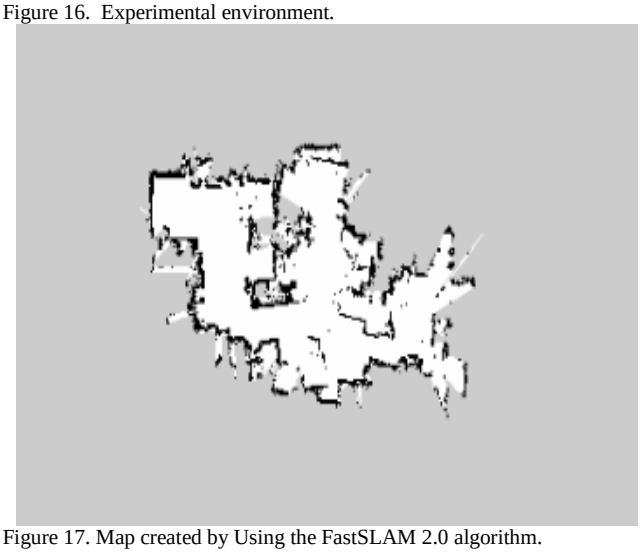


Figure 17. Map created by Using the FastSLAM 2.0 algorithm.

Figure 18. Experimental pictures of the autonomous navigation methods when the MWOR encountered with static obstacles.

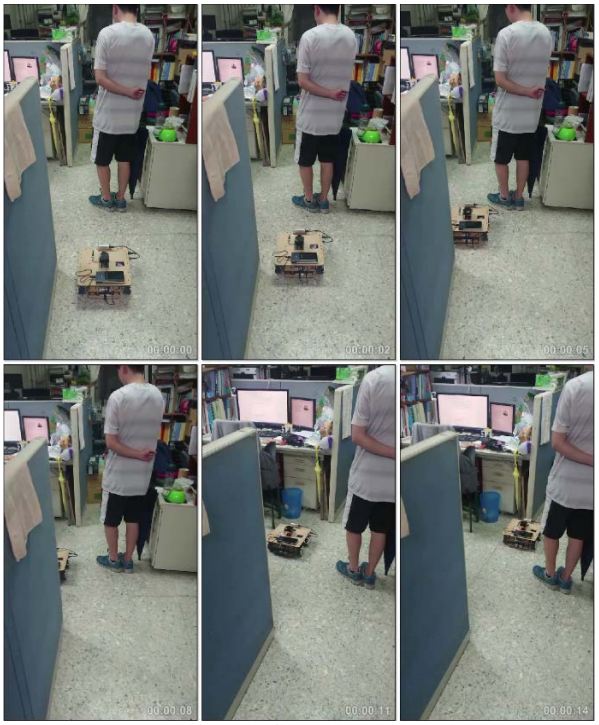


Figure 19. Experimental pictures of the autonomous navigation methods when the MWOR encountered with a moving people.

Figure 20. Experimental pictures of the used SegNet that recognized some objects in the environment.

VII. Conclusions and Future Work

The paper has presented an autonomous navigation method using SegNet to steer the WMOR from one starting place to another in an indoor environment. The built map and SegNet have been systematically used to find the shortest global and local paths. During the autonomous navigation stage, the FastSLAM 2.0 algorithm under ROS has been implemented to find the poses of the MWOR. The SegNet has been utilized to carry out objects recognition in the environment. The dynamic window approach (DWA) and SegNet have been combined together to find a safe local path for obstacle avoidance. Through experimental results, the proposed approach has been shown capable of enabling the MWOR to carry out autonomous navigation usefully and efficiently. An interesting research topic for future work would be to install an onboard 3D RGBD camera on the MWOR, in order to obtain useful external environment information.

References

[1] C. C. Yu, C. F. Hsu, and F. C. Tai, "A laboratory course on mobile robotics education, "*iRobotics*, vol. 1, no.4, pp.37-43, December 2018..

[2] J. Tang, J. Li, and X. Xu, "Segnet-based gland segmentation from colon cancer histology images," in *Proc. of 2018 33rd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, Nanjing, 2018, pp. 1078-1082.

[3] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," *in Proc. of IEEE International Conference on Robotics and Automation*, Atlanta, GA, USA, 1993, pp. 802-807 vol.2. doi: 10.1109/ROBOT.1993.291936.

[4] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes," in *Proc. of 2016 Int. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 3234-3243.

[5] S. Isobe and S. Arai, "Inference with model uncertainty on indoor scene for semantic segmentation," in *Proc. 2017 IEEE Global Conf. on Signal and Information Processing (GlobalSIP)*, Montreal, QC, 2017, pp. 1170-1174.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[7] J. Long ,E.Shelhamer,and T. Darrell,"Fully convolutional networks for semantic segmention,"in CVPR, pp.343331-3440,2015.

[8] S. loffe and C. Szegedy,"Batch normailization :Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[9] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481-2495, Dec. 1 2017.

[10] D. Naik and C. D. Jaidhar, "Image segmentation using encoder-decoder architecture and region consistency activation," *2016 11th International Conference on Industrial and Information Systems (ICIIS)*, Roorkee, 2016, pp. 724-729.

[11] S. Isobe and S. Arai, "Inference with model uncertainty on indoor scene for semantic segmentation," *2017 IEEE Global Conf. on Signal and Inform. Processing (GlobalSIP)*, Montreal, QC, 2017, pp. 1170-1174.

[12] L. Reger, "Securely connected vehicles - what it takes to make self-driving cars a reality," *2016 21th IEEE European Test Symposium (ETS)*, Amsterdam, 2016, pp. 1-1.

Po-An Wei he is currently pursuing Master's degree in Department of Electrical Engineering from National Chung Hsing University, Taichung, Taiwan, ROC. His current research interests include Mecanum robot, exploration, cooperative 3D SLAM and their applications for the robot autonomous navigation.

Ching-Chih Tsai received the Diplomat in Electrical Engineering from National Taipei Institute of Technology, Taipei, Taiwan, ROC, the MS degree in Control Engineering from National Chiao Tung University, Hsinchu, Taiwan, ROC and the Ph.D degree in Electrical Engineering from Northwestern University, Evanston, IL, USA, in 1981, 1986 and 1991, respectively. Currently, he is currently a Distinguished Professor in the Department of Electrical Engineering, National Chung-Hsing University, Taichung, Taiwan, where he served the Chairman in the Department of Electrical Engineering from 2012 to 2014. He is a Fellow of IEEE, IET and CACS.

Dr. Tsai served as the Chair, Taipei Chapter, IEEE Control Systems Society, from 2000 to 2003, and the Chair, Taipei Chapter, IEEE Robotics and Automation Society from 2005 to 2006. In 2007, he was the program chair of 2007 CACS international automatic conference sponsored by Taipei chapter, IEEE control systems society. In 2010, he served as the program co-chair of SICE 2010 annual conference in Taiwan, which was technically sponsored by IEEE CSS; in 2011, he served as the General Chair, 2011 International conference on service and interactive robotics; in 2012, he has served as the General Chair, 2012 International conference on Fuzzy Theory and Its Applications, the General Chair, 2012-2015 CACS International Automatic Control Conferences, and the General Chair, 2016-2017 International Conference on Advanced Robotics and Intelligent Systems. Dr. Tsai served the two-term President, Chinese Institute of Engineers in Central Taiwan, Taiwan from 2007 to 2011, and two-term President of Chinese Automatic Control Society from 2012 to 2015. Since 2008, he has been the Executive Directors in Boards of Government of thee professional associations, including Robotic Society of Taiwan, Taiwan Fuzzy Systems Association, and Taiwan Systems Association. He has served as the Chair, Taichung Chapter, IEEE Systems, Man, and Cybernetics Society since 2009, the Chair of IEEE SMC Technical Committee on intelligent learning in control systems since 2009, the President of Robotics Society of Taiwan since 2016, the steering committee of Asian Control Association since 2014, a BOG member of IEEE Nanotechnology council since 2012, the Vice President of International Fuzzy Systems Association since 2015, and a BOG member of the IEEE SMCS since 2017.

Dr. Tsai has published more than 500 technical papers, and seven patents in the fields of control theory, systems technology and applications. Web of Science has indexed his paper entitled "Adaptive Neural Network Control of a Self-Balancing Two-Wheeled Scooter" in the category Automation Control Systems, where the paper was ranked 408 out of 37607 articles (published between 2010 to 2014). Dr. Tsai is respectively the recipients of the Third Society Prize Paper Award from IEEE Industry Application Society in 1998, the Outstanding Automatic Control Engineering Award in 2008 from Chinese Automatic Control Society (CACS), and the Outstanding Engineering Professor Award in 2009 from the Chinese Institute of Engineers in 2009, the IEEE Most Active SMC Technical Committee (TC) Award in 2012 from IEEE SMC Society, the Outstanding Electrical Engineering Professor Award from the Chinese Institute of Electrical Engineering in 2014, Outstanding Industry Contribution Award from Taiwan Systems Association in 2016, the best paper award in the International Journal of Fuzzy Systems in 2017, and many best paper awards from many international conferences technically supported by IEEE. He is the advisor, IEEE SMC student branch chapter at National Chung Hsing University; this chapter was the recipient of certificate of appreciation from IEEE SMCS in 2009. He has served as the associate editors of International Journal of Fuzzy Systems, and IEEE Transactions on Systems, Man and Cybernetics: Systems, IEEE

Transactions on Industry Informatics, and International Journal of Electrical Engineering. Recently, he has served as the Editor-in-Chief of a new international robotics journal called “iRobotics”. His current interests include advanced nonlinear control methods, deep model predictive control, fuzzy control, neural-network control, advanced mobile robotics, intelligent service robotics, intelligent mechatronics, intelligent learning control methods with their applications to industrial processes and intelligent machinery.

Self-Piloting of an Indoor Quadrotor Using Deep Reinforcement Learning

Hsiu-Chen Tsai and Ching-Chih Tsai, *Fellow*, *IEEE*

***Abstract*-This paper presents a self-piloting method using deep reinforcement learning (DRL) for an indoor quadrotor flying from one place to another. The self-piloting system is equipped with one ultrasonic sensor, one Intel RealSense depth camera, one Jetson TX2 computing module (Nvidia), and a quadrotor with two cameras. The ultrasonic sensor together with the look-down camera is used to accomplish constant height flight. A deep Q-network (DQN) is employed to proceed with images acquired from the quadrotor and learn motion control commands from these images using end-to-end reinforcement learning, in order to generate motion commands to fly the quadrotor autonomously. The Jetson TX2 module is then utilized to implement the improved DQN network. Simulations and experimental results are conducted to show the effectiveness and merit of the proposed self-piloting method.**

***Index Terms*—Deep Q-network (DQN), deep reinforcement learning (DRL), quadrotor, self-piloting**

I. INTRODUCTION

Nowadays, self-piloting is an important issue for flying mobile robots including UAVs. There are various topics related to this issue, such as robot control for UAVs or vehicles with applications to forests, deserts, streets or indoor environments, and etc. There are a lot of problems needed to be solved in each type of environment. Among many deep learning methods, deep reinforcement learning algorithms, such as DNN, DRL and/or DQN, have been shown to provide outstanding solutions for self-piloting of autonomous UAVs flying in unknown or complicated environments.

Several past research results with different settings have obtained distinct technical contributions for autonomous navigation or self-piloting of UAVs. For example, Giusti et al. [1] studied the problem of perceiving forest or mountain trails from a single monocular image acquired from the viewpoint of a robot traveling on the trail itself, and Wang et al. [2] modeled autonomous navigation of an UAV in a large-scale unknown complex environment as a discrete-time continuous control problem and solved it using deep reinforcement learning. Moreover, Mnih et al. [3] used deep reinforcement learning with Q learning, called deep Q-network or DQN, successfully in achieving outstanding human-level operations on the Atari games. About localization and navigation in GPS-denied environment, Zhang et al. [4] proposed a vision-based localization method for an indoor, small-size quadrotor that using parallel tracking and mapping algorithm (PTAM algorithm) with the onboard camera. Furthermore, Mirowski et al. [5] created a reinforcement learning method for learning to navigate from raw sensory input in complicated 3D mazes, approaching human-level performance even under conditions where the goal location changes frequently.

As authors' best understanding, there are no studies of using DRL and DQN for indoor self-piloting. This motivates us to apply the DQN algorithm to address the problem that the quadrotor can autonomously navigate any indoor GPS-denied environment.

The objectives of the paper are to propose a self-piloting method using the improved DQN algorithm for a quadrotor flying from one place to another in a GPS-denied and unknown indoor environment, and to verify the applicability of the improved DQN algorithm by conducting simulations and one experiment. The improved DQN algorithm will be implemented on a real quadrotor to process the images acquired from the front camera of the quadrotor and then generate motion control commands to fly autonomously. The presented contents are written in two principal contributions. One is the proposal of the self-piloting method using the improved DQN algorithm, and the other is effectiveness verification of the proposed method using three simulations in a powerful computer with one 1080-TI GPU, and one real experimental testing via a real quadrotor with one on-board NVIDIA Jetson TX2 AI computing module. The constructed techniques would provide useful references for professionals working for unmanned aerial vehicles and/or quadrotors.

The rest of this paper is organized as follows. Section II briefly describes the system architecture of the self-piloting indoor quadrotor. Section III details the improved DQN structure and algorithm for the self-piloting quadrotor. In Section IV, three simulations are conducted for illustration of the effectiveness and superiority of the proposed approach in Section IV. Section V presents and discusses the experimental results to show the applicability of the proposed method. Section VI concludes the paper.

II. SYSTEM STRUCTURE AND DESCRIPTION OF AUTONOMOUS INDOOR QUADROTOR

This section is aimed to describe the system structure of the proposed system by including the system configuration of the experimental indoor quadrotor under the ROS environment. The ROS will be used as a software framework for the system, where each sensor or module will be registered as a node and communicate with each other via the master of the ROS. The autonomous driving of the quadrotor will be implemented in the ROS environment.

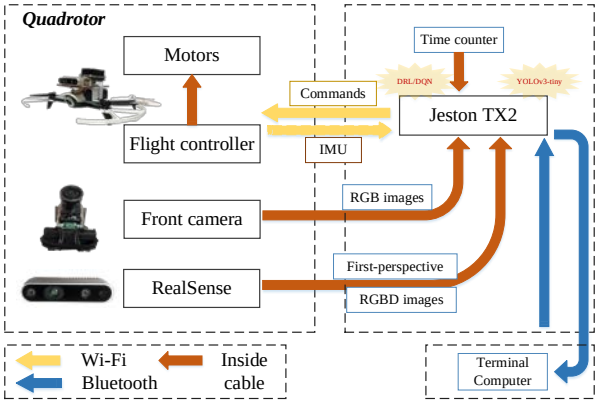


Figure 1. System Structure of the autonomous indoor quadrotor.

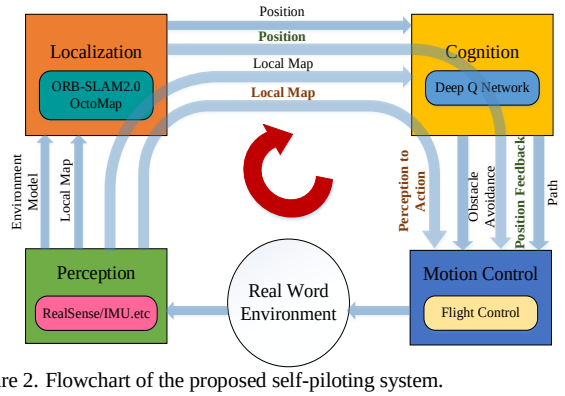


Figure 2. Flowchart of the proposed self-piloting system.

Figure 3. Physical picture of the experimental autonomous quadrotor: (a) top view of the complete experimental set-up; (b) exploded illustration of NVIDIA Jetson TX2 and Intel RealSense depth camera.

2.1 System Structure

Self-piloting in real indoor environments is a challenging problem without the help of GPS. Figure 1 shows the system structure of the autonomous indoor quadrotor, which is Bebop2 from Parrot. In Figure 1, the front camera and installed Intel RealSense depth camera device are used to acquire environmental images in front of the quadrotor, and measure the distances from the quadrotor to its surrounding walls and/or obstacles, respectively. All the readings from both sensors will be passed on to the main controller, which is made by NVIDIA Jetson TX2 AI computing board. The computing board is exploited to execute an improved DQN algorithm with the previous input data, in order to generate flight control commands. The flight controller of the quadrotor receives these commands to carry out required flight motions. Worthy of mention is that the improved DQN method is employed to offline train the quadrotor to fly from one place to another with its environmental perception and the front image camera and on-board Intel RealSense in an unknown indoor environment. Figure 2 shows the flowchart of the general navigation control and obstacle avoidance system.

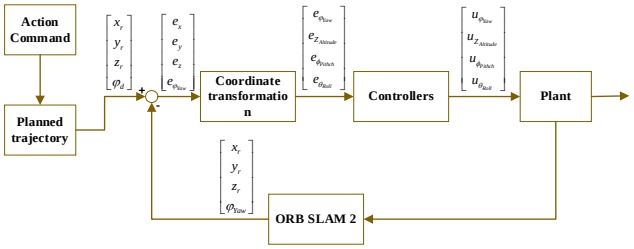


Figure 4. Block diagram of the flight control system.

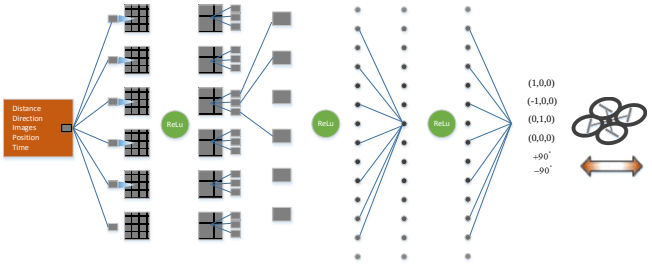


Figure 5. Improved DQN-based control architecture for the autonomous indoor quadrotor.

Figure 3 shows the physical picture of the experimental autonomous quadrotor, Bebop2 from Parrot, with the on-board front camera, AI computing module, light-weight NVIDIA Jetson TX2 board-ACE-N510, and Intel RealSense, model D435i. Inside this type of TX2 board, there exist an integrated 256-core NVIDIA Pascal GPU, a hex-core ARMv8 64-bit CPU complex, and 8GB of LPDDR4 memory with a 128-bit interface, and one dual-core NVIDIA Denver 2 alongside a quad-core ARM Cortex-A57. Worthy of mention is that this TX2 board is particularly useful in accelerating cutting-edge deep neural-network (DNN) architectures using the cuDNN and ensorRT libraries, which support for recurrent neural networks (RNNs), long-short-term-memory networks (LSTMs), and online reinforcement learning. On the other hand, being a depth camera, the Intel RealSense obtains 85° sensing data around the quadrotor and also supports the USB interface for the use of ROS.

2.2 Flight Control of the Experimental Quadrotor

This subsection is devoted to briefly describing the flight control system of the experimental quadrotor. Figure 3 depicts the block diagram of the quadrotor flight control system, whose working principle is explained as follows. First of all, the pose tracking error $[e_x \ e_y \ e_z \ e_{\varphi_{Yaw}}]$ is found by computing the desired and current trajectories of the quadrotor. Afterwards, the pose error vector, $[e_x \ e_y \ e_z]^T$, is then transformed via the following inverse coordinate transformation in (1), in which the resultant Euler error angle vector , $[e_\phi \ e_\theta \ e_\varphi]^T$ is then computed.

TABLE I
CNN STRUCTURE WITHOUT POOLING LAYER [3].

Layer	Input	Filter size	Stride	Num filters	Activation	Output
conv1	84x84x4	8x8	4	32	ReLU	20x20x32
conv2	20x20x32	4x4	2	64	ReLU	9x9x64
conv3	9x9x64	3x3	1	64	ReLU	7x7x64
fc4	7x7x64			512	ReLU	512
fc5	512			18	Linear	18

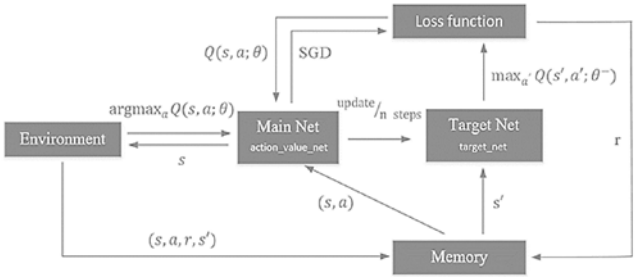


Figure 6. Fixed Q-target structure.

$$\begin{bmatrix} e_\phi \\ e_\theta \\ e_\varphi \end{bmatrix} = \begin{bmatrix} 1 & S_\phi T_\theta & C_\phi T_\theta \\ 0 & C_\phi & -S_\phi \\ 0 & S_\phi / C_\theta & C_\phi / C_\theta \end{bmatrix} \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} \quad (1)$$

where ϕ is the pitch angle, θ the roll angle, and φ the yaw angle; S_ϕ=sin(ϕ); C_ϕ=cos(ϕ); C_θ=cos(θ); T_θ=tan(θ). Finally, once the Euler error angles of the quadrotor have been calculated, the control command signals are obtained from the four proportional altitude and attitude controllers, thus achieving closed-loop flight control. To achieve autonomous navigation, it is necessary to find the current position of the quadrotor by using a new Oriented Fast and Rotated BRIEF-Simultaneous Location and Mapping (ORB-SLAM2) algorithm.

The quadrotor is controlled to fly at a fixed speed and height, and move along the yaw direction, x and y frames. Note that one ultrasonic sensor and one overlook camera are used by the quadrotor to accomplish the fixed altitude control. Since action commands will be generated by the DQN algorithm, we provide six commands with their trajectory generators, which are "go forward for one meter", "go right for one meter", "go left for one meter", "stop", "turn left", and "turn

III. IMPROVED DQN CONTROLLER

This section will describe the improved deep Q-network (DQN) controller, which generates the six commands. With the ORB-SLAM2, six commands can be done by the flight control system as shown in Figure 4. The aforementioned six commands are then generated according to the incoming data from all the on-board sensors. Figure 5 shows the schematics of the improved DQN-based control architecture for the self-piloting indoor quadrotor, where the DQN consists of two-layer convolutional neural networks with the rectified linear activation functions (ReLUs) and two-layer fully connected neural networks for generating these six flight commands. In what follows begins with a brief introduction to DQN and then proposes the improved DQN with new rewards settings.

3.1 Brief Introduction to DQN

Q Learning is indeed a model-free reinforcement learning technique, which find the action-value function Q through experiences with the working environment. Once the Q value has been obtained, the Q learning method will select desired action that gives the biggest expected reward, thereby achieving the optimal policy specified by designers. It is worthwhile to note that the deep Q learning is about using deep learning techniques to represent the Q table.

Table I shows the CNN structure with pooling layer in the DQN. The CNN structure is composed of three convolutional layers and two fully connected NNs with their input sizes, filter sizes, strides, and numbers of used filters, activation functions and output sizes. To match the input sizes of the CNN structure, all the incoming images have to be scaled down to the sizes of 84×84 and then converted into 8-bit grayscale ones at the outset. Next, those found 84×84×4 tensors are inputted to the CNN, which will have one output for each action, namely that a corresponding Q value will be generated by the CNN for each possible action. Afterwards, the real Q values will be iteratively obtained by continuously proceeding with the CNN. Learning the parameter vector, θ, which presents all the weights of the Q network, is the goal of the DQN. Having learned vector θ, the network will approximate the optimal value function Q^*, as shown in (2).

$$Q(s,a,\theta) \approx Q^*(s,a) \quad (2)$$

where S is the current state, a is a possible action to take at the current state. With the Bellman equation, (3) is given.

$$Q(s,a,\theta) = r + \gamma \max_{a'} Q^*(s',a') \approx Q^*(s,a) \quad (3)$$

where r is the reward, γ is the discount factor, s' and a' are respectively the expected state and action. Now, the goal is to find the optimization on minimizing the mean-squared error loss, which is the loss function.

Due to the CNN being a regression model, the loss function is chosen as the following squared error loss function,

$$L = (Q^*(s,a) - Q(s,a,\theta))^2 \approx (r + \gamma \max_a Q(s',a',\theta) - Q(s,a,\theta))^2 \quad (4)$$

where $r + \gamma \max_a Q(s',a',\theta)$ is the target to be maximized, and $Q(s,a;\theta)$ is the current Q value. The Q function, $Q(s,a)$, is defined as maximum expected discounted reward of future that will take an action a on state s, and choose actions by the policy derived from Q. Usually the Q function can be obtained by using the following iterative Bellman equation.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[\gamma \max_{a'}(Q(s',a')) - Q(s,a)] \quad (5)$$

where α denote the learning rate. In (5), the estimated Q value is usually wrong in the beginning, but after a period of time, if the experience from the environment will give a correct reward r, then the Q value will iteratively turn out to become correct one.

TABLE II. PSEUDOCODE OF ALGORITHM 1.

Algorithm 1: DQN with experience replay.

1. Initialize replay memory M to capacity C
2. Initialize acwtion-value function Q with random weights vector θ
3. Initialize target action-value function Q^* with weights $\theta^- \leftarrow \theta$
4. **For** episode = 1, E **do**
5. Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
6. **For** t=1,T **do**
7. With probability ε select a random action a_t
8. Otherwise select $a_t = \arg\max_a Q(\phi(s_t), a; \theta)$
9. Execute action a_t in emulator and observe reward r_t and image x_t'
10. Set $s_t' = s_t, a_t, x_t'$ and preprocess $\phi_t' = \phi(s_t')$
11. Store transition $(\phi_t, a_t, r_t, \phi_t')$ in M
12. Set $y_t = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
13. Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters θ
14. Every N steps rest $\hat{Q} = Q$.
15. **End For**
16. **End For**

3.2 DQN with Fixed Q-target and Experience Replay

This subsection is aimed to delineate the DQN with fixed Q-target and prioritized experience replay (PER). Figure 6 shows the system structure of the DQN with fixed Q-target. The concept of the fixed Q-target was introduced by Google DeepMind team, who desired to estimate the real temporal difference (TD) target. Via the Bellman equation, the team found that the TD target is the reward after taking the action at the state plus the discounted highest Q value for the next state. $Q(s,a)$ is the Q target to be computed in (6)

$$Q(s,a) = r(s,a) + \gamma \max_a Q(s',a) \tag{6}$$

where $r(s,a)$ is the reward of taking that action at that state and $\gamma \max_a Q(s',a)$ is the discounted maximized Q value among all possible actions from the next state. To this end, the TD error, $R + \gamma \max_a Q(s',a,\theta)) - \hat{Q}(s,a,\theta)$, is calculated by finding the difference between the Q-target and estimated $\hat{Q}$. Afterwards, the change in parameter, $\Delta\theta$, is obtained from (7)

$$\Delta\theta = \alpha[(R + \gamma \max_a Q(s',a,\theta)) - \hat{Q}(s,a,\theta)]\nabla_\theta \hat{Q}(s,a,\theta) \tag{7}$$

where is gradient of the current predicted Q-value. As in (6), the same weights are used to estimate the target and Q value such that the changing TD target and weights are related. Hence, below are two ideas for the fixed Q-target:

1. Use a separate network with a fixed weight to estimate the TD target.
2. After several (T) steps, the weights are copied from DQN network to update the target network.

After several (T) steps, the fixed parameters, θ^- ,will be updated with θ from the main network in (8)

$$\Delta\theta = \alpha[(R + \gamma \max_a Q(s',a,\theta^-)) - \hat{Q}(s,a,\theta)]\nabla_\theta \hat{Q}(s,a,\theta) \tag{8}$$

Thus, the fixed Q-target makes learning more stable due to the target function staying fixed for a while.

Q-learning with experience replay is an off-policy offline learning method that can be learned using the previous experience. During offline training, some experiences may be more important than others, but occur less frequently. Therefore, the PER method [5] aims to take in priority experience which has the big difference between the prediction and TD target. The actions, rewards and states of the PER will be stored into its memories and then some of them will randomly be joined to the q-target network. The advantages of experience replay hinges on the fact of reducing the relevance between experiences.

Another important issue is to define rewards. Depending on the situations, the quadrotor will receive different rewards and penalties to learn more its correct behavior. For example, crashed penalty, obstacle penalty, time penalty, direction reward, pixel reward, arrived reward, and so on. Each reward or penalty has its own way of calculation, and some may be constant values, and some may vary from state to state. The proposed DQN algorithm with experience replay, called Algorithm 1, is shown in Table 2.

IV. SIMULATIONS AND DISCUSSION

This section will conduct three simulations to show the effectiveness of the proposed improved DQN algorithm with experience replay using Unreal Engine and AirSim. Unreal Engine, which has been widely used for computer games or scenario creations, is utilized to build the three simulation environments. AirSim [5], developed by Microsoft, is here exploited to provide application programming interfaces to interact with the unmanned quadrotor, in order to retrieve images, get current states, control the vehicles or robots and so on. The AirSim modules for these simulations were coded by using well-known Python and C++ programming languages. The improved DQN algorithm was also implemented using Python. Figure 6 depicts the indoor three simulation environments and DQN-controlled quadrotor constructed by employing the Unreal Engine and AirSim.

The first simulation is performed to examine the effectiveness of the proposed improved DQN algorithm to fly the indoor quadrotor from its starting point to a desired ending location in a simple indoor environment as Figure 7(a) shows. Figure 7(a) displays the simulation results and motion pictures during the simulation. The results in Figure 7(a) reveal that the DQN-controlled quadrotor flies autonomously from the beginning point to the desired position.

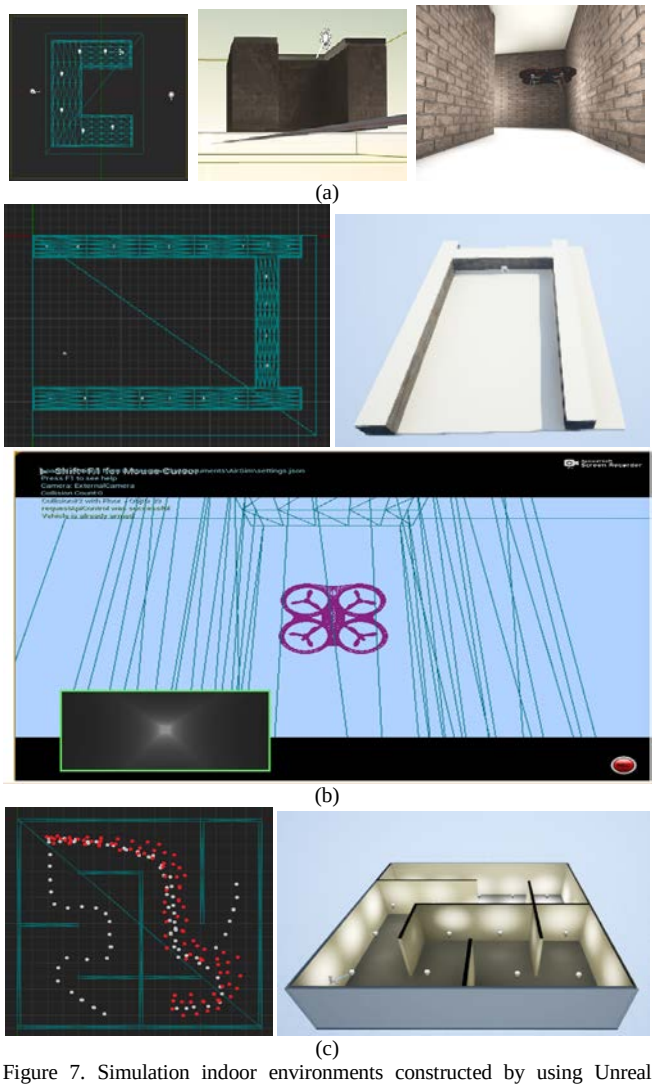


Figure 7. Simulation indoor environments constructed by using Unreal Engine.

The second simulation is conducted to validate the proposed improved DQN algorithm in a slightly complicated environment as drawn in Figure 6 (b) by comparing to the first simulation. Figure 6(b) illustrates the simulation settings, in which the total length of the flight routine is 3000 meters, its width is 2050 meters and height is 2.7 meters. The bottom picture in Figure 7(b) illustrates one glimpse of the simulation done for the environment, showing that the self-piloting system works as predicted.

Unlike the previous two simulations, the third one is particularly carried out to verify the proposed algorithm in a maze environment as displayed in Figure 7(c). Figure 7(c) describes the simulation results, thus ensuring the merit of the proposed method in this more complex maze environment. As can be observed in Figure 7(c), the results confirm the capability of the proposed algorithm, where the gray points denote the trajectories of the quadrotor during iterations, and the red points represent the quadrotor's trajectories in the last three iterations.

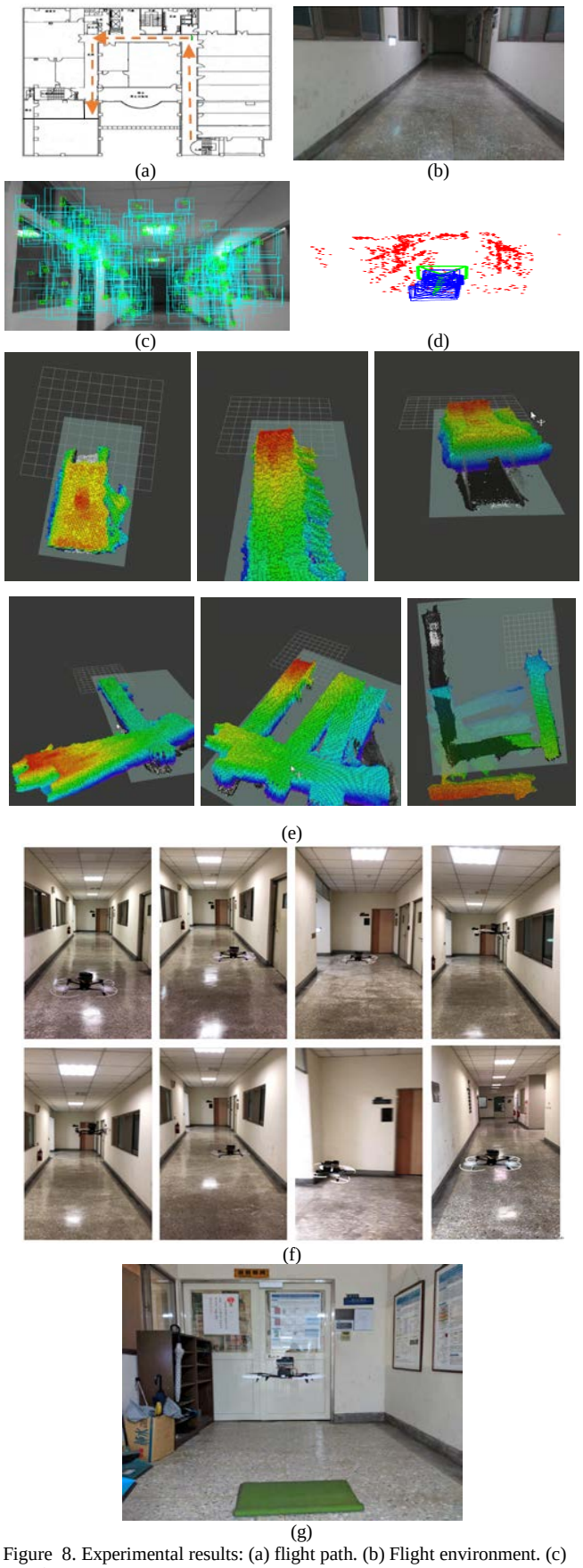


Figure 8. Experimental results: (a) flight path. (b) Flight environment. (c) Key features using ORB-SLAM2.0. (d) 3D cloud-point map and actual flight trajectories. (e) OctoMap construction process. (f) Experimental pictures during self-piloting. (g) Destination marked by a mat.

V. Experimnetal Results and Discussion

This section is aimed to present and discuss the experimental results of the experimental quadrotor using the improved DQN algorithm by conducting one experiment in a GPS-denied indoor environment. The experimental environment is almost identical to the second simulation environment. Figure 8(a) depicts the experimental environment and the expected flight path, which is highlighted by the orange line. During the motion, the image from the front camera of the quadrotor is displayed in Figure 8(b). Figs. 8(c) and (d) show the ORB-SLAM 2.0. Figure 8(e) shows the construction process of the OctoMap experimental results obtained from the on-board TX2 AI computing module, thereby showing the implementation success of the overall system integration. As shown in Figure 8(f), the quadrotor flied with all the previous described devices. Through the results in Figs. 8(f) and (g), the proposed DQN controller was shown to autonomously fly online the quadrotor from the starting point to the destination position marked with the special mat.

VI. Conclusions and Future Work

This paper has presented a self-piloting method using an improved DQN algorithm for an indoor quadrotor flying from one place to another. The experimental quadrotor has been equipped with its on-board front camera, Intel RealSense depth camera and NVIDIA Jetson TX2 AI computing board. The improved DQN algorithm has been proposed by including designated rewards and penalties, and then implemented and executed in order to process the images acquired from the front camera of the quadrotor and then generate motion control commands to fly autonomously. Simulations via AirSim and Unreal Engine have been conducted to show the feasibility and effectiveness of the proposed DQN-based autonomous driving method. Through the experimental results, proposed DQN-based autonomous driving method has been shown capable of navigating the quadrotor to reach its destination in one floor of a building in real time. An interesting future work world be to control the quadrotor to fly from one floor to another by passing through staircases.

References

[1] A. Giusti, J. Guzzi, D. C. Cire¸san, Fang-Lin He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, “A machine learning approach to visual perception of forest trails for mobile robots,” *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661-667, July 2016.

[2] C. Wang, J. Wang, X. Zhang, and X. Zhang, "Autonomous navigation of UAV in large-scale unknown complex environment with deep reinforcement learning," *2017 IEEE Global Conf. on Signal and Inform. Processing (GlobalSIP)*, Montreal, QC, pp. 858-862, 2017.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. *518.* pp. 529–533, 26 February 2015.

[4] X. Zhang, B. Xian, B. Zhao, and Y. Zhang, "Autonomous Flight Control of a Nano Quadrotor Helicopter in a GPS-Denied Environment Using On-Board Vision," *IEEE Trans. Industrial Electronics*, vol. 62, no. 10, Oct. 2015.

[5] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, “Learning to Navigate in Complex Environments,” *ICLR 2017*, arXiv:1611.03673, Jan. 2017

[6] T. Schaul, J. Quan, L. Antonoglou, and D. Silver, “Prioritized Experience Replay,” *ICLR 2016*, *a*rXiv:1511.05952, Feb. 2016.

Hsiu-Chen Tsai

Ching-Chih Tsai received the Diplomat in Electrical Engineering from National Taipei Institute of Technology, Taipei, Taiwan, ROC, the MS degree in Control Engineering from National Chiao Tung University, Hsinchu, Taiwan, ROC and the Ph.D degree in Electrical Engineering from Northwestern University, Evanston, IL, USA, in 1981, 1986 and 1991, respectively. Currently, he is currently a Distinguished Professor in the Department of Electrical Engineering, National Chung-Hsing University, Taichung, Taiwan, where he served the Chairman in the Department of Electrical Engineering from 2012 to 2014. He is a Fellow of IEEE, IET and CACS.

Dr. Tsai served as the Chair, Taipei Chapter, IEEE Control Systems Society, from 2000 to 2003, and the Chair, Taipei Chapter, IEEE Robotics and Automation Society from 2005 to 2006. In 2007, he was the program chair of 2007 CACS international automatic conference sponsored by Taipei chapter, IEEE control systems society. In 2010, he served as the program co-chair of SICE 2010 annual conference in Taiwan, which was technically sponsored by IEEE CSS; in 2011, he served as the General Chair, 2011 International conference on service and interactive robotics; in 2012, he has served as the General Chair, 2012 International conference on Fuzzy Theory and Its Applications, the General Chair, 2012-2015 CACS International Automatic Control Conferences, and the General Chair, 2016-2017 International Conference on Advanced Robotics and Intelligent Systems. Dr. Tsai served the two-term President, Chinese Institute of Engineers in Central Taiwan, Taiwan from 2007 to 2011, and two-term President of Chinese Automatic Control Society from 2012 to 2015. Since 2008, he has been the Executive Directors in Boards of Government of thee professional associations, including Robotic Society of Taiwan, Taiwan Fuzzy Systems Association, and Taiwan Systems Association. He has served as the Chair, Taichung Chapter, IEEE Systems, Man, and Cybernetics Society since 2009, the Chair of IEEE SMC Technical Committee on intelligent learning in control systems since 2009, the President of Robotics Society of Taiwan since 2016, the steering committee of Asian Control Association since 2014, a BOG member of IEEE Nanotechnology council since 2012, the Vice President of International Fuzzy Systems Association since 2015, and a BOG member of the IEEE SMCS since 2017.

Dr. Tsai has published more than 500 technical papers, and seven patents in the fields of control theory, systems technology and applications. Web of Science has indexed his paper entitled "Adaptive Neural Network Control of a Self-Balancing Two-Wheeled Scooter" in the category Automation Control Systems, where the paper was ranked 408 out of 37607 articles (published between 2010 to 2014). Dr. Tsai is respectively the recipients of the Third Society Prize Paper Award from IEEE Industry Application Society in 1998, the Outstanding Automatic Control Engineering Award in 2008 from Chinese Automatic Control Society (CACS), and the Outstanding Engineering Professor Award in 2009 from the Chinese Institute of Engineers in 2009, the IEEE Most Active SMC Technical Committee (TC) Award in 2012 from IEEE SMC Society, the Outstanding Electrical Engineering Professor Award from the Chinese Institute of Electrical Engineering in 2014, Outstanding Industry Contribution Award from Taiwan Systems Association in 2016, the best paper award in the International Journal of Fuzzy Systems in 2017, and many best paper awards from many international conferences technically supported by IEEE. He is the advisor, IEEE SMC student branch chapter at National Chung Hsing University; this chapter was the recipient of certificate of appreciation from IEEE SMCS in 2009. He has served as the

associate editors of International Journal of Fuzzy Systems, and IEEE Transactions on Systems, Man and Cybernetics: Systems, IEEE Transactions on Industry Informatics, and International Journal of Electrical Engineering. Recently, he has served as the Editor-in-Chief of a new international robotics journal called “iRobotics”. His current interests include advanced nonlinear control methods, deep model predictive control, fuzzy control, neural-network control, advanced mobile robotics, intelligent service robotics, intelligent mechatronics, intelligent learning control methods with their applications to industrial processes and intelligent machinery.

Single-Domain Reptile Meta-Tracking

Chi-Yi Tsai*, *Member, IEEE*, and Shang-Jhih Jhang

***Abstract*—Visual tracking has been one of the main topics in computer vision for decades, and it is still a challenging topic. The goal of visual tracking is to continuously locate a specific target in a predefined bounding box throughout an incoming video stream or a sequence of images. Typically, this issue requires tracking algorithms to recognize and locate the target with robustness against a variety of uncertainties such as appearance changes, illuminance changes, and image blurring, etc. This requirement produces some unique challenges especially for some tracking algorithms based on deep learning techniques that require online learning during the tracking process. Although deep learning methods provide really strong and robust feature representation, they are easy to be over-fitted if given a really small set of training data and thus making the overall performance throughout tracking poorly. To deal with this issue, this paper presents a novel deep-learning based meta-tracker, which adopts a first-order meta-learning technique so that during online initialization, the visual tracker only requires few training samples and a few steps of optimization to perform well in online tracking. Experiment results show that the proposed method outperforms eight state-of-the-art deep visual trackers and achieves up to 66.4% of average success rate on OTB2015 dataset using one-pass evaluation.**

***Index Terms*—Deep learning, deep visual tracking, Reptile meta-learning, few-shot learning, single-domain neural network.**

I. Introduction

DESPITE visual tracking has been one of the main topics in computer vision for decades, it is still a very challenging topic. When a visual tracker receives an initial location of the target in the first frame of a video sequence, its main objective is to locate the location of the target in all the remaining frames of the video sequence. This goal creates a unique challenge for visual tracking algorithms because they have to perform online learning to adapt to target changes such as appearance changes, illuminance changes, motion blurring, and partial/full occlusion, etc. In recent years, deep learning methods become urgency due to their great capacity in applications of object classification [1-3], image segmentation [4-5], and image recognition [6], etc. So far, there have been several visual tracking algorithms that incorporate the deep learning technique to improve tracking robustness and tracking accuracy. In this paper, we focus our attention on the current deep learning based visual tracking algorithms, in which we examine them into two categories according to their online tracking strategies.

The first category of the convolutional neural network (CNN) trackers needs to fine-tune the neural network or to update the target classifier in the tracking stage. For instance, Wang et al. proposed a fully convolutional network based tracker (FCNT) [7], which consists of a general network (GNet) and a specific network (SNet). The GNet captures the category information of the target using feature maps from top CNN layers, while the SNet discriminates the target from the background with similar appearance using feature maps from lower ones. To avoid the background noise introduced during tracking the target, they fixed GNet and only updated SNet after the initialization in the first frame. In [8], Hong et al. proposed an online visual tracking method, which trains a CNN to produce a discriminative saliency map and uses a support vector machine (SVM) to perform online learning of discriminative target appearance models. In [9], Danelljan et al. investigated the impact of convolutional features for visual tracking problem, and they found that the convolutional features provide improved results compared to standard handcrafted features. Based on this observation, they proposed to use activations from the CNN layer in the discriminative correlation filter (DCF) framework [10], which updates the correlation filter based on a linear interpolation rule. In [11], Nam and Han proposed a multi-domain neural network (MDNet) learning method, which trains a general CNN model during offline learning while using binary regularization heavily during online learning. However, the processing speed of the MDNet is very slow because it uses three CNN layers to produce 512 feature maps for each selected image patch.

On the other hand, the second category of the CNN trackers ignores fine-tuning the neural network in the tracking stage by using the Siamese neural network (SNN), which is a two-input and multiple-output neural network architecture that inputs two images individually into two CNNs shared with the same weights to extract feature maps of both images. The main advantage of using the SNN is that the trained SNN does not need online fine-tuning at test time, making its processing speed very fast. Several studies already use large-scale datasets to train the SNN model for object tracking applications [12-14]. In [12], Held et al. proposed a generic object tracking using regression networks (GOTURN) based on the SNN architecture. The GOTURN tracker

This work was supported by the Ministry of Science and Technology of Taiwan under Grant MOST 107-2218-E-032-004 and MOST 107-2221-E-032-049.

The authors are with the Department of Electrical and Computer Engineering, Tamkang University, No. 151, Yingzhuan Road, Tamsui District, New Taipei City 251, Taiwan R.O.C. (the corresponding author's e-mail: chiyi_tsai@mail.tku.edu.tw).

produces target bounding box simply by regression and is the first neural network based generic object tracker achieving real-time performance about 100 frames per second (fps). In [13], Bertinetto et al. proposed a fully-convolutional SNN with a cross-correlation layer without online learning operation. The cross-correlation layer uses convolution operators to compute cross-correlation between feature maps of the reference image and of the query image. The position of tracking target is then determined by the position of the maximum score relative to the center of the output score map. In [14], Tao et al. proposed a Siamese instance search tracker (SINT), which uses SNN to learn a robust and generic feature embedding network for object tracking, aiming to be invariant to all appearance variations in the robust tracking scenarios. Although the SNN can work well with high processing speed, its tracking performance is still worse than the MDNet because it is without fine-tuning during the tracking process.

Although the deep CNN model provides really strong and robust feature representation for visual tracking, it is easy to be over-fitted since only a single training data from the first frame of the video sequence can be used in the online learning process. To deal with this issue, choosing small network architecture is more suitable for fine-tuning and online-updating the CNN tracker during the online tracking process in order to keep the robustness of the tracker. For instance, the MDNet only uses a small CNN model composed of three CNN layers and three fully connected (FC) layers to deal with the multi-domain visual tracking problem. The MDNet produces state of the art performance; however, the initial online learning process at the first frame of the sequence costs significantly more time than other deep visual tracking methods. In [15], Park and Berg proposed another solution, which combines the model-agnostic meta-learning (MAML) technique [16] and a variety of existing methods to obtain a competitive tracking performance but with less initialization time at the beginning of the sequence. However, the MAML technique requires second-order derivative operations, which greatly increase the computational cost of the offline training process. This issue motivates us to develop a computational efficient meta-tracking architecture based on [11] and [15] while providing robust tracking performance. To achieve this, this paper presents a novel deep-learning based meta-tracker, which adopts a first-order meta-learning technique [17] so that during online initialization, the visual tracker only requires few training samples and a few steps of optimization to perform well in online tracking. Experiment results show that the proposed method outperforms eight state-of-the-art deep visual trackers and achieves up to 66.4% of average success rate on OTB2015 dataset using one-pass evaluation.

The remainder of this paper is organized as follows. Section II introduces the related few-shot meta-learning algorithms. The proposed Reptile meta-tracking algorithm is introduced in Section III. Section IV reports experimental results to evaluate the tracking robustness and tracking accuracy of the proposed algorithm compared to four state-of-the-art deep visual trackers. Section V concludes with the contributions of this paper.

II. Few-Shot Meta-Learning

One of the most struggling problems of the current deep neural nets is the ability to learn different tasks quickly with little data. The main reason behind this problem is that the deep neural net itself and the traditional gradient-based learning process are not designed to reuse precious knowledge learned from other tasks. On the other hand, a human can easily adapt to a wide variety of new and previously unseen tasks with little supervision, this kind of fast adapting ability is now considered the key to achieving the ultimate goal of AI: human level intelligence [18]. Meta-learning, also known as learning to learn, is one way to achieve such a fast adapting ability, which can leverage past experiences to learn faster when presenting a new task, like the goal of generic object tracking. Generally, a meta-learning algorithm has to learn rapidly within each task while accumulating knowledge across different tasks. In order to perform meta-learning, a training dataset needs to be exposed to a large number of tasks.

In few-shot meta-learning problems, a dataset is first separated as a meta-training and a meta-testing set as shown in Figure 1. Each set has a unique collection of classes and no repeated examples between two sets. During meta-learning, the model is trained to learn tasks in the meta-training set. For example, in the context of five-way one-shot learning, every single task contains five training examples, one example for each class, and some test examples to test how well it learned from that five training examples. Hence, the goal of few-shot meta-learning is to train a model that can be adapted to new tasks or new environments fast with little among of data. To solve this problem, the model is first trained in a meta-learning phase on a wide variety of tasks, but each task only contains a few training examples. There are several meta-learning methods proposed in recent years, below are two of the methods related to this work.

A. Model-Agnostic Meta-Learning (MAML)

Considering a meta-learning model f for mapping an input x to an output y, the model is trained to be capable of adapting to huge among of tasks quickly during the training phase of meta-learning. In the K-shot learning problem, each new task T_i is drawn from a distribution over tasks $p(T)$ with only K examples per class drawn from a distribution over initial observations. Then, the model is trained on the task with a loss L_{T_i} generated from the training data in the task T_i. After training, the model is tested on the test data in the task T_i to improve the model f based on the feedback test error. At the meta-testing phase, new tasks are drawn from $p(T)$, which are used to measure how well the model performs after training from K samples per class.

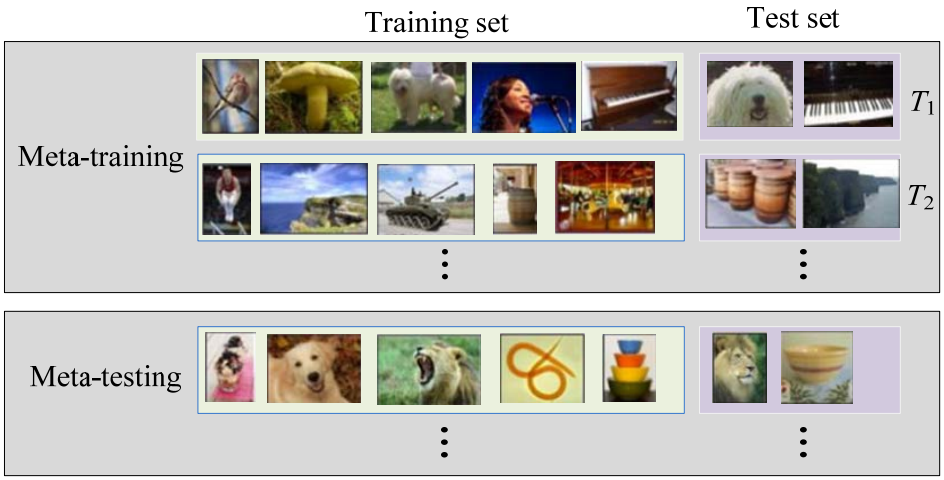


Figure 1. Data setup of few-shot meta-learning.

The purpose of the MAML method is to learn the parameters of any model through meta-learning for capable of fast adapting. Its main idea is to find internal feature representations that are more transferrable to wide arrange of tasks. Unlike previous works, the MAML does not put any constraint on the model architecture itself; instead, the model only needs few update steps to perform well on a new dataset. To find this general purpose feature representations for the model of choice, the MAML method directly optimizes the model on a completely new task using traditional gradient-based learning rules without overfitting.

Algorithm 1. Pseudo Code of the Model-Agnostic Meta-Learning (MAML) [16]

Require : $p(T)$: distribution over tasks
Require : α, β: step size hyperparameters
Randomly initialize θ, the initial parameters of the model
while not done **do**
  Sample batch of tasks $T_i \sim p(T)$
  for all T_i **do**
    Evaluate $\nabla_\theta L_{T_i}(f_\theta)$ with respect to K examples
    Compute adapted parameters with gradient descent : $\tilde{\theta}_i = \theta - \alpha \nabla_\theta L_{T_i}(f_\theta)$
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_i L_{T_i}(f_{\tilde{\theta}_i})$
end while

Algorithm 1 shows the pseudo code of the MAML. Considering a model f_θ that is parameterized by parameters θ. When the model is trained on a new task T_i using gradient descent, the i-th adapted parameters $\tilde{\theta}_i$ is computed using the loss from the task T_i as follows:

$$\tilde{\theta}_i = \theta - \alpha \nabla_\theta L_{T_i}(f_\theta), \tag{1}$$

where α is the step size, $L_{T_i}(f_\theta)$ is the loss function associated with the model f_θ given the task T_i that only contains few examples per class, and $\nabla_\theta L_{T_i}(f_\theta)$ is the corresponding gradient vector with respect to (w.r.t.) the current parameters θ. When computed the adapted parameters of all tasks, the model parameters are then trained using a total loss function

$$\sum_i L_{T_i}(f_{\tilde{\theta}_i}), \tag{2}$$

which measures how well the updated model $f_{\tilde{\theta}_i}$ performs on the test data in tasks sampled from $p(T)$. The goal is to minimize the total loss function w.r.t. θ such that with a small number of update step (1) the model is able to achieve maximum effectiveness on the task. According to the total loss function (2), the meta-learning update rule to improve the model's fast adapting ability can be written as:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_i L_{T_i}(f_{\tilde{\theta}_i}), \tag{3}$$

where β is the step size, and $\nabla_\theta \sum_i L_{T_i}(f_{\tilde{\theta}_i})$ is the gradient vector of the total loss function w.r.t. to θ through stochastic gradient descent (SGD).

B. Reptile

The main drawback of the MAML algorithm is that the optimization process involves second-order derivatives. Specifically, when calculating the derivatives of (2), the derivatives of $\nabla_\theta L_{T_i}(f_\theta)$ w.r.t. θ will also need to be calculated. This operation not only consumes a lot of additional memory but also slows down the overall optimization speed. To address this issue, Alex Nichol et al. proposed Reptile algorithm [17], which simplifies the MAML to improve computational efficiency. Similar to MAML, the Reptile also aims to train a network initialization that can be adapted to new tasks quickly using only little training data and a small number of gradient steps. At each meta-training iteration, it first samples a task from the training dataset, trains on it using mini-batch gradient descent for few steps, and moves the initialization towards the trained weights on that task using a simple update method. After training on many meta-training iterations, it is able to obtain a general

weight initialization that is close to the optimal weights for many tasks.

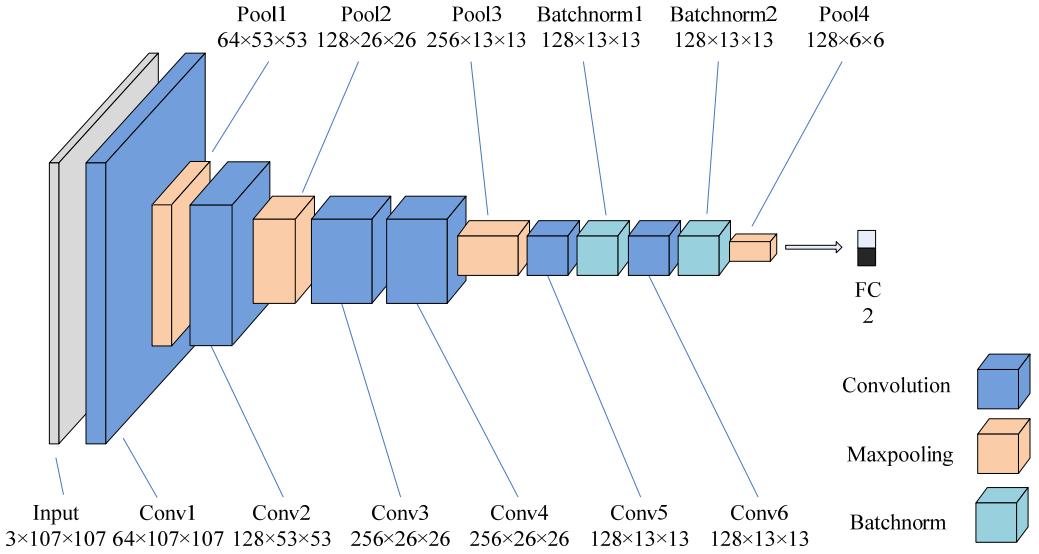


Figure 2. Neural network architecture of the proposed CNN tracker.

Algorithm 2. Pseudo Code of the Reptile in Serial Version

Require : $p(T)$: distribution over tasks
Require : ε: step size hyperparameter
Require : k: iteration number of gradient descent
Randomly initialize θ, the initial parameters of the model
for iteration $i = 1,2,3,\ldots$ do
  Sample a task $T_i \sim p(T)$
  Compute $\widetilde{\theta}_i = U_{T_i}^k(\theta)$, denoting k steps of SGD or Adam
  Update $\theta \leftarrow \theta + \varepsilon(\widetilde{\theta}_i - \theta)$, ε is a real number
end for

Algorithm 2 shows the pseudo code of the serial version of Reptile, which also aims to learn a general initialization for parameters of a neural network model to make the model having the ability to generalize quickly from the new task. Let $U_T^k(\theta)$ denote the operator that updates θ k times using gradient descent or Adam [19] on batches of data sampled from T. At the i-th iteration, the task-adaptation parameters $\widetilde{\theta}_i$ defined in Eq. (1) can then be computed using the update operator such that

$$\theta \leftarrow \theta + \varepsilon(\widetilde{\theta}_i - \theta), \tag{5}$$

where ε is a real number. However, the Reptile can be extended into a parallel or batch version. Instead of only sampling a single task at each iteration, we can evaluate n tasks at each iteration to update the model parameters θ using

$$\theta \leftarrow \theta + \varepsilon \frac{1}{n} \sum_{i}^{n} (\widetilde{\theta}_i - \theta), \tag{6}$$

where n is a nonzero positive number.

III. The Proposed Reptile Meta-Tracking

In this section, we introduce the proposed meta-tracking algorithm, which is developed based on the MDNet method but with different offline learning and online learning process inspired by the Reptile meta-learning algorithm.

A. The Proposed Neural Network Architecture

Figure 2 illustrates the neural network architecture of the proposed CNN tracker, which has a total of seven layers, including six CNN layers (Conv1-Conv6) and one fully-connected (FC) layer at the last layer. The proposed network receives a 107×107 RGB image patch as input. The first four layers are identical to the first four layers of VGG11 [6] with pre-trained parameters on ImageNet dataset [20]. The following two layers are CNN layers, each of them with 128 filters, batch normalization, and ReLU activation function. The 6th convolutional layer has a max-pooling operation, and the last layer is an FC layer with two output units that classify the input image patch belonging to the target or the background class. We trained the proposed CNN tracker using cross-entropy loss. Unlike the network architecture in MDNet, it does not have multi-domain branches in the last layer. Therefore, the proposed architecture is a single-domain network.

B. Parameters Update during Offline Reptile Meta-Learning

The goal of offline meta-learning is to learn a generic feature representation of targets that can be easily adapted to a new task or domain with only a few updates and few training examples. During training, the parameters of the first four CNN layers (Conv1-Conv4) remain fixed while the parameters of the last two CNN layers (Conv5-Conv6) and the FC layer are meta-trained.

Denote w_j as the parameters of the model in the j-th layer. We first randomly initialize the parameters w_5 to w_7. The parameters w_1 to w_4 are pre-trained on the ImageNet dataset and remain fixed throughout the offline meta-training as well

Algorithm 3. Pseudo Code of the Proposed Reptile Meta-Learning

Require : Ω_D: training dataset
Require : ε: step size hyperparameter
Require : k: iteration number of gradient descent
Import w_1 to w_4 from pre-trained model
Randomly initialize the parameters w_5 to w_7 as the symbol θ
for iteration i = 1,2,3,⋯ do
  Randomly sample a task T_i from the training dataset Ω_D
  Compute $\tilde{\theta}_i = U_{T_i}^k(\theta)$, denoting k steps of SGD
  Update $\theta \leftarrow \theta + \varepsilon(\tilde{\theta}_i - \theta)$
end for
Output the general model θ^* for online meta-tracking

as online tracking. We denote the trainable parameters w_5 to w_7 as the symbol θ. At the i-th iteration, we sample a video sequence from the training dataset Ω_D and randomly sample a total of 64 positive sample regions and 192 negative sample regions to form a training task T_i. The positive samples have an intersection of union (IoU) overlap ratio with ground truth bounding boxes equal to or greater than 0.7, while the negative samples have the IoU overlap ratio with the ground truth bounding boxes equal to or less than 0.5. After the i-th training task T_i is sampled, we perform k steps of SGD on the parameters θ with the training task T_i as input, resulting in new parameters $\tilde{\theta}_i$ according to Eq. (4), then we update θ using Eq. (5). We repeat this training procedure until a predefined iteration is reached. After the offline meta-training, we obtain general model parameters θ^* that can be used during online tracking. The pseudo-code of the proposed Reptile meta-learning algorithm is shown in **Algorithm 3**.

C. Parameters Update during Online Reptile Meta-Tracking

1) Target estimation

In the online tracking procedure, the general model parameters θ^* is updated using the new video sequence. Similar to MDNet, the tracker estimates the target at frame t by first sampling N target candidates $\{\mathbf{x}_t^1, \mathbf{x}_t^2, \ldots, \mathbf{x}_t^N\}$ around the estimated target location from the previous frame t–1, then evaluates them using the fine-tuned model f_θ to obtain positive score $f_\theta^+(\mathbf{x}_t^n)$ and negative score $f_\varphi^-(\mathbf{x}_t^n)$ for n=1~N. The optimal target state $\mathbf{x}_t^*$ at frame t is estimated by finding the candidate with the maximum positive score such that

$$\mathbf{x}_t^* = \arg\max_{\mathbf{x}_t^n} f_\theta^+(\mathbf{x}_t^n), \tag{7}$$

where the target candidates $\mathbf{x}_t^n = (u_t^n, v_t^n, s_t^n)$ for n=1~N at frame t are sampled in the translation of (u,v) coordinates and scale dimension s of the n-th sample in Gaussian distribution. The standard deviation of the translation dimension (u,v) is 0.3r and the standard deviation of the scale dimension s is 0.5, where r is the mean of width and height of the $\mathbf{x}_{t-1}^*$ in previous frame t–1. The total number of target candidates for every frame is N=256.

2) Parameters update

There are three types of parameters update during online tracking. The first one is the initial update, which samples 32 positive examples with IoU overlap ≥ 0.7 and 96 negative examples with IoU overlap ≤ 0.5 as training data, and performs three steps of SGD on the parameters θ^* using the sampled training data. The second one is the short-term update, which is performed using the positive and negative samples collected for a short period of time when the estimated optimal target state is considered unreliable ($f_\theta^+(\mathbf{x}^*) \leq 0.0$). The third one is the long-term update, which is performed every ten frames using positive samples collected for a long period of time and negative samples collected for a short period of time.

For long-term and short-term update, the tracker holds two frame index sets T_l and T_s, which respectively store 100 and 20 most recent frame indexes of reliable estimated target state ($f_\theta^+(\mathbf{x}^*) > 0.0$). When the frame index t is multiple of ten, we perform a few steps of SGD on θ to compute $\tilde{\theta}$ using Eq. (4) with positive samples collected in long-term and negative samples collected in short-term, then update θ using Eq. (5). If $f_\theta^+(\mathbf{x}^*) \leq 0.0$, then we perform a few steps of SGD on θ to compute $\tilde{\theta}$ using positive samples and negative samples collected in short-term, then overwrite θ directly using $\tilde{\theta}$.

3) The proposed tracking procedure

The full tracking procedure is described as follows. At the start of the sequence, the tracker draws positive samples $S_{t_0}^+$ and negative samples $S_{t_0}^-$ from the first frame and update θ^* using 3 steps of SGD to result in the initialized parameters θ. Then it trains a bounding box regression model [2] using the target bounding box of the first frame. After the initialization, the tracker draws target candidates $\mathbf{x}_t^n$ and estimates the optimal target state $\mathbf{x}_t^*$ using Eq. (7) at frame t. If $f_\theta^+(\mathbf{x}_t^*) > 0.0$, the tracker draws positive samples S_t^+ and negative samples S_t^- for the online learning and adjusts the bounding box of $\mathbf{x}_t^*$ using bounding box regression. Next, we store the frame index t into the frame index sets T_l and T_s, if the number of indexes in T_l and T_s is greater than 100 and 20, respectively, then delete the smallest index in the set.

When the frame index t is multiple of ten, we perform a few steps of SGD on θ using positive samples collected in long-term and negative samples collected in short-term, resulting $\tilde{\theta}$, then update θ using Eq. (5). If $f_\theta^+(\mathbf{x}_t^*) \leq 0.0$, we perform a few steps of SGD on θ using positive samples and negative samples collected in short-term, resulting $\tilde{\theta}$, then overwrite θ directly using $\tilde{\theta}$. The proposed Reptile meta-tracking algorithm is shown in **Algorithm 4**.

Algorithm 4. Pseudo Code of the Proposed Reptile Meta-Tracking

Require : θ^*: general model parameters
Require : ε: step size hyperparameter
Require : k: iteration number of gradient descent
Import model f_θ with trainable parameters θ^* from the offline meta-learning
Draws positive and negative samples $S_{t_0}^+$ and $S_{t_0}^-$ from the first frame t_0
Train a bounding box regression model using the target bounding box of the first frame
Update θ^* using three steps of SGD to θ
Store the first frame index t_0 into both frame index sets T_l and T_s
for frame t = 1,2,3,⋯ do
  Draw target candidate samples $\mathbf{x}_t^n$ for n=1~N with frame t
  Estimate $\mathbf{x}_t^* = \arg\max_{\mathbf{x}_t^n} f_\theta^+(\mathbf{x}_t^n)$
  if $f_\theta^+(\mathbf{x}_t^*) \le 0.0$
    Short-Term Update
    Perform k steps of SGD on θ using candidate samples $S_{T_s}^+$ and $S_{T_s}^-$ to compute $\tilde{\theta}$
    Update $\theta \leftarrow \tilde{\theta}$
  else
    Draw training candidate samples S_t^+ and S_t^-
    Adjust $\mathbf{x}_t^*$ using bounding box regression
    Update the frame index t into both frame index sets T_l and T_s
  if t mod 10 = 0
    Long-Term Update
    Perform k steps of SGD on θ using candidate samples $S_{T_l}^+$ and $S_{T_s}^-$ to compute $\tilde{\theta}$
    Update $\theta \leftarrow \theta + \varepsilon(\tilde{\theta} - \theta)$
end for

IV. Experimental Results

This section shows the experiment results of the proposed method compared to other state-of-the-art methods using the OTB2015 dataset for visual tracking benchmark.

A. Detail Settings

We implemented the proposed algorithm in Ubuntu 16.04 using Python 3.5 as the programming language and Pytorch 0.3.0 as the deep learning framework. For SGD optimization, we used Adam optimizer with learning rate 0.001, the first order momentum β_1=0, and the second-order momentum β_2=0.999. During the initialization in online learning, the tracker samples 32 positive samples with IoU overlap ≥ 0.7 and 96 negative samples with IoU overlap ≤ 0.5. The last three layers of the model are trained for three steps of Adam optimization.

After the initialization, the tracker draws 256 samples around the previous target at each frame and estimates the target bounding box by the bounding box with a maximum positive score. During the short-term update, the algorithm trains the last three layers of the model for five steps using Adam optimization. For the long-term update, the tracker first trains the parameters of last three layers θ of the model for five steps using Adam optimization, resulting in new parameters for the last three layers $\tilde{\theta}$, and then updates θ using Eq.(5) with ε=0.1.

For the OTB2015 experiment, we used a large scale ImageNet detection dataset [20] and VOT dataset [21] as the training dataset. For the VOT dataset, we used the sequences in VOT2013, VOT 2014 and VOT 2015, excluding the video sequences that are also in OTB2015.

B. Performance Evaluation

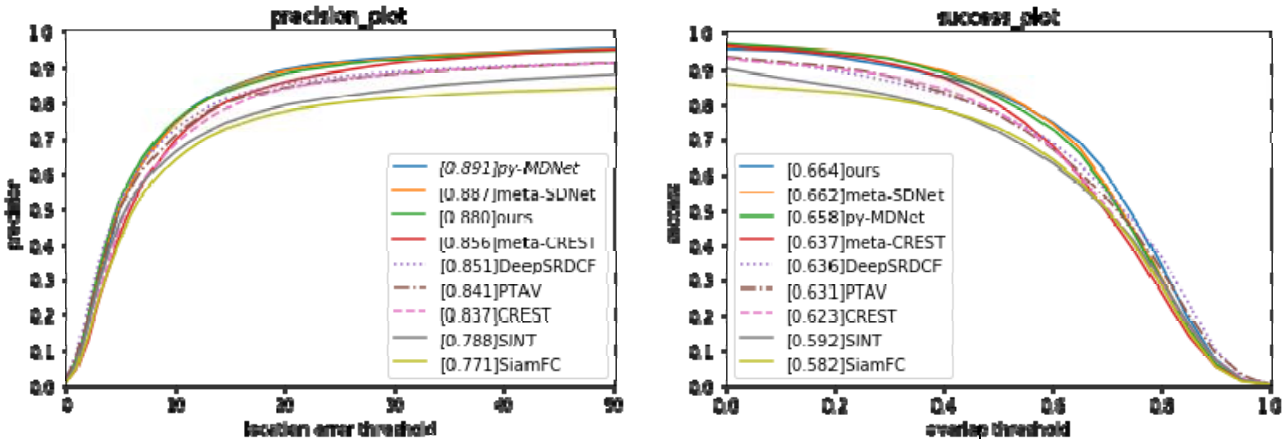
The dataset used for performance evaluation is OTB2015, also known as OTB100 or TB-100. It is a data set with a total of 100 video sequences. Each image in every sequence is annotated with the ground truth of the target bounding box. Each ground truth bounding box includes four values, the top left corner of the bounding box (x, y) as well as its width and height (w, h).

There are two types of evaluation criteria used in OTB2015. The first one is the precision plot, which is defined as the average Euclidean distance between the center location of the estimated targets and ground truth over all the frames in one or multiple sequences. The second one is success plot, which measures the overlap score in each frame using the following equation:

$$Score = \frac{\left| r_t \cap r_g \right|}{\left| r_t \cup r_g \right|}, \tag{8}$$

where r_t and r_g denote the estimated bonding box and ground truth bounding box in each frame, respectively. To measure the success rate (SR) on multiple sequences, the success plot counts the number of frames that have overlap scores higher than certain thresholds, and then divides the number with the total number of frames. This work uses one-pass evaluation (OPE) on the success plots and the precision plots to evaluate the tracking performance.

For the result comparison, we select eight state-of-the-art visual tracking algorithms including DeepSRDCF[9], MDNet[11], SiamFC[13], SINT[14], Meta-SDNet[15], Meta-CREST[15], CREST[22] and PTAV[23]. Figure 3 shows the OTB2015 precision and success plot of the proposed method as well as the other compared methods mentioned above. Both plots in Figure 3 show that the proposed Reptile meta-tracker achieves competitive performance on OTB2015 in terms of tracking precision and tracking success rate. Figure 3(b) show that the proposed Reptile meta-tracker obtains an average SR of 66.4%, outperforming the existing meta-SDNet and meta-CREST tracker by 0.2% and 0.7%, respectively. The proposed method also outperforms the other deep visual trackers.



(a) (b)

Figure 3. Result comparison: (a) Precision plots and (b) Success plots on OTB2015 using OPE.

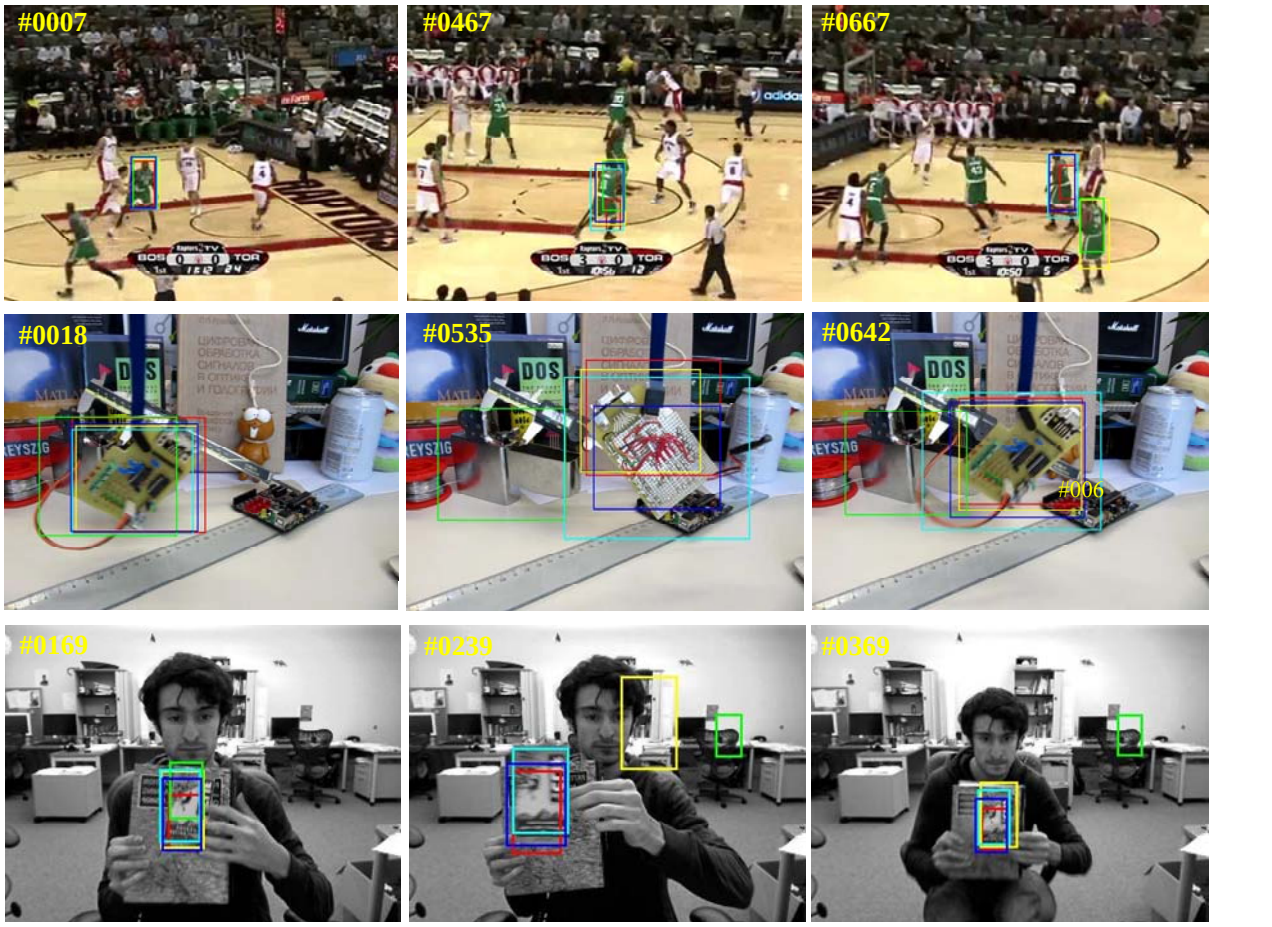


Figure 4. Tracking result for sequence Basketball (Row1), Board (Row2), and ClifBar (Row3).

C. Tracking Results

This section presents some of the most challenge sequences in OTB2015 and compares the tracking results of the proposed method against four state-of-the-art deep visual trackers: DeepSRDCF[9], MDNet[11], SiamFC[13], and Meta-SDNet[15]. Figure 4 shows tracking results of three challenge sequences in background clutter: *Basketball* (Row1), *Board* (Row2), and *ClifBar* (Row3). In the *Basketball* sequence, the target is one of the basketball players wearing the green shirt. During the sequence, there are multiple occasions where the target is running fast, resulting in motion blur effect. There are also several objects in the background that are very similar to the target, which causes some tracking algorithms (e.g., MDNet and SiamFC) to track the wrong object toward the end of the sequence. In the *Board* sequence, the target is a

Figure 5. Tracking result for sequence DragonBaby (Row1), Bird1 (Row2), MotorRolling (Row3), and Skiing (Row4).

motherboard moving around in a cluttered background. During the sequence, the target is rotated for several times or even flipped to the opposite side. This out-of-view issue causes the SiamFC tracker to lose the target while the other compared trackers may not produce an accurate bounding box of the target. In the *ClifBar* sequence, the target is a card with a special texture on it. During the sequence, the target is moved around and rotated by a person's hand. The scale of the target changes dramatically in some parts of the sequence. Moreover, the target is out-of-view for a short time. These issues cause some trackers like SiamFC and MDNet to lose the target. By contrast, the proposed meta-tracker can track and redetect the target successfully.

Figure 5 shows tracking results of four challenge sequences in fast motion and scale variation: *DragonBaby* (Row1), *Bird1* (Row2), *MotorRolling* (Row3), and *Skiing* (Row4). The target of the *DragonBaby* sequence is the head of the baby. During the sequence, the target constantly moves very fast while rotating, causing the SiamFC tracker to lose the target. The target of the *Bird1* sequence is one of the birds in the scene. This sequence is very challenging because the target is occluded for a long time toward the middle of the sequence, causing the search area of all compared trackers to drift away and never recover. On the contrary, the proposed meta-tracker is able to stay in the position until the target re-appears and then tracks the correct target toward the end of the sequence.

The target of the *MotorRolling* sequence is a rolling motorcycle, which constantly rotates in some occasions with a cluttered background. This type of target causes some trackers to either have bad bounding boxes or completely lose the target. However, the proposed meta-tracker is able to track the target accurately throughout the sequence. Next, the target in the *Skiing* sequence is a person skiing down the hill very fast with a large scale variation. During this challenging sequence, the proposed meta-tracker still can track the target throughout the sequence. In contrast, the SiamFC and DeepSRDCF trackers lose the target when it passes through the tree in the background.

Figure 6 shows tracking results of two challenge sequences in fast motion with illumination variation and occlusion: *Ironman* (Row1) and *Matrix* (Row2). The target of the *Ironman*

Figure 6. Tracking result for sequence Ironman (Row1) and Matrix (Row2).

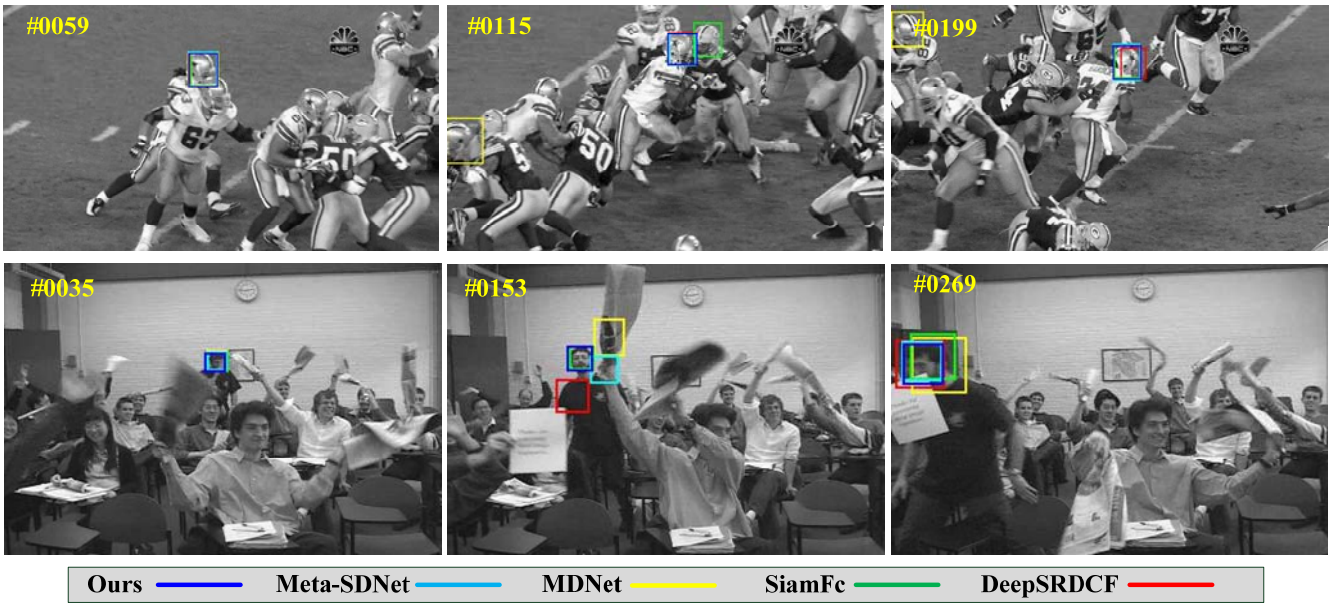


Figure 7. Tracking result for sequence Football (Row1) and Freeman4 (Row2).

sequence is head of the ironman. This sequence is very challenging because the target not only suffers from very poor lighting conditions, but also constantly moves very fast while rotating. In addition, the target is also occluded in some occasions. This causes most of the trackers to finally lose the target including the proposed one. Next, the target of the *Matrix* sequence is the head of a person. During the sequence, the target moves very fast while in a dark lighting condition. The proposed meta-tracker and MDNet are able to track the target most of the time while the other trackers drift away from the target toward the end of the sequence.

Figure 7 shows tracking results of two challenge sequences in rotation and occlusion: *Football* (Row1) and *Freeman4* (Row2). The target in the *Football* sequence is one of the football payers' head. The biggest challenge is that there are many similar objects in the background and very close to the target most of the time. This issue causes some trackers to drift away such as SiamFC and MDNet. By contrast, the proposed meta-tracker can track the target throughout the whole sequence. Finally, the target in the *Freeman4* sequence is the head of a walking person. The other people in the background are waving books while the person is slowly walking through them, causing the target to be partially or completely occluded several times. The proposed meta-tracker is able to track the target without drifting to background throughout the sequence. Therefore, the above experimental results validate the tracking performance and robustness of the proposed meta-tracking method. More experimental results can refer to online web-page [24].

V. Conclusions and Future Work

The proposed meta-tracking method combines a first order meta-learning technique called Reptile into general initialization of a deep visual tracker. Moreover, a novel CNN architecture is also proposed to implement a single-domain CNN tracker. The proposed single-domain meta-tracker performs well on the OTB2015 dataset, which includes many challenging conditions such as occlusion, motion blur, fast motion, rotation, deformation, and background clutters, etc. The proposed meta-tracking method achieves average SR up to 66.4% across all sequence. Compared to eight

state-of-the-art methods, the proposed meta-tracking method can achieve comparable or even better performance in term of tracking precision and average SR given less training data and fewer update steps during online tracking. Experimental results not only validate the tracking performance and robustness of the proposed Reptile meta-tracker, but also show that meta-learning has a great potential to be used in the field of generic object tracking. In the future, the combination of Reptile meta-learning with other deep visual trackers will be further investigated.

ACKNOWLEDGMENT

The authors would like to thank Yen-Chang Feng and Yu-Kai Su of Tamkang University for their participating in experiments.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *International Conference on Neural Information Processing Systems*, Lake Tahoe, Nevada, pp. 1097-1105, 2012.

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

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, USA, pp. 779-788, 2016.

[4] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 39, No. 4, pp. 640-651, 2017.

[5] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 40, No. 4, pp. 834-848, 2018.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations*, San Diego, USA, 2015.

[7] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," *IEEE International Conference on Computer Vision*, Santiago, Chile, pp. 3119-3127, 2015.

[8] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," *International Conference on Machine Learning*, Lille, France, pp. 597-606, 2015.

[9] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," *IEEE International Conference on Computer Vision Workshop*, Santiago, Chile, pp. 621-629, 2015.

[10] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Discriminative scale space tracking," *IEEE Transactions Pattern Analysis and Machine Intelligence*, Vol. 39, No. 8, pp. 1561-1575, 2017.

[11] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, USA, pp. 4293-4302, 2016.

[12] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," *European Conference on Computer Vision*, Amsterdam, the Netherlands, pp. 749-765, 2016.

[13] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional Siamese networks for object tracking," *European Conference on Computer Vision*, Amsterdam, the Netherlands, pp. 850-865, 2016.

[14] R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese instance search for tracking," *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, USA, pp. 1420-1429, 2016.

[15] E. Park and A. C. Berg, "Meta-tracker: Fast and robust online adaptation for visual object trackers." *European Conference on Computer Vision*, Munich, Germany, pp. 587-604, 2018.

[16] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *International Conference on Machine Learning*, Sydney, Australia, pp. 1126-1135, 2017.

[17] A. Nichol and J. Schulman, "Reptile: A scalable meta learning algorithm," arXiv:1803.02999, 2018.

[18] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," Behavioral and Brain Sciences, Vol. 40, pp. 1-72, 2017.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, San Diego, USA, 2015.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "ImageNet: A large-scale hierarchical image database," *IEEE Conference on Computer Vision and Pattern Recognition*, Miami, USA, pp. 248-255, 2009.

[21] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Čehovin, G. Fernández, T. Vojír, G. Häger, et al., "The visual object tracking VOT2015 challenge results," *IEEE Conference on Computer Vision Workshop*, Santiago, Chile, pp. 564-586, 2015.

[22] Y. Song, C. Ma, L. Gong, J. Zhang, R. W.H. Lau, M.-H. Yang, "CREST: Convolutional residual learning for visual tracking," *IEEE Conference on Computer Vision*, Venice, Italy, pp. 2574-2583, 2017.

[23] H. Fan and H. Ling, "Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking," *IEEE Conference on Computer Vision*, Venice, Italy, pp. 5487-5495, 2017.

[24] Experimental results of single-domain Reptile meta-tracking: https://www.youtube.com/watch?v=q1JWHTwNUI8

Chi-Yi Tsai received the B.S. and M.S. degree in electrical engineering from National Yunlin University of Science and Technology, Yunlin, Taiwan, in 2000 and 2002, respectively, and the Ph.D. degree in electrical and control engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2008.

In 2010, he joined the Department of Electrical Engineering, TamKang University, New Taipei City, Taiwan, where he is currently a Professor. His research interests include image processing, color image enhancement processing, visual tracking, visual servoing, deep learning and computer vision.

Shang-Jhih Jhang received the B.S. and M.S. degree in electrical engineering from Tamkang University, New Taipei City, Taiwan, in 2015 and 2018, respectively. In 2018, he joined the Mindtronic AI, Taipei, Taiwan, where he is currently a computer vision engineer responsible for developing driver monitoring system. His research interests include deep learning and computer vision.

Mobile Spherical Robot Design

Chin-Cheng Liu
Department of Electrical and Computer Engineering
Tamkang University
New Taipei City, Taiwan
136382@mail.tku.edu.tw

Cheng-En Tsai
Department of Electrical and Computer Engineering
Tamkang University
New Taipei City, Taiwan
hanknot83@gmail.com

Wei-Fan Lai
Department of Electrical and Computer Engineering
Tamkang University
New Taipei City, Taiwan
a13572468u@gmail.com

Yang-Han Lee
Department of Electrical and Computer Engineering
Tamkang University
New Taipei City, Taiwan
yhleepp@gmail.com

Ching-Chang Wong*
Department of Electrical and Computer Engineering
Tamkang University
New Taipei City, Taiwan
wong@ee.tku.edu.tw

Chia-Hao Hsu
Green Energy & Environmental Laboratories
Industrial Technology Research Institute
Chutung, Hsinchu, Taiwan
haohaohsu@itri.org.tw

Yang-Guang Liu
Green Energy & Environmental Laboratories
Industrial Technology Research Institute
Chutung, Hsinchu, Taiwan
ygliu@itri.org.tw

***Abstract*—In this paper, a simulation system of a spherical robot is designed and implemented. The Gazebo, a physics simulation engine of Robot Operating System (ROS), is used to implement the simulation system to simulate and analyze the spherical robot. The spherical robot can move freely on the field. The outside of the spherical robot is a hollow spherical shell. There is a two-wheeled car driven by motors on the inside of the spherical robot. When the motor of the two-wheeled car rotates, the driving wheels drive the two-wheeled car. Then the spherical robot changes the posture in the hollow spherical shell to change the center of gravity of the spherical robot so that the spherical robot moves on the field. Then, the spherical robot is used as a carrier. A fan is mounted above the sphere and the spherical robot still can move on the field. After installing the fan device above the spherical robot, some results show that the implemented motion controller and balance controller are effective.**

Keywords—Spherical Robot, Physics Simulation Engine, Fuzzy System, Motion Control, Balance Control

I. INTRODUCTION

In 1996, Halme et al. [1] proposed a mobile robot ball, a spherical surface omni-directional mobile robot driven by an inside drive unit, and analyzed the uphill movement and obstacles crossing. Bicchi et al. [2] designed a two-wheeler into a hollow ball and described its kinematics, dynamics, and motion planning. The device is an unrestricted spherical vehicle that automatically rolls on the floor and can be moved anywhere in the environment. Bhattacharya and Agrawal [3] designed a spherical robot based on the principle of conservation of angular momentum, placing two motors vertically inside, and using the angular momentum generated by the high-speed rotation of the motor to move the robot in the opposite direction. Mukherjee et al. [4] proposed a spherical robot and used the center of gravity displacement to move the robot. The telescopic limbs and cameras are added to the sphere to facilitate unmanned missions such as battlefield reconnaissance and environmental detection. Javadi and Mojabi [5] proposed a new prototype of an omnidirectional robot system and analytical studied this spherical rolling robot. Sun et al. [6] proposed a double-driven moving sphere robot by changing its center of gravity and orthogonal crossing mechanism. Xiao et al. [7] designed a type of spherical mobile robot to serve as a platform for carrying sensing devices or actuators in environments where demanding conditions and stability of mechanical platforms are critical. The moving sphere robot BHQ-1 was designed by Zhan et al. [8] and the prototype BHQ-1 robot was designed to improve the structure of the BHQ-1G spherical mobile robot. Zhan et al. [9] developed a spherical mobile robot BHQ-1 for environmental detection. The spherical mobile robot has a radius of 200mm and has a camera and an infrared sensor mounted on one side of the robot spindle. Liu et al. [10] designed an environmentally-detected spherical mobile robot BHQ-2 with two cameras, which is a non-holonomic constraint system. Based on the spin theory, the velocity Jacobian matrix of the spherical mobile robot is derived. Q-Taro ball-type robot developed by SONY [11] and the test machine exhibited in 2002. It has dozens of sensors and basic single-word speech recognition that can be moved away from obstacles, interact with the user, or automatically return to the base. GroundBot mobile monitoring ball robots developed by Rotundus [12]. The ball has a height of 60cm, a width of 80cm and a weight of 25kg. It can work in -30℃~40℃ environment such as snow, sand, mud, and other roads.

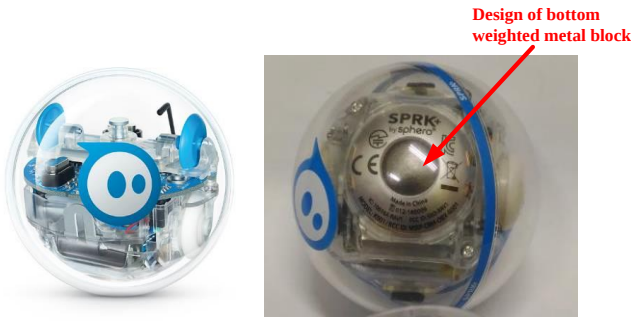
This paper intends to develop a simulation system specially designed for a spherical robot with Gazebo, a physics simulation engine of Robot Operating System (ROS) [13]. In the future, virtual scene testing of the venue or workplace can be linked to the robot-related technologies to verify and confirm some functions of the spherical robot. The rest of the paper is organized as follows: Design of the mobile spherical robot is described in Section II. Motion control of the spherical robot is described in Section III. Some straight-line simulation results are presented in Section IV. Two-dimensional balance control of the spherical robot is described in Section V. Some

Bureau of Energy, Ministry of Economic Affairs of the Republic of China

two-dimensional simulation results are presented in Section VI. Finally, some conclusions are made in Section VII.

II. Design of Mobile Spherical Robot

In the design of the spherical robot, it mainly referenced to two related products: (1) Sphero SPRK plus [14] and (2) Sphero Star War BB-9E [15]. The entity of Sphero SPRK plus is shown in Fig. 1, where a two-wheeled car is in the inside of the robot ball and a metal object is placed at the bottom of the two-wheeled car. The object lowers the center of gravity of the moving sphere. The center of gravity needs to be kept as low as possible to improve the stability of movement. The entity of Sphero SPRK plus is shown in Fig. 2, where the upper object is fixed above the robot with magnetic force and two small contact rollers are arranged on the contact surface to facilitate the movement on the spherical surface.



(a) (b)

Fig. 1. Entity of Sphero SPRK plus [14].

(a)

(b)

Fig. 2. Entity of Sphero Star War BB-9E [15].

The mathematical model of the spherical robot is obtained from the two-wheeled car and the schematic diagram of the two-wheeled car is shown in Fig. 3, where L is the distance between the wheel and center of the two-wheeled car and α is the angle between the direction of the spherical robot and the field coordinates. When the forward speeds of the two wheels are equal, the spherical robot will roll forward. When forward speeds of the two wheels are not equal, the spherical robot will change its rolling direction. In this paper, a fuzzy controller is proposed to control the spherical robot to achieve the desired motion trajectory.

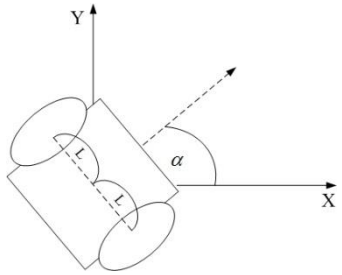


Fig. 3. Schematic diagram of the two-wheeled car.

The moving speed and the angular velocity of the two-wheeled car can be respectively described as

$$v = \sqrt{v_x^{\,2} + v_y^{\,2}} \tag{1}$$

and

$$\omega = \dot{\theta} \tag{2}$$

Its double wheels conversion matrix can be described as

$$\begin{bmatrix} v_L \\ v_R \end{bmatrix} = \begin{bmatrix} 1 & -L \\ 1 & L \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2}$$

In the straight-line simulation experiment, the bottom of the two-wheeled car is provided with an object, which lowers the center of gravity of the moving sphere. When the two-wheeled car moving inside the spherical shell, the center of gravity of the spherical robot is changed and the spherical robot moves. The diameter and weight of the sphere are 40 cm and 0.5 kg, respectively. The weight of the body of the two-wheeled car inside the sphere is 1 kg. In order to make the sphere move more easily, an additional object of 4 kg is added under the main body of the two-wheeled car to lower the center of gravity of the entire sphere. Simulation of a straight-line moving sphere is shown in Fig. 4. Detailed specifications of the two-wheeled car, spherical shell, and wheel are listed in Table I, Table II, and Table III, respectively.

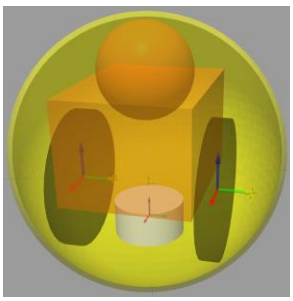


Fig. 4. Simulation of spherical robot.

TABLE I. SPECIFICATIONS OF TWO-WHEELED CAR

Description	Value
Length (cm)	22.5
Width (cm)	20
Height (cm)	32.1
Body weight (kg)	1
Attachment height (cm)	2.5
Attachment weight (kg)	4

TABLE II. SPECIFICATIONS OF SPHERICAL SHELL

Description	Value
Diameter (cm)	40
Thickness (cm)	0.8
Weight (kg)	0.5

TABLE III. SPECIFICATIONS OF WHEEL

Description	Value
Diameter (cm)	20
Thickness (cm)	2.5
Weight (kg)	0.2
Distance between two wheels (cm)	20

III. MOTION CONTROL OF SPHERICAL ROBOT

An straight-line moving simulation of the spherical robot is designed and described in Fig. 5. There is a motor-driven two-wheeled car inside and it can move freely on the field. When two motors drive the two-wheeled car to move inside the hollow spherical shell, the center of gravity of the spherical robot is shifted at the same time so that the spherical robot rotates and moves on the field. Its system diagram is described in Fig. 6. The trajectory planning, motion control, and simulation are respectively described as follows:

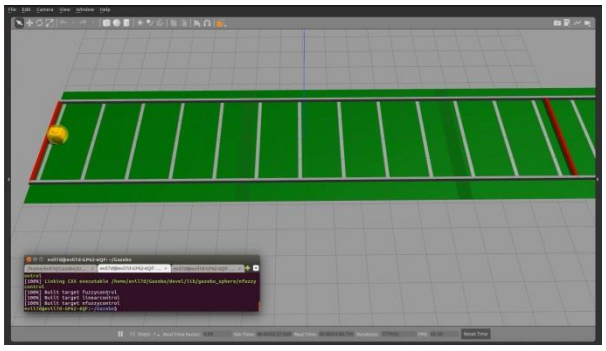


Fig. 5. Straight-line moving simulation of the spherical robot.

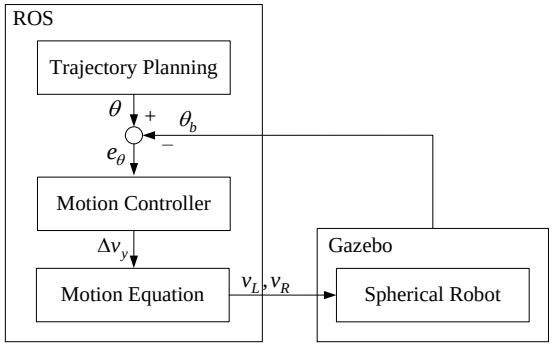


Fig. 6. System diagram of straight-line moving simulation of the spherical robot.

A. *Trajectory Planning*

The spherical robot is set to go forward from left to right, that is, the moving target direction θ of the spherical robot is 0, the forward speed v_x is a constant and the side moving speed v_y is 0.

B. *Motion Controller*

The fuzzy theory is used to design the motion controller. This controller is to calculate the compensation amount of the side movement speed based on the front angle when the sphere robot moves so that the robot can maintain the forward direction in the desired trajectory. The motion controller is a single-input-and-single-output system, where the input variable e_θ is the difference between the moving target direction θ of the robot and the actual forward direction θ_b, and the output variable Δv_y is the compensation speed of the lateral movement direction of the robot. The input variable e_θ is can be described as

$$e_\theta = \theta - \theta_b \tag{4}$$

The ranges of the input variable e_θ and the output variable Δv_y are defined respectively as

$$e_\theta \in [-90, 90] \tag{5}$$

$$\Delta v_y \in [-3, 3] \tag{6}$$

The following term sets are used to describe the fuzzy sets of each input and output fuzzy variables:

$$T(e_\theta) = \{\mathrm{P,Z,N}\} = \{A_1, A_2, A_3\} \tag{7}$$

$$T(\Delta v_y) = \{\mathrm{P,Z,N}\} = \{B_1, B_2, B_3\} \tag{8}$$

where the following linguistic terms (Positive (P), Zero (Z), and negative (N)) are considered to describe the input and output variables of the fuzzy system. In the definition of the fuzzy set, as shown in Fig. 7, the fuzzy set of input variables e_θ are described by the triangle membership function and the trapezoidal membership function, where $\mu_{A_i}(e_\theta)$ represents the attribution of the input variable. On the definition of the fuzzy set of output variables Δv_y, as shown in Fig. 8, the singleton membership functions are used to describe the fuzzy set, where $\mu_{B_i}(\Delta v_y)$ is its degree of attribution.

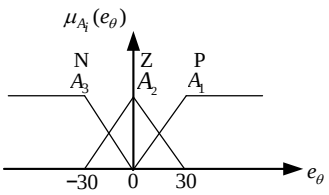


Fig. 7. Membership functions of input variable e_θ.

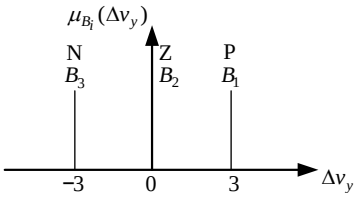


Fig. 8. Membership functions of Output variable Δv_y.

The proposed rule bases for the motion controller are described in Table IV. The proposed fuzzy rules of motion controller can be described as follows:

Rule $R_i(j_1)$:

$$\text{If } e_\theta \text{ is } A_{j1} \text{ THEN } \Delta v_y \text{ is } B_{j2} \tag{9}$$

where $A_{j1} \in T(e_\theta)$, $B_{j2} \in T(\Delta v_y)$, $j_1 \in \{1,2,3\}$ and $j_2 \in \{1,2,3\}$.

TABLE IV. FUZZY RULE BASE OF MOTION CONTROLLER

	e_θ		
	N	Z	P
Δv_y	$N(B_3)$	$Z(B_2)$	$P(B_1)$

Based on the weighted average defuzzifier, Δv_y is taken as the output of this fuzzy control system, so the output of the motion controller can be expressed as

$$\Delta v_y = \frac{\sum_{i=1}^{3} w(j_1)\cdot c(B_{j_2})}{\sum_{i=1}^{3} w(j_1)} = \frac{\sum_{i=1}^{3} \mu_{A_i}(e_\theta)\cdot c(B_{j_2})}{\sum_{i=1}^{3} \mu_{A_i}(e_\theta)} \tag{10}$$

where $w(j_1)$ is the appropriate degree to which the $R_i(j_1)$ rule is touched, and $c(B_{j_2})$ is the crisp value represented by the singleton membership function.

C. Simulation Environment

In the simulation environment, a physics engine simulator Gazebo is used to build related virtual scenes. Gazebo is an open-source 3D robotics simulator. The structural parts of the designed spherical robot can be imported into Gazebo to build a virtual robot. Gazebo can be used to control the robot motion and verify the algorithm by the simulation results. One environment for the designed spherical robot is shown in Fig. 9.

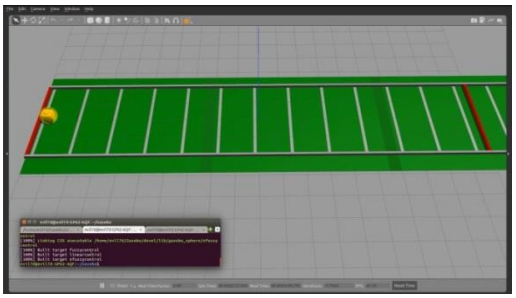


Fig. 9. One simulation environment for the designed spherical robot.

IV. STRAIGHT-LINE SIMULATIONS OF MOTION CONTROL

In order to effectively test the effect of the motion controller, the simulator Gazebo is used to build an experimental field to compare the control results without and with the motion controller. As shown in Fig. 9, the leftmost position and the rightmost position in the experimental field are the initial position and the end position of the spherical robot, respectively. When the simulation starts, the spherical robot rolls in a straight line to the right side of the experimental field until it rolls to the far right of the experimental field. The movement trajectory is the center horizontal line of the site.

Some simulation results of the moving process of the spherical robot without the motion controller are shown in Fig. 10 and the trajectory errors are shown in Fig. 11. The initial position of the spherical robot is at the center of the leftmost position of the field. At the starting time, the error is 0. When the spherical robot rolls to the right side, an error occurs. The spherical robot gradually moves away from the straight trajectory of the center horizontal line and the trajectory error gradually becomes larger.

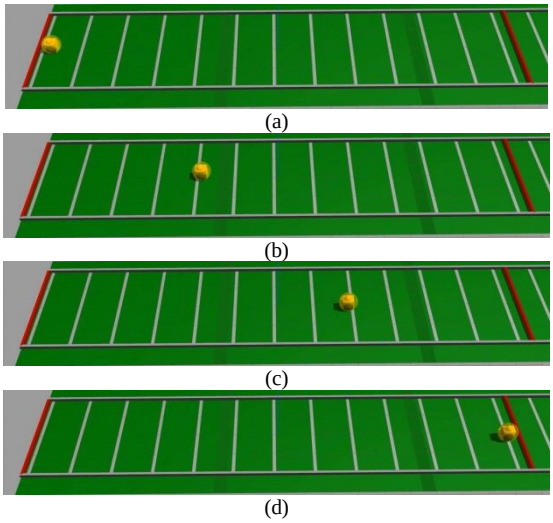


Fig. 10. Simulation results of the moving process of the robot without the motion controller (Depart from the center of the far left of the field).

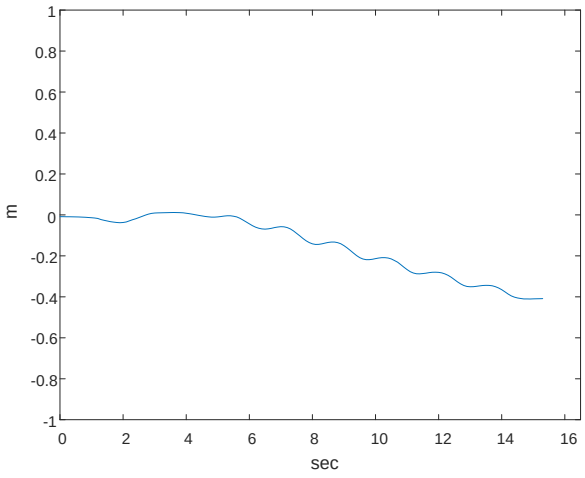


Fig. 11. Trajectory error without the motion controller (Depart from the center of the far left of the field).

In Fig. 12, the initial position of the spherical robot is at 0.5m above the center of the leftmost position of the field, so there is an initial error. When the spherical robot is in the rolling, the error gradually becomes larger, making the spherical robot more offset from the linear line of the center horizontal line. These trajectory errors are shown in Fig. 13.

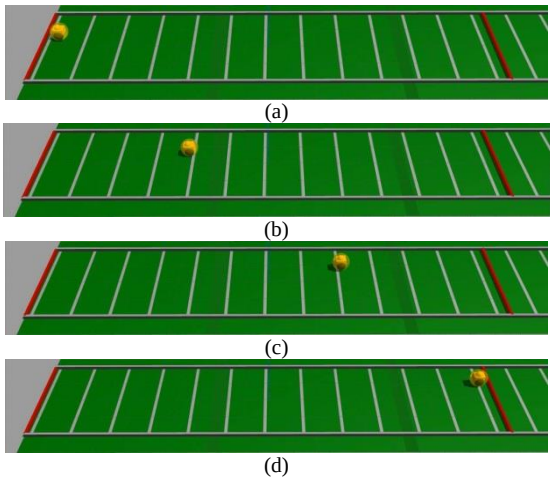


Fig. 12. Simulation results of the moving process of the robot without the motion controller (Depart from the top 0.5m of the center of the leftmost position of the field).

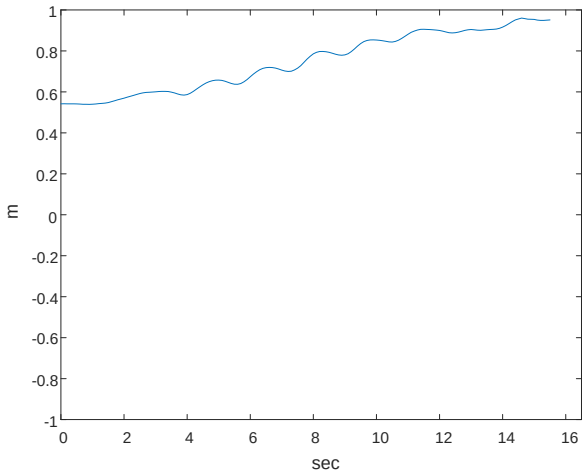


Fig. 13. Trajectory error without the motion controller (Depart from the top 0.5m of the center of the leftmost position of the field)

Another moving process of the spherical robot with the motion controller is described in Fig. 14. The initial position of the spherical robot is at the center of the leftmost position of the field. At the starting time, the error is 0. When the spherical robot rolls to the right side, an error occurs. The spherical robot is kept near the straight trajectory of the center horizontal line and the trajectory error is kept within a certain range. These trajectory errors are shown in Fig. 15.

In Fig. 16, the initial position of the spherical robot is at 0.5 above the center of the leftmost position of the field, so there is an initial error. The motion controller performs the control compensation and the spherical robot still moves near the straight track close to the center horizontal line. These trajectory error are shown in Fig. 17.

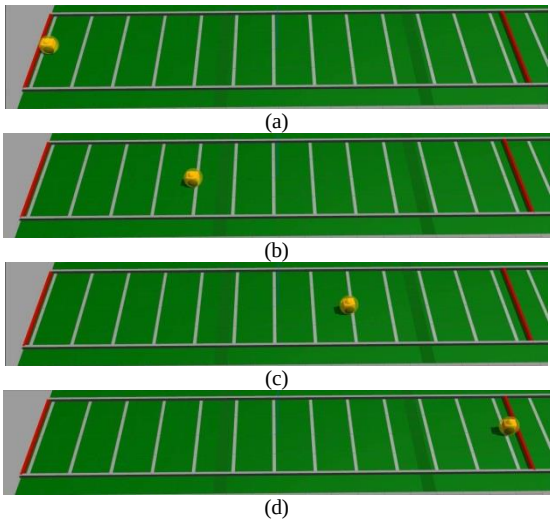


Fig. 14. Simulation results of the moving process of the robot with the motion controller (Depart from the center of the far left of the field).

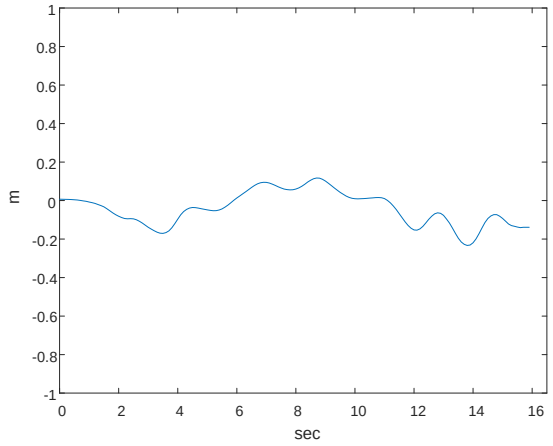


Fig. 15. Trajectory error with motion controller (Depart from the center of the far left of the field).

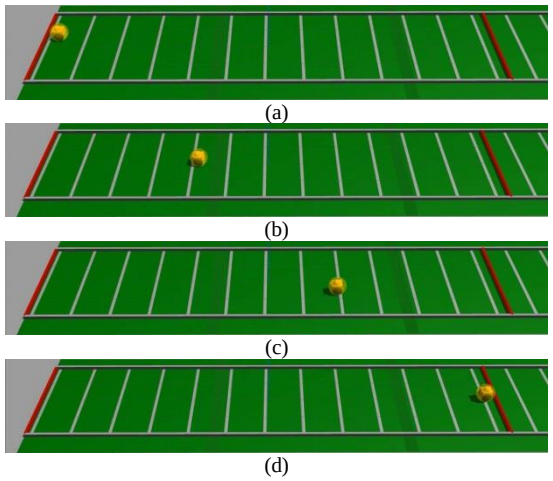


Fig. 16. Simulation results of the moving process of the robot with motion controller (Depart from the top 0.5m of the center of the leftmost position of the field).

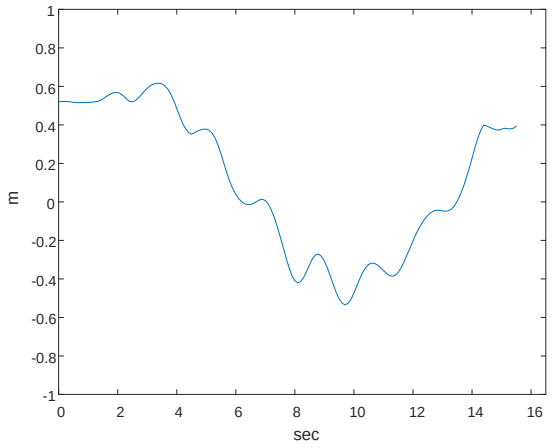


Fig. 17. Trajectory error with motion controller (Depart from the top 0.5 of the center of the leftmost position of the field).

The spherical robot rolls by the center of gravity offset. When the spherical robot is rolling, the direction error of traveling is likely to occur at this time. The two-wheeled car inside the robot continues to have errors in the direction of travel and the error will continue to expand. The experimental results represent the proposed motion controller can effectively control the spherical robot and let the trajectory error within a certain range.

V. Two-Dimensional Simulation of Motion Control of Spherical Robot

In the two-dimensional simulation, a fan is fixed above the vertical axis of the center of the spherical robot, and the distance is from the vertical axis of the center of the two-wheeled car to the distance between the spherical shells. The size of the fan has no effect on the simulation. The main effect is the weight of the fan. The weight of the fan mechanism, circuit and battery needs to be concentrated on the bottom of the fan. It is assumed that the fan can be completely adsorbed on the spherical robot by the magnetic force and the relative position of the fan and the spherical robot is fixed. This problem does not calculate the problem of magnetic attraction, but in the future, the magnetic force problem needs to be considered in the actual production. The two-dimensional simulation of the spherical robot is shown in Fig. 18 and the detailed specifications of the fan are shown in Table V. The system diagram of the two-dimensional simulation of the spherical robot is shown in Fig. 19. A balance controller is added in this system diagram and described as follows:.

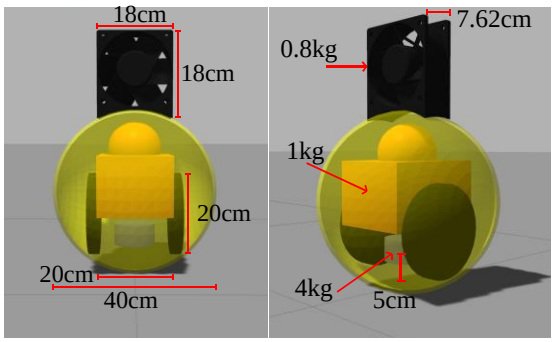


Fig. 18. Two-dimensional simulations of the spherical robot.

TABLE V. Detailed Specifications of Fan

Description	Value
Length (cm)	18
Width (cm)	7.62
Height (cm)	18
Weight (kg)	0.5

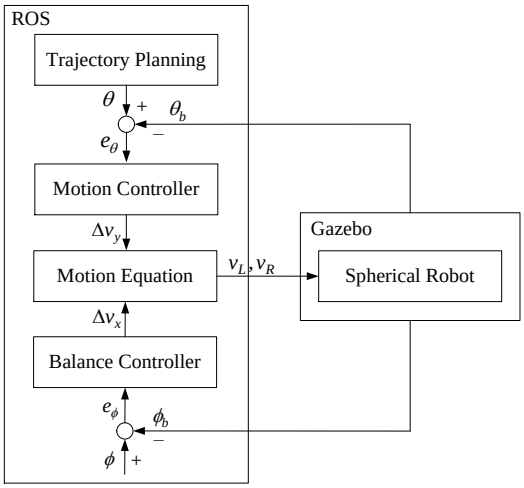


Fig. 19. System diagram of two-dimensional simulation of the spherical robot.

The fuzzy theory is used to design the balance controller. When the spherical robot moves, this controller observes the fan tilt angle and determines a compensation amount for the forward movement acceleration and deceleration so that the fan is maintained above the robot. The balance controller is a single-input-and-single-output system, where the input variable e_ϕ is the difference between the moving target direction ϕ of the robot and the actual forward direction ϕ_b, and the output variable Δv_x is the compensation speed of the lateral movement direction of the robot. The input variable e_θ is can be described as

$$e_\phi = \phi - \phi_b \tag{11}$$

The ranges of the input variable e_ϕ and the output variable Δv_x are defined respectively as

$$e_\phi \in [-180, 180] \tag{12}$$

$$\Delta v_x \in [-1, 1] \tag{13}$$

The following term sets are used to describe the fuzzy sets of each input and output fuzzy variables:

$$T(e_\phi) = \{S, M, L\} = \{C_1, C_2, C_3\} \tag{14}$$

$$T(\Delta v_x) = \{P, Z, N\} = \{D_1, D_2, D_3\} \tag{15}$$

where the linguistic terms (Small (S), Middle (M), and Large (L)) and (Positive (P), Zero (Z), and Negative (N)) are

considered to describe the input and output variables of the fuzzy system. As shown in Fig. 20, the triangle membership function and the trapezoidal membership function are used to describe the fuzzy sets of the input variable e_ϕ. On the definition of the fuzzy set of the output variables Δv_x, as shown in Fig. 21, the singleton membership functions are used to describe the fuzzy set.

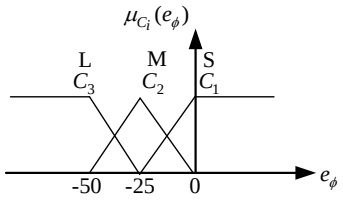


Fig. 20. Membership functions of input variable e_ϕ.

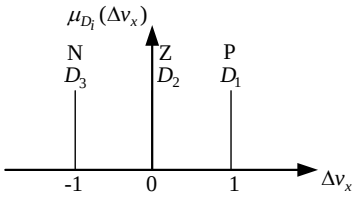


Fig. 21. Membership functions of output variable Δv_x.

The proposed rule base for the motion controller is described in Table VI. The proposed fuzzy rules of the motion controller can be described as follows:

Rule $R_i(j_3)$:

$$\text{If } e_\phi \text{ is } C_{j_3} \text{ THEN } \Delta v_x \text{ is } D_{j_4} \tag{16}$$

where $C_{j_3} \in T(e_p)$, $D_{j_4} \in T(\Delta v_x)$, $j_3 \in \{1, 2, 3\}$ and $j_4 \in \{1, 2, 3\}$.

TABLE VI. FUZZY RULE BASE OF BALANCE CONTROLLER

	e_ϕ		
	N	Z	P
Δv_x	$N(D_3)$	$Z(D_2)$	$P(D_1)$

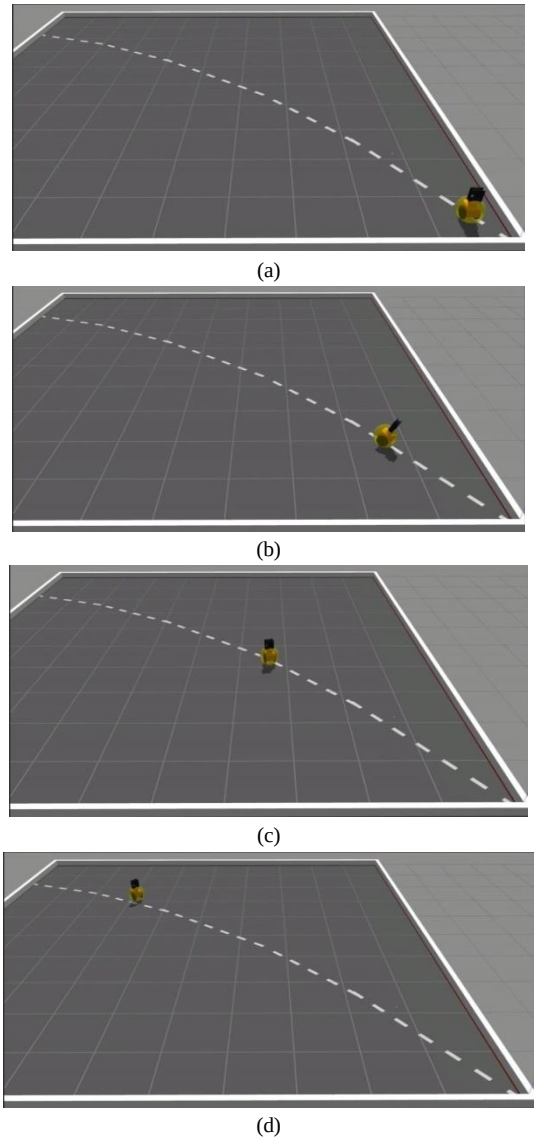
Based on the weighted average defuzzifier, Δv_x is taken as the output of this fuzzy control system, so the output of the balance controller can be expressed as follows:

$$\Delta v_x = \frac{\sum_{i=1}^{3} w(j_3) \cdot c(D_{j_4})}{\sum_{i=1}^{3} w(j_3)} = \frac{\sum_{i=1}^{3} \mu_{C_i}(e_\phi) \cdot c(D_{j_4})}{\sum_{i=1}^{3} \mu_{C_i}(e_\phi)} \tag{17}$$

where $w(j_3)$ is the appropriate degree to which the $R_i(j_3)$ rule is touched, and $c(D_{j_4})$ is the crisp value represented by the singleton membership function.

VI. TWO-DIMENSIONAL SIMULATIONS OF BALANCE CONTROL

When the spherical robot follows a curved trajectory in a two-dimensional space, some simulation results of the balance controller are shown in Fig. 22. It illustrates the spherical robot can actually follow the desired curved trajectory. The tilt angle of the fan can be kept within a certain range. The trajectory error of the robot and the tilt angle of the fan are respectively shown in Fig. 23 and Fig. 24. These results show that the designed motion controller can effectively control the motion of the spherical robot and limit the trajectory error within a certain range. However, after adding the fan in the top of the spherical robot, the weight of the fan tilts the spherical robot backwards, increasing the difficulty of balancing for the spherical robot. But the balance controller can limit the tilt angle of the fan within a certain range.



(a)

(b)

(c)

(d)

Fig. 22. Simulation results of the two-dimensional moving process of the spherical robot with the motion controller and balance control.

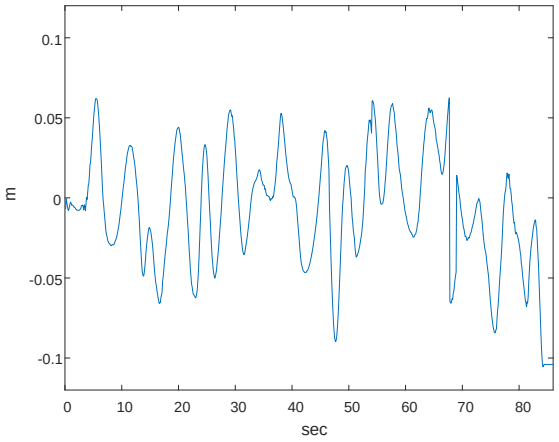


Fig. 23. Moving trajectory errors of the spherical robot follows a curved trajectory in a two-dimensional space.

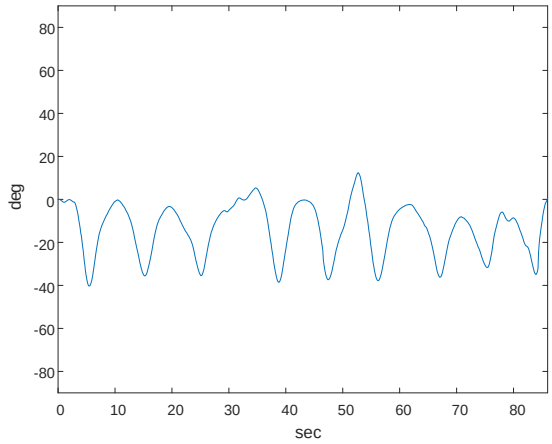


Fig. 24. Fan tilt angle of spherical robot when it follows a curved trajectory in a two-dimensional space.

VII. Conclusion

In this paper, a simulation system of a spherical robot is designed and implemented. A 3D physics engine simulator Gazebo is used to build a simulation environment and a spherical robot is designed in the Gazebo to test the designed functions. Through the simulation environment, the situation of the viewing motion mode can be presented. Before the completion of the spherical robot, the simulation environment can be used to verify the posture control and movement of the spherical robot. First, the spherical robot model and its related parameter are established. Then, a motion controller and a balance controller based on the fuzzy theory are designed for the spherical robot. Some simulation results are presented to illustrate the efficiency of the proposed method. We can see that the dedicated simulation environment makes the development of the control strategy for the spherical robot is quite convenient.

Acknowledgment

This research was supported in part by the Bureau of Energy, Ministry of Economic Affairs of the Republic of China.

References

[1] A. Halme, T. Schonberg, and Y. Wang, "Motion control of a spherical mobile robot," *in Proc 4th IEEE International Workshop on Advanced Motion Control AMC' 96*, Japan, 1996, pp. 100-106.

[2] A. Bicchi, A. Balluch, D. Prattichizzo, and A. Gorelli, "Introducing the "SPHERICLE": An experimental testbed for research and teaching in nonholonomy," *in Proc 1997 IEEE Int. Conf. on Robotics and Automation*, Albuquerque, New Mexico, 1995, pp. 2620-2625.

[3] S. Bhattacharya and S. K. Agrawal, "Design, experiments and motion planning of a spherical rolling robot," *in Proc 2000 IEEE International Conference on Robotics & Automation*, San Francisco, 2000, pp. 1207-1212.

[4] R. Mukherjee, M. A. Minor, and J. T. Pukrushpan, "Simple motion planning strategies for Spherobot: a spherical mobile robot," *in Proc. IEEE Int. Conference on Decision and Control*, Phoenix, 1999, pp. 2132-2 137.

[5] A. H. Javadi and P. Mojabi, "Introducing august: A novel strategy for an omnidirectional spherical rolling robot" *in Proc. IEEE Int. Conference on Robotics and Automation*, 2002, pp. 3527 - 3533.

[6] H. Sun, A. Xiao, Q. Jia, and L. Wang, "Omnidirectional kinematics analysis on bi-driver spherical robot," *Journal of Beijing University of Aeronautics and Astronautics*, vol. 31, pp. 736 -739, 2005.

[7] A. Xiao, H. Sun, and Q. Liao, "The design and analysis of a kind of spherical mobile robot," *Development & Innovation of Machinery & Electrical Products*, vol. 17, pp. 14-16, 2004.

[8] Q. Zhan, C. Jia, X. Ma, and M. Chen, "Analysis of moving capability of a spherical mobile robot," *Journal of Beijing University of Aeronautics and Astronautics*, vol. 31, pp. 744 -747, 2005.

[9] Q. Zhan, T. Zhou, M. Chen, and S. Cai, "Dynamic trajectory planning of a spherical mobile robot" *in Proc IEEE International Conferences on Robotics, Automation & Mechantronics* (RAM 2006), Bangkok, 2006, pp. 714-719.

[10] Z. Liu, Q. Zhan, and Y. Cai, "Motion control a spherical mobile robot for environment exploration," *ACTA Aeronautica et Astronautica Sinca*, vol. 29, pp. 1673-1679, 2008.

[11] Website of Sony Q-Taro: http://www.sonyaibo.net/aboutqtaro.htm.

[12] Website of Rotundus: http://www.rotundus.se/.

[13] URL "ROS.org | Powering the world's robots.": http://www.ros.org.

[14] Website of of Sphero SPRK plus: https://www.sphero.com/sphero-sprk-plus

[15] Website of Sphero Star War BB-9E: https://www.sphero.com/bb-9e-app-enabled-droid

Information for Authors

Aim and Scope

The ***i*Robotics** is an official journal of Robotics Society of Taiwan (RST) and is published quarterly. The ***i*Robotics** will consider high quality papers that deal with the theory, design, and application of intelligent robotic system, intelligent artificial system, and extension theory systems ranging from hardware to software. Survey and expository submissions are also welcome. Submission of a manuscript should indicate that it has been neither published, nor copyrighted, submitted, accepted for publication elsewhere (except that the short version may have been presented at the conferences). Submitted manuscripts must be typewritten in English and as concise as possible.

Process for Submission of a Manuscript

The ***i*Robotics** publishes two types of articles: regular papers and technical notes. All contributions are handled in the same procedure, where each submission is reviewed by an associate editor who makes an initial decision to send the manuscript out for peer review or to reject without external review. Articles can be rejected at this stage for a variety of reasons, such as lack of novelty, topics outside the scope of the Journal, flaws in the scientific validity, or unprofessional presentation. We are therefore not normally able to provide authors with feedback on rejected manuscripts. If the associate editor believes the article may be of interest to our readers, it is then sent out for external peer review by at least two external reviewers. According the recommendation of the associate editor, the Editor-in-Chief makes the final decision. All manuscripts should be submitted electronically in Portable Document Format (PDF) through the manuscript submission system at [http://www.rst.org.tw]. The corresponding author will be responsible for making page proof and signing off for printing on behalf of other co-authors. Upon acceptance of a paper, authors will be requested to supply their biographies (100 to 200 words) and two copies of the final version of their manuscript (in DOC format and in PDF format).

Style for Manuscript

Papers should be arranged in the following order of presentation:

1) First page must contain: a) Title of Paper (without Symbols), b) Author(s) and affiliation(s), c) Abstract (not exceeding 150 words for Papers or 75 words for Technical Note, and without equations, references, or footnotes), d) 4-6 suggested keywords, e) Complete mailing address, email address, and if available, facsimile (fax) number of each author, f) Preferred address for correspondence and return of proofs, and g) Footnotes (if desired).
2) The text: Submitted manuscripts must be typewritten double-spaced. All submitted manuscripts should be as concise as possible. Regular papers are normally limited to 26 double-spaced, typed pages, and technical notes are 12 double-spaced, typed pages. Please see the Page charge for those who want to submit long papers.
3) Acknowledgements of financial or other support (if any).
4) References: References should be numbered and appear in a separate bibliography at the end of the paper. Use numerals in square brackets to cite references, e.g., [15]. References should be complete and in the style as follows.
   [1] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller - Part I," *IEEE Trans. Syst. Man Cybern.*, vol. 20, no. 2, pp. 404-418, 1990.
   [2] C. Golaszewski and P. Ramadge, "Control of discrete event processes with forced events," in *Proc. of 26th IEEE Conf. Decision and Control*, Los Angeles, CA, pp. 247-251, Dec. 1987.
   [3] P. E. Wellstead and M. B. Zarrop, *Self-Tuning Systems*, New York: Wiley, 1991.
   [4] Project Rezero, available at http://rezero.ethz.ch/project_en.html (last visited: 2017-07).
5) Tables
6) Captions of figures (on separate sheet of paper)

Style for Illustrations

1) It is in the author's interest to submit professional quality illustrations. Drafting or art service cannot be provided by the Publisher.
2) Original drawings should be in black ink on white background. Maximum size is restricted to 17.4 by 24.7 cm. Glossy prints of illustrations are also acceptable.
3) All lettering should be large enough to permit legible reduction of the figure to column width, sometimes as small as one quarter of the original size. Typed lettering is usually not acceptable on figures.
4) Provide a separate sheet listing all figure captions, in proper style for the typesetter, e.g., "Fig. 5. The error for the proposed controller."
5) Illustrations should not be sent until requested, but authors should be ready to submit these immediately upon acceptance for publication.

Page Charges

After a manuscript has been accepted for publication, the author's company or institution will be approached with a request to pay a page charge to cover part of the cost of publication. The charges include:

1) NT$ 5000 for the 10 printed pages of a full paper or for the 6 printed pages of a short paper, and the excess page charge of NT$ 1500 per extra printed page for both full and short papers.
2) For color figures or tables, an additional cost will be charged. The cost, depending on the number of color figures/tables and the final editing result, will be given upon the acceptance of this paper for publication.

Copyright

It is the policy of the RST to own the copyright of the technical contributions. It publishes on behalf of the interests of the RST, its authors, and their employers, and to facilitate the appropriate reuse of this material by others. Authors are required to sign a RST Copyright Form before publication.

Manuscripts (in PDF Format) Submission Website: http://www.rst.org.tw

Editor-in-Chief: Prof. Ching-Chih Tsai, Department of Electrical Engineering, National Chung Hsing University, Taiwan
Email: cctsai@nchu.edu.tw
Prof. Tzuu-Hseng S. Li, Department of Electrical Engineering, National Cheng Kung University, Taiwan
Email: thsli@mail.ncku.edu.tw

Managing Editor: Dr. Feng-Chun Tai, Department of Electrical Engineering, National Chung Hsing University, Taiwan
Email: fctai@nchu.edu.tw

iRobotics

VOLUME 2, NUMBER 2 JUNE, 2019

CONTENTS

REGULAR PAPERS

TECHNICAL NOTE

iRobotics Volume 2, Number 2, JUNE, 2019