Deep Learning-based Cooperative Trail Following

Mingyang Geng[1], Yiying Li[1], Bo Ding[1], and Huaimin Wang[1]
[1]National Key Laboratory of Parallel and Distributed Processing,
College of Computer, National University of Defense Technology
ChangSha, China
{gengmingyang13, liyiying10, dingbo, hmwang}@nudt.edu.cn

***Abstract*—Vision perception for unmanned ground vehicle trail following has received significant attention from the robotics community. Recently, a new approach which is based on deep learning used as a supervised image classifier has achieved great success and outperforms the traditional methods which aim to find the low-level features. However, the robots' views are limited and may be corrupted by substantial obstacles. One approach to address this problem is to complement the viewing capabilities of an unmanned ground vehicle with overhead data gathered from an aerial source. But the traditional methods of UAV/UGV coordination based on the vision focused more on the process of valuable feature-extraction, which is complicated and time-consuming. This paper presents techniques to achieve accurate UGV trail following by applying CNN models to the coordination of UAV and UGV. Qualitative and quantitative results computed on a large simulation and real-world datasets show that our approach has significantly improved the accuracy of the ground vehicle alone.**

Keywords—Visual-Based Navigation; Cooperative perception; Aerial Robotics; Ground Robotics; Deep Learning

I. Introduction

Deep learning has recently emerged as a powerful tool for various tasks in robotics. The advantages of deep learning over common alternatives are generality: reduces the need for feature engineering and has best-in-class performance on problems in vision, speech, language areas. A typical robotic application scenario is vision-based trail following [1], i.e., the problems of making robots autonomously follow a man-made trail (such as those normally traversed by hikers or mountain-bikers). Solving this problem would be the most efficient and safest way for a ground robot to travel medium and long distances in different environments for search and rescue. The navigation problem was casted as an image classification task and solved by using specific deep learning models to learn a control strategy that mimics the choice of an expert driver based on the visual input. Unlike the previous works [2], [3] dealing with trail perception solved a segmentation problem, i.e., aimed at determining which areas of the input image correspond to the image of the trail, deep learning techniques bypass the need by directly operating on the raw RGB frames and provide high-level information. The accuracy of it is comparable to the performance of humans, reaching 85.2%, much higher than that of the traditional method using image saliency [4] (52.3%).

Most of the existing studies on deep learning-based trail following are for single-robot. With the development of technology, cooperative multi-robot system is becoming more and more important in civil and military areas for better performing tasks. For example, in the search and rescue problem, the images acquired by the UAV can be adopted to survey the surrounding environment with the aim of getting information about the target. Then the UGV can follow the UAV to rescue the target in the shortest time. When the ground robot encounters enormous obstacles, such as trees and buildings, the UAV can provide a much broader view from a higher altitude to help the UGV get an adequate understanding of the environment. This method can effectively complement the viewing capabilities of an unmanned ground vehicle by taking advantage of the heterogeneity of a mixed robotic system. Therefore, how to combine the cooperative trail following method with deep learning techniques remains a great challenge which we are facing now.

There are two main challenges in the deep learning – based cooperative trail following method: (1) Data Fusion and (2) Quality of Service (QoS) Assurance. The data fusion problem aims to make full use of the data from aerial and ground. That means we need to utilize the useful information and make up the limitations of the two images for the purpose of achieving a global optimization. The QoS assurance is the problem of maximally utilizing the aerial data while minimally sacrificing the recognition latency because the robot interacts directly with the physical world. Therefore, the applications used in robots generally give considerable attention to QoS, such as latency or real-time assurance. For example, a low response time in an auto-driving vehicle may cause a disastrous traffic accident.

In order to solve these problems, we proposed the corresponding solutions to the challenges above. Towards the first problem, the machine learning technologies, which have been used for a long time [5,6] to map visual inputs to steering commands [7, 8, 9], can solve this by making full use of the data. The technologies could be implemented in

the cloud, using the architecture of "Cloud Robotics" [10], which can make robots break through the limitations of their resources and perform complex algorithms. To be concrete, the ground robot and the aerial robot can capture the images from different altitudes and then send the data to the cloud to get a final decision. The cloud stores the final classifier and builds a bridge for the communication between the ground and aerial robots. There are three main reasons for using the "Cloud" architecture. The first one is that we need a complex mechanism that will not only guarantee the coordination of the aerial and ground robots but also perform the complex algorithms corresponding to the DL-Cooper (Deep Learning Cooperative) method. Meanwhile, the whole process can be implemented on the cloud because the cloud can accelerate the calculation process, which will save much time sacrificing little cost of transmission. The second one is that this approach can make the robot save the storage space and the energy consumption compared with the method which makes the ground robot deal with the problem alone. The third one is knowledge integration, that is to say, the cloud can integrate the knowledge from the aerial and ground and make them complement with each other.

To guarantee the QoS requirement, we also proposed the methods which are aiming for decision fusion, not feature fusion. Because the feature contains more information than the decision and the process of transformation will be time-consuming. We proposed the concept of **"Threshold"** which stands for the confidence of the ground robot to minimally sacrifice the recognition latency as well. That is to say, when the recognition task remains within the capability of the ground robot (the confidence is larger than the **"Threshold"**), the result will be directly returned to the ground robot and the time delay can be quickly eliminated because everything is under control. However, when the task is beyond the capability of the ground robot (the confidence is less than the **"Threshold"**), the ground robot would forward the request and seek help from the cloud. In this paper, we developed a deep learning-based cooperative (DL-Cooper) trail following method for mobile robots to explore the wild environment with the support of cloud. Compared with other trail following methods, this method has efficiently dealt with the cases where the ground robot does not have enough confidence to make decisions (surrounded with obstacles) and also significantly promoted the accuracy for the final action.

The rest of this article is organized as follows: Section II reviews some related works. Section III introduces the problem formulation. The techniques used to train the classifiers based on CNN, the decision fusion algorithms and the evaluation of DL-Cooper are described in Section IV. In Section V, the experiments performed and the results obtained both in simulation and real-world environment are presented.

II. Related work

Our method covers various research areas, including deep learning for the path planning of robots, ground vehicle navigation with the support of aerial data and cloud robotics.

A. Deep Learning for the Path Planning Problem

Path planning is a challenging and mostly unsolved task for robotics. Solving such problem is important for many applications, including wilderness mapping [11], search and rescue. Most of the solutions using deep learning technologies are based on single-robot. A CNN-based reinforcement learning method [12] was proposed for mobile robots to explore an unknown environment based on raw sensor information. The policy search problem was transformed into a deep reinforcement learning problem that uses the convolutional neural networks (CNN) for the complex mapping between states and actions. The method is not end-to-end because it separated the whole architecture into two parts. Another approach [13] is to learn a left/right controller for an unmanned aerial vehicle (UAV) based on image data. The UAV was able to autonomously navigate through a forest while successfully avoiding collisions with trees in the majority of the cases. The approach [14] using a laser-based and data-driven motion planning method based on deep auto-encoders was also proposed. The collision avoidance capabilities of this method are shown in simulation and on a robotic platform.

Among the path planning problems, trail following would be the most efficient and safest way for a ground robot to travel medium and long distances in different environments for search and rescue. Unlike the methods above, our approach casts the navigation problem as a supervised image classification task and solved it by using Alexnet [15] model, a typical convolutional neural network. Meanwhile, our method is based on multi-robot and has successfully applied machine learning technologies to the decision fusion process in order to make full use of the valuable information.

B. Ground Vehicle Navigation with the Support of Aerial Data

When it is not possible for the ground robot to get an adequate understanding of the environment, aerial sensing can dramatically improve path planning performance by detecting large obstacles such as buildings and bodies of water as well as areas of preferable terrain such as roads. Sensory perception from aerial data has been studied by many researchers within the robotics community. The National Robotics Engineering Consortium developed a novel semi-autonomous unmanned ground vehicle (UGV) that utilized a dedicated unmanned helicopter that flew ahead of the UGV to detect holes and other hazards ahead of the vehicle [16]. The helicopter served as a scout to explore terrain before the UGV had to traverse it, allowing the UGV to replan its route to avoid certain areas entirely based on elevation hazards detected by the helicopter. The

General Dynamics Robotic Systems (GDRS) used a prior data from a manned aircraft to perform path planning and air-ground terrain registration for robot localization. In [17], the authors presented techniques to classify three-dimensional (3-D) points as load bearing surfaces or vegetation. During the mission, the robot followed the prior path while avoiding obstacles using on-board perception. The load-bearing surface recovered from the air and the ground vehicle was also co-registered in 3-D to estimate the absolute pose of the robot in the prior map.

In our method, the data from the ground robot is also taken into account. We want to make full use of the data from aerial and ground with the aim of making them complement with each other. Our purpose is to find the heading of the trail, not the traversability of the terrain. The deep learning technologies are also applied in the multi-robot coordinated method to further increase the accuracy and bypass the need to extract different kinds of features for better performance.

C. Cloud Robotics

Cloud robotics is a relatively new research field; it was initially proposed in 2010[10] aimed at extending the capability of robots by leveraging the rich services provided in cloud and offloading the complex robotic algorithms to cloud. There are many existing works which are based on the cloud robotics architecture. RoboEarth is a European FP7 project that aims to build a cloud to support knowledge sharing, computation offloading, and collaboration. It is based on the three-layer architecture: the back-end database, the cloud engine, and robots [18]. Rapyuta is a part of RoboEarth that helps robots offload heavy computation to the cloud by providing each robot with a customizable virtual computing environment [19]. RoboBrain provides a set of cloud services to store and share knowledge among robots [20]. A collection of practices also exists in cloud robotics that aims at the computation offloading of specific robotic algorithms such as SLAM [21, 22, 23].

In our method, we use the cloud to receive the vision input from the aerial and ground robots and then deal with the decision fusion process as a backend platform. The cloud also has the responsibility of keeping the robots coordinated. Instead of simply integrating robots with public cloud services, we introduce a task-specified mission architecture in order to minimize the recognition latency. That is to say, we seek help from the cloud only when the task is beyond the capability of the ground robot. Otherwise, the ground robot can make decision alone according to its own experience.

III. Problem formulation

Consider a general scene (Figure 1) that the UAV and UGV are coordinated to follow a trail with the aim of performing a search and rescue task. The UAV flies above the ground robot, facing the same direction as it. To be concrete, the ground robot is just in the middle of the aerial

Fig. 1. UAV/UGV coordinated architecture

image's bottom line. Then, the UAV can provide a vision system as a relative sensor, and fuse this information with the absolute position of the ground robot in order to achieve an absolute measurement of the position of the obstacles. By doing this, it is possible to obtain a mechanism of collaborative navigation and perform the rescue task effectively.

The scenario in the trail following problem is the same as the scene above. So we need to compute suitable steering commands based on the vision data from the ground and aerial. Given expert demonstrations we try to find a function

$$u = \begin{cases} h\left(p_g, p_a\right) & confidence \leq threshold \\ f\left(y_g, y_a\right) & confidence > threshold \end{cases} \quad \textbf{(1)}$$

that maps the vision data $\boldsymbol{y_g}$, $\boldsymbol{y_a}$ (the image from ground and aerial) to desired steering commands $\boldsymbol{u}$ when the confidence of the ground robot is larger than the threshold. The function $\boldsymbol{f}$ is parametrized by the weights of the deep learning network. During training, we find the weights that best explain a set of training data which is comprised of vision data and the according motion commands. When the confidence of the ground robot is less than the threshold, we aim to map the possibility $\boldsymbol{p_g}$, $\boldsymbol{p_a}$ (the possibility produced by the ground and aerial classifier) to desired steering commands $\boldsymbol{u}$. The function $\boldsymbol{h}$ means the corresponding decision fusion algorithm. The optimization criterion is based on (2), the difference between the predicted steering commands and the ones provided by the expert operator $\boldsymbol{u_{exp}}$. During deployment, the model parameters are given and the steering commands can be computed given the input data $\boldsymbol{y_g}$ and $\boldsymbol{y_a}$.

$$\begin{cases} \left|h\left(p_g, p_a\right) - u_{exp}\right| & confidence \leq threshold \\ \left|f\left(y_g, y_a\right) - u_{exp}\right| & confidence > threshold \end{cases} \quad \textbf{(2)}$$

In order to optimize the loss function above, we have to overcome several challenges. Firstly, the relevant information has to be extracted from the vision data. Secondly, using this information, a model has to be found that describes the relationship between the observations and the actions to take. Thirdly, during deployment, this model has to be used to take the right decisions as soon as new observations are available. Fourthly, the decision fusion algo-

rithms have to make full use of the aerial and ground data, extracting valuable information and discarding the dregs.

The label of the image is defined as follows. Consider a generic scene with a single trail in a wilderness setting. Our input from the ground robot is an image captured by a camera situated above the ground. Let $\vec{V}$ be the direction of the camera's optical axis; we assume that $\vec{V}$ lies on the horizontal plane. Furthermore, let $\vec{t}$ be the dominant direction of the trail: we define $\vec{t}$ as the (horizontal) direction towards which a hiker would start walking if standing at the position of the robot, with the goal of avoiding the obstacle. Let α be the signed angle between $\vec{V}$ and $\vec{t}$: we consider three classes, which correspond to three different actions that the (human or robotic) carrier of the camera should implement in order to remain on the trail, assuming that the camera is heading for the direction of motion. The details are shown in Figure 2.

- Turn Left (TL) if $-90^\circ < \alpha < -\beta$; i.e., the trail is heading towards the left part of the image.
- Go Straight (GS) if $-\beta \leq \alpha < +\beta$; i.e., the trail is heading straight ahead, at least in the close range.
- Turn Right (TR) if $+\beta \leq \alpha < +90^\circ$; i.e., the trail is heading towards the right part of the image.

Given the input image, our goal is to classify it to one of these three classes. In the following, we consider β = 15 degrees. The relationship between the two images is shown in Figure 3.

Fig. 2. Label description

"Turn Left" (Ground) "Turn Left" (Aerial)

a) In simulation environment

"Go Straight" (Ground) "Go Straight" (Aerial)

b) In real-world environment

Fig. 3. Relationship between the two images from aerial and ground

IV. The DL-Cooper Method

DL-Cooper is to make mobile robots follow the trails using deep learning methods with the aid of cloud. We separate this architecture into two parts: training classifiers and decision fusion. In this section, we will firstly introduce the framework of our architecture and then describe the details about our supervised learning, which can find the right steering commands according to the visual input. Finally, we will introduce the decision fusion algorithms as well as the performance benefits.

A. Framework of DL-Cooper

DL-Cooper, as shown in Figure 4, aims to apply deep learning technologies to the coordination of multi-robots. We separate this approach into two processes, training classifiers using deep learning models and fusing decisions based on the QoS requirement. Firstly, we built a supervised learning model as the classifier of the ground robot and the UAV by taking images as input and the command of the robot as output. The datum was manually labeled with control commands to tune the moving directions of the mobile robot. Secondly, the decision fusion methods are applied to combine the preliminary classification from the individual data sources and the fusing process is dealt with in the cloud.

B. Alexnet for Training

Since we solve the trail following problem as an image classification problem, a model has to be found that describes the relationship between visual input and the actions to take. We take Alexnet as our model which shows excellent advantages in image classification recent years. To be noticed, the main purpose of DL-Cooper is to combine the cooperative trail following method with deep learning technologies. So what we focus on is the comparison between the deep learning single-robot trail following method and DL-Cooper. Meanwhile, Alexnet is just a typical network which stands for the deep learning technologies. In our future work, we will try other techniques (i.e. deep reinforcement learning) for further research. Alexnet is composed of 5 convolutional layers and three fully connected layers. To be mentioned, we replace the last fully connected layer fc8 with a layer consisting of three nodes. We also disable "mirror" during training and testing, because it cannot apply to our experiment: an image with "Turn Right" command becomes an image that a robot should move to left by mirroring, but the flipped image still has a label as "Turn Right."

We train our net using back propagation for 90 epochs, which requires 46 hours on a workstation equipped with the Nvidia Tesla K80 GPU and NVIDIA CUDNN. The training process for real-world environment is the same as that in the simulation environment.

C. Decision Fusion Algorithms

Considering the problem of taking decisions, the previous method is merely taking the action whose value is the larg-

est. To achieve the goal of trail following through the help of cloud, we also consider the fact that robotic applications give considerable attention to the QoS because they interact directly with the physical world. So we proposed two decision fusion algorithms which aim to guarantee the QoS requirements.

The algorithms are implemented when the task is beyond the capability of the robot. The concept of "Threshold" which we mentioned before is to measure the confidence of the ground robot to finish the task alone. We use the max possibility the classifier's "SOFTMAX" layer out puts as the standard for measuring. That means, when the max possibility is under the threshold, the ground robot and quadrotor will take the decision fusion algorithms. Otherwise, the ground robot will take the decision according to its own classifier.

The algorithm below gives the pseudocode of "SVM"/"SOFTMAX" algorithms. The first algorithm "SVM" is to use the SVM classifier. We firstly splice the two three-dimensional vectors into one six-dimension vector. Then we take the new vector and the label as the input for training SVM classifier using the training dataset. So when the test data is available, we can use the same procedure to get the final output. The second algorithm "SOFTMAX" is to replace the SVM classifier with SOFTMAX classifier, and the other is the same as the "SVM" algorithm.

D. **Performance Benefits of Our Algorithms**

The benefits of introducing the decision fusion algorithms into our new method can be analyzed theoretically from the following aspects.

1) Accuracy improvement compared with the robot alone

By adding decision fusion algorithms to our method, the incorrect recognition results provided by the ground classifier may be corrected. Therefore, for samples that the ground classifier has recognized incorrectly, the final accuracy in our architecture can be promoted from 0 to

$$(1-p)P \tag{3}$$

where P is the recognition accuracy of the final classifier ("SVM/SOFTMAX" classifier), and p is the false positive rate of the confidence measurement transition mechanism. (i.e., the ground classifier recognized the image incorrectly but the transition mechanism does not send the sample to be further dealt with)

However, the promotion of recognition accuracy is not sure for all types of images. The "collateral damage" of the transition mechanism should be considered. For example, an image already correctly recognized by the ground classifier may be mistakenly sent to further deal with. We denote the accuracy of the ground classifier as M. Then total accuracy H can be calculated as follows:

$$H = M(1-p) + MpP + (1-M)(1-p)P \tag{4}$$

Here, $M(1-p)$ refers to the samples which are classified correctly by the ground robot and taken as the final decision. MpP means that the samples are already classified correctly by the ground robot but mistakenly sent to further deal with and finally classified correctly by the decision fusion algorithm. $(1-M)(1-p)$ means that the samples are wrongly classified by the ground robot but finally rectified attributed to the correctness of the transition mechanism. Through a simple deformation, we could obtain the following theorem:

(Theorem 1) Recognition accuracy $H \geq M$ if and only if

$$P \geq \frac{Mp}{Mp + (1-M)(1-p)} \tag{5}$$

2) Optimization of the time delay compared with the method that execute the algorithms all the time

The objective of introducing the "Threshold" is to optimize the request QoS, particularly, the latency for the images that the ground robot is familiar with. The total latency of images that DL-Cooper recognized can be calculated with

$$ld = lg + \mu lp \tag{6}$$

where lg is the average latency of the classifier stored in the ground robot, lp is that of the cloud and μ is the probability that the transition mechanisms believe it is the time to seek help from the cloud. Compared with the method that the ground robot seeks help from the cloud all the time, we can obtain the following theorem:

(Theorem 2) If $\mu < 1 - \frac{lg}{lp}$, then $ld < \mathrm{lp}$

lg is significantly less than lp in practice because the cloud is on the Internet while the ground classifier is deployed near the robots. As a reference, $\frac{lg}{lp}$ in the experiments presented in the next section is frequently below 0.1. Consequently, $1 - \frac{lg}{lp}$ is usually a number near 1 and the final average latency for the classification is certainly less than the method without "Threshold".

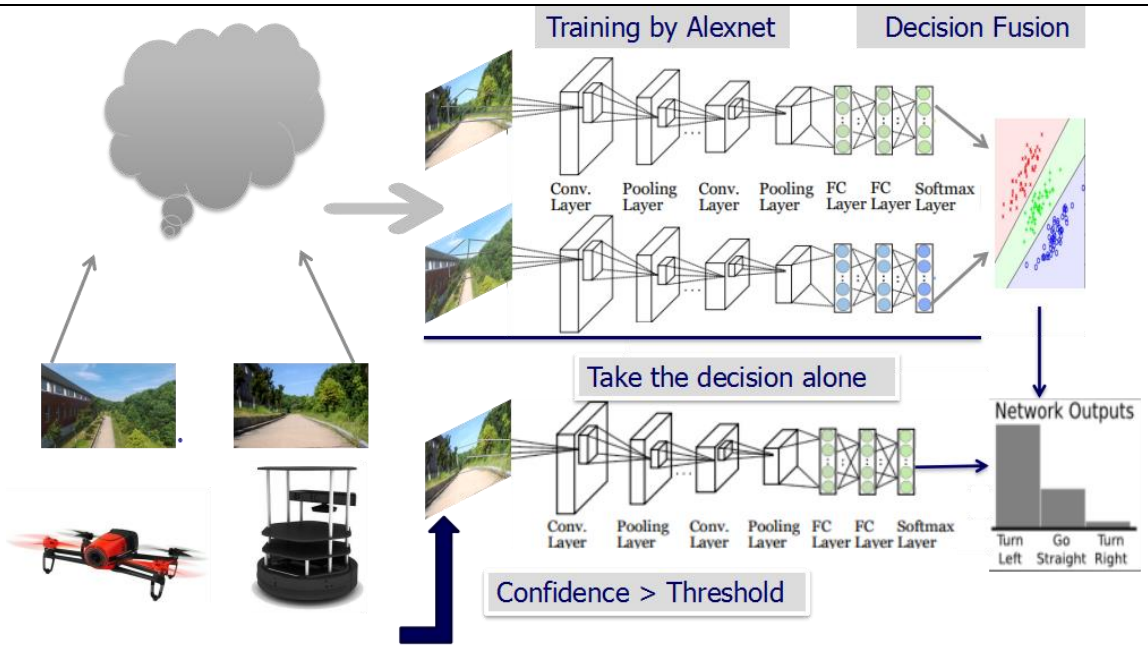


Fig. 4. Framework of DL-cooper

TABLE I. THE "SVM" / "SOFTMAX" DECISION FUSION ALGORITHM

Algorithm 1 : The SVM/SOFTMAX classifier

Input: image X_{ground} captured by the ground robot

Output: class label C defined in Problem Formulation

1 Recognizing X_{ground} by the ground robot classifier, getting a tuple ($PG_{straight}$, PG_{left}, PG_{right}) (the possibility of certain label)and a confidence Ψ

2 **if** Ψ < limit, **then**

3 Seeking help from the UAV,UAV capture the image X_{aerial} and send it to cloud

4 Recognizing X_{aerial} by the aerial robot classifier, getting a tuple ($PA_{straight}$, PA_{left}, PA_{right})

5 Sending the new list [$PG_{straight}$, PG_{left}, PG_{right}, $PA_{straight}$, PA_{left}, PA_{right}] to the SVM/SOFTMAX classifier

6 The SVM classifier outputs label C_{svm}

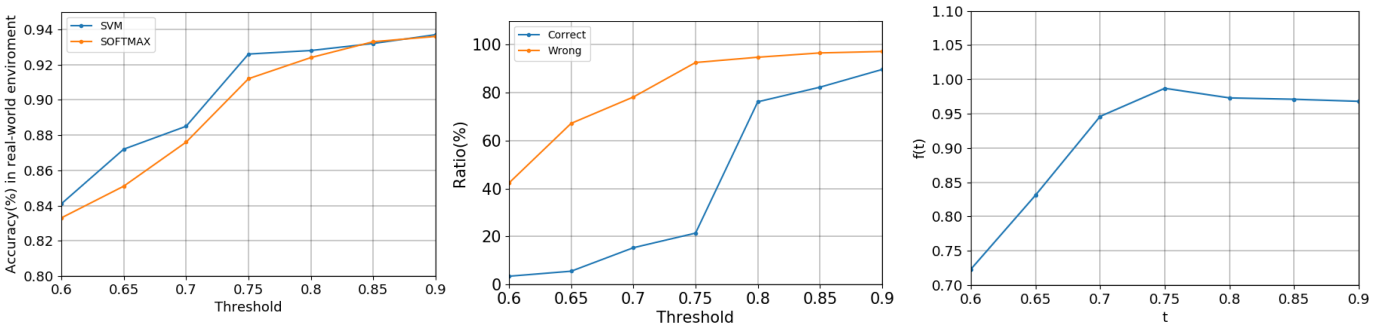
7 $C = C_{svm}$

8 **else**

9 $C = C_{ground}$

10 **end if**

11 **return C**



a) The accuracy of "SVM/SOFTMAX" b) The ratio of samples under different thresholds c) f (t) under different thresholds

Fig. 5. Experiment results for the fix of "Threshold"

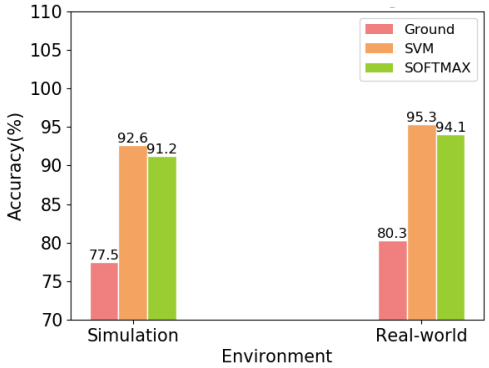


Fig. 6. The accuracy of "SVM"/"SOFTMAX" algorithm

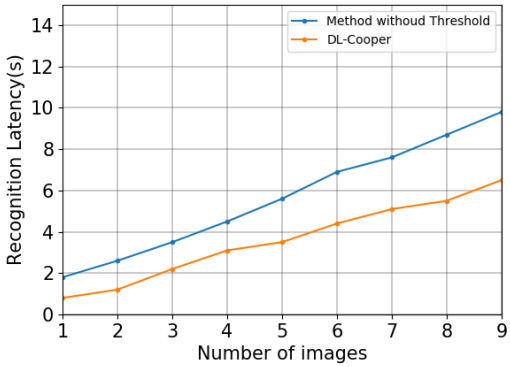


Fig. 7. Recognition latency comparison

V. Experimental Results

We evaluate our work through a set of experiments on the data we collected in simulation and real-world environments in order to demonstrate how well different algorithms perform in contrast with the two basic ones.

A. Dataset

We solve the problem as a supervised machine learning task, which is extremely challenging because of the wide appearance variability of the trail and its surroundings: perceptions are heavily affected by lighting conditions, vegetation types, altitude, local topography and many other factors. We deal with such challenges by gathering a large and representative labeled dataset, covering a large variety of trails and a long distance on each.

The dataset in the simulation environment is acquired in Gazebo 8.0 of robot operation system (ROS).To generate the training dataset, we control the quadrotor using the keyboard. When the quadrotor flies in the low altitude (1.3m), it collects the data representing the views of the ground robot. When it flies in the higher height (21.3m), the sight of it stands for the aerial data. The dataset in the real-world environment is acquired in a village. We use a quadcopter and a digital camera to collect the data for the aerial and ground robot using the three commands. To be concrete, the quadrotor is controlled to follow the trail by an expert pilot with an extremely low speed. Every 5 seconds, the quadrotor rotates left and right 90 degrees respectively from the center line of the trail. Meanwhile, an expert driver holding the digital camera follows the quadrotor all the time below facing the same direction. The pitch angle between them is 60 degrees in order to compensate the height difference. Both the quadcopter and the digital camera are turned to recording mode and the images are captured and matched later. Our quadcopter is the Parrot Bebop Drone and from the Bebop's stream connection, we receive an image of the resolution of (1920×1080) pixels, the same as that from the digital camera. The dataset covers approximately 5 kilometers of hiking trails acquired at altitudes ranging from 300m to 900m, different times of the day and weather. Meanwhile, many different trail types and surroundings are represented, ranging from sloped narrow alpine paths to wider forest roads.

Each image is labeled by an expert driver, associated with its ground true class. We also augment the training dataset by synthesizing left/right mirrored versions of each training image. In detail, a mirrored training image of class TR (TL) yields a new training sample for class TL (TR). A mirrored GS training sample yields another GS training sample. Additionally, mild affine distortions (±10% translation, ±15% rotation, ±10% scaling) are applied to training images to increase the number of samples further. To train a classifier robust to noise, we generate additional images and augment our dataset size by adding Gaussian white noise of mean 0 and variance 0.01 to our dataset. Therefore, the size of our final dataset is twice larger than the previous one. The dataset has been split in disjoint training (12320frames) and testing (4855 frames) sets. The split was defined by carefully avoiding that the same scene appears in both the training and testing set.

B. Fix the Value of "Threshold"

In order to fix the suitable value of "Threshold", we firstly test the accuracy of the two algorithms under different thresholds. The time delay is definitely proportional to the threshold because the images to be dealt with are increasing. From Figure 5.a, we found that the accuracy is also proportional to the threshold. So we decide to find the point that best measures the confidence of the classifier according to its classification results. We divide the samples in the testing dataset into two kinds, "Correct" and "Wrong" based on the classified result. Towards each kind, we give the propagation of the samples, the max probabilities of which are under the corresponding threshold. (Figure 5.b) And the threshold t needs to satisfy the condition: the number of the "Correct" samples the confidence of the ground robot on which is larger than t is as many as possible; the number of the "Wrong" samples the confidence of the ground robot on which is less than t is as many as possible. According to this, we need to find the threshold t that maximizes the function below:

$$f(t) = \underset{t}{argmax}\left(\frac{(1-CL(t))WL(t)}{CL(t)(1-WL(t))}\right) \quad (7)$$

Here, $CL(t)/WL(t)$ means the number of samples the confidence of the ground robot on which is less than t in the "Correct" /"Wrong" dataset. From Fig 5.c, we can see that 0.75 makes the function 7 maximum, so we set the threshold to 0.75 because it can best reflect the "Landmark" of the ground robot's classifying ability.

C. Experiments on the Algorithms

In this subsection, we test the "SVM" and "SOFTMAX" algorithms in two methods. The first one is to use the testing dataset in order to show the improvement of accuracy. The second one is to implement our method on a real platform with the aim of showing the effectiveness of our approach and the optimization of time delay compared with the method that not using the "Threshold".

To be noticed, in these two methods, we should firstly train the SVM/SOFTMAX classifier using the training dataset as the final classifier. The mode of SVM is one to the rest because our problem has three labels. The ratio between the training and testing size is 6:4. The kernel function which we use to cut the plane best is "RBF." And the parameters of it (C=5e3 and Gamma=0.005) are obtained from two basic sets, we choose all kinds of pairs to see which one has the best performance. Considering the fact that the "SOFTMAX" classifier uses time seed to give random digitals, we do the experiments ten times and calculate the average performance.

1) Experiments using the testing dataset

We test the two algorithms using our testing dataset in order to show the improvement of accuracy under the best threshold 0.75. From Figure 8, we can clearly see that in the simulation environment, the accuracy of the "SVM" and "SOFTMAX" algorithms increase significantly from 77.5% to 92.6% "SVM" and 91.2% "SOFTMAX". In the real-world environment, the accuracy also reaches 95.3% "SVM" and 94.1 "SOFTMAX", much higher than the previous one (80.3%).This shows that the possibilities extracted by the classifiers have their own value and can be supplemented with each other. And it also verified our suppose that the view from global can make up the limitation of local views. Of course, this should also be attributed to the success of the machine learning approach.

2) Implementation on a real platform

We use a TurtleBot [24], a wheeled mobile robot and the Parrot Bebop Drone to perform the trail following task. And they are directly connected to a DELL R430 server which stands for the cloud via Wi-Fi. The server stores the ground classifier and the aerial classifier which have been trained using the real-world dataset. We implemented a simple reactive controller which translates the "SOFTMAX" algorithm's output to control signals as follows. Yaw (i.e. steering) is proportional to P (TR)-P (TL) ; a positive value steers the robot to the right, and a negative value steers the robot to the left. Speed is proportional to P (GS). When the confidence of the classifier on the TurtleBot is less than 0.75, it would send a message to the server for help. The server then received the image from the Parrot Bebop Drone and deal with them using "SOFTMAX" algorithm. The value of Speed and Yaw was then sent to both the TurtleBot and the Parrot Bebop Drone. When the TurtleBot is confident enough to make decision alone, it would calculate the control signals based on its own classifier (saved on the TurtleBot) and then send them to the Parrot Bebop Drone through the server. The two robots will send signals to the cloud immediately when they finish the instant movements with the aim of coordination.

We record the recognition latency as the TurtleBot moves forward. In order to show the optimization of the time delay, we repeat the last experiment using the method without "Threshold", which is to perform decision fusion algorithms on the cloud all the time and record the results. As shown in Figure 7, the latency of the method without "Threshold" is relatively high and increased in a constant speed. The DL-Cooper method is more predictable because its latency is lower and this validated the Theorem 2 effectively.

The main problem we observed during our tests in realistic conditions is that in the forested environments, the obstacles (i.e. branches of the trees) may block the way of the Parrot Bebop Drone. The quadrotor needs to be coordinated with the TurtleBot, which means it would take the same action as the TurtleBot. Unfortunately, the action may have a bad result on the quadrotor. We also observed that the ground robot is often unable to follow the trail if there is not enough free space besides the trail centerline. This may be caused by the rough translation of the signal because the classifier compensates a lateral shift only when the robot is about one meter off the center line in our experiments. But in other relatively open environments with wide trails, the ground robot and quadrotor were able to successfully follow the trail for a few hundred of meters.

VI. Conclusion

This paper presented a new method to realize trail following in the wild environment by the CNN-based supervised learning network with the aid of cloud. The proposed modular network architecture provided a convenient way to transfer and update in the future. By fusing the decisions of the ground robot and the aerial robot, the exploration task was accomplished effectively. The test results in the simulated environment and real-world environment showed that our method could achieve great accuracy improvement with the aid of aerial data.

However, several accessible aspects should be considered in the future to improve it. The whole framework should be implemented to train end-to-end. That means there should be no separated classifier. We can also use the unsupervised techniques to finish this task in order to improve the ability to adapt to new environments.

References

[1] Giusti, Alessandro, et al. "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots." IEEE Robotics & Automation Letters 1.2(2017):661-667.

[2] C. Rasmussen, Y. Lu, and M. Kocamaz, “Appearance contrast for fast, robust trail-following,” in Proc. IROS, pp. 3505–3512, 2009.

[3] Santana, Pedro, et al. "Tracking natural trails with swarm - based visual saliency." Journal of Field Robotics 30.1(2013):64–86.

[4] Toet, Alexander. "Computational versus Psychophysical Bottom-Up Image Saliency: A Comparative Evaluation Study." IEEE Transactions on Pattern Analysis & Machine Intelligence 33.11(2011):2131-46.

[5] J. Schmidhuber and R. Huber, “Learning to generate artificial fovea trajectories for target detection,” International Journal of Neural Systems, vol. 2, no. 1 & 2, pp. 135-141, 1991

[6] Pomerleau, Dean Arthur. "Neural network perception for mobile robot guidance." Springer International 239(1992).

[7] Lecun, Yann, et al. "Off-road obstacle avoidance through end-to-end learning." International Conference on Neural Information Processing Systems MIT Press, 2005:739-746.

[8] Pomerleau, Dean A. "Efficient Training of Artificial Neural Networks for Autonomous Navigation." Neural Computation 3.1(2014):88-97.

[9] Bojarski, Mariusz, et al. "End to End Learning for Self-Driving Cars." (2016).

[10] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," IEEE Transactions on Automation Science and Engineering, vol. 12, pp. 398-409, 2015..

[11] Google, “Maps trekker.” www.google.com/maps/about/partners/streetview/trekker.

[12] Tai, L., and M. Liu. "Mobile robots exploration through cnn-based reinforcement learning:." Robotics & Biomimetics 3.1(2016):24.

[13] S. Ross, et al., “Learning monocular reactive uav control in cluttered natural environments,” in Robotics and Automation (ICRA), 2013 IEEE International Conference on. IEEE, 2013, pp. 1765‑1772.

[14] J. Sergeant, et al., “Multimodal deep autoencoders for control of a mobile robot,” in In Proc. of Australasian Conf. for Robotics and Automation (ACRA), 2015.

[15] Krizhevsky, Alex, I. Sutskever, and G. E. Hinton. "ImageNet classification with deep convolutional neural networks." International Conference on Neural Information Processing Systems Curran Associates Inc. 2012:1097-1105

[16] Kelly, Alonzo, et al. "Real-Time, Multi-Perspective Perception for Unmanned Ground Vehicles." 2003.

[17] Vandapel, Nicolas, R. Donamukkala, and M. Hebert. Experimental Results in Using Aerial LADAR Data for Mobile Robot Navigation. Field and Service Robotics. Springer Berlin Heidelberg, 2006:103-112..

[18] M. Waibel, M. Beetz, J. Civera, R. D Andrea, J. Elfring, D. Galvez-Lopez, et al. "A world wide web for robots," IEEE Robotics & Automation Magazine, vol. 18, pp. 69-82, 2011.

[19] G. Mohanarajah, D. Hunziker, R. D'Andrea, and M. Waibel, "Rapyuta: A cloud robotics platform," IEEE Transactions on Automation Science and Engineering, vol. 12, pp. 481-493, 2015.

[20] A. Saxena, A. Jain, O. Sener, A. Jami, D. K. Misra, and H. S. Koppula, "RoboBrain: Large-Scale Knowledge Engine for Robots," arXiv preprint arXiv:1412.0691, 2014.

[21] R. Arumugam, V. R. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. F. Kong, et al., "DAvinCi: A cloud computing framework for service robots," IEEE International Conference on in Robotics and Automation (ICRA), 2010, pp. 3084-3089.

[22] G. Mohanarajah, V. Usenko, M. Singh, R. D'Andrea, and M. Waibel, "Cloud-based collaborative 3D mapping in real-time with low-cost robots," IEEE Transactions on Automation Science and Engineering, vol. 12, pp. 423-431, 2015.

[23] L. Riazuelo, J. Civera and J. Montiel, "C 2 TAM: A Cloud framework for cooperative tracking and mapping," Robotics and Autonomous Systems, vol. 62, pp. 401-413, 2014.

[24] TurtleBot, http://www.turtlebot.com/, 2015.