A Deep Learning based Framework for UAV Trajectory Pattern Recognition

Xingyu Pan, Pascal Desbarats, Serge Chaumette

A Deep Learning based Framework for UAV Trajectory Pattern Recognition

Xingyu PAN
Univ. Bordeaux, CNRS, Bordeaux INP
LaBRI, UMR 5800
F-33400, Talence, France
xingyu.pan@labri.fr

Pascal Desbarats
Univ. Bordeaux, CNRS, Bordeaux INP
LaBRI, UMR 5800
F-33400, Talence, France
pascal.desbarats@labri.fr

Serge Chaumette
Univ. Bordeaux, CNRS, Bordeaux INP
LaBRI, UMR 5800
F-33400, Talence, France
serge.chaumette@labri.fr

***Abstract*—Recognizing the trajectory patterns of Unmanned Aerial Vehicles (UAV) allows to identify their missions also to detect abnormalities for intelligent supervision. To take advantage of the developments of deep learning on image classification, we proposed a method that converts 3D trajectory data into 2D images and trains a deep network adapted to sketch-like images. We achieved a promising recognition rate of 99.5% on the database of simulated UAV trajectory images.**

***Keywords*—pattern recognition, Unmanned Aerial Vehicles (UAV), trajectory, deep learning**

I. Introduction

Unmanned Aerial Vehicles (UAV), also known as drones, are widely used in a variety of civil and military missions. Meanwhile, increasing misuses of UAVs challenge public and military security. It is therefore important to develop an intelligent system to detect and supervise UAV missions since a similar human process has multiple limitations such as cost, tiredness and accuracy. To address this problem, several solutions were proposed in 2017 in *ShafeShore* "bird-vs-drone challenge" to distinguish UAVs from birds in video records. Most approaches using deep learning for object detection are based on the assumption that UAVs and birds have different shapes [1]. However, this assumption is not always true. Furthermore, simply detecting the shape of a UAV is not enough to analyze its behavior, which is required for intelligent supervision. In this paper, instead of detecting UAV by its shape, we recognize UAV missions by using trajectory data with a deep learning framework. Notice that in real life the end-to-end pipeline contains trajectory acquisition, reconstruction and recognition. Here we mainly focus on the recognition phase.

Our contributions can be summarized as follows:

- It is the first study to recognize UAV mission patterns based on trajectory data.
- We proposed methods to represent 3D-space trajectory data in 2D images so as to reduce data and to transfer the problem into image recognition; we also build a database of simulated trajectory images.
- We proposed an adapted and light deep learning method to recognize sketch-like trajectory images resulting from the above 3D to 2D conversion with very promising and interesting results.

A. Related work

Trajectory data mining has been a hot topic for many years. Object movement trajectories are often considered as sequential or spatio-sequential patterns [2]. Therefore, Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) models are widely studied in various applications for tracking and prediction [3] [4]. However, in our context we focus on spatial patterns of trajectory data because: first, a UAV mission (exploitation, delivery, *etc.*) is usually associated to a completed trajectory from taking off to landing; second, spatial patterns in form of images are ideal data to take advantage of the great performance of Convolutional Neural Networks (CNN) as used in computer vision. We call image representation of trajectory data in terms of trajectory images. Recently, more and more trajectory images are used to classify diverse moving objects, such as migrant animals [5], game players [6] or urban transportation modes [7], but not yet for UAVs. A trajectory image can be binary, simply containing sparse or connected coordinates, or using grayscale or color to encode more information. For example, Endo *et al.* used intensity to present moving speed, according to the numbers of GPS records within a grid [7]; Singh *et al.* employed two colors to represent a pair of trajectories [6]; Yang and Gidofalvi encoded directional information of a trajectory set by multiple channelled images [8]. Despite different image generation methods, trajectory images are like sketches with abstract features. Compared to textural photos as in ImageNet and to binary characters as in MNIST, sketch-like images have neither texture nor color block.

Since 2012 CNN-based architectures have dramatically improved the performance of image recognition notably in the competition of ImageNet, but most deep learning models are designed for images with rich textural information [9]. For free-hand sketch recognition, Eitz *et al.* built *TU-Berlin* sketch database and investigated classic object recognition methods by extracting local features [10]. In addition, Sangkoly *et al.* proposed *Sketchy* database which is the first large scale collection of sketch-photo pairs for sketch-based image retrieval as well as sketch recognition [11]. With the development of CNN in recent years, deep networks and their variations are investigated to improve the accuracy of sketch recognition. Yu *et al.* used a customized CNN architecture called Sketch-a-Net which beat the human recognition rate of 73% on *TU-Berlin* database [12]. By using a residual network, Sedatti *et al.* achieved an accuracy of 79% on *TU-Berlin* benchmark and a success rate of 93% for sketch retrieval on *Sketchy* database [13]. Larger filters, larger pooling size, higher dropout and removing Local Response Normalization (LRN) are empirical measures to increase performance and to reduce computational cost for sketch-like images.

B. Reminder of the paper

The rest of the paper is organized as follows: Section II describes the data preparation including trajectory simulation, class definition and image generation. Section III presents our CNN-based approaches adapted to UAV trajectory images. Section IV shows different experiments carried out on our trajectory database and similar sketch database, and analyzes the results. In the final section, conclusions and perspectives are drawn.

II. Data preparation

A. Trajectory simulation

Collecting a large UAV trajectory database and annotating with mission-based categories is a complex job. Thanks to Drone Harmony (DH)[1], a drone data capture tool with a mission planner, we are able to simulate UAV missions on synthetic scenes. During the simulation, a series of GPS waypoints are generated in an automatic way by selecting a predefined mission. Thus, by exporting generated waypoints, we can obtain simulated trajectories labeled by mission.

Instead of using fully unreal scenes, we created at first two synthetic scenes, Bordeaux city center and campus of University of Bordeaux (see Fig. 1), with almost real GPS coordinates powered by OpenStreetMaps, in order to guide a drone to fly in corresponding real areas in the future. Then we chose a predefined category of DH mission planner to simulate a UAV missions. Among all predefined mission categories, we preselected 5 classes with typical flying patterns as follows:

- *Facade inspection (fcd):* mission for photometry and close-up inspection of walls that have high 3D detail, such as balconies or sculptures. The trajectory could go either horizontally or vertically.

- *Helix tower inspection (hlx):* mission mainly for inspecting and collecting data around single tall structures, for example cell phone towers. The trajectory is characterized by spiral movement.

- *Orbit mapping (obm):* mission for generating 3D models of large structures as well as accurate point clouds of terrain. The trajectory is featured by overlapped loops.

- *Perimeter (per):* mission for capturing one or more structures from the sides. The trajectory could follow the outline of the targets or use approximation perimeters like circles, ellipses or convex hulls.

- *Top-down mapping (tdm):* mission for creating orthomosaics of mostly flat structures. The trajectory is mainly featured by its sinusoidal undulation.

[1]https://droneharmony.com

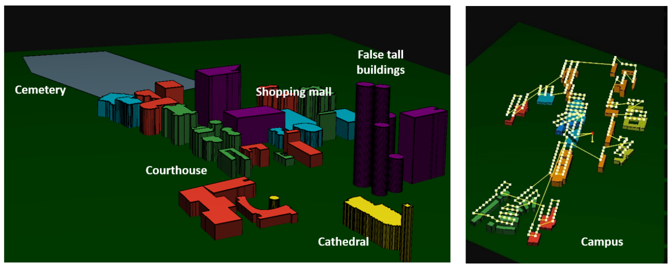

Fig. 1. Synthetic scenes of Bordeaux city center (left) and campus of University of Bordeaux (right)

B. Image generation

The exported trajectory data consists of ordered GPS waypoints given by $\mathbf{P} = \{p_i\}_{i=0}^{N}$, where N denotes the number of total waypoints in the trajectory. Let $p_i(x_i, y_i, z_i)$ be a given point in a 3D space where x_i, y_i and z_i represent respectively its latitude, longitude and altitude. Assuming that a trajectory pattern depends on the relative positions of waypoints rather than absolute GPS locations, we normalize absolute GPS values of each axe between 0 to 255. The normalized trajectory is denoted by $\hat{\mathbf{P}}(\hat{\mathbf{X}}, \hat{\mathbf{Y}}, \hat{\mathbf{Z}})$. In the previous studies of trajectory images, trajectories are always plotted on a plane perpendicular to the ground. However, altitude variations cannot be overlooked for UAVs. Since using 3D representation is resource consuming, here we proposed two methods to generate 2D images from the normalized trajectories.

1) 1-channelled XY-images: according to our observation, in most cases UAV movements on XY-plane are more dominant than on other planes. We plot $(\hat{\mathbf{X}}, \hat{\mathbf{Y}})$ of $\hat{\mathbf{P}}$ into the image I_{XY} of 256 × 256 pixels. Each normalized waypoint $\hat{p_i}(\hat{x_i}, \hat{y_i}, \hat{z_i})$ indicates a trajectory pixel at the position $(\hat{x_i}, \hat{y_i})$ in I_{XY}. As familiar with black sketch on white paper, we set background as white and intensity of trajectory pixels from 255 to 0 inversely proportional to $\hat{z_i}$. Instead of plotting only scattered GPS records, we connect adjacent trajectory pixels with linear transitional intensity. Thanks to the normalization step, all trajectories are centred in the image with the same size. Examples are shown in Fig. 2 (top).

2) 3-channelled XYZ-images: to make each axe presents equal information, we plot successively three planes (XY, YZ and YZ) into the same image I_{XYZ} of 256× 256 pixels and separate them by different colors. For example, we keep using black for XY-trajectory, then use red and blue for YZ-trajectory and XZ-trajectory, respectively. Since the information of all planes is included, we do not need to vary intensity to add redundancy. Hence I_{XYZ} is composed of three superposed trajectory projections in constant colors. Examples are shown in Fig. 2 (bottom).

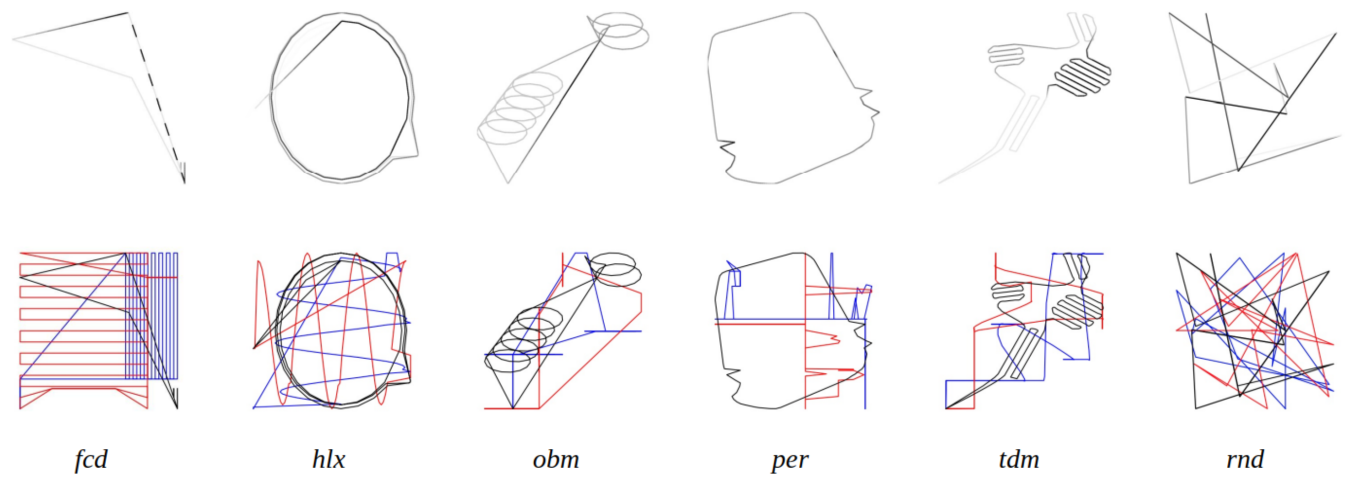

Fig. 2. Examples of trajectory images (top: 1-channelled XY-images; bottom: 3-channelled XYZ-images)

Notice that the trajectory generation of DH mission planner is not fully automated. Manual input of a bunch of parameters (UAV model, relative altitude, overlap percentage, *etc.*) is necessary for each simulation. In consequence, creating thousands or more trajectory images in the same way becomes extremely tedious. Hence we performed data augmentation to obtain enough adequate data for deep learning. We proposed basic transformations including rotation, horizontal or vertical flip, swirl distortion, background noise and partial erasing. We randomly applied one or multiple above transformations to generate different noise degrees. Fig. 3 shows XY-trajectory examples created by data augmentation from one DH simulated trajectory.

C. *Unknown trajectories*

In real applications not all input trajectories belong to the limited number of predefined classes. It is therefore important to detect unknown trajectories produced by UAVs flying in unrecognizable ways, or by other flying objects like birds. Currently without acquisition or simulation for such unknown trajectories, we created stochastic trajectories as noise data in the following way.

Given $\mathbf{P}\{(x_i, y_i, z_i)\}_{i=0}^{N}$, we set N as an random integral between 10 and 30, then make x_i, y_i and z_i vary randomly between 0 and 255. Stochastic trajectory images were generated in both methods that we proposed. Those stochastic trajectories are labeled as random *rnd* (see Fig. 2).

D. *Data summary*

Finally, we created 3904 trajectory images in total: half are 1-channeled XY-images and half are 3-channeled XYZ-images. Each image type contains 1954 trajectories including 1604 known trajectories partitioning into 5 categories, and 350 unknown stochastic trajectories. Among those known trajectories, 104 images are generated by DH simulation and the rest are created by data augmentation. Data distribution is quasi-equal with about 320 images per known class. We named this database *simTrj*, short for simulated trajectories.

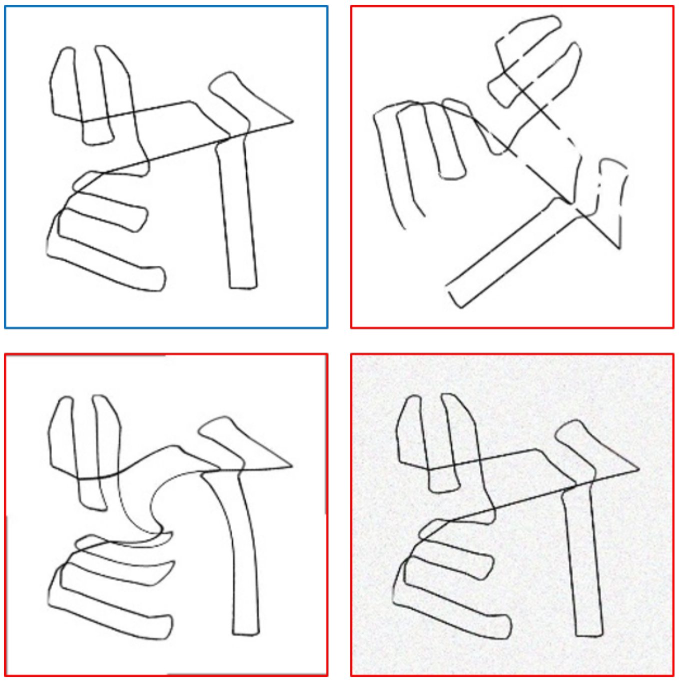

Fig. 3. Original 1-channelled XY-trajectory image (in blue) and examples generated by data augmentation (in red)

III. Deep learning methodology

In this section we introduce our CNN architecture adapted to trajectory images. A general structure is composed of convolutional, pooling and fully connected layers. In spite of the developments of deep learning so far, it remains an open question how to design the entire architecture for a specific visual recognition task. Based on the state of the art to similar data and on our experiments, AlexNet appears to be the most suitable baseline architecture for our context rather than other deeper networks. According to our tests, similar to Sketch-a-Net, removing LRN layers tends to augment performance on our trajectory images. However, other proposed improvements in Sketch-a-Net such as using larger filters or larger pooling size tend to decrease the accuracy. We call our network SkalexNet since it is a modified version of AlexNet designed and tuned for sketch-like trajectories.

In our context, recognition of unknown trajectories is as important as classification of known patterns. Intuitively, the softmax output describes categorical probability distribution. We may think in a native way to treat softmax output as kind of confidence score. Hence even if the test sample is predicted as one of the training classes, it will be considered as unknown if the corresponding softmax output below a certain threshold. However, previous studies have already pointed out that this naive approach is not robust to imperceptible perturbations [14]. Therefore, we proposed two approaches to deal with unknown trajectories based on our deep network models.

A. Multiclass classification

In this approach, we consider all unknown trajectories belonging to a novel class *rnd* and train a traditional multiclass classifier. Having N known classes, the outputs of the network is $N+1$. Stochastic trajectories in the *simTrj* database are also used for supervised learning. This approach can be used in condition that we have enough unknown data which are totally random but discriminant compared to all known classes.

B. One-vs-rest classification

One-vs-rest classification consists of N separate binary classifiers. Each class is used as positive training data for one time and other classes as negative ones. After training the separated networks, we apply N classifiers to a test sample and predict it according to the classifier which gives a positive response with the highest confidence score. If all classifiers reports negative response, the sample should out of the known classes. We assume that the negative training data from other known classes are random enough to provide a robust discriminator for a given class. The principal advantage of this approach is no need of learning "out-of-set" data. Although it is computationally costly, only training limited classifiers is acceptable in practice.

IV. Experiments and results

A. Experiments

In this section, we evaluated our methods on the database *simTrj*. We chose Caffe (Convolutional Architecture for Fast Feature Embedding) as deep learning framework since it is easy to code and supports GPU acceleration [15]. AlexNet and SkalexNet, the variation of AlexNet for sketch-like images by removing LRN layers, were tested. To achieve the best performance, hyperparameters of networks should be tuned carefully and empirically. Concretely, using Stochastic Gradient Descent (SGD), we set training batch size as 35, learning rate starting from 0.001 then reducing gradually with a step of 2500 iterations. We trained the whole network from scratch.

We started from multiclass classification on different datasets. Datasets *simTrj2gray* and *simTrj2color*, for 1-channelled XY-images and 3-channelled XYZ-images respectively, contain only images of 5 known classes, while datasets *simTrj3gray* and *simTrj3color* include stochastic trajectories in addition. For each dataset, we used 200 images per class as training set and all the rest as test set. Cross validation was proceeded for all datasets.

Then, we performed one-vs-rest classification 5 times for every known class. Even having more negative samples than positive ones, we still chose randomly 200 per class training data (400 in total) and put all the rest as test data. Stochastic trajectories are unlabeled test data not used for training process but for testing trained models.

At last, we tested the same networks on the *Sketchy0* database, a preselection of *Sketchy* database containing 4184 freehand sketches in 7 classes (apple, bottle, cup, fish, pear, sword and teapot). The goal is to check the generalizability of our networks on similar data. We intentionally preselected trajectory-like sketches, so sketches with complicated and trivial details were not considered. Notice that interclass similarity of *Sketchy0* is relatively high even for human recognition (see Fig. 4). For experiments, we divided the *Sketchy0* database into 3500 training data (500 per class) and 684 test data.

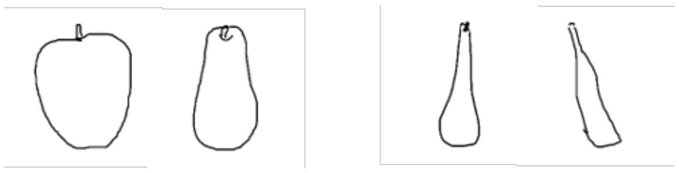
Fig. 4. Examples of similar sketches: apple, pear (left); pear, bottle (right)

B. Results

The current results summarized in Table I are quite promising. By training from scratch we achieved an accuracy of more than 97% for all tests. For 5 known classes of *simTrj*, the best result we obtained is 99.5% by using SkalexNet on 3-channelled XYZ images. In general, SkalexNet outperformed traditional AlexNet, which shows that LRN layer in the classic CNN architecture for brightness normalization is not effective for sketch-like images. In addition, we can see that XYZ-images are generally better than XY-images. It is logical because XYZ-images contain more altitude information than XY-images, especially for class *fcd* and *hlx*.

Notice that all reported results are average values obtained by cross validation. We do observe under some training-test divisions, using AlexNet or XY-images may obtain higher accuracy than using SkalexNet and XYZ-images. Moreover, even for given training and test data, the results may vary within a small range for the same test due to random process of computation. An example of learning curve is shown in Fig. 5.

TABLE I
RESULTS ON DIFFERENT DATASETS OF *simTrj* AND ON *Sketchy0*

Database	**N. class**	**N.train**	**N.test**	**AlexNet**	**SkalexNet**
SimTrj2gray	5	1000	604	98.5%	98.8%
SimTrj2color	5	1000	604	99.2%	99.5%
SimTrj3gray	6	1200	754	97.8%	97.5%
SimTrj3color	6	1200	754	98.4%	98.6%
Sketchy0	7	3500	684	98.3%	98.9%

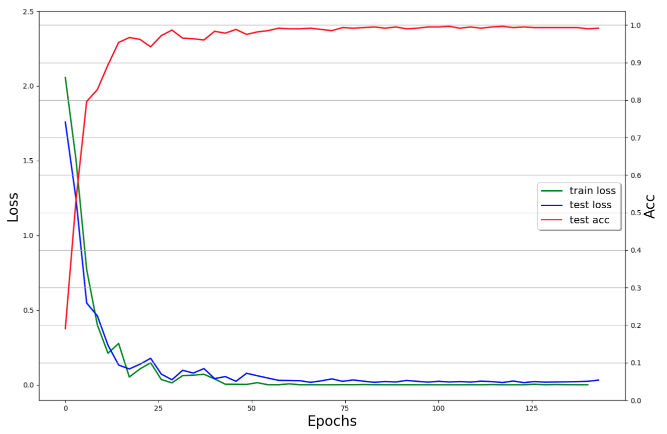

Fig. 5. Learning curve of SkalexNet on *simTrj2color*

To see which classes are more easily confused with others, we computed the confusion matrix denoted by M_c. The diagonal of M_c represents correct classification rate and all values outside the diagonal represent false recognition rate. Table III and Table II show respectively M_c obtained by multiclass classification and by one-vs-rest classification, using skalexNet on dataset *simTrj3color*. Interestingly, for the one-vs-rest classification, more than one third of stochastic trajectories are wrongly recognized as the class *tdm* (top-down mapping). However, for multiclass classification including the class *rnd*, confusions between classes are very few. We would like to say that the multiclass classification do find a decent discrimination between *tdm* and *rnd*, while the one-vs-rest classifier for *tdm* is not enough to discriminate top-down trajectories from random ones.

TABLE II
CONFUSION MATRIX OBTAINED BY MULTICLASS CLASSIFICATION

	fcd	**hlx**	**obm**	**per**	**tdm**	**rnd**
fcd	99,06 %	0,00 %	0,00 %	0,00 %	0,94 %	0,00 %
hlx	0,00 %	99,09 %	0,00 %	0,00 %	0,91 %	0,00 %
obm	0,87 %	0,00 %	94,78 %	2,61 %	0,87 %	0,87 %
per	0,00 %	0,00 %	0,00 %	100,00 %	0,00 %	0,00 %
tdm	2,19 %	0,00 %	0,73 %	0,00 %	97,08 %	0,00 %
rnd	1,33 %	0,00 %	0,67 %	0,00 %	0,00 %	98,00 %

The problem may be caused by our data. We hoped that every one-vs-rest classifier extracts the most discriminant features for the corresponding class, but we observed that top-down mapping trajectories have a higher interclass variability than "stochastic" ones. Precisely, overall top-down trajectories may be totally unlike if one or more factors (relative altitude, overlap rates, one target or several sparsely distributed targets, *etc.*) are changed. On the contrary, our "stochastic" trajectories in form of random polylines share obvious intraclass similarities even though we attribute random values to number and positions of waypoints. Moreover, human can quite easily recognize a top-down mapping by its symbolic XY-undulation patterns, but for deep networks it is closer to a real random class than our "stochastic" trajectories. To verify this observation, we tested several bird migrant trajectories (see Fig. 7) on the 6-class well-trained model. Not surprisingly, most of them were recognized as the class *tdm* instead of *rnd*.

TABLE III
CONFUSION MATRIX OBTAINED BY ONE-VS-REST CLASSIFICATION

	fcd	**hlx**	**obm**	**per**	**tdm**	**rnd**
fcd	98,67 %	0,00 %	0,00 %	0,00 %	0,67 %	0,67 %
hlx	0,00 %	100,00 %	0,00 %	0,00 %	0,00 %	0,00 %
obm	0,00 %	0,00 %	94,67 %	0,00 %	4,67 %	0,67 %
per	0,67 %	0,00 %	0,00 %	94,67 %	1,33 %	3,33 %
tdm	0,00 %	0,00 %	0,00 %	0,00 %	100,00 %	0,00 %
rnd	0,29 %	0,00 %	0,00 %	0,00 %	36,86 %	62,86 %

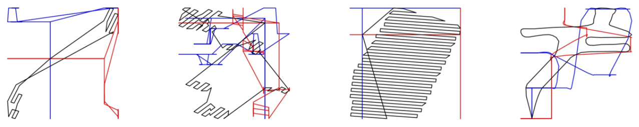

Fig. 6. Interclass variability of top-down mapping trajectories (XYZ-images)

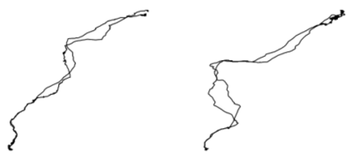

Fig. 7. Examples of migrant trajectories of birds (XY-images)

C. Discussion

According to the results, even the overall accuracy is quite encouraging, detection of unknown samples especially among the confusing top-down mapping trajectories remains a great challenge.

1) Trajectories vs. freehand sketches: despite of similarities between trajectory images and sketches, there are still some differences between them for recognition tasks. Only for databases *simTrj* and *Sketchy0* used for experiments, trajectories are intuitively less ambiguous than sketches for human recognition by its symbolic patterns that we described in the Section II. According to our observation, sketches are mainly depicted by global features while certain trajectory patterns are characterized by local features. For example, it is hard to recognize an apple only from a leaf-like drawing, because it may be a leaf of an apple, or a fish fin, but all apple drawings are more or less globally similar. In contrast, some trajectory categories such as orbit mapping or top-down mapping can be recognized by its local features (overlapped loops or undulations) even though they are globally quite different. The current network performs generally well for both trajectories and sketches but can still be optimized and specialized for one of them.

2) Limitations of current database: firstly, to develop robust deeper networks and to compare with previous work on large sketch databases (Berlin-TU and Sketchy), the database *simTrj* needs to be largely extended in scale, even though it is not that small compared to related work on trajectory images. Another drawback of using small database is that the differences are slight between results obtained by good solutions and the best solution.

Secondly, in the current database the majority of data were created by data augmentation. Although by doing this we increased the size of database efficiently and improved the ability of generalization of the fit model, too many augmented data derived from much fewer original images might overfit the model to the current database.

Moreover, the way we created ”stochastic” trajectories is too naive to represent all unknown cases. It is worthwhile to find a better way to generate real random noise trajectories, for example chaos theory models.

Finally, the proposed 3-channelled XYZ-image is a decent 2D representation but may be not enough for more sophisticated mission patterns. As like in the related studies, it is possible to encode more information such as speed, direction, recurrent position, *etc.*, through a more comprehensive image representation. Furthermore, it is also interesting to study image generation with different weights of three superposed planes.

3) Limitations of CNN models: SkalexNet derived from AlexNet baseline model is simple but proficient on the current database. However, for extended database and more sophisticated image representation as we discussed above, we will probably need deeper and more complicated networks.

V. Conclusion and future work

In this paper we have presented a method recognize UAV mission patterns from raw GPS trajectories. Due to lack of published data, we built the *simTrj* database of simulated 2D trajectory images, and achieved a promising accuracy of 99.5% by using SkalexNet, a concise CNN architecture tuned for sketch-like trajectories.

As one of the first studies to analyze mission-based UAV trajectories, the future work will be multifold. From the aspect of data, we need to simulate more trajectories and annotate them by more and finer categories with help of powerful simulation tools and UAV experts; and for real UAV trajectory data, trajectory acquisition (camera, radar, *etc.*), data preprocessing and constraints in real environment need to be investigated. For detecting unknown trajectories, one-vs-rest classification will be practically impossible when we have too many classes, so outputting robust confidence scores instead of softmax becomes a trend for open set recognition problems [16]. In addition, to distinguish UAVs from birds, we also need to simulate or acquire corresponding databases of bird and UAV trajectories in similar conditions, for example, hovering above a building. However, most published bird trajectory data are only migrant patterns. Furthermore, in the frame of this project, we will extend our work from single trajectory to a set of trajectories for multi-agent systems.

Acknowledgment

This study has been carried out with financial support from the French State, managed by the French National Research Agency (ANR) in the frame of the ”Investments for the future” Programme IdEx Bordeaux - SysNum (ANR-10-IDEX-03-02).

References

[1] A. Coluccia, M. Ghenescu and et al. ”Drone-vs-bird detection challenge at IEEE AVSS 2017,” in IEEE Int. Conf. on Advanced Video and Signal Based Surveillance, 2017, pp.1–6.

[2] Y. Zheng ”Trajectory data mining: an overview,” ACM Trans. on Intelligent Systems and Technology, 2015, vol.6(3), pp.29, 2015.

[3] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, L. and S. Savarese, ”Social LSTM: Human trajectory prediction in crowded spaces,” in IEEE Conf. on CVPR, 2016, pp. 961-971.

[4] X. Jiang, E. N. de Souza, A. Pesaranghader, B. Hu, D. L. Silver, and S. Matwin, ”TrajectoryNet: an embedded gps trajectory representation for point-based classification using recurrent neural networks,” in 27th Int. Conf. on Computer Science and Software Engineering, 2017, pp. 192-200.

[5] S. Wang, ”Exploring ocean animal trajectory pattern via deep learning,” M.S.thesis, 2016.

[6] K. Y. Singh, N. Davis, C. P. Hsiao, M. Jacob, K. Patel, and B. Magerko, ”Recognizing actions in motion trajectories using deep neural networks,” in 12th Artificial Intelligence and Interactive Digital Entertainment Conference, 2016.

[7] Y. Endo, H. Toda, K. Hishida, and J. Ikedo ”Classifying spatial trajectories using representation learning,” Int. J. of Data Science and Analytics, 2016, vol.2, pp.107–117.

[8] C. Yang and G. Gidofavi, ”Classification of regional dominant movement patterns in trajectories with a convolutional neural network,” Spatial Big Data and Machine Learning in GIScience, 2017, 17.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ”Imagenet classification with deep convolutional neural networks,” in Advances in Neural Information Processing Systems, 2012, pp.1097-1105

[10] M. Eitz, J. Hays, and M. Alexa, ”How do humans sketch objects?” ACM Trans. on Graphics, 2012, Vol.31, pp.44-1.

[11] P. Sangkloy, N. Burnell, C. Ham, and J. Hays, ”The sketchy database: learning to retrieve badly drawn bunnies,” ACM Trans. on Graphics, 2016, vol.35, pp 119.

[12] Q. Yu, Y. Yang, F. Liu, Y. Z. Song, T. Xiang, and T. M. Hospedales, ”Sketch-a-net: A deep neural network that beats humans,” Int. J. of Computer Vision,” 2017, vol.122, pp.411-425.

[13] O. Seddati, S. Dupont, and S. Mahmoudi, ”Deepsketch 3: Analyzing deep convolutional neural networks for better sketch recognition and sketch-based image retrieval,” Multimedia Tools and Applications, 2017, vol.76, pp.22333-22359.

[14] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, ”Intriguing properties of neural networks,” 2016, arXiv preprint arXiv:1312.6199.

[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, ... and T. Darrell, ”Caffe: Convolutional architecture for fast feature embedding,” in 22nd ACM Int. Conf. on Multimedia, 2014, pp.675-678

[16] A. Bendale and T. E. Boult, ”Towards open set deep networks,” in Int. Conf. on CVPR, 2016, pp.1563-1572.