Pedestrian Tracking from an Unmanned Aerial Vehicle

Chao Bian, Zhen Yang, Tao Zhang, and Huilin Xiong
Department of Automation, Shanghai Jiao Tong University, Shanghai, China
Email: {bianchao, yangzhen5771, sjtu--zt, hlxiong}@sjtu.edu.cn

***Abstract*—In this paper we present a scheme for pedestrian tracking from an unmanned aerial vehicle (UAV), which includes the motion control of the UAV, and the visual tracking of a specific pedestrian from the moving platform. In the visual tracking part, we use an online updating feature queue and the Locality-constrained Linear Coding (LLC) method to match the pedestrian target. The ground station receives video stream from the UAV, and sends back commands to control the UAV's motion. If one pedestrian is specified as the target when the UAV is hovering, the UAV will track the pedestrian and keep a fixed distance from it. We only use visual information, which comes from the single front camera of the flying robot; no any GPS device is used. A Parrot AR Drone 2.0 and a Parrot Bebop 2 are adopted in the experiments, which are carried on in outdoor conditions, and experimental results verify the effectiveness of our scheme.**

Keywords*—*pedestrian tracking; unmanned aerial vehicle; feature queue; Locality-constrianed Linear Coding

I. Introduction

In recent years small-sized unmanned aerial vehicles are becoming increasingly prevalent, which are gradually equipped with various functions besides taking photos. Visual tracking research has made substantial progress in the past decade, and numerous trackers have been developed and are well-known for their robustness or high speed. Naturally, the application of computer vision on UAVs arises [1-5], visual tracking included. One of the common visual tracking tasks is pedestrian tracking. The process of pedestrian tracking from an UAV is initialized with an image patch containing the target, and the goal is to find the specific pedestrian in subsequent frames and keep a fixed distance between the target and the UAV.

A vision-based control strategy for tracking and following objects with an UAV is introduced in the paper [6], and the tracking method adopted in that paper is TLD [7], which is known for its idea of tracking-learning-detection, and the UAV used in that article is a Parrot AR Drone 2.0. However, for many tracking methods used by UAVs currently, when the pedestrian target is shaded by other pedestrians partly, or the other pedestrians are dressed up similarly, the UAV often fails to catch the target. We observe that some modifications may be necessary when implementing usual visual tracking methods on UAVs in order to get better performance on the moving platform.

A pedestrian tracking method for UAVs is introduced in this paper. We adopted a Parrot AR Drone 2.0 when we started the work, and the initial control program was developed based on the official SDK. For better performance, we made a switch to a Parrot Bebop 2 after it came out. The Robot Operating System(ROS) is well-known for its capability and convenience, and our later work is based on it. Both UAVs we use are closed, and we need to send commands via wireless network to move them.

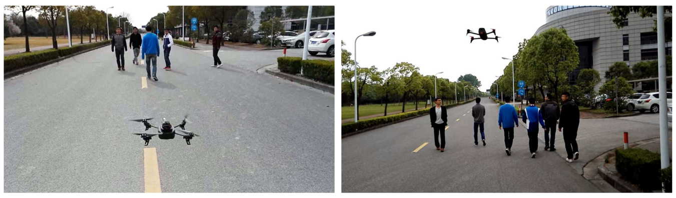
(a) (b)

Fig. 1. (a) Tracking with a Parrot AR Drone 2.0, whose front camera is set straight ahead. (b) Tracking with a Parrot Bebop 2, which is equipped with a front camera inclining downwards.

The tracking method we develop is based on detection, and we prepare a dedicated feature descriptor for the case of pedestrians. In terms of matching, we introduce a way of sparse reconstruction using the LLC method based on a feature queue which is updated online with certain rules we may talk about, and that's mainly where our contribution of this paper is.

II. Description in Detail

In this section, we describe our method in detail. Firstly, we introduce the control strategy we adopt, which has been improved in numerous experiments and validated to be effective. Secondly, the pedestrian detection module is described, and it is based on the classical combination of the Histogram of Oriented Gradients (HOG) [8] and the Support Vector Machine (SVM) [9]. We just show the steps and some differences we make about the size of the images used to train the model. Next, a dedicated feature descriptor is prepared for pedestrians. Finally, we describe the queue-based matching process thoroughly.

A. Motion Control

By typing keys on the laptop, we move the UAV to some point after it is started, and it keeps hovering there. When the interested pedestrian appears in the view of the UAV, we specify the target by drawing a bounding box containing it roughly, and then the tracking process goes automatically. Subsequent bounding boxes are calculated by the tracking module if possible. If no target is found in one frame, the UAV will receive a hovering command and the searching region will be expanded according to some rules.

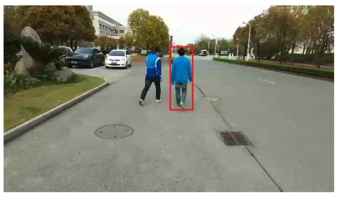

Fig. 2. One frame of the tracking process. The found target pedestrian is marked exactly with a red bounding box.

The UAV is a flying robot running freely, thus we need to implement control measures in both horizontal and vertical directions. On every frame of the video stream, the target pedestrian is expected to be at the center of the image, though some shift in the vertical direction is allowed on the occasions that we want the UAV to fly high. When the target moves towards left or right, the UAV rotates horizontally and approaches simultaneously.

Using one single front camera, the UAV tries to maintain a fixed distance between the target and itself. Supposing the height of the image is H and the height of the bounding box h, the distance is estimated by the value of h/H. The width information is ignored for the reason that when the target makes a turn, the width change is something that may cause trouble if we take the area of the bounding box to calculate the distance. The median filtering method is used to reduce the effect of any sudden change.

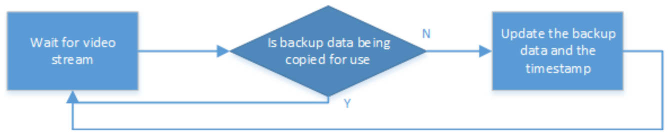
Wait for video stream
Is backup data being copied for use
N
Update the backup data and the timestamp
Y

Fig. 3. The flow chart of the receiving thread, which receives video stream from the UAV and communicates with the auto-control thread.

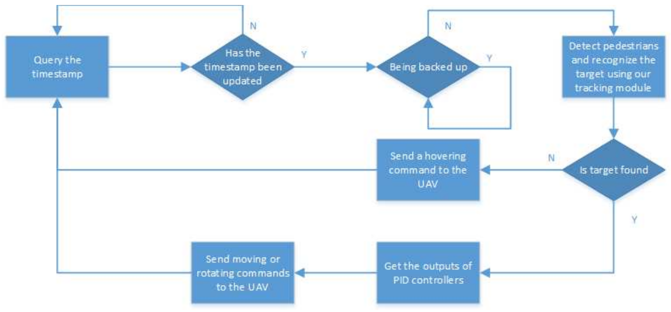
Query the timestamp
N
Has the timestamp been updated
Y
N
Being backed up
Y
Detect pedestrians and recognize the target using our tracking module
Send a hovering command to the UAV
N
Is target found
Y
Send moving or rotating commands to the UAV
Get the outputs of PID controllers

Fig. 4. The flow chart of the auto-control thread, which is the main working thread after the auto-control mode is started.

Three PD controllers are designed to move the UAV. One of them takes the charge of rotation, while another one is responsible for moving the UAV forwards and backwards, and the last one ensures that the bounding box is generally in the middle of the image vertically. PD controllers are chosen for their quick response to situations and their simplicity.

B. *Pedestrian Detection*

The tracking method we develop is based on the output of the detection module. The classical combination of the HOG feature and the SVM classifier is used to detect pedestrians. We prepared 4171 positive samples, which were manually cut out from the INRIA dataset, and 3890 negative samples cut out from the INRIA dataset randomly, and another 1100 negative samples cut out randomly from some scenic photos of China. The training steps go as follows:

a) Extract the HOG features of the positive samples and the negative samples respectively.

b) Get the initial SVM model by training with the output of the last step.

c) Try to detect pedestrians in the negative samples with the model. Extract the wrongly-detected patches and save them.

d) Add the wrongly-detected patches into the negative samples, and train the model again.

e) Tune the parameters and try more times.

The *size* of our positive samples is 24x64. We have also tried the *size* 48x128, but the former turns out to be better when it comes to small pedestrians, and 3:8 is the ratio we conclude from experiments that contains little background information.

C. *Feature Extraction*

Only color information is used in our method, which is a combination of RGB and Lab. In the case of Lab, the channel L is discarded to reduce the effect of illumination changing.

On every frame, we try to find all the pedestrians by using the SVM model we trained, and the detection goes globally only if necessary, when the target is lost for example. Generally, we detect the pedestrians in a local region around the center of the target in the last frame.

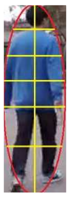

Fig. 5. Extracting the color feature of a pedestrian. The inscribed elliptical area is divided into stripes and then blocks. The upper part and the lower are processed with different techniques.

Lisanti et al [10] points out that focusing on the elliptical area containing the pedestrian is simple yet effective. Given a patch of pedestrian, we suppose the height of the patch is h

and the width is w, then set $a = w/2$ and $b = h/2$. We take the center of the patch as the origin, and $f(x, y)$ denotes the pixel value of the point (x, y):

$$f(x,y)=\begin{cases} f(x,y), \dfrac{x^2}{a^2}+\dfrac{y^2}{b^2}\le 1, \\ 0, othersize. \end{cases} \tag{1}$$

Divide the patch of the pedestrian into two parts. The upper half denotes the upper part of the body, and the lower half denotes the lower part. The division ratio is set to 0.55 in our experiments. For the upper half, divide it into 4 equal stripes vertically and then divide each stripe into 2 equal blocks horizontally. For the lower half, divide it into 2 stripes vertically and then divide each stripe into 2 blocks horizontally. For the upper half, we use Gaussian distribution function to change the weight of each point according to the x value of it:

$$g(x)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{2}$$

We set μ=0 and σ=1 in our experiments. Take $g(x)$ as the statistical weight. For each block, extracting the RGB features and the "ab" features respectively, where "ab" represents the a channel and the b channel of Lab color space. We use a 6×6×6 cube to store the RGB values of every block, and a 15×15 square to store the "ab" values of each block. The outputs of each block are resized into a vector. Chaining the vectors of the blocks, we get the color feature descriptor of the upper half.

For the lower half, we do not use any kernel function and just chain the RGB features and the "ab" features of each block, because the area around the vertical midline of the lower half usually contains too much background information. The lower half differs from the upper half notably.

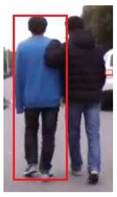

Fig. 6. A diagram of the importance of the usage of blocks. If the target pedestrian is shaded by another pedestrian partly, the order of blocks in a stripe matters.

We divide each stripe into blocks since we learn from experiments that when two or more pedestrians are close to each other and the target pedestrian is among them, the order of cubes and squares in a stripe matters.

D. Queue-based Matching

We introduce a queue to store the feature descriptors of recently highly matched yet not exactly matched pedestrians. Sparse reconstruction is implemented upon the queue in real time, and we decide whether to trust the current patch and whether to update the queue according to the value of the reconstruction error.

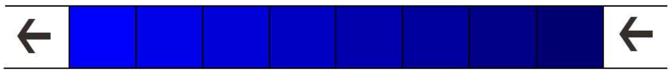

Fig. 7. The feature queue. The queue is used by the LLC method to measure the distance between a pedestrian and the queue itself, and it is updated by highly matched feature descriptors online.

After we specify the target pedestrian with a bounding box, the tracking mode starts, and the feature descriptor of the image patch in the box is added into the tail of the queue. During a short period of time subsequently, calculate the Euclidean distances between the feature descriptor of the to-be-checked pedestrian and the feature descriptors in the queue from the pedestrian who is closest to the center of the recognized target in the last frame, and make a census of matched times. If the number of matched times is not smaller than a certain ratio of the current size of the queue, 50% for example, the searching work will stop in the current image and the patch will be trusted to be the target and go on otherwise. Furthermore, if the number of matched times is not smaller than a higher ratio of the current size of the queue, 80% for example, the feature descriptor of the patch will be pushed into the queue. Before the size of the queue gets to the maximum, 15 for example, no feature descriptor will come out of the queue. This is the procedure that the queue is constructed, and it usually gets finished within several seconds.

Once the queue is full, the rules change. Sequence the pedestrians found in one frame by the distances between the center of the patches and the center of the recognized target in the last frame, and check the pedestrian patches one after another. Supposing the feature descriptor of the current patch is y and the matrix consisting of the current feature descriptors in the queue is C, we use the LLC criteria [11] and replace the l_1-norm constraint with a l_2-norm constraint and then we get :

$$\min_{\alpha} \| \mathrm{y}-C\cdot\alpha \|_2^2 + \lambda \sum_i \left(\alpha_i \cdot \exp\left(\frac{\| y - C_i \|_2}{\sigma} \right) \right)^2, \tag{3}$$

$$s.t.\ \mathbf{1}^T\alpha = 1. \tag{4}$$

Here λ controls the sparsity degree and σ is used to adjust the decay speed of the locality adaptor. Then we get the response coefficient vector α, and calculate the reconstruction error e:

$$e=\frac{\| y-C\cdot\alpha \|_2}{\| y \|_2}. \tag{5}$$

Compare the error with the threshold we set, and deem the patch as the possible target if the error is not greater than the threshold. If the number of remaining to-be-checked pedestrian patches are not less than 2, repeat the calculation twice and find the smallest error, and if there is only one patch left, calculate the construction error of it and get the smaller. The pedestrian patch with the smallest reconstruction error is trusted to be the target, if the error is not greater than the threshold of course. Furthermore, if the error is small enough, within 50% of the previous threshold for example, and not too close, beyond 3% of the previous threshold for example, preventing the feature descriptor of the patch being almost the same with one in the queue, push the corresponding feature descriptor into the queue. To ensure that the recognized patch be the best locally, we calculate several errors and take the smallest one, and this is quite useful when the target and the neighboring pedestrians are similar.

The advantage of introducing the queue is that it makes the matching be more precise. Meanwhile, because the queue is updated online, and the changing of the appearance of the target pedestrian, which may be caused by his posture or the illumination, go frame by frame, the queue can hold the changes, or put it another way, it can learn.

The tracking module receives frames from the control module, and send results back. The location of the last recognized patch is stored in the tracking module, and it is sent back to the control module if no target is found in one frame, with a flag indicating the incident.

III. Experimental Results

Three datasets which were collected by ourselves are used in the experiments. Dataset A was collected with a Parrot AR Drone 2.0 in order to test our tracking method, and dataset B and C were from real tracking experiments with a Parrot Bebop 2. In dataset A, some images are blurred, and this may happen sometimes during the communication between the UAV and the ground station, and a tracking method for unmanned aerial vehicles should bear that.

In dataset A, which contains 2351 images, being a sequence of frames, the target and another pedestrian are similarly dressed up. When the target pedestrian walks, the other pedestrian shades the target from the view of the UAV from time to time. In dataset B, which contains 2212 images, the pedestrians start in an open space, make a sharp turn and go down the road, and turn into a parking lot at last. Dataset C is made of 2312 images and it takes place besides a building, and when the target walks around, the other pedestrians try to interfere with the progress. The width of any image in the datasets is 640 in pixels and the height is 368. The CPU used in our experiments is an Intel Core I7 4720HQ.

Four performance indexes are used to compare the tracking methods. The first one is the average error of the distance between the calculated center of the target pedestrian and the real one, which is measured in pixels, and the second index is the maximum error. The following index is the percentage of frames at which the corresponding tracking method fails to find the target. Frames per second, or FPS, is the last index, which indicates the speed of a tracking method.

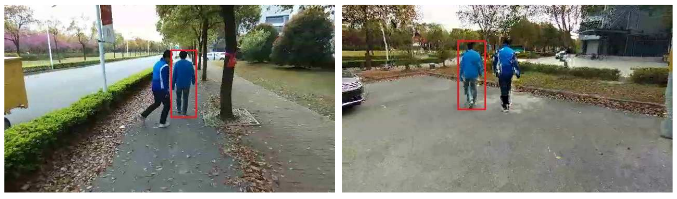
(a) (b)

Fig. 8. (a) The view of the UAV at one frame of the dataset A. (b) The view of the UAV at another frame of the dataset A; this frame is blurred due to incomplete video stream transmission.

TABLE I. Performance on Dataset A

Tracking Methods	Average Error	Maximum Error	Lost Frames	FPS
MIL	202.20	421.88	84.64%	5.74
BOOSTING	136.91	369.93	75.83%	38.65
MEDIANFLOW	36.44	124.87	44.68%	134.44
TLD	48.86	391.99	27.15%	4.87
KCF	32.71	164.31	22.60%	107.04
Ours	5.40	113.51	3.06%	16.80

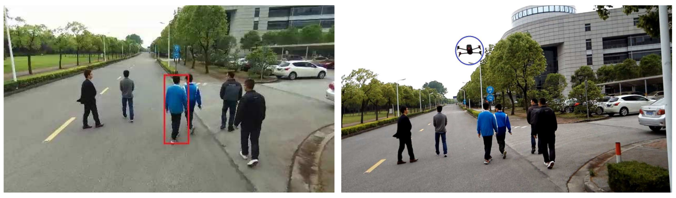
(a) (b)

Fig. 9. (a) The view of the UAV at one frame of the dataset B. (b) The view of a bystander recording the tracking process with a camera of a cell phone. (a) and (b) happened at the same time.

TABLE II. Performance on Dataset B

Tracking Methods	Average Error	Maximum Error	Lost Frames	FPS
MIL	51.25	322.00	13.66%	5.07
BOOSTING	30.24	261.44	13.89%	28.88
MEDIANFLOW	59.95	110.49	77.75%	131.70
TLD	45.95	364.33	31.03%	8.75
KCF	11.33	44.48	0	152.01
Ours	3.22	98.59	0.36%	15.82

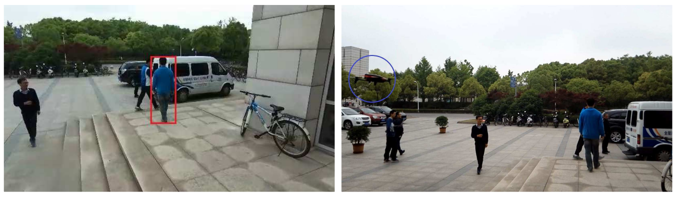
(a) (b)

Fig. 10. (a) The view of the UAV at one frame of the dataset C. (b) The view of a bystander recording the tracking process with a camera of a cell phone. (a) and (b) happened at the same time.

TABLE III. Performance on Dataset C

Tracking Methods	Average Error	Maximum Error	Lost Frames	FPS
MIL	248.40	359.52	87.32%	6.93
BOOSTING	205.67	357.17	74.95%	33.36
MEDIANFLOW	223.33	283.92	97.45%	136.92
TLD	54.33	352.41	24.88%	7.86
KCF	198.42	511.03	52.32%	179.41
Ours	4.12	76.78	0.04%	15.57

The experimental results show that our method is slower than some fast methods such as MEDIANFLOW and KCF [12], but its speed is still fine. The average error of our method is quite small, which means that our method usually finds the exact location of the target pedestrian, which is very important in real tracking scenes. The larger the maximum error is, the more possibly the UAV loses the target. The percentage of the lost frames of our method is also small enough.

In real-world situations, the UAV often fails to track the target pedestrian if the maximum error is too large or if the method fails to find the target for many frames, because the UAV will turn towards the wrongly found pedestrian or just hover and the real target may walk or run away. Also, if the average error is too large, the UAV often fails to keep a fixed distance between the target and itself, which in turn may cause the tracking to fail.

IV. Conclusions

In this paper we present a pedestrian tracking scheme with a new tracking method based on a feature queue which is updated online. The speed-improved Locality-constrained Linear Coding method is used based on the queue to get a more precise matching. Our tracking method is equipped with the ability of learning due to the characteristics of queue, and it is robust to the changing of the posture of the target and the illumination if the changes grow gradually. The effectiveness of our method has been verified by experiments carried on under different weather conditions and in different seasons.

The construction of the feature of one pedestrian is a large topic itself, and in our method only color information, or to be exact, RGB and Lab, is used dedicatedly. Some other features such as HOG, tends to be of little use if occlusion problems are complicated enough in real situations. We may have a better solution in future work to improve our tracking method.

Compared to fast methods such as KCF, our method is slower due to the detection module we use, which is dependent on the HOG feature and the SVM classifier. We trained the model ourselves, with small pedestrian patches cut out containing little background information in order to detect small pedestrians exactly. Since we detect the pedestrians locally and the whole image is used only if the target is lost, our method is still fast enough with a speed of about 15 FPS. Smaller detection region will improve the speed, but it may weaken the robustness.

Though our method is designed for pedestrian tracking, other objects such as cars are also possible if we change the construction of the feature descriptor. The matching process based on the LLC method with a queue is a general approach. The experiments can be more persuasive if an unmanned aerial vehicle equipped with a powerful camera and a strong anti-wind ability is available.

V. Acknowledgements

This work was supported by the National Natural Foundation of China under grant no. 61375008.

References

[1] B. Yu, L. Wang, Z. Niu, M. Shakir, and X. Liu, "Constructions detection from unmanned aerial vehicle images using random forest classifier and histogram-based shape descriptor," *Journal of Applied Remote Sensing,* vol. 8, pp. 083554-083554, 2014.

[2] H. Zhou, H. Kong, L. Wei, D. Creighton, and S. Nahavandi, "Efficient road detection and tracking for unmanned aerial vehicle," *IEEE Transactions on Intelligent Transportation Systems,* vol. 16, pp. 297-309, 2015.

[3] R. Montanari, D. C. Tozadore, E. S. Fraccaroli, and R. A. Romero, "Ground vehicle detection and classification by an unmanned aerial vehicle," in *2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR)*, 2015, pp. 253-258.

[4] B. Jeon, K. Baek, C. Kim, and H. Bang, "Mode changing tracker for ground target tracking on aerial images from unmanned aerial vehicles (ICCAS 2013)," in *Control, Automation and Systems (ICCAS), 2013 13th International Conference on*, 2013, pp. 1849-1853.

[5] J. Maier and M. Humenberger, "Movement detection based on dense optical flow for unmanned aerial vehicles," *International Journal of Advanced Robotic Systems,* vol. 10, 2013.

[6] J. Pestana, J. L. Sanchez-Lopez, P. Campoy, and S. Saripalli, "Vision based gps-denied object tracking and following for unmanned aerial vehicles," in *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2013, pp. 1-6.

[7] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE transactions on pattern analysis and machine intelligence,* vol. 34, pp. 1409-1422, 2012.

[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, pp. 886-893.

[9] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning,* vol. 20, pp. 273-297, 1995.

[10] G. Lisanti, I. Masi, A. D. Bagdanov, and A. Del Bimbo, "Person re-identification by iterative re-weighted sparse ranking," *IEEE transactions on pattern analysis and machine intelligence,* vol. 37, pp. 1629-1642, 2015.

[11] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010, pp. 3360-3367.

[12] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 37, pp. 583-596, 2015.