Research On Obstacles Avoidance Technology For UAV Based On Improved PTAM Algorithm

Guanghua Bai, Xiaojia Xiang, Huayong Zhu, Dong Yin, and Lei Zhu

College of Mechatronics and Automation
National University of Defense Technology
Changsha, Hunan, China

bgh0918@163.com

***Abstract* -When used in counter-terrorism and disaster rescue, UAVs may possibly fly in buildings. In an indoor closed environment, GPS signal may be blocked, which will make UAVs difficult to obtain their location. In addition, a complex indoor environment also brings challenges to the safety of flight. Therefore, it is necessary to identify complex environments and avoid obstacles in real-time. This paper introduces the PTAM algorithm to the UAV ground control module. To deal with poor illumination of indoor environment, less feature points and other issues, we improve the algorithm and design a self-adaptive map feature point generating mechanism, reducing the dependency of the algorithm on the number of feature points and illumination conditions. On this basis, this paper proposes an obstacle recognition algorithm and designs a warning mechanism for the ground control module. Finally, we use a small quad-rotor UAV to test this algorithm. The results show that the proposed method can provide warning to the obstacles during flight in unknown indoor environments with self-localization. The experiments demonstrate that this algorithm is effective in real-time, which has great significance to explore autonomous positioning and flight safety for UAVs in GPS denied environments.**

Index Terms - UAV; Ubuntu; Feature points; Obstacle Warning; PTAM

I. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) can perform many tasks in structured or unstructured environments, such as exploration and mapping of multi-story buildings, manipulation and transportation of objects, surveillance, and search and rescue. Usually, GPS (Global Positioning System) information is needed to achieve self-localization during flight in real-time. Operators monitor drone's flight status from ground station, so they can control drones avoid obstructions timely, which is really helpful to the safety of flight. However, there are some limitations in this mode. UAVs can acquire good GPS signals outside the buildings while they are not able to get their position information because of the block of signal inside the buildings. Furthermore, the data-link between drones and ground-based station may disconnect. As a result, operators will not know the environment around UAVs clearly and it's hard to control UAVs to avoid obstacles, which will finally cause potential dangers in their flight. In order to deal with such situations above, we need to design methods to localize drones without dependence on traditional GPS based positioning techniques and develop new methods to avoid obstacles autonomously. By doing these we can increase the safety of drones when they fly in a GPS denied indoor environment.

978-1-4673-9088-0/15/$31.00 ©2015 IEEE

Simultaneous Localization and Mapping (SLAM) is an self-localization method in mobile robot domain. That means robots construct maps of unknown environment autonomously and realize localizing themselves using this map in meantime. During the procedure of mapping and localizing simultaneous, robots can complete autonomous localization relying on various sensors on board only , without the necessity of acquiring a priori of working environment in advance. Among all the SLAM algorithms, PTAM algorithm which utilizes visual sensors is applied to UAV domain gradually. With the development of computer vision technology and the continuous improvement in capability of on board CPU, the positioning method for UAV based on PTAM becomes mature day by day.

In this paper, we design a self-adaptive landmark generation mechanism used in the situations that feature points are hard to get when using PTAM algorithm due to the poor illumination conditions. Moreover, we make experiments on how to detect suspicious obstacles around drones during their flights according to the distribution of landmarks in the map and generate obstacles warning information for drones when they fly inside the buildings.

II. BACKGROUND

SLAM is one of the most important problems in mobile robotics domain [1]. A creative work about SLAM began with a research on the estimation of space uncertainty done by R.C. Smith and P.Cheeseman in 1986[2]. It seems simple to state but challenging to solve: "How can a mobile robot operate in an a priori unknown environment, using only on-board sensors to simultaneously navigate and build a map of its workspace?" The difficulty of SLAM lies in the uncertainty inherent in a robot's interactions with real world through sensors and actuators with noises.

Solutions to SLAM are especially important in providing mobile robots with the ability to achieve real autonomy. For robots in hostile environments like collapsed buildings or the surface of other planets, SLAM is a significant technology which helps navigation. The number of service robots in commercial use is substantial and growing fast (Taking medical service and warehouse management as well as rescue and environment monitoring systems for example), forming a strong incentive to use SLAM in these challenging domains. Low-cost consumer-level robots for home-management and

entertainment are now also becoming a reality. Without SLAM, robots will cost much more and are incapable of changes in their work environment. SLAM offers a viable way to make mobile robotics an all-pervasive and truly useful technology.

Researches on SLAM with visual sensors like cameras have become a hot issue owing to the population of digital camera devices like mobile phones and its reasonable prices [3].There is a method called Monocular SLAM derived from visual SLAM domain and many researchers have acquired many creative achievements. MonoSLAM is a solution of Extended Kalman Filter (EKF) based SLAM which is proposed by professor Andrew J. Davison by using only one consumer level camera [4]. This algorithm maintains full camera and feature covariance and limited to Gaussian uncertainty. Later prof. Davison makes a more complete description of Monocular SLAM including augmented reality and humanoid robot implementations in collaboration with other researchers [5]. Prof. Davison and his group have achieved a series of eye-catching successes in this research area [6]. Contemporary researchers Eade and Drummond from Cambridge University employ a FastSLAM2.0 filter algorithm and they use particle filter for camera pose hypothesis as well as kalman filter for extracted feature points, leading to maintain a much larger map [7][8]. These two Monocular SLAM systems both are incremental mapping methods: tracking and mapping are intimately linked, so current camera pose and the position of every landmark are updated together at every single video frame with tremendous sum of computation. In 2007, Georg Klein and David Murray form University of Oxford propose a novel SLAM method named Parallel Tracking and Mapping (PTAM)[9]. Different from previous two methods, this system splits tracking and mapping into two separate tasks which processed in parallel threads on a dual-core computer: one thread deals with the task of robustly tracking swaying hand-held motion of camera, while the other core produces a 3D map of point features from previously observed video frames. At the same time, it employs a batch process optimization which makes this method capable of mapping a relatively detailed workspace with a few thousand tracked feature points. Since the appearance of PTAM, many researchers have achieved progresses with further efforts in domains like augmented reality and feature based tracking. Among all these newly proposed methods, Dense Tracking and Mapping (DTAM) is an outstanding frame [10]. DTAM is a system for real-time camera tracking and reconstruction which relies not on feature extraction but on a dense pixel-based method. DTAM achieves real-time performance using current commodity GPU hardware, and the dense model they create permits superior tracking performance under rapid motion compared to the state of the art methods using feature-based method. Pollefeys et al. [11] propose a 3D reconstruction method using mobile phone camera, which can create absolute scale dense 3D model of the scene and provide the users a real-time interaction to feed back. By using the inertial sensor integrated in the phone, the system can deal with fast motion of the phone during the tracking and mapping procedure and can estimate the metric scale of the scene acquired. They adopt an accurate and effective mechanism for dense stereo match, which promises to reduce the processing time below the interaction time.

Faced with the booming of the robots like humanoid robots and UAVs, there is a need to create a unified software platform for simulation and application. The Robot Operating System (ROS) is a flexible framework for writing robot software. It's not an independent operating system and relies on host system like Ubuntu in which ROS owns some already formed software repositories. ROS is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. From the robot's perspective, problems that seem insignificant to humans often become challenges to robots. As a result, ROS is built to encourage collaborative robotics software development. ROS provides seamless integration with popular open source projects like GAZEBO, OpenCV and PCL along with a message passing system that allows users to easily swap out different data sources, from live sensors to log files.

What we try to do is to make UAVs have the capacity of positioning themselves in indoors environment. We try to make use of both PTAM algorithm and ROS development environment to testify all our methods which later proved to be efficient. However, just like Georg Klein and David Murray have found during their experiments that when camera moves fast, this will cause high level of movement blur which leads to the loss of most of the corner features thus the tracking will fail. Moreover, without enough corners detected, stereo matches will fail to initialize the system. Besides, the map generated contains only some point cloud data and they never try to extract geometric information from it and make further use of that. These are the same troubles we will confront and it will be more challenging to fulfill our plans. We will reveal our solutions in the next chapters.

III. METHOD

A. Improvement for PTAM

PTAM algorithm is implemented by researchers who hold a camera move around. By visual tracking it provides registration of the work space without a prior model. Researchers split tracking and mapping into two threads. Tracking thread performs responsible estimation of camera pose and rendering augmented graphics which should be made as robust and accurate as possible. While mapping thread is responsible for providing the map and should be made as rich and accurate as possible. During tracking process, when system receives a key frame, it will create a four-level image pyramid of greyscale 8 bit per pixel. Then Features from Accelerated Segment Test (FAST) corner detection algorithm will be executed on each pyramid level [12].The system adopts a constant velocity model to update camera pose, which can be represented as follows:

$$v_t = \left(p_t - p_{t-1}\right) / \Delta t \tag{1}$$

$$p_{t+1} = p_t + \Delta t'(v_t) \quad (2)$$

Where p_t is the previous camera pose and p_{t-1} is the current camera pose, while Δt is the time step between these two pose. Then p_{t+1} is the estimated camera pose and $\Delta t'$ is the time interval between current camera pose and its estimated one. Then the system needs to find map points in the current frame with a fixed-range search method around the points' predicted location. By generating 8*8 matching template which is warped from source key frame, the system will search for a fixed radius around projected position in the target pyramid level using zero-mean sum of squared distance (SSD) and search at FAST corner points only. Especially, patch search and pose update are done twice from coarse to fine and system estimates the quality of tracking at every frame with a recovery procedure when tracking is considered lost. As for mapping thread, there are also many tasks to perform to make the algorithm work well. The system employs a five-point stereo algorithm to initialize the map with requirement for researchers to click keyboard twice for two key frames with smooth motion of camera, which provides the system a pair of frames and feature correspondences. As a result, a sparse map is initialized. In order to grow the map and have a more comprehensive understand of the work space, the system needs more key frames. Key frames are only added if there is a sufficient baseline to other key frames while tracking quality is at least good. Whenever a key frame is added, the mapping thread stops whatever it is doing and all points in the map are measured in the key frame. Once new map points are found, they will be added to the map. In order to have a detailed map, the system needs as many points as possible. However, the system needs only salient points as candidates to be new map points. In order to satisfy this requirement, the system will check all the maximal FAST corners in current key frame with Shi-Tomasi score based thresholding and inspect whether some points have already been in the map. Then the system implements epipolar search in a neighbouring key frame to get depth information for new map points. After being triangulated when a match has been found, the map point will be inserted into the map. All these procedures are repeated in four image pyramid levels. At the same time, the system uses a structure form motion (SFM) batch method Bundle Adjustment which seeks to minimize a geometric cost function (e.g., the sum of squared re-projection errors) by jointly optimizing both the camera and point parameters using non-linear least sqares. After this, the map is continually refined and expanded by the mapping thread as new key frames are added by tracking thread.

In order to apply PTAM algorithm to UAV visual positioning, we must know the relations between different coordinates involved in the algorithm. There are three coordinates chosen to represent objects from real world to image plane: a world coordinate system W, a camera-centred coordinate system C and an image coordinate system I. A feature point in world coordinate system is represented in a homogeneous coordinate form: $p_{iW} = (x_{iW}\ y_{iW}\ z_{iW}\ 1)^T$. When we finish camera calibration, we can translate feature points from world frame W to image frame I. In order to complete this transform, we first transform them from world coordinate W to camera-centred coordinate C. This is done with a 4×4 matrix denoted as M_{CW}. M_{CW} is determined by the relative bearing of camera to the world coordinate system and is known as the camera extrinsic parameter matrix. Then the translation formula can be written as: $P_{iC} = M_{CW} P_{iW}$ (3)

with $p_{iC} = (x_{iC}\ y_{iC}\ z_{iC}\ 1)^T$. Next we will project points in camera coordinate system into image coordinate and a calibrated camera projection model matrix M_{cam} is used. Here is the formula:

$$z_{iC}\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{dy} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0^T & 1 \end{bmatrix} P_{iW} \quad (4)$$

$$= \begin{bmatrix} a_x & 0 & u_0 & 0 \\ 0 & a_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0^T & 1 \end{bmatrix} P_{iW} = M_{cam} M_{CW} P_{iW}$$

In this formula, $a_x = f/dx$ and $a_y = f/dy$. As for dx and dy, they represent the metric magnitude of each pixel in X and Y axis while (u_0, v_0) is the coordinate of the point where the camera optical axis intersects with image plane. The matrix M_{cam} is completely determined by parameters a_x and a_y as well as u_0 and v_0, which is only related with inner structure of camera and known as intrinsic parameter matrix.

There are a few limitations in some aspects of PTAM algorithm. Researchers collect data by rotating and translating a hand-held camera, in which way they can control translation speed and rotation angles of the camera, creating a relatively ideal condition to test the algorithm. The light is well-distributed in the experiment environment and the objects detected have clear textures on their surfaces. As a consequence, a relatively dense corner points can be acquired from the images.

In our experiments when we apply PTAM to do some tests with a drone, we find out that when there is not enough light in indoor environment and the degree of difference in the exterior of the objects around is low, we can't get enough FAST corner points to track, leading to the failure of the algorithm. In order to complete the task of positioning a drone in this kind of environment, we have done large quantities of experiments and designed an adaptive map points generation mechanism. All our efforts promise that the improved algorithm can extract enough FAST corners to initialize the map to implement the tasks of tracking and mapping well. The flow chart of the adaptive FAST corner detection algorithm is shown in figure 1:

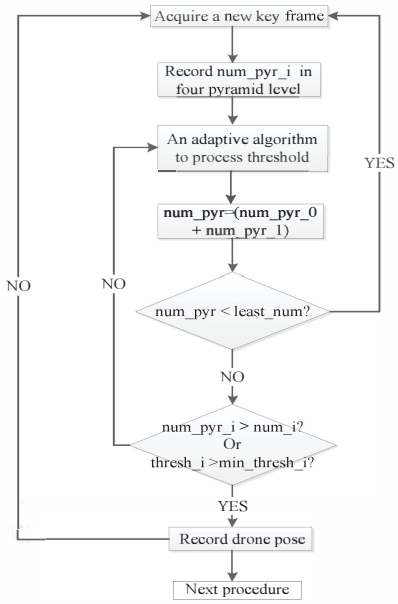

Fig. 1 The flow chart of adaptive FAST corner detection

The procedures of the algorithm are as follows:

Step1.For each key frame acquired, Compute num_pyr_i, record it then go to Step2.

Step2. Execute the formula:

num_pyr = (num_pyr_0 + num_pyr_1), if (num_pyr < least_num), return back to key frame acquirement procedure, else go to Step3.

Step3. If num_pyr_i < num_i, an adaptive method will be carried out to both num_i and thresh_i, until at least one of these two formula is true: num_pyr_i >= num_i or thresh_i <= min_thresh_i.

Step4. Record the pose of drone in current frame for the following computation.

Parameters used in the procedure above are explained as follows: num_pyr_i is the number of the FAST corners detected in each current key frame from lowest image pyramid level to its highest. While least_num is the least number of feature points with which a frame is identified as a new key frame. In terms of num_i, that's a value set previously for each pyramid level that FAST-10 algorithm should detect more corners than this number. As for thresh_i, this is a threshold used on each pyramid level for corner points detection in every key frame. As to min_thresh_i, that's a termination to the adaptive algorithm. All the postfix "i" starts from number 0 and ends with number 3 with step length 1.

B. Obstacles perception and warning

After applying our methods to augment corner points extraction procedure, the improved algorithm can execute tracking and mapping tasks more stably and generates a map for drone with feature points of its flight environment. Next we need to compute the relative distance between drone and some suspicious obstructions around. Then we compare this distance with a presetting threshold to determine whether the system should warn drone of dangers. At first we classify the map points using a clustering algorithm based on kd-tree and K-means algorithm, where K-means is an iterative algorithm based on distance judgment. We choose k-d tree to store all the map points and this really improve the efficiency of the algorithm by traversing tree data structure. We classify all the map points into k categories by choosing k different points randomly from map as the centers of initial clusters. The strategy is that we should make sure the distance between every map point to its cluster center must be closer to other cluster centers. According to this strategy we will traverse all the other points and find out a series of map points for each cluster center and add these points to each cluster. Based on the initial clustering result, we computer all the new cluster centers and keep traversing all the map points again and again until the cluster centers of consecutive two iteration stay the same. We compute every cluster center of the map by selecting Minkowski distance which is represented as:

$$d_{ij} = \sqrt{\sum_{k=1}^{n} |x_{ik} - x_{jk}|^{\lambda}} \tag{5}$$

Here we select $\lambda = 2$, namely, the Euclidean Distance as the criterion to calculate the distance. After clustering procedure completed, we will first estimate whether a cluster is an obstacle by compare the total number of it with a threshold. If it is a suspicious obstacle, we will compute the core of all the points within and get the relative distance between this core and drone in current frame. Next we will compare this distance with a presetting warning threshold and the system will display warning information on GUI if this relative distance is less than the waning threshold. The flowchart of the strategy is shown below:

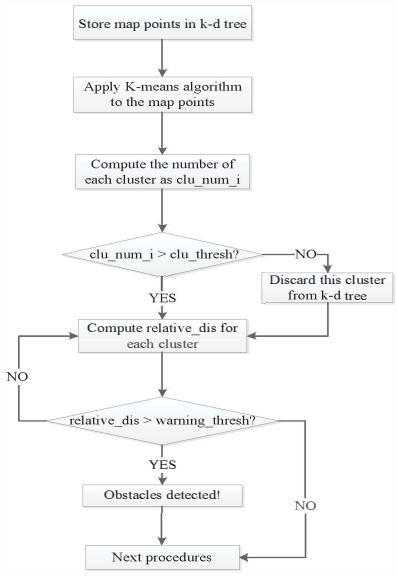

Fig. 2 The flowchart of obstacles perception and warning algorithm

All the parameters in the flowchart above are described here: clu_num_i represents the number of map points in each category after clustering and its suffix "i" stands for which cluster the value belongs to. As for clu_thresh, this is the least number upon which a cluster can be treated as a suspicious

obstacle. Argument relative_dis stands for the relative distance between drone and the core of a cluster which has been considered as an obstacle. In term of warning_thresh, this is the warning threshold set up in advance. Our method will compare the distance between UAV and obstacles around with warning_thresh, beyond which system will tell us we must take measures to ensure that our drone can avoid unknown danger. Parameter warning_thresh is interrelated both with the indoor environment and the speed of drone. In our experiments, it is triple of average horizontal speed of UAV. Next chapter we will test and verify the effectiveness of our proposed algorithm with various experiments. The results and some data will prove that our endeavours really make sense.

V. EXPERIMENT

During the experiments, we choose micro quadcopter: AR.Drone 2.0 as the flight platform. This quadcopter makes itself as a hotspot and can be connected to a computer by Wi-Fi using a driver program integrated in ROS distributed system [13]. We can control its flight with keyboard by choosing corresponding option on a GUI or use its autonomous flight module integrated in the driver to make drone enter into different modes like hover over a desk. All the computations are executed based on an Intel Core2 2.26GHz CPU with Ubuntu14.04 where we modify the algorithm and process all the data as well as display our results.

Our work environment is a small-scale meeting room where UAV will get stuck by different obstacles like tables or chairs. The light there is relatively dim with curtains all closed, so there is almost no natural light in the room. In such environment, PTAM algorithm will not be able to track enough feature points which results in the difficulty of its initialization. Even through the initialization accomplished, PTAM status will often be changed into lost or poor due to the lack of feature points, which leads to the failure to update the pose of drone and the position information of map points. As a result, it is hard for UAV to achieve autonomous positioning. Aiming at solving these annoying problems, we have done a great deal of works to improve the stability of tracking procedure of the algorithm. After integrating PTAM with our adaptive threshold processing scheme for FAST corner points, we find out that system can extract more feature points for drone which raises success rate of the initialization of PTAM algorithm and makes its running status keep at least good or even best when drone flies around for a pretty long time. Also the poses of UAV and the position information of the map points will be updated timely and recorded for further use. By changing source code for map generation, the operator can acquire more information real-time from user interface. Pictures next reveal a distinct contrast before and after the improved algorithm:

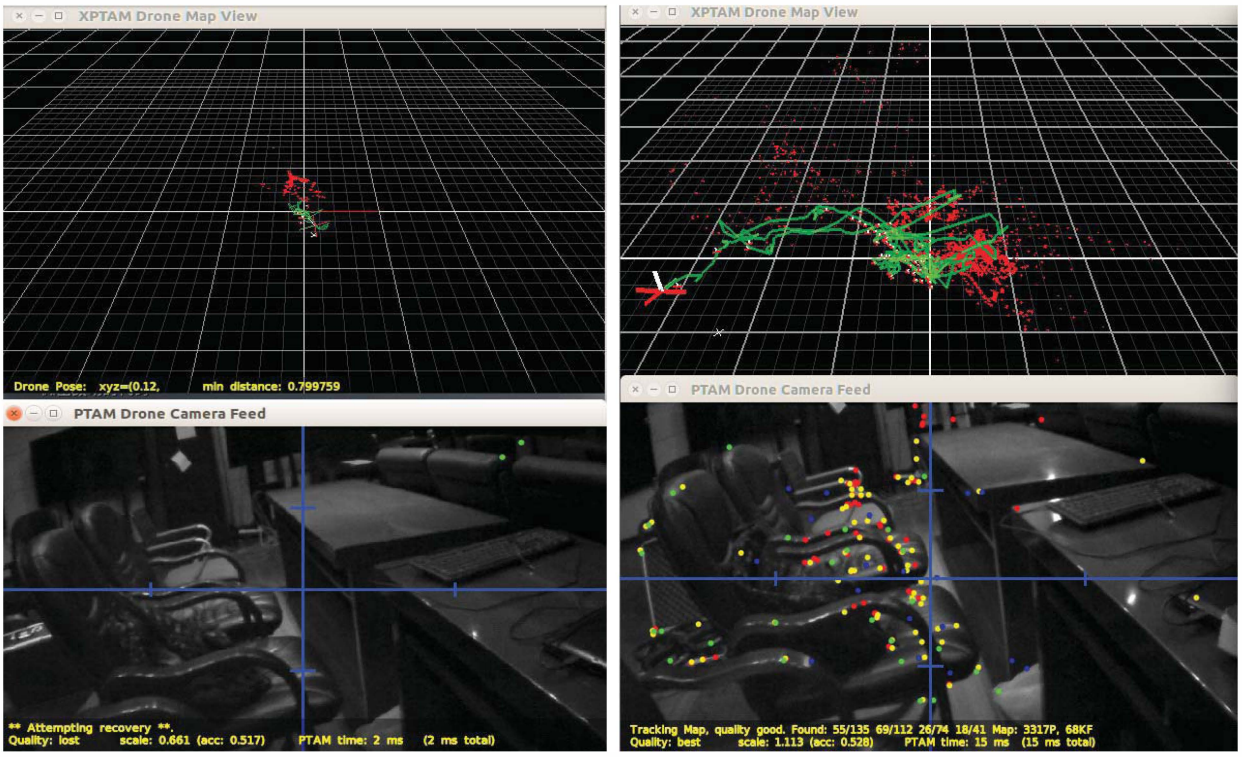

Fig. 3 A contrast of feature detection between PTAM and its improved one

Picture left shows that initial PTAM algorithm can't detect and extract enough feature points when the light is dim,

which leads to the failure of the visual algorithm. The right picture illustrates that our improved algorithm acquires relatively more feature points at the same scene under the same light condition.

With drone, we have done many flight tests to collect a large number of data of our room. From which we get some mean values of the number of FAST corner points extracted on every level of image pyramid. Here is a table of comparison data next:

TABLE 1
THE COMPARISON OF FAST POINTS GENERATED BY PTAM AND OUR IMPROVED ALGORITHM

	LEVEL0	LEVEL1	LEVEL2	LEVEL3
Initial PTAM	43/126	38/115	25/66	23/41
Improved PTAM	60/177	53/158	42/82	33/52

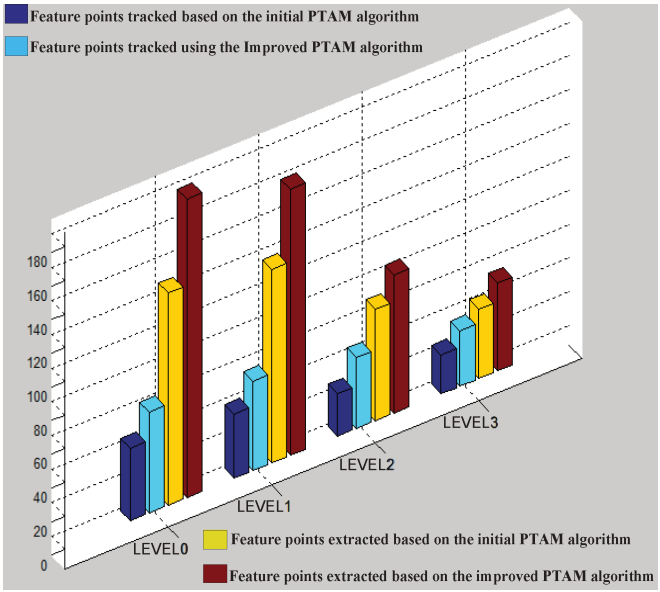

Fig. 4 A comparison of the number of FAST points generated by PTAM and our improved algorithm on different pyramid levels

As shown in the table and figure above, we generate histograms of FAST feature points on four pyramid levels. The dark-blue one represents the number of the feature points tracked based on the initial PTAM algorithm while the light-blue one means the number of the feature points tracked after PTAM algorithm has been improved. With regard to the yellow one, it stands for the number of FAST feature points extracted based on the initial PTAM algorithm and the dark red one represents the number of feature points extracted after we improve the algorithm. By contrast we can see the performance of the algorithm becomes better, proving that the method of manually choosing a threshold for FAST corner detection in initial PTAM algorithm is lack of applicability. However, there always exists some uncertainty during the experiments because of different experimental environment and laboratory equipment.

Based on improvement above, we design an obstacle perception and warning mechanism for UAV. When our drone encounters obstacles during its flight and flies into the early warning regions, the system will send out effective warning information to remind operators to manipulate the drone to avoid the barriers timely. We add a function of showing the relative distance between drone and the obstacles around. In initial user interface we can't find this functionality and it really helps when we add it . Here is an example to see:

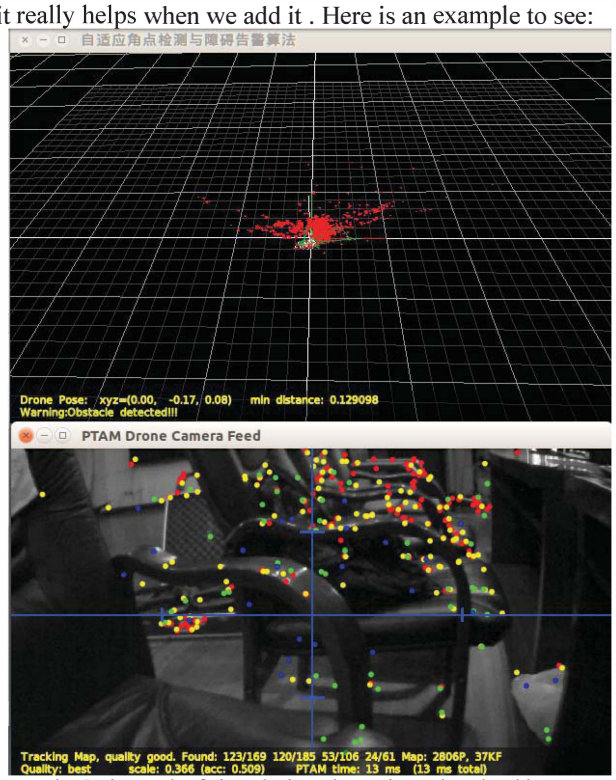

Fig. 5 The result of obstacle detection and warning algorithm

In the beginning, the warning information on the UI is not clear enough for users and the system doesn't send out warning information until our drone is about to hit on the chairs, owing to a wrongly warning threshold set in advance. Aiming at these faults, we make further efforts to modify our algorithm and display the warning information in a more highlighted way with which operators will be informed and take measures in time. By means of analyzing massive flight data stored in flight log files, we find out that when we set the warning threshold to be three times of the average horizontal speed of drone, there will be relative more time for drone to fly away the obstacles.

Two sets of pictures below are the procedures of a drone flying close to chairs and books on desks respectively. UI can display the relative distances between drone and obstacles around. The system will send out warning information once any of the distances calculated by our algorithm is less than the modified threshold set beforehand. During the flight, the distance to the potential obstacles changes gradually. This time we can see clearly from user interface the information we need and we will have much more time to make decisions

to tell drone how to fly with safety.

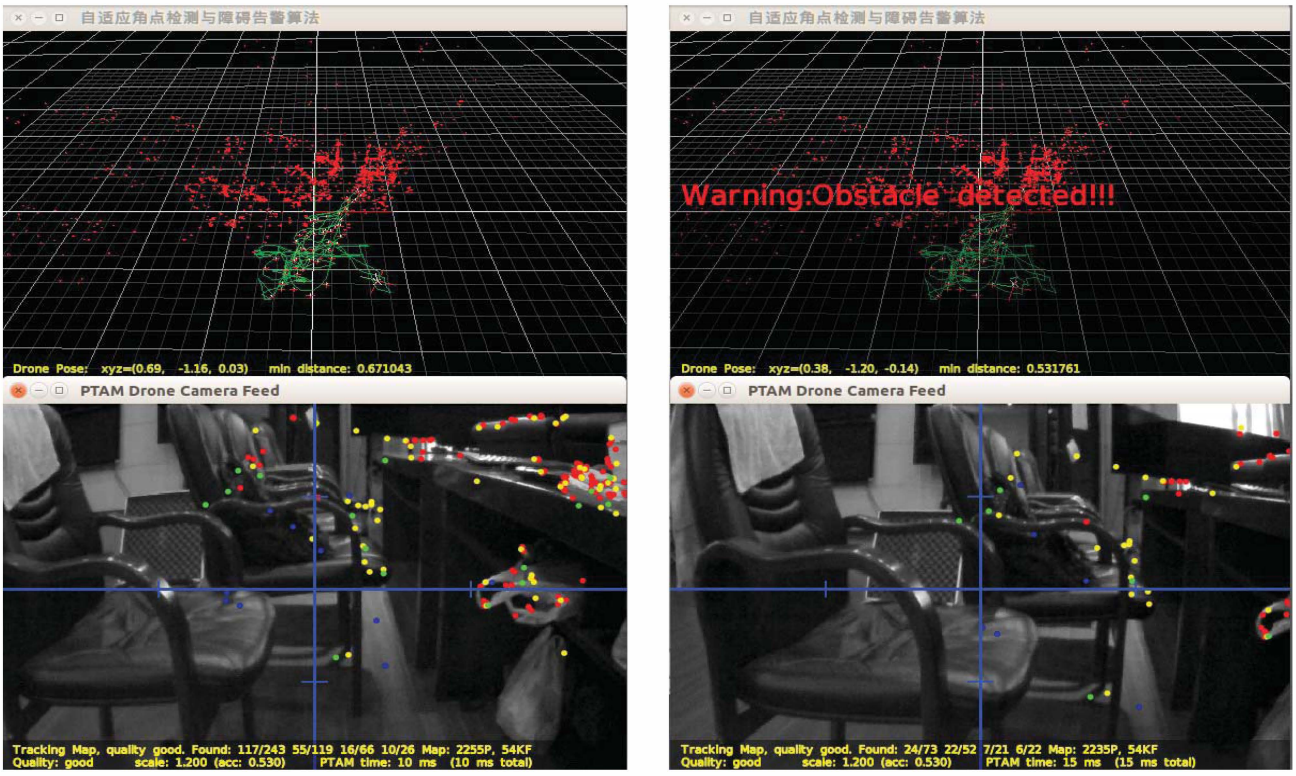

Fig. 6 The result of obstacle detection and warning algorithm

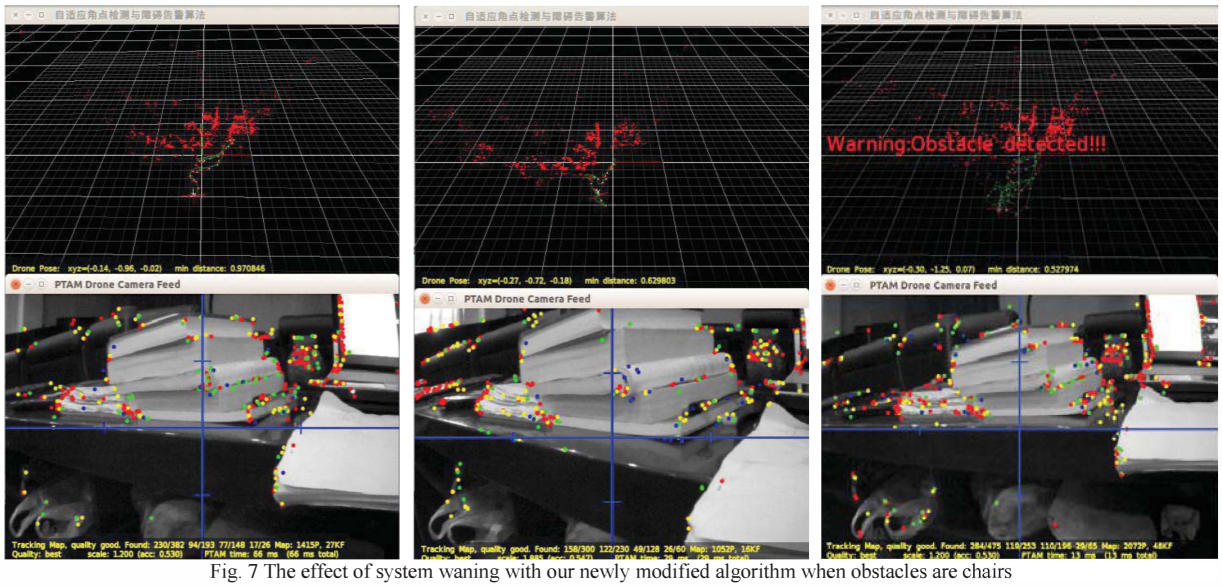

Fig. 7 The effect of system waning with our newly modified algorithm when obstacles are chairs

Results above proved that with our algorithm, the probability of initializing PTAM successfully has been raised and the drone can have more features to track and map in an environment with almost no natural light or the lighting effect is not ideal. And the mechanism we designed to detect obstacles among drone's flight area also work effectively and the system can send out warning information autonomously when necessary. This really is significant to enhance the safety of UAV flight.

VI. CONCLUSION

The algorithm proposed in this paper effectively reduces the dependency of PTAM on light condition and environment features, making it capable to more universal indoor

environment. Based on this, we add a function of suspicious obstacles detection and judgment as well as sending out warning information of obstructions to UAV. The experiment results have demonstrated that the methods proposed in this paper is in support of environment perception for UAV and helps to alert with obstacles which threaten the safety of drone's flight as soon as completing autonomous positioning for them. All the results show that the algorithm in this paper is effective and can perform well in real-time, which have great meanings to explore autonomous positioning and flight safety for UAV in GPS denied environment. Nevertheless, there are still much can be done to make our algorithm better. We can raise the ability of concurrent computation to speed up the method. As a result, the efficiency of the algorithm will be improved a lot with a better CPU and GPU. Furthermore, if we can understand the geometric information of landmarks in the map more deep, we definitely will have a few new research thoughts in obstacles warning for UAV.

REFERENCES

[1] Soren Riisgard;Morten Rufus Blas, "SLAM for Dummies-A Tutorial Approach to Simultaneous Localization and Mapping."

[2] Smith,R.C.;Cheeseman,P.On the Representation and Estimation of Spatial Uncertainty. The International Journal of Robotics Research.1986,5(4): 56–68

[3] Niklas Karlsson, Enrico Di Bernardo, James P. Ostrowski, Luis Goncalves, Paolo Pirjanian, Mario E. Munich: The vSLAM Algorithm for Robust Localization and Mapping. ICRA 2005: 24-29

[4] A. Davison, W. Mayol, and D. Murray. Real-time localisation and mapping with wearable active vision. In Proc. 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'03), Tokyo, October 2003.

[5] A. Davison, I. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. IEEE Trans. Pattern Analysis and Machine Intelligence, 2007.

[6] Richard A. Newcombe and Andrew J. Davison. Live Dense Reconstruction with a Single Moving Camera. IEEE Conference on Computer Vision and Pattern Recognition 2010.

[7] E. Eade and T. Drummond. Edge landmarks in monocular slam.In Proc. British Machine Vision Conference (BMVC'06), Edinburgh,September 2006. BMVC.

[8] E. Eade and T. Drummond. Scalable monocular slam. In Proc. IEEE Intl. Conference on Computer Vision and Pattern Recognition, pages 469–476, New York, NY, 2006.

[9] Georg Klein, David Murray. Parallel Tracking and Mapping for Small AR Workspaces. In Proc. International Symposium on Mixed and Augmented Reality (ISMAR'07, Nara)

[10] Richard A. Newcombe, Steven J. Lovegrove and Andrew J. Davison. DTAM: Dense Tracking and Mapping in Real-Time.ICCV2011

[11] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco Paulsen, O. Saurer, M. Pollefeys, Live Metric 3D Reconstruction on Mobile Phones, proc. Int. Conf. on Computer Vision

[12] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In Proc. 9th European Conference on Computer Vision, Graz, May 2006.

[13] Jakob Engel, Jürgen Sturm, Daniel Cremers, Scale-aware navigation of a low-cost quadrocopter with a monocular camera, Robotics and Autonomous Systems, Volume 62, Issue 11, November 2014, Pages 1646-1656