The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Visual Steering of UAV in Unknown Environments

Chunrong Yuan, Fabian Recktenwald and Hanspeter A. Mallot
Department of Cognitive Neuroscience
Eberhard Karls University of Tübingen
{chunrong.yuan, fabian.recktenwald, hanspeter.mallot}@uni-tuebingen.de

Abstract— **In this paper, we propose a novel approach for the visual navigation of unmanned aerial vehicles (UAV). In contrast to most available methods, a single perspective camera is used to estimate the complete set of 3D motion parameters undergone by the UAV. We establish robust point correspondences between consecutive image frames captured by the flying vehicle. Based on the estimated motion parameters as well as the reconstructed relative scene depth, a visual steering algorithm has been realized so that the UAV is capable of avoiding obstacles during navigation. The advantage of our approach lies in the fact that decision for collision avoidance is made immediately, by using purely visual information extracted from the live video sequence. Furthermore, it eliminates the time–consuming steps of explicit obstacle recognition and global reconstruction of the environment. Experimental evaluation has been carried out based on computer simulation as well as using a commercially available flying drone. It has been shown that the UAV is capable of autonomous navigation in unknown environments with arbitrary configuration of obstacles.**

I. Introduction

Biological systems use mainly visual information for navigation in the external world. In combination with other sensor modalities, vision–based navigation strategies have been applied for a number of autonomous systems. The frequently used sensors include laser range finder, IMU (inertial measurement unit), GPS, stereo or omni–directional vision systems. For the autonomous navigation of small–size UAVs, it is usually not possible to use many sensors due to weight or upload limit of the vehicles.

Shown in Figure 1 is the AR–100 UAV that we have used for the development of visual navigation strategies. For this UAV, only a GPS, a barometer sensor, an IMU and a perspective camera are kept on board, as the weight limit is 1 kg. GPS and barometer are mainly used for position and height holding. The IMU is used for attitude holding. The sensor used mainly for the purpose of visual navigation is a single perspective camera. GPS has a lower resolution and works only outdoors. Without spatial information of the environment, GPS cannot be used for obstacle avoidance. Neither is IMU used for navigation purpose, as our vision–based algorithm can calculate the 3-DOF (degree of freedom) orientation of the UAV in real time.

The AR–100 UAV is capable of vertical take–off and landing as well as autonomous flight. It has a diameter of 1 m and can fly at a speed up to 5 m/s. It can be tele–operated either by a human pilot via a remote controller (in the form of 3D joysticks) or by a controlling software running on a laptop which serves as the ground station of the UAV. In both cases the controlling commands are sent via an interface built upon radio communication and the control laws are carried out internally on the drone. Image sequences captured by the camera mounted on the UAV are also sent via radio link to the ground station.

Currently, the main challenge such a UAV system faces is the lack of efficient mechanisms for avoiding obstacles on the 3D path of its mission. If the UAV can be controlled completely via software by the ground station, it will then be able to navigate with full autonomy. Our first step toward this goal is to develop a visual steering approach capable of determining the most favorable flying direction during the course of UAV navigation.

This work was supported by the European Commission, for the research project μDrones with the contract number FP6–2005–IST–6–045248.

II. Main Contribution

In the literature, several authors have implemented obstacle avoidance strategies based on simple 2D balance of optical flow field, by calculating the divergence of optical flow between selected directions (e.g. left and right) using complex vision sensors [1] [2]. The divergence is then used to guide ground vehicles in a corridor. Here the only obstacles are the left and right walls. Although performing well in the corridor environment, simple 2D balance of optical flow fails to work in more complex environments. It has been found that the 2D balancing strategy may drive the robot straight toward walls and into corners, as it ignores the frontal obstacle detection and avoidance [3]. In addition, navigation of ground vehicles is a less challenging problem, because their movement space is 3 DOF.

For ground vehicles as well as some air vehicles such as helicopters without weight limit, another approach is to use additional sensors such as stereo or laser range finder to measure distance [4] [5] , which eases the problem of obstacle avoidance substantially. Furthermore, most of the current visual navigation approaches focus on visual odometry or SLAM (simultaneous localization and mapping). Although these approaches share some common visual processing steps with our approach, our objective is neither localization nor mapping but visual steering for safe navigation. Moreover, we have implemented advanced strategies for achieving fast and robust visual processing results. In this sense, we contribute to the state of the art not only with perception but also with decision generation from visual information.

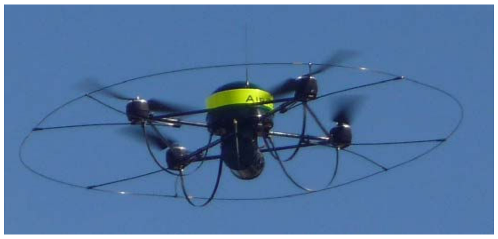

Fig. 1. AR–100 with camera mounted underneath the flight board.

In our point of view, both advanced visual processing and effective guidance strategy are needed for air vehicles with 6–DOF motion space. Due to environmental factors, such vehicles can have irregular rotations resulting e.g. from fast 3D turns. Navigation in such complex environment would require the ability to sense all obstacles in front of the robot's path and to determine in an intelligent way the most favorable flying direction based on the location and configuration of these obstacles. To achieve this, it is important to recover the full set of motion parameters including both heading and rotation information of the UAV.

Based on above considerations, we develop in this work a new approach toward autonomous visual navigation of UAV that works in general unknown environments with arbitrary configuration of obstacles. In particular, immediate decision of the next flying direction is made once the current image has been taken by a single perspective camera.

III. Robust Motion Estimation

Suppose we have two video frames captured by a camera between time t and $t+1$. Let the brightness of an image point located at $p(x, y)$ be $f(x, y, t)$. Let's use f_x and f_y to represent the spatial image gradient and f_t for the temporal image derivative, the local smoothness constraint of optical flow can be expressed as

$$\epsilon(\boldsymbol{v}) = \sum_{(x,y)\in w} (f_x u + f_y v + f_t) = 0 \;, \tag{1}$$

where $\boldsymbol{v} = [u, v]^{\mathrm{T}}$ is the image flow of a 2D point $\boldsymbol{p}$ and w a small neighborhood around $\boldsymbol{p}$. The flow vector $\boldsymbol{v}$ can hence be solved as

$$\boldsymbol{v} = \boldsymbol{G}^{-1}\boldsymbol{b} \;, \tag{2}$$

with

$$\boldsymbol{G} = \sum_{(x,y)\in w} \begin{bmatrix} {f_x}^2 & f_x f_y \\ f_x f_y & {f_y}^2 \end{bmatrix} , \tag{3}$$

and

$$\boldsymbol{b} = -\sum_{(x,y)\in w} \begin{bmatrix} f_t f_x \\ f_t f_y \end{bmatrix} . \tag{4}$$

This is the central concept of the Lucas–Kanade algorithm [6]. We have made a pyramidal implementation of this algorithm, which starts by creating a pyramid of downscaled versions of the original images. Image flow is computed in several iterative steps beginning with the smallest scale images and then refined in consecutive steps.

A. Robust image flow measurement

In modeling optical flow, one assumes that the brightness of a point does not change between frames. As a consequence, illumination changes may corrupt the velocity estimates and cause outliers in the set of image flow points. We can imagine that flow vector for some points with stronger brightness variations will be less accurate. In the following, we refer to such points as outliers. To remove these outliers resulted from the inherent deficiency of the optical flow model, we use two different approaches.

Suppose we have calculated the flow vectors $\{\boldsymbol{v}_k\}$ for a set of 2D image points $\{\boldsymbol{p}_k\}$ in frame $\boldsymbol{f}^t$. This means that we can find another set of image points $\{\boldsymbol{q}_k\}$ in frame $\boldsymbol{f}^{t+1}$ with

$$\boldsymbol{q}_k = \boldsymbol{p}_k + \boldsymbol{v}_k \;. \tag{5}$$

Our first approach is to define a motion constraint and try to find those scene points whose motion patterns are inconsistent with the motion of the camera.

We parameterize the camera motion between the two frames using a 2D rotation $\boldsymbol{R}$, a 2D translation $\boldsymbol{t}$ and a scale factor s so that

$$\boldsymbol{q}_k = s\boldsymbol{R}\boldsymbol{p}_k + \boldsymbol{t} \;. \tag{6}$$

Using a least–squares estimation method [7], the three parameters $\{s, \boldsymbol{R}, \boldsymbol{t}\}$ can be determined uniquely. For each point pair $(\boldsymbol{p}_k, \boldsymbol{q}_k)$, a distance measure d_k is calculated, where

$$d_k = ||(\boldsymbol{p}_k + \boldsymbol{v}_k) - (s\boldsymbol{R}\boldsymbol{p}_k + \boldsymbol{t})||^2 \;. \tag{7}$$

This distance measure determines whether the 2D velocity of each point agrees with the motion model specified by $\{s, \boldsymbol{R}, \boldsymbol{t}\}$. If a point pair $(\boldsymbol{p}_k, \boldsymbol{q}_k)$ has a large distance, the image flow $\boldsymbol{v}_k$ can hence be regarded as an outlier.

Though it is possible to use a fixed threshold to separate the inliers from outliers, e.g. by setting a fixed percentage of flow points as inliers, we apply a different strategy. Based on the distance $\{d_k\}$, we build a distance histogram $h(j)$, with $j = 0, \ldots, L$, where

$$L = \max\{d_k\} \;. \tag{8}$$

Now the problem becomes finding a threshold κ so that points whose d_k is bigger than κ will be identified as outliers. This is a general two–class pattern classification problem. An optimal threshold κ can be found automatically by maximizing the probability density function of the interclass difference. For details of implementation, please refer to an earlier work of Yuan [8].

By using an automatic distance threshold, those outliers with bigger distance to the motion model can be removed successfully. In order to achieve more accuracy in the estimated 3D motion parameters, we use also a second approach, which is done through backward calculation of flow vectors. This means, a backward flow vectors $\hat{\boldsymbol{v}}_k$ is calculated for each point $\boldsymbol{q}_k$ in frame $\boldsymbol{f}^{t+1}$. Hence we can get another set of points $\hat{\boldsymbol{p}}_k$ with

$$\hat{\boldsymbol{p}}_k = \boldsymbol{q}_k + \hat{\boldsymbol{v}}_k \;. \tag{9}$$

If a flow vector $\boldsymbol{v}_k$ is correct, it should satisfy

$$e_k = \| \boldsymbol{p}_k - \hat{\boldsymbol{p}}_k \| = 0\ . \tag{10}$$

The final inliers are selected by choosing those points whose $e_k < 0.1$ pixel and whose motion agrees with the motion model defined by $\{s, \boldsymbol{R}, \boldsymbol{t}\}$.

Based on the above two principles, we are able to achieve an optimal measurement of image flow. The measurement is optimal in the sense that now a data set $\{(\boldsymbol{p}_i, \boldsymbol{q}_i)\}$ with maximal accuracy of correct point correspondences between $\boldsymbol{f}^t$ and $\boldsymbol{f}^{t+1}$ is achieved.

B. Motion and depth estimation

Suppose that the 2D image flow is caused solely by a 3D rigid motion between the camera and the scene. Let's denote the motion parameter with a rotation vector $\boldsymbol{\omega} = [r_x, r_y, r_z]^{\mathrm{T}}$ and a translation vector $\boldsymbol{T} = [t_x, t_y, t_z]^{\mathrm{T}}$, the following two equations hold for a perspective camera with a unit focal length [9]:

$$u_i = \frac{t_x - x t_z}{Z_i} + [-r_x xy + r_y(x^2+1) - r_z y]\ , \tag{11}$$

$$v_i = \frac{t_y - y t_z}{Z_i} + [-r_x(y^2+1) + r_y xy + r_z x]\ , \tag{12}$$

where Z_i is the depth of a 3D point $\boldsymbol{P}_i$ and $\boldsymbol{v}_i = [u_i, v_i]^{\mathrm{T}}$ is the image flow calculated at the projected 2D point $\boldsymbol{p}_i = [x, y]^{\mathrm{T}}$. As can be seen, the translation part of the motion parameter correlates with the point depth, while the rotation part does not. Without knowledge of the exact scene depth, it is only possible to recover the direction of $\boldsymbol{T}$. For this reason, the recovered motion parameters are 5 DOF and the recovered depth Z_i is a relative scene depth.

The problem of motion and depth recover from 2D point correspondences is an ill–posed problem. To achieve a solution, extra constraints have to be sought after. One possible constraint is to use the epipolar geometry, by finding the essential or fundamental matrix [10] first and then recovering the motion parameters. We take a similar linear minimization approach using a spherical representation of the scene points, as suggested by [11]. The advantage is, after the parameter $\boldsymbol{T}$ and $\boldsymbol{\omega}$ are calculated, the relative depth of the detected 3D scene points can be obtained immediately as

$$Z_i = \frac{1 - (\boldsymbol{m}_i{}^{\mathrm{T}} \boldsymbol{T})^2}{\boldsymbol{m}_i{}^{\mathrm{T}}(\boldsymbol{\omega} \times \boldsymbol{T}) - \boldsymbol{m}_i{}^{\mathrm{T}} \boldsymbol{T}}\ . \tag{13}$$

where $\boldsymbol{m}_i$ is a spherical representation of the 2D scene point $\boldsymbol{p}_i$, i.e., $\boldsymbol{m}_i$ is the vector $[x - c_x, y - c_y, f]^{\mathrm{T}}$ normalized to a unit length. Here the parameter f is the focal length, c_x and c_y the coordinate of the principal point of the camera. In order to achieve accurate estimation, we use RANSAC to initialize and refine the 5 DOF motion parameters.

IV. Visual Steering

Once the motion parameters as well as the relative scene depths are calculated, we now obtain the heading direction of the UAV together with the location of a set of 3D scene points relative to the UAV. As each image point $\boldsymbol{p}_i$ in $\boldsymbol{f}^t$ corresponds to a 3D point $\boldsymbol{P}_i$ with a depth Z_i, the depth value indicates the time–to–contact of a possible obstacle in the environment. Our algorithm for obstacle detection does not try to reconstruct the global geometry of the environment, as is the case in the state–of–the–art SLAM approaches. Instead, we use a concept built upon directional distance sensors to find the most favorable moving direction based on the distance of nearest obstacles in several viewing directions. This is done through a novel idea of cooperative decision making from visual directional sensors.

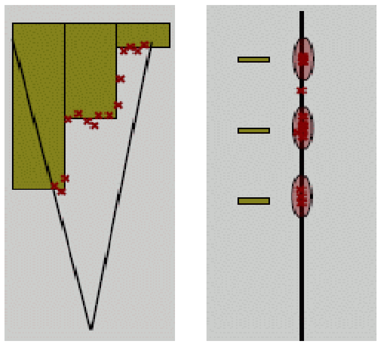
Fig. 2. Directional distance sensor and the depth clusters.

A. Directional distance sensor

A single directional sensor is specified by a direction $\boldsymbol{d}$ and an opening angle α which defines a viewing cone from the camera center. All the scene points lying within the cone define a single set of depth measurements given by

$$D(\boldsymbol{d}, \alpha) = \{Z_i | A(\boldsymbol{P}_i, \boldsymbol{d}) \le \alpha/2\}\ , \tag{14}$$

where $A(\boldsymbol{P}_i, \boldsymbol{d})$ is the angle between the two vectors, which can be computed as

$$A(\boldsymbol{P}_i, \boldsymbol{d}) = \arccos\{\frac{\boldsymbol{P}_i{}^{\mathrm{T}} \boldsymbol{d}}{\|\boldsymbol{P}_i\|\ \|\boldsymbol{d}\|}\}\ . \tag{15}$$

According to the depth measurements, the set D can be divided into several depth clusters. Each cluster K is a subset of D with at least N scene points, which satisfy

$$\forall Z_i \in K : |Z_i - d(K)| < \lambda\, d(K)\ , \tag{16}$$

where $d(K)$ is the mean distance of scene points belonging to K. The parameter λ and N are chosen depending on the size of the viewing cone and the density of depth measurements.

Shown in Figure 2 on the left is a directional distance sensor with the set of detected scene points lying within the viewing cone. Based on equation 16, the set is divided into three depth clusters, as is shown on the right of Figure 2. Now we can find the subset K_κ whose distance to camera is shortest, which is represented by the bottom cluster in Figure 2.

By examining the field of view using several directional sensors, we obtain a set of nearest depth clusters K_{κ_i} whose distance to camera is d_{κ_i}. We encode each d_{κ_i} in a fuzzy way as near, medium and far. Based on the nearness of the encoded distances, preference values can be defined for possible motion behaviors.

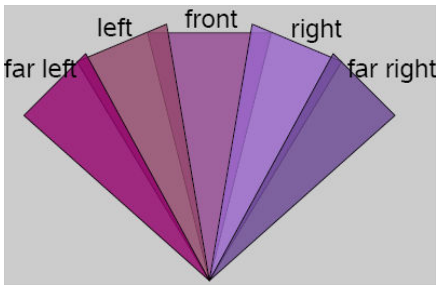

Fig. 3. Five directional distance sensors arranged symmetrically around the heading direction

TABLE I
DECISION RULES FOR HEIGHT CONTROL.

t_y	Distance to ground		
	near	medium	far
up	keep speed	decrease speed	decrease speed
zero	increase speed	keep speed	decrease speed
down	increase speed	increase speed	keep speed

B. *Visual steering strategies*

In the current visual steering algorithm, three control strategies are considered: height control, view control and horizontal motion control.

A directional distance sensor looking downwards is used for the height control of the UAV. It determines the nearness of obstacles on the ground. In addition, we take into account the vertical component of the estimated motion parameter, which is t_y. The direction of t_y can be up, zero or down. The goal is to let the UAV maintain approximately constant distance to the ground and avoid collision with both ground and ceiling. This is performed by setting the rising/sinking speed of the UAV. Decision rules for the height control of the UAV can be found in Table I.

The view control ensures that the camera is always looking in the direction of flight. This is done by setting the rotation speed around the vertical axis of the UAV.

The main visual steering is the horizontal motion control. It has five possible behaviors: left ($\leftarrow$), forward and left ($\nwarrow$), forward ($\uparrow$), forward and right ($\nearrow$) and right ($\rightarrow$). Once the flying direction is determined, motion of the UAV is achieved by setting the forward motion speed, left or right motion speed and turning speed, which are defined in the interface protocol of the AR–100 UAV respectively as pitch, roll and yaw speed. The yaw control is necessary because we want to ensure that the camera is aligned with the direction of flight for maximal performance of obstacle avoidance. Hence a left motion will also result in modifying the yaw angle by rotating to the left via the view control.

For the horizontal motion control, we have altogether five directional distance sensors arranged symmetrically around the estimated heading direction. This corresponds to a symmetric division of the visual field into far left, left, front, right and far right, as is shown in Figure 3.

For every fuzzy distance value obtained from each of the five horizontal directional sensors, we define a set of preference rules for all possible motion behaviors. The preference values for each behavior are: favorable (FA), acceptable (AC), not acceptable (NA). An example of rules for the frontal sensor is given in Table II. A further example is shown in Table III for the far–left sensor.

TABLE II
PREFERENCE RULES FOR THE FRONTAL DISTANCE SENSOR.

Distance	Behavior Preferences				
	$\leftarrow$	$\nwarrow$	$\uparrow$	$\nearrow$	$\rightarrow$
far	AC	AC	FA	AC	AC
medium	AC	FA	AC	FA	AC
near	FA	AC	NA	AC	FA

TABLE III
PREFERENCE RULES FOR THE FAR–LEFT DISTANCE SENSOR.

Distance	Behavior Preferences				
	$\leftarrow$	$\nwarrow$	$\uparrow$	$\nearrow$	$\rightarrow$
far	FA	AC	AC	AC	AC
medium	AC	FA	AC	AC	AC
near	NA	AC	FA	AC	AC

After evaluation of the output of all the five directional sensors, the motion behavior with the highest preference is selected. Suppose the fuzzy distance value of the five directional sensors (from left to right) are near, far, far, medium and near, the preference values for each motion behavior can be determined individually, as shown in Table IV. If we take all the sensors into account by adding all the preferences appeared in each column, the final preference value for each motion direction can be obtained, as is shown in the second line of Table IV from bottom. It can be seen that the highest preference value is achieved for the forward direction. Hence the safest flying direction is moving forward.

V. EXPERIMENTAL EVALUATION

The proposed approach has been implemented using C++ on a SAMSUNG M60 laptop. Beside the whole visual steering approach, we have also implemented software interfaces for bilateral communication with the flying drone and the grabbing of transmitted video images. While the control laws incorporating system dynamics are implemented directly on the drone by the UAV manufacturer, the controlling commands for avoiding detected obstacles are calculated by our visual steering algorithm and sent via radio transmission to the drone. In order to test the proposed approach, we

TABLE IV
EXAMPLE OF DECISION MAKING BASED ON ALL DISTANCE SENSORS.

Sensor	Distance	Behavior Preferences				
		$\leftarrow$	$\nwarrow$	$\uparrow$	$\nearrow$	$\rightarrow$
far left	near	NA	AC	FA	AC	AC
left	far	AC	FA	AC	AC	AC
front	far	AC	AC	FA	AC	AC
right	medium	AC	AC	FA	AC	AC
far right	near	AC	AC	FA	AC	NA
All sensors		1 NA 4 ACs	1 FA 4 ACs	4 FAs 1 AC	5 ACs	1 NA 4 ACs
Decision				×		

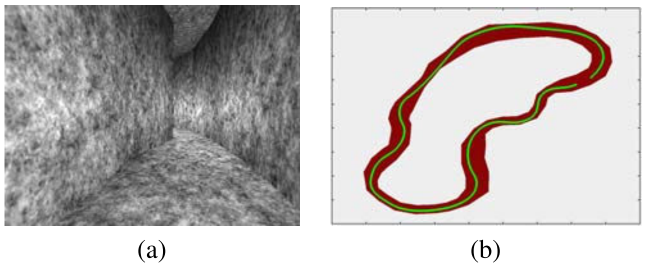

Fig. 4. Experiment in the s-shaped corridor.

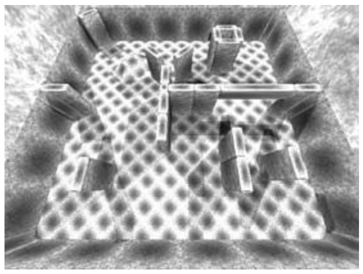

Fig. 5. The maze–like environment.

have carried out experiments in both simulated and real–world environments. We vary the illumination in the simulated environment to have lighting changes, which appear usually in the real world. Considering speed, the system can achieve a frame rate of 15 to 20 frames/s, including robust image flow calculation, motion and depth estimation as well as the fuzzy determination of flying direction based on the output of several directional sensors. The accuracy of the recovered motion parameters, which is calculated using simulated environment where the ground truth is available, is in average 1^o for rotation and 2^o for the heading direction.

The first visual steering experiment is carried out in an s–shaped corridor. Shown in Figure 4(a) is an image of the corridor. A top view of the whole corridor is shown in Figure 4(b) in color red. After the motion and scene depth have been calculated from two initial images, the UAV is set to move in the direction determined by our visual steering algorithm. Then the UAV captures the next frame, determine its next moving direction and moves accordingly. During the experiment, we have recorded the trajectory of the UAV. Shown in Figure 4(b) in color green is the recorded trajectory of the UAV. As demonstrated by this green curve, the vehicle is able to navigate through the corridor without any collision.

Further experiments are carried out in a maze–like environment with general configuration of obstacles, as is shown in Figure 5. In the beginning, the vehicle is placed in an arbitrary position within the maze. The flying direction is set to the viewing direction of the camera, i.e., the forward direction. Several experiments with different starting positions show that the vehicle is capable of navigating safely within the maze by finding its way automatically. Particularly, it is able to regulate both flying direction and height to avoid collision with walls, corners, building, the ground, the arch etc. Some images showing the path of the drone during its navigation are shown in Figure 6.

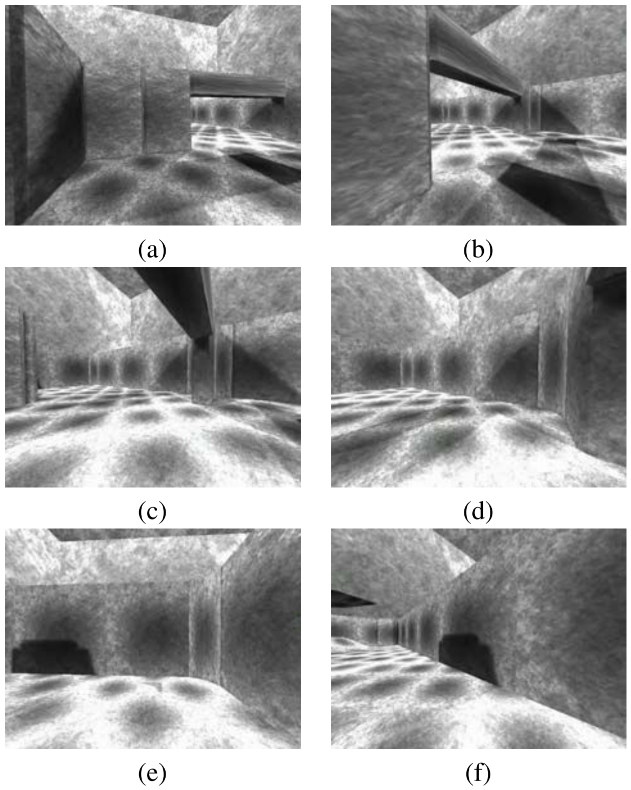

Fig. 6. Autonomous navigation in the maze. From (a) to (b): The vehicle has found free space for navigation and change its flying direction toward the arch to avoid collision with walls and corners. From (b) to (c) and to (d): Flying through the arch. From (d) to (e): Begin to turn around the corner. (f): After having turned around the corner, flying along the wall and adjust height due to the level of terrain and the obstacle floating in the air.

In addition to the corridor and maze experiments, we have also performed real–word experiments. Several experiments have been carried out in an indoor environment. As illustrated in Figure 7 and 8, one can see a T–shaped corridor with isolated obstacles in it. Besides the static obstacles, the doors may be closed and opened so that the environment is unknown. This is indeed an environment with general configuration of obstacles.

The first experiment is to keep the platform in the available free space while avoiding different obstacles. We show in Figure 7 the visual path of the flying platform together with the decision made by our algorithm with red arrows. Note that the different lengths of these arrows, which is set in inverse proportion to the distance of obstacles. A longer arrow indicates a shorter distance to the detected obstacle and hence a faster speed needed by the UAV for flying away from it. As can be seen, the platform is kept in the middle of the free space. It turns to the left when it comes near to the table.

A second experiment carried out in the same environment is to fly the platform manually along the left side and see how the system will react. We show in Figure 8 the results. As can be seen, it always tells the system to fly in the forward–right direction as the distance to obstacles on the left is smaller than that on the right.

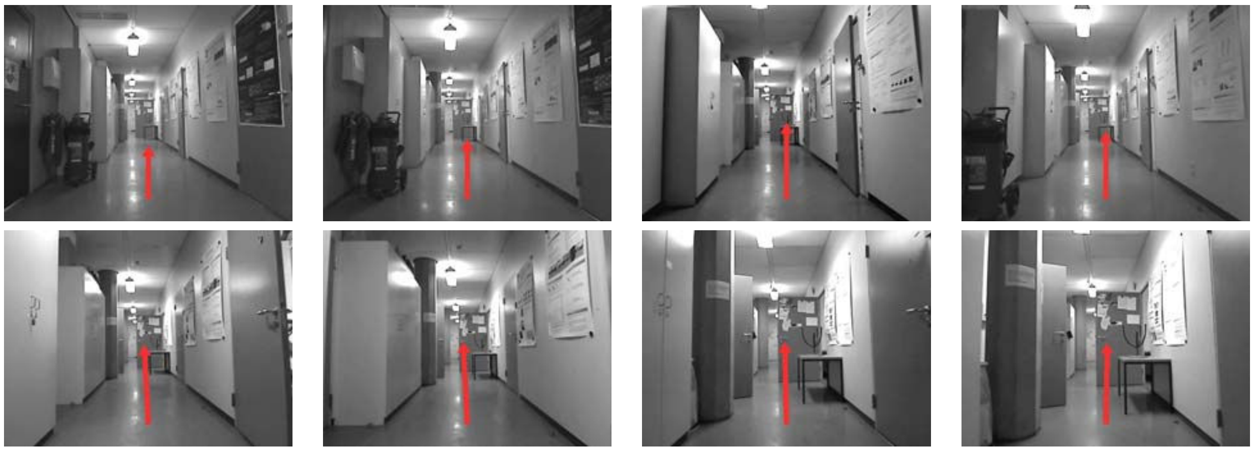

Fig. 7. First experiment in the indoor environment.

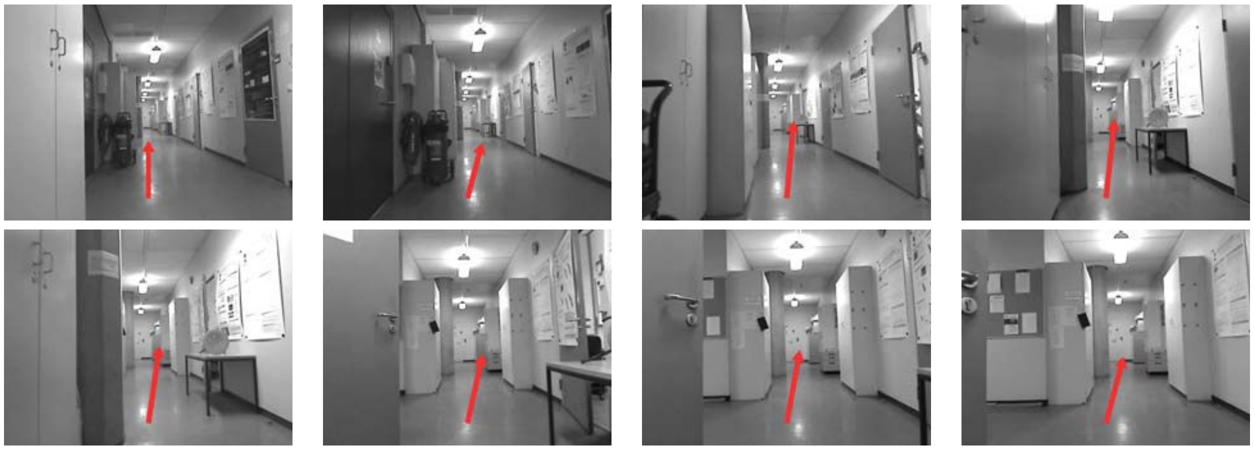

Fig. 8. Second experiment in the indoor environment.

VI. Conclusion

Providing flying robots with autonomous visual navigation capabilities for real–world operations is a challenging problem with high application potentials. In this work, we realize a systematic approach for the accurate recovery of motion and depth parameters from robust point correspondences and propose an advanced approach for the visual steering of UAV. Our vision–based approach requires no additional sensor information and can work in real–word environment with general configuration of obstacles. In our point of view, we have developed a novel approach for the visual navigation of light–weight UAV, which is not only new but also unique. As demonstrated by several examples, the UAV is capable of safe navigation in unknown 3D environments using entirely visual information. Currently we are working parallel on the detection of independently moving objects based on visual motion information. With the estimated motion parameters of both the camera and the individually moving objects, we will be able to integrate our current visual steering approach for the avoidance of both static and dynamic obstacles.

References

[1] A. Argyros, D. Tsakiris, and C. Groyer, "Biomimetic centering behavior," *IEEE Robotics & Automation Magazine*, vol. 11, pp. 21–30, 2004.

[2] J. Lim, M. C., D. Shaw, and N. Barnes, "Insect inspired robotics," in *The Australasian Conference on Robotics and Automation (ACRA 2006)*, 2006.

[3] L. Muratet, S. Doncieux, Y. Briere, and J. Meyer, "A contribution to vision-based autonomous helicopter flight in urban environments," *Robot and autonomous systems*, vol. 50, no. 4, pp. 195–209, 2004.

[4] J. Minguez and L. Montano, "Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios," *IEEE Trans. Robotics and Automation*, vol. 20, no. 1, pp. 45–59, 2004.

[5] D. Shi, M. Selekwa, E. Collins Jr., and C. Moore, "Fuzzy behavior navigation for an unmanned helicopter in unknown environments," in *IEEE Int. Conference on Systems, Man and Cybernetics*, 2005.

[6] T. Kanade and B. Lucas, "An iterative image registration technique with an application to stereo vision," in *Int. Joint Conf. Artificial Intelligence (IJCAI'81)*, 1981.

[7] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[8] C. Yuan, "Simultaneous tracking of multiple objects for augmented reality applications," in *7th Eurographics Symposium on Multimedia (EGMM 2004)*, 2004, pp. 41–47.

[9] B. Horn, *Robot Vision*. MIT Press, 1986.

[10] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, UK: Cambridge University Press, 2000.

[11] K. Kanatani, "3-D interpretation of optical flow by renormalization," *Int. Journal of computer vision*, vol. 11, no. 3, pp. 267–282, 1993.