2010 8th IEEE International Conference on
Control and Automation
Xiamen, China, June 9-11, 2010

Application of Matlab to the Vision-Based Navigation of UAVs

Y.C. Chen, F.Y. Hsiao*, J.F. Shen, F.C. Hung, and S.Y. Lin

Abstract— **The application of Matlab to the vision-based navigation of unmanned aerials vehicles (UAVs) is investigated in this paper. Speaking of hardware implementation, researchers conventionally turn to the C code. However, it is usually not easy as dealing with sophisticated algorithms. Alternatively, we think Matlab an ideal candidate to navigate and control UAVs, not only because of its powerful functions on matrix operation and image processing, but also due to its capability of collaboration with hardware. In this research, the UAV is navigated with vision-based equipments, and Matlab is selected as the platform of development, from analysis to the whole hardware hierarchy. Two types of navigation problems are introduced as potential applications, and corresponding navigation laws are proposed taking Matlab into account. Experiments are also designed to examine the results, and processing time and successful rate are introduced as indices of performance. It is shown that Matlab and the proposed navigation laws are ideal platform and algorithms, respectively, to navigate UAVs.**

Keywords: Navigation, Vision-based navigation, Unmanned aerial vehicles (UAVs), Hardware implementation, Matlab.

I. Introduction

This paper studies navigation of unmanned aerial vehicles (UAVs) using vision-based technology with Matlab as the programming language. UAVs are recently of great interests among scientists due to their low cost in manufacture and wide range of applications. Besides UAVs, investigation of machine vision are also hot topics in recent years. In order to achieve autopilot of UAV, several navigation and guidance algorithms are proposed. Among them navigating using vision based technology is one interesting and challenging topic [1], [2], [3], [4], [6], [13].

On the other hand, researchers usually do simulations of algorithms in Matlab while doing hardware implementation using different languages, such as *C*. This action makes the process of establishing the whole system inconsistent and uneasy, since one needs to program one algorithm in two languages, one for simulation and the other for implementation.

In this research we plan to navigate the UAV using vision-based technology and construct the whole system using Matlab. Two types of navigation problems are discussed in this paper – specific terrain feature tracking, and real-time target determination and tracking.

Yan-Chuan Chen is a graduate student of Aerospace Engineering, Tamkang University, Tamsui 251, Taiwan 697430295@s92.tku.edu.tw

Fu-Yuen Hsiao is with faculty of Aerospace Engineering, Tamkang University, Tamsui 251, Taiwan fyhsiao@mail.tku.edu.tw

Jeng-Fu Shen is with Research Assistant of Performance Technologies Laboratory, Center for Art and Technology, Taipei National University of the Arts Beitou, Taiwan scott_ngk@dance.tnua.edu.tw

Fu-Ching Hung and Si-Yu Lin are undergraduate students of Aerospace Engineering,Tamkang University, Tamsui 251, Taiwan

Robotic Embedded Systems Laboratory of University of South California has proposed and implement a vision-based landing algorithm, which enables an unmanned helicopter to recognize a halipad and align with it [12]. Kim proposed algorithms with which a UAV navigate itself using roads and structures [9]. Cheng navigates an unmanned halicopter along specific terrain features, such as rivers and roads [5]. Other than that, automatic real-time target detection and tracking are investigated by some researchers, such as [7], [10], [11]. However, most researches investigate this topic using infrared camera, while in this research the regular optical CCD cameras are used.

In detail, we intend to navigate a UAV using vision methodology provided two applications – track a specific terrain or track a real-time determined object. First, this article gives a detailed definition of our mission. Basic image processing algorithms are then briefly reviewed, followed by our proposed navigation algorithms, mostly in matrix operation since the whole system is to be built on Matlab. Different from other researches who often use C for implementation, we also use Matlab for numerical simulations and the construction of the whole hardware system, including ground station, communication with onboard computer and so on. Hardware-in-loop experiments are done to demonstrate the fine results of our algorithms and system.

II. Mission Definition

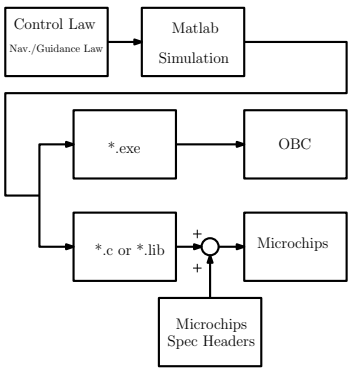

Fig. 1. Flow chart of UAV navigation using Matlab, from theory analysis and numerical simulations to hardware implementation.

In this research vision-based navigation of UAVs using Matlab is investigated, including specific-feature terrain tracking and real-time-selected object tracking. In the first mission, the UAV is going to fly along a specific terrain feature, such as coastlines, rivers or roads. In the second mission, the UAV is to track a moving object, which is

selected in real time. Several specific algorithms are selected in this research in order to make use of powerful matrix manipulation capability in Mablab.

Developed by the *Mathwork* corporation, *Matlab* has been a convenient tool for dynamics analysis and control law design. On the other hand, however, hardware implementation is completely a different world. For a long term researchers usually program the hardware using *C code*, taking advantage of its strong capability of manipulating *hardware, I/O ports* and *memories*.

C language and its group indeed are good choice for manipulating hardware, except for *programming* itself. Since *C language* is a *low-level* programming language, it is really painful to write and compile it, much less to implement a very complicated control algorithms, which usually requires background knowledge of numerical methods.

As one of a *high-level* programming language, *Matlab* is relatively easier to use. Furthermore, *Matlab* has powerful toolboxes for numerical analysis, especially in the aspects of dynamics and control. In the past this capability is constrained in numerical simulations due to the lack of ability in manipulating hardware registers and memories. Nevertheless, for the recent years *Matlab* has developed the ability to control I/O ports, manipulate registers and memories, as well as generate high efficient *C codes* [14]. Therefore, we are interested in the way to apply *Matlab* not only to numerical analysis, but to hardware implementation.

Figure 1 shows the flow chart of how we investigate this problem. We first design control law or navigation law based on control and image processing theories, such as fuzzy logic control and so on. Then, the control law is simulated in Matlab, either by programming on our own or by introducing the default toolbox. If the simulation results are good enough, the developed programming functions are compiled to generate *executable files* (`*.exe`) or *C codes* (`*.c`) for the purpose of hardware implementation. Processing time is measured as an index to quantify the performance of this research.

III. Image Processing

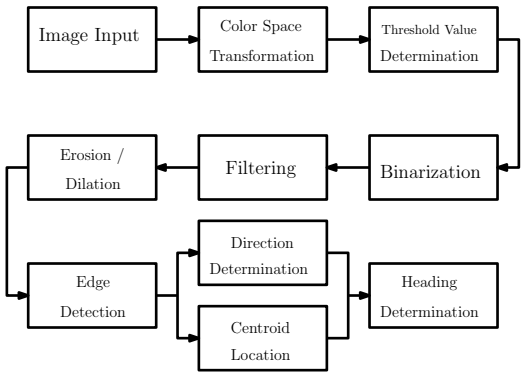

Fig. 2. Flow chart of standard procedure on image processing

Before navigation laws, we would like to briefly review image processing procedure. Figure 2 gives an example on the standard procedure of image processing. When an onboard camera shoots a video, images are captured from frames of the video and input into the image processing algorithms for further analysis. In general images are in the RGB format. This format is difficult to analyze due to the sensitive nature of RGB to the variation of light strength.

Consequently, color space transformation is generally necessary, and two types of color space are selected and compared in this paper. They are *HSI* and *YCbCr*. The details of those transformations are available in [5]. Having transformed to another color space, the image is binarized with a selected threshold value.

$$g(i,j) \quad = \quad \begin{cases} 1 & T_{\min} \le g(i,j) \le T_{\max} \\ 0 & \text{otherwise} \end{cases}$$

where $g(i,j)$ is the pixel value, and $T_{\min}$ and $T_{\max}$ are the threshold. In this research, there are two ways to determine the threshold values. When discussing terrain feature tracking, we assign a pre-selected value. When discussing moving object tracking, however, we pick up the object to track at beginning, and the system can analyze the color space feature of the chosen object automatically. The color space feature is then assigned as the threshold value. As a result, we are capable of tracking a randomly chosen object in real time.

Then the standard procedures in *filtering, erosion and dilation* are introduced to eliminating noises, and *edge detection* is applied to identify the object (terrain features or moving target) to track. Based on different navigation rules, which are further discussed in the later section, we compute the heading error and back feed the results to guide the UAV. Figures 3(a) – 4(d) series give examples of the whole procedures to obtain the coastline.

Fig. 3. a) The original figure; b) Transformed to the HSI color space; c) Determine the threshold value; d) Binerization

Fig. 4. a) Filtering is applied; b) Dilation is applied; c) Erosion is applied; d) Edge detection to obtain the sea shore.

IV. Navigation Rules

Two common applications of navigating the UAV using vision-based technology are discussed in this research. The first one is to track a moving target, while the other is to track a specific terrain feature. There are slight differences in navigation rules between these two missions. To track a moving target, we have to locate the centroid point of the object and determine the heading based on this information. To track a terrain feature, however, there is no centroid point. Instead, we have to determine the "trend" of the feature,

such as the "main direction" of a curved road or a curved coastline.

Moreover, the threshold value usually can be assigned in advance for the mission of tracking terrain features since the color of sea and the color of roads don't change too much. To track a moving object, however, we'd better have the ability to identify the target in real time. A very common scenario goes like this: a UAV is sent to a location to reconnoiter, but, at the beginning of the mission, we might not know what a target exactly looks like. When the UAV transmits video back to the ground station, we must determine the object to track online. Consequently, the computer has to determine the threshold value in real time instead of being set up in advance.

A. Tracking Terrain Features

The determination of reference heading in tracking terrain features is important in navigating a UAV. Having performed image process described in the preceding section, the edge of the interested terrain is assumed identified. To obtain the reference heading, we have to figure out the "trend" of the edge.

Hough transform is a common way to indicate the "trend" [5]. The Hough transform makes computer to transform every pixel of the coastline into the (r, ρ) space, and to determine the "trend" by looking for the local maxima in a so-called accumulator space. This manipulation requires to do the reverse of trigonometric function to every pixel, and may take a lot of time.

Different from the traditional way, we determine the route by treating the pixels of the terrain edge, say seashore, as "discrete dots", and apply the technique of first order curve fitting, the least-square method, to get the "trend" of these "discrete data". There are two advantages of doing this. First of all, the least-square method can be manipulated with matrix operation, which just meets the expertise of *Matlab*. Second, in this operation we don't perform the reverse of trigonometric function to every pixel, only to the final result where the slope is the cotangent value of the reference heading. It must save plenty of time and computer resources. An example of indicating reference heading is provided in Fig. 5 (a), where the coastline is the one shown in Fig. 4(d).

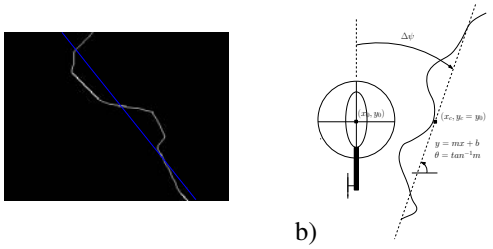

Fig. 5. a) The approximate reference heading determined by the least square method. b) The first order regressive curve and determination of reference heading

The other consideration on navigating the UAV is that it may fly "parallel to" the desired track, but not "right above" it. To solve this problem, we compute the x-value of the first order regressive curve and compare it with the x-value of the center point, using the same y value. If they don't match, we add in an additional heading error so that the helicopter will turn the heading and track the new command again. A positive error is added if the helicopter is at the left hand side of the regressive line which a negative error is applied if at the right hand side. A cartoon depicting this algorithm is provided in Fig. 5 (b)

B. Tracking Moving Objects

To track a moving target, the capability of automatic real-time target determination is necessary. As mentioned earlier, some missions require the UAV to reconnoiter above potential targets and the operator in ground station determines the target to track online. Therefore, once the target is chosen, the computer has to figure out its features immediately, and keep tracking the target based on this result.

In this paper we develop a simple algorithm for automatic real-time target determination with regular CCD camera. Different from analyzing infrared images which are mostly discussed in this field, our system needs to deal with regular colorful images, consisting of three fundamental colors – red, green, and blue. As mentioned in the preceding section, they are easily influenced by the environment. Consequently an alternative color space must be introduced as determining target features.

1) Target Extraction: Suppose the UAV flies above certain potential targets and transmits the image back to the ground station. Suppose a target is selected to be locked up. The target can then be extracted from the whole image by clicking the mouse on the diagonal points around it. As an example shown in Figs 6(a) and 6(b), the dark blue magnet is selected from magnets by clicking the mouse on the diagonal points around it.

Let $M_{CS}^{m\times n\times 3}$ denote the image matrix of size $m \times n$ in certain color space $CS = \{RGB, HSI, YCbCr\}$. In this paper, we deal with image of size 320×240 as a compromise of resolution and processing time. Let $T_{CS}^{p\times q\times 3}$ be the extracted target image matrix of size $p \times q$ in certain color space. Moreover, define

$$B_{CS}^{m\times n\times 3} = M \backslash T$$

as the background image by "carving" T matrix out of M matrix. The pixel values where T locates in B are set 0's.

Fig. 6. a) A scene composed of colorful magnets; b) The dark blue magnet is selected by the operator by clicking the mouse on the diagonal points around it. c) The dark blue magnet is successfully identified by the computer by automatic feature determination

2) Feature Determination in HSI space: Having obtained the target and background image matrices, we than apply *HSI* transformation to T and B. Usually, the *hue* value is more distinctive than saturation and intensity. Thus we only consider the H value and discard the S and I values in this research. The threshold value for binary manipulation is determined from the H value who has the peak in the histogram.

However, H value is not distinctive enough for some colors. More information should be added in. We propose that one of the fundamental colors can be the additional information. Let $\mathbf{V}_R \in \mathbb{N}^{256}$ denote the vector that contains information of histogram of red color space, i.e., the nth element of $\mathbf{V}_R$ represents the amount of pixels whose red color is $R = n - 1$. $\mathbf{V}_G$ and $\mathbf{V}_B$ are defined in the similar way. Define $\mathbf{V}(T)$ and $\mathbf{V}(B)$ to denote the histogram vectors of target and background matrices, respectively.

The "orthogonality" of the target and background histogram vectors can be introduced to determine the how distinctive they are. Define an index J as the inner product of the histogram vectors of target and background matrices. In detail, they are

$$J_R = \mathbf{V}_R(T) \cdot \mathbf{V}_R(B)$$
$$J_G = \mathbf{V}_G(T) \cdot \mathbf{V}_G(B)$$
$$J_B = \mathbf{V}_B(T) \cdot \mathbf{V}_B(B)$$

Given the extreme condition that the target is complete red and the background contains no red color, we conclude $J_R = 0$. As a result, by looking for $\min\{J_R, J_G, J_B\}$ we can find the most distinctive property between the target and the background in the RGB color space. This property can also be selected as a threshold value for binary manipulation.

Assume the original image $M_{RGB}^{320\times240\times3}$ is binarized in the H space and in one of the fundamental R, G, B spaces, denoted as $M_H^{320\times240}$ and $M_f^{320\times240}$, respectively. To obtain a clearer image of target, we take the the intersection of M_H and M_f.

$$M_{final} \quad = \quad M_H \bigcap M_f$$

where the binary operation for every pixel is defined as

$\bigcap$	0	1
0	0	0
1	0	1

An example of the result using the above algorithm is shown in Fig. 6(c). After the image is successfully binerized, the continuing image process can be applied and the centroid of the target can be found, also shown in Fig. 6(c).

3) Feature Determination in YCbCr space: To extract a target using the *YCbCr* space is similar to the procedure described in the previous section. We have to determine the target by clicking the mouse on the diagonal points around it. Then transform the target to the *YCbCr* color space. However, we determine the threshold value from both Cb and Cr.

The procedure is similar to what we have done to the H space. Threshold values are defined for the Cb and Cr spaces, respectively, from the histograms in order to manipulate binarization.

Assume M_{Cr} and M_{Cb} are the results of binarization using Cr and Cb spaces, respectively. Similarly, to obtain a clearer image of target we take the the intersection of M_{Cr} and M_{Cb}.

Because the transformation to $YCbCr$ is easier than to HSI, and the procedures to detect the target is easier in $YCbCr$. The overall process time is shorter. However, the tradeoff is the image quality. Since $YCbCr$ has simpler procedures, the quality of processed image is not as good as that using HSI transformation.

V. Navigation Using Matlab Tools

Different from other papers, in this research we establish the whole system using Matlab. Roughly speaking, to accomplish this research we need the following hierarchy:

- Analysis
  - Algorithm development
  - Numerical simulations of image processing
- Implementation
  - Receiving video from visual equipments
  - Establishment of ground station
    - Graphic user interface (GUI)
    - Navigation using image processing method
    - Guidance commands transmitting to the UAV
- Stand along package

As mentioned earlier, most researchers do the analysis in Matlab but do the implementation using other languages. This requires to re-program Matlab functions used in analysis under the circumstance of other programming language. In this paper, however, we make works on analysis and implementation consistent. In addition to imaging processing toolboxes for numerical simulation, Matlab also provides functions that communicate with hardware.

A. Image Input Processing

The video usually comes in two format – analog or digital. If the video comes in digital format, such as a webcam, the Matlab can import and play the video using the command `vidr()`. If the video comes in as analog signals, we use image grabber to transform it into digital. We can set the format of color space in the video directly from accessing the property of `vidr()`. Another command `getsnapshot()` can be introduced to grab images, stored as the format we set for `vidr()`. Those images can further be loaded for image processing work.

Once one image is loaded, the powerful toolboxes for image processing in Matlab can be applied immediately. There is no provided functions for *HSI* transformation so that we program on our own, while the *YCbCr* transformation can be accomplished with `rgb2ycbcr()`. `imdilate()`, `imerode()`, `imhist()` are applied to perform the dilation, erosion, histogram, respectively.

In addition to image processing toolbox, `polyfit(X,Y,N)` are introduced to fit a curve up

to the *N*th order. In our situation, we set $N = 1$ and the function returns the slope and intersection parameters for the later use of navigation. The centroid of a tracking target can be found by summing up pixel values along the X- and Y-direction, respectively, and use the command `find()` to find the pixel number where there is a peak value.

B. Ground Station

The establishment of ground station requires an interface for users. Nowadays a graphic one is the most popular. The Matlab support development of GUI, by using the command `guide`. The GUI in Matlab is bolstered by background *m-files*. Consequently, all programs programmed as m-flies (even as Simulink files) can be included into as background supporting functions.

Suppose we have developed a wonderful algorithm, and obtained wonderful simulation results in the stage of analysis. Then the developed functions can be import to the ground station system immediately, no need to program in other languages again.

The matlab also supports variety of communicating methods, including RS232, TCP/IP, UDP, and so on. The navigation and guidance commands are then transmitted to the plant via a proper method [8].

C. Stand Along Program

In addition to executing the above system under the Matlab platform, we are able to deploy the development result to other computers. By executing the function of `deploytool`, all programs developed in matlab can be packed in an stand along executable package, the `*.exe` files. If the whole system is to perform in a computer where there is no Matlab installed, the user simply needs to execute the complied `*.exe` files, provided that a portable Matlab program named *MCR* should be installed prior to everything.

VI. Experiments

Experiments are designed to demonstrate the feasibility of algorithms mentioned in the previous section, including algorithms for terrain feature tracking, and real-time target determination and tracking.

A. Experiments of Terrain Feature Tracking

Experiments of terrain feature tracking are simulated in X-plane, which is a common software for aircraft simulation. The virtue aircraft in X-plane can communicate, using UDP protocal, with external computer via network. The simulated system is integrated as shown in Fig. 7 (a), and an unmanned helicopter in X-plane is selected in this research [8]. X-Plane provides view observed from the bottom of the aircraft, which simulates a camera installed at the bottom of the aircraft. In order to transmit the view back to the ground station, we set up a camera to simulate the observation. An onboard computer (OBC) is connected to X-plane, simulating the OBC in a real UAV.

As mentioned earlier, we use Matlab to establish the whole ground station, shown in shown in Fig. 7 (b), which

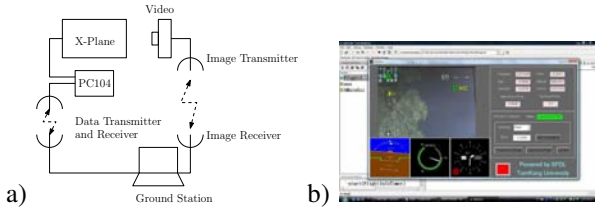

Fig. 7. a) The integrated autonomous flight system. b) A demonstration of the control panel in ground station built by Matlab

is capable of doing image processing and issuing flight command to the OBC via data transmitter. Actually, the flight control program installed in the OBC are also programmed by Mablab and packed as a stand along package. The update rate of navigating commands is up to 10 Hz, from the input of image to the transmission of navigating commands.

Figure 8 demonstrates a series of images simulating the view from the onboard CCD camera, and the these images demonstrate that our algorithms for navigation and controls for attitude stabilization work very well. The images in Fig. 8 arrange from top to down, left to right. We can see from Figs. 8-1 to 8-3 that the rotary UAV encounters the coastline in certain angle at beginning. Then our navigation algorithm starts to work and corrects the heading angle automatically, as shown in Figs. 8-4 to 8-6. Eventually, the rotary UAV cruises on top of the coastline.

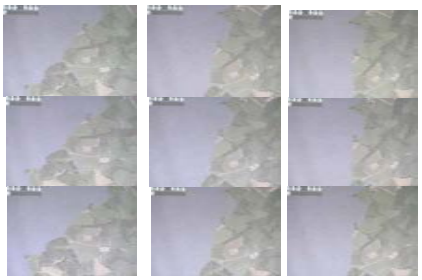

Fig. 8. A series of images showing the flight trajectory of the rotary UAV (The series arrange from top to down, left to right.)

B. Experiments of Real-Time Target Tracking

Although X-plane provides an excellent platform to simulate dynamics of an aircraft, unfortunately we cannot simulate the experiments of real-time target tracking on it. The reason is that X-plane only provides simulation of aerial vehicles, but not ground moving objects.

We do this experiments in two ways. First of all, airborne pictures from *google map* are downloaded to simulate the view from aircraft, as shown in Fig. 9 (a). On the image we randomly choose cars to track. In some cases our algorithm does work, as shown in Fig. 9 (b). However, if there are two or more cars in the same color, the computer will detect them all, shown in Fig. 9 (c). The worst case happens when the car has similar color to the road or other equipments on road. The identification under this situation will contain many noises that can't be filtered out.

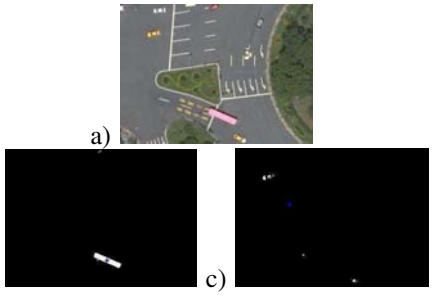

Fig. 9. a) An image from *google map* to simulate the view from aircraft; b) Automatic detection of bus; c) Automatic detection of yellow cars

The other experiment for this topic is provided as shown in Fig. 6(a). A plenty of colorful magnets are put on the wall and the ground station watches those magnets via a camera which is connected to a servo. The user then randomly selects one magnet at beginning. After the computer automatically finishes determining its color feature and learning the knowledge of the feature, the user moves the magnet at will and the computer must issue the navigation and guidance command automatically to track the object based on its knowledge of the selected magnet.

Processing time between two algorithms, *HSI* and *YCbCr*, are provided in the following table. Out of 35 random trials the successful rate of detection using *HSI* is 100%, while that using *YCbCr* is 71.43%. As a result, we conclude that *HSI* has a better performance in successful rate while *YCbCr* reacts faster. All experiments stated above demonstrate that our algorithm of real-time target detection and tracking works very well and successfully.

TABLE I

PROCESSING TIME (SEC) USING DIFFERENT ALGORITHM.

Color \ Ave. Proc. Time	Ave. (*HSI*)	Ave. (*YCbCr*)
Dark Blue	0.120	0.022
Red	0.113	0.020
Purple	0.125	0.017
Yellow	0.102	0.017
Pink	0.110	0.019
Green	0.115	0.018
Blue	0.106	0.018

VII. CONCLUSION

In this paper we discuss vision-based navigation under the platform of Matlab. Famous for powerful capability of matrix operation and various toolboxes in dynamics and control problems, Matlab has been used for analysis and simulation for a long term. However, Matlab is also capable of communicating with hardware, so that it can be introduced to construct the whole system, from ground station to onboard flight control system. In this paper, we discussed two types of navigation problems using machine vision technology. In order to introduce Matlab, we use the least-square error method to approximate the direction of a specific terrain. The update rate of navigating commands is up to 10 Hz, from the input of image to the transmission of navigating commands. Unlike a commonly discussed problem where infrared camera is used, we also develop an algorithm which enables us to track a moving target online with regular optical CCD camera based on automatic real-time target detection. Experiments are designed to examine our algorithm. The successful rate is more than 70% and processing time is at most $\sim$0.1 seconds for an image of size 320×240. This concludes that vision-based navigation of UAVs under the platform of Matlab is a feasible and easy way.

VIII. ACKNOWLEDGEMENT

The authors wish to thank Mr. Pei-Chung Chen, Dr. Fei-Bin Hsiao and Dr. Jing-Ming Tang and Tang's research group for their help through insightful discussion.

REFERENCES

[1] D. Arun. *Computer vision and fuzzy-neural systems*. Prentice Hall, Inc., 2001

[2] M. Benallal, J. Meunier, "Real-time color segmentation of road signs," in *Proc. IEEE Canadian Conf. on Electrical and Computer Engineering*, IEEE CCECE 2003, Canadian, vol. 3, pp.1823V1826. 4-7 May, 2003.

[3] V. Carbone, M. Carocci, E. Savio, G. Sansoni, and L. Chiffre. "Combination of a vision system and a coordinate measuring machine for the reverse engineering of freeform surfaces". *Advanced Manufacturing Technology*, Vol. 17, Pages 263-271, 2001.

[4] Y.L. Chen. *A real-time object tracking system using the stereo vision*. Master Thesis, Chung Yuan Christian University, 2003.

[5] Yu-Hsiang Cheng, *Vision-Based Track Following for Unmanned Aerial Vehicle*. Master's thesis, National Cheng Kung University, 2008.

[6] J. Franco, M. Lapierre, and E. Boyer. "Visual shapes of silhouette sets". *Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 397V404, 2006.

[7] A. Howard, C. Padgett, and K. Brown, "Real time intelligent target detection and analysis with machine vision," in *Proc. 3rd International Symposium on Intelligent Automation and Control, World Automation Congress (ISIAC-WAC '00)*, Maui, Hawaii, June 2000.

[8] F.Y. Hsiao, J.F. Shen, and Y.C. Chen, "Autonomous Flight Control of Rotorcraft UAV using Vision-based Navigation", *Proceeding (651) Control and Applications*, #651-114, 2009

[9] S. Rathinam, Z. Kim, A. Soghikian and R. Sengupta, "Vision Based Following of Locally Linear Structures using an Unmanned Aerial Vehicle", *Proc. 44th IEEE Conference on Decision and Control and European Control Conference ECC*, 2005.

[10] O. Murakami, K. Amano, Y. Ohnishi, S. Nishiyama, "Automatic Targe Detection For Vision Metrology With Using Coloured Target," *ISPRS Commission V Symposium – Image Engineering and Vision Metrology*, IAPRS Volume XXXVI, Part 5, Dresden 25-27 September 2006

[11] M. Nicholas, J. Wood, and J. Nothard, "Real time automatic target identification system for air to ground targeting," *Proc. SPIE*, Vol. SPIE-5988, pp. 142-151, 2005

[12] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Visually-guided landing of an unmanned aerial vehicle," *IEEE Transactions on Robotics and Automation*, vol. 19, pp. 371V381, Jun 2003.

[13] W. G. Shadeed, D. I. Abu-Al-Nadi, M. J. Mismar, "ROAD TRAFFIC SIGN DETECTION IN COLOR IMAGES," in Proc. 10th IEEE Int. Conf. Electronics, Circuits and Systems, ICECS 2003, Sharjah, United Arab Emirates, vol. 2, pp. 890V893, 14-17 Dec., 2003.

[14] http://www.mathwork.com/

[15] C.E. Daniell, D.J. Kemsley, W.P. Lincoln, W.A. Tackett, G.A. Baraghimian, "Artificial neural networks for automatic target recognition", Optical Engineering, Vol. 31(12), pp. 2521-2530, Dec. 1992.

[16] S. Greenberg, J. Guterman, "Neural-network classifiers for automatic real world aerial image recognition", Applied Optics, Vol. 35, No. 23, pp. 4598-4608, Aug. 1996.

[17] W.A. Thoet, T.G. Rainey, D.W. Brettle, L.A. Slutz, F.S. Weingard, "ANVIL neural network program for three-dimensional automatic target recognition", Optical Engineering, Vol. 31, No. 12, pp. 2532-2539, Dec. 1992.