YAAS-Yet Another Altitude Sensor based on Stereoscopic Vision

JOSÉ OLIVEIRA, RUI GONÇALVES
MAP-I Universidade do Minho Aveiro Porto
Doctoral Programme in Computer Science
PORTUGAL
jpo@di.uminho.pt – rjpg@fe.up.pt
http://www.map.edu.pt/i

***Abstract*—A new approach for altitude data extraction in UAV's based on stereoscopic vision is presented in this paper. We propose an alternative solution for applications where the precision acquired by normal methodology, using barometric pressure sensor and GPS, is not satisfactory. The use of stereoscopic vision to infer about the distances of objects relative to cameras is a classical method in the computer vision field. Our solution takes in account the computational restrictions impose by an embedded system (component of a Small_UAV).**

Key-Words: Stereo Vision, UAVs, Altitude estimation, autonomous landing.

I. INTRODUCTION

THIS paper report a new approach to extract the altitude data of a UAV (Unmanned Aerial Vehicle) based in computer vision techniques, in particular using the stereoscopic vision methodology. Extract the altitude of an AUV is not an easy issue to solve when the precision is crucial. Some task/maneuvers require high precision in this data field on the estimated state of the UAV, not only on the landing maneuver (Fig. 1), where inaccurate data can compromise the UAV entire mission, but also in other tasks made on regular flight time. Namely: terrain mapping, target tracking, structure tracking [[1] and even in normal navigation.

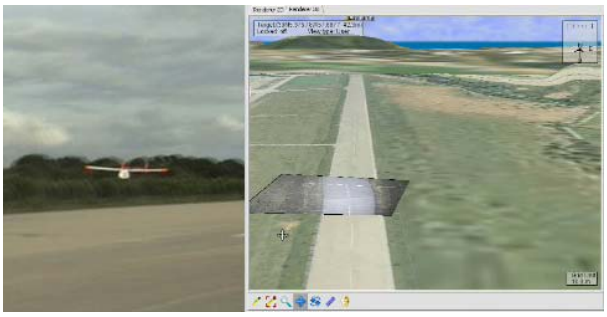

Fig. 1. At left AUV autonomous landing maneuver using pressure sensor combined with GPS for altitude estimation and knowing *a priori* the landing track altitude. At right automatic real-time terrain mapping based on the UAV pose.

The work was developed in a PC-104 system; however our laboratory (ASASF/LSTS[1]) has been using Piccolo Avionic System for small UAVs [2]. The Piccolo system includes avionics hardware and software, ground-station hardware and software, and a development and simulation environment. The normal approach to estimate altitude, used by Piccolo system, it is made by a barometric pressure sensor, and a board temperature sensor. Together these sensors provide the ability to measure true air speed and altitude. Combining the GPS positioning data, using a Kalman filter, reasonable altitude measurement can be achieved for normal flight navigation. However this methodology reveals to be insufficient for robust autonomous landing maneuver due to the imprecise nature of the sensors and the obligation of manually feeding the system with the absolute altitude (relative to the sea surface) of the landing track.

We propose a new methodology which upgrades the system for a new step of autonomation, adding back the factor of human intervention and facilitating the system to face unpredictable scenarios (e.g. autonomous emergence landing in an unknown track, atmospheric pressure variations, etc). This method uses stereoscopic vision to provide another source of data to the system. As in any other stereoscopic vision based system the setup requires accurate calibrations of the cameras. For the calibration phase, described in this paper, we used the Calib Matlab toolbox [3]. This software presents us the intrinsic parameters.

The paper is organized as follows. In section 2 we describe the platform in terms of hardware and software. In section 3 we present implementation details and how we applied the different APIs. Section 4 is used to describe the camera setup calibration using the Calib Matlab toolbox. In the last section touch the correlation and triangulation problems.

.

[1] LSTS- Laboratório de Sistemas e Tecnologias Subaquáticas

II. System Platform

In this section we describe the test platform used in terms of hardware and software:

A. Hardware

The processing module is a PM-LX-800 [8], a PC/104 form factor embedded computer. The PM-LX-800 is particularly suitable for low power and fan-less applications and is equipped with an on-board low-power consumption and high performance AMD Geode LX 800 processor. It also contains a DDR SO-DIMM socket that supports 1GB of memory.

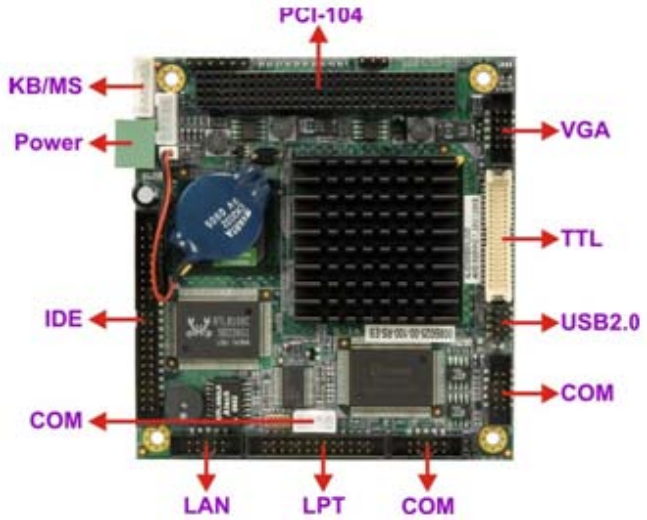

Fig. 2 - PC-104 Processing module PM-LX-800

The frame grabber is an Imagenation PXC200A [9] series color frame grabber. This frame grabber is able to handle up to four cameras and NTSC, PAL, and SECAM Video formats.

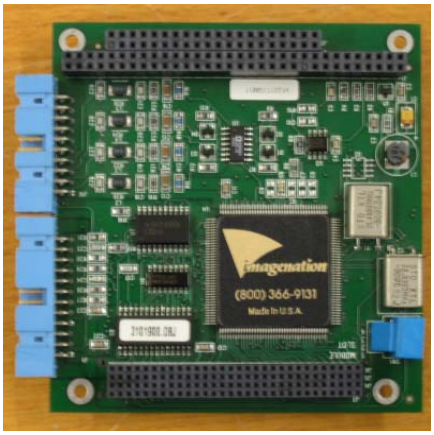

Fig. 3 - Frame Grabber Imagenation PXC200

The cameras used were 12 VDC 380 line CCD cameras.

Fig. 4 - CCD camera

B. Software

The PC104 processing unit is running Arch Linux, a Linux based operating system, with an 2.6.22 linux kernel optimized for the i586 architecture.

The image acquisition is done using the V4L2 (Video For Linux Two) API [4]. This low-level API defines a kernel interface for analog radio and video capture and output drivers. The main method of communication of this API is the ioctl system call.

The image processing is done using OpenCV [5] (The Open Source Computer Vision Library). This library includes over 500 functions implementing computer vision, image processing, and general-purpose algorithms. Its portability is rather good as it compiles and runs in every major platform: Win32, Linux, and MacOSX. Its performance can even be further boosted using the Intel IIP (Integrated Performance Primitives) library. Also, its BSD-like license makes its free for academic and commercial use.

III. Implementation

In this section we show how the combination of Video For Linux Two and OpenCV APIs was made.

A. Image acquisition

The image acquisition was performed through the V4L2 interface for two main reasons: *i)* performance, and *ii)* the lack of stability of image acquisition functions of the OpenCV library v1.0.0. The lack of stability in our setup could be exposed by the following code, which simply tried to acquire a set of images and save them to disk:

```
CvCapture* capture =
          cvCaptureFromCAM(CV_CAP_ANY);
IplImage* image = NULL;

for (i = 0; i < 30; ++i)
{
    IplImage* frame = cvQueryFrame(capture);
    snprintf(fname, 256, "frame-%03d.jpg", i);
    cvSaveImage(fname, frame);
}
cvReleaseCapture(&capture);
```

List. 1 - OpenCV code to expose the unstability of video acquisition

The execution of above code usually aborted during the dequeuing of a memory mapped buffer in the V4L driver (according to the error message, the code aborted during the invocation of the V4L2 VIDIOC_DQBUF ioctl call).

A successful image acquisition using the V4L2 interface relies in a proper setup of the image acquisition format and in the proper manipulation of memory mapped memory buffers (V4L2 ioctls VIDIOC_QBUF e VIDIOC_DQBUF).

```
fmt.type          = V4L2_BUF_TYPE_VIDEO_CAPTURE;
fmt.fmt.pix.width       = WIDTH;
fmt.fmt.pix.height      = HEIGHT;
fmt.fmt.pix.pixelformat = V4L2_PIX_FMT_BGR24;
fmt.fmt.pix.field       = V4L2_FIELD_INTERLACED;

ioctl (fd, VIDIOC_S_FMT, &fmt);
```

List. 2 - V4L2 input video image capturing format

Mplayer turned out to be one of the best Unix tools with V4L2 support. As it was rather tolerant to missing V4L2 ioctl calls, it became an excellent testing and debugging tool. Its use allows us to debug our code when we were trying to test it in a MacBook running Fedora 9 and using its iSight camera (partial success). Using the following command we can see how it is possible to specify the driver type, the device to use, and V4L2 options that can be passed to the driver:

```
mplayer  -tv \
driver=v4l2:device=/dev/video0:input=1:width=640
:height=480:fps=30  tv://
```

List. 3 - mplayer command line options to access the frame grabber

B. *Image processing*

Even though the OpenCV API could not handle in a proper way (specify the input channel of the frame grabber) the hardware device initialization and perform the video capture it was a fundamental library for image processing operation of our project. To use OpenCV we had to find a compatible image format representation in memory between V4L2 and OpenCV. Using the V4L2 image format constant V4L2_PIX_FMT_BGR24 we were able to transfer the image buffer from V4L2 to an OpenCV matrix directly without copying it pixel by pixel. The following OpenCV code creates an OpenCV object representing the image from from V4L2 image buffer.

```
CvMat img = cvMat( HEIGHT, WIDTH,
                   CV_8UC3,
                   v4l2_image_buffer );
```

List. 4 - V4L2 to OpenCV image convertion

Having two image objects, acquired from the two cameras (channels) with V4L2 API, we became able to use the OpenCV functions to resolve the stereo vision correspondence problem. The RANSAC algorithm built in OpenCV can now be used to extract matching points in the images.

IV. Camera Calibration

In order to reconstruct the 3D coordinates of the world points we have to triangulate the correspondence points, found in a previous step of image processing. For this operation to be successful we must know exactly the intrinsic and extrinsic cameras parameters. We used the Calib Toolbox to calculate these parameters. This toolkit only requires the cameras to observe a planar pattern shown at a few (at least two) different orientations. Either the camera or the planar pattern can be freely moved. The motion need not be known. Radial lens distortion is modeled [7].

Once we install the cameras in the airplane eight simultaneous shots were taken from the cameras, from the calibration pattern, and then were fed to the calibration toolbox. The following variables were obtained: Intrinsic from the left and right cameras (focal length, principle point, skew and distortion) and the extrinsic parameters of the cameras setup (rotation and transaction vectors; relative position of the right camera with regard to the left camera).

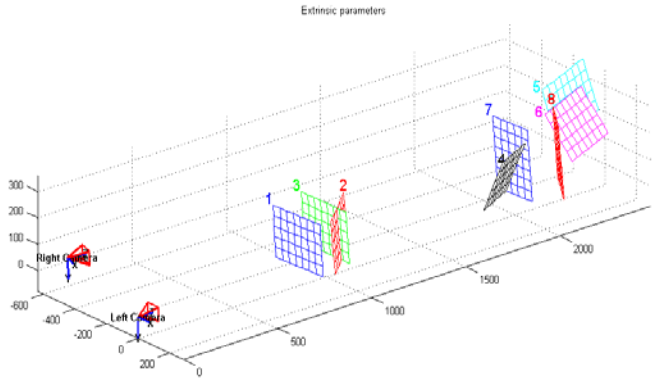

Fig. 5- Extrinsic parameters of the cameras in the AUV

With these results the computational system is fully configured. Reconfigurations of system will be only required if the position of the cameras are changed or the cameras themselves. These calibration results are also required for the correct computation of the world coordinates done by the triangulation procedure.

```
void distortion ( double k[5], double xd[2],
                  double x[2]);
void normalize ( double x_kk[2], double fc[2],
                 double cc[2], double kc[5],
                 double alpha_c, double xn[2]);
void triangulation (
   double xL[2], double xR[2],
   double om[3], double T[3],
   double fc_left[2], double cc_left[2],
   double kc_left[5], double alpha_c_left,
   double fc_right[2], double cc_right[2],
   double kc_right[5], double alpha_c_right,
   double XL[3], double XR[3]
);
```

List. 5 - Interface of the triangulation procedure library

V. Future Work

For different mission profiles, e.g. landing and normal flight, it would be advisable have different preset cameras pose configurations. The cameras pose could change using pan-and-tilt mechanisms. To facilitate the correspondence problem, particularly during the landing manoeuvre, a laser beam could be used. The use of digital cameras would release the frame grabber hardware, releasing payload, and CPU time.

Acknowledgment

Thanks for professor Manuel João Ferreira for providing the triangulation code and to professors Miguel Coimbra and Cristina Santos for insightfull comments and João Cunha for suggestions to the hardware setup.

References

[1] R. Bencatel, J.Correia, J.Sousa, G. Gonlçalves, Élói Pereira, "Video Tracking Control algorithms for UAV" , Proceedings of DSCC 2008 ASME 2008 Dynamic Systems and Control Conference October 20-22, 2008, Ann Arbor, Michigan, USA.

[2] Piccolo home page, <www.cloudcaptech.com>, 2008

[3] Calib Matlab toolbox, <http://www.vision.caltech.edu/bouguetj/calib_doc/> , 2007

[4] Video for Linux Two API Specification; Revision 0.24; Michael H. Schimek and al.; <http://www.linuxtv.org/downloads/video4linux/API/V4L2_API/>

[5] Open Source Computer Vision Library; <http://www.intel.com/technology/computing/opencv/>

[6] Learning OpenCV- Computer Vision with the OpenCV Library; Gary Bradski, Adrian Kaehler; first edition August 2008 (est.); <http://oreilly.com/catalog/9780596516130/>

[7] Flexible Camera Calibration By Viewing a Plane From Unknown Orientations; Zhebgyou Zhang; Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on Volume 1, Issue , 1999 Page(s):666 - 673 vol.1;

[8] PM-LX-800 PC104 Board, <http://www.ieiworld.com>.

[9] ImageNation PXC200 Frame Grabber, <http://www.imagenation.com/pxcfamily.html>