JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 7, October 2010

Statistical Algorithm for Attitude Estimation from Real-Time Aerial Video

Rami D. Abousleiman* , Osamah A. Rawashdeh† , and Mohammad-Reza Siadat‡
Oakland University, Rochester, Michigan 48309

DOI: 10.2514/1.47957

Almost all autonomous unmanned aerial vehicles are used for reconnaissance and intelligence gathering roles. This means that cameras, video transmitters, and/or video recorders are already integrated in the system and are part of the unmanned aerial vehicle payload. Current attitude estimation sensors are expensive, heavy, and consume more power than most micro aerial vehicles can tolerate. Vision-based attitude estimation can be used to augment inertial sensors for increased accuracy, or as primary pitch and roll sensing resulting in reduced vehicle cost, size, and weight. This paper presents a fast, real-time algorithm to estimate pitch and roll angles for an aerial vehicle from video frames captured using a downward-pointing camera with a mounted fisheye lens. The fisheye lens is installed to ensure the visibility of most of the earth's horizon at sufficient altitudes. Attitude angles are estimated by the horizontal and vertical movement of the horizon circle, which moves in relation to the center of the video frame image. The system was tested and implemented on a radio controlled plane and the results proved to be successful with over 85% of the results within $\pm 3^\circ$ when compared to a traditional inertial measurement unit.

I. Introduction

ADEQUATE autonomous flight of unmanned aerial vehicles (UAVs) requires several telemetric variables to be measured and controlled, some of which are altitude, pitch, roll, yaw, and air speed. Conventionally, UAVs use inertial measurement units (IMUs), which integrate gyroscopes and accelerometers, to estimate roll and pitch angles [1,2]. Considering that many UAVs carry a camera for surveillance or other imaging purposes, it would be advantageous to extract information from the in-flight video to aid in estimation of the telemetric variables that are essential for successful autonomous flying.

This paper describes an image processing algorithm that is applied to real-time aerial video stream captured through a downward-pointing camera equipped with a fisheye lens. The fisheye lens ensures the visibility of the entire earth's horizon, even at relatively low altitudes. The roll and pitch angles are estimated by the horizontal and vertical movement of an estimated circle in relation to the center of the video frame. The results captured are compared to a commercial IMU to test for accuracy. The algorithm first converts the image into grayscale and then uses Canny edge detection to find the image foremost edges. The selected image edge points are divided into four sets, based on their location, and used as inputs to a circle estimator using direct least-square fitting method. For each of the four sets, the algorithm then finds the radius and center of the circle that lies on the best set of edge points. The roll and pitch angles are estimated from the horizontal and vertical position of the estimated circle center and radius

Received 2 November 2009; accepted for publication 8 September 2010.

* Graduate Student, Department of Electrical and Computer Engineering, AIAA Student Member, rdabousl@oakland.edu
† Assistant Professor, Department of Electrical and Computer Engineering, AIAA Member, rawashd2@oakland.edu
‡ Assistant Professor, Department of Computer Science and Engineering, siadat@oakland.edu

in relation to the center and radius of the frame's main image. The processing stages of the algorithm are described and the examples of roll and pitch angle estimation from aerial fisheye video frames are shown and compared to actual IMU data.

The remainder of this paper is organized as follows. Section II overviews related work. Section III shows the system setup. Section IV discusses the video processing algorithm for estimating the horizon's center and radius in the aerial images. Section V explains how roll and pitch angles are estimated. Section VI previews the results and discussions. Section VII describes future work. Finally a conclusion is given in Sec. VIII.

II. Related Work

Using vision to control a robot is not new. Researchers have been investigating the use of vision in the feedback control loop since the early 1970s [3]. The term "*visual servoing*" was first used in 1979 by Hill and Park to refer to a control system that uses vision directly in the robot control feedback loop [4]. Practical implementations have been successfully developed and tested in manufacturing robots especially in the assembly lines. But the demand for more vision controlled robots has spawned research in many different directions, including real-time image processing techniques.

Some work has been done in applying optical flow and pattern recognition techniques that focus on using moiré analysis and other feature tracking techniques in aerial images [5]. This procedure will acquire the six degrees of freedom that is essential for the operation of vehicles in close proximity to other crafts and landing platforms. In other related research, Beyeler et al. used optic flow to help estimate the attitude angles by using a gray-level complementary metal oxide semiconductor linear camera but attitude control was for the most part passively stabilized by the airplane geometry [6]. Different sensors integrated with vision were used for attitude estimation. Grzywna et al. [7] developed an autopilot that uses vision coupled with global positioning system (GPS) and altitude sensors but this system had human input interface, which meant that humans had to interfere to fix any unexpected behavior. Gurtner et al. [8] used a microcontroller-based flight control system. Their system gathers data from multiple sensors and is capable of closed-loop control to enable autonomous flight and navigation. A 3D position sensor and a GPS were used to assist in the control of the aircraft. The common theme between all of the work discussed above and the proposed algorithm in this paper is that vision sensors are used to perform attitude angle estimation. While all the above approaches use vision to augment other sensors, we treat the video image receiver as the sole sensor input that will be used to estimate the attitude angles.

Demonceaux et al. [9–11] discussed the use of catadioptric systems for attitude estimation and they used different approaches. While they also relied on a single image sensor to perform attitude angle estimation, their algorithm was different than ours in many ways. In one paper, horizon estimation was achieved using Markov random fields for image segmentation to detect the horizon line. The detected line is then projected on an equivalent sphere model to form a catadioptric system [9]. In another paper, an image is segmented into two classes, the earth and the sky, using Markov random fields segmentation based on color cues [10]. Finally in a third paper, only horizontal and vertical line estimation was performed to approximate the attitude, assuming that a UAV will only be flying in an urban environment [11]. Demonceaux's algorithms took some significant time to process each frame, which can be as much as 5 s for each image [10]. Another work similar to the one done by Demonceaux is presented by Mondragón et al. [12]. They used an omnidirectional vision system to do roll, pitch, and other UAV telemetric calculations. They also used a catadioptric system and an image processing algorithm to isolate the skyline.

The work presented in this paper estimates the attitude angles by looking at the horizon and by performing circle estimation. The best circle fit is then selected and used to find the absolute roll and pitch angles. The developed system makes use of wide-angle images generated by a fisheye lens. Reduction in image quality and lens distortion is handled though experimental characterization of the sensor and lens setup. The developed algorithm analyzes each video frame and estimates roll and pitch angles in less than 850 ms, practically exceeding in speed the proposed approaches in related work.

III. System Setup

The video acquisition setup is based on a basic video camera and a wide-angle fisheye lens. A low-cost charge coupled device (CCD) camera with a one-third-inch Sony lens and 420 television lines (TVL) horizontal-resolution

Fig. 1 Fisheye lens and its barrel distortion.

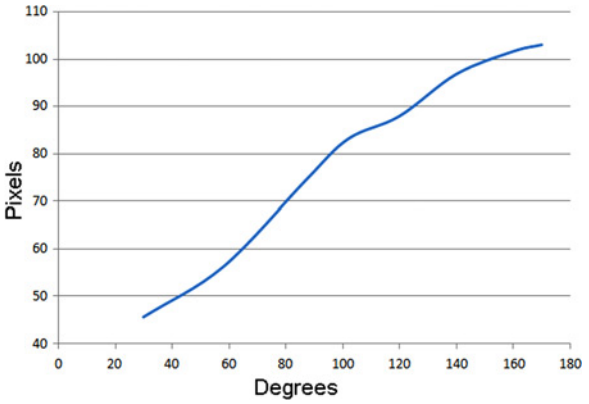

Fig. 2 Relation between degrees and pixels.

is used along with a 2.10 mm focal length aspherical lens providing 188° super wide-angle sight allowing full view of the horizon (Fig. 1). The super wide-angle of view allows full capture of the horizon even at relatively low altitudes.

Wide-angle lenses suffer from barrel distortion. Barrel distortion causes points on the image plane to be displaced in a nonlinear fashion. Furthermore, it affects the image by reducing intensification as pixels move farther away from the optical axis. As shown in Fig. 1, the lens' barrel distortion warps the imaged regular rectangular grid to a spherical appearance. Fisheye lenses take advantage of barrel distortion to map very wide-angles of the object plane onto the limited area of the CCD sensor. An experiment was conducted to quantify the barrel distortion, by taking images of the horizon using a downward-pointed camera with the 2.10 mm employed fisheye lens. The experiment incorporated changing the degree of view for the camera (angle between the line perpendicular to the plane and the optical axis where the camera is pointing) and recording how the horizon area increases or decreases. The angle of the camera was changed along one axis, say the X-axis. It was observed that both the x and y pixel coordinates changed. This relationship was mapped and plotted as shown in Fig. 2. A linear trend was assumed, and thus a linear equation can be used to translate the output of the image processing algorithm from pixels to degrees that can be used as an input by the avionics system [13].

The IMU–camera setup used during preparing this paper was tested using a small radio controlled plane. The complete system setup is shown in Fig. 3. A wide-angle fisheye lens was installed on the camera to replace the normal view angle one (3.6 mm). The camera is then mounted on the center of the IMU box in a vertical way. The IMU–camera setup is then mounted on the radio controlled plane. The plane was flown and data were recorded over the time of flying. The radio controlled plane also carried a video transmitter that transmitted video to the ground station and it also carried a ZigBee RF module to transmit IMU data angles. The real-time video recorded at the ground station is later analyzed and compared with the actual IMU data to test for accuracy.

Fig. 3 The IMU and camera setup as mounted on the radio controlled plane.

IV. Video Processing

The setup, source code, and testing of the algorithm is implemented in Matlab. At a high-level, the algorithm performs the following five steps for each frame in order:

1) Gray scaling;
2) Edge detection and grouping;
3) Circle fitting in each group;
4) Best circle selection;
5) Estimating attitude angles from selected circle.

First, an audio video interleave video is fed into the Matlab script. The script then reads the video frames as matrices. An input frame is called *FRAME_ANALYZED*. The algorithm implemented to find roll and pitch angles can be summarized in the following next steps. *FRAME_ANALYZED* is first converted into a grayscale image as shown in Fig. 4a. Canny edge detection is then applied to *FRAME_ANALYZED* to obtain an edge detected binary image as shown in Fig. 4b. Canny detector is considered to be the optimal edge detector and it is known for having the lowest error rate and thus it is the one selected to be used in the paper. Canny detector finds the edges by searching for the local maxima of the gradient of the input image. The gradient is calculated using the derivative of the Gaussian filter. This method used two thresholds that are set here to be 0.13 and 0.45. If the gradient is less than 0.13 it is discarded, if it is greater than 0.45 it is considered to be an edge, and if it is between 0.13 and 0.45 it will only be considered to be an edge if it is connected to another edge. After Canny has been applied, edges found are marked and put into four different arrays (*EDGES_TOP*, *EDGES_BOTTOM*, *EDGES_LEFT*, and *EDGES_RIGHT*). Each of these

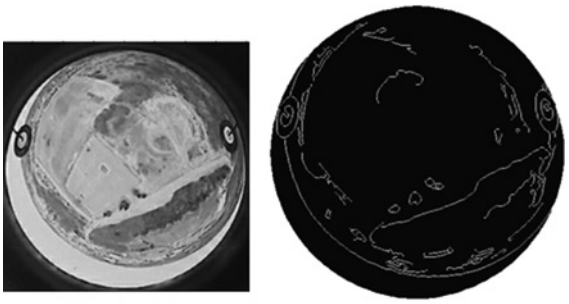

Fig. 4 a) Original image after gray scaling and b) after applying the Canny detector.

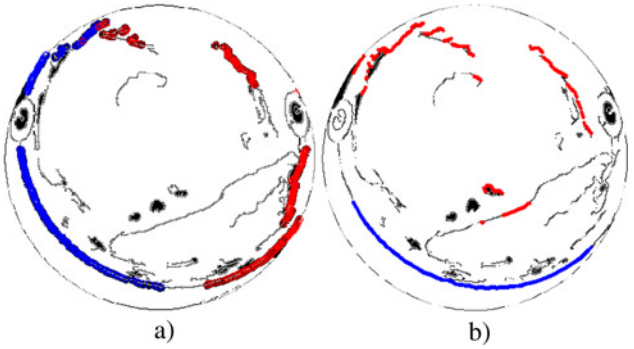

Fig. 5 a) *FRAME_ANALYZED* edges for LEFT and RIGHT and b) *FRAME_ANALYZED* edges for TOP and BOTTOM.

arrays contains n set of points which mark the $x-y$ coordinate position of the edge points in *FRAME_ANALYZED*. Figure 5a shows the foremost left and right edges points and Fig. 5b shows the foremost top and bottom edges.

The four obtained arrays are again divided into $n/6$ subsets. Each subset contains the coordinates of six randomly picked edge points. Then each one of these subsets is used to do circle estimation using the direct least-square fitting method. The output will be a set of points representing (X, Y, and R), where X and Y represent the center of the estimated hypothetical circle and R represents its radius. The results are put in 12 different arrays, four representing the radius (one for TOP, one for BOTTOM, one for LEFT, and one for RIGHT), four representing the X-axis coordinate (one for TOP, one for BOTTOM, one for LEFT, and one for RIGHT), and, finally, four representing Y-axis coordinate (one for TOP, one for BOTTOM, one for LEFT, and one for RIGHT).

The kernel density probability density function (PDF) of each of the 12 generated arrays is then calculated. Matlab's KSDENSITY function was used with default parameters. KSDENSITY computes a probability density estimate of the sample in a vector X. KSDENSITY evaluates the density estimate at 100 points covering the range of the data. The estimate is based on a normal kernel function, using a window parameter (bandwidth) that is a function of the number of points in X. The output is convoluted in the following order: PDF of the radius from TOP is convolved with PDF of the X-axis from TOP and the output is convoluted with the PDF of the Y-axis from TOP. This step is repeated three more times, one for each side until four plots are obtained each representing one side (TOP, BOTTOM, LEFT, and RIGHT). The maximum peak of each plot is calculated and recorded. The maximum peak will help decide which side to pick to do the most accurate circle estimation. In this example, the TOP side will generate a scarcer radius, X-axis, and Y-axis data and its convolution would not peak as high as the BOTTOM side. Figure 6 shows the four generated output plots of each side after convolution. It is clear that the plot of the BOTTOM side has the highest peak for this specific frame. The plots in Fig. 6 show that the plot representing the bottom side peak is the greatest. Thus we discard all the sides and focus now on the bottom side which contains the points of interest. No information is extracted from Fig. 6, instead it helps in making a decision on which set of edges we have to look at to help estimate the correct circle. Looking back at the BOTTOM side, Fig. 7 shows the kernel density function plots for the radius, X-axis, and Y-axis which were generated earlier, these plots will contain the necessary information to extract the correct and most accurate radius, X-axis, and Y-axis coordinates of the circle center. Looking at the peak values of PDF R BOTTOM, PDF X BOTTOM, and PDF Y BOTTOM the following observations are acquired: PDF R has a maximum value at index 57. Index 57 gives 189.9194. PDF X has a maximum value at index 43. Index 43 gives 179.9474. PDF Y has a maximum value at index 53. Index 53 gives 221.6769. A new circle is estimated and for demonstration purposes the new circle is plotted on *FRAME_ANALYZED* with white pixels. Figure 8a shows the original *FRAME_ANALYZED* just after Canny edge detection and Fig. 8b shows the output of the algorithm.

V. Roll and Pitch Angle Estimation

The algorithm estimates a circle by calculating its radius length, and the $x-y$ coordinates of its center. These circle parameters will be used to perform absolute roll and pitch angle estimations. Assuming that the UAV is not going to

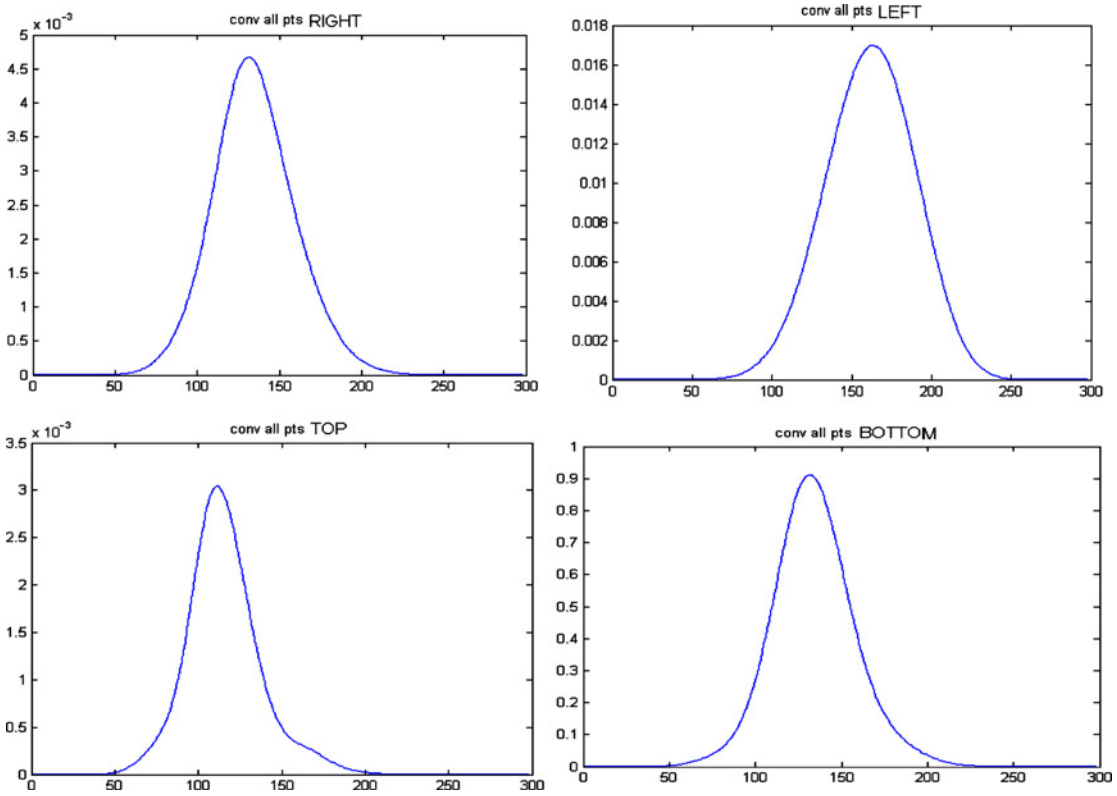

Fig. 6 Plots after convolution for LEFT, RIGHT, TOP, and BOTTOM sides.

maneuver in extreme angles, which causes circle deformations, the circle assumption generates acceptable results, as discussed in Sec. VI.

The original circle's (black full circle in Fig. 8a) parameters are R_0, X_0, and Y_0 and the newly estimated circle's parameters are R_1, X_1, and Y_1, where R_i represents the radius length and X_i and Y_i represent the location on the center of the circle on the X- and Y-axes, respectively. Figure 9 shows the parameters with the two circles plotted on the same axis. The pitch's angle is defined as rotation around the X-axis and roll angle around the Y-axis. Basically, the area of interest can be considered to be the following set of points: $R_0 - (R_1 \cap R_0)$.

First check if $X_1 + R_1 < X_0 + R_0$, if this is the case, then *PITCH_AXIS* will be called $-(X_1 + R_1) + (X_0 + R_0)$ if not, then *PITCH_AXIS* will be set to$(X_1 - R_1) - (X_0 - R_0)$. The same consideration is done for the *ROLL_AXIS*. If $Y_1 - R_1 < Y_0 - R_0$, then *ROLL_AXIS* will be set to $(-Y_1 - R_1) + (Y_0 + R_0)$ if not, then *ROLL_AXIS* will be set to $(-Y_1 + R_1) + (Y_0 - R_0)$. Once *PITCH_AXIS* and *ROLL_AXIS* are found, calibration can be done by assuming any tilting position and calculating *PITCH_AXIS* and *ROLL_AXIS* and relating their values to the actual roll and pitch angles found by any IMU. Once this one time calibration is done, absolute roll and pitch angles can be calculated by simply calculating the values of *PITCH_AXIS* and *ROLL_AXIS* without referring to the IMU. If the setup changes for any reason, for example, a new lens or a new CCD camera sensor is installed, the calibration process has repeated.

VI. Results and Discussions

The algorithm was tested in various environmental and climate conditions and over 85% of the results were found to be within $\pm 3^\circ$ when compared to an IMU. The system showed good results for most of the given conditions. A test was done in optimal weather conditions (sun with minimal clouds) and several IMU and video frame reading were acquired at the same time to be compared and analyzed. The reading's sample space included around 150 reading and the difference between the video analysis output and the IMU sensor reading reported slight differences. Figure 10a and b shows the pitch degree output from the actual observed IMU data and the algorithm, respectively. Figure 11

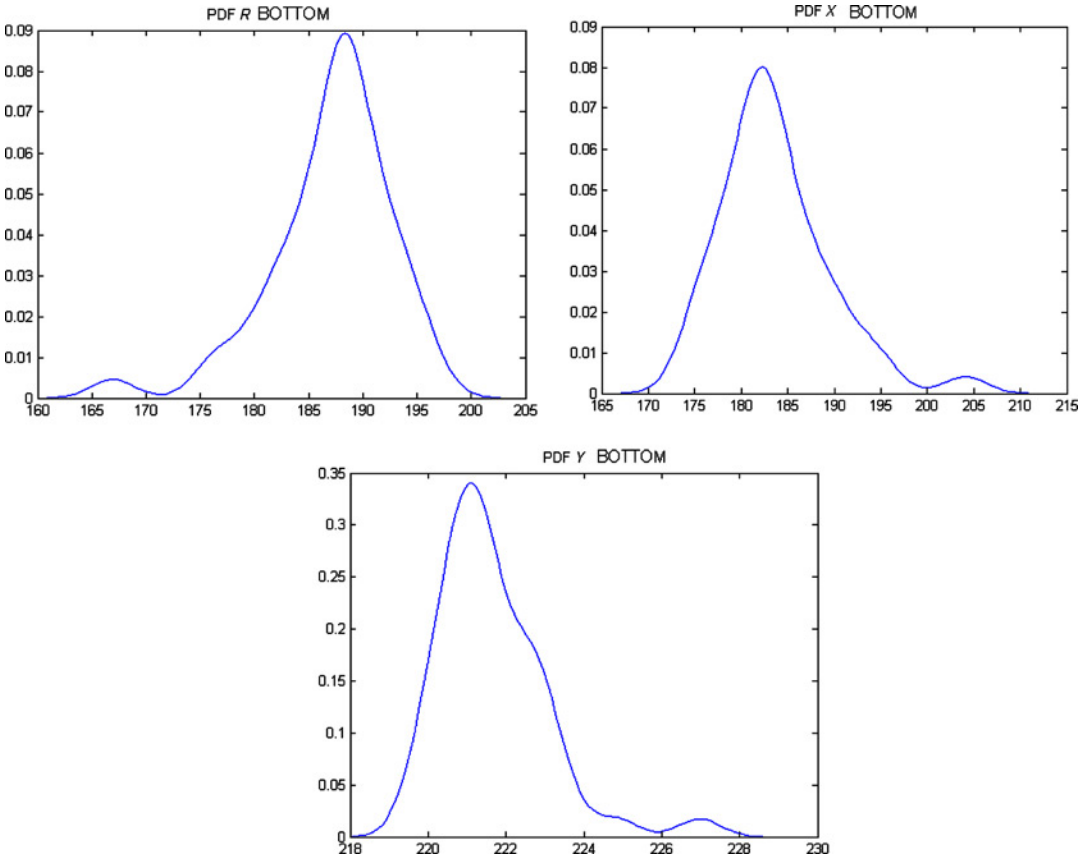

Fig. 7 Kernel density function plots for *R* BOTTOM, *X* BOTTOM, and *Y* BOTTOM, respectively.

shows the stem plot differences between the system pitch angles estimation analysis and the IMU reading. Figure 12a and b shows the roll angle degree output from the actual observed IMU data and the algorithm under the same time interval, respectively. Figure 13 shows the stem plot differences between the system roll angle estimation analysis and the IMU reading. The mean value of the difference between the IMU pitch and the calculated pitch is -0.8851 and the standard deviation is 1.2389. The mean value of the difference between the IMU roll and the calculated roll is 0.7258 and the standard deviation is 1.1766. As observed, the results are precise and consistent. These results prove that the algorithm is viable for estimating roll and pitch angles of a UAV.

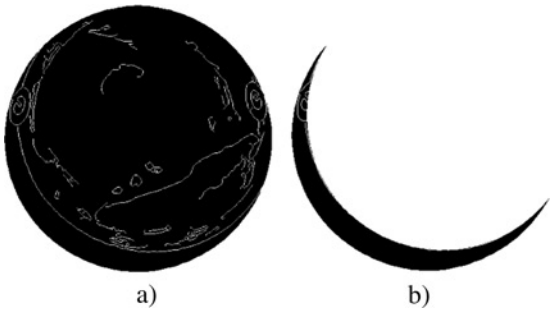

Fig. 8 a) Canny edge detection results and b) output of the algorithm.

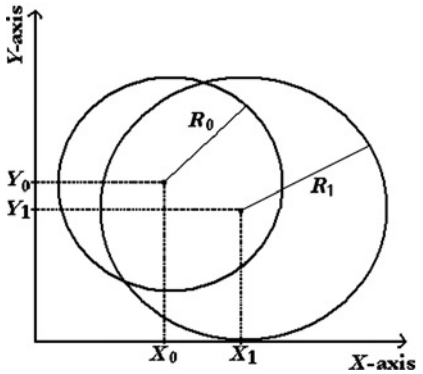

Fig. 9 Original circle with the newly estimated one and their parameters.

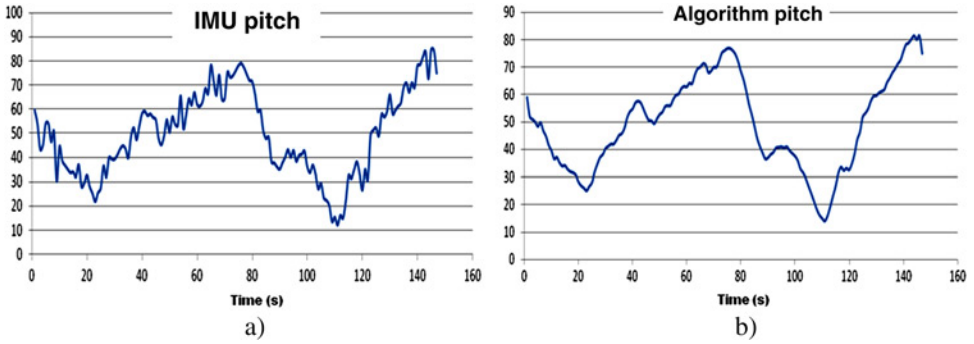

Fig. 10 a) Pitch angles output from the IMU and b) pitch angles output from the algorithm.

On average, the algorithm took around 800 ms to process each frame. A 1.46 GHz computer running Windows Vista and Matlab 7.6.0 (R2008a) was running the Matlab script. The proposed system was tested on several noise levels and Gaussian white noise was applied to various video frames. Different signal to noise ratios (SNR) were tested and recorded. Matlab's 'imnoise' function was used to generate the noise. The SNR of an image is defined as the ratio of the mean pixel values to the standard deviation of the pixel values. The effect of noise on various images is shown in Figs. 14–16. The images shown in Figs. 14–16 show different noise levels applied to an image with an original SNR of 6.28 before applying the noise. The system tolerated noise levels that caused a SNR of 5.23 on average. The performance will start to decrease as soon as the contrast ratio between the horizon and the earth starts to decrease (e.g., due to darkness or fog). Assuming $\pm 3^\circ$ output accuracy is an acceptable result, the algorithm had

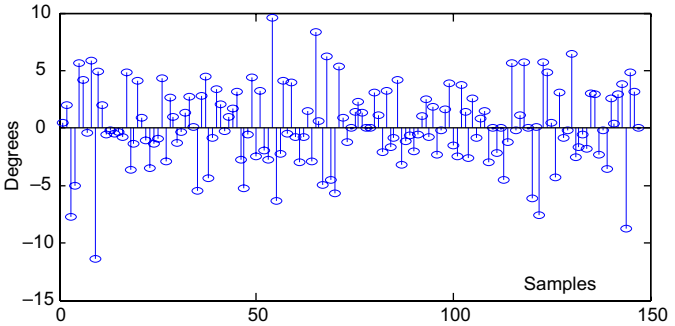

Fig. 11 Pitch angle difference between the algorithm and the IMU.

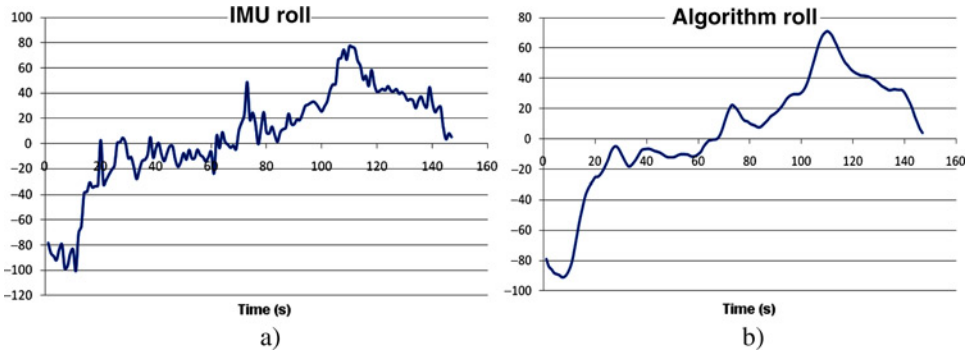

Fig. 12 a) Roll angles output from the IMU and b) roll angles output from the algorithm.

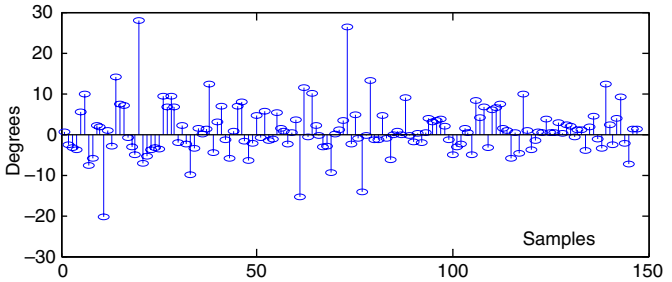

Fig. 13 Roll angle difference between the algorithm and the IMU.

a success rate of over 85%. In extreme snowy weather conditions with temperatures of around −18°F, the camera stopped functioning correctly and its output turned blue-tinted. In that case, the algorithm's success rate dropped to less than 70%. The sun will cause the horizon–sky contrast ratio to decrease; this will cause the success rate to drop to 75%. This drop only occurs when the camera has been pointed towards the sun's beam long enough to cause saturation.

The algorithm assumes a 600 by 420 input video frame, this yields to 2.3 pixels to degree ratio and thus the resolution is considered to be around 0.5°. In some cases, the system failed to generate the correct output. The error is caused by different factors all occurring at the same time. In the preceding example, a frame that failed is discussed

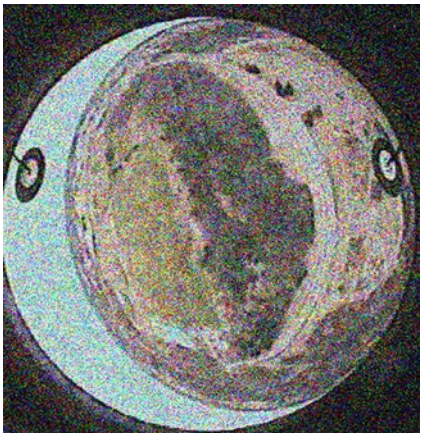

Fig. 14 A frame that produced successful output with a SNR of 5.25.

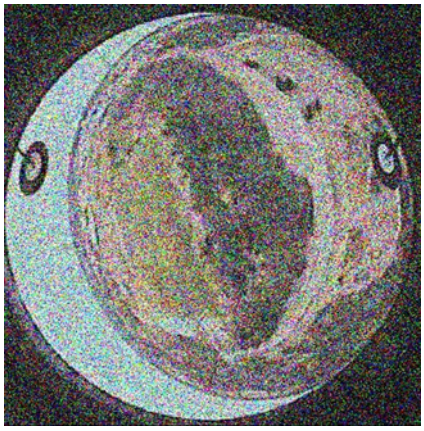

Fig. 15 A frame that produced failed output with a SNR of 5.22.

Fig. 16 Failed output due to a 5.2 SNR as applied to Fig. 15.

along with the reasons that yielded to its failure. Figure 17 shows a single video frame that is used as an input to the system along with its Canny edge detection output. Figure 18 shows an unsuccessful result. Looking at the reasons that caused the system to fail, the convolution peak results are:

$$\text{max\_conv\_R} = 0.031$$

$$\text{max\_conv\_L} = 7.98 \times 10^{-4}$$

$$\text{max\_conv\_BOTTOM} = 6.34 \times 10^{-4}$$

$$\text{max\_conv\_TOP} = 9.28\text{E} \times 10^{-4}$$

Although the left side should give the highest peak but it did not, the maximum peak was reported to be at the right side. The right side generated the highest peak mainly due to two reasons. First, looking at the Canny output in Fig. 17, Canny was not able to detect a small edge because it happened that the asphalt in the parking lot decreased the contrast ratio between the horizon and the earth and thus Canny did not report that part as an edge. It also happened that the road in the right side of the video frame is taking a circular structure. These two reasons caused the system to be misled by the fake edge. Urban environments such as roads might take circular shapes and might cause problems. One way to solve this problem is to ignore circles that are too big or too small because the approximate value of the earth's radius is known and can be computed suing altitude and optical sensor characteristics. Urban environment

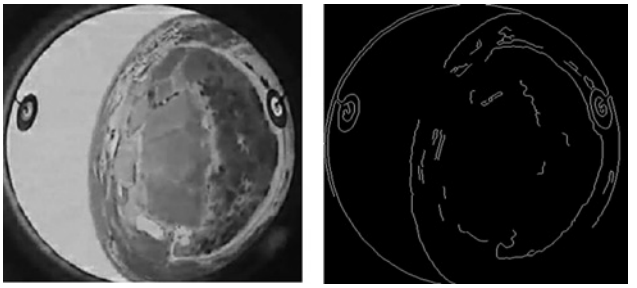

Fig. 17 An input frame that failed along with its Canny detector output.

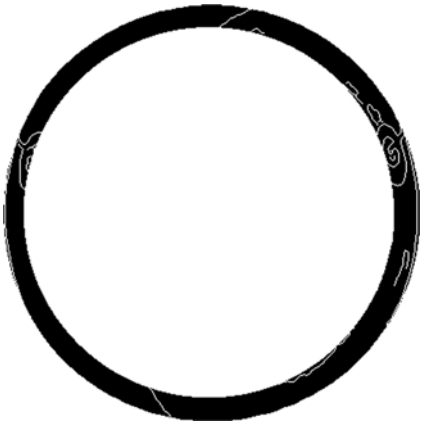

Fig. 18 The algorithm's result with Fig. 17 used as input.

as well as big obstacles such as big trees or mountains can cause the system to fail at lower altitudes. When such obstacles fill most of the camera's frame they block the horizon and thus the circle estimation would fail.

Reduced accuracy was observed due to the sun in the field of view and snow ground covering. The next examples will illustrate few such cases and discuss the reasons why some of them did fail. The sun analysis was done on 104 different frames that were directly hit by the sun. The sun caused the performance to degenerate and yielded to less accurate results. Seventy five percent of the results were reported as acceptable. The main reason behind this performance degeneration is that the contrast ratio between the sky and the earth will decrease and Canny would not report the earth as an edge. Figure 19 shows a sample video input with a direct sun hit and the equivalent Canny output. Figure 20 shows the system's output. This specific frame was able to resist the sun effect even though the right side had around 50% loss of the good points due to the sun hit.

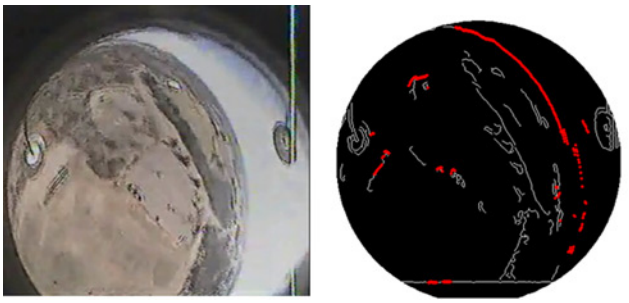

Fig. 19 Video frame with a direct sun hit.

Fig. 20 System's successful result.

Fig. 21 Failed video frame input.

In the other example, the system was not able to successfully generate good results, consider Fig. 21, for example. The edges points are shown in Fig. 22, and the result is shown in Fig. 23. Although the system was successfully able to pick the correct side, which is the RIGHT in this case, but the RIGHT points did not generate a correct circle due to the very noisy edge. One way to prevent the causes of these errors is to use a photo resister that detects the sun hits and change the Canny thresholds accordingly.

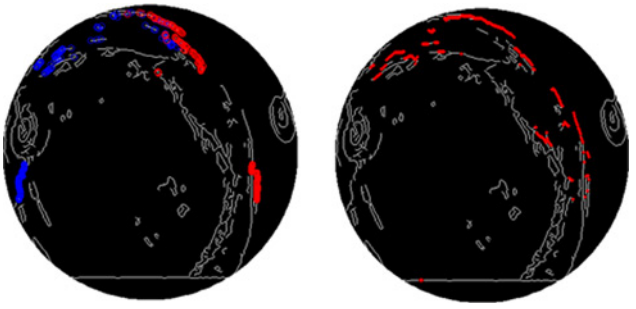

Fig. 22 Edges detected.

Fig. 23 Failed result due to the sun effect.

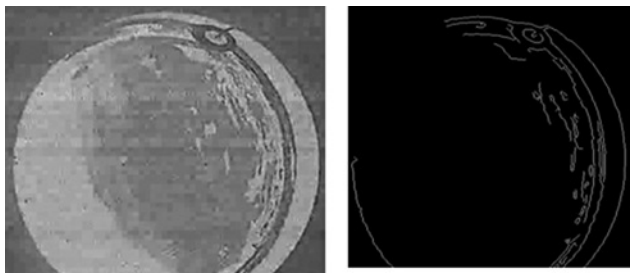

Fig. 24 Example of extreme weather condition video frame analysis.

The snow and cold temperatures effect on the other side had different concerns. Whenever extreme cold weather is noticed the camera starts to fail. The snow analysis was done on 66 video frames, and results reported to be 75% successful. Snow had less bad effects than the sun. This specific test was done in a temperature of around 0°F. At this temperature, the camera started to fail proving a bluish output but still the system proved to be successful 75% of the time. Figures 24 and 25 show a successful case in which the system correctly estimated the correct circle.

Fig. 25 Successful system output of Fig. 24.

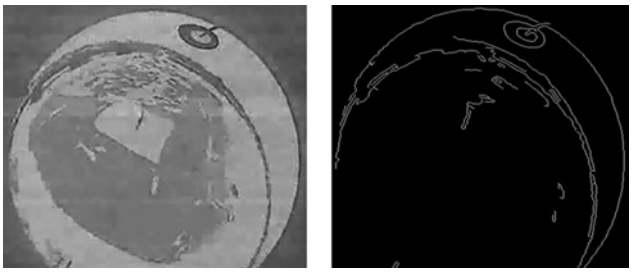

Fig. 26 Video frame that produced bad results.

Fig. 27 Failed result due to the snow effect.

Another example, in the same situation is shown in Fig. 26. Figure 27 failed to produce a correct result. A closer look at the reasons that caused this error will yield us to conclude the following. First, the system picked the correct side which is the RIGHT side here. Looking at the PDF plots of the radius, x location of the center and the y location of the center, one can note that the PDF of the radius and the PDF y coordinate of the center point give the correct results. The PDF plot of the x coordinate of the center point generated a PDF that is shown in Fig. 28. A closer look

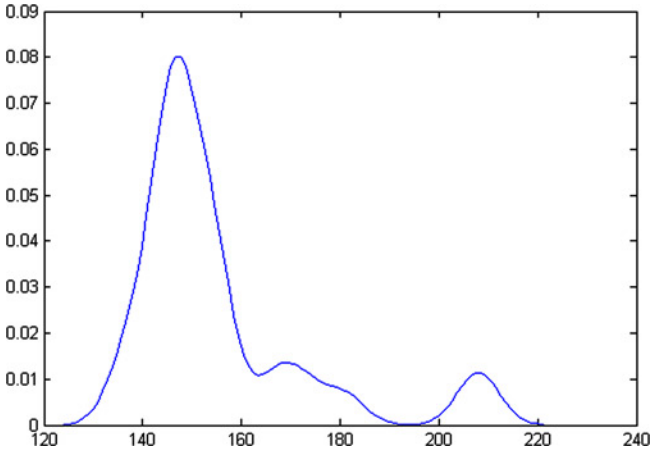

Fig. 28 PDF plot of the X-axis location of the center of the hypothetical circle.

at this plot one can conclude that the bad points caused a peak at around 145 and there is another smaller peak at around 210. The system chose the 145 peak, which is the wrong one. The correct peak is the one at around 210 but the bad points conquered the PDF to generate this unwanted peak. One solution is to look at the frame that happened just before this one, and make a judgment that the roll or pitch angles would not jump a value of 65 pixels in this short interval and thus discard the result.

VII. Future Work

Further improvements can be done on the algorithm to help the system to perform better or to extend the system to be applied on different applications. The following are some options for improving the robustness of the proposed algorithm and how these improvements may be beneficial for the overall system performance.

Canny edge detection thresholds may be manipulated in real-time to produce better results especially in different weather or environmental conditions. The developed algorithm may change the Canny thresholds depending on specific parameters of the input image such as calculating the mean contrast value of the input image and change the Canny thresholds accordingly.

Another development area is the expansion of the algorithm to measure calibrated altitude from the radius length of the earth in the video frame. In the current work, the relation between the earth's radius and altitude is neglected because it varies very slightly in our tests. For applications that require high altitude stabilization, the fisheye property can be further studied to perform altitude stabilization.

The direct least-square fitting method can be further improved to predict the accuracy of the circle estimation. The accuracy variable can be used as an extra input variable to our system to help make a better circle estimation based on prioritizing higher accuracy results and neglecting those with less accuracy.

Gray scaling of the images is done to speed the processing and due to the fact that grayscale images are easier to handle in Matlab. A color image contains more information, which we did not use. Future work must be able to make use of all of the information content in the color image. Color images might yield better results especially that Canny estimates better edges in color images.

VIII. Conclusion

In this paper, a fisheye lens mounted on an aerial downward-pointing camera is used with an image processing algorithm to obtain UAV pitch and roll angles. Although several of the parameters that might affect the algorithm's result, such as edge detection, sun, snow, and artificial surrounding are likely to vary in different environments and might help increase error, our experiments showed that the proposed vision-based system can serve as a viable pitch and roll estimator. The proposed system is less costly and lighter compared with the existing systems and yet the estimated pitch and roll angles proved to be reasonably accurate when compared to an actual IMU unit. The system also proved to be robust even in different environments such as rain, snow, and fog. Results proved that the fisheye video approach is capable of performing attitude estimation as an IMU backup or even as a replacement.

References

[1] Rawashdeh, O., Yang, H., AbouSleiman, R., and Sababha, B., "Microraptor: A Low-Cost Autonomous Quadrotor System," *Proceedings of the 2009 ASME/IEEE International Conference on Mechatronic and Embedded Systems and Applications*, 2009, paper DETC2009-86490.

[2] Mostafa, M. M., and Hutton, J., "Direct Positioning and Orientation Systems: How Do They Work? What is the Attainable Accuracy?" *Proceedings of the American Society of Photogrammetry and Remote Sensing* (*ASPRS*) *Annual Conference*, St. Louis, MO, 23–27 April 2001.

[3] Shirai, Y., and Inoue, H., "Guiding a Robot by Visual Feedback in Assembling Tasks," *Pattern Recognition*, Vol. 5, No. 2, 1973, pp. 99–108.
doi: 10.1016/0031-3203(73)90015-0

[4] Hill, J., and Park, W. T., "Real-Time Control of a Robot with a Mobile Camera," *Proceedings of the Ninth International Symposium on Industrial Robots*, Washington, DC, March 1979, pp. 233–246.

[5] Tournier, G., Valentiy, M., Howz, J., and Feron, F., "Estimation and Control of a Quadrotor Vehicle Using Monocular Vision and Moiré Patterns," *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, AIAA, Reston, VA, 2006; also AIAA paper 2006-6711.

[6] Beyeler, A., Mattiussi, C., Zufferey, J., and Floreano, D., "Vision-Based Altitude and Pitch Estimation for Ultra-Light Indoor Microflyers," *Proceedings of the IEEE International Conference on Robotics and Automation* (*ICRA*), Orlando, FL, 15–19 May 2006, pp. 2836–2841.

[7] Grzywna, J., Plew, J., Nechyba, M., and Ifju, P., "Enabling Autonomous MAV Flight," *Proceedings of the Sixteenth Florida Conference on Recent Advances in Robotics*, Dania Beach, FL, May 2003.

[8] Gurtner, A., Boles, W., and Walker, R., "A Performance Evaluation of Using Fisheye Lenses in Low-Altitude UAV Mapping Applications," *Proceedings of the Twelfth Australian International Aerospace Congress* (*AIAC*), Melbourne, Australia, 2007.

[9] Demonceaux, C., Vasseur, P., and Pegard, C., "Omnidirectional Vision on UAV for Attitude Computation," *Proceedings of the IEEE International Conference on Robotics and Automation*, Orlando, FL, 15–19 May 2006, pp. 2842–2847.

[10] Demonceaux, C., Vasseur, P., and Pegard, C., "Robust Attitude Estimation with Catadioptric Vision," *Proceedings of the International Conference on Intelligent Robots and Systems* (*IROS*), Beijing, 9–15 Oct. 2006, pp. 3448–3453.

[11] Demonceaux, C., Vasseur, P., and Pegard, C., "UAV Attitude Computation by Omnidirectional Vision in Urban Environment," *Proceedings of the IEEE International Conference on Robotics and Automation*, Roma, 10–14 April 2007, pp. 2017–2022.

[12] Mondragón, I. F., Campoy, P., Martinez, C., and Olivares, M., "Omnidirectional Vision Applied to Unmanned Aerial Vehicles (UAVs) Attitude and Heading Estimation," *Robotics and Autonomous Systems*, Vol. 58, No. 6, 2010, pp. 809–819.

[13] Gurtner, A., Duncan, G., Greer, R., Mejias, L., Walker, R., and Boles, E., "Investigation of Fisheye Lenses for Small-UAV Aerial Photography," *IEEE Transactions on Geosciences and Remote Sensing*, Vol. 47, No. 3, March 2009, pp. 709–721. doi: 10.1109/TGRS.2008.2009763

Christopher Rouff
Associate Editor