Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2011
August 29-31, 2011, Washington, DC, USA

DETC2011/MESA-007

DRAFT: VISUAL ATTITUDE ESTIMATION FOR LOW-COST PERSONAL REMOTE SENSING SYSTEMS

Tobias Fromm, Long Di, YangQuan Chen

Center for Self-Organizing and Intelligent Systems
Department of Electrical and Computer Engineering
Utah State University
Logan, UT 84322, USA
tobias.fromm@hs-weingarten.de

Holger Voos

Dept. of Electrical Engineering and Computer Science
Ravensburg-Weingarten University of Applied Sciences
88250 Weingarten, Germany

ABSTRACT

Remote Sensing using unmanned aerial vehicles (UAV) is gathering a lot of attention at the moment by researchers and developers, especially in terms of low-cost aircrafts which still maintain sufficient accuracy and performance.

This paper introduces a low-cost approach to increase airworthiness by using a forward-looking camera to estimate the attitude of a UAV. It not only focuses on using machine learning to classify ground and sky, but also uses image processing and software engineering methods to make it fault-tolerant and really applicable on a miniature UAV. Additionally, it is able to interface with an autopilot framework to being used productively on flight missions.

INTRODUCTION

Remote sensing denotes the process of gathering information about an object without actually getting into contact. This is especially interesting when it comes to tasks that are known to be dull, dirty or dangerous. One way to apply remote sensing is the usage of an unmanned aerial vehicle (UAV) to investigate the object of interest from the air. Equipped with different kinds of sensors, they can be used for various tasks.

AggieAir [1] is a UAV being developed at the Center for Self-Organizing and Intelligent Systems (CSOIS) at Utah State University. It is used for taking multispectral aerial images of a region with the ability to fly autonomously using the open-source autopilot *Paparazzi* [2]. The main goal aimed at with *AggieAir* is to decrease production and operation costs of remote sensing platforms while still maintaining airworthiness, i.e. the possibility for an aircraft to be certified as suitable for safe flight. One way to increase the UAVs reliability is to make different kinds of sensors cooperate to enable a more stable flight. In the optimal case, a newly added sensor can compensate for any drawbacks the existing sensors have.

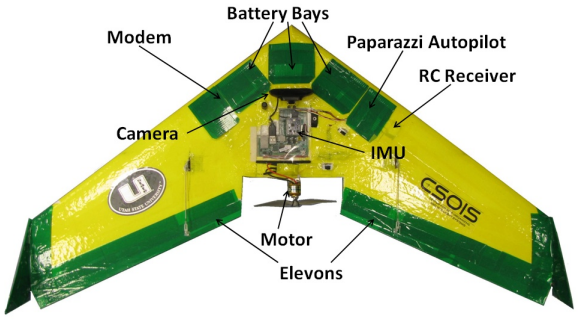


FIGURE 1. 48" UAV COMPONENT LAYOUT

Since cameras are quite cheap in comparison to other sensors like inertial measurement units (IMUs), vision-based sensing provides an approach which is very cost-effective in the long run. The goal of the approach presented here is to visually es-

timate the attitude of the UAV, taking images with a forward-looking camera mounted on top of it. This is accomplished for roll and pitch, but not for yaw, because a UAV moving in parallel to the earth surface does not observe a horizon change.

There are already some existing projects which deal with visual attitude estimation, some using only image processing [3] [4] [5] [6], others also adding machine learning techniques [7] [8] [9]. Indeed, to the authors' best knowledge, there are no projects so far with such a low-cost, but effective approach being productively used on an actual UAV. The main contribution of this paper consequently is a working system which can be applied on scientific and commercial flight missions. This approach has the capability of being used in a flexible and fully autonomous way on a miniature UAV.

VISUAL ATTITUDE ESTIMATION

Approach

Using computer vision to calculate an aircraft's attitude can be achieved in several different ways. The basic image preprocessing steps are comparable in the existing approaches, but some of them additionally rely on machine learning methods to make it more stable and reliable. Especially when it comes to changing environmental conditions (clouds, fog, lighting conditions), machine learning based approaches generally provide better results. In the extreme case of uneven horizon lines which cannot be detected by conventional approaches, machine learning can still provide suitable results. Several existing papers on the subject contain detailed surveys on the accuracy of their approaches which reach values of more than 90% [7] [8].

In addition to the detection of the horizon line from the input image, also an estimation of roll and pitch angle has to be done. So, the whole visual attitude estimation framework is separated into the following steps:

0. **build classifier** using training images (to be done only once)
1. **classify the image**, separating between sky and ground
2. **find all possible horizon lines**
3. determine the **optimal horizon line**
4. **calculate attitude** from horizon line and camera parameters

Sky/Ground Classification

There are many different algorithms which can divide a data set into different pre-defined classes. For this approach, a decision tree was chosen due to simple usage, high classification speed and intuitive understandability [10]. In an experiment carried out in [7], the accuracy of different classifiers on a sky/ground classification task was found to be only slightly different. Therefore, classification does not seem to be significantly improved or worsened, no matter which of those popular classifiers is used.

Significant differences on the accuracy may result by the choice of classification attributes. For the sky/ground classification task, color values are certainly the first thought. To take these into account, preference was given to the HSV (Hue-Saturation-Value) model instead of the usual RGB (Red-Green-Blue) model. This model allows for an easier relation of a visual color to its values. Additionally, in this way the classifier can be kept as simple as possible. Another very important reason for the preference of HSV is its slightly better performance on image classification tasks according to [11].

Using color information only to find the sky in an image may not lead to sufficient results as soon as the images become more difficult to classify. Because of that, texture information is used beneath the well-known color information, this means small square chunks of a certain size (e.g. 5×5 pixels), centered around each pixel. In detail, in addition to the hue, saturation and value values of each pixel, for each color channel, the following attributes of the respective chunks are calculated, x_i denoting the respective color value of pixel i :

Mean:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1}$$

Standard Deviation:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2} \tag{2}$$

A classification example can be found in Figure 2(b).

Representation of Horizon Lines

The usual approach to depict straight lines

$$y = mx + c \tag{3}$$

has some drawbacks like no possibility to represent vertical lines and near-infinite values for near-vertical lines, so the *normal form* can be used to represent horizon lines instead where

$$\rho = x\cos\theta + y\sin\theta. \tag{4}$$

For simplification reasons inside the software framework, we use this principle while taking the height of the horizon line's center above ground and its orientation towards the x-axis as parameters instead.

For the reason of exploding computing time, we cannot simply take all possible horizon lines into account and attempt to find the best fitting one. Depending on the resolution of the line orientation θ and the normal line length ρ, which may rise up to the diagonal length of the image, a maximum parameter space size of

$$N_{steps} \cdot \sqrt{{w_{img}}^2 + {h_{img}}^2} \tag{5}$$

may result with N_{steps} denoting the number of steps to evaluate according to the angular resolution (e.g. 360 steps for a 1° resolution), w_{img} and h_{img} denoting width and height of the image, respectively. For not having to evaluate this number of choices for every pixel, some image preprocessing is done.

Image Smoothing

With the help of image *smoothing*, the quantity of possible horizon lines can be significantly reduced [8]. This approach is based on the fact that binary images always show strong transitions between different image regions as there are only two possible values. We use a built-in function of the computer vision framework *OpenCV* with a radius of three pixels. However, smoothing a binary image creates soft transitions (cf. Fig. 2(c)). If now only these "soft" values in between the originally used binary values are considered as candidates for separating image regions, the amount of possible horizon lines decreases dramatically. Thus, after smoothing the binary image, it is reduced to only the parts where transitions occur (cf. Fig. 2(d)). This is done by taking only the gray values in the range of approximately $[0.3, 0.7]$ after normalizing them to to $[0, 1]$.

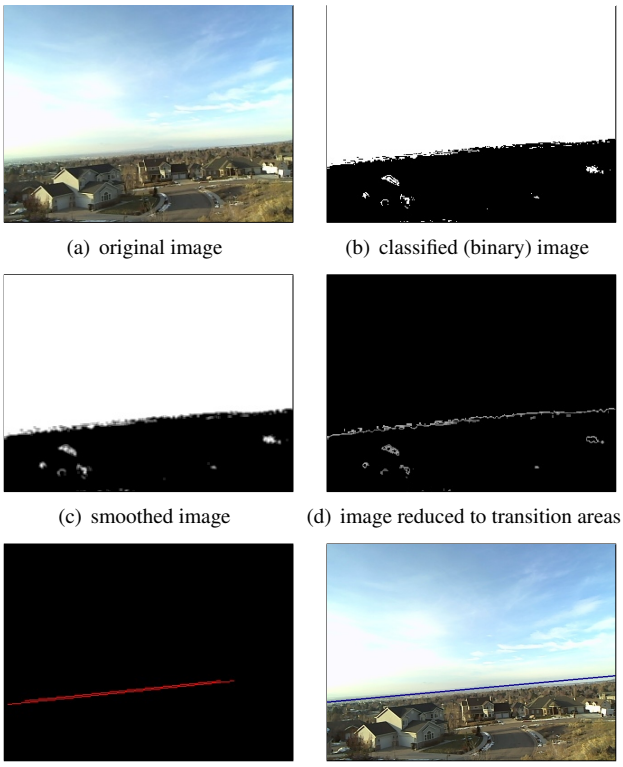
Finding Possible Horizon Lines via Edge Detection

Edge detection is done by applying the *Hough Transform* on the preprocessed image. To be precise, instead of the *Standard Hough Transform* (SHT), we use the improved *Probabilistic Hough Transform* (PHT) which uses only a subset of points lying on the lines to detect, thus drastically reducing computing time requirements [12].

Determining the Correct Horizon Line

After the input image has been classified into horizon and non-horizon pixels, the properties of a line separating both clusters must be found. This can be done by using a cost function which calculates the amount of wrongly located pixels with certain horizon line properties. Evaluating the cost function for every possible horizon line configuration, it has to be minimized to find the horizon line which fits best.

The cost function C [8] is described as follows, where x_{ij} denotes the classification of the very pixel at coordinates i/j. It has to be parsed for each image pixel.



(a) original image (b) classified (binary) image

(c) smoothed image (d) image reduced to transition areas

(e) edge detection results (f) final result, horizon line marked

FIGURE 2. HORIZON DETECTION PROCESS EXAMPLE

$$C(\rho,\theta) = \sum_{i,j} e(x_{ij})$$
$$e(x_{ij}) = \begin{cases} 0 \text{ if } \textit{true positive} \\ 1 \text{ if } \textit{false positive} \\ 1 \text{ if } \textit{false negative} \\ 0 \text{ if } \textit{true negative} \end{cases} \tag{6}$$

After having determined the costs for all lines H_i, the one with the lowest costs is taken:

$$H_{opt}(\rho,\theta) = argmin\, C_i(\rho,\theta) = argmin \sum_{i,j} e(x_{ij}) \tag{7}$$

Eventually, the optimal horizon line H_{opt} could be determined. The whole horizon line detection process is summarized in the example images in Figure 2.

Attitude Estimation from Horizon Line

Dusha et al. [4] developed a method for calculating roll and pitch angles from the horizon line. This horizon attitude estima-

TABLE 1. CAMERA PARAMETERS NECESSARY FOR ATTITUDE ESTIMATION

	Logitech HD Pro Webcam C910	*Microsoft LifeCam Cinema*
Focal Length	4.3 mm	4.5 mm
Sensor Diagonal	$\frac{1}{2.5}$ inches	$\frac{1}{4}$ inches
Sensor Height	4.29 mm	2.40 mm
Sensor Width	5.76 mm	3.20 mm
Pixel Height	0.03575 mm	0.02 mm
Pixel Width	0.036 mm	0.02 mm

tion problem is based on several transformations between world and camera coordinate systems and can be found in detail in [4].

Camera Parameters. Some assumptions about the camera used on the airplane have to be made for the following calculations. We used standard USB webcams of type *Logitech HD Pro Webcam C910* and *Microsoft LifeCam Cinema* on *AggieAir*. The following camera parameters have to be taken into account:

Sensor Size: Size of the image sensor usually given as a value of $\frac{1}{x}$ inches. The resulting sensor height and width can be looked up in [13].

Pixel Size: Actual size of one pixel depending on the sensor size. Images are taken with a resolution of 640×480 pixels and later reduced to 160×120, therefore one pixel has the following attributes:

$$\text{Pixel Height} : h_p = \frac{\text{Sensor Height}}{120} \tag{8}$$

$$\text{Pixel Width} : w_p = \frac{\text{Sensor Width}}{160} \tag{9}$$

Focal Length: Distance between optical center of camera lens and its corresponding point on the image plane (i.e. the sensor in case of a digital camera) in mm.

The respective parameter values for the used camera models are listed in Table 1.

Attitude Estimation. Described in detail in [4], the aircraft's roll angle (output in radians) can be calculated as follows:

$$\phi = arctan(m_{line}) \tag{10}$$

where m_{line} is the slope of the horizon line on the image plane which can easily be calculated from the horizon line's representation.

The pitch angle is determined by

$$\theta = arctan\left(\frac{u \cdot sin(\phi) + v \cdot cos(\phi)}{f}\right) \tag{11}$$

while f denotes the focal length u and v are the metric position of an arbitrary pixel $P(x/y)$ on the horizon line as it is depicted on the *image plane* (the two-dimensional representation of the camera image located inside the camera).

u and v have to be calculated from the image's pixel coordinates first. As reference point, the middle of the horizon line with $x = \frac{w_{img}}{2}$ and $y = h_{line} - \frac{h_{img}}{2}$ is used:

$$u = w_p \cdot x = w_p \cdot \frac{w_{img}}{2} \tag{12}$$

$$v = h_p \cdot y = h_p \cdot \left(h_{line} - \frac{h_{img}}{2}\right) \tag{13}$$

where h_p and w_p are pixel height and width in mm, h_{line} is the height of the horizon line's center, and h_{img} and w_{img} represent image height and width.

Hence, equation 11 can be modified as follows:

$$\theta = arctan\left(\frac{w_p \cdot \frac{w_{img}}{2} \cdot sin(\phi) + h_p \cdot \left(h_{line} - \frac{h_{img}}{2}\right) \cdot cos(\phi)}{f}\right) \tag{14}$$

Accuracy Improvements. In an experiment presented later in this paper it became apparent that, using the aforementioned equation to calculate the pitch angle, the values were not exact enough when the roll angle rises. This is due to the fact that $sin(\phi)$ rises quickly on rising roll angle, but the only part where information to calculate pitch comes into play is in $h_p \cdot \left(h_{line} - \frac{h_{img}}{2}\right) \cdot cos(\phi)$ with the parameter h_{line}. So, as ϕ rises, the important part for calculating pitch becomes less significant. This leads to the undesirable behavior of the calculated pitch angle deviating from the real pitch angle on increasing roll.

As a solution for this problem, a constant reduction factor α

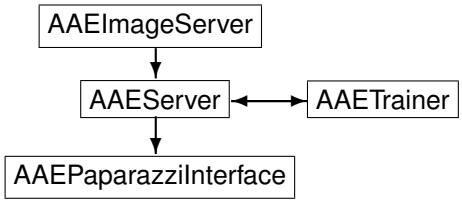


FIGURE 3. AGGIEATTITUDEESTIMATOR SOFTWARE ARCHITECTURE

was added to the equation as follows:

$$\theta = arctan\left(\frac{\alpha \cdot w_p \cdot \frac{w_{img}}{2} \cdot sin(\phi) + h_p \cdot \left(h_{line} - \frac{h_{img}}{2}\right) \cdot cos(\phi)}{f}\right) \tag{15}$$

Naturally, α has to take a value in $(0,1]$ to make the equation work properly. The respective experiment we describe in the next section eventually shows that $\alpha = 0.1$ proves as a good choice.

Software Architecture

The algorithms elaborated in this paper were implemented in a framework called *AggieAttitudeEstimator*. It uses tools like the computer vision library *OpenCV* [14], the machine learning suite *WEKA* [15]. *AggieAttitudeEstimator* is separated into four parts like shown in Figure 3. These components are responsible for the following tasks:

- *AAEImageServer*: capturing camera images, preprocessing
- *AAEServer*: image classification and attitude calculation
- *AAEPaparazziInterface*: interface with the open-source autopilot *Paparazzi* running on the UAV
- *AAETrainer*: graphical user interface to perform classifier training by making the user determine the horizon line manually on training images

Figure 4 shows the whole visual attitude estimation process as it runs in a loop on the UAV.

As the ground control station (GCS) has to be involved into the attitude estimation process at least when it comes to training the classifier, the framework includes the remote procedure call framework *Thrift* [16] to pass through machine borders and overcome platform restrictions.

With this approach, it is possible to run the whole framework on the aircraft. To the authors' best knowledge, there are not many other approaches to do visual attitude estimation on an

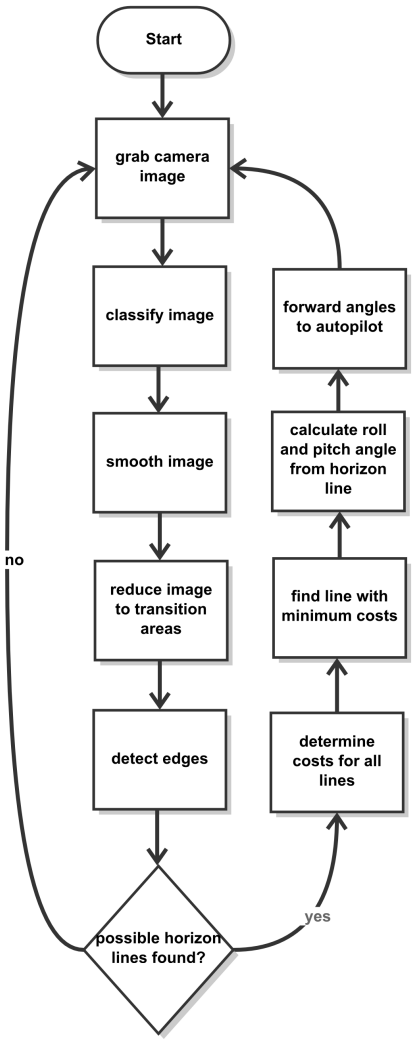
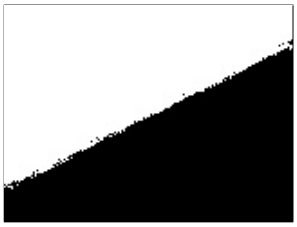


FIGURE 4. AGGIEATTITUDEESTIMATOR SOFTWARE LOOP

autonomous aircraft itself without the active help of a GCS. Most of the other research groups working on similar projects perform their image processing and/or classification on the GCS [3] or make no statement about it at all [4] [7]. The AINS Center for Collaborative Control of Unmanned Vehicles at UC Berkeley [8] is the only research group we found running their whole vision processing on the UAV so far.

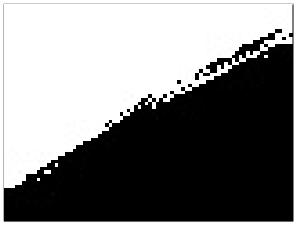
The integration with *AggieAir* was realized by the development of an I^2C interface between the onboard flight computer of type *Gumstix Overo* and the existing autopilot *Paparazzi*. In addition to the low-level hardware connection, we also developed a messaging protocol to simplify further communication between both computers. This will come handy by eventually activating better communication between components and thus more features of *AggieAir* [1].

(a) original image



(b) classified image (160×120)

(c) result (160×120)



(d) classified image (80×60)

(e) result (80×60)

FIGURE 5. RESULTS FOR DIFFERENT IMAGE SIZES

EXPERIMENTAL RESULTS

Camera images are taken with a resolution of 640×480 pixels, but they have to be resized for performance improvement. Smaller images mean increased pre-processing and classification speed, but worse classification results. Thus, we ran the visual attitude estimation process twice with the same input image, first resized to 160×120, then to 80×60 pixels. The results are shown in Figure 5. It becomes apparent that the horizon line in the lower resolution image significantly differs from the real horizon.

To be able to make a quantitative statement about this fact, we also compared the number of correctly classified pixels in images of both resolutions. For more input images taken in different landscape with varying lighting conditions, an average classification correctness of 95.3% could be achieved for images of a 80×60 resolution compared to 97.8% when using 160×120 pixel images.

In another experiment, we attempted to find the optimal reduction factor α used for attitude estimation in Equation 15. Therefore, roll and pitch values of *AggieAttitudeEstimator* were recorded for different values of α as well as IMU values as a reference. As a metric to evaluate the accuracy of the estimations in comparison with the IMU reference we use the root mean square

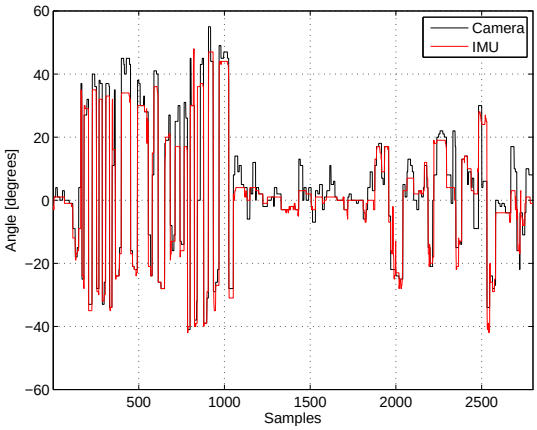


FIGURE 6. ROLL ANGLE COMPARISON

error (RMSE) which is calculated as follows:

$$RMSE(\alpha) = \sqrt{\frac{\sum_{i=1}^{n} (x_{i,IMU} - x_{i,Cam}(\alpha))^2}{n}} \tag{16}$$

with x_i as the respective measurements of IMU and camera and n the total number of samples.

Due to different message reporting frequencies in the autopilot software, dynamic time warping (DTW) is used to adjust the time scales of IMU and visual attitude measurements. This stretches the measurements on the time axis to fit the respective other graph and complicates a quantitative comparison. Nevertheless, its applicability in this context is still justified as only subject to change is parameter α and therefore the graphs' amplitudes. We also cannot take the RMSE as an absolute accuracy value, but only to compare different measurements taken under similar conditions. There are no major changes on the time axis within the experiment which would possibly result in completely different warping results, invalidating error calculations. So, the combination of DTW and RMSE calculation forms a feasible metric to estimate an accuracy difference between two possible values of α.

In Figure 6, the recorded roll angles are depicted. Figure 7 shows the comparison of IMU reference and visual attitude estimation pitch angles for α values 1.0, 0.3 and 0.1. Further reduction does not provide significant changes to the RMSE, thus 0.1 was chosen as the best value to set for α, modifying the original equation the least.

Both figures clearly show a positive difference of the pitch values in the middle section, regardless of the reduction factor. The reason for this is a general limitation of any visual attitude estimation approach: For high pitch values, the camera may not

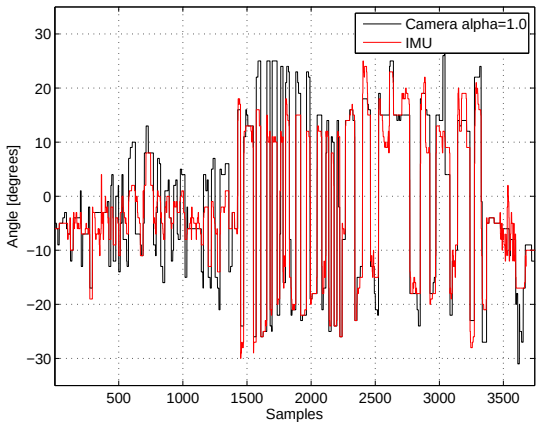
actually have the horizon line in view and therefore produce random output. This can happen everytime the UAV ascends with a big angle of attack and is not avoidable as well as the jitter in the IMU measurements at the end of the test sequence. In a sensor fusion approach like the one we developed in succession of this framework [17], these problems can be taken into account and thus gotten rid of.

CONCLUSION AND FUTURE WORK

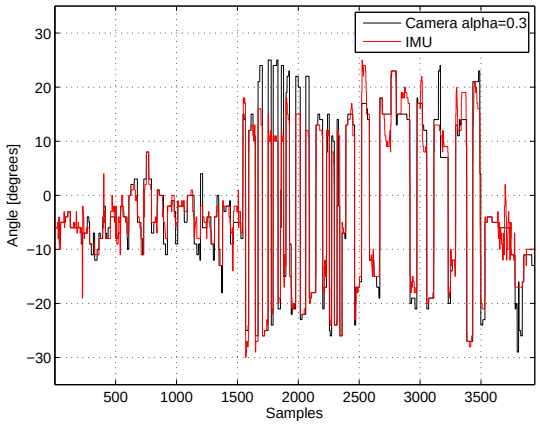
In this paper, we have combined some established and well analyzed approaches to develop a visual attitude estimation framework being able to run completely autonomously on a UAV. This system is able to detect the horizon in a camera image, determine the aircraft's attitude angles and use these as sensor inputs for the autopilot.

The accuracy mainly depends on how exactely and extensively the classifier was trained before, if the parameters are set to focus rather on accuracy or on performance and, of course, on the weather and lighting conditions. We included some image pre-processing steps and other considerations and implemented a connection to the Paparazzi autopilot system. This enables the system to be used productively. In particular, the ability of using Visual Attitude data as an input of a sensor fusion system [17] which combines the advantages of different low-cost sensors is very important. That system directly benefits from the approach introduced herein and absolutely improves the airworthiness of *AggieAir*.

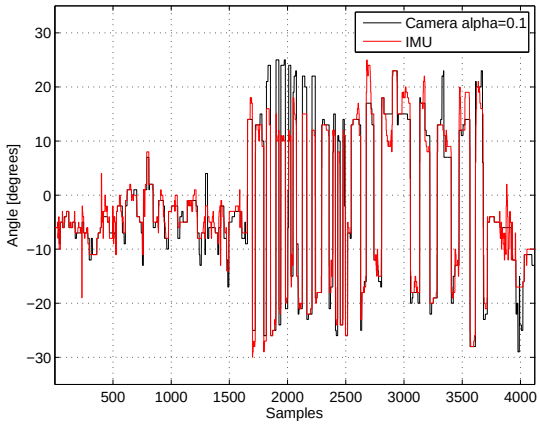
Some improvements have to be made to *AggieAttitudeEstimator* to make it more usable, though. So far, the performance of *AAEServer* is not sufficient to productively run it on the UAV onboard computer with all parameters set to reach the accuracy shown in this paper. This part of the software is currently being re-implemented to improve its performance. As soon as this is finished, the complete framework should reach an update frequency close to 1 Hz while running in fully autonomous mode which is sufficient for our sensor fusion algorithm.



(a) pitch for $\alpha = 1.0$ vs. IMU, RMSE: 4.94 degrees



(b) pitch for $\alpha = 0.3$ vs. IMU, RMSE: 3.84 degrees



(c) pitch for $\alpha = 0.1$ vs. IMU, RMSE: 3.67 degrees

FIGURE 7. PITCH ANGLE COMPARISON FOR DIFFERENT α

REFERENCES

[1] Coopmans, C., and Han, Y., 2009. "AggieAir - An Integrated and Effective Small Multi-UAV Command, Control and Data Collection Architecture". In Proc. of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference.

[2] Ecole Nationale de l'Aviation Civile. Paparazzi. *http://paparazzi.enac.fr/*.

[3] Ettinger, S., Nechyba, M., Ifju, P., and Waszak, M., 2002. "Towards Flight Autonomy: Vision-Based Horizon Detection for Micro Air Vehicles". In Proc. of the 2002 Florida Conference on Recent Advances in Robotics, Vol. 2002.

[4] Dusha, D., Boles, W., and Walker, R., 2007. "Attitude Estimation for a Fixed-Wing Aircraft Using Horizon Detection and Optical Flow". In Proc. of the 9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications, pp. 485–492.

[5] Zafarifar, B., and Weda, H., 2008. "Horizon detection based on sky-color and edge features". In Proc. of SPIE, Vol. 6822.

[6] Cornall, T., and Egan, G., 2004. "Measuring horizon angle from video onboard a UAV". In Proc. of the IEEE International Conference on Autonomous Robots and Agents, pp. 13–15.

[7] Fefilatyev, S., Smarodzinava, V., Hall, L., and Goldgof, D., 2006. "Horizon Detection Using Machine Learning Techniques". In Proc. of the 5th International Conference on Machine Learning and Applications (ICMLA), pp. 17–21.

[8] McGee, T., Sengupta, R., and Hedrick, K., 2006. "Obstacle Detection for Small Autonomous Aircraft Using Sky Segmentation". In Proc. of the 2005 IEEE International Conference on Robotics and Automation (ICRA), pp. 4679–4684.

[9] Todorovic, S., and Nechyba, M., 2003. "Sky/ground modeling for autonomous MAV flight". In Proc. of the IEEE International Conference on Robotics and Automation (ICRA), Vol. 1, pp. 1422–1427.

[10] Ertel, W., 2011. *Introduction to Artificial Intelligence*. Springer Verlag. (to appear).

[11] Chen, W., Shi, Y., and Xuan, G., 2007. "Identifying Computer Graphics using HSV Color Model and Statistical Moments of Characteristic Functions". In Proc. of the 2007 IEEE International Conference on Multimedia and Expo, pp. 1123–1126.

[12] Kiryati, N., Klviinen, H., and Alaoutinen, S., 2000. "Randomized or Probabilistic Hough Transform: Unified Performance Evaluation". *Pattern Recognition Letters,* ***21***(13-14), pp. 1157–1164.

[13] Wikipedia. Image Sensor Format. *http://en.wikipedia.org/wiki/Image_sensor_format*.

[14] Willow Garage, Inc. OpenCV. *http://opencv.willowgarage.com/*.

[15] The University of Waikato. WEKA. *http://www.cs.waikato.ac.nz/ml/weka/*.

[16] Apache Software Foundation. Apache Thrift. *http://thrift.apache.org/*.

[17] Di, L., Fromm, T., and Chen, Y., 2011. "Low-Cost Data Fusion System for Attitude Estimation of Miniature UAV". In Proc. of the 2011 International Conference on Unmanned Aircraft Systems. (submitted).