Above Ground Altitude Estimation for General Aviation Aircraft During Landing Phase

CS231A Project Report

Zouhair Mahboubi

***Abstract*—Accurately judging the above-ground altitude (AGL) is an important skill to safely land an airplane. Pilots rely on visual cues to determine their AGL, but it takes many hours for a student pilot to hone this skill. In this project, we explore the use of an on-board camera to provide AGL as a helping tool that could be used as part of training new pilots. We collected data from a cockpit view of an aircraft, and implemented a pipeline which detects runway edges and estimates the aircraft AGL. Final results were noisy and could benefit from Kalman filtering.**

I. Introduction

Ask a student pilot what the most difficult part of flying is, and you will often get landing as an answer (Air Traffic Control comes in as a close second). Indeed, landings account for more than a third of all general aviation (GA) accidents. While they are often non-fatal due to the relatively low energies involved, they are responsible for hundreds of damaged aircraft per year [1].

Aircraft (most of them at least) are designed to fly: Line one down the runway and throttle up, and it will pick up enough speed and take-off. In an introductory flight lesson, after a one hour ground lesson (including some simulator time), a student pilot will be able to take-off with minimal supervision. Landing on the other hand is a ”controlled crash”, where the aircraft is brought to its slowest possible flying speed and stalled a few feet near the ground. This is known as the Roundout or the Flare, and it often takes tens of hours and hundreds of landings before the student masters the art of landing and can fly solo.

What makes flaring an aircraft so difficult? There has been a fair amount of research into the physiological aspect of landing, and the focus is on the use of visual cues to time the flare. This is a rather difficult skill that pilots eventually learn in order to properly judge the time to flare [8]. Instructors and flying handbooks provide tricks and tips to help with this, but it all boils down to using the visual cues. These cues are used to estimate the height above the ground and therefore the time to contact, which in turn helps determining when the pilot needs to transition the aircraft from a descent path to a path parallel to the runway as illustrated in Fig. 1.

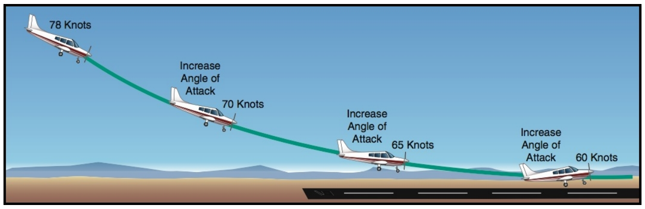


Fig. 1: When the angle of attack is increased, the lift is momentarily increased, which decreases the rate of descent. During the roundout (flare), the airspeed is being decreased to touchdown speed while the lift is being controlled so the airplane will settle gently onto the landing surface (source [2]).

While flying, a pilot has access to a wide range of instruments to help determine the state of the aircraft (airspeed, attitude, etc.), but there rarely is one for above ground altitude (AGL). A pressure altimeter is used throughout the flight to control altitude, but its slow time constant and inaccuracy are inappropriate for the flaring task. A Global Positioning System (GPS) can provide altitude, but even with selective availability turned off, vertical accuracy is only 5 meters(15 feet). If a Satellite Based Augmentation System (SBAS) such as Wide Area Augmentation System (WAAS) is available, this can be reduced to 1.3 meters(4 feet) in practice. While SBAS might be accurate enough, not all GPS chipsets (especially those in mobile devices) support it, and WAAS is currently only available in North-America [14]

II. Previous Work

The problem of accurately measuring aircraft AGL has been well studied, especially in the context of autonomous landing for aircraft which was pioneered by the British due to notoriously foggy London [13]. Most modern civilian aircraft are equipped with radar altimeters that are used as part of autoland functionalities [9]. However, radio altimeters are not used on small aircraft. This is due to weight and power constraints, and the associated costs of developing and certifying such a system. That being said, some have proposed radio altimetry to help with the task of flaring small aircraft. Vidmar implemented a system with tonal beeps proportional to AGL [12]. FlareAssist^{TM}[3] is a commercially available product which annunciates AGL below 15ft in 1-2 ft increments. However, it costs $2500 and can only be used on experimental aircraft as it is not FAA certified.

An alternative way to determine the aircraft AGL is to utilize a vision system. While the use of vision for civilian autoland systems is not ideal as it is often needed when visual conditions are poor, it is an active area of research for autonomous landing of Unmanned Air Vehicles [4]. There are a few projects that have looked into implementing vision based landing for GA aircraft [5]. Trisiripisal et al. used two cameras installed on an aircraft on approach (one on each wing-tip) to extract runway edges and other features [11]. Using stereography they determined the position and attitude of the aircraft relative to the touchdown point. Sasa et al. demonstrate that the aircraft position (normalized by the runway width) and attitude can be computed from a single image by extracting the runway edges and the horizon line [10] and demonstrated promising performance with 0.4 meters $\approx$ 1 foot standard deviation for altitude. However, they relied on runway color and brightness thresholding to detect it. Naidu and Garlin propose a more robust method for runway edge extraction, but do not use it for position extraction [7].

In this project we plan to investigate the use of a single camera to estimate an aircraft AGL during landing. The idea is to leverage the work in [10] combined with the more recent work in [7] which proposes improvements to runway edge detection. The goal is to eventually implement the piepline on a phone in order to provide a cheap and simple solution. The ubiquitous use of iPads and other mobile devices in GA cockpits means that if this proves to be robust, it is likely to be used by pilots, especially students, as a training device.

III. Problem Formulation

Determining the aircraft AGL using computer vision is effectively a problem of single view metrology. In this section, we outline the mathematical background and the proposed graphical pipeline to extract the AGL from video frames.

A. Single View Metrology

The mathematical basis behind the problem is relatively simple. The idea is that during a slanted approach to a runway, the visual angle (Ψ or θ) illustrated in Fig. 2 formed by the projection of the runway edges changes as a function of altitude. If we are able to extract this angle from a picture, and assuming we know the camera intrinsics, we can infer the position of the camera up to a scalar. This scalar ambiguity can be resolved if the width of the runway is provided a-priori. Runway widths are published by the FAA and can either be entered manually, or in a more automated fashion by using GPS to determine which runway the aircraft is approaching.

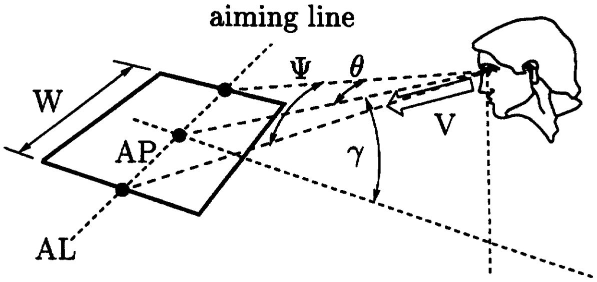


Fig. 2: Geometry of visual angle. [source [6]]

More precisely, the slope of each runway edge is related to the relative position and attitude of the aircraft. These relationships are given in [10]:

$$\tilde{k_1} = tan(atan(k_1) - \hat{\phi}) \tag{1}$$

$$\tilde{k_2} = tan(atan(k_2) - \hat{\phi}) \tag{2}$$

$$\hat{h} = \frac{\tilde{k_1}\tilde{k_2}}{\tilde{k_2} - \tilde{k_1}} \frac{cos(\hat{\theta})}{\cos(\hat{\psi})} B \tag{3}$$

Where:

- $\hat{\phi}, \hat{\theta}, \hat{\psi}$ are the roll, pitch and heading estimates.
- k_1, k_1 are the extracted runway edges
- B is the runway width
- $\hat{h}$ is the estimated altitude relative to the runway

Note that this slightly different than the cited paper. While Sasa et al. use the horizon line to estimate the aircraft attitude, we are assuming that the roll, pitch and heading are available from inertial sensors on the phone.

B. Runway Edge Extraction Pipeline

We now describe how the runway edge extraction is performed. Note that part of this work is derived from the work of [7].

Cockpit Clutter Removal: The first part of the pipeline is to differentiate between the portions of the image that constitute the out-of-the-window view, and those that constitute the interior view. The portions associated with the in-cockpit view are likely to be cluttered with information which will constitute noise when we try to detect the runway edges.

We experimented with two methods to remove cockpit clutter. In the first one, we averaged multiple frames together, and used subtracted the averaged image from each frame. In the second method, we simply used a brightness threshold to extract a mask, with the assumption that the cockpit is likely to be darker than the out-of-the-window view. In both cases, the mask was extracted in a pre-processing step, and used for all subsequent frames, i.e., the mask remained constant throughout the data processing. A third alternative, which was not tested but is likely to be the simplest and more robust, is to allow the user on the phone interface to select which parts of the image are to be masked out.

***Image Dilation**:* Once the cockpit is removed, an image dilation operation as suggested by [7] is performed. We experimented with different stencils, and eventually settled on a 3×3 square stencil.

***Edge Detection and Area opening**:* We experimented with different edge detectors, but eventually settled on using the Sobel edge detector. The results of the edge detection were further processed by an expanded version of the background mask to remove the artifial edge of the mask itself. Additionally, an area opening operation to get rid of small regions was performed.

***Hough Transform**:* The detected edges are processed using a Hough transform for line detection. The range of angles that is being searched is restricted based on the previously detected runway edges. The initial range of angles requires an initialization, but a rough guess which can be deduced from coarse GPS altitude is sufficient.

***Line Scoring**:* Each peak of the Hough transform is a candidate line in the original image. However, we are not interested in a single line, but rather in a pair of lines that represent the runway edges. Therefore, we take the top 20 peaks of the transform, and we form all possible $\binom{20}{2}$ pair combinations. Each pair is then scored and ranked. The score is designed to favor lines that exhibit the following properties:

- Assuming the pilot is approaching with a small side-slipe and along the runway centerline, we expect runway edges to be roughly symmetric to each other.
- Likewise, we expect the pilot to be able to see the horizon during the approach, and therefore the lines should intersect within the image frame.

Additionally, we expect that between successive frames, the lines should not change too much from each other. Therefore, we keep track of a smoothed intersection point, and we score the line pairs based on their distance from this vanishing point.

***Altitude Extraction**:* The slopes from the best pair in the Hough transform are used to extract the estimated altitude per Eq. (3). Note that this requires knowing the aircraft attitude. There are two ways to determine the attitude. The first one is follow the method described in [10], where the slope of the horizon line and the runway edges, as well as the position of the vanishing point, are used to infer the aircraft attitude (the horizon line can be extracted from the Hough transform). The second method is to assume that the aircraft attitude is provided by an inertial system. This is a reasonable assumption for this application since we are targetting an eventual implementation on a mobile device.

***Kalman Filtering**:* Unfortunately, due to the various different noise sources, the identified lines and therefore the extracted visual altitude are likely to be noisy. However, if we think of the visual altitude as a noisy measurement in a stochastic process describing the aircraft altitude, we can combine it with other altitude measurements in a Kalman filter to smooth the predicted AGL. Some possible measurements that could be used include GPS altitude, GPS vertical velocity, and barometric pressure. As mentioned previously, GPS on its own might not provide enough accuracy due to the large bias. Pressure altitude suffers from the same problem and can be laggy. The visual altitude on the other hand will have a fast time response and a small bias, but is likely to suffer from noise. We hypothesize that combining these two measurements would yield good results.

IV. Experimental results

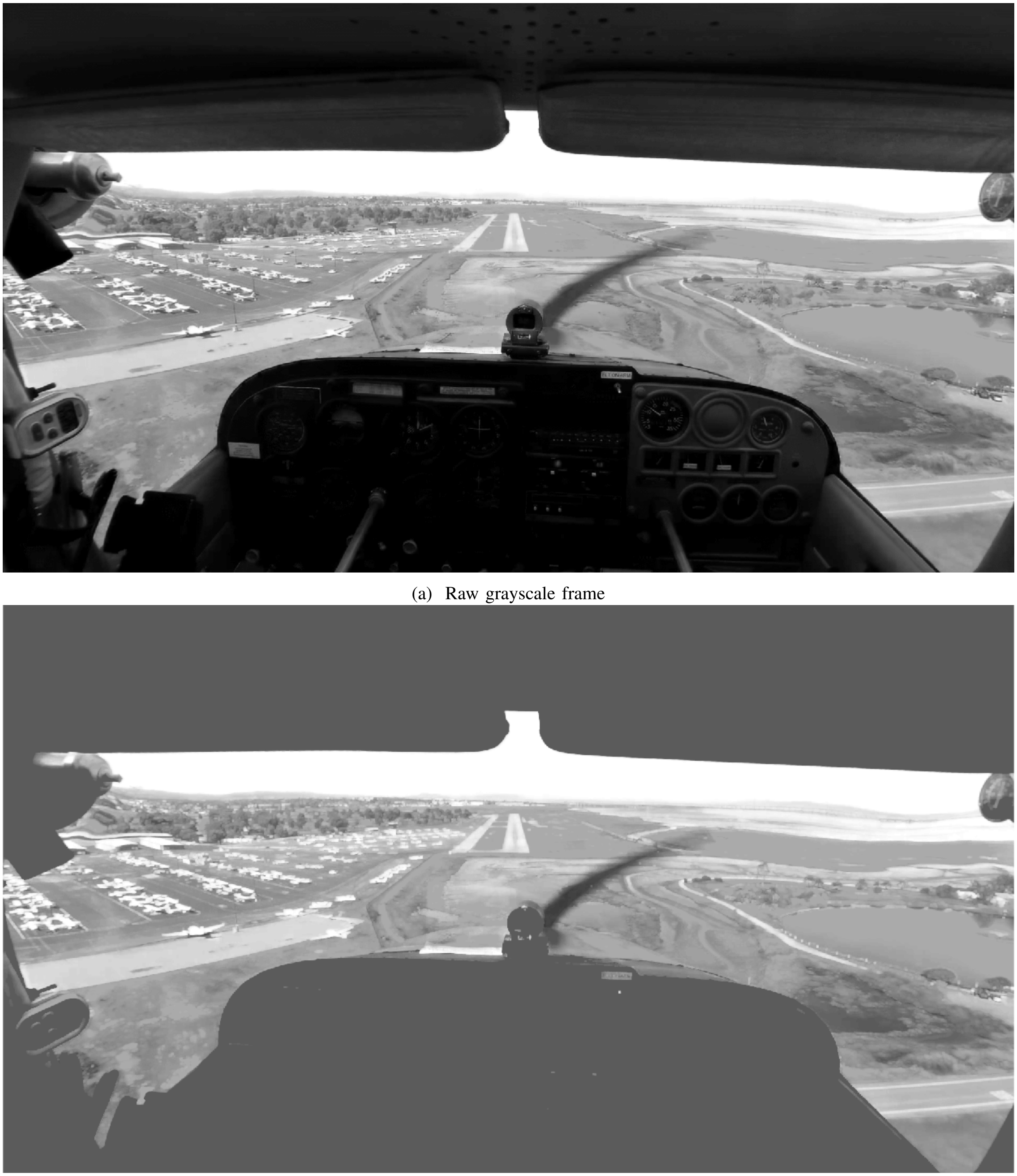
In this section, we describe our experimental setup and present examples of the different steps of the altitude extraction pipeline.

A. Data collection

A camera was installed inside of the cockpit of a Cessna 172, and multiple landings were performed. We decided to use Garmin Virb$^{\text{TM}}$ camera to collect the data. We chose this camera because it was marketed as having GPS and attitude information as part of the metadata stored with the video. Unfortunately, it turned out that during our data collection, GPS measurements were only available at a low rate (1Hz), and we did not have any GPS velocities. Additionally, the attitude data was presented as accelerometer data which was very noisy due to vibrations in the airplane. This would be easily overcome if a reasonable attitude filter using gyro rates were to be used. Due to the lack of this data, we were forced to assume level flight. We experimented with different camera settings, framerates, resolution, and camera locations. We eventually settled on wide field of view, 720p resolution, and 30 frames per second. A sample of the video frame sequence (in grayscale) is included in Fig. 3a

B. Pipeline Performance

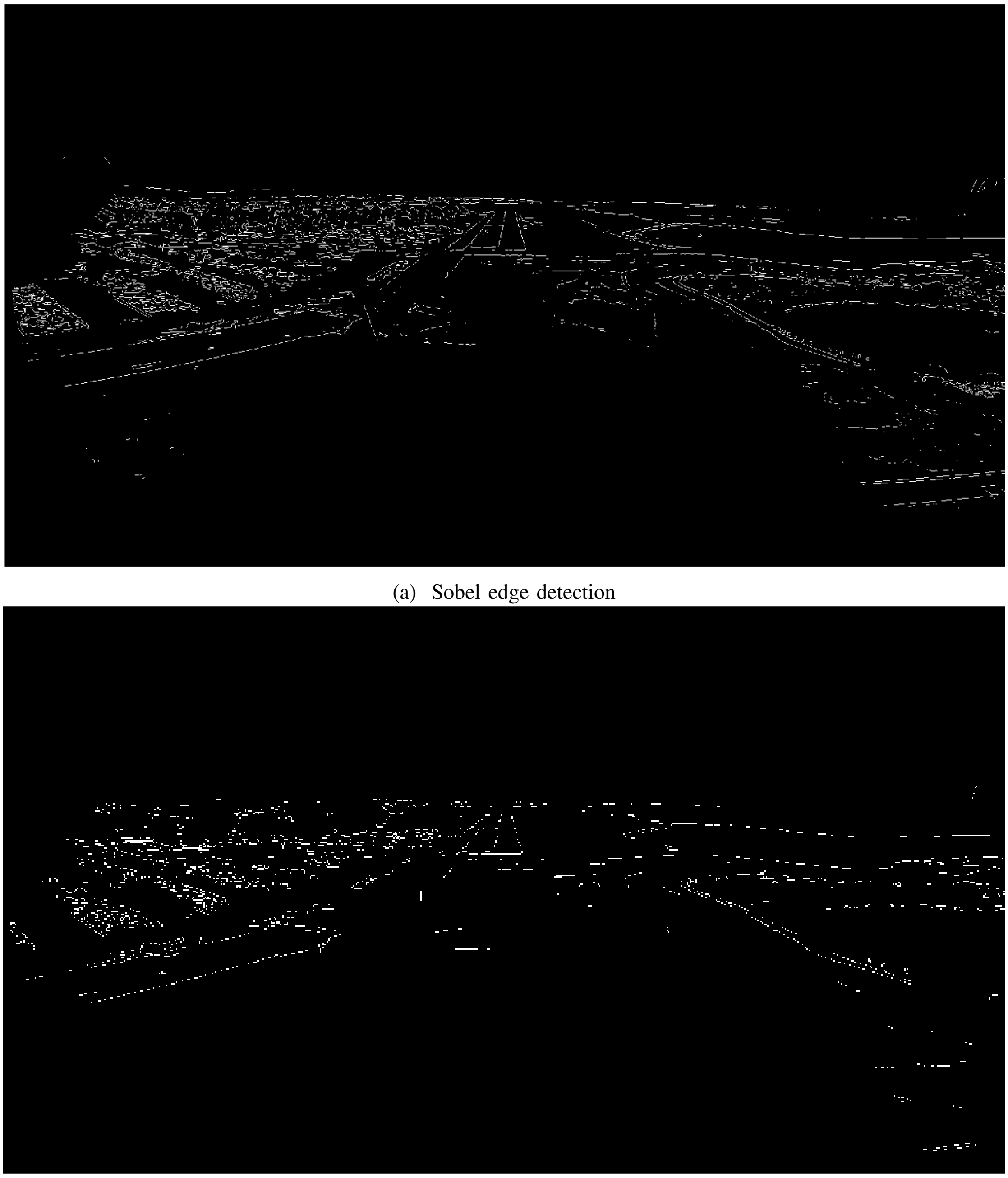
As mentioned in Section III-B, we experimented with two methods for masking the cockpit. We actually found that a combination of both methods worked best. I.e, we averaged the frames together, then applied the brightness threshold on the averaged to extract a mask which is subsequently used to remove the foreground (i.e. cockpit). The resulting masked image with dilation is shown in Fig. 3b.



(a) Raw grayscale frame

(b) Masked and Dilated frame

Fig. 3: Runway Edge Detection Pipeline: Pre-Processing



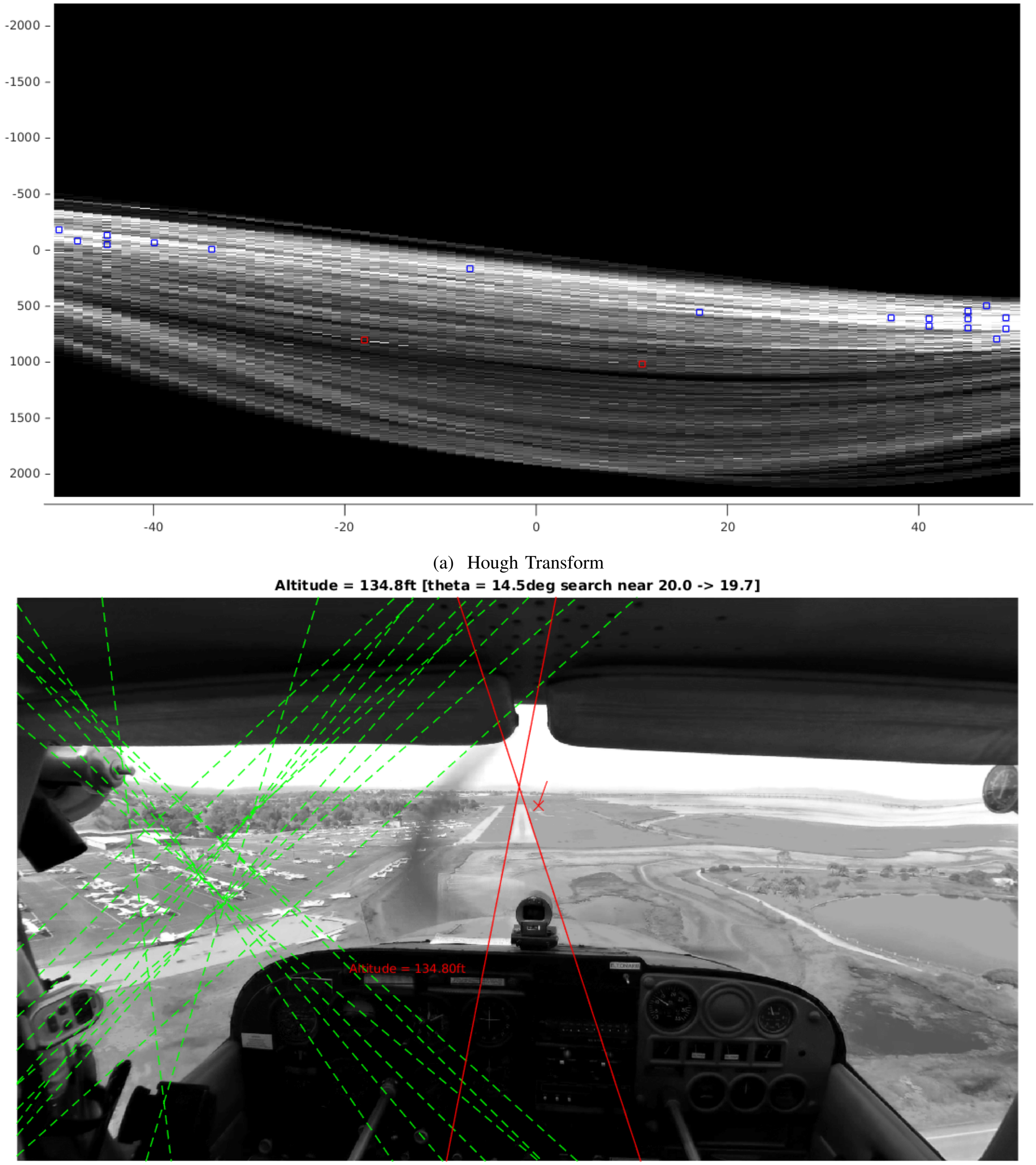
(a) Sobel edge detection

(b) Area Opening

Fig. 4: Runway Edge Detection Pipeline: Edge detection

A sample result of the edge detection is shown in Fig. 4a. Despite the area opening which removes some of the blobs as seen in Fig. 4b, many cluttered regions remain in the left portion of the image near the taxi area of the airport. We could further remove some of the blobs, but there is a trade-off to be made as a more aggressive area opening risks removing the runway edges themselves.

A sample of the Hough transform for these images is shown in Fig. 5a. The top 20 lines are all identified in the Hough space in blue, and the top scoring pair is highlighted in red. All 20 lines are projected back into the 2D image in Fig. 5b.



(a) Hough Transform

(b) Detected Runway Edges

Fig. 5: Runway Edge Detection Pipeline: Hough Transform

We notice in Fig. 5b that many of the false-positive lines go through the region with the blobs from the taxi area that were not removed by the area opening. But the top scoring pair (shown in red) correctly matches the runway edges.

The slope of these two lines are then used to compute the above ground altitude using Eq. (3). The result is overlaid in text on Fig. 5b, where we can see that the extracted altitude (134 feet) matches the value displayed on the altimeter (150 feet) reasonably well (The airport is roughly at sea-level, so the altimeter is a good proxy for AGL).

The extracted altitude for a 30 seconds long portion is shown in Fig. 6. The results look reasonable. However, there are a few outliers in the results (some worse than others). We mentioned in Section III-B that using a Kalman filter would smooth some of this noise and help reject outliers. Unfortunately, due to the problem with data collection in our first flights, combined with time constraints for this project, we were not able to collect data with other sensor values to experiment with the Kalman filtering.

Results of post-processing the video using the pipeline can be visualized at http://youtu.be/QWjFQfFFzrY

The Matlab source code for the pipeline described in this project is available at http://github.com/zoohair/visioFlare

V. Conclusions

In this project, we implemented a runway edge detection pipeline using images captured from the cockpit view of an aircraft. We combined this pipeline with concepts from single view metrology to estimate the above ground altitude of a landing aircraft. Although the results were a bit noisy, the extracted altitude seemed reasonable.

We demonstrated the feasibility and soundness of the approach in this project, and we hope that this will get implemented on a phone in the future. The goal being that it can be used realtime in an aircraft to help student pilots improve their skills at the flare.

Unfortunately, due to problems gathering data and time constraints, we were not able to experiment with Kalman filtering to reduce some of the noise. However, we believe that outlier rejection should be possible based on innovation thresholding. Another way the Kalman filtering can be used in this framework is by tightly coupling it with the line-pair scoring part of the pipeline. Indeed, rather than scoring each pair based on the heuristic metrics we described in Section III-B, we could treat each pair of lines as a potential measurement for the stochastic process. The innovation from each measurement can then be compared to the estimated covariance of the altitude, and measurements that have too large of an innovation can be rejected.

References

[1] *Aviation Safety - Landing Accidents and Runway*. http://www.aviationsafetymagazine.com/airplane/landing-accidents-and-runway.html. 2012.

[2] FAA. *Maneuvering: Approach and Landing*. https://www.faasafety.gov/gslac/ALC/course_content.aspx?cID=34&sID=164&preview=true.

[3] *FlareAssist Ultrasonic Altimeter*. http://seareyspecialist.com/flareassist.html. (Visited 02/02/2015).

[4] Weiwei Kong et al. "Vision-based autonomous landing system for unmanned aerial vehicle: A survey". In: *Multisensor Fusion and Information Integration for Intelligent Systems (MFI), 2014 International Conference on*. IEEE. 2014, pp. 1–8.

[5] Tianshu Liu et al. "Photogrammetric techniques for aerospace applications". In: *Progress in Aerospace Sciences* 54 (2012), pp. 1–58.

[6] Max Mulder et al. "The effects of pictorial detail on the timing of the landing flare: Results of a visual simulation experiment". In: *The International Journal of Aviation Psychology* 10.3 (2000), pp. 291–315.

[7] VPS Naidu and Delphina Garlin. "Detection of Airport Runway Edges using Line Detection Techniques". In: (2011).

[8] Stephen Palmisano et al. *Investigation of visual flight cues for timing the initiation of the landing flare*. Tech. rep. Technical Report, 2005.

[9] SKYbrary Aviation Safety. *Radio Altimeter*. http://www.skybrary.aero/index.php/Radio_Altimeter. (Visited 02/02/2015).

[10] S Sasa et al. "Position and attitude estimation using image processing of runway". In: *AIAA: Aerospace Sciences Meeting and Exhibit, 38th, Reno, NV*. Vol. 66. 2000.

[11] Phichet Trisiripisal et al. "Stereo analysis for vision-based guidance and control of aircraft landing". In: *44th AIAA Aerospace Science Meeting and Exhibit*. 2006, pp. 1–17.

[12] Matjaž Vidmar. "A landing radio altimeter for small aircraft". In: *Power Electronics and Motion Control Conference, 2006. EPE-PEMC 2006. 12th International*. IEEE. 2006, pp. 2020–2024.

[13] Wikipedia. *The Development of Autoland*. http://en.wikipedia.org/wiki/Autoland. (Visited 02/02/2015).

[14] Wikipedia. *Wide Area Augmentation System*. http://en.wikipedia.org/wiki/Wide_Area_Augmentation_System. (Visited 02/02/2015).

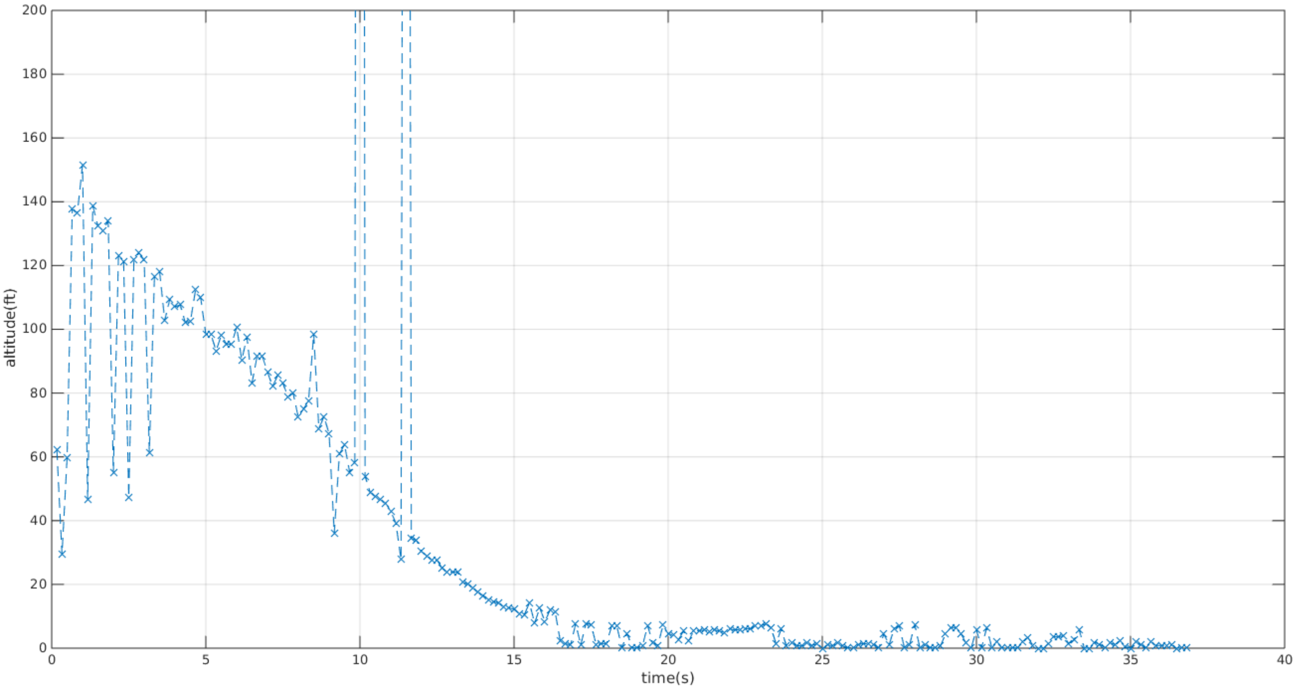


Fig. 6: Extracted Altitude from Video Sequence