Presented at the DLSU Research Congress 2014
De La Salle University, Manila, Philippines
March 6-8, 2014

Instrumentation and Sensor Fusion Implementation for Automated Takeoff and Landing Sequence of a Single-Rotor RC Helicopter

Patrick Atienza, Hans Lim, Patrick Lo, Jan Mercadejas, and Clement Ong*
Computer Technology Department, College of Computer Studies, De La Salle University
**clem.ong@delasalle.ph*

Abstract: The Autonomous Takeoff and Landing Sequences of an RC Helicopter (ATLaS) aims to develop a control system that can automate the takeoff and landing sequence, as well as maintain a low-altitude hover, of a Radio-controlled (RC) helicopter. These three areas of helicopter flight commonly involve vehicle operation in so-called "Ground-effect" region. The ground-effect region, which occurs up to an altitude of approximately the rotor diameter, is caused by rotor tip vortices being unable to form properly due to interference, and the rotor downwash being interrupted by the ground. This results in extra lift and speed, which, whilst being possibly seen as positive attributes, requires control system parameter retuning. Without such compensation, the increased control sensitivity results in helicopter instability. It is therefore important that sufficiently accurate estimates of vehicle state (primarily: attitude and altitude) be available to the control system.

In this paper a rudimentary fusing of Inertial Measurement Unit (IMU) and ultrasonic ranging for proximity (altitude) measurement is discussed and presented. These sensors are incorporated intoa TREX600-ESC scale RC helicopter (rotor diameter 1.6m). Raw performance of the Ardupilot's (an ATMEGA-based, low-cost autopilot) IMU is characterized, showing significant noise is present in the accelerometer output. The ultrasonic sensors on the other hand are shown to be accurate to within 1cm up to a distance of 2m. Results show that a simple fusion algorithm consisting of correcting the initial accelerometer integral based on measured altitude keeps errors from propagating and within 30cm of actual height at all times. Although not particularly accurate, the approach is merited by very low computation requirements (i.e. versus Kalman fusion), and should be sufficient to determine when the helicopter control system should switch into and out of ground-effect operating parameters.

Key Words: IMU; Proximity Sensor; Sensor Fusion; Unmanned Aerial Vehicle

1. INTRODUCTION

1.1 Overview

Landing and takeoff for single-rotor remote controlled helicopters have always been difficult to execute even for experienced helicopter pilots, whether it is on full scale helicopters or on miniature RC versions. Despite being a miniature version of an actual helicopter, a single-rotor RC helicopter has flight characteristics similar to a full-scale real helicopter. It takes a significant amount of training and experience to be able to competently fly the more complicated, fully featured models which more closely resemble full scale helicopters in terms of functionality and control. Thus, automating the takeoff and landing process can be considered a challenging problem and an important start to any significant developments towards unmanned aerial vehicles (UAVs). This is in part due to the ground effect which can potentially disrupt flight stability. This is because hovering or approaching ground effect range can reduce the induced velocity of the helicopter's rotor blades which tends to cause additional lift and control sensitivity to the helicopter.

In order to compensate for this, various parameters have to be monitored and accounted for when designing a control system for a helicopter, most of which involve conditions present in the environment as well as the individual movements of the helicopter itself. Because the helicopter is constantly subjected to inertia and other external forces such as ground effect, a control system will require multiple sensors to monitor all these factors. Based on the values obtained from these different sensors, an algorithm can be used to properly control the actuators of the helicopter. However, the output of a sensor must still be conditioned in order for the values to be accurate and stable. Maintaining the stability of the sensor readings will ensure that the actuators will not oscillate uncontrollably and to prevent the helicopter from failing.

2. METHODOLOGY

2.1 ATLaS Sensor System

The Automated Takeoff and Landing System (ATLaS) aims to develop an automated takeoff and landing sequence for the helicopter using multiple integrated sensors in order to compensate for the ground effect, ensuring smooth takeoff and landing as well as stable low-level flight. In order to successfully perform this, the system must be able to approximate its own flight state using a discrete group of sensors. Approximating the helicopter as a 6-degree-of-freedom rigid body, the state information necessary to allow it to accurately map itself includes linear acceleration, velocity and position of the helicopter as well as its rotational acceleration, velocities and position. However, above all of these is the helicopter's vertical position since it is where the system will depend on the most in order to perform accurate ground effect compensation. This is because the primary factor that is in effect during ground effect is the helicopter's distance from the ground, which modifies the airflow underneath the rotor tip vortices of the helicopter which then contributes to the increased sensitivity and "bubble" effect that many pilots experience when operating RC helicopters. (Baluta, S., 2009).

Currently, the ATLaS system is implemented onboard an ArduPilot Mega 2.5 (Elder, C., 2011), which is an Arduino based microcontroller that acts as both the sensor platform and control system. Communication with this microcontroller is made possible through a 3DR radio that is connected to a mobile computer functioning as the ground control module. Though it was made primarily for multi-rotor UAVs instead of single rotor crafts, it is open source which creates a "friendly" software environment with an active community to better facilitate in its development. It is a fairly powerful processing unit that is more than enough for most processing requirements with plenty of additional ports for expansion.

2.1.1. Accelerometer

An important factor which must be monitored in the takeoff and landing of an RC helicopter is the amount of inertia the helicopter is subjected to during each stage in its operation so one can naturally compensate for any abrupt changes in the readings of the system in order to make necessary adjustments during the landing and takeoff process. The accelerometer is an instrument for measuring the change in an objects position due to the influence of a static or dynamic vector.

2.1.2. Ultrasound Proximity Sensor

Since traditional inertial measurements are insufficient in measuring the helicopter's altitude relative to the ground, a more accurate sensor must be utilized. This is because accelerometers and gyroscopes are subject to long-term integral drift. GPS on the other hand has an absolute position (longitude/latitude) accuracy of about 10m, and less when measuring vertical position. For an RC, this level of certainty is not adequate. Of the several possible alternative sensors, ultrasonics-based time-of-flight seems a logical choice given the low cost, fair accuracy, small size and low weight of such sensors. Ultrasonic transducers are unaffected by environmental conditions, do not require integrations steps and hereby avoiding accumulations of errors, and most importantly, are able to attain sufficiently accurate readings at low altitudes wherein the ground effect region is situated (Mohammad, T., 2009).

2.1.3 Control System

The system uses an external Arduino board to convert the raw PWM data of the proximity sensor into a byte data that is outputted via analog pin of the APM. Every byte of data composes of 9 data bits and 1 parity bit. The sensor data is assigned to the A0 pin of the APM, and the clock signal is assigned to the A1 pin of the same system. Each data bit and clock signal bit updates every ~40 ms, however each byte data updates every ~70 ms to avoid data shift errors. The APM board reads the sensor data at a refresh rate of ~10 ms to make room for other user-defined functions like the pitch control. The whole data byte, including the delay, is processed at ~470 ms which is approximately half a second. The clock signal reflects the delay of the data byte thus it also has a ~70 ms delay. The clock signal is also shifted by ~20 ms to right by the time when a data bit is outputted and this is because every data bit is read whenever the clock toggles.

2.2 Sensor Fusion Algorithm

The Sensor fusion is the process of combining one or more sensor outputs in order to form a more accurate representation of the object of interest. It is usually done to minimize the errors inherent in different kinds of sensors by combining their values with respective weights in attempt to minimize the square errors of the resulting output. For sensors such as accelerometers, which accumulate errors over time due to the integration process involved in converting acceleration to velocity or displacement, they would need a way of "resetting" their values in order to purge the accumulated noise. For the proximity sensor implementation on the other hand, despite providing accurate readings that are not as susceptible to noise as accelerometers, have limited range (no echo when distance is too far, not enough time when distance is too near) and require differentiation to calculate speed and a second differentiation to determine acceleration. Sensor fusion offers a means to allow both sensors to work together, eliminating each other's weaknesses.

Dead reckoning is a process used in order to predict the position of an object. It is derived by using an object's most recent known position with projects based on the speed of the object. However, this technique is only an approximation based on the data used. Deriving position from the accelerometer of the Inertial Measurement Unit (IMU) of the helicopter is an integrative. As a result, any errors present in any of the readings accumulate over time, resulting in potentially inaccurate readings as time increases. However, through sensor fusion, the pitfalls associated with dead reckoning can be largely compensated for through the use of additional sensors. This is done by regularly purging the system of the accumulated errors brought about by the integration steps performed during dead reckoning.

The sensor fusion algorithm first involves initializing the pre-takeoff/landing state to the proximity sensor readings, then using dead reckoning with the IMU to fill in the gaps while waiting for the next proximity sensor update. Since proximity sensor data is deemed much more reliable than IMU data, its measured altitude and computed velocity take priority from the previous sample.

$$\mathbf{V_{PROX}} = (\mathbf{D_{CUR}} - \mathbf{D_{PREV}}) / (\mathbf{t_{prox}}) \quad \text{(Eq. 1)}$$

Where:

V_{PROX}= Velocity based on the Proximity Sensor
D_{CUR} = Current Altitude Reading (Prox Sensor)
D_{PREV}= Previous Altitude Reading (Prox Sensor)
t_{PROX} = Proximity Sensor Update Time (Rate)

Pseudo-code:

(1) When new data acquired from proximity Sensor
$D_{CUR} = < \text{Get Data From proximity Sensor} >$
$t_{PROX} = < \text{Time it takes the prox sensor to update} >$
(2) Compute velocity using Equation 1:

$$\mathbf{V_{PROX}} = (\mathbf{D_{CUR}} - \mathbf{D_{PREV}}) / (\mathbf{t_{prox}})$$

Velocity is computed by subtracting the previous proximity sensor reading from the most recent proximity sensor reading as shown in Equation 1, since the result is the difference in distance in the time it takes for an update to occur. This provides the average velocity measured based on the last proximity sensor update. The difference between this and the average velocity from accelerometer readings is then added to the cumulative accelerometer readings to obtain the instantaneous velocity at any point in time.

1. Initialize altitude to 0 and initial velocity to proximity sensor's current reading
2. Wait for the next proximity sensor update. Use IMU information to determine craft vertical position and velocity by integrating accelerometer reading
3. Once proximity sensor updates, correct IMU estimates of velocity and position though computing the instantaneous velocity and altitude as shown in equation 2 and 3.

This way, the accumulated errors generated by the integration steps on IMU data are reduced to the amount of errors that can be generated in between proximity sensor updates, which are well within tolerable margins. This is because the noise and errors in the IMU data acquisition are cancelled once the proximity sensor data arrives. Through this, both sensors end up complementing one another, with the IMU being able to provide fast readings that degrade over time while the proximity sensor provides accurate data that do not degrade.

2.2.1. Instantaneous Velocity

$$\mathbf{V_{INST}} = \boldsymbol{\alpha}\mathbf{t} + \mathbf{V_{INST}} \qquad \text{(Eq. 2)}$$

Where:

V_{INST}= Instantaneous Velocity

V_{PROX}= Velocity based on the Proximity Sensor

α = Accelerometer Z-Axis Reading

t= Accelerometer Update Time (Rate) = 1/50Hz

Pseudo-code:

(1) When new data acquired from Proximity Sensor

$V_{AVE} = V_{AVE}$ / Number of Samples

$V_{INST} = V_{INST} + (V_{PROX} - V_{AVE})$

(2) Procedure

α = <Get Accelerometer Z-Axis Data>

$V_{INST} += \alpha t$

$V_{AVE} += V_{INST}$

(3) Repeat (2) while waiting for proximity sensor data to update

2.2.2. Instantaneous Altitude

$$\mathbf{D_{INST}} = \frac{\boldsymbol{\alpha}\mathbf{t^2}}{\mathbf{2}} + \mathbf{V_{INST}} + \mathbf{D_{CUR}} \qquad \text{(Eq. 3)}$$

Where:

D_{INST}= Instantaneous Altitude
D_{CUR} = Current Altitude Reading (Prox Sensor)
V_{PROX}= Velocity based on the Proximity Sensor
α = Accelerometer Z-Axis Reading
t= Accelerometer Update Time (Rate) = 1/50Hz

Pseudo-code:

(1) $D_{INST} = D_{CUR}$When new data acquired from proximity sensor

(2) Procedure

α = <Get Accelerometer Z-Axis Data>

$$D_{INST} += \frac{\alpha t^2}{2} + V_{PROX} * t_{PROX}$$

Where $t_{PROX} =$ Prox Sensor Update Time

(3) Repeat (2) while waiting for proximity sensor data to update

3. RESULTS AND DISCUSSION

Multiple tests were conducted in order to properly assess the capabilities of the sensors as well as to identify how the sensor data should be fused. The first test involved static tests of the accelerometer. The second test is a controlled path test which was used to test the sensor fusion algorithm in a controlled environment. The device was connected directly to the APM board while the helicopter was manually moved along a fixed vertical path. Lastly, the system was tested in an actual flight in a large outdoor area, which involved multiple instances of takeoff and landing followed by some basic flight.

An initial test was conducted wherein the helicopter was placed stationary (Steady) to the ground to verify the readings obtained through the accelerometer of the APM board. The 3 axes being tested are the vertical (yaw), lateral (pitch) and longitudinal (roll) axis respectively for the accelerometer. The raw data was then obtained by measuring the effect that gravity plays on the helicopter, while the logs were collected and plotted.

A. Static Test – Steady

STD (Standard deviation): 0.0151 m/s2
MEAN (Average): -9.9081 m/s2

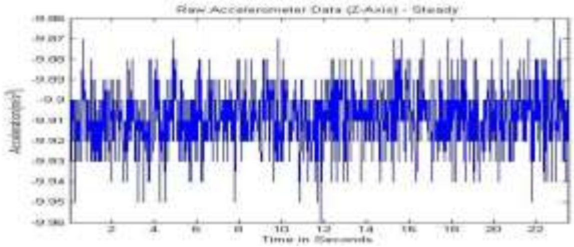

Fig. 1. Static – Steady (Accelerometer)

B. Static Test – Facing Downward

STD (Standard deviation): 0.0817 m/s2
MEAN (Average): -0.3528 m/s2

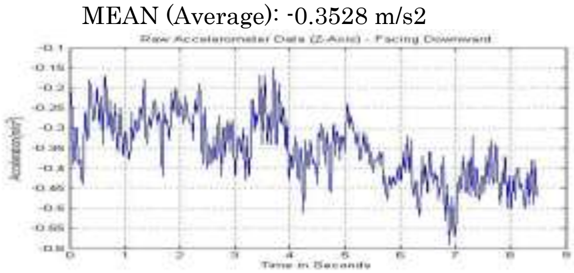

Fig. 2. Static – Facing Downward (Accelerometer)

C. Static Test - Onside

STD (Standard deviation): 0.0371 m/s2
MEAN (Average): -0.5513 m/s2

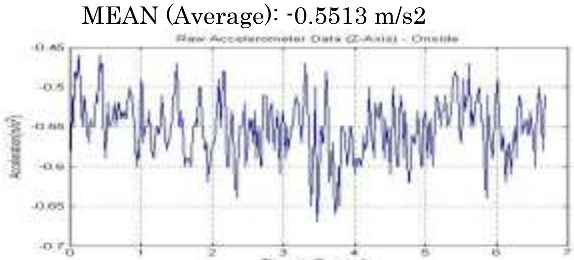

Fig. 3. Static – Onside (Accelerometer)

Accelerometer data shows that the Z axis of the accelerometer has a relatively low standard deviation in the static test, which is important because it means that the maximum amount of errors that it will accumulate in a fixed time interval are kept smaller. Though it is important to note that the mean value is not exactly 0 when the helicopter was in a sideward or downward position, nor was it really 9.8m/s^2 either which means that there is still some noise that is acting on the system. It should be noted that for the downward and sideward tests the helicopter was hand-held, thus perfect orientation was not guaranteed (and subject to movements as well).

However, despite the low standard deviation, the accumulated errors will still grow to exponential levels after two integration steps are performed to obtain the altitude. Since the amount of errors inevitably accumulate over time, sensor fusion then has to be applied to the proximity sensor in order to compensate for the errors as the accumulated errors are reset to 0 upon each proximity sensor update. And since the proximity sensor updates once per second, the maximum amount of error that can accumulate is kept at a manageable 15cm.

Initial tests on the proximity sensor have confirmed the accuracy of the sensor with errors not exceeding 1cm which results in a relative accuracy of over 99.5 % (assuming a 2m maximum). One key weakness however, is the time taken for the proximity sensor to obtain a reading, taking almost one second to update. This amount of time is significant for a helicopter in flight, since RC helicopters can achieve 60km/h speeds.

D. Proximity Sensor Tests - Indoor

In this test, the device was mounted on the helicopter and tested by manually moving the helicopter above the ground at fixed values using a meter stick. The following figures shows the proximity sensor readings, accelerometer readings (after removing the gravity vector) and the measured altitude and velocity after performing sensor fusion. A two-step process involving differentiation and integration are performed on the accelerometer to attempt to remove the gravity vector from the sensor readings.

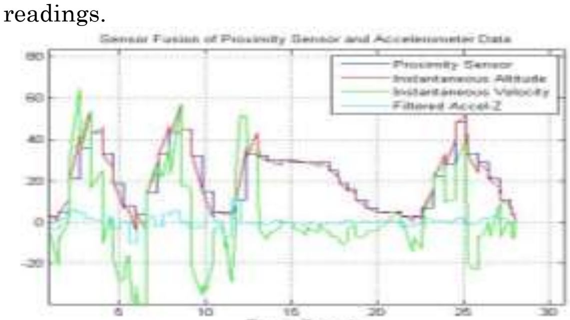

Fig. 4. Sensor Fusion Test (Controlled Environment)

The accelerometer's readings show a significant amount of noise, resulting in fairly erratic velocity readings. Sensor fusion results in a few overshoots in obtaining altitude information, with the error not exceeding 15cm at ground level or taking off. It is worth noting that even when there is no motion detected on the proximity sensor, the accelerometer still registers some motion, which can be attributed to residual elements from the gravity vector which may not have been completely eliminated.

E. Proximity Sensor Tests - Outdoor

The final test involved manual takeoff and landing operations by an experienced pilot under fair weather conditions in an open field. This test was aimed at assessing whether or not the algorithm devised is capable of performing properly in actual flight conditions, which often have much more potential for noise and errors than controlled test environments.

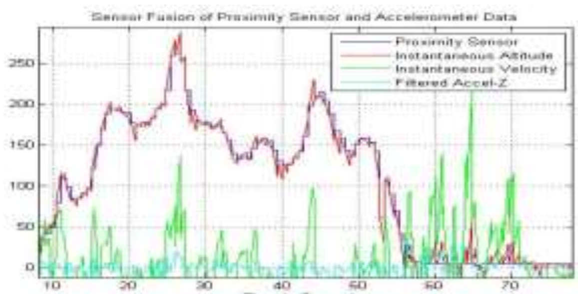

Fig. 5. Sensor Fusion Test (Outdoor) – Test 1

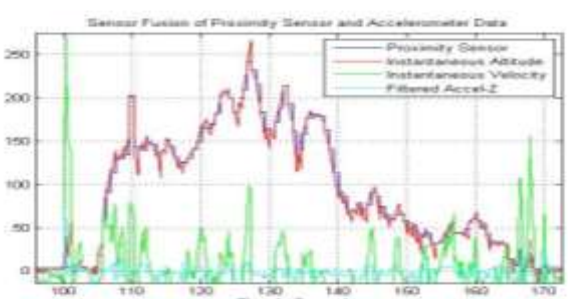

Fig. 6. Sensor Fusion Test (Outdoor) – Test 2

The actual flight test is noticeably noisier than the controlled path test, with measured errors reaching a maximum of 30cm. This can be attributed to a number of factors such as issues with eliminating the gravity vector which occurs due to changes in the gyroscope orientation during flight which can distort the vector, creating more residual elements when performing a derivative-integral step. Also, the vibrations of the helicopter can also create some additional noise, which can increase the potential errors accumulated during the integration steps. It is worth noting how the errors seem to be greatest when the altitude is supposedly unchanged, while staying relatively accurate when the system is in motion. At higher velocities, the relative accuracy of the system is much higher, however, at lower velocities, even if the helicopter is kept at a constant elevation, the error is much more apparent. This phenomenon can be attributed to the effect of the gravity vector because the measured acceleration becomes small, buried in the residual elements of the gravity vector.

Despite this, the altitude measurements are relatively consistent with an accuracy of 88%. Though the developed sensor fusion algorithm is fairly promising, it still holds a few key weaknesses that have to be addressed in order to ensure that the helicopter is able to adequately assess its current state.

4. CONCLUSIONS

The sensor fusion algorithm combines proximity sensor readings with accelerometer data from the IMU of the APM board. It is able to minimize drift errors to a few centimeters in fixed locations, though the amount of errors increases noticeably during actual flight, due to rotor vibration and craft torsions reaching the IMU. The proximity sensor updates significantly help cancel out the accumulated errors caused by the two integration steps performed on the accelerometer data, there is still a noticeable amount of error that accumulates quickly that needs to be better compensated. Because the gravity vector is difficult to compensate, its residual effects are present whenever the helicopter is in motion, contributing to the total errors which influences the system.

As such, the sensor system is adequate to determine when the craft is close enough for ground effects to be influential, allowing the control system to switch over to alternate parameters to better stabilize the helicopter flight.

5. REFERENCES

Abiera, G., Lee, J., Mandac, B., & Que, S. (2013). Helicopter Stability Regulation Module Using an Embedded System. De La Salle University - College of Computer Studies, Manila.

Baluta, S. (2009). A Guide to using IMU (Accelerometer and Gyroscope Devices) in Embedded Applications. Retrieved from http://www.starlino.com/imu_guide.html

Elder, C. (2013). The ArduPilot Mega (APM 2.5) Autopilot Introduction. 3DRRobotics. Retrieved from https://code.google.com/p/ardupilot-mega/wiki/Introduction.

Mohammad, T. (2009). Using Ultrasonic and Infrared Sensors for Distance Measurement. Retrieved from waset.org/journals/waset/v27/v27-51.pdf.

RCHelisite. (2008). How Do RC Helicopters Work? Performance Marketing. Retrieved from http://www.rchelisite.com/how_do_rc_helicopters_work.php.