Proceedings of ASME 2016 Dynamic Systems and Control Conference
DSCC2016
October 12-14, 2016, Minneapolis, USA

DSCC2016-9842

ON AERIAL INDOOR POSITION CONTROL AND SYSTEM INTEGRATION FOR QUADCOPTERS USING LIDARS AND INERTIAL MEASUREMENT UNITS

Nathan Zimmerman, Kellen Carey, and Cristinel Ababei
Department of Electrical and Computer Engineering
Marquette University
Milwaukee, Wisconsin 53233, USA
Email: {nathan.zimmerman,kellen.carey,cristinel.ababei}@marquette.edu

ABSTRACT

The main contribution of this paper is to introduce a computationally efficient iterative closest line (ICL) algorithm for determining indoor position drift of a quadcopter using minimal lidar data. In addition, we present the system-level design and implementation of a new quadcopter both as hardware and flight control algorithms. Such a platform allows us to develop and experiment new control and system optimization techniques. As an example, we discuss how the proposed ICL algorithm is used for position hold and control purposes by plugging it into the low level implementation of the flight control algorithm of the quadcopter. For testing and validation we use simulations with real world data. As part of the system-level design aspects, we present an investigation of the quadcopter power consumption. We are interested in power consumption because it is the major factor that determines the flight time of a typical quadcopter. We believe that this work is a contribution toward achieving better quadcopter design and control for indoor autonomous navigation.

1 Introduction

Recently, aerial multi-rotor or drone systems have been applied in numerous applications such as surveillance, aerial photography, and other general consumer use. Owing to their affordability and versatility, these applications continue to increase and much research has been done regarding their future applications. For example, Amazon Prime Air, an autonomous outdoor delivery system, has received widespread media coverage. However, while outdoor applications for multi-rotor systems are numerous, indoor autonomous use has been less discussed in the available literature. That is in part because indoor autonomous control is more difficult than outdoors where access to the global positioning system (GPS) is reliable. Also, minor positional drift outdoors is generally of less consequence than indoors where space is confined. Microelectromechanical systems (MEMS) based inertial measurement units (IMUs) alone have limited or no capacity to sense and prevent indoor positional drift. Consequently, a variety of solutions have been proposed to address this issue. A common such solution is to use infrared (IR) beacons and a network of cameras to tag the drone and to determine its absolute position. Another popular solution is to use an optical flow sensor attached to the bottom of the drone to reduce positional drift. However, these solutions generally are limited in their ability to maintain absolute control of position.

An emerging approach for indoor autonomous position control is the use of a lidar sensor. A lidar is is a scanning 2D laser range finder, which can provide accurate position information. Lidars have been increasingly utilized in various applications and their cost continues to decrease. An example application is the Neato XV, which is an autonomous vacuum cleaner robot [1]. This robot uses a lidar sensor to map and navigate the environment and employs the simultaneous mapping and localization (SLAM) algorithm. However, while SLAM performs well for ground based vehicles, it is more difficult to implement for aerial multi-rotor systems due to positional drift. Consequently, indoors SLAM for multi-rotor systems is an active area of research. Here, we see mostly studies that use lidars which cost thousands of dollars and utilize computationally expensive algo-

rithms that require high performance processors and GB of RAM memory.

To address these issues, in this paper, we focus on using a low cost lidar for quadcopter position control and develop practical algorithms that can be run on sub GHz ARM processors with minimal RAM memory. Specifically, our main contributions include: 1) We introduce a computationally efficient algorithm for determining position drift with minimal lidar data, 2) We validate this algorithm with data from an affordable lidar sensor, 3) We integrate the proposed algorithm within the main flight control algorithm of our in-house built quadcopter, and 4) We present also a discussion of energy consumption in drone systems, which has not been done in the prior available literature.

2 Related Work

Drones, particularly quadcopters, have been studied by academic, commercial, and hobbyist communities. Quadcopters are the most popular due to their very low moment of inertia and six degrees of freedom, which results in better stability and agility of the quadcopter. A strong and active online community of professionals and hobbyists alike are contributing to the body of knowledge regarding the hardware and software development of drones.

2.1 Related Work on Drone Modeling and Development

Looking at the available literature, we find previous studies that can be classified into several different types of contributions: studies that investigate different system modeling and control techniques, trajectory planning algorithms, autonomous designs, and system hardware development using off-the-shelf or in-house components.

As examples in the first category, various control techniques have been studied, including PID control [2,3], backstepping [4], nonlinear H_∞ [5], and Kalman filtering [6].

Prior works in the second category deal with trajectory planning, which is very challenging. While most discussions assume that the trajectory exists, the question of the feasibility of a given trajectory has been studied too. Planning can be solved in two steps. In the first step, trajectories containing no time information are calculated from a class of motion primitives. Then, the trajectory is parametrized in time by choosing the trajectory speed such that dynamic feasibility constraints are enforced. An algorithm that allows the calculation of flight trajectories by combining the above two steps for quadrocopters is presented in [7].

In the case of autonomous flight, processing and decision making has to be done on board. Operating at the optimal tradeoff between flight performance, sensors, and computing resources has its own challenges. Steps towards addressing these challenges are studied in [8] within the context of indoor exploration inside a house of known shape and dimensions in order to detect objects and be able to return outside.

The online community has been very active and numerous online articles report drone designs developed on a large variety of hardware platforms. In addition, we see an increasing number of commercial offerings. The survey in [9] presents a summary of publicly available open-source projects on quadrotor unmanned aerial vehicles, including Arducopter, Openpilot, Paparazzi, and others. Because some commercial offerings such as the AR.Drone provide application programming interfaces (APIs), they can be used for educational and research purposes with focus on position stabilization, object following, and autonomous navigation [10].

Further details on the modeling and control of quadcopters can be found in the excellent tutorial in [11]. Several key insights discussed in this tutorial include: 1) blade flapping and induced drag are fundamental effects that are of high importance in understanding the natural stability of quadrotors and how state observers operate, 2) smaller quadrotors can produce faster angular accelerations while the linear acceleration is at worst unaffected by scaling, 3) position estimation can be done via a variety of means, including GPS, acoustic, laser-ranging, infrared, and vision/camera systems, and 4) usually a hierarchical control approach is used for quadrotors to control the rotor rotational speed (lowest level), vehicle attitude, and position along a trajectory.

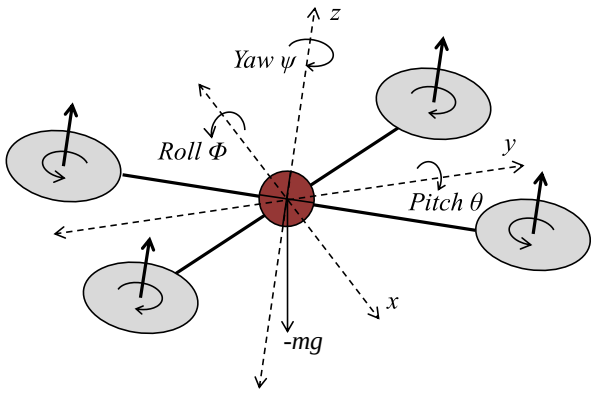
2.2 Related Work on Lidar Usage in Drone Development

The key state estimates required for the control of a quadrotor include its height, attitude, angular velocity, and linear velocity [11]. The attitude and angular velocity are the most important because they are the primary variables used in attitude control. The most basic instrumentation carried by any quadrotor is an inertial measurement unit (IMU) often augmented by some form of height measurement, either acoustic, infrared, barometric, or laser based. A typical IMU includes a three-axis rate gyro, three-axis accelerometer, and three-axis magnetometer. The accelerometers and magnetometers can be used to provide absolute attitude information on the vehicle while the rate gyroscope provides complementary angular velocity measurements. Because we place special emphasis on the usage of lidar data for position control of drones, we review here also literature related to this topic.

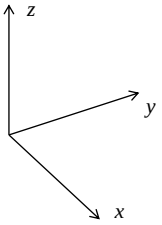
There has been a lot of work done using lidar for positioning of unmanned ground vehicles and robots [12–14]. However, there is little prior work done on using lidars directly for drone control. The only previous work that we are aware is [15]. This dissertation is a demonstration of using a lidar indoors to map in real time the environment using the iterative closest point (ICP) technique fused with IMU data. This approach is the closest to our work. In contrast, we use a less powerful embedded processor with only 64KB or RAM versus 2GB and focus primarily on position/localization control compared to simultaneous mapping and localization.

3 Background on Modeling and Control Goals

To design and develop flight control algorithms for quadcopters we need to model their dynamics. In this paper, we adopt a modeling approach and notation similar to that in [16]. This modeling approach uses two frames of operation. The first one is the so called drone *body frame* and is illustrated in Fig.1. The second frame, *inertial frame* is defined by the ground, with gravity pointing in the negative direction of z.



(a)



(b)

FIGURE 1. (a) Simplified diagram of a quadcopter to illustrate the three rotation angles in the body frame (roll, pitch, and yaw), gravitation force, and torque. (b) Inertial frame, i.e., the ground frame of reference.

The roll, pitch, and yaw rotation angles are defined in the body frame as ϕ, θ, ψ, with corresponding angular velocities $\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$. These variables are used to control the orientation of the drone. Therefore, when we want to control the orientation of the drone while flying and performing different maneuvers (e.g., move forward, left, etc.) or hovering (i.e., $z > 0$ and constant) the goal of the control algorithm can be expressed as:

$$\begin{aligned} &\phi \to \phi_r \quad \theta \to \theta_r \quad \psi \to \psi_r \\ &\dot{\phi} = 0 \quad \dot{\theta} = 0 \quad \dot{\psi} = 0 \end{aligned} \tag{1}$$

Where ϕ_r, θ_r, ψ_r are the desired or target orientation angles that the drone is controlled to attain.

The position and velocity of the drone is defined in the inertial frame (i.e., with respect to the ground frame) as x, y, z and $\dot{x}$, $\dot{y}$, $\dot{z}$. These are important in the position control. Hence, another control goal is that of position hold, which can be expressed as:

$$\dot{x} = 0 \quad \dot{y} = 0 \quad \dot{z} = 0 \tag{2}$$

The above goal is addressed in traditional approaches by optical flow sensors, which have limitations as discussed earlier. Equation 2 can be interpreted as the goal of maintaining the drone at the current location, at a constant or fixed distance (within the Cartesian system of coordinates as defined by the ground or inertial frame) from the initial flight starting point. In this paper, we attempt to achieve these two control goals by using IMU and lidar data fusion with new techniques for lidar data processing integrated with the position and stability control algorithm.

4 Computationally Efficient Aerial Lidar Data Processing as Key to Indoor Position Control

In this section, we present a computationally efficient method for processing lidar data. This method helps us to achieve the control objective described in equation 2. By using MEMS IMU sensors alone, aerial vehicles cannot maintain their absolute position and are subject to positional drift. To combat this drift, additional techniques such as GPS or sensors such as lidar or optical flow sensors can be employed. Lidar, a 2D scanning laser range finder, provides crucial distance information and is fully functional indoors. While highly useful, lidar sensors previously were too expensive for common applications. However, recently, the cost of general purpose lidars decreased from several thousand to just a few hundred USDs and this trend is expected to continue. For example, this project utilizes a lidar sensor that can be purchased for about 150 USD. This lidar is a replacement component for a popular autonomous vacuum cleaner, the XV-11.

With regards to aerial autonomous indoor navigation, SLAM algorithms have been used to autonomously map unknown environments with on-board sensors [15]. This is generally accomplished by the fusion of lidar and IMU sensor data. While developments in this area are promising, existing algorithms run on powerful processors that run at multiple GHz clock speeds and use several GB of RAM. In contrast to SLAM, the method proposed in this paper is used primarily for positional control and can be used on sub GHz processors.

In addition to high data throughput, lidar data processing algorithms are computationally expensive due to the iterative nature of existing processing algorithms. That is because there is no simple closed form solution that yields the Cartesian translation (movement) from one lidar data set to another, gathered

during two consecutive revolutions. To illustrate that, we use the example lidar data set from Fig.2.

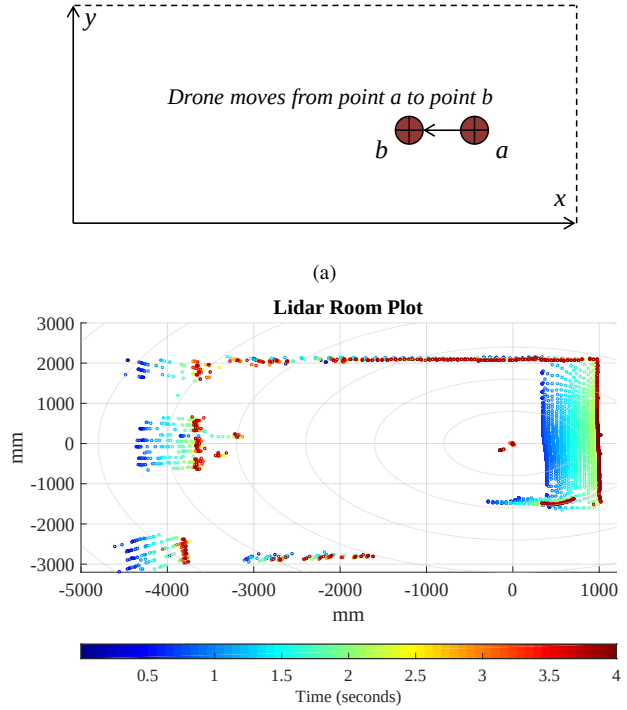


FIGURE 2. (a) Sketch of room inside which lidar experiments are done. Lidar sensor is moved in the x direction. (b) The *data clouds* collected from the lidar sensor for each revolution during the movement from point a to point b.

The lidar outputs data in polar coordinates, $(r\angle\theta)$, which we then convert to Cartesian coordinates, $x = r \cdot cos(\theta)$, $y = r \cdot sin(\theta)$. These coordinates are used to compute or estimate the displacements in the x and y directions, Δx and Δy, between multiple revolutions of the lidar sensor. In particular, we are interested in computing these displacements between *consecutive* revolutions because these displacements help us to find if the drone drifted or not within the time duration between the two lidar sensor revolutions. The challenge is that the estimation cannot be done by directly summing up the individual displacements of different points from the data set collected during any one given revolution. We illustrate this issue with the help of the example from Fig.3.a. Here, the mean of the *resulting displacement vector* has components in both x and y. However, the true displacement is only in the x direction. To identify that correctly, we need a technique that knows to reject and invalidate *false* individual point displacements such as Δy_2 and Δy_3 in Fig.3.b.

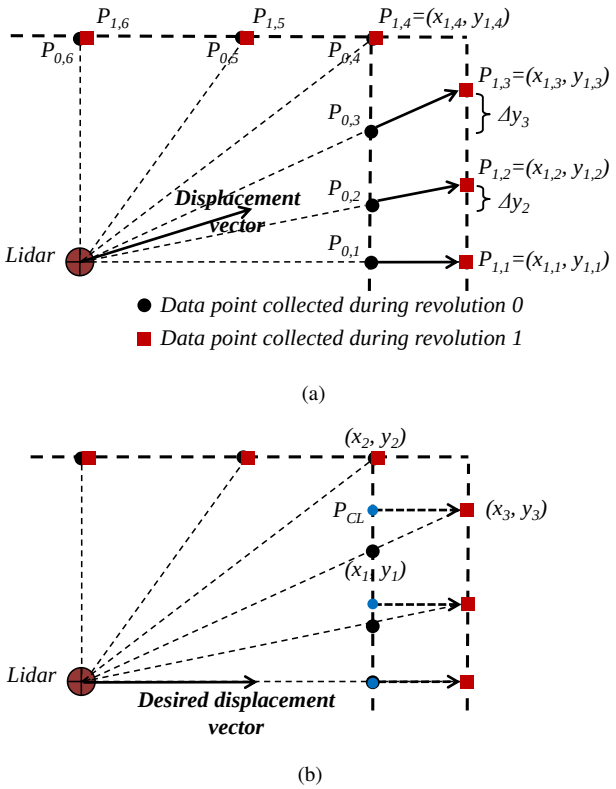


FIGURE 3. (a) Illustration of lidar reference frame issue observed during a lateral movement in the x direction of the inertial frame. In this diagram, points $P_{revolution,index}$ represent the data points provided by the lidar at different locations (indicated via *index*) during two consecutive revolutions indicated as $revolution = 0$ and $revolution = 1$. (b) The desired resulting displacement vector is obtained with the proposed technique, which uses the so called closest point, P_{CL}, defined on a line formed by two closest points collected during the previous revolution of the lidar.

In this paper, we propose an algorithm to address this issue. This algorithm is computationally efficient and targeted at ARM Cortex based processors with only 64KB of RAM memory. This algorithm is described in the next section.

4.1 Proposed Iterative Closest Line (ICL) Method

Here, we describe a simple, accurate, and computationally efficient method to estimate the overall displacements Δx and Δy between two consecutive revolutions of the lidar sensor. This method, which we refer to as the iterative closest line (ICL), can be seen as a variant of the iterative closest point (ICP) technique [17–19]. The pseudocode description of the proposed method is shown in Fig.4. Essentially, it calculates the displacements $\Delta x, \Delta y$ between two consecutive revolutions of the lidar sensor.

ICL uses a *point to line metric* in order to determine the overall resulting displacement between two point cloud sets. This point to line metric represents the *squared error* that is mini-

Algorithm: Gradient descent based ICL Method

Input: Cloud data sets of two consecutive revolutions, $k-1$, k
Output: Δx_k, Δy_k for second revolution, k
for $u \leftarrow 1$ to n **do** // n: number of iterations of algorithm
  for $i \leftarrow 1$ to 360 **do** // measurements per revolution
    $\delta x_k = \delta x_k + (x_{k,i} - x_{CL})$
    $\delta y_k = \delta y_k + (y_{k,i} - y_{CL})$
  end for
  for $i \leftarrow 1$ to 360 **do** // simultaneous update all $x_{k,i}, y_{k,i}$
    $x_{k,i} = x_{k,i} - \alpha \delta x_k$ // α is the learning rate
    $y_{k,i} = y_{k,i} - \alpha \delta y_k$
  end for
  $\Delta x_k = \Delta x_k + \delta x_k$ // accumulate displacement
  $\Delta y_k = \Delta y_k + \delta y_k$
end for

FIGURE 4. Pseudocode of the proposed ICL method. It uses a gradient descent based technique, which employs the point to line metric, to find out the displacements.

mized with a gradient descent based algorithm [20]. The point to line metric error for a point $P_{k,i}(x_{k,i}, y_{k,i})$ from the point cloud set collected during revolution k and assigned index i is denoted as $J_{k,i}$ and described by the following equation:

$$J_{k,i} = (x_{k,i} - x_{CL})^2 + (y_{k,i} - y_{CL})^2 \tag{3}$$

Point $P_{CL}(x_{CL}, y_{CL})$ represents the closest point with respect to a point to line metric. This point is calculated based on a line formed by the two closest points from the previous lidar revolution and a current data point, $P_{k,i}(x_{k,i}, y_{k,i})$. An example of such a point is shown in Fig.3.b. $J_{k,i}$ represents basically the squared distance between points $P_{k,i}$ and P_{CL}. The summation of all such distances for all points from a point cloud set gives the *squared error* that is minimized with a gradient descent based algorithm.

To calculate the coordinates (x_{CL}, y_{CL}) for each data point of a lidar revolution, first, a search must be performed to determine the two previous closest points to a new sample point. Next step is to define a line between these points using the general line form equation. This is used as opposed to the slope-intercept form in order to avoid potential divisions by 0. In this way, we find the closest point on that line to the new data point as described by the following well known equations:

$$\begin{aligned} x_{CL} &= \frac{b(bx_3 - ay_3) - ac}{a^2 + b^2} \\ y_{CL} &= \frac{a(-bx_3 + ax_3) - bc}{a^2 + b^2} \\ a &= y_2 - y_1, \quad b = x_1 - x_2, \quad c = -by_1 - ax_1 \end{aligned} \tag{4}$$

Where (x_1, y_1) and (x_2, y_2) are the two closest points on a previous lidar revolution with respect to a new point (x_3, y_3) as illustrated in Fig.3.b. The gradient descent technique will minimize the summation of all $J_{k,i}$. For that purpose, it uses the following quantities for its update laws from Fig.4:

$$\begin{aligned} \delta x_k &= \sum_{i=1}^{N} \frac{\partial J_{k,i}}{\partial x} = \sum_{i=1}^{N} (x_{k,i} - x_{CL}) \\ \delta y_k &= \sum_{i=1}^{N} \frac{\partial J_{k,i}}{\partial y} = \sum_{i=1}^{N} (y_{k,i} - y_{CL}) \end{aligned} \tag{5}$$

To estimate the overall displacement from an initial starting point (i.e., many revolutions ago), we accumulate or integrate $\Delta x, \Delta y$ across all revolutions. As an example, the traditional ICP and proposed ICL algorithms are used to estimate the overall displacements Δx and Δy using the dataset from Fig.2.b. The results of this estimation are reported in Fig.5.

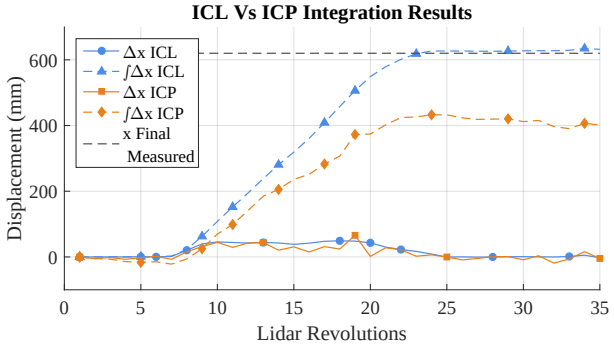


FIGURE 5. Demonstration of ICP and ICL algorithms using the dataset from Fig.2.b.

We can see that the proposed ICL algorithm performs better as its accumulated error is smaller than the error obtained with the ICP algorithm. Our method can be applied on an embedded microprocessor in real time without large matrix inversions or large memory requirements. Note that the ICP algorithm can still be successfully utilized as reported in [15], but it would require a larger number of data points collected during a single revolution of the lidar. In addition, it also needs a more expensive and powerful processor to be able to handle larger numbers of points per revolution.

4.2 A Technique for Outliers and Dynamic Object Rejection

To improve the accuracy and reliability of the proposed ICL algorithm, we implement an outlier rejection technique, which is

described in this section. Such a technique is crucial for the practical application of the proposed ICL algorithm in a real world setup. This is so because without a rejection mechanism, significant positional error can accumulate over time and mislead the position control part of the flight control algorithm. Outliers can include spurious lidar measurements as well as data points corresponding to highly dynamic objects such as humans. The proposed outlier rejection technique is robust yet computationally efficient. Its idea is to use the previous estimates $\Delta x_{k-1}, \Delta x_{k-1}$ and checks if the cost of the current point $J_{k,i}$ is significantly larger than $(\Delta x_{k-1}^2 + \Delta y_{k-1}^2)$, case in which the point is rejected. We have tested this technique with good practical results. For example, for the dataset shown in Fig.6, the total error in estimating the displacements is very small as reported in Fig.7. This figure also shows the performance of the ICL algorithm without an outlier rejection technique.

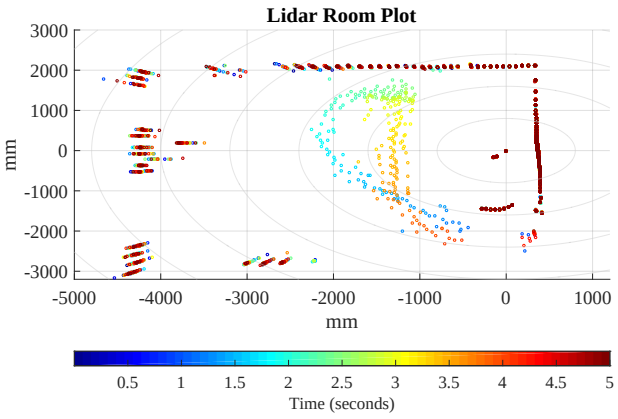


FIGURE 6. Lidar data with human movement in a stationary room.

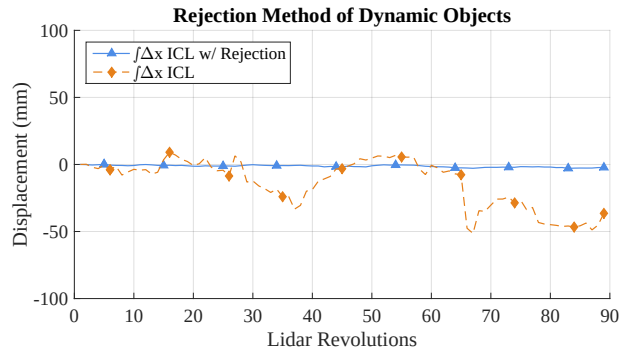


FIGURE 7. Dataset in Fig.6 with and without dynamic rejection.

4.3 Integration into the Main Flight Control Algorithm

In this section, we describe how the proposed ICL algorithm is integrated with the main flight control algorithm of the quadcopter drone. This is illustrated in Fig.8, where the lidar data provided by the lidar sensor is fed into the ICL algorithm, which then computes the displacement estimates Δx and Δy. In the absence of user input (which is received from the user's hand-held RC remote controller as $\phi_r, \theta_r, \psi_r, h_r$) the objective of the flight control algorithm is to hold the position of the quadcopter. In this case, the displacement estimates Δx and Δy are used to achieve position hold by eliminating positional drift.

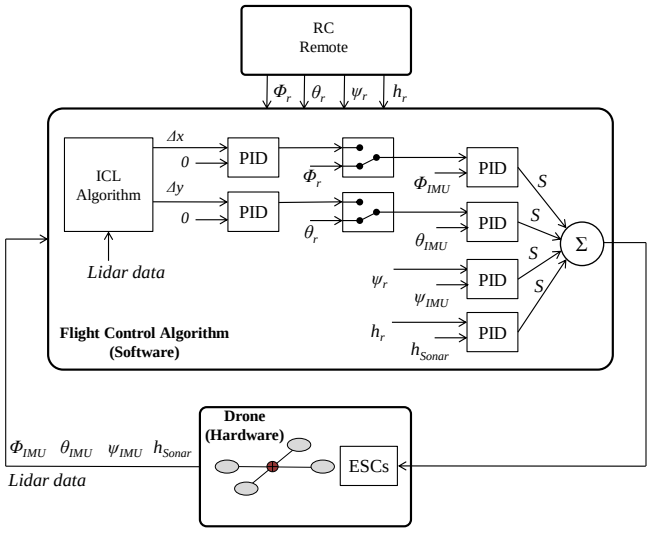


FIGURE 8. Flight control algorithm in the overall quadcopter system.

As the primary control technique, the flight control algorithm employs several proportional integral derivative (PID) controllers to control the orientation angles ϕ, θ, ψ and the height h. Basically, the role of these PID controllers is to drive ϕ, θ, ψ and h to the desired target values. The PID controllers that use the inputs Δx and Δy are responsible with the elimination of positional drift. For height control, data from a sonar distance sensor is utilized. The PID block with input h_{Sonar} controls the height of the quadcopter.

In Fig.9, S represents the vector of individual motor speeds $[S_1, S_2, S_3, S_4]$ as calculated by each of the four PID blocks. The calculation of S by each PID block is dependent on the physics based model of the quadcopter. Example derivations of the model can be found in [16] and are out of scope for this paper.

The flight control algorithm from Fig.8 was tested and validated through simulations using a physics based model for the quadcopter. To simulate positional drift, external forces were applied to the quadcopter frame. For example, the plot in Fig.9 shows how θ, ϕ are attained by the PID controllers that use Δx

and Δy as inputs. While our extensive simulations verify the correct operation of the overall flight control algorithm, we are currently working on the physical implementation and integration of the lidar and sonar sensors on the quadcopter. This is a work in progress.

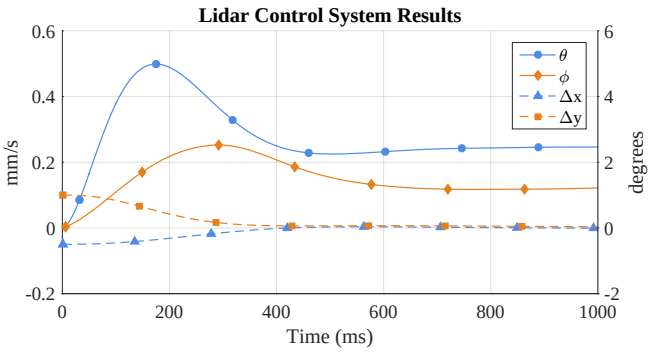


FIGURE 9. Plot showing how the flight control algorithm from Fig.8 drives Δx and Δy to 0 in order to eliminate positional drift.

5 Hardware Implementation - System Integration Aspects

In this section, we describe the implementation of our in-house built quadcopter drone. Specifically, we discuss system integration and energy consumption aspects.

5.1 Overall System Integration

In building a drone, one must decide whether to do it using off-the-shelf components or to design and build it from scratch. The issue with systems built using off-the-shelf components is that, while much easier, we have limited capability in customizing the system and in developing new flight control algorithms. On the other hand, designing a drone system from scratch offers flexibility in any design aspect: we can integrate into the system any microcontroller unit (MCU) and sensors we would like, we can customize the hardware (at the PCB level) to allow us to perform in depth performance and energy consumption studies, and we can investigate any novel idea for flight control algorithms without being limited by restrictive vendor specific application programming interfaces (APIs). In this project, we decided to implement the entire system from scratch because we were particularly interested in several design exploration directions including: using specific IMU and lidar sensors, investigating energy consumption, and experimenting with new flight control algorithm ideas.

Designing and building a quadcopter from scratch is a challenging undertaking. That is because the quadcopter system combines several hardware (HW) electronic and mechanical components as well as firmware components as shown in Fig.10.a. The four brushless motors are supplied with power and controlled by the four electronic speed controllers (ESCs). The flight controller is responsible with tasks including data fusion of data from sensors (inertial measurement unit, IMU, lidar, etc.) and the primary flight control algorithm. The remote control of the drone is done via a custom RF hand-held controller, which communicates with the radio block from Fig.10.a. The battery supplies power to all components on board of the quadcopter. As software (SW) components, the quadcopter system has two programs (shown as *Firmware* 1 and *Firmware* 2 in Fig.10.a), which implement the main flight control algorithm and the control algorithm that runs of on the four ESCs.

Each of the four ESCs controls an individual brushless direct current (BLDC) motor. The method that we use to control each of the motors is using the so called back electromotive force (back-EMF) technique [21–23]. The key to BLDC commutation is to sense the rotor position, then energize the phases that will produce the most amount of torque. The rotor travels 60 electrical degrees per commutation step. The appropriate stator current path is activated when the rotor is 120 degrees from alignment with the corresponding stator magnetic field, and then deactivated when the rotor is 60 degrees from alignment, at which time the next circuit is activated and the process repeats [22].

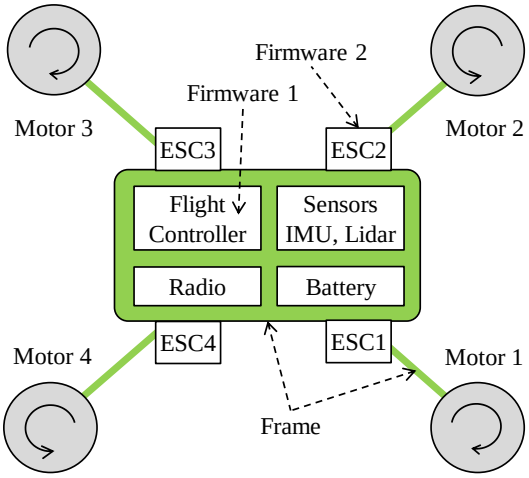
Due to space limitations we cannot provide here comprehensive datasheet specifications of the individual components that we used in building our in-house drone. Instead we created a website to document in detail both the hardware and software, which we make publicly available as open source [26]. We do however, present specifications of the lidar device that we use in our experiments in the next section.

5.2 Lidar Specifications

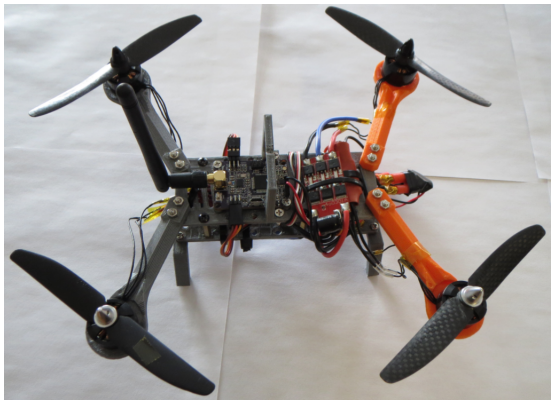
In this section, we present the specifications of the lidar device that we use, XV-11, in order to compare it with the Hokuyo UTM-30LX lidar that is used by the study in [15]. The photographs of these two lidar devices are shown in Fig.11 while the technical specifications are presented in Table 1. We note that the XV-11 lidar is significantly inferior, however it is much cheaper and convenient especially during hardware prototyping.

TABLE 1. Specifications of the two lidars from Fig.11.

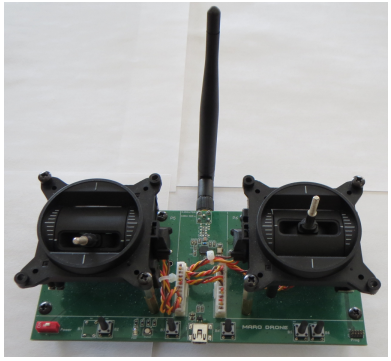
Lidar device:	**Hokuyo UTM-30LX**	**Neato XV-11**
Points per revolution	1,440	360
Revolution speed (ms)	25	250
Max range (m)	30	4
Cost (USD)	4,825	150



(a)



(b)



(c)

FIGURE 10. (a) Block diagram of a typical quadcopter drone. All major hardware components are illustrated, including the mechanical frame. The two major software or firmware components are programs that implement the main flight control algorithm and the control of the four motors via the ESCs. (b) Photo of the in-house built quadcopter. (c) Photo of the in-house built RC hand-held remote control.

(a) (b)

FIGURE 11. (a) Hokuyo UTM-30LX lidar used in [15]. (b) Neato XV-11 lidar device used in this project.

5.3 Investigation of Energy Consumption

One of the goals in designing and implementing from scratch our own quadcopter was to investigate energy consumption of the entire system. Note that there is not much available literature on the topic of energy consumption and techniques to optimize it[1]. We wanted to understand what are the main variables that affect energy consumption and the rate of consumption in order to identify ways to improve energy consumption, thereby increasing flight time on a single battery charge. We are particularly interested in energy consumption because the rather short flight time (directly determined by the power consumption for a given drone design and battery technology) of currently available quadcopters is we believe one of the primary reasons that hinder a faster wide spread adoption of quadcopters for civil applications. For example, in our experience with Parrot's AR.Drone v1.0 [25], if the flight time was initially when the drone was new about 14 minutes after two years of rare usage and about 50 battery charges, the flight time decreased to a merely 1-2 minutes.

In this section, we present our findings of the energy consumption investigation, which we performed on our own in-house built quadcopter. In our investigation, we focused on the energy consumption of the major components that make up the drone system (see Fig.10.a): the DC motors, the ESCs, and the flight controller which includes the radio and sensor blocks. Our measurements are done on an actual real implementation of our quadcopter, which is shown in Fig.10.b. The results of our investigation are summarized in Fig.12.

We notice that the 95% lion-share of the overall power consumption is attributed to the DC motors. The ESCs themselves account for 4% while the flight controller consumes about a merely 1%. One reason for which the flight controller does not consume more is because the radio block in our current implementation does not include an amplifier (which would be more power hungry) and because of which the user-control distance is only tens of feet. Zooming into the ESCs, we notice that power

[1] Beyond the obvious means of selecting better batteries and motors, or lighter mechanical frames [24].

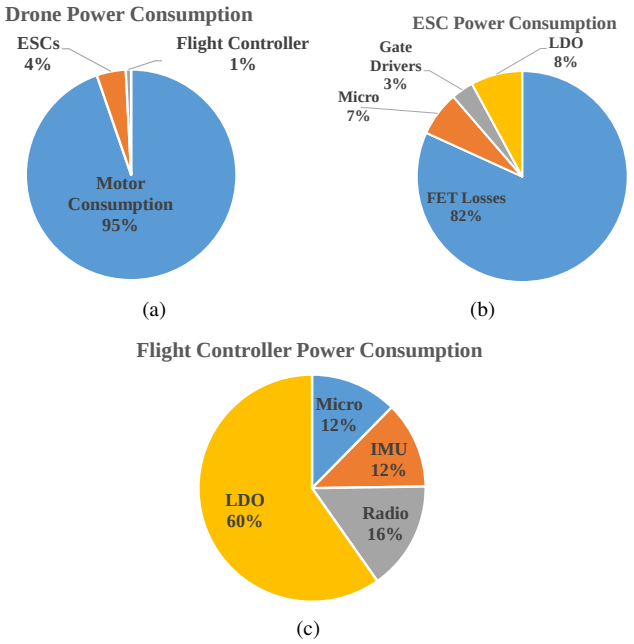


FIGURE 12. (a) Breakdown of the power consumption among the main components of the quadcopter drone system. (b) Breakdown of the power consumption at the level of the ESCs. (c) Breakdown of the power consumption at the level of flight controller.

is dissipated as losses primarily into the field effect transistors (FETs, i.e., commutation switches) and the low-dropout (LDO) regulators. The LDO regulator is also accounting for the majority of the power consumed by the flight controller.

Obviously, one could reduce power consumption by selecting high efficiency power converters and high performance FETs (i.e., low ON resistance) and thus minimize losses. Additionally, one could decrease the weight of the whole system by using light weight materials for the frame. Furthermore, high performance DC motors and propellers and improved battery technology (i.e., small volume and high energy capacity) would help to improve the power consumption profile. However, once these *hardware* design decisions are made[2], the question is: Is there anything else that one can do to improve the power consumption profile, and thereby increase the flight time?

Looking at the pie-chart from Fig.10.a, one would think that not too much could be done aside from selecting better motors. However, the view offered by the charts in Fig.10 is only partial. In other words, these breakdown charts only tell us which are the major contributors to the power consumption of the overall quadcopter system, but they do not capture how fast or at what *rate* energy is consumed from the fixed capacity battery, which is truly directly impacting the battery state of charge and thereby the duration of the flight time. Currently, we do not have test and validation results of what we suspect might be the only factor under our control which could have a significant impact on the rate of battery energy depletion. Aside from the type of maneuvers or flight characteristics[3], the quadcopter performs and aerial conditions (such as wind or air drifts), we suspect, based on our insights gained while developing the current quadcopter prototype, that the actual flight control algorithm (stabilization and positioning) may play a significant role in how fast energy is consumed from the battery. Our insight essentially says that the amount of energy and the rate of its consumption is less for flight control algorithms that are more optimal in the sense that the stabilization portion is smoother and less jittery. When such stabilization is wobbly and the system is on the verge of being unstable (the quadcopter would flip in such unstable mode of operation), the motors are forced to operate at faster speeds and the speeds are changed more often (i.e., motors spend more time in transient like operation mode). Such operation modes place a bigger burden on motors, which thus consume energy faster. Note that jittery flight can also be affected by the imperfections and differences between motors and propellers themselves, which can introduce difference between the current drawn by the motors. We suspect that better flight control algorithms can impact the rate at which energy is consumed and thus on the battery and flight time. A more in depth investigation of this is left to our future work.

6 Conclusion and Future Work

We introduced a computationally efficient iterative closest line (ICL) algorithm for determining indoor position drift of a quadcopter. We presented details of the implementation of the proposed algorithm as well as of a technique for outlier and dynamic object rejection from lidar data clouds. We validated the proposed algorithm using simulations with real world data. Furthermore, we discussed how the proposed ICL algorithm was integrated into the flight control algorithm of an in-house built quadcopter drone. This integration was verified in simulation. Finally, we discussed system-level design and implementation aspects of the quadcopter, including an investigation of the quadcopter power consumption.

While the the proposed ICL algorithm for position control has been validated on real world data and its testing, as part of the overall flight control algorithm, was done in simulation, and while the in-house drone is completed, we still have to physically place the lidar sensor on the drone and perform lidar and IMU data fusion. This is part of our ongoing and future work.

[2]Indeed, one can assume that the best hardware components are used; most likely at higher financial costs.

[3]For example, if a quadcopter keeps some forward motion at all times, it requires much less power than hovering. This is attributed to the so called *helicopter translational lift* phenomenon.

Acknowledgment

This work was supported in part by the Department of Electrical and Computer Engineering at Marquette University.

REFERENCES

[1] Neato Robotics, LaserSmart Mapping & Navigation, 2016, https://neatorobotics.com/lasersmart-mapping-navigation

[2] A.L. Salih, M. Moghavvemi, H.A.F. Mohamed, and K.S. Gaeid, "Flight PID controller design for a UAV quadrotor," *Scientific Research and Essays*, vol. 5, pp. 3360-3367, 2010.

[3] L. Argentim, W. Contrimas, P.E. Santos, and R. Aguiar, "LQR and LQR-PID on a quadcopter platform," *IEEE Int. Conference on Electronics, Informatics and Vision*, 2013.

[4] T. Madani and A. Benallegue, "Backstepping control for a quadrotor helicopter," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006.

[5] G.V. Raffo, M.G. Ortega, and F.R. Rubio, "An integral predictive/nonlinear H infinity control structure for a quadrotor helicopter," *Automatica*, vol. 46, pp. 29-39, 2009.

[6] M. Jun, S.I. Roumeliotis, and G.S. Sukhatme, "State estimation of an autonomous helicopter using Kalman filtering," *Intelligent Robots and Systems*, 1999.

[7] M. Hehn and R. D'Andrea, "Quadrocopter trajectory generation and control," *IFAC World Congress*, 2011.

[8] T. Tomic et al., "Toward a fully autonomous UAV: research platform for indoor and outdoor urban search and rescue," *IEEE Robot. Autom. Mag.*, vol. 19, no. 3, pp. 46-56, Sep. 2012.

[9] H. Lim, J. Park, D. Lee, H.J. Kim, "Build your own quadrotor: open-source projects on unmanned aerial vehicles" *IEEE Robot. Autom. Mag.*, 2012.

[10] T. Krajnik, V. Vonasek, D. Fiser, and J. Faigl, "AR-Drone as a platform for robotic research and education" *Eurobot Conference*, 2011.

[11] R. Mahony, V. Kumar, and P. Corke "Multirotor aerial vehicles: modeling, estimation, and control of quadrotor," *IEEE Robot. Autom. Mag.*, vol. 19, no. 3, pp. 20-32, Sep. 2012.

[12] Y. Zhang, J. Wang, X. Wang, C. Li, and L. Wang, "3D LIDAR-based intersection recognition and road boundary detection method for unmanned ground vehicle," *IEEE Int. Conf. on Intelligent Transportation Systems(ITSC)*, 2015.

[13] D. Jeon and H. Choi, "Multi-sensor fusion for vehicle localization in real environment," *IEEE Int. Conf. on Control, Automation and Systems (ICCAS)*, 2015.

[14] Z. Zheng and Y. Li, "LIDAR data registration for unmanned ground vehicle based on improved ICP algorithm," *IEEE Int. Conf. on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, 2015.

[15] S. Winkvist, "Low computational SLAM for an autonomous indoor aerial inspection vehicle," *Ph.D. Thesis*, University of Warwick, 2013.

[16] A. Gibiansky, "Quadcopter dynamics, simulation, and control," *Tutorial*, 2016. [Available], Online: http://andrew.gibiansky.com

[17] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *Int. Journal of Computer Vision*, vol. 13, no. 2, pp. 119-152, 1994.

[18] A. Censi, "An ICP variant using a point-to-line metric," *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2008.

[19] M. Alshawa, "ICL: iterative closest line, a novel point cloud registration algorithm based on linear features" *ISPRS 2nd summer school in Ljubljana*, pp. 1-6, 2007.

[20] Stephen Boyd and Lieven Vandenberghe, Convex Optimization, Cambridge University Press, First edition, 2004.

[21] P. Kettle, A. Murray, and F Moynihan, "Sensorless control of a brushless DC motor using an extended Kalman estimator," *Intelligent Motion (PCIM)*, 1998.

[22] W. Brown, "Brushless DC motor control made easy," *Microchip Technology Inc. Application Note*, AN857, 2002.

[23] L. Prokop and L. Chalupa, "3-Phase BLDC motor control with sensorless back EMF zero crossing detection using 56F80x," *Freescale Semiconductor Application Note*, AN1914, Rev. 1, 2005.

[24] J.A. Benito, G. Glez-de-Rivera, J. Garrido, and R. Ponticelli, "Design considerations of a small UAV platform carrying medium payloads," *Design of Circuits and Integrated Circuits (DCIS)*, 2014.

[25] Parrot's AR.Drone v1.0 purchased in 2012. [Available], Online: http://www.parrot.com/usa.

[26] MESS Lab at Marquette University, 2016. [Available], Online: http://dejazzer.com/hardware.html