A Signal-Processing Approach to Terrain Following for Remote-Sensing Remotely Piloted Aerial Systems

by

Robert Mark William Skelly
B.Sc., University of Victoria, 2015

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Geography

A Signal-Processing Approach to Terrain Following for Remote-Sensing Remotely Piloted Aerial Systems

by

Robert Mark William Skelly

B.Sc., University of Victoria, 2015

Supervisory Committee

Dr. K. Olaf Niemann, Supervisor

(Department of Geography)

Dr. Yvonne Coady, Outside Member

(Department of Computer Science)

ABSTRACT

Recent advances in technology have enabled an explosion of interest in compact remotely piloted aerial system (RPAS). RPAS are of particular interest to remote sensing scientists, as they provide an accessible, low-cost way to perform surveys over small sites at extremely high levels of detail. The interaction between low flight altitude and high relative terrain relief introduces safety, efficiency and data-quality issues which can only be resolved by closely following the terrain.

A terrain following system would ensure that specific measures of data quality – point density, the signal-to-noise ratio and scale distortion – are held as near as possible to constant. Safety and efficiency would be addressed by ensuring that a terrain following trajectory does not contain gradients or inflections that exceed the vehicle's dynamic limits, in particular its maximum acceleration and velocity.

This research develops a forward-looking, predictive terrain following system using a signal-processing approach, in which the terrain itself is conceived as a signal and filtered appropriately to extract a trajectory which achieves these objectives.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

My supervisors, Drs. K. Olaf Niemann and Yvonne Coady, for opening this door and seeing me through it.

My friends and colleagues at HLRG, Georgia Clyde, Diana Parton, Cydne Potter, Geoff Quinn, Roger Stephen and Vabio Visintini, for the perfect combination of genius and chaos that made me feel at home.

My parents, Sonia Alexandra and Robert Evan Skelly, for loving me no matter what.

Chapter 1

Introduction

Remotely piloted aerial systems (RPAS), also known as unmanned aerial vehicles (UAVs), are small, reusable aircraft, controlled remotely by a human operator or (semi-)autonomously, which can range in size from insect-scale to jet-powered military aircraft (Avadhanula et al., 2002; Xinyan Deng et al., 2003). RPAS are the subject of an explosion in engineering, scientific and commercial interest, with the military sector driving much of the research into the development of RPAS and related technologies. Perhaps surprisingly, the development of RPAS began in 1916 and continued with the involvement of the United States military, culminating in extensive utilization of RPAS during the war in Vietnam (Cook, 2007; Valavanis, 2007). The emergence of an entrepreneurial technological culture in the late 20^{th} and early 21^{st} centuries, and the ability of firms to design and produce highly sophisticated, miniaturized components has enabled basement tinkerers, commercial startups and academics alike to exploit the capabilities offered by RPAS, rapidly and at little cost.

In particular, the availability of compact, low-power computers with high processing speeds, along with advances in real-time operating systems, have enabled the development of flight-control systems which can safely manage the inherent in-

stability of multi-rotor aircraft (Valavanis, 2007). The proliferation of system-on-a chip (SoC) microporicessors and microelectromechanical sensor (MEMS) sensors, including optical and chemical sensors, accelerometers, gyroscopes and magnetometers facilitiates the production of compact sensing instruments and inertial measurement units suitable for flight control. Advances in battery technology are providing energy at a density, light weight and form factor that could previously be attained only with hydrocarbon fuels (along with the deleterious vibration caused by reciprocating engines). Importantly, one does not need access to electrical engineering knowledge, nor to private production facilities, to assemble sophisticated electronic hardware into a remote-sensing RPAS. The building blocks are now small enough, cheap enough and accurate enough that almost anyone can build an aircraft and sensor package to their requirements (Kröger, 2010).

In the scientific remote-sensing field, where the execution of an aerial survey could entail hundreds of thousands of dollars in costs for planning, licensing, instrumentation, pilots and aircraft, the advent of RPAS provides researchers with the opportunity to conduct research at much lower cost with little turnaround time. Naturally, there are compromises to be made between traditional aerial remote sensing and the use of RPAS. RPAS tend to be limited to low altitudes, short flight durations and small site sizes. The instrumentation – multi- and hyperspectral imagers and LiDAR – has only recently achieved a level of quality sufficient for research, and form factor small and light enough for inclusion on a RPAS. Additionally, many of these instruments are designed for uses other than remote sensing, in particular LiDAR devices, which are often designed for the automotive market, and spectral sensors which are designed for use in manufacturing quality control. However, with the drawbacks come advantages. The level of detail attainable with a low-altitude RPAS survey would be impossible with a traditional aerial campaign and the attendant costs, danger and

potential disruption (e.g., by rotor downwash and noise) of the subject.

Traditional aerial surveys have the advantage that, at typical survey altitudes of $250\,\mathrm{m} - 1000\,\mathrm{m}$, variations in terrain relief and vegetation canopy height are insignificant relative to the platform altitude – except in extreme cases, such as alpine terrain – with minimal scale distortion in the resulting imagery. In addition, because there are many structures, both natural and human-made, that may project above an RPAS's altitude, the vehicle must have the ability to detect and avoid hazards. Crewed aircraft, with an alert pilot and high altitude, rarely face such obstacles.

The quality of remotely-sensed data can be quantified in many ways, but this research will focus on those that are directly related to instrument-to-subject distance or platform altitude: the point density and detector footprint of LiDAR data, the scale distortion of spectral imagery and signal-to-noise ratio (SNR), which affects both types of data in different ways.

The foregoing quality and safety concerns can be resolved, at least in part, by a terrain-following system that accurately maintains the altitude of the aircraft and, by implication, its instrument range. Such systems are currently available but fall into two general categories: real-time altitude-control systems that use a nadir-oriented laser rangefinder to manage the aircraft's altitude *reactively*, and those that require the pre-existence of a digital terrain model (DTM) or digital elevation model (DEM).

The first type of system cannot anticipate variations terrain relief and so has no way of optimizing its trajectory to accommodate the physical limitations of the platform or data quality constraints. DEM-based terrain following systems require a high-resolution, up-to-date terrain model. If this is not available, a preliminary LiDAR or photogrammetric survey is required, meaning that at least two full surveys must be conducted. Remote sensing surveys are time-sensitive (the optimal time for hyperspectral imaging is near solar noon) and flight plans must be adaptable to

prevailing conditions, such as weather. Since the terrain model must be produced first, the researcher has no way of knowing if conditions will still be suitable by the time the second survey begins. A real-time terrain following system obviates the need for additional flights. Also, field work is expensive and exhausting; cutting field survey times by at least 50% would be a boon.

Figure 1.1: Typical spacing of vineyard rows. Culmina Winery, Oliver, BC. August 2017.

Any terrain-following system must be able to accommodate the different surface characteristics and mission objectives that a researcher may encounter. For example, a vineyard's rows (figure 1.1) are spaced roughly 3 m apart, creating a periodic series of depressions and ridges. It would be undesirable for the platform to descend into these spaces; rather, its trajectory should carry it smoothly over the surface implied

Figure 1.2: Low-frequency terrain variation, Culmina Winery, Oliver, BC. August 2017.

by the tops of the rows while it faithfully follows the low-frequency variations in the terrain (figure 1.2). In this sense, the surface is analogous to a signal and the terrain-following system a low-pass filter.

As it happens, the terrain-as-signal metaphor gestures towards a solution to the terrain-following problem. If the terrain is understood as a signal composed of sub-signals with a variety of frequencies and magnitudes, a filter can supply the means of extracting a representative model of the terrain. The physical body of the aircraft cannot negotiate arbitrarily steep gradients, so a functional representation of the terrain should be smooth and at least twice-differentiable, with the first two derivatives as proxies for the vertical velocity and acceleration of the platform, respectively.

This research therefore employs a signal-processing approach to the extraction of a navigable trajectory and develops an optimal, forward-looking, predictive terrain-following system which will be implemented as an auxiliary controller mounted on a RPAS fitted with a forward-mounted laser rangefinder. Candidate trajectory functions will be applied to existing point datasets to model their suitability and identify the ideal parameters. The selected trajectory function will be implemented in the flight controller and tested in the field.

Chapter 2

Literature Review

2.1 Remote Sensing

Remote sensing is the acquisition of information about an object or phenomenon without the necessity of physical contact between it and the observer. The human eyes, nose and ears are remote sensing instruments, collecting the electromagnetic, chemical and mechanical signals that comprise sight, smell and hearing, respectively (Gupta, 2018; Schott, 2007).

In the geographic and earth sciences, remote sensing is typically (though not exclusively) conducted from above the earth using aircraft or orbiting satellites carrying instruments that measure electromagnetic or gravitational emissions from the Earth (Schott, 2007). Electromagnetic sensors may be further categorized as *active*, including LiDAR and radio detection and ranging (RADAR) which emit a pulse of radiation, then measure the time-of-flight of the echo, along with its intensity and other qualities; and *passive*, which capture the reflected energy from natural sources such as sunlight or endogenous emissions like thermal infrared radiation (i.e. heat) (Gupta, 2018; Schott, 2007).

This research considers two types of instruments typically employed in RPAS remote sensing, a "pushbroom"-type hyperspectral imaging spectrometer (passive) and a scanning LiDAR (active).

2.1.1 Hyperspectral Imaging Spectrometry

The imaging spectrometers typically mounted on RPAS are of the linear-array "push-broom" type, which have no moving parts and are therefore small, efficient and reliable enough for RPAS remote sensing (Schott, 2007). In a pushbroom scanner, incoming radiation passes through a slit and reflects off of a diffraction grid which decomposes the beam into narrow spectral bands. The sensor elements – frequently of the silicon complementary metal-oxide-semiconductor (CMOS) type – are arranged in a two-dimensional grid such that across-track physical space occupies one dimension and spectral bands the other. Individual frames produced by the sensor constitute the third, along-track, dimension (so long as the sensor is moving). The three dimensions are interleaved into a multi-band raster file – a "data cube" (Headwall Photonics Inc., 2017).

The pushbroom scanner moves with the vehicle, with each sensor element recording in a continuous swath parallel to the flight path. As the sensor moves across a target, each element records the radiant intensity in one band, continuously. This provides the advantage, over line- or whiskbroom scanners, of increasing the scanner's dwell time (Schott, 2007). Dwell time, or integration time is simply the amount of time that radiance from the scene is recorded by the sensor. The sensor accumulates the continuous data stream from each element, storing the result at intervals determined by the frame period as a digital value. The frame period – assuming there are no gaps between frames – is the inverse of the frame rate of the sensor (Schott, 2007). While the across-track resolution is determined by the across-track field of view of

the elements of the linear array and altitude of the vehicle, the along-track resolution is determined by the along-track field of view, frame period, altitude and velocity.

Signal-to-Noise Ratio

The signal-to-noise ratio (SNR) is a measure of the magnitude of the true measurement (photons received from the emitter over a defined spatial and spectral extent) versus noise (specious activations of the sensor elements generated by external sources of radiation or internally). There are several sources of noise in hyperspectral imagery (Barducci et al., 2007; Gupta, 2018; Rogaß et al., 2011; Ruyten, 1999; Schott, 2007):

- shot noise is due to the natural Poisson-distributed variability in photon flux;
- thermal noise, or dark current, results from electrons generated spontaneously within the photosensitive material;
- read noise is produced during the generation of a charge from the instrument, and during analog-to-digital conversion;
- charge transfer errors result from defects in the sensor material;
- round-off error, or quantization error, which results when a continuous analog signal is digitized;
- and smearing, which occurs as the photosensor remains active during transfer of the accumulated charge out of the sensor.

These sources have in common that they cannot easily be mitigated by the user, but are intrinsic to the instrument and environment. However, by increasing the magnitude of the signal, the relative influence of noise can be reduced (Ben-Dor et al., 2012; Schott, 2007). Unfortunately, as spectral bands are narrowed, each band's

share of the total flux is reduced, which worsens the SNR (Schott, 2007; Villafranca et al., 2012).

While the human eye can detect patterns when the SNR is as low as 0.1, a SNR greater than 10 is preferred for quantitative remote sensing applications. Greater than 40 is considered excellent and 100 is ideal (Villafranca et al., 2012). Some of the factors affecting the strength of the signal can be directly controlled, by (Gupta, 2018; Schott, 2007):

1. increasing the transmissivity of the sensor optic (lowering the f-number [1]);
2. maximizing the irradiant intensity (flying during solar noon, in good weather);
3. maximizing the dwell time (increasing the frame period and flying more slowly);
4. increasing the surface area of the target scanned during integration (flying at a higher altitude).

The last two items are related by the sensor integration time and V/H (velocity over height) ratio (Gupta, 2018). For a given V/H ratio the SNR remains constant (ignoring atmospheric effects). The researcher then has three degrees of freedom with respect to SNR at flight-time: the sensor frame period, velocity and altitude.

Geometric Distortion

The low altitude of RPAS tends to induce disproportionate scale distortions in remotely-sensed imagery and point data. In terms of the scale (s) (Avery & Berlin, 1992; Schott, 2007),

$$s = \frac{f}{H}, \tag{2.1}$$

[1] f-number is the ratio of the focal length to aperture or pupil diameter, used as a proxy for lens "speed." Not to be confused f, for focal length.

for given a constant focal length, $f = 50\,\text{mm}$, we can calculate the effect of a change to H (altitude) of 10 m, due to terrain relief, on an imaged object from 1000 m vs. 20 m, a typical RPAS remote-sensing survey altitude:

$$\begin{aligned} 1 - \frac{s_{990}}{s_{1000}} &: 1 - \frac{s_{10}}{s_{20}}, \\ 1 - 1.010101 &: 1 - 2.00000 \\ 0.010101 &: 1.000000 \\ 1 &: 99 \end{aligned} \tag{2.2}$$

Though the 1000 m flight is 50 times higher than the 20 m flight, the scale distortion for the latter is 99 times larger. This geometric identity affects both point clouds and spectral imagery.

The distortion of image scale is not uniform across an image. Gupta (2018) provides an extensive survey of the causes geometric of distortion, grouped into two categories – systematic and non-systematic. Non-systematic distortions result from unpredictable factors such as airframe perturbation due to wind, while systematic distortions are those caused by fixed or planned effects, such as aircraft altitude and relief displacement.

Aircraft pitch, roll and yaw are usually considered to be non-systematic in that they are induced by atmospheric disturbances. Pitch changes cause stretching and compression in the along-track dimension; roll causes across-track displacement and yaw causes differential stretching and compression across-track. Taking the case of yaw distortion: as the aircraft rotates on its z-axis, the scanner elements on the outside of the turn sweep the subject at a higher velocity, resulting in the acquisition of radiant energy from a large spatial extent which is spatially compressed into the rectified pixel. The scanner elements on the inside of the turn experience the reverse

– radiance from a smaller spatial extent is stretched to fill the pixel. In either case, the integration time and SNR are equal, but two pixels of the same size contain information from regions of different size. Variations in forward velocity cause similar distortions, though they remain constant across the track. Variations in altitude induce proportional changes in overall image scale, along with variations in SNR, as at higher elevations, the effective size of the reflector becomes larger. The reverse is true at lower elevations.

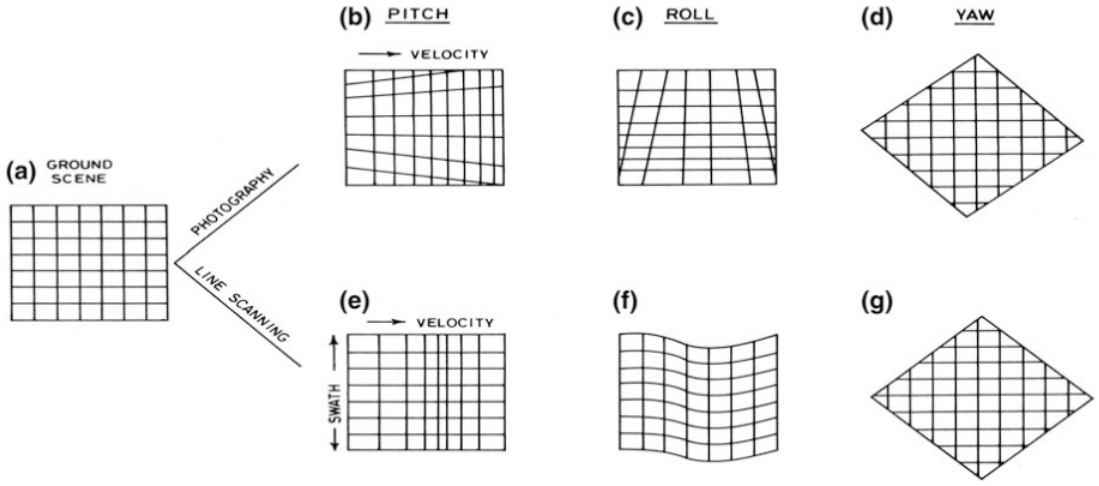


Figure 2.1: Types of geometric distortion. Items (e), (f) and (g) refer to push-broom scanners (Gupta, 2018).

From the sensor's perspective, there is no distortion; the sensor always produces a uniform three-dimensional grid of discrete values. Distortion occurs when the grid is rectified so that each measured quantity is identified spatially with the its target. Whether grid cells are conceived as points, circles or rectangles, it is implied that the measurement represents a target with a specific, uniform shape and extent. The scale distortions discussed above affect rectified imagery only in the sense that the quantity of measured radiance and its spatial scope may vary from cell to cell. These variances may be minimized by reducing variations in platform velocity, altitude and

orientation.

2.1.2 LiDAR

LiDAR operates by emitting a pulse of light from a laser towards a target and measuring the time required for the reflection, or echo, to return to the detector. The relation,

$$r = \frac{tc}{2}, \tag{2.3}$$

gives the range r or distance to the target where t is the time of flight and c is the speed of light, or (e.g., Behroozpour et al., 2017; Hancock et al., 2017; Hancock et al., 2015; Smith et al., 2014). This time-of-flight LiDAR (as opposed to phase-shift LiDAR, which is seldom used in remote sensing) can be subdivided into two main categories: discrete and waveform. In either case, a pulse with a width on the order of a few nanoseconds (Burns, 1991) is emitted and the echo is received as a waveform, modified by its interaction with the target and intermediate reflectors (Hancock et al., 2017). Whereas the emitted pulse waveform has a single, roughly gaussian peak (Burns, 1991), the echo may contain peaks at varying intensities, distributed over its length due to backscattering from interactions with objects in the path of the laser. Discrete LiDAR is produced by extracting one or more peaks from the echo and storing them as range and intensity values (figure 2.2) (Hancock et al., 2017).

LiDAR has uses in terrain modelling and bathymetry (Brasington et al., 2012), hydrology (Brisco et al., 2015; Shook et al., 2013) and aquatic vegetation studies (Gilmore et al., 2008); forest ecology (Aubrecht et al., 2010), health (Hopkinson et al., 2013; Niemann et al., 2015; Solberg et al., 2006), structure (Niemann et al., 2011) and biomass (Barbosa et al., 2014; Zhao & Popescu, 2009) and fire studies (Martín-alcón et al., 2015); urban infrastructure inventories (Huang et al., 2008); and as an accessory to other remote sensing techniques (Niemann et al., 2007). While the list of

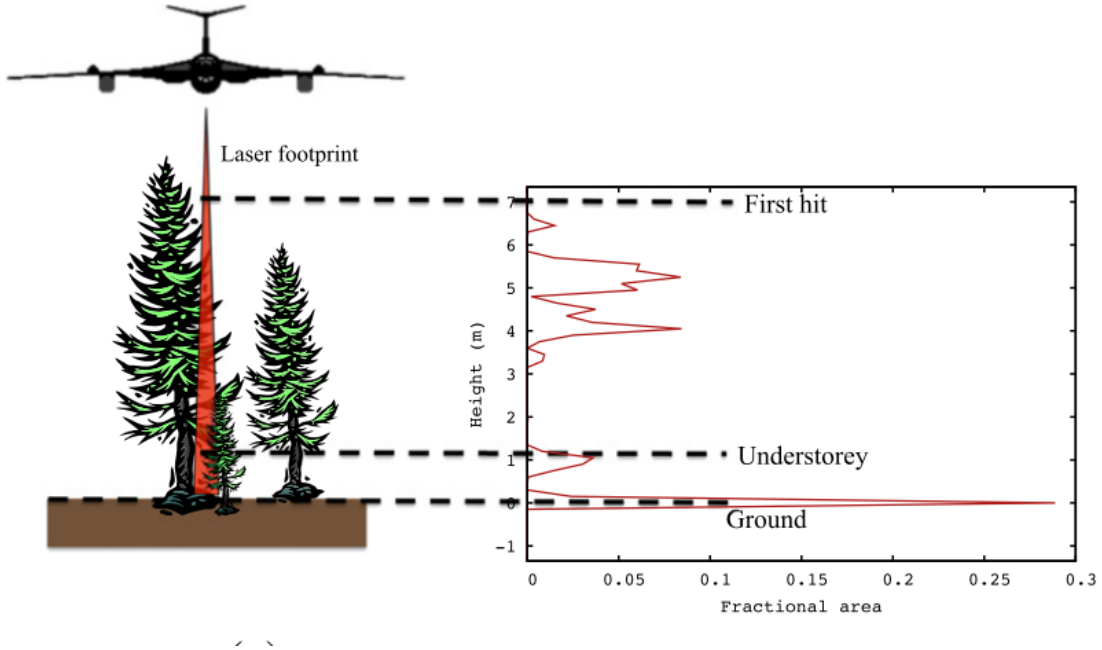


Figure 2.2: Discretization of LiDAR echo waveform into three discrete "hits" (Hancock et al., 2017).

potential uses is virtually endless, certain data quality considerations are universal.

Signal-to-Noise Ratio

As in spectral imaging, the SNR is a major determining factor in the quality of LiDAR data. Burns (1991) simulated the interplay between LiDAR hardware properties and noise, range and accuracy. For this research the salient relations concern the effects of distance to the target on noise. Observe that given the fixed beam divergence of a laser, the altitude of the platform is proportional to the diameter of the detector's footprint (m^2) by,

$$A_s = \frac{\pi a^2}{\cos \theta_s}. \tag{2.4}$$

The background flux P_B (W) on the detector is then,

$$P_B = L_\lambda A_s \cos \theta_s \omega_D \delta_\lambda T_R \exp(-\sigma R), \tag{2.5}$$

where L_λ is solar spectral radiance ($\mathrm{Wm^{-2}\mu m^{-1}sr^{-1}}$), θ_s is the angle between the target normal and the receiver axis (rad), ω_D is the laser solid angle (sr), δ_λ is the receiver spectral bandpass (µm), T_R is the transimissivity of the receiver optical system, σ is the atmospheric extinction coefficient ($\mathrm{m^{-1}}$) and R is the slant range to the target (m) (Burns, 1991). Clearly, the quantity P_B is proportional to A_s which is, in turn proportional to the platform altitude.

Meanwhile, the detected laser power P_S (W) is given by,

$$P_S = L_T A_T \cos\theta_s \omega_D T_R T_F \exp(-\sigma R), \tag{2.6}$$

where L_T is the target radiance ($\mathrm{Wm^{-2}\mu m^{-1}sr^{-1}}$) and T_F is the transmissivity of the receiver's spectral filter (which removes extraneous wavelengths of background radiation).

Note that the derivation of L_T,

$$L_T = \frac{4P_L T_T \eta \cos\theta_s \exp(-\sigma R)\rho_T}{\pi^2 {\beta_T}^2 R^2}, \tag{2.7}$$

contains a squared R (slant range) term in the denominator and an exponential atmospheric extinction term in the numerator, in addition to the extinction term in 2.6. The derivation of L_λ,

$$L_\lambda = \frac{E_\lambda \rho_B}{\pi}, \tag{2.8}$$

is not attenuated by an extra exponential or inverse quadratic term: the variance in the platform elevation has a disproportionate effect on signal power relative to background noise.

In the RPAS context – due to the low power of the laser and the large variance in terrain relief relative to altitude – this can impact pulse detection and range accuracy.

Burns (1991) uses a probability of false alarm (PFA) and probability of single pulse detection (PSP) of 0.001 and 0.999, respectively, to constrain the SNR. That is, the SNR should be sufficient to enable single pulse detection with 99.9% reliability and reject false pulses (noise) 99.9% of the time (figure 2.3).

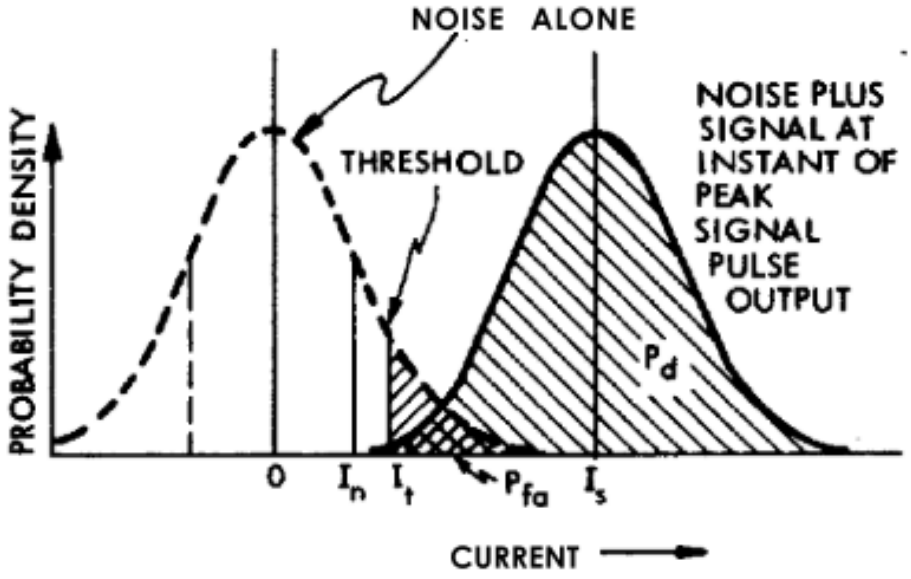


Figure 2.3: Depiction of probability of detection vs. false alarm at a given threshold (Burns, 1991).

Burns (1991) provides a method for modelling the precise relationship between SNR and range using coefficients from RCA electro-optics handbook (RCA Corporation, 1974), but there is not sufficient information provided for the rangefinder used in this research to precisely determine the SNR or range. The nominal range of 100 m (175 m for highly-reflective targets) is presumably dictated by the SNR, which varies disproportionately with the altitude of the vehicle. A survey in hilly terrain, or at a high nominal altitude, will experience data loss if it exceeds the useful range of the laser. Even if the altitude remains within the useful range of the rangefinder, there may be altitude-driven changes in the range accuracy or the resistance to specious

echoes.

Point Density

The other major effect of aircraft altitude on LiDAR data is in terms of point, or posting density. The point density is a description of the resolving power of the instrument. For example, Cryderman et al. (2015) showed that the high-density point clouds could eliminate the need for break lines along rapid changes in slope; Berger et al. (2014) showed that low-density scans could inhibit the automated reconstruction of scanned surfaces; and Côté et al. (2011), Côté et al. (2012) showed that point density had impacts on the ability to reproduce the interior architecture of tree canopies due to occlusion. Of course, the posting density also has implication for the power of parametric statistics, as in Niemann et al. (2012).

Point density is proportional to altitude, and given approximately by the relation,

$$D_P = \frac{\theta A F}{V}, \tag{2.9}$$

where θ is the scan angle (rad), A is the altitude above the surface (m), F is the pulse frequency (kHz) and V is the platform velocity ($\mathrm{m\,s^{-1}}$). The point density may be multiplied by overlapping flight lines; the width of a flight line is, of course, also determined by the platform altitude. When an aircraft flying at altitude H transitions to a new altitude H', the density D changes to D':

$$D' = \frac{D(H - H')}{H}. \tag{2.10}$$

Point density becomes an issue when point data are discretized into a grid space as statistical derivatives. If a vehicle flying at 10 m altitude records points at a density

of 3000 pts/m^2, a terrain feature 3 m tall will reduce the density to,

$$\begin{aligned} D' &= \frac{3000 * (10 - 3)}{10} \\ D' &= 2100. \end{aligned} \tag{2.11}$$

A density of 2100 pts/m^2 may seem large at first glance, but one must note that low-altitude RPAS remote sensing can produce outputs at much higher resolution than traditional aerial surveys – 5 cm or less. If one were to produce a 5 cm LiDAR-derived biometric raster to correspond with a hyperspectral image of the same subject, the nominal point density in this example would be 5.25 pts/cell, and the reliability of any derived statistical estimates would become precarious.

An interesting consequence of density variation has to do with the fractal nature of physical structures (Burrough, 1981, 1983; Mandelbrot, 1967): biophysical processes occur at multiple scales simultaneously and the resolving power of an instrument can dictate whether an inquiry can succeed at the desired scale. Lovell et al. (2005) note that, in sparse canopies, the odds of a small footprint laser sampling the actual top of any tree is dependent on the posting density. At low densities the canopy height may be under-estimated. Non-parametric methods to estimate surface rugosity (e.g., Du Preez, 2014) also suffer from the dependence on point density.

2.2 Terrain Following

It is clear that the quality of remotely-sensed data depends, at least in part, on the careful maintenance of platform altitude. Terrain following/terrain avoidance (TF/TA) – occasionally stated with the inclusion of threat avoidance (TF/TA^2) – is a topic of much interest in engineering, artificial intelligence and military research. All research into TF/TA has in common the goal of balance competing objectives,

namely, obedience to the dynamic constraints on a vehicle and pilot (e.g., maximum rate of climb or maximum g-force) and the need to fly close to the terrain without colliding with it.

Military research into TF/TA is primarily interested in "terrain masking", that is, hiding the aircraft from enemy ground fire as it approaches a target (Asseo, 1988; Funk, 1976; Krachmanlick et al., 1968; Menon et al., 1991; Starling & Stewart, 1971), while preserving its speed or time-of-arrival or minimizing energy expenditures. These early efforts in TF/TA involved the use of RADAR and slow, bulky flight control systems (Starling & Stewart, 1971) and sometimes a human pilot following a computer-generated guide (Menon et al., 1991).

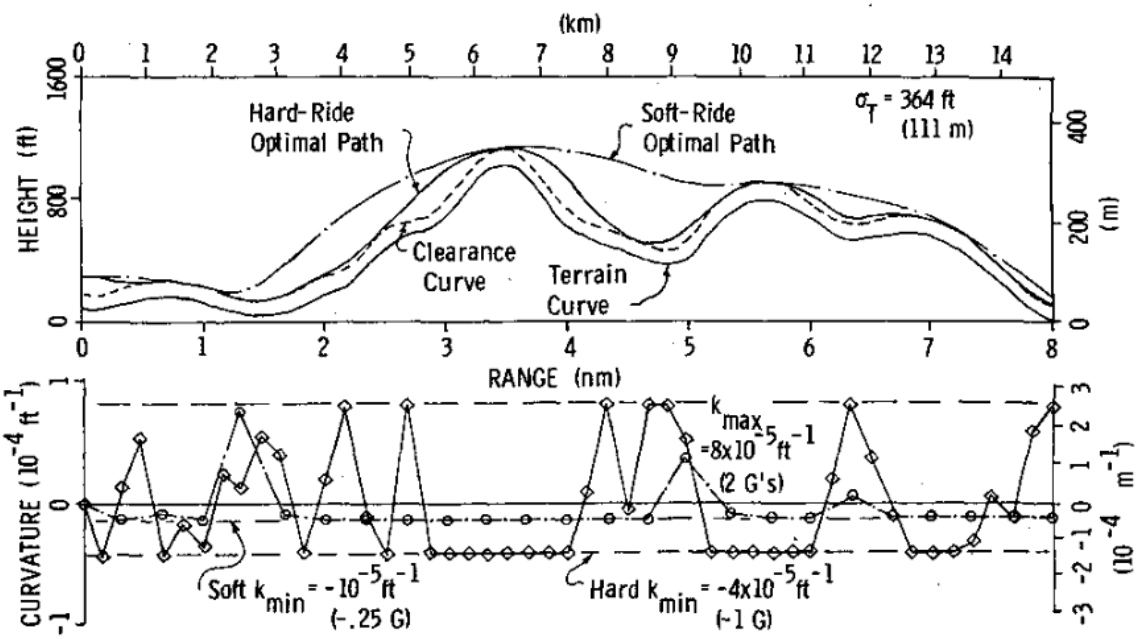


Figure 2.4: Cubic spline trajectories for "moderately rough terrain" showing "hard" and "soft" rides and required accelerations (Funk, 1976).

Funk (1976) noted some issues with previous research into terrain following for military applications: the paths they generated were not smooth enough to be negotiated by a physical aircraft, they did not respect the dynamic constraints of the

aircraft and they tended to only acknowledge single critical points (i.e., peaks) on the terrain. The junctions between trajectory segments could stress the pilot, control system and airframe. Funk's solution (figure 2.4) was to determine a two-dimensional cubic spline trajectory, optimized by nonlinear programming. Splines could be forced to respect an aircraft's (and pilot's) dynamic limits by constraining its derivatives, and individual segments could be joined smoothly by holding the derivatives equal at the junctions. Funk developed the notion of "hard" and "soft" trajectories which would balance the need for terrain fidelity with the dynamic constraints on the pilot and airframe. Lu and Pierson (1995) extended this idea, using an inverse dynamics approach to optimal thrust control.

Funk's "frame length" concept concerns the distance required to execute a "characteristic manouvre" (Funk, 1976) – a "symmetric manoeuvre" over an obstacle of "maximum expected height" (typically three times rugosity, or the standard deviation of terrain elevation), including maximum pull-up, push-over and transition arcs. The symmetric manoeuvre is represented using piecewise cubic splines whose integrals must sum to zero for the resumption of level flight. In the fixed-wing military aircraft context, the constraints are those imposed by flight envelope of the craft. A remote-sensing RPAS is no different, though the set of constraints varies slightly to include the availability of thrust and rotational velocity (for thrust vectoring) and data-quality.

Through "efficient programming" for the CDC 6600 computer, Funk proposed a 3 s frame time which, at Mach 1, would yield a trajectory update every 1067 m (Funk, 1976). For a relatively high (for a RPAS) horizontal velocity of $10\,\mathrm{ms}^{-1}$, this gives an update every 30 m. Given the advances in computer technology in the intervening 41 years – the CDC 6600 was introduced in 1967 and weighed over 4.5 t – 3 s would be an extremely pessimistic estimate for a trajectory update period using currently-

available hardware.

There were several other early attempts at functional terrain representation. Menon et al. (1991) developed a helicopter-oriented terrain following strategy, also using cubic splines, but suggested the use of polynomial or spatial-frequency domain smoothing to ensure the second-order derivability of the trajectory. Menon et al.'s (1991) use of a visible "guide" implies additional smoothing, as the pilot's reflexive delay in incorporated into the realization of the trajectory. Twigg et al. (2003) expanded on Menon et al.'s (1991) work by computing constant-energy – as opposed to constant-velocity – trajectories. This could be an interesting line of inquiry, given the scarcity of power on board an RPAS, however, energy in this case is conserved by altering the vehicle's path *horizontally* to find a low-energy path. The remote-sensing RPAS does not have this freedom as it is confined both to a predetermined ground track and to a specific horizontal velocity.

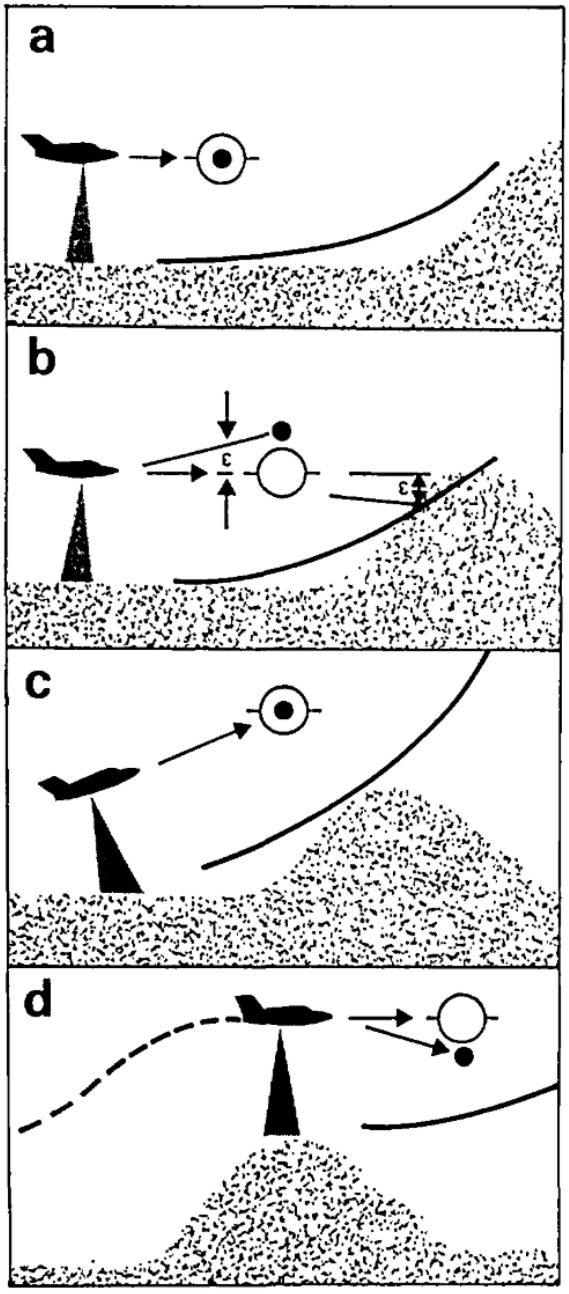


Figure 2.5: The Starling "ski" (Starling & Stewart, 1971).

Starling and Stewart (1971) proposed the use of a quadratic "ski" (figure 2.5), so called due to its shape and (metaphorical) purpose. In level flight, the straight section of the ski would be oriented horizontally, with its end beneath the aircraft. If a hill intruded through the upturned forward end of the ski, the difference between the terrain height and the ski surface, ϵ, would be used to calculate a rotation around the aircraft's pitch axis, and the vehicle would climb over the hill. Having surmounted the hill, the ski would no longer intersect with any terrain, and would return to the level position, causing the aircraft to descend at some controlled rate. This solution is, in effect a smoother which could be adjusted by varying the shape of the ski and its influence over the aircraft's controls (gain).

In more recent times, research into TF/TA has expanded into civilian and research activities, using more advanced, compact sensors and highly sophisticated optimization or artificial intelligence solutions. Several broad categories of research have emerged in terms of sensors and of trajectory construction and optimization algorithms. Researchers have even adopted biorobotic approaches to autonomous vehicle control, inspired by the sensory and cognitive apparatus of living creatures.

The two obvious sensory approaches for biorobotic control are sound – echolocation or sound navigation and ranging (SONAR) – and vision. Mwakibinga and Lee (2005) built a simple, bat-inspired binaural echolocation navigation system, though the detection range was severely constrained due to the unavoidable problem of poor sound propagation in compressible media (e.g., air).

Digital cameras mimic the function of insect eyes at low resolutions, and mammalian eyes at high resolutions. Many researchers (Beyeler & Floreano, 2009; Hérissé et al., 2010; Netter & Franceschini, 2002; Ruffier & Franceschini, 2004, etc.) have investigated the potential of monocular optic flow algorithms inspired by "primitive" insect nagivation. When a single optical sensor is used (as in the model of an insect's

compound eye), objects can be identified by the colour and intensity of their consituent pixels, and by the shapes of pixel clusters. Is not possible to infer anything about the spatial relationships between objects in a scene so long as the sensor is stationary and the controller has no prior knowledge. When the sensor is in motion, however, the trajectories of objects through the scene can be used to compute their positions relative to the sensor and to each other (Hérissé et al., 2010; Klein & Murray, 2007).

This is the basis for monocular simultaneous location and mapping (SLAM) technology, a topic of intense study in the field of augmented reality. The relative alignment of object paths, through a series of images over time, conforms to certain patterns depending on the properties of the optical system, the shape of the terrain and the motion of the sensor relative to it. For example, a nadir-oriented sensor on a vehicle moving parallel to flat terrain will "see" objects moving on parallel paths (accounting for lens distortion), through the field of view. For a vehicle descending along the optical axis of the sensor, objects will appear to move radially in the field of view, away from the optical axis. The generation of maps from un-referenced scenes using monocular SLAM (e.g., Klein & Murray, 2007) is the precursor to the autonomous control of insect-like RPAS. Fu et al. (2014) developed a visual-inertial SLAM-based control system using the panel tracking and mapping (PTAM) method (Klein & Murray, 2007) to generate a 3-dimensional map from optical data fused with inertial measurements to positively locate the vehicle within the map. This "proprioceptive" insight is used by the vehicle to estimate the map's absolute scale.

The recent availability of high-quality, compact laser rangefinders has spurred some late research interest into laser-based navigation. Clark and Roberts (2017) compared a forward-looking laser rangefinder-based solution to a SONAR-based terrain following mechanism built into the Matrice 100 made by DJI. They found the

sonar's range to be inadequate, losing effectiveness at 5 m and failing completely at 9 m – also a limitation of Mwakibinga and Lee's (2005) work. Clark and Roberts used a LightWare SF11/C – similar to the SF30/C used in this research – mounted on a gimbal to measure the height of terrain forward of the moving vehicle. The gimbal obviated the need to incorporate inertial measurement unit (IMU) information in the calculation of Cartesian coordinates from the laser ranges, however, the gimbal is something of a "black box" and the authors did not appear to dedicate any effort to calibrating the laser's position with respect to the vehicle's body frame. Cartesian coordinates were calculated using knowledge of the laser's angle with respect to the horizon, and simple trigonometric identities.

The terrain-following algorithm developed by Clark and Roberts was fairly straightforward:

1. set a goal location, as a Cartesian coordinate (x_{goal}, y_{goal}, z_{goal});
2. set the target altitude (tf_height) to the z-coordinate of the goal;
3. take off and ascend to tf_height;
4. begin moving towards the goal;
5. at some interval, compute the Cartesian coordinate from the laser range and calculate the change in altitude (δz);
6. derive an intermediate goal from step 5 and move towards it; and
7. repeat steps 5-7 until goal is reached.

Livshitz and Idan (2018, 2019) modelled a terrain-following system using a conceptual laser rangefinder and a terrain generated from a second order Markov process. Unlike Clark and Roberts, Livshitz and Idan used a fixed-wing aircraft model and

one or more rigidly-mounted rangefinders. These authors gave careful consideration to the types of errors that could arise and their overall effect on the outcome, specifically, the static error in the measured angle of the laser with respect to the vehicle's body frame and the stochastic error inherent in the vehicle's inertial measurements (i.e., position and orientation). Under this model, the "virtual path" was produced by sampling the Markov terrain and smoothing it within a window, given predetermined error bounds. From the smoothed output, navigable waypoints were extracted.

The A* ("A-star") (Hart et al., 1968), genetic (Li Qing, 1997), evolutionary (Nikolos et al., 2007), fuzzy (Bagherian, 2018; Rahim & Móhammad-Bagher Malaek, 2011) and gradient descent (Asseo, 1988) algorithms and neural networks (Artale et al., 2016; Kosari et al., 2015; Waldock, 1995, etc.) have been applied to the problem of robot and RPAS route planning. All incorporate the notion a cost function and iterate towards a locally- (e.g., gradient descent) or globally-optimal (e.g., dynamic programming) solution. It would be a simple matter to incorporate the constraints of interest in this research into such a cost function, however these solutions are computationally-intensive and some are not guaranteed to converge on an optimal solution deterministically or in a finite time. Most assume the pre-existence of a complete terrain model or pre-flight, offline computation of the trajectory.

Interesting work on autonomous navigation has been conducted in the automated underwater vehicle (AUV) field. Waldock (1995) used an artificial neural network (ANN) to control an AUV using sensors in a similar arrangement proposed for this research. Of all the research cited in this paper, Bovio et al.'s (2006) comes nearest to anticipating its objectives. Bovio et al. (2006) discussed methods for following submarine terrain and dealing with high-frequency terrain noise, expanded on the use of inertial navigation system (INS) and position sensing, encountered the deleterious effects of vegetation on height measurement and addressed model-based and

model-independent control systems. The last item in particular is of interest: it was noted that model-based approaches to vehicle control are problematic due to the difficulty of characterizing the vehicle and the effects of changes to the payload. Model-independent control solutions, such as the PID controller, require no knowledge of the of the physical behaviour of the craft and respond only to deviations from a desired state. Jalving (1994) also explored the use of PID controllers for AUVs and Bovio et al. (2006), Feijun Song and Smith (2000), Goheen and Jefferys (1990), Healey and Lienard (1993), Juul et al. (1994) undertook studies of model-independent alternatives to the PID controller for AUVs.

Much of this recent research into TF/TA seems to have been conducted with an interest in machine learning (ML) and aritificial intelligence (AI), and in navigating dynamic, three-dimensional environments in a way that replicates the navigational strategies of living creatures. Previous military-oriented and AUV research took a more directed approach, finding two-dimensional paths over complex 2.5-dimensional terrains which would not exceed the pilot's or aircraft's capabilities. In part, this may be due to the speed and scale of military operations, or merely due to the limitations of technology at that time, but the military approach may have more relevance to this study than more recent RPAS-focused studies.

2.3 Dynamics and Kinematics

Many aspects of vehicle dynamics are unique to multi-rotor RPAS. In particular, unlike a single-rotor helicopter, the rotors' torque forces cancel out, obviating the need for a tail rotor, and the centre of mass is suspended between the rotors, rather than in line with the axis of rotation (McKerrow, 2004). A multi-copter is said to be an under-actuated system (McKerrow, 2004; Mian & Wang, 2008; Valavanis, 2007)

meaning that, though the vehicle has six degrees of freedom (rotation and translation along three axes), it has only four possible control inputs: roll, pitch, thrust and yaw. For this reason a multi-rotor RPAS tends to be unstable: unlike a helicopter, it is virtually impossible for a human pilot to control a multi-copter without the assistance of a computer. The advent of small, fast microprocessors, electro-mechanical control systems and sensors (e.g., gyroscopes and accelerometers) has made the entire RPAS remote sensing field possible (Hoffmann et al., 2007).

A multi-rotor performs all movements by controlling the differential thrust and torque of its propellers using throttle adjustments; it has no control surfaces and behaves like a single thrust vector, oriented around its centre of mass but not necessarily aligned with the direction of travel unless climbing vertically. To increase horizontal velocity in level flight, a multi-rotor must rotate the thrust vector and modulate the amount of thrust, so that the vertical component is sufficient to maintain its rate of climb (zero, in level flight), and the horizontal component is sufficient to cause a horizontal movement at the desired velocity in the correct direction.

One can perform a very basic analysis using a much-simplified model of the vehicle dynamics – a more comprehensive treatment can be found in Gibiansky (2012), McKerrow (2004), Valavanis and Vachtsevanos (2015, etc.): the vehicle is imagined as a single thrust vector with all propellers either aligned with the axis of thrust or in such a way that the non-vertical components of their respective thrust vectors cancel out. Thrust imparts an acceleration to a mass along the thrust vector according to,

$$F = ma. \tag{2.12}$$

To climb, the flight controller supplies sufficient thrust to overcome the acceleration of gravity, which, via its interaction with the mass of the vehicle, presents an opposing force. With thrust force equal to the gravitational force (providing the vehicle has

sufficient elevation to avoid ground effects), the vehicle hovers at a constant altitude.

The thrust vector of a hovering multi-rotor has a non-zero vertical component and an (ideally) zero horizontal component. To initiate a horizontal movement, the thrust vector must rotate, which redistributes some of the total thrust to the horizontal component. This reduces the magnitude of the vertical component, requiring that the total thrust magnitude – i.e., the engine revolutions per minute (RPM) – be increased to compensate. The relationship between the horizontal and vertical components of thrust (T) in two dimensions is,

$$\begin{aligned} T_{vertical} &= T_{total} * sin(\theta); \\ T_{horizontal} &= T_{total} * cos(\theta), \end{aligned} \tag{2.13}$$

where θ is the rotation from vertical. Then the thrust vector need only be scaled so that the vertical component is equal to the magnitude of the total thrust at hover.

The transition from level flight to climb requires that the vertical component of thrust increase, and the constant horizontal velocity constraint requires that the horizontal component remain static. As these quantities are related to platform rotation by a trigonometric identity, it is clear that a change in the rate of climb cannot be achieved with a constant orientation. The only option is to minimize the amount of rotation by minimizing the second derivative of the trajectory, which determines the amount of vertical thrust. Not coincidentally, this is the subject of Funk's (1976) "hard" versus "soft" discussion.

2.4 Control Theory

Much of the early research into terrain following assumes a "bang-bang" approach to throttle control – the throttle is either on or off. In the present circumstances,

low-level inputs (such as individual throttle control) will be actuated implicitly by sending higher-level inputs to the controller (linear or angular velocity), and letting the controller calculate the appropriate throttle response. Given a target velocity, the controller will use the maximum available thrust to achieve it in the shortest time. A multi-rotor aircraft vectors this thrust by rotating in space, perhaps excessively, in the context of a remote sensing mission where stability is more desirable than the speed of response.

Consider a vehicle approaching a turn at $5\,\mathrm{ms}^{-1}$: if the controller simply changes the velocity vector to a new direction, the vehicle will overshoot the turn point and may also overshoot its new orientation as it revolves. If the vehicle is turning into a long flight line, the overshoot will move the starting point of the line, which will be at variance with the flight plan. This could induce problems with flight line overlap, or gaps in the imagery. A mechanism for returning the vehicle to its target state is required.

PID controllers (e.g., Sánchez et al., 2012) are widely used for controlling industrial processes. In general, the PID controller takes as input the current state of a system, and the desired state of the system (the "set point"). In the present instance, the set point might be the target altitude and heading and the system state the current altitude and heading, respectively. The controller has no knowledge of the dynamics of the system (e.g., the mass of the platform) and is instead tuned using a trio of gain parameters: the P in PID stands for "proportional", the I for "integral" and the D for "derivative". Any of the gains can be set to zero for a P, PI or PD controller.

The equation of the controller is,

$$u(t) = K_d e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \tag{2.14}$$

where K_d, K_i and K_d are the gains, e is the error in the system at time, t (i.e., the

difference between the set point and the current state) and u is the control output. In effect, the proportional term moves the system towards the set point, the integral term provides damping (it becomes negative if the state overshoots the set point) and the derivative term provides insight into the error trend, reducing the effect of the other terms as the state approaches the set point.

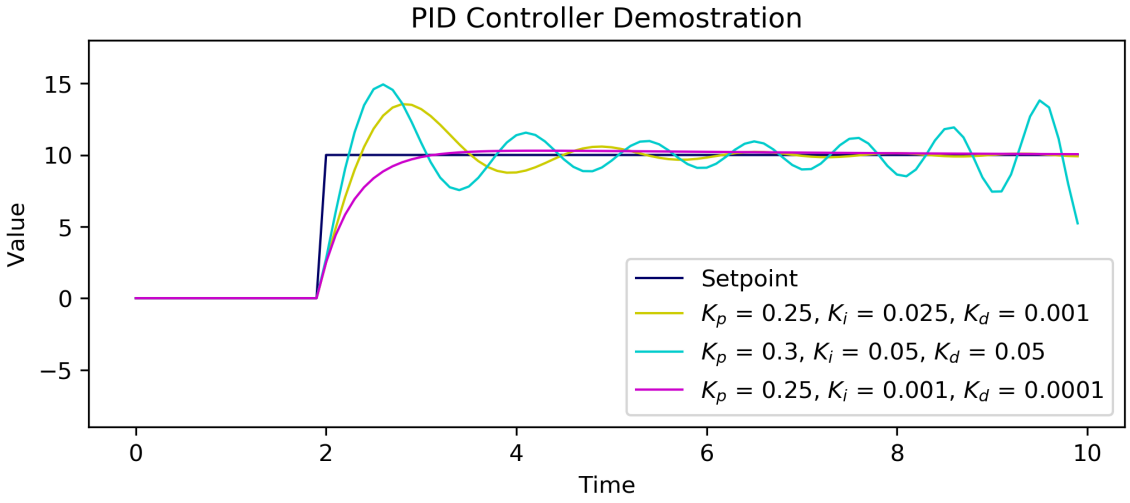


Figure 2.6: Action of PID controllers with different tunings.

Figure 2.6, shows three PID functions approaching the same set point. The first (yellow) line shows a rapid acceleration and overshoot, followed by a decaying oscillation towards the set point. In the last (fuchsia) line, the overshoot and oscillation is well-controlled. Note that the steep increase at the beginning of each adjustment is misleading. For a physical body, when a throttle adjustment occurs, the response of the body is delayed due to inertia in both the propulsion system and the aircraft itself.

The cyan line exhibits a phenomenon called "integral wind-up" in which the integral term accumulates rapidly, inducing a complete loss of control. This can be controlled by using a small gain or by limiting the time span over which the integral is calculated. Integral wind-up could also be controlled by setting the integral gain to

zero, implying a P or PD controller, however the action of a proportional controller is asymptotic – it never actually reaches the set point and must be "nudged" by one of the other terms.

2.5 Surface Extraction

The definition of a "surface" is problematic: As Mandelbrot famously observed 1967, the length of the coastline of Britain depends on the length of the ruler – the coastline (or, by analogy, a surface) is fractal. For this application, it is important to recognize that increasing the density of measurements, and thus the level of surface detail, does not necessarily provide more useful information about the surface at a given scale of interest. However, decreasing the density of measurements may obscure extraneous detail to reveal the "true" form of the surface – in other words, to extract the signal from the noise.

The nature of terrain surfaces can vary: non-vegetated terrains such as soil or pavement are non-porous and every LiDAR point will represent an upper surface, while vegetated terrains are porous and many points will represent objects within the canopy and of no interest for navigation. The purpose of surface extraction is twofold: to reduce the point count, thereby reducing the computational load on the trajectory-extraction function, and to isolate the upper surface of a potentially-deep point cloud. The ultimate goal is to find a smooth, differentiable function representation of the surface signal by fitting the function to the surface point set, so the surface extraction must simultaneously remove extraneous data and preserve enough data to represent the signal. There are several strategies for this, in two broad categories: geometric and statistical.

Berger et al. (2017) provide a comprehensive survey of three-dimensional surface

reconstruction methods based on point clouds, but these are intended for highly-detailed representations of physical objects, use *a priori* point sets and amount to overkill. A two-dimensional, online, space- and time-efficient solution would be preferred.

Convex Hull

A basic representation of shape is given by the convex hull – the unique set that encloses all segments joining each pair of points in the point set or, put another way, the hull that contains all constituent hulls (de Berg et al., 2008). The convex hull is well defined: for any given point set, there is exactly one convex hull, but of course the convex hull is always convex – a definition of shape that allows for natural indentations in the surface would be preferred.

α-shapes

The α-shape (Edelsbrunner & Mücke, 1994) is a generalization of the convex hull, which is best understood using the "ice cream" metaphor (Da et al., 2018): if one imagines the point cloud as chocolate chips embedded in ice cream, one can scoop out the ice cream between the chips leaving behind a shape. The shape of the remaining ice cream is determined by the squared radius (α) of the scoop. An $\alpha = \infty$ scoop (a plane) will leave the convex hull, while an $\alpha = 0$ scoop will leave only the chocolate chips (the original point set). For any given α there is exactly one solution, but α itself is bounded by $[0 - \infty]$.

The α-shape construction may be valuable for this case as it offers a parameter that satisfies the requirement that pits or holes in the terrain of a certain size may be closed or filled in, and the algorithm is implemented using the Delaunay triangulation which can theoretically be computed incrementally on a point stream. However, the

worst-case run time [2] is $O(n^2)$ (Edelsbrunner & Mücke, 1994).

There have been other attempts to reconstruct shapes from point clouds, for example, the α-concave hull (Asaeedi et al., 2017), the crust and β-skeleton (Amenta et al., 1998) and the χ-shape (Duckham et al., 2008), all of which run in $O(nlogn)$ time. There are simpler, more efficient ways to extracting a surface in real time.

"Concave" Hull

The "concave" hull is a modifification of an existing convex hull algorithm to allow indentations, though unlike the convex hull there is no definition or proof of correctness. Andrew's (1979) algorithm runs in linear ($O(n)$) time on sorted input. Since in the present case, most of the point cloud is likely to remain static with updates concentrated at one end, the sorting operation – e.g., Quicksort at $O(nlogn)$ (Hoare, 1961) – is presumably confined to a small subset of the input. Limiting the segment length of the convex hull and confining it to the upper half of the point set should produce a concave shape in near-linear time, though the generality of this solution is not proven. Of course, the previous solutions operate in two or more dimensions while this solution operates only in two.

Binning

An alternative to geometric solutions is to perform a statistical surface extraction, either by binning the point set or by performing a proximity search and and extracting statistics from the result. Proximity searches using a k-d tree (Bentley, 1975) entail an $O(logn)$ time cost for insertion and search, but enable k-nearest neighbours searching

[2] "Big O" notation is a convention in computer science used to denote the time- or space-efficiency of an algorithm in relation to the size of the input. For example, $O(1)$ is "constant" or independent of the input size, $O(n)$ is proportional to the input size and $O(n^2)$ is a quadratic function of the input size.

which eliminates the problem of voids in the point set: in instances where the point cloud is sparse, it is possible to locate neighbours at any distance in reasonable time.

Binning is more efficient than any geometric or tree-based solution. A coordinate is simply truncated to the desired precision, placing the point implicitly in a bin. A surface can be extracted by this strategy in $O(n)$ time. While the potential biases introduced by bin selection are well-known in physics (e.g., Krislock & Krislock, 2014; Lougovski & Pooser, 2014), in the terrain-avoidance context the bias could prove beneficial: as points are placed in the bin, they are dissociated from their horizontal components and assigned those of the bin itself. The vertical component of the bin becomes that of the highest point entered.

2.6 Trajectory Functions

The notion of terrain following as a signal processing exercise appears only rarely in the terrain following literature (e.g., Starling & Stewart, 1971), and the question of whether terrain does constitute a signal is contested. Chakravorty (2018) argues that, to constitute a signal, a phenomenon must not be static in all dimensions: a two-dimensional image is not a signal unless the viewer traverses it in a specified direction. By analogy, terrain is the image of a function in x and y and is static on time scales relevant to this research. However, since an aircraft moves over time, the terrain is a function of time and is therefore a signal.

Since terrain forms a closed surface, it necessarily has both global and local means, and due to physical factors, such as uplift and erosion, it exhibits periodicity on multiple scales (Hanley, 1977; Perron et al., 2008; Shaler, 1899, etc.). Living matter also exists in periodic arrangements on the landscape. Vineyards (above) are planted in rows at a single frequency, while natural forest canopies exhibit a multi-spectral

signature (Couteron et al., 2005). If the Earth's surface can be understood as a complex signal, a signal-processing approach is appropriate for the extraction of a trajectory function.

According to the goals of this research, from the trajectory function it must be possible to ascertain in flight that:

1. the vehicle has sufficient thrust to navigate the trajectory;
2. the trajectory is smooth and differentiable;
3. the trajectory minimizes geometric distortions due to variances between the aircraft altitude and the terrain; and
4. the fidelity between the trajectory and the surface (e.g., in the least-squares sense) can be ascertained.

The following sections investigate several methods for extracting the terrain signal.

Cubic Splines

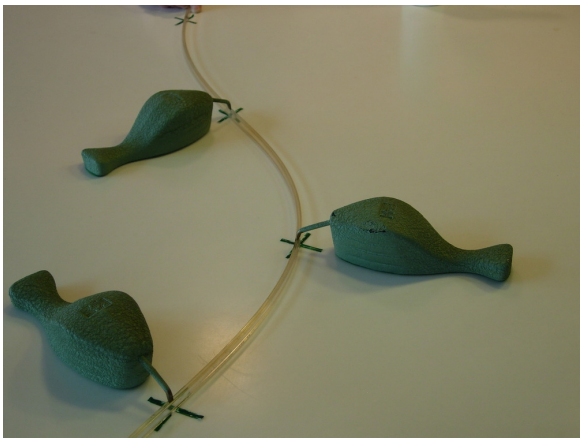


Figure 2.7: A draftsman's spline and "ducks" (de Boor, 2006).

Funk's (1976) TF/TA solution addresses items 1 and 2 using cubic splines. The word "spline" originates in drafting, where long, flexible wooden or plastic beams, called splines (figure 2.7), were used to model curves. A spline would be placed on the drawing, held down by two or more lead weights called "ducks," and would seek a shape which minimized the energy stored in the beam. The spline

could thus be considered an optimal interpolator (Wegman & Wright, 1983).

In mathematics, a spline consists of a set of piecewise (usually cubic) polynomials, one each between each pair of ducks, or "knots." At each knot, the zeroth, first, and possibly higher, derivatives of the adjoining polynomials are held equal, resulting an a curve that is smooth and differentiable along its entire length. The minimum-energy constraint is modelled by minimizing mean square curvature of the polynomial (Wegman & Wright, 1983).

In this instance, the spline function is used to model the behaviour of a physical body in space, with velocity and acceleration given by the first and second derivatives, respectively. The second derivative is of particular interest because it is constrained by the aircraft's dynamic properties given by Newton's second law (2.12) where F is force in Newtons, m is the mass of the vehicle in kg, and a is the acceleration in ms^{-2}. Forces on the air frame are generated by the propulsion system, gravity and aerodynamic drag. When the second derivative is positive, its magnitude can be no greater than the acceleration provided by the net of these forces (climbing under maximum thrust). When negative, its magnitude can be no greater than net of these forces with thrust set to near zero (free fall). These constraints may be derived from the aircraft's thrust, as documented by the manufacturer, and its mass and air resistance, which must be measured. The maximum vertical velocity of most RPAS is documented by the manufacturer and limited by the flight controller. Whether the vehicle can achieve the velocity limit depends on its mass (including cargo) and the force supplied by its propulsion system. (The flight controller cannot control thrust directly – only by adjusting the electrical power supplied to the motors – but it can measure the resulting changes to velocity and acceleration directly using its built-in accelerometers.)

Though cubic splines are smooth and twice differentiable, the second derivative (acceleration) is not smooth, having cusps at the knots. For any polynomial, for some value of $m > 0$, the mth derivative will be linear; for cubic splines, this occurs at $m = 2$. This is not a concern, so long as the change of thrust by the aircraft's flight controller is assumed to occur instantaneously. It does not, of course, as it takes time to supply the motors with power, and for their rotational velocity to change. Here, the maximum thrust, or some multiple less than 1 could be used as an estimate of the limit.

The mechanical spline and its mathematical equivalent are interpolators and pass through each of the knots exactly. If the knots of the spline are close together, or form a steep gradient, un-resolvable loops or overshoots could form in the spline, resulting in a physically-impossible trajectory. A variation on the interpolating spline, the weighted spline, allows the calculation of a weight, applied at each knot, which can suppress the tendency for overshoots (Lancaster & Šalkauskas, 1986). However this breaks the continuity of the second derivative, which in turn breaks the physical model.

The cubic spline does not accommodate the conception of terrain as a signal: as an interpolator, it over-fits the data implicitly and cannot extract a generalized terrain function. An alternative construction – the smoothing or approximating spline – allows the spline to recover the terrain function without passing through each datum exactly.

Smoothing Spline

A smoothing spline has two conflicting objectives (de Boor, 2001; Drakos & Moore, 2002; Lancaster & Šalkauskas, 1986; Reinsch, 1967):

1. to minimize the sum of squared distances between the function estimate and

the ordinates, and

2. to minimize the curvature of the function.

This leads to de Boor's formulation (1978),

$$S(x) = p\sum_{i=1}^{n}\left(\frac{(Y_i - \hat{f}(x_i))^2}{\sigma_i^2}\right) + (1-p)\int\left(\hat{f}^{(m)}(x)\right)^2 dx, \tag{2.15}$$

where the first term is essentially a χ^2 measure [3] of the data distribution and the second a measure of its curvature, both weighted by $p \in [0-1]$ or its negation. Here, Y is the ordinate, $\hat{f}$ is the spline estimate and σ represents the variance of the ordinate set. p has the effect of controlling the number of knots: as p approaches 1, the first term is emphasized, forcing a closer fit to the data, while as p approaches 0, the integral is emphasized, forcing a straighter (smoother) curve. As the fit improves, the number of knots and their positions approach those of the ordinates; as smoothness increases, a straight line emerges with two knots at the ends.

The system is constrained, such that,

$$\begin{aligned} S(x_{i-1}) &= S(x_i), \\ S'(x_{i-1}) &= S'(x_i), and \\ S''(x_{i-1}) &= S''(x_i). \end{aligned} \tag{2.16}$$

At each knot, the ends of adjoining splines have not only the same value, but the same first (slope or velocity) and second (curvature or acceleration) derivatives. In the case of the so-called "natural spline", the second derivatives at the first and last knots are held to zero.

It remains to develop a methodology for selecting weight and smoothness parameters appropriate to these constraints, given the available instrumentation and

[3] χ^2 or "chi-squared" is a fundamental statistical measure of goodness of fit (Pearson, 1900).

computational resources. The Fitpack library documentation, provided in inline comments (Dierckx, 1979), suggests a method for selecting the smoothing parameter and calculating per-ordinate weights. The former should be in the range,

$$m - \sqrt{2m} \leq s \leq m + \sqrt{2m}, \tag{2.17}$$

where s is the smoothing factor and m is the number of input points. This range is predicated on the assumption that the weights are given as,

$$\frac{1}{\sigma}, \tag{2.18}$$

where σ is the standard deviation of the elevation values. Throughout the documentation for libraries by both de Boor (1978) and Dierckx (1979), it is recommended that the user assess the results graphically, as parameter selection is an inexact science.

Convolution

Convolution is a method frequently used in signal processing, by which a function – the "kernel" – is passed over a signal to produce a new signal. The kernel function, depending on its shape, can serve a number of purposes. The simplest of these is the moving average, frequently used to smooth signals – that is, to eliminate high-frequency noise or variation (Orfanidis, 1996).

In this scheme, values within the neighbourhood of a location of interest – usually in terms of time, τ – are weighted and summed to produce a new value at that location. In the case of the moving average filter, each weight is the reiprocal of n, where n is the number of elements covered by the kernel. The smoothed values are assumed to represent the function underlying the data. The moving average function can be visualised as a rectangular curve, often called the "boxcar" function (Orfanidis,

1996). For a five-element kernel, the estimate $\hat{y}$ at location 0 is given by,

$$\hat{y}_0 = \sum_{i=-2}^{2} y_i * 0.2. \tag{2.19}$$

Another form is the distance-weighted, or triangular, kernel. Each datum in the neighbourhood of the position of interest contributes a share of its value relative to its proximity to that position:

$$\hat{y}_0 = \sum_{i=-2}^{2} y_i * |x_i - x_0|/2. \tag{2.20}$$

The Gaussian smoothing function can also be represented as a kernel:

$$\hat{y}_0 = \sum_{i=-2}^{2} y_i * \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}((x_i - x_0)/\sigma)^2}. \tag{2.21}$$

Collectively, these filters are known as finite impulse response (FIR) filters (Orfanidis, 1996).

The Savtizky-Golay Filter

Convolution is a credible means of extracting the terrain function from the point set, but what can be done about the removal of important peaks, or the obedience to dynamic constraints? How does one determine the maximum slope or curvature of the convolved signal?

The Savitzky-Golay filter (Savitzky & Golay, 1964) fits – in the least-squares sense – a polynomial curve to a data subset centred on a specific point. However, rather than fitting the curve at each point by an expensive linear solution, the SG filter pre-computes a matrix of coefficients which are applied to the neighbourhood of every point in the data set (excluding the ends, which require a special set of

coefficients). This enables not only the smoothing of the data set but the computation of derivatives, exactly as efficiently as the moving average. (In fact, the moving average filter can be implemented using the SG technique – the polynomial is merely a horizontal straight line, i.e., a constant.)

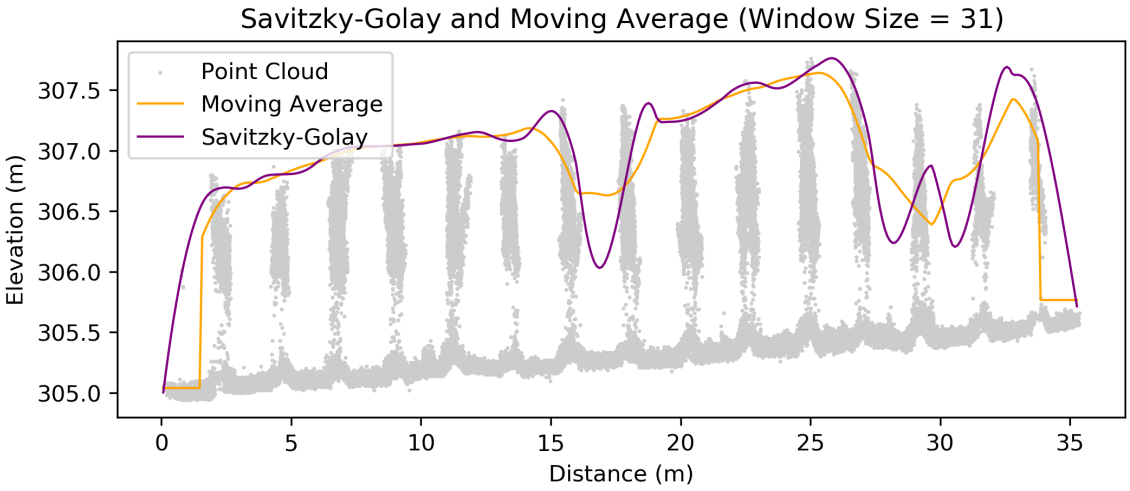
Tables of coefficients – including errors that were later corrected – were given by Savitzky and Golay (1964). The only necessary parameters are the window size, the degree of the smoothing polynomial, the order of the derivative and the distance between data points, which must be evenly spaced.

Like other smoothers, the SG filter tends to suppress high-frequency noise while preserving the shape of the signal, including its peaks and valleys (Thornley, 2006) (figure 2.8a). Unlike other smoothers, which tend to reduce the extrema of a signal and increase its half-width, the SG filter better maintains these properties. This comes at the expense of some noise-suppression ability (Orfanidis, 1996) but for this application, the preservation of peaks is desirable.

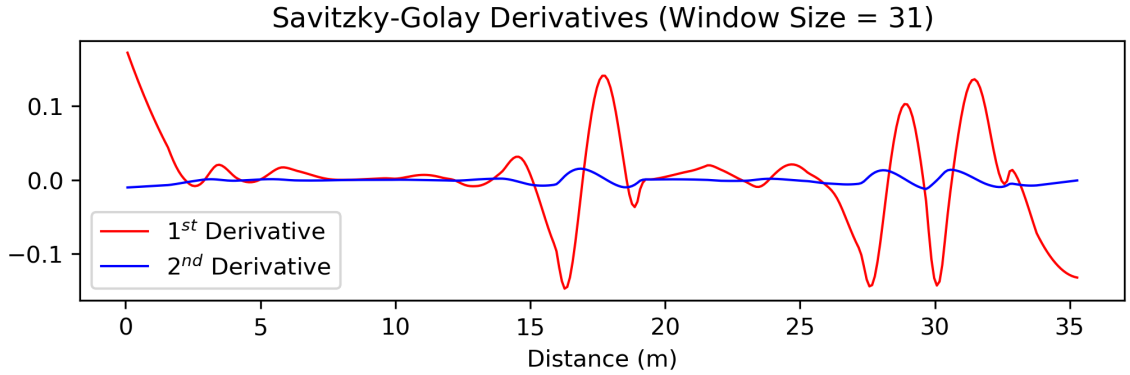
The Savitzky-Golay filter consists of the function,

$$\hat{y}_i = \frac{\sum_{j=0}^{k_{size}} k_j (y_{i-\frac{k_{size}}{2}+j})}{n}, \tag{2.22}$$

where i is the index into the data set, k_{size} is the kernel window size, k_j is the kernel coefficient at j and n is a normalization factor. Partial tables for the 0^{th}, 1^{st} and 2^{nd} derivatives are reproduced in tables 2.1, 2.2 and 2.3. A comparison between the moving average filter and a SG filter is given in figure 2.8a, along with the derivatives for the latter.



(a) Comparison of Savitzky-Golay and moving average filters.



(b) First and second derivatives of the SG filter from figure 2.8a.

Figure 2.8: Comparison of cubic Savitzky-Golay and moving average filters on a real-world point cloud. α is deliberately under-sized to emphasize peaks, which the SG filter preserves better than moving average.

	Window Size				
Index	13	11	9	7	5
-6	-11				
-5	0	-36			
-4	9	9	-21		
-3	16	44	14	-2	
-2	21	69	39	3	-3
-1	24	84	54	6	12
0	25	89	59	7	17
1	24	84	54	6	12
2	21	69	39	3	-3
3	16	44	14	-2	
4	9	9	-21		
5	0	-36			
6	-11				
Norm.	143	429	231	21	35

Table 2.1: Savitzky-Golay cubic smoothing coefficients for window sizes from 5 to 13.

	Window Size				
Index	13	11	9	7	5
-6	1133				
-5	-660	300			
-4	-1578	-294	86		
-3	-1796	-532	-142	22	
-2	-1489	-503	-193	-67	1
-1	-832	-296	-126	-58	-8
0	0	0	0	0	0
1	832	296	129	58	8
2	1489	503	193	67	-1
3	1796	532	142	-22	
4	1578	294	86		
5	660	-300			
6	-1133				
Norm.	24024	5148	1188	252	12

Table 2.2: Savitzky-Golay 1_{st} derivative coefficients for cubic function, window sizes from 5 to 13.

	Window Size				
Index	13	11	9	7	5
-6	22				
-5	11	15			
-4	2	-6	28		
-3	-5	-1	7	5	
-2	-10	-6	-8	0	-2
-1	-13	-9	-17	-3	-1
0	-14	-10	-20	-4	-2
1	-13	-9	-17	-3	-1
2	-10	-6	-8	0	-2
3	-5	-1	7	5	
4	2	6	28		
5	11	15			
6	22				
Norm.	1001	429	462	42	7

Table 2.3: Savitzky-Golay 2_{nd} derivative coefficients for cubic function, window sizes from 5 to 13.

Chapter 3

Methods and Materials

3.1 Hardware

3.1.1 Aircraft

The airframe selected for this project was a DJI Matrice 600 hexacopter, a medium-sized six-rotor RPAS frequently used by the film industry for its reasonably large payload. The Matrice carries both satellite – global positioning system (GPS) and global navigation satellite system (GLONASS) – and inertial navigation sensors, as well as a three-axis magnetometer (compass) and barometric altimeter. The Matrice is controlled by the DJI A3 flight controller, which exposes a serial interface for bidirectional communication using one of several DJI-provided application programming interfaces (APIs). The APIs can provide high-frequency telemetric data and enable the control of the vehicle either by individual low-level commands (e.g., thrust, angular velocity), high-level commands (e.g., to climb to a given elevation) or complete missions, which are executed autonomously (DJI, 2018b).

The Matrice's ground station (hand-held controller) provides a variety of services in addition to the usual manual stick control, most importantly the ability to man-

ually interrupt an autonomous mission in progress. An Apple iPad is mounted on the controller to give access to configuration and status and to control the onboard camera. The iPad can also run third-party mission-planning software packages, such as Drones Made Easy, which uploads and executes waypoint missions from the iPad wirelessly via the controller.

Vehicle weight (TB47S batteries)	9.1 kg
Vehicle weight (TB48S batteries)	9.6 kg
Maximum vehicle weight	15.1 kg
Maximum pitch angle	25°
Maximum pitch velocity	$300\,^\circ\,\mathrm{s}^{-1}$
Maximum yaw velocity	$150\,^\circ\,\mathrm{s}^{-1}$
Maximum ascent velocity	$5\,\mathrm{m\,s}^{-1}$
Maximum descent velocity	$3\,\mathrm{m\,s}^{-1}$
Maximum horizontal velocity	$18\,\mathrm{m\,s}^{-1}$
Maximum wind velocity	$8\,\mathrm{m\,s}^{-1}$
Maximum thrust (per rotor)	5100 g

Table 3.1: DJI Matrice 600 Specifications (DJI, 2017).

Some specifications for the vehicle are given in table 3.1. These may not correspond to the absolute physical limits on vehicle performance, but rather to limits imposed by the flight controller. Still, they provide a useful guide to the vehicle's performance limitations. The nominal maximum thrust, T_n, generated by the DJI6100/DJI2170 motor/propeller combination is given in grams of pulling force per rotor. The propulsive forces of the six rotors cannot be summed directly because their axes are not aligned, however it is a simple matter to decompose the thrust vector of each into its T_x and T_z, components (T_z is vertical; T_y is not required) and sum the T_z components. The T_x components can be ignored because, being radially opposed, they cancel out.

Measured from CAD drawings (figure 3.1) supplied by DJI (DJI, 2017), each arm of the hexacopter is inclined upwards from horizontal by 8 deg, or ≈ 0.139 radians.

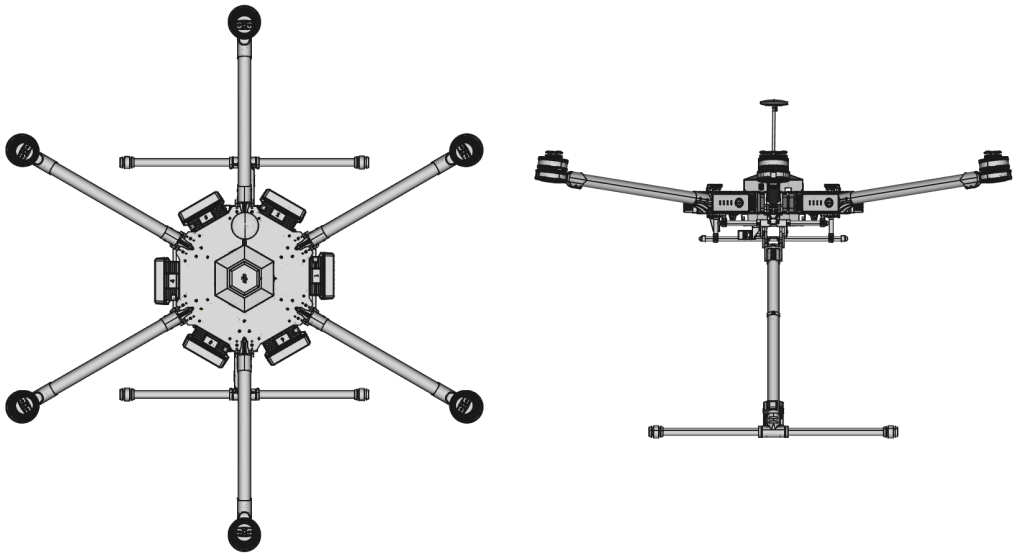


Figure 3.1: Top and side views of DJI Matrice 600 (DJI, 2017).

The T_z component is then,

$$\begin{aligned} T_z &= sin(\theta) * T_n \\ T_z &= sin(0.139643114) * 5.1 \\ T_z &= 0.990265734 * 5.1 \\ T_z &\approx 5.050\,\text{kg} \end{aligned} \tag{3.1}$$

This gives a total maximum thrust T_{max} of,

$$\begin{aligned} T_{max} &= 6 * T_z \\ T_{max} &\approx 30.302\,\text{kg}. \end{aligned} \tag{3.2}$$

The maximum available thrust as a force (F_t) in Newtons – assuming a vertical

orientation of the thrust vector and the acceleration of gravity as $g = 9.086\,65\,\mathrm{N}$ – is:

$$\begin{aligned} F_t &= T_{max} * g \\ F_t &= T_{max} * 9.08665 \\ F_t &\approx 297.162\,\mathrm{N}. \end{aligned} \tag{3.3}$$

At the maximum vehicle weight of $M_v = 15.1\,\mathrm{kg}$, this gives a maximum vertical linear acceleration of,

$$\begin{aligned} a &= (F_t - F_g - F_d)/M_v \\ a &= (297.162397521 - (9.80665 * 15.1) - 0)/15.1 \\ a &\approx 9.873\,\mathrm{ms}^{-2}, \end{aligned} \tag{3.4}$$

where F_g is the force of gravity and F_d is the force of drag – zero, under the assumption that vertical accelerations begin during level flight.

With the TB47S battery set, the hover time (at 10 m above sea level with no wind) is given as 35 min with no load and 16 min with the maximum 6 kg payload. With the TB48S batteries, the Matrice can hover for 40 min with no load and 18 min with a 5.5 kg payload (DJI, 2017).

3.1.2 Sensors

The purpose of the sensor package is twofold: first, to generate a point cloud forward of the vehicle as it flies along its trajectory; second, to measure, in real time, its current altitude above the surface. The DJI Matrice does have a barometric altimeter but a laser altimeter provides measurements with respect to the physical surface, rather than the isobaric surface, which is not of interest for remote sensing or navigation. The laser altimeter provides timely measurements for both navigation and validation

of the terrain-following system.

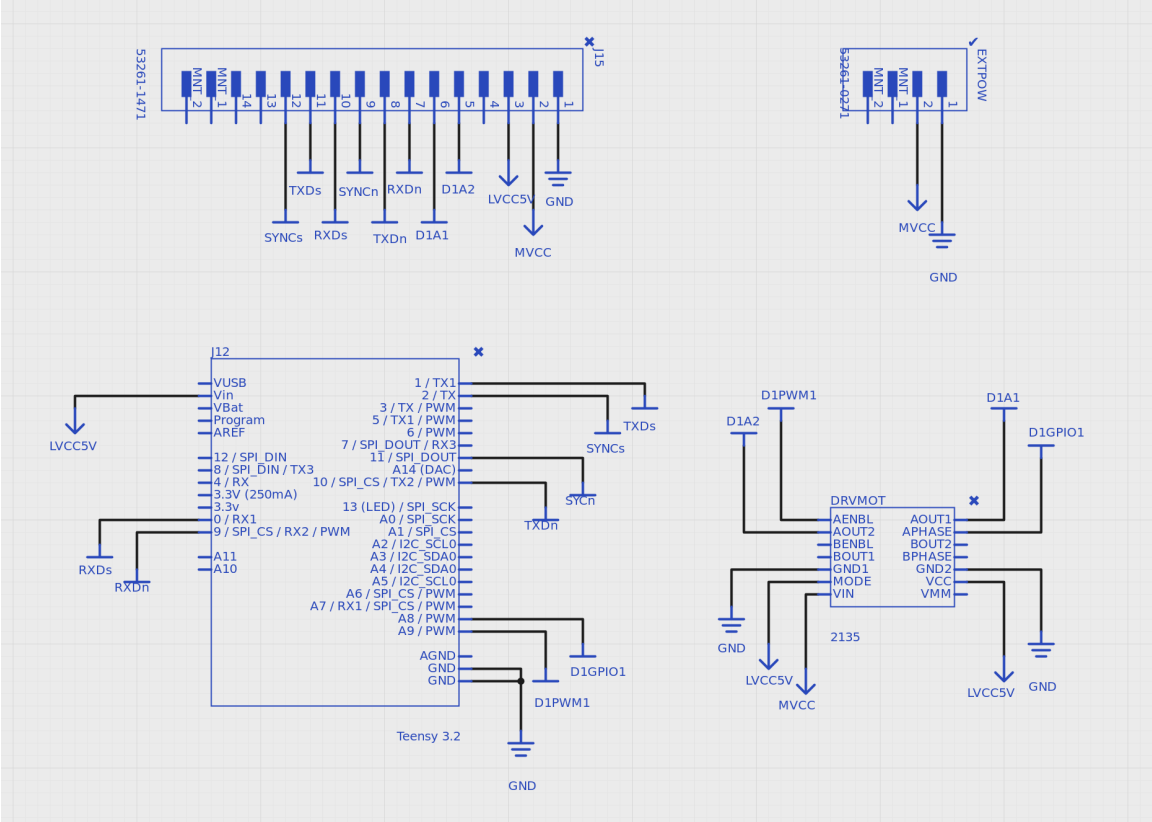


Figure 3.2: Circuit board schematic for Teensy, DRV8835 and connections.

The sensors and actuators are wired directly to a circuit board-mounted (figure 3.2) PJRC Teensy 3.2 microcomputer, featuring an ARM Cortex-M4 processor and many multi-purpose input/output ports which can be reconfigured to handle hardware interrupts, serial communication, analog-to-digital conversion and pulse-width modulation. The Teensy has a micro universal serial bus (USB) port through which it communicates with the Raspberry Pi flight controller. The Teensy is programmed using C or Arduino code; Arduino was selected in this case due to the availability of an easy-to-use integrated development environment (IDE) and programming software provided by PJRC.

Both lasers are LightWare OptoElectronics SF30/C laser rangefinders. Specifica-

tions for this device are given in table 3.2. The lasers transmit raw range data to the Teensy via serial universal asynchronous receiver/transmitter (UART) lines. Before each pulse, the laser pulls a synchronization line low (i.e., reduces its voltage), which triggers a hardware interrupt in the Teensy. In a real-time (i.e., non-multitasking) system such as the Teensy, hardware interrupts are serviced immediately, rather than being shunted off into a de-prioritized thread of execution, as happens in a typical multitasking operating system. This ensures that the sync pulse is handled in a deterministically-timely manner. The interrupt-handling code records the clock time and associates it with the next pulse to arrive on the serial wire. The maximum over-clocked processor speed of the Teensy is 98 MHz, more than enough to accommodate the laser's maximum pulse frequency of 20 kHz. In practice, since the lasers are not used for remote sensing, the pulse frequency can be much lower; it is configurable from any computer with a USB port using a terminal application.

Laser wavelength	904 µm
Maximum range	> 100 m
Maximum pulse frequency	> 20 kHz
Peak laser power	20 W
Weight	35 g

Table 3.2: LightWare SF30/C rangefinder specifications (LightWare Optoelectronics, 2019).

The forward-facing SF30/C is mounted on a freely-revolving shaft which allows the laser to oscillate (figure 3.3). The shaft is driven by a four-bar linkage connected to a generic $2.5\,\text{V} - 7.5\,\text{V}$ geared, brushless direct current (DC) micromotor. The speed of the motor (and thus the scan frequency of the laser) is governed by pulse width modulation (PWM), which is hard-coded on the Teensy, and the degree of sweep is governed by the length of the driver link, which must be adjusted manually. The PWM signal is generated by the Teensy and sent to a Texas Instruments

DRV8835 dual H-bridge motor driver mounted on a breakout board manufactured by Pololu Corporation. The DRV8835 supplies external power to the motor and regulates its speed using the PWM signal. The scanning facility was added to ensure that the terrain-following system would be "conscious" of threatening objects or terrain features, not merely in a straight line ahead of the vehicle, but within a "safety zone" up to several metres wide.

The nadir-aligned SF30/C is mounted statically (figure 3.3) at manually-adjustable angle. The serial transmit and receive (TX and RX), 5 volt power, ground and sync lines of both lasers were soldered directly to the pins of the female header socket to which the Teensy was mounted. The Teensy packages the range and nadir measurements and a checksum as a binary stream which it sends over serial, via the USB port, to the Raspberry PI. The rangefinders produce range values in centimetres as unsigned short (16 bit) integers.

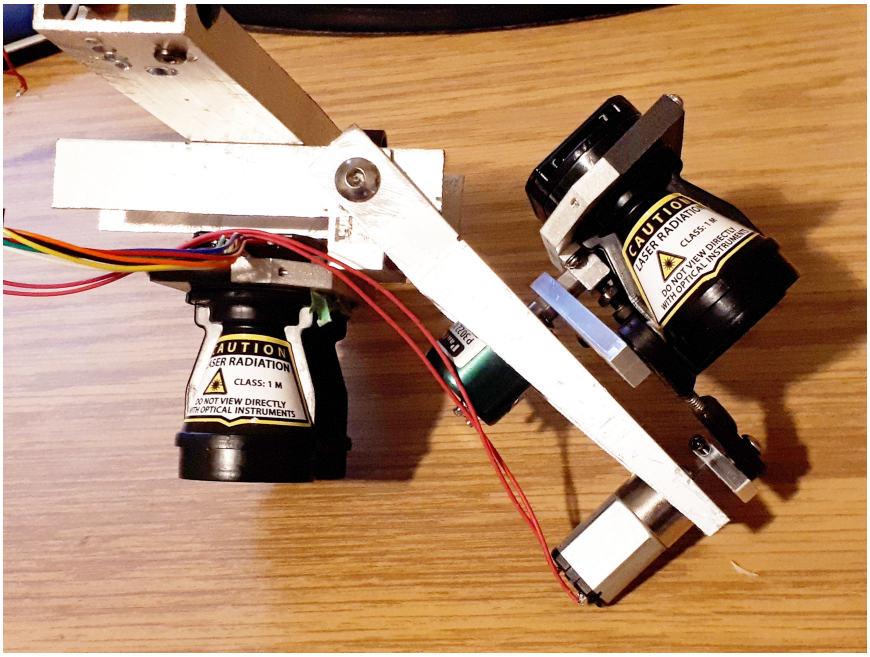
3.1.3 Flight Control Computer

A single Raspberry Pi 3 B+ (figure 3.3) multi-tasking computer is used to receive inputs from the sensor package and telemetry from the A3, generate the point cloud, calculate the trajectory and transmit course corrections to the DJI flight control API. The computer runs the standard Raspbian Buster Lite operating system. Raspbian is a Debian Linux derivative; the "lite" designation signifies the absence of user-interface and other unnecessary components.

The Raspberry Pi hosts numerous general-purpose input/output (GPIO) pins which can be assigned a variety of functions, including pulse width modulation (PWM) and serial input/output. In this application, general-purpose input/output (GPIO) pins are connected to the DJI A3's serial port for bi-directional transmission of telemetry and control messages.

(a) Raspberry Pi connected to DJI A3 by serial connection.

(b) Laser scanning mechanism.



(c) Side view of laser mounting bracket.

Figure 3.3: Flight control computer and laser rangefinders.

3.1.4 Power

The Raspberry Pi, Teensy, lasers and scanner motor are powered by one or more generic, 5 V batteries, which supply power separately to the Pi via a micro USB cord, and to the lasers and scanner motor by a modified USB cable. The Teensy is powered by the Pi through the serial micro USB cable. The Raspberry Pi can supply a limit of 1 A of continuous current to peripheral devices, enough for the Teensy, but not adequate to supply the motor and lasers together.

3.2 Algorithms

Surface reconstruction and trajectory extraction algorithms were developed from methods in the literature and tested on extant point clouds. These were produced from aerial LiDAR campaigns conducted in British Columbia by Hyperspectral-LiDAR Research Group (HLRG), over a variety of terrains, including a gently-sloping vineyard in the Okanagan Valley, British Columbia and a hilly, mixed Garry Oak and Douglas Fir landscape with rock outcrops in Victoria, British Columbia.

The first data set was created using a Velodyne VPL-16 LiDAR device mounted on a DJI Matrice 600 Pro, similar to the aircraft used for the present research. The second was collected using a piloted aircraft. The surface and trajectory extraction methods were applied in different combinations with a variety of parameterizations to assess their utility, and to develop a configuration strategy.

3.2.1 Surface Extraction

Concave Hull

The Mototone Chain convex hull algorithm (Andrew, 1979) runs in linear time on a sorted point set. By modifying the main loop of the algorithm to limit the length of

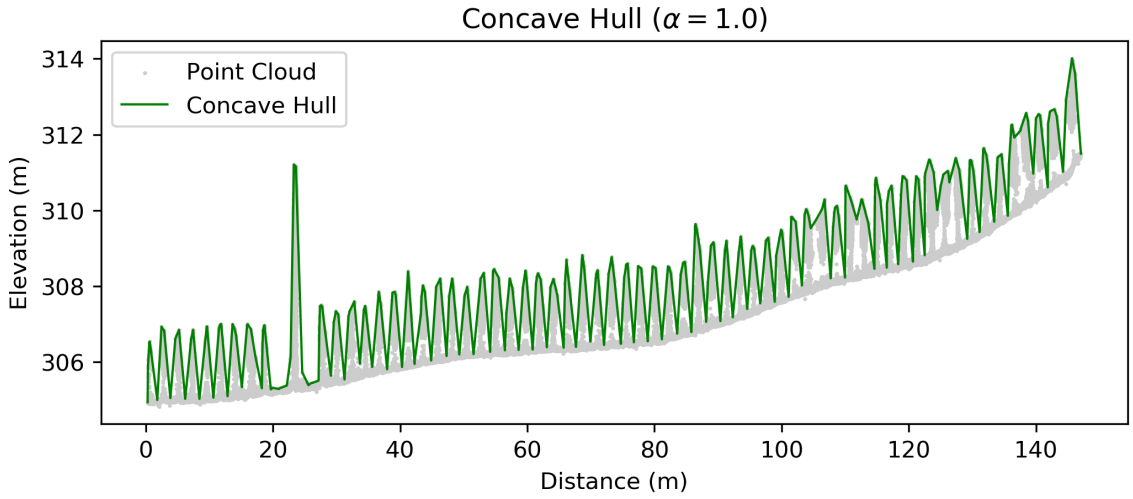
any given segment (via a parameter, α) and running it only on the upper surface of the input, it becomes a configurable surface-extraction tool. In this application the point set cannot be expected to be sorted, however on each iteration of the control loop, the point cloud can be replaced by the extracted surface and new measurements appended to it. The number of points will be much reduced on each iteration and only the newly-added points need be sorted.

Though the maximum segment length can be enforced, there can be no minimum, leading to possible clustering of vertices near highly curved, convex sections of the surface. The method is also asymmetrical: the hull computed from left to right would differ from one computed in the opposite direction. When a depression is encountered, a maximum-length segment will "reach" across the depression to the opposite side, then a cluster of vertices will occur as the algorithm "climbs" the convex slope of the far side.

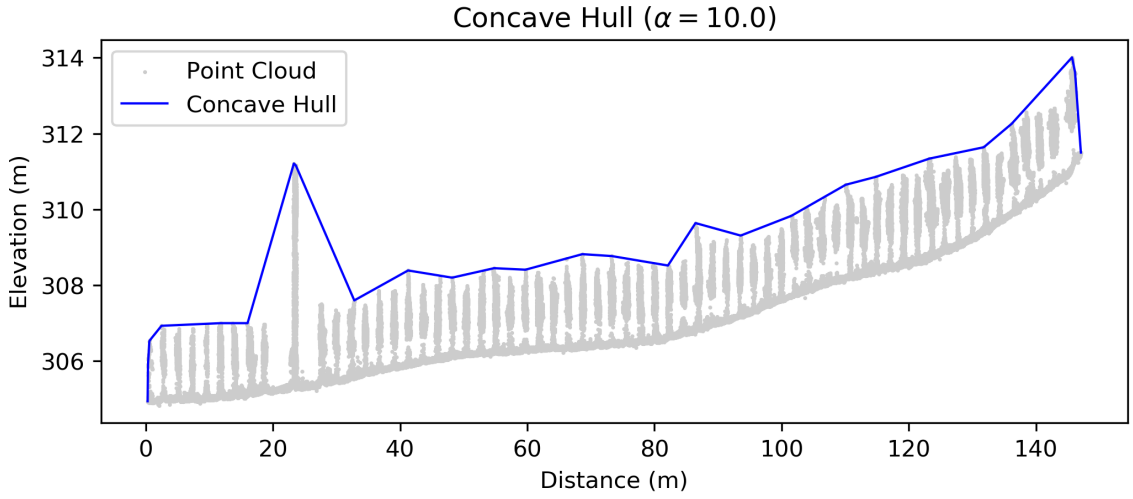
It is interesting to note that this behaviour imitates one of the limiting behaviours of the forward-facing rangefinder: as the vehicle passes over a transition from flat terrain to a down-slope, the terrain may pass beyond the laser's useful range, blinding the system (though in this case, the nadir-facing laser can still measure the vehicle's altitude). As the vehicle approaches the up-slope on far side of the depression, and the terrain comes within range again, the measurements will be clustered more tightly on the up-slope than on level ground.

A serious potential drawback to the segment-limiting scheme is that a horizontal gap in measurements greater than the maximum segment length – as could occur in the downslope scenario above – will cause the algorithm to fail. This can be resolved by iteratively increasing the segment length when this occurs, however a segment length longer than the laser's range will not correct the problem.

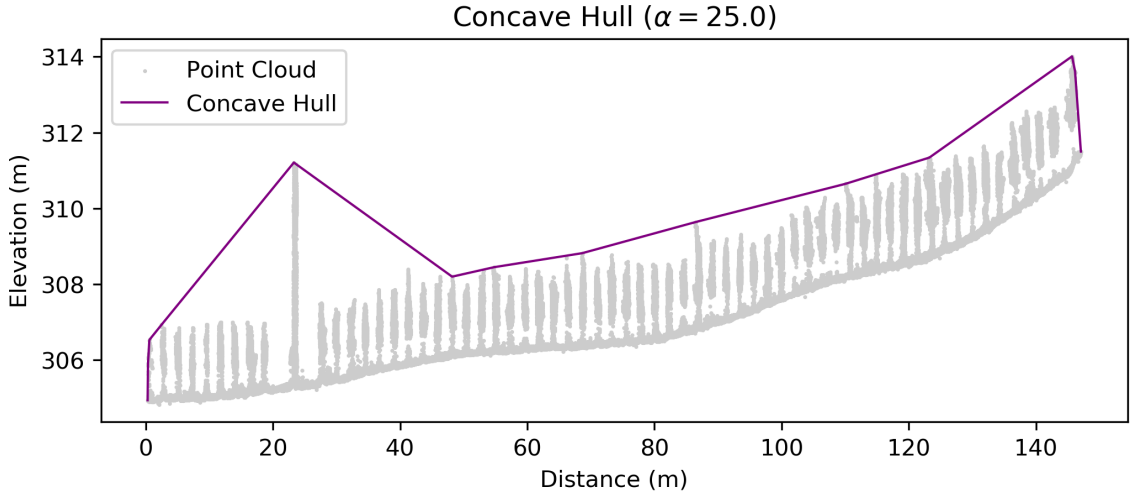
The vertices of the concave hull would form the point-set upon which the trajectory



(a) Concave hull surface extraction using $\alpha = 1.0\,\mathrm{m}$.



(b) Concave hull surface extraction using $\alpha = 10.0\,\mathrm{m}$.



(c) Concave hull surface extraction using $\alpha = 25.0\,\mathrm{m}$.

Figure 3.4: Concave hull surface extraction using different α values.

is computed. In general, a larger α value results in a smaller number of vertices and less surface detail. A Python implementation of the concave hull algorithm is given in listing 1.

```
def hull(pts, alpha):

        hull = pts[:2]

        for i in range(2, len(pts)):

                while len(hull) >= 2
                                and cross(hull[-2], hull[-1], pts[i]) >= 0
                                and lengthY(hull[-2], pts[i]) <= alpha ** 2:

                        hull.pop()

                hull.append(pts[i])

        return hull
```

Listing 1: Modified Monotone Chain algorithm for constructing a convex hull. The `cross` function determines whether the segment makes a clockwise or counterclockwise turn; `lengthY` gives the squared distance between points in y. The `pts` array is a list of points, sorted on y; `alpha` is the maximum segment length.

α-Shapes

A 2-dimensional α-shape algorithm resolves the issues with the modified convex hull algorithm, namely the problem of gaps in the point set, and the direction-dependent asymmetry. Both problems is resolved due to the fact that a contiguous Delaunay triangulation is constructed on the entire point set and eroded by the α parameter, leaving the structure intact.

α-shapes have the additional benefit that they can be computed efficiently in three dimensions, as well as in two, unlike the modified convex hull above. However, the necessity of computing a Delaunay triangulation on a potentially-large point set on each iteration of the control loop is discouraging. For this reason, α-shapes passed

over in favour of binning.

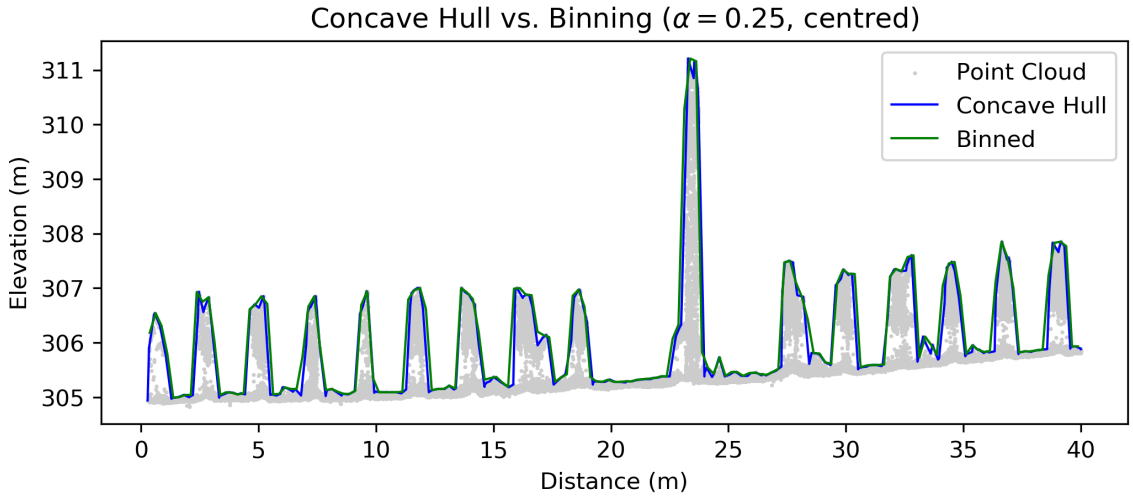
Binning

An alternative to geometric strategies for extracting a surface is to bin the point cloud horizontally and select, for each bin, the point with the maximum elevation. Optionally, a new point can be created in the center of the bin to produce an evenly distributed point set. Figure 3.5 shows the result.

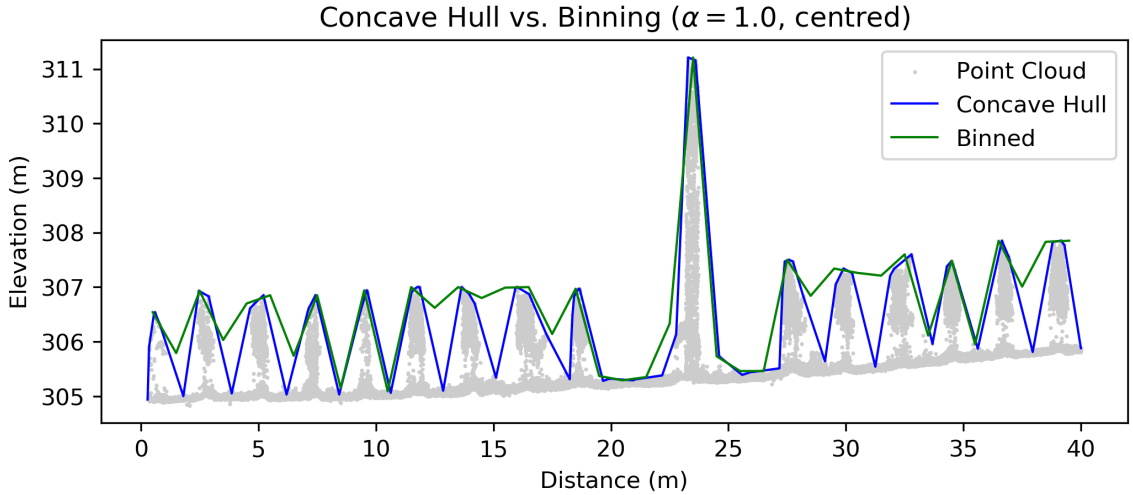
Point binning gives a worst-case performance of $O(n)$ in terms of both time and space. No sorting is performed, and on each iteration, only one point is preserved for each bin. Thus the amount of required space can be computed exactly given the mission parameters. Like α-shapes, binning works as well in three dimensions as in two.

There is cost to binning: because the maximum elevation in each bin is retained, the resulting surface may be biased upward. Theoretically, the bias tends towards zero as the bin size approaches zero though the efficiency gains of binning are also lost. In practice, the bias and efficiency concerns can be balanced by the selection of an appropriate bin size.

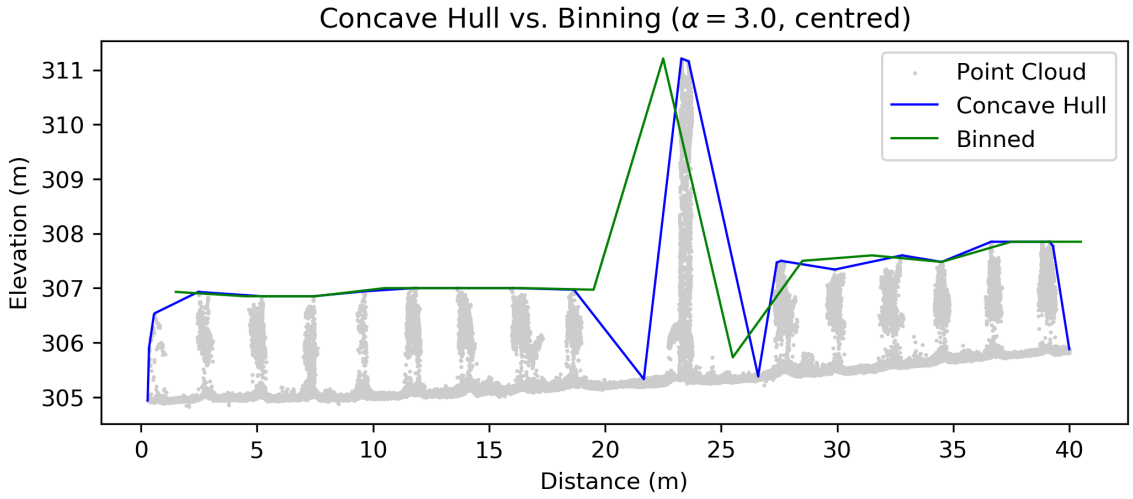
Figure 3.5 compares a binned surface to a concave hull, using a bin size, or α value, of 1 m. It is evident that the 1.0 m bin size selection results in a bin frequency that differs from the row frequency: at the 5^{th} and 6^{th} rows, the binned point occupies the bottom of the trench, while at other rows the bin is biased upwards by the contribution of points from the adjacent rows. In contrast, the concave hull accurately represents each row and trench in detail. The visual evidence is misleading, however: whereas binning creates data at a regular frequency (so long as there is data to be binned), the concave hull algorithm isn't constrained to regularity and therefore cannot be out of phase. The situation is resolved by selecting a smaller bin size (e.g., 0.25 m).



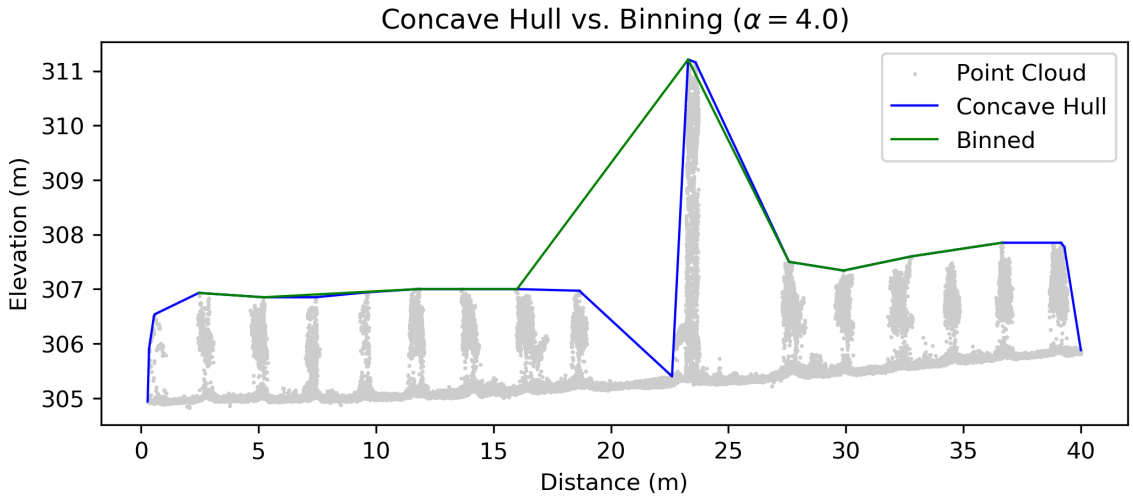
(a) Comparison of concave hull and binning with $\alpha = 0.25\,\mathrm{m}$ and maximum binned point centred.



(b) Comparison of concave hull and binning with $\alpha = 1.0\,\mathrm{m}$ and maximum binned point centred.



(c) Comparison of concave hull and binning with $\alpha = 3.0\,\text{m}$ and maximum binned point centred.



(d) Comparison of concave hull and binning with $\alpha = 4.0\,\text{m}$ and maximum binned point centred.

Figure 3.5: Comparison of concave hull and binning at different α values.

For practical use the α selection should be larger to increase the pass band of the filter. The row frequency in figure 3.5 – approximately 3.0 m – is both too high to be safely navigated by the vehicle, and not relevant to the remote sensing objective – to image the tops of the vines. A bin size of 3.0 m obscures the row signal while conforming to the terrain and maintaining the tall obstacle, yet still reaches to the bottom of one trough. Also, due to the centring of the coordinate, the top of the obstacle is offset. The 4.0 m bin size without centring resolves those issues. It is clear by the process of selecting a bin size that most of the work of filtering the terrain signal is performed at this stage, and that some care must be taken to balance the competing concerns of adequate detail and extraneous point removal.

3.2.2 Trajectory Functions

The trajectory algorithm extracts a smooth, differentiable signal function from the point cloud representation of the terrain. Two such methods were examined, the smoothing spline and the Savitzky-Golay filter.

Smoothing Splines

Most real-world uses of the smoothing spline call one of the Fortran libraries written by de Boor (de Boor, 1978) and Dierkx (Dierckx, 1993). All accept a smoothing parameter and a list of weights for each ordinate. SciPy's Python `UnivariateSpline` class, for example, calls into Dierkx's Fitpack library. The implementation in this project linked to the Fitpack library from C++, through a minimal C-style interface.

A single weight value may be selected for all ordinates, or a unique weight for each. This provides the opportunity to reduce the weight of outliers or points that would induce sharp bends in the spline. A naive strategy for computing weights is to compute the angle described by three adjacent points and assign a lower weight to

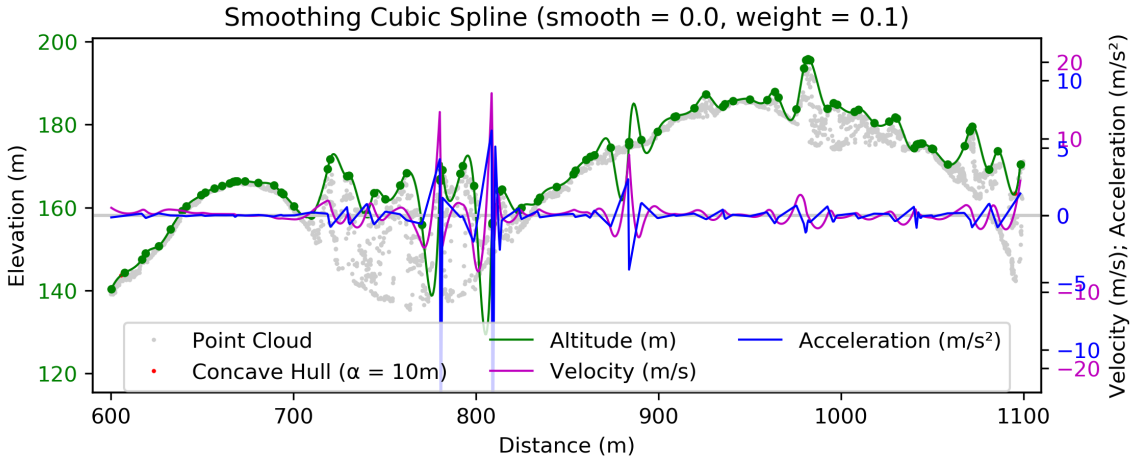
more acute angles. Or, the gradient between pairs of points can be calculated, with each point's weight depending on the maximum slope of its two adjoining segments. Thus there are three parameters to the smoothing spline: the point count (which depends on the surface-extraction algorithm, discussed below), the smoothing value and the weights. Figure 3.7 shows a smooth spline terrain that provides some reasonable fidelity to the imaged surface (though the slopes in this example would exceed the dynamic limits of the Matrice 600).

Figure 3.6 demonstrates the effect of different combinations of smoothing and weight parameters. The first instance demonstrate the zero-smoothing result, an interpolator. The others demonstrate increasing smoothness, decreasing velocity and acceleration and increasing root mean square (RMS) due to increasing smoothness.

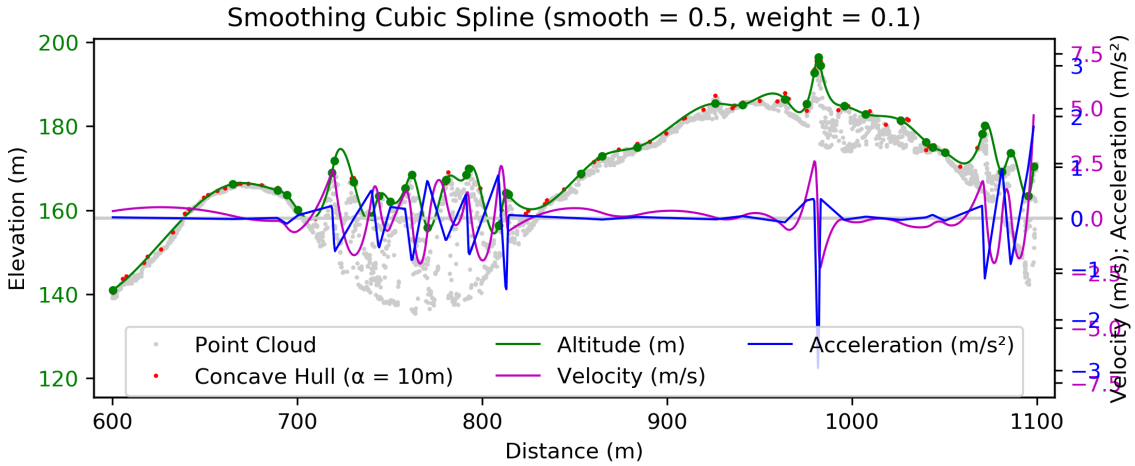
It is apparent that there is a great deal of variability in terms of the length of a trajectory, the characteristics of the terrain relief, the high-frequency variation in the height of the surface, and the intentions of the surveyor. It is clearly possible to extract a trajectory from a point cloud, using a smoothing spline, which provides a means of quantifying the suitability of the trajectory for research and safety objectives. However, there are some fatal flaws with the strategy.

First, given the diversity of site characteristics and mission objectives, and the wide range of serviceable parameters, some expert intelligence or intuition would be required for the initial selection of parameters. Dierckx' documentation comments describe the hunt for a suitable smoothing parameter (Dierckx, 1979):

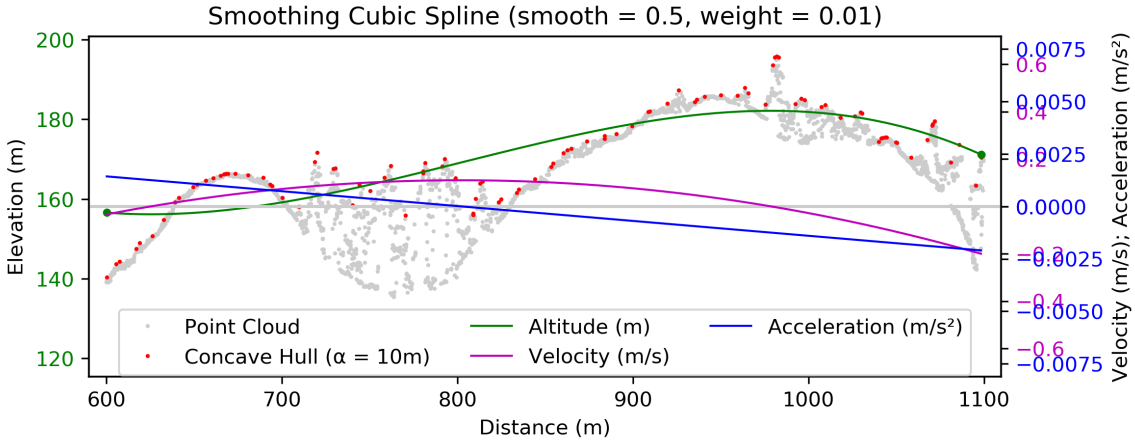
> [R]ecommended values for s depend on the weights $w(i)$. if these are taken as $1/d(i)$ with $d(i)$ an estimate of the standard deviation of $z(i)$, a good s-value should be found in the range $(m-sqrt(2*m), m+sqrt(2*m))$. [I]f nothing is known about the statistical error in $z(i)$ each $w(i)$ can be set equal to one and s determined by trial and error, taking account of



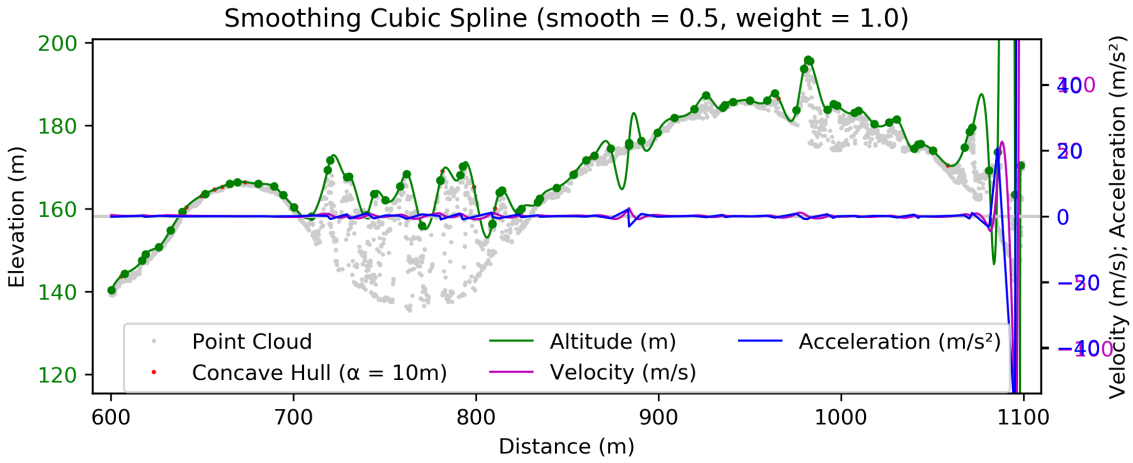
(a) Smooth spline and derivatives with smoothing = 0.0 and weights = 0.1.



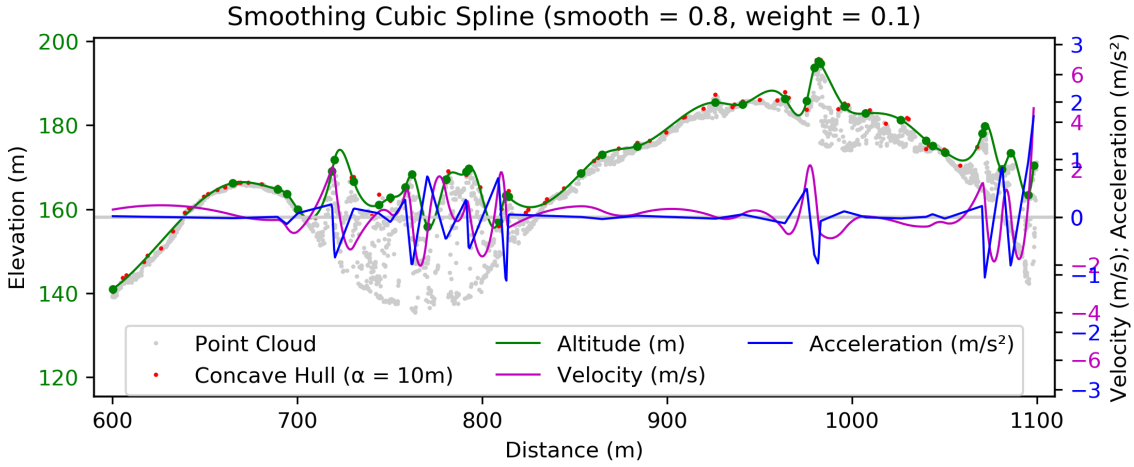
(b) Smooth spline and derivatives with smoothing = 0.5 and weights = 0.1.



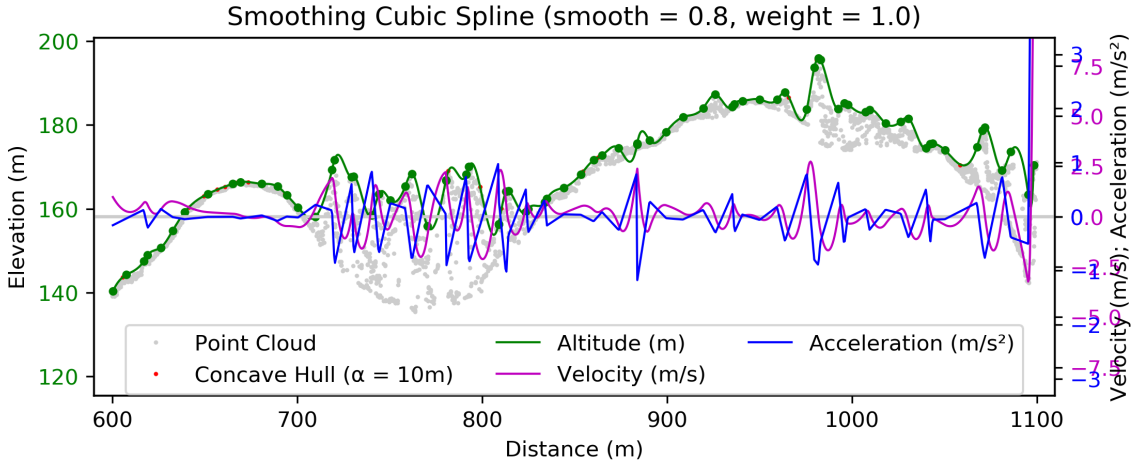
(c) Smooth spline and derivatives with smoothing = 0.5 and weights = 0.01.



(d) Smooth spline and derivatives with smoothing = 0.5 and weights = 1.0.

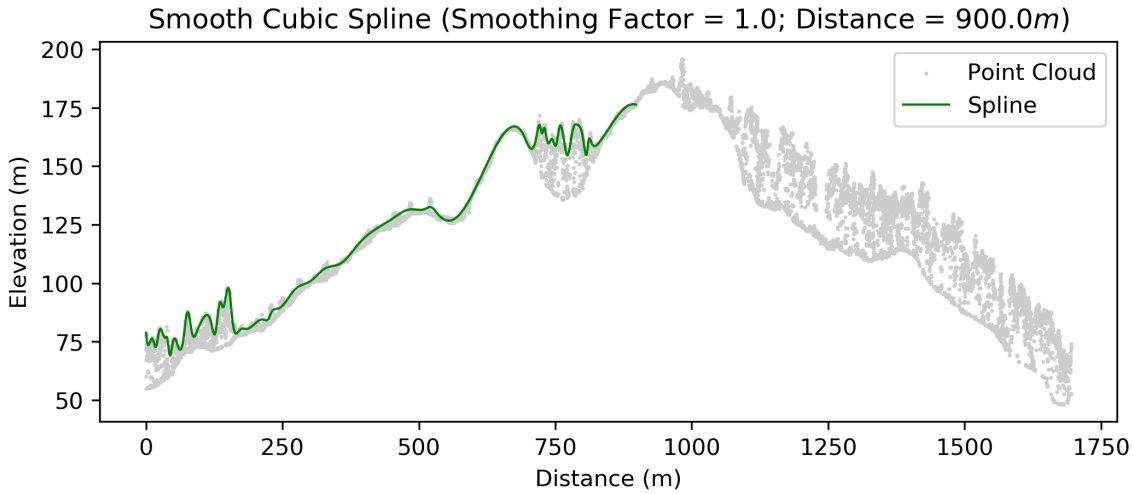


(e) Smooth spline and derivatives with smoothing = 0.8 and weights = 0.1.

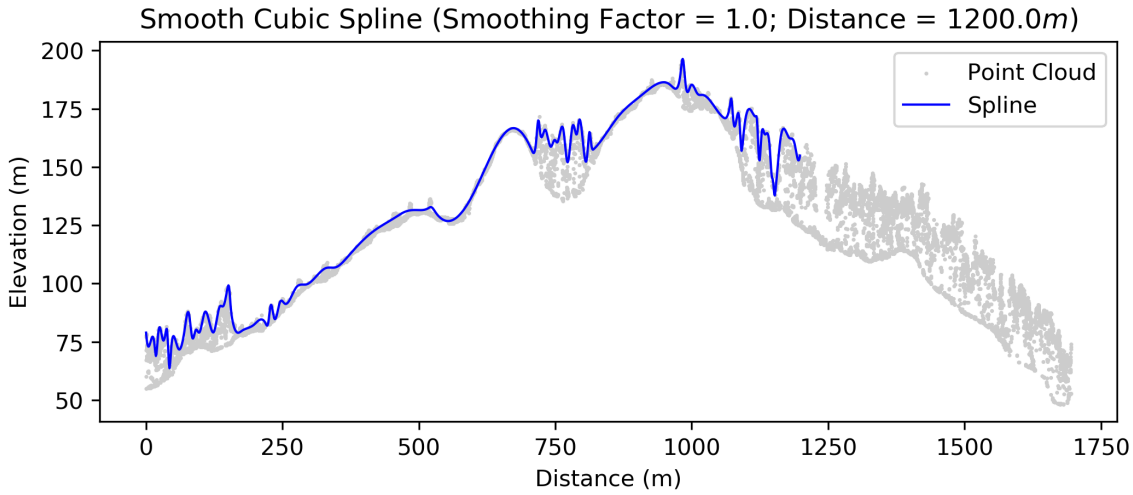


(f) Smooth spline and derivatives with smoothing = 0.8 and weights = 1.0.

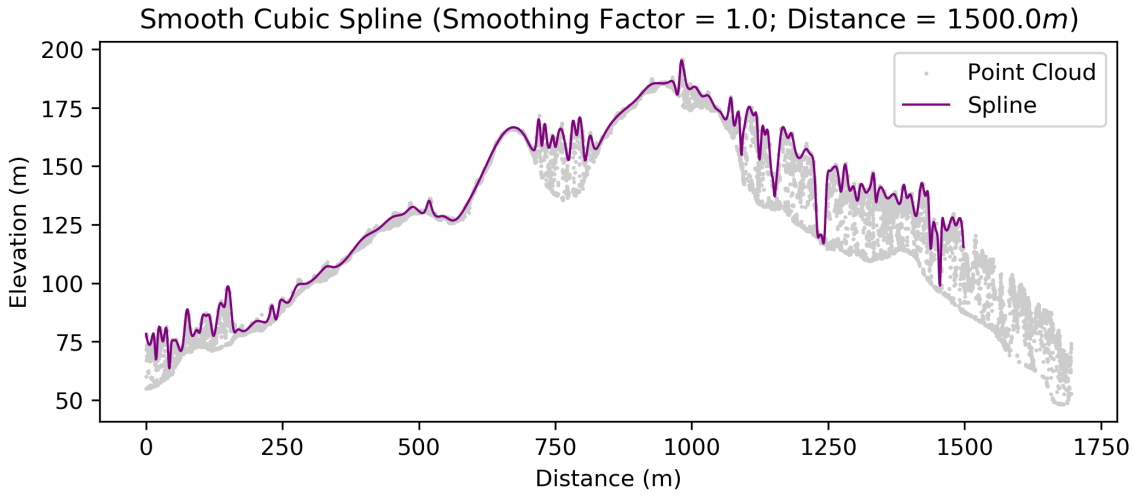
Figure 3.6: Effects of smoothing and weight parameters on smooth spline trajectory.



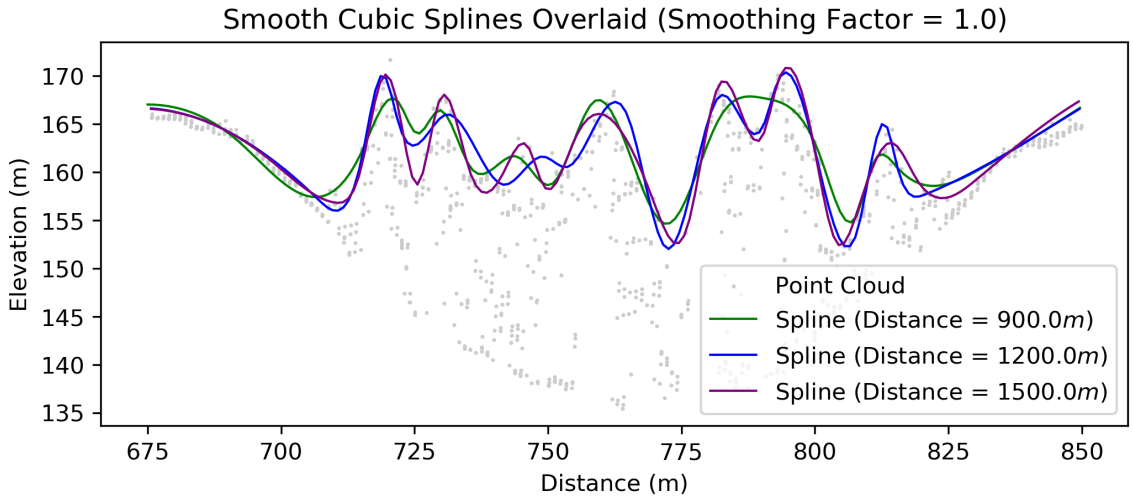
(a) Smooth spline at 900 m with smoothing = 1.0.



(b) Smooth spline at 1200 m with smoothing = 1.0.



(c) Smooth spline at 1500 m with smoothing = 1.0.



(d) Detail of overlay of splines from figures 3.7a, 3.7b and 3.7c.

Figure 3.7: Sequence showing the change in trajectory elevation as the aircraft advances from left to right. During each interval, the right-facing laser adds points to the point set and the smoothing spline is recalculated along its entire length. The final image shows a the three trajectories overlaid near the 780 m mark.

> the comments above. the best is then to start with a very large value of s (to determine the least-squares polynomial and the corresponding upper bound $fp0$ for s) and then to progressively decrease the value of s (say by a factor 10 in the beginning, i.e. $s = fp0/10$, $fp0/100$,...and more carefully as the approximation shows more detail) to obtain closer fits.

Selecting the smoothing parameter is clearly more art than science.

Second, as the vehicle progresses along its trajectory, appending surface points and re-computing the spline at intervals, the knots and coefficients are recalculated along its *entire* length, including the section currently under the aircraft (figure 3.7). This induces instantaneous "jumps" in the trajectory at each x-coordinate, which are not only deleterious in terms of data quality but physically impossible for the aircraft to negotiate, due to the singularity in the first derivative at the vehicle's current location.

Two possible solutions to the latter problem come to mind. First, blocks of the point cloud of a fixed length could be "finalized", and a trajectory built piece wise upon finalized blocks only. The addition of future blocks would have no effect on the finalized blocks. Because the positions and derivatives of the spline can be held fixed at the ends, the spline can be computed smoothly across the boundaries between blocks. So long as a block of point can be finalized early enough that a trajectory is available by the time the aircraft approaches the boundary, the problem appears solved. However, though the spline can be smooth and continuous across boundaries, it cannot be guaranteed that the derivatives will obey the dynamic limits of the vehicle; a cluster of points straddling the boundary will be weighted for each spline segment independently, potentially inducing loops or other problematic conditions.

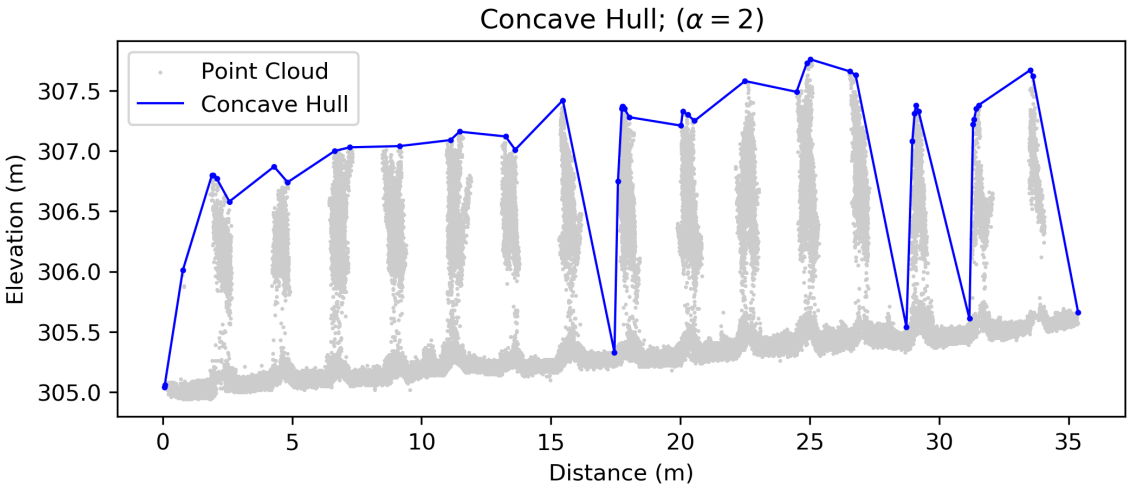
Savitzky–Golay Filter

The Savitzky-Golay filter was implemented from scratch in C++, using the open-source Eigen library for linear algebra support. The filter class implements configuration step (listing 2) which computes the table of coefficients and an evaluation (listing 3) step which extracts the function value and derivatives from the neighbourhood of a point in a list of points. The filter was implemented only for the 0^{th} to 2^{nd} derivatives, as those are required to satisfy the physical model.

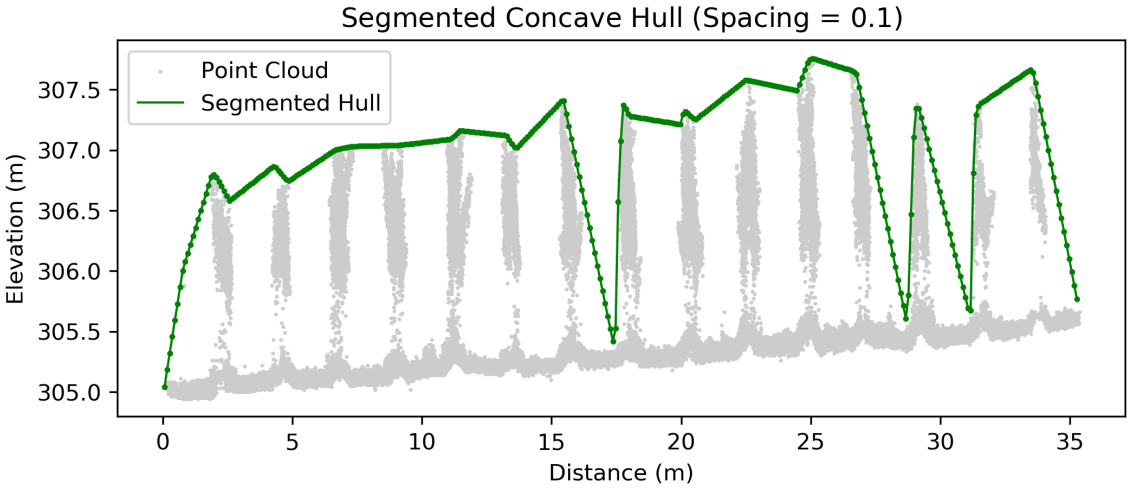
Unlike the spline method, a kernel requires the existence of an evenly-spaced point set. Binning can accomplish this implicitly, though both binning and hull generation may encounter the problem of gaps which must be closed by interpolation. An implementation of the interpolation method is shown in listing 4, with results in figure 3.8.

The kernel method requires that, in order for a reasonable window size to be used – to generate a trajectory at sufficient resolution for navigation – the point set must be suitably dense. Interpolation can accomplish this also, but this leads to the problem evident in figures 3.5 and 3.10, namely that surface detail obscured by the α parameter is replaced with an unrepresentative interpolated surface. The interpolated surface always describes a straight line from one bin or hull point to the next. Whereas the spline would form a smooth cubic curve through the sparse original point set, the interpolated linear surface forces the kernel to converge quickly towards the straight lines connecting those points. In practice this does not seem to be problematic, since the spline will not deviate far from the interpolated path in any case.

Figure 3.10 shows a comparison between the Savitzky-Golay filter and smoothing spline. When α is 0.25 m, enough surface detail is preserved that some trenches are retained; the functions take a similar shape, though the spline experiences more

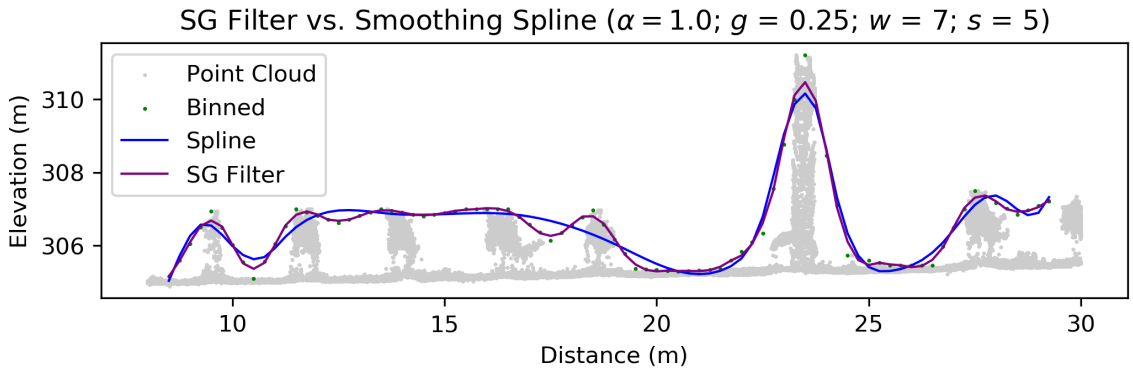


(a) Concave hull with $\alpha = 2.0\,\mathrm{m}$.

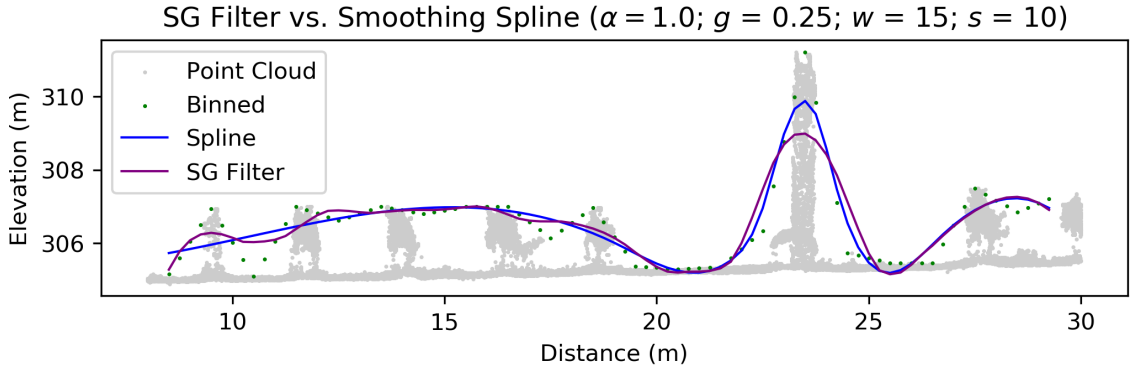


(b) Segmented concave hull from figure 3.8a with spacing $= 0.10\,\mathrm{m}$.

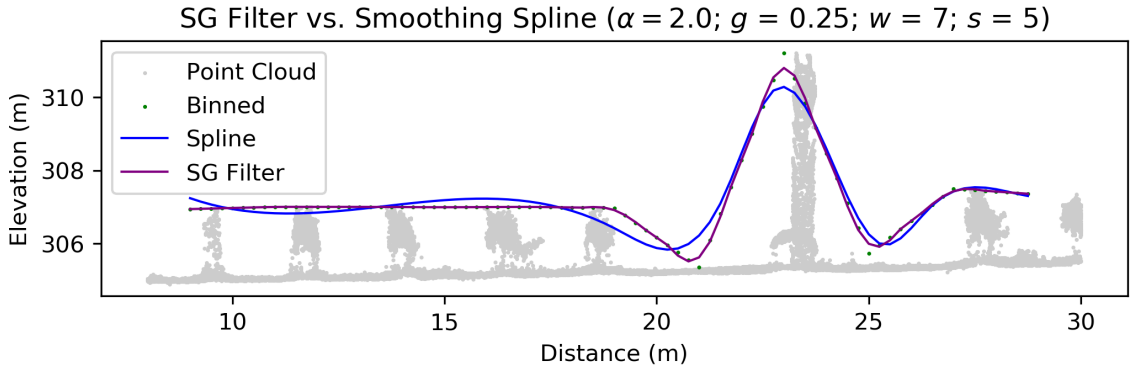
Figure 3.8: Comparison of concave hull with and without and segmentation.



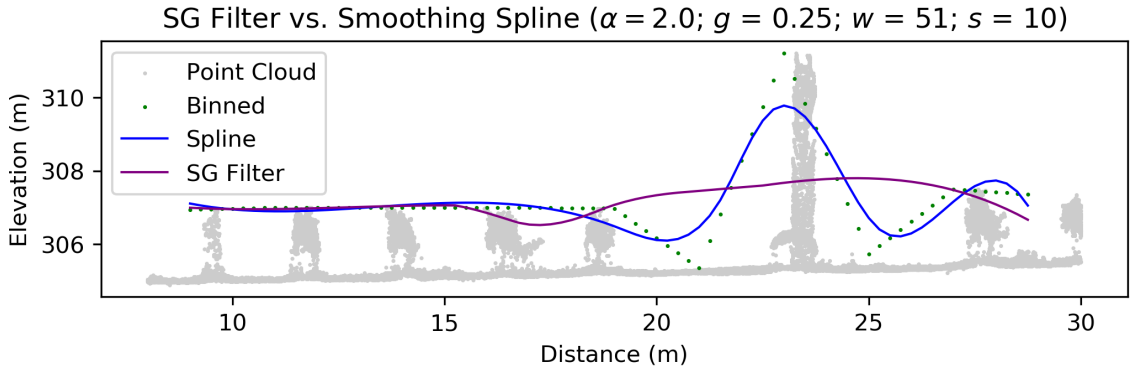
(a) SG filter with window size 7 and smoothing spline with smoothing = 5.0. Binned with $\alpha = 1.0\,\text{m}$ and segmented at $0.25\,\text{m}$.



(b) SG filter with window size 15 and smoothing spline with smoothing = 10.0. Binned with $\alpha = 1.0\,\text{m}$ and segmented at $0.25\,\text{m}$.



(a) SG filter with window size 7 and smoothing spline with smoothing = 5.0. Binned with $\alpha = 2.0\,\text{m}$ and segmented at $0.25\,\text{m}$.



(b) SG filter with window size 51 and smoothing spline with smoothing = 10.0. Binned with $\alpha = 2.0\,\text{m}$ and segmented at $0.25\,\text{m}$.

Figure 3.10: Sequence comparing the Savitzky-Golay and Smooth Spline trajectories. α is the bin size, g is the segmentation size, w is the SG windows size and s is the smoothing parameter (with constant weighting of $1/\sigma$).

overshoot due to its global response while the kernel's local response enables it to better conform to local features (e.g., the span between 7 m and 13 m in the first image). Increasing the window size and smoothing factor reduces the fidelity of both functions in different ways. At this scale fine control of the response of the SG filter is impossible, as the window size can only be adjusted in integral steps of two. The spline's smoothing factor, by contrast, is effectively continuous.

With the α set to 2 m, the importance of the window size and α decline rapidly, as most of the "noise" in the signal has already been removed by the binning process. However, the SG window size must be kept small enough to preserve the detail in the tall obstacle in figure 3.10.

3.3 Flight Control

The flight control system consists of four separate but related threads of execution. The DJI flight controller runs on the A3 flight computer and is concerned with the low-level dynamic control of the aircraft. The DJI control loop is a "black box" which can only be queried and manipulated using on of several DJI-provided software development kits – the Onboard SDK, in this instance – via a serial connection. The SDK is installed on the Raspberry Pi and handles serial communications with the A3, telemetry subscriptions and flight commands.

The three remaining threads – the sensor package, the platform and the planner – control the laser rangefinders, low-level sensory and control functions and communications with the DJI API, and trajectory extraction and flight planning, respectively. Figure 3.11 gives a high-level overview of the control process. The `Platform` and `Flight Planner` are described in detail below.

`Flight Planner`

1. The mission begins by loading a flight plan from the database.
2. The planner waits for telemetry information from the `Platform`.
3. If a command is available, it is retrieved.
4. If an error has occurred and the command can be skipped, the `Flight Planner` attempts to load the next command without executing the current one.
5. If if the command is not skippable or there is no error, the command is executed repeatedly until it returns successfully. Command execution may consume trajectory information from the `Platform` and send flight commands to the A3.
6. When the command has executed successfully, the `Flight Planner` attempts to load the next command.
7. If a command is available, the loop continues at step 3. If no command is available, the mission is ended.

`Platform`

1. The A3 flight computer (via the DJI API) supplies the `Platform` with telemetric information at up to 200 Hz, including orientation, position and velocity.
2. The nadir-facing rangefinder provides the absolute height above the surface at up to 20 kHz.
3. The `Platform` determines whether a collision or other undesirable event is likely to occur, and if so aborts. This suspends the dispatch of commands to the A3 which causes the vehicle to stop, allowing the pilot to take over.

4. Ranges supplied by the forward rangefinder are transformed into Cartesian points using the orientation information supplied by the A3, the subset of points comprising the "surface" of the point cloud are extracted by the concave hull or binning algorithm and the vertical trajectory of the vehicle is calculated.

5. If the derivatives of the calculated trajectory do not exceed the configured limits, course corrections are transmitted to the `Flight Planner`, otherwise an abort occurs.

The DJI Onboard SDK

Developers can subscribe to a number of telemetry topics at configurable intervals, including the vehicle's barometric altitude, orientation with respect to the Earth's inertial frame, velocity and fused GPS- and INS-derived position. Orientation is transmitted as a quaternion at up to 200 Hz while the fused GPS/INS position is transmitted at up to 50 Hz. Position, velocity and orientation metrics are pre-smoothed by the A3 using a Kalman filter or similar. Accuracy is reported as $< 1°$ for pitch and roll, $< 3°$ for yaw (without RTK), $0.05\,\mathrm{ms}^{-2}$ for horizontal velocity, $0.10\,\mathrm{ms}^{-2}$ for vertical velocity and $< 3\,\mathrm{m}$ for horizontal position (DJI, 2018b).

Flight commands are provided at a variety of levels of abstraction. High-level commands such as "arm" and "disarm" motors (start and stop), take off and land and return-to-home, are dispatched through the API once, whereupon the A3 automatically performs the complete action. Low-level commands can directly control the collective thrust and pitch, roll and yaw rates, though the A3 supervises the vehicle to ensure that its dynamic limits are obeyed. Low-level commands must be transmitted repeatedly: if a velocity command is sent once, the vehicle begins to accelerate to the target velocity and then stops unless it receives further commands.

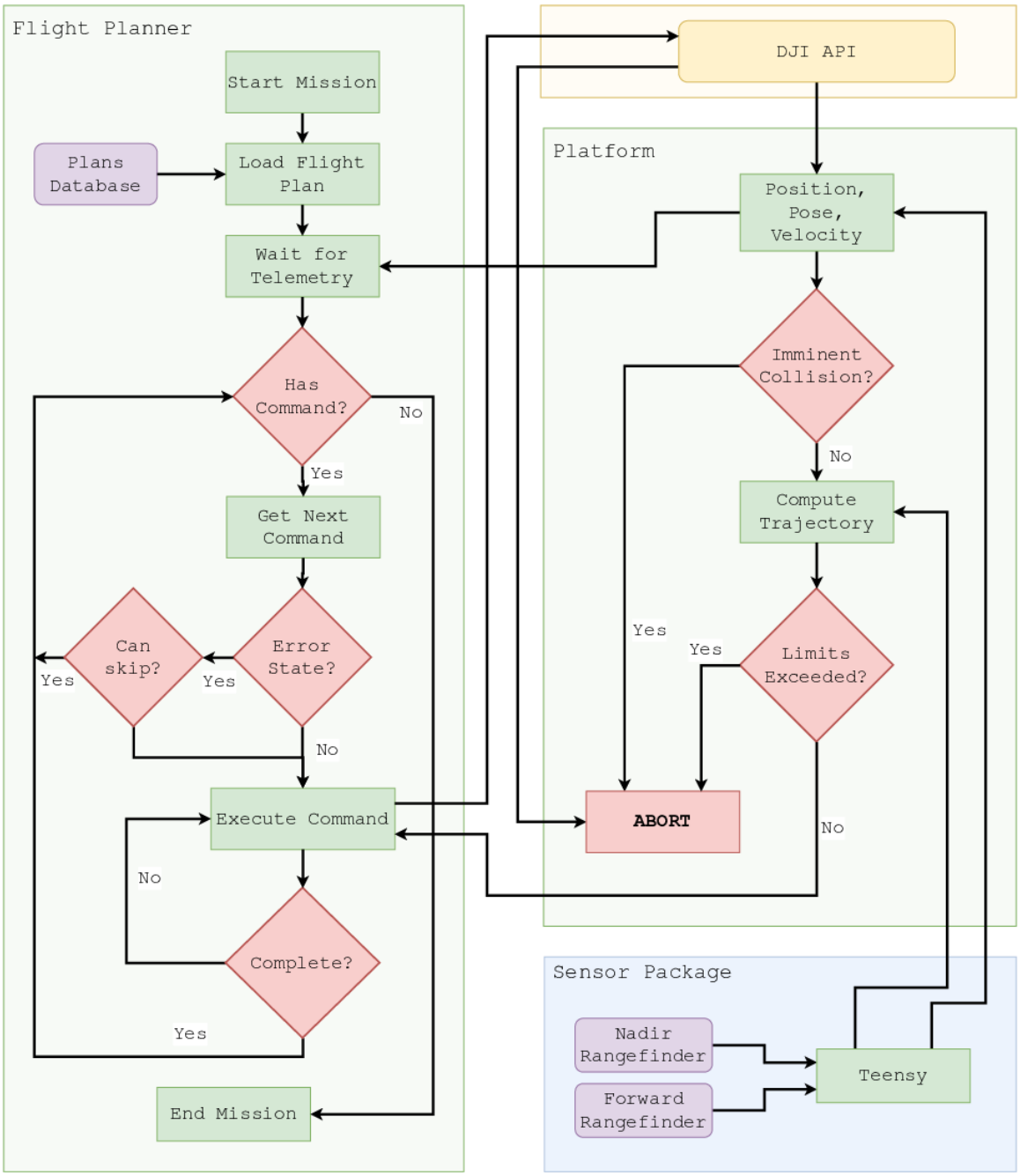


Figure 3.11: Flight control loops showing the Sensor, Platform and Planner threads of execution.

The Sensor Package

The sensor package software consists in two parts: the range collection and transmission loop on the Teensy and the receiver on the Raspberry Pi. Software on the Teensy is written in Arduino code, and in C++ on the Pi.

A range is emitted by the SF30/C rangefinder on a two-wire serial connection as a two-byte sequence, with `byteh` and `bytel` as the high- and low-order bytes respectively (LightWare Optoelectronics, 2019). The bytes are masked and OR-ed together to produce an unsigned short integer containing the range in centimetres. To guarantee the correct ordering of the bytes, the highest-order bit of the first byte is set, while that of the second is not. This can be checked as the bytes arrive – if the first byte to arrive has its most significant bit unset it is discarded. The algorithm for producing a range from the byte stream is given in Python in listing 5.

The SF30/C provides a synchronization capability via a single wire that is pulled low (i.e., its voltage is reduced) at the start of each laser pulse. This triggers a hardware interrupt on the Teensy. When the interrupt is received, the program records the current processor time and reads the ranges from the serial wires. By this method, each pulse is assigned a timestamp and the ranges and timestamps are stored in a circular array. This process occurs identically for the forward-looking scanning rangefinder and the nadir-oriented rangefinder. As the Teensy has no floating-point unit, and fractional metres would require four bytes of storage rather than two, they are transmitted in the original form.

The main program loop on the Teensy checks whether any nadir or forward-looking ranges have become available since the last iteration. If so, it packages them and transmits the packet on the serial wire. Each packet is in the form given in listing 6.

`HEAD_BYTE_1` and `HEAD_BYTE_2` are constants arbitrarily selected to identify the

start of a packet; `DATA_LENGTH` is the length of the entire packet, excluding the header bytes, length and checksum; and the `CHECKSUM_1` and `CHECKSUM_2` bytes comprise a checksum to verify the integrity of the packet.

The data stream consists of 8-bit bytes, or octets, which have no unambiguous interpretation. A single distinct octet can simultaneously represent the integer 65, the ASCII letter 'a', some other meaningful value, or a part of one. There is therefore no way to uniquely identify the start of a packet using bytes that could not appear in a given order in the interior of a packet, or in a corrupted byte stream. With two bytes used to identify the header, the odds of detecting an apparent header in a random byte stream are 1 in $256 * 256 = 65536$. Given the volume of packets expected, these odds are not acceptable. The header is therefore used to identify a *candidate* packet, and the checksum is used to verify it.

To improve the odds of rejecting a corrupt packet (and decrease the odds of rejecting a valid one) Fletcher's 16-bit checksum algorithm is used (Fletcher, 1982). Fletcher's original algorithm contained inefficiencies so Nakassis' more efficient implementation (Nakassis, 1988) was substituted. In essence, Nakassis' algorithm removes the expensive modulus operations from the inner loop, postponing them for a number of iterations determined by the variable size (i.e., 8, 16 or 32 bits). The implementation is given in listing 7, where `b` is the octet buffer, `M` is the modulus (255), `L` is the length of the buffer and `n` is a value suggested by Nakassis for governing the inner loop.

Fletcher's algorithm is marginally less efficient than a simple modular sum of the packet's bytes but, unlike the latter, provides a check on the ordering. Fletcher's algorithm is much more efficient than the common cyclic redundancy check method. The latter is often implemented in hardware for similar applications but is not available in this case.

When a packet arrives in the flight controller, it is read in its entirety based on the length value after the header (whose maximum value is 255). The checksum is calculated on the entire buffer after the header, and compared to the transmitted checksum. If there is a mismatch, the packet is simply rejected – the large volume of data in transit, and the complexity of a resend mechanism would not be worth the extra effort.

Within the packet, lists of forward-looking and nadir ranges are each identified by a single byte flag (`RANGE_BYTE` or `NADIR_BYTE`) followed by a count. Each range is stored as a two-byte unsigned short integer representing the measurement and a four-byte unsigned long integer representing the timestamp. Data that do not fit into a single packet are split over multiple packets.

The receiver is implemented in the `SerialScanner` class in C++ on the Raspberry Pi. `SerialScanner` inherits from `Serial`, which implements a basic low-latency socket IO paradigm based on `POSIX` system calls. `SerialScanner`'s main loop calls a non-blocking `read` call which moves received bytes into a circular buffer. The program searches this buffer for the header bytes, packet length and checksum, then waits for a sufficient number of bytes to have been received. It then verifies the checksum and reads the forward-looking and nadir ranges into their respective arrays. A callback method is used to notify an interested object (the `DJIPlatform` class, discussed below) whenever a packet is received.

The Platform

The `DJIPlatform` class represents the vehicle and manages all sensor and telemetry inputs and all commands to the A3. When the `SerialScanner` class notifies the `DJIPlatform` of an update to the forward and nadir range lists, it copies these values to an internal list and clears the scanner's lists. The time of arrival of the first forward

and nadir ranges is marked using the Pi's internal timestamp – minus an estimated delay to account for the serial communication lag – and the times for each subsequent point in each list recalculated based on the timestamp of the first point.

The `DJIPlatform` also subscribes to and receives telemetric updates from the A3. Three subscription packages are enabled (table 3.3). The 1 Hz package receives the altitude of the home point according to the altimeter. At the start of a flight, the controller waits for this value to become available before allowing the mission to begin, as all subsequent barometric altitudes will use it as a reference. The 50 Hz package receives information about the flight status (airborne or not), position and velocity of the vehicle, as well as the status of the controller mode switch which is used to abort a mission. The 200 Hz package tracks the quaternion and altitude. It is not necessary to check the altitude 200 times per second, however a bug in the DJI API causes a bus error when altitude and position are registered at the same frequency.

Topic	Topic Key	Freq. (Max.)
Altitude	TOPIC_ALTITUDE_FUSIONED	200 Hz (200 Hz)
Quaternion	TOPIC_QUATERNION	200 Hz (200 Hz)
Velocity	TOPIC_VELOCITY	50 Hz (200 Hz)
Is Airborne	TOPIC_STATUS_FLIGHT	50 Hz (50 Hz)
Position	TOPIC_GPS_FUSED	50 Hz (50 Hz)
Mode Switch	TOPIC_RC	50 Hz (50 Hz)
Home Altitude	TOPIC_HEIGHT_HOMEPOINT	1 Hz (1 Hz)

Table 3.3: Telemetry topics with maximum and as-implemented frequencies (DJI, 2018b).

Upon receipt of each telemetry update, the controller updates a `PlatformInfo` instance with the current telemetry. Before this occurs the previous state is transferred to another `PlatformInfo` object. Together, these `PlatformInfo` objects represent the state of the vehicle at the beginning and end of the interval.

Range to Point Conversion

When ranges are received from the `SerialScanner`, the timestamps are checked to ensure that they fall within a reasonable distance of the times of the telemetry updates as represented by the two `PlatformInfo` objects. The conversion of rangefinder ranges to Cartesian points requires information about the position and orientation of the instrument in space at the instant the pulse was fired. This is calculated by linearly interpolating the position vectors and quaternions in the `PlatformInfo` objects. The quaternions are interpolated by the SLERP (Shoemake, 1985) method, which is built in to the Eigen library's quaternion implementation.

The position and orientation of each rangefinder is given by a linear translation, $\vec{P_g}$, and rotation, O_g, relative to the inertial centre of the aircraft, in and around each of the orthogonal axes, X, Y and Z. The position, $\vec{P_v}$ and orientation, O_v, of the aircraft is given by another translation-rotation pair, corresponding to the instant of a laser pulse, as above.

Each range is represented by a vector $\vec{v_r}$,

$$\vec{r} = (r, 0, 0)^T, \tag{3.5}$$

where r is the range, a scalar. The range vector is transformed into a Cartesian point $\vec{p}$ in the world frame by,

$$\vec{p} = O_v * ((\vec{P_v} + \vec{P_g}) + (O_g * \vec{r})). \tag{3.6}$$

$\vec{p}$ is then added to the list of surface points.

The Flight Planner

The `DJIFlightPlanner` class is responsible for loading missions and coordinating the flight activities of the `DJIPlatform` – it is the "brain" of the terrain following system. Flight plans are loaded from a Spatialite database stored in on the flight controller's hard drive. Each plan consists of a sequence of waypoints in the form of a `Linestring` geometry type, a nominal flight altitude and a nominal forward velocity. The parameters of the binning algorithm and Savitzky-Golay filter are also stored – the bin size, the window size and the degree of the smoothing function.

A flight, or mission, is decomposed into a series of `FlightCommand` objects. Specialized subclasses of `FlightCommand` are designed for specific tasks including taking off, starting and stopping the terrain-following system and navigating to positions. A typical remote sensing flight pattern consists of a sequence of parallel lines, with the vehicle starting at the initial point of one line, flying to the end point, turning, and beginning at the end point of the adjacent line and so on.

The main loop of the `DJIFlightPlanner` class is responsible for retrieving a `FlightCommand`, executing it, waiting until it completes and executing the next one. A typical mission would decompose into the following object sequence:

1. Take off (`TakeOffCommand`).
2. Go to the start of the mission (`GoToPositionCommand`).
3. Start the rangefinders and terrain-following algorithm (`StartFollowingCommand`).
4. Fly the mission waypoints in sequence (`GoToPositionCommand`; one for each waypoint).
5. Return to the home point (`GoToPositionCommand`).
6. Land and shut down (`LandCommand`).

The `GoToPositionCommand` uses the vehicle position and orientation information and moves the vehicle towards its target position and altitude using the velocity and yaw rate commands provided by the DJI API. The vehicle's initial and target heading, `w0, w1` (rad), are derived from the horizontal displacement, `dx, dy` (m), with respect to the target waypoint. `w1` is limited to the range $0-2\pi$, in the direction requiring the shortest turn. The yaw rate, `da` (rads^{-1}) is calculated using a proportional-integral-derivative (PID) controller, as described below.

The `GoToPositionCommand` can be set to stop at the destination coordinate or continue through it. If stopping is required, the same proportional control is used for the forward velocity as for the yaw rate. There is no perfectly reliable means of determining whether the vehicle has passed through the waypoint, given the limits of GPS accuracy and precision. With real-time kinematic GPS (RTK) location enabled, the position should be accurate to within a few centimetres, and without, to within a metre (DJI, 2018a). However, even with RTK enabled, temporary failures could cause the vehicle to miss the waypoint. When this occurs, it simply turns and attempts to pass through it again before continuing on to the next waypoint. A target radius of 1 m was selected as a compromise between precision and the likelihood of the vehicle failing to detect its destination. Horizontal navigation features of the `GoToPositionCommand` class are enabled by default. When the terrain following mechanism is activated, the vertical component is enabled. The target altitude is calculated from the trajectory elevation, with the nominal mission altitude added.

The velocity and acceleration extracted from the trajectory derivatives are not used to set the vehicle's actual vertical velocity, which is instead calculated by a PID controller from the difference between its present altitude, the desired altitude, the horizontal distance to the next waypoint, and the forward velocity. The trajectory derivatives merely provide information about whether the trajectory would be non-

negotiable under ideal conditions. Otherwise, the vehicle will attempt to continue.

Listing 8 shows the configuration and dispatch of a flight command to the DJI API. The `flightCtrl` method is called repeatedly in the main loop of `DJIFlightPlan` until the waypoint is reached, at which point the current `GoToPositionCommand` is discarded and the next command executed.

Proportional-Integral-Derivative Control

Simple PID controllers were used for the heading, altitude and course deviations. This was done primarily to prevent or resolve overshoot. Integral wind-up was controlled by using a circular buffer to accumulate errors over time. Old error values are simply overwritten by new ones as they arrive. Tuning was performed by observing the performance of the controllers in the field and adjusting the parameters accordingly.

Global Abort Function

Safety is a critical concern for any autonomous vehicle. DJI (2018b) suggests the use of the controller mode toggle switch as the means by which an operator should regain control of a vehicle under the command of the OnboardSDK. A mission should also abort if an inter-process communication (IPC) signal (e.g., a segmentation fault, bus error or interrupt) is received from the system or if the trajectory function produces a path which is not navigable according to its derivatives.

A globally-accessible `abort()` function was implemented to be called from anywhere in the system at any time. The effect of this call would be to exit the control threads and return control to the user. In this scenario, the vehicle would stop immediately and hover in place.

Mission Planning

Some means of loading mission plans onto the flight controller was required, one that would not be too onerous to update in the field. A simple web service and web page were created using NodeJS, which could be accessed from a web browser. The web page allows for the creation of a mission plan using an uploaded `mme` file produced by the MapsMadeEasy mission planning software, or a GeoJSON geometry saved as a `json` file. Additional attributes such as the nominal altitude and horizontal velocity, bin size, command interval and Savitzky-Golay parameters are stored along with the mission plan. With the Raspberry Pi configured to work as a wireless access point, missions can be configured from any computer or cellular phone with WiFi capability. A screenshot of the administration page is shown in figure 3.12.

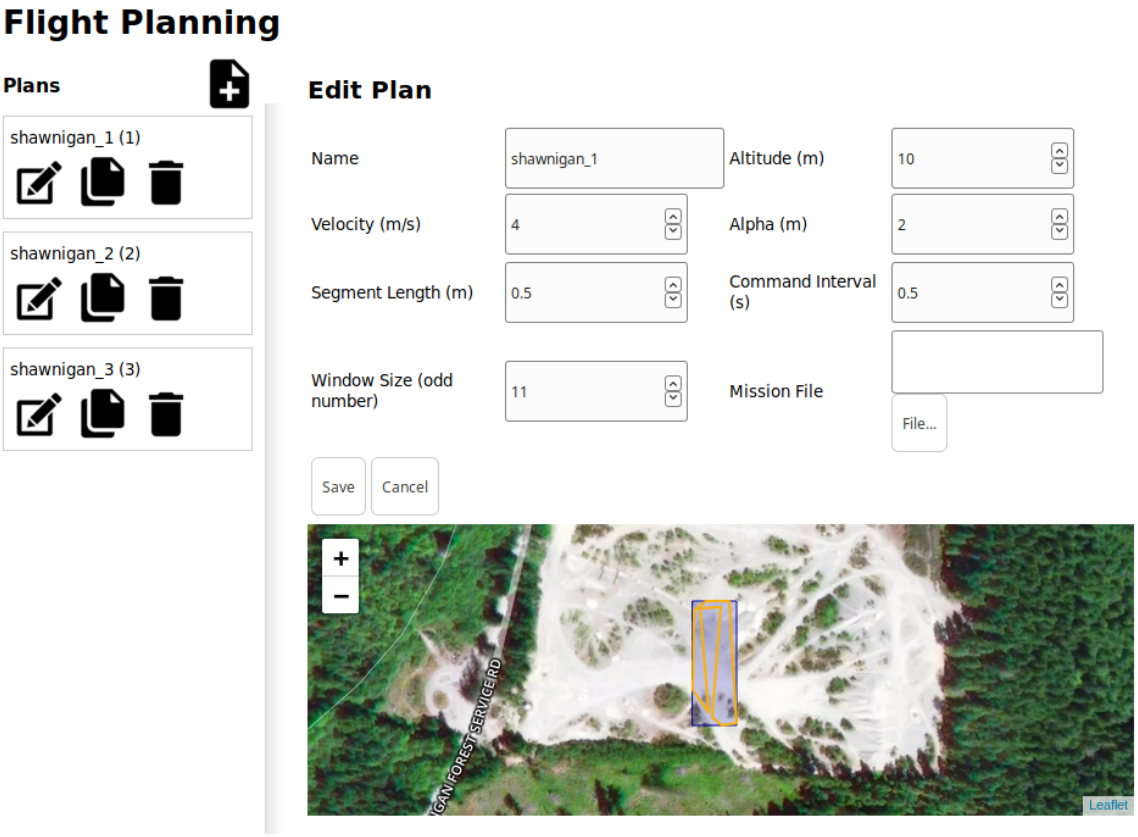


Figure 3.12: Mission administration page, hosted on the flight controller.

```
typedef Matrix Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic>;
typedef Vector Eigen::VectorXd;

void configure(int windowSize, int order = 3) {

        m_configured = false;
        m_windowSize = windowSize;

        double delta = 1;
        int halflen = windowSize / 2;

        Matrix A(order + 1, windowSize);

        int i = 0,  j = 0;
        for(int x = halflen; x >= -halflen; --x) {
                for(int o = 0; o < order + 1; ++o) {
                        A(j, i) = (double) std::pow(x, o);
                        ++j;
                }
                j = 0;
                ++i;
        }

        Vector y(order + 1);
        for(int i = 0; i < order + 1; ++i)
                y(i) = 0;

        char flag = Eigen::ComputeThinU|Eigen::ComputeThinV;

        // Entries for each derivative.
        y(0) = factorial(0) / std::pow(delta, 0);
        m_c0 = A.bdcSvd(flag).solve(y);

        y(0) = 0;
        y(1) = factorial(1) / std::pow(delta, 1);
        m_c1 = A.bdcSvd(flag).solve(y) * -1;

        y(1) = 0;
        y(2) = factorial(2) / std::pow(delta, 2);
        m_c2 = A.bdcSvd(flag).solve(y);

        m_configured = true;
}
```

Listing 2: Savitzky-Golay filter configuration step in C++.

```
double evaluate(const std::vector<P>& input, size_t position, int deriv) {

        if(!m_configured)
                throw std::runtime_error("Not configured.");

        size_t half = m_windowSize / 2;

        if(position < half || position >= input.size() - half)
                return std::nan("");

        double v = 0;

        switch(deriv) {
        case 0:
                for(int j = 0; j < m_windowSize; ++j)
                        v += input[position - half + j].z() * m_c0(j);
                return v;
        case 1:
                for(int j = 0; j < m_windowSize; ++j)
                        v += input[position - half + j].z() * m_c1(j);
                return v;
        case 2:
                for(int j = 0; j < m_windowSize; ++j)
                        v += input[position - half + j].z() * m_c2(j);
                return v;
        default:
                throw std::runtime_error("Invalid derivative: "
                        + std::to_string(deriv));
        }

}
```

Listing 3: Savitzky-Golay filter evaluation step in C++.

```
def segment(pts, size):

        # Add first point to the output.
        out = [pts[0]]

        tx = pts[0][0]

        for i in range(len(pts) - 1):
                # Get the first two points.
                ax, ay = pts[i]
                bx, by = pts[i + 1]
                if bx == ax:
                        # If they're coincident, skip.
                        continue
                # Compute the slope.
                m = (by - ay) / (bx - ax)
                # For each point along the interval.
                while tx < bx:
                        # Calculate the y of the new point.
                        ty = ay + (tx - ax) * m
                        out.append((tx, ty))
                        # Advance the x by the given interval.
                        tx += size

        return out
```

Listing 4: Linear interpolation algorithm.

```
def get_range(stream):

        # Get the high byte
        byteh = stream.read()

        # If the byte's most significant bit is not set, read another.
        if not byteh & 0x80:
                byteh = stream.read()

        # Read the low byte.
        bytel = stream.read()

        # Mask, shift and OR; then return.
        return ((byteh & 0x7f) << 6) | (bytel & 0x7f)
```

Listing 5: Algorithm for producing ranges from SF30/C serial byte stream.

```
{ HEAD_BYTE_1, HEAD_BYTE_2, DATA_LENGTH, CHECKSUM_1, CHECKSUM_2,
        { RANGE_BYTE, COUNT, { range, timestamp }, ... },
        { NADIR_BYTE, COUNT, { range, timestamp }, ... }
}
```

Listing 6: Packet structure for ranges, and errors.

```
uint16_t fletcher15(uint8_t* b, int L) {
        uint8_t M = 255;
        int n = 21;
        int i1 = 1;
        int i2 = min(n, L);
        uint16_t d0 = 0;
        uint16_t d1 = 0;
        while(i1 <= L) {
                for(int i = i1; i <= i2; ++i) {
                        d0 = d0 + b[i];
                        d1 = d1 + d0;
                }
                d0 %= M;
                d1 %= M;
                i1 = i2 + 1;
                i2 = min(i2 + n, L);
        }
        d0 %= M;
        d1 %= M;
        return (d0 << 8) | d1;
}
```

Listing 7: Implementation of Nakassis' Fletcher checksum in 16 bits (Nakassis, 1988).

```
        // Control the vehicle's horizontal velocity and yaw rate in the
        // body frame.
        int flag = Control::HORIZONTAL_VELOCITY|Control::YAW_RATE
                |Control::HORIZONTAL_BODY;

        // If terrain following is enabled, also set the vertical velocity.
        if(hasAlt)
                flag |= Control::VERTICAL_VELOCITY;

        // Send the command to the DJI API.
        pf->control()->flightCtrl(Control::CtrlData(flag, vx, 0, vz, wr));
```

Listing 8: Sending horizontal and vertical commands to the DJI API.

Chapter 4

Experiments

To test the efficacy of the terrain following system, several flights were made over variable terrain with optical targets placed in strategic locations. Test flights were performed in a former gravel pit currently used by off-road motorcyclists, featuring a gradual down-slope form West to East and distinct and steep, high-frequency features (i.e., hills) with little vegetative cover.

Two flight plans were created, one traversing a North-South line of low relief and one an East-West line with higher relief. Targets were distributed as shown in figures 4.1 and 4.2. All targets were white except 9, which was black, due to the small number of white targets. During all flights, the surface was imaged continuously by the nadir-aligned DJI Zenmuse X3 camera. On November 7, targets were imaged exclusively with video; on November 20, the first trial was imaged with still photographs at 2 s intervals and all others with video. While still images are sharper than extracts from video, the default maximum frequency available through the DJI Go application was too low (≥ 2 s) so video was used instead.

Each flight plan was performed with and without terrain following, with one or two sets of parameters for the Savitzky-Golay filter. The non-terrain following trials

Figure 4.1: Orthophoto of study area with locations of flight lines and targets.

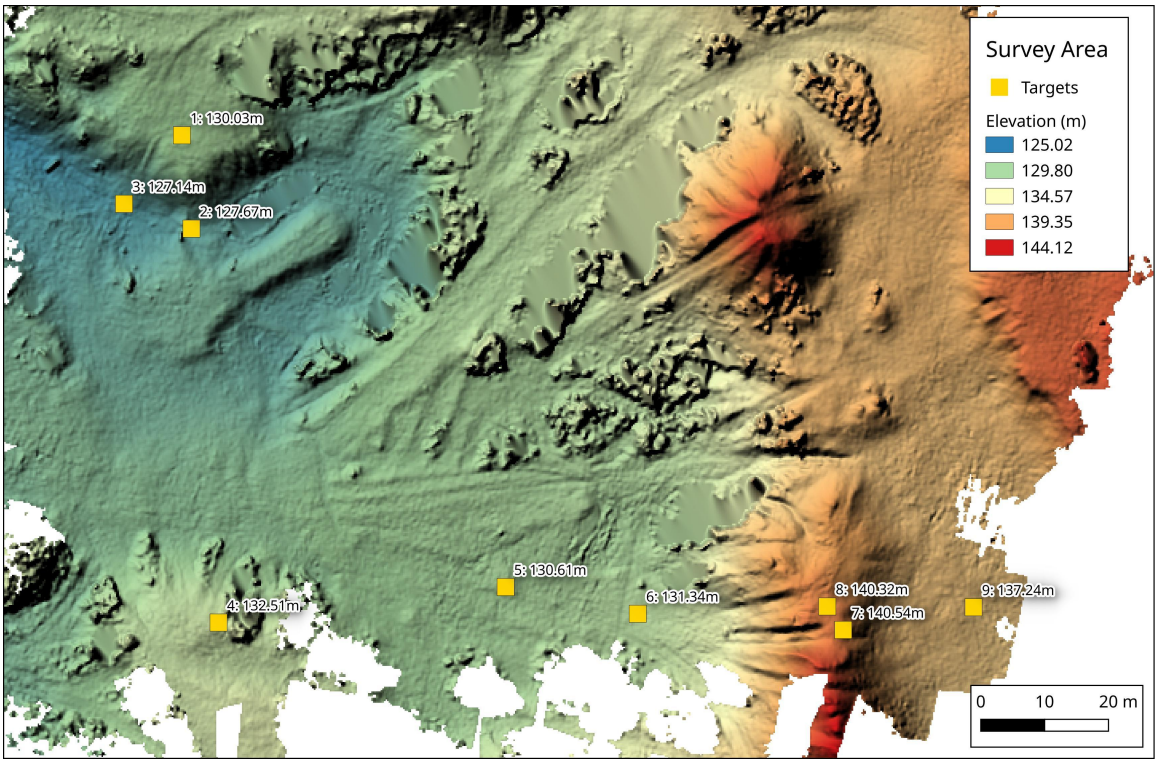


Figure 4.2: Hillshade relief map of study area with locations of targets.

were conducted at 10 m above the home or take-off point; terrain following trials were conducted at a nominal altitude of 10 m. The parameters for terrain following trials are given in tables 5.1 and 5.2.

Window and bin sizes were selected largely from intuition: with a bin size of 0.25 m and window size of 15, a curve would be fit by the SG filter over a distance of 3.75 m. A window size of 9 fits a distance of 2.25 m. Given the shape of the terrain at each trial, these distances seemed reasonable. The selection declination angles for the forward laser was not motivated by any particular criterion. In future work, each of these parameters should be examined in detail.

Flight	Follow	Declination Angle (°)	Altitude (m)	Velocity (m/s)	Alpha	Window Size	Bin Size (m)	Cmd. Freq. (Hz)
1	Y	42.5	10	4	1	15	0.25	4
2	N	42.5	10	4	1	15	0.25	4
3	Y	42.5	10	4	1	15	0.25	4
4	N	42.5	10	4	1	15	0.25	4
5	N	55.0	10	4	1	15	0.25	4
6	Y	55.0	10	4	1	15	0.25	4
7	N	55.0	10	4	1	15	0.25	4
8	Y	55.0	10	4	1	15	0.25	4

Table 4.1: Flights and parameters on November 7, 2019.

Flight	Follow	Declination Angle (°)	Altitude (m)	Velocity (m/s)	Alpha	Window Size	Bin Size (m)	Cmd. Freq. (Hz)
1	N	33.8	10	4	1	9	0.25	4
2	N	33.8	10	4	1	9	0.25	4
3	Y	33.8	10	4	1	9	0.25	4
4	Y	33.8	10	4	1	9	0.25	4
5	N	33.8	10	4	1	9	0.25	4
6	N	33.8	10	4	1	9	0.25	4
7	Y	33.8	10	4	1	9	0.25	4

Table 4.2: Flights and parameters on November 20, 2019.

Chapter 5

Analysis

In this research, the goal is not only to achieve a terrain following system but, more importantly, a forward-looking, predictive system that can optimize the quality of remotely sensed data products and determine in advance whether a trajectory is navigable. This analysis approaches the problem in three ways:

1. a subjective examination of the flight data records determines whether the system worked, and how it may have failed;

2. a statistical analysis of scale distortions in images of the radiometric targets shows whether remote sensing objective has been satisfied; and

3. the computed and actual trajectories are compared to an ideal modelled trajectory and to level flight.

5.1 Flight Data

The flight control software records certain flight data, not unlike the so-called "black boxes" used on commercial airliners. This flight data consists of:

- the vehicle GPS-derived position, barometric altitude and orientation with respect to the earth frame;
- the surface point cloud generated by the scanning laser at every position update interval; and
- the platform altitude as measured by the nadir-facing laser.

A visualization of these attributes provides a quick way of assessing the information received by the controller, and the decisions made thereon.

Figures 5.1 and 5.2 show the laser- and digital elevation model (DEM)-derived ground surface, the forward laser's point cloud, the extracted trajectory and the platform's altitude for five flights on each date. Some observations appear immediately:

1. the point cloud and the ground surface do not correspond closely, particularly on November 20;
2. the nadir laser- (red) and DEM-derived (green) ground elevations appear to correspond, though features appear in the point cloud (gray) and laser-derived ground surface that do not appear in the DEM-derived surface;
3. the aircraft altitude follows the trajectory (with an offset of 10 m, as designed) on November 7 and not on November 20, while on November 20 the aircraft follows the terrain with apparent fidelity;
4. there is a period on November 20 where the aircraft altitude is flat, despite the variance in both the measured surface and the planned trajectory; and
5. there is a noticeable delay between perturbations in the trajectory (November 7) or the ground (November 20) and the changes in altitude of the aircraft.

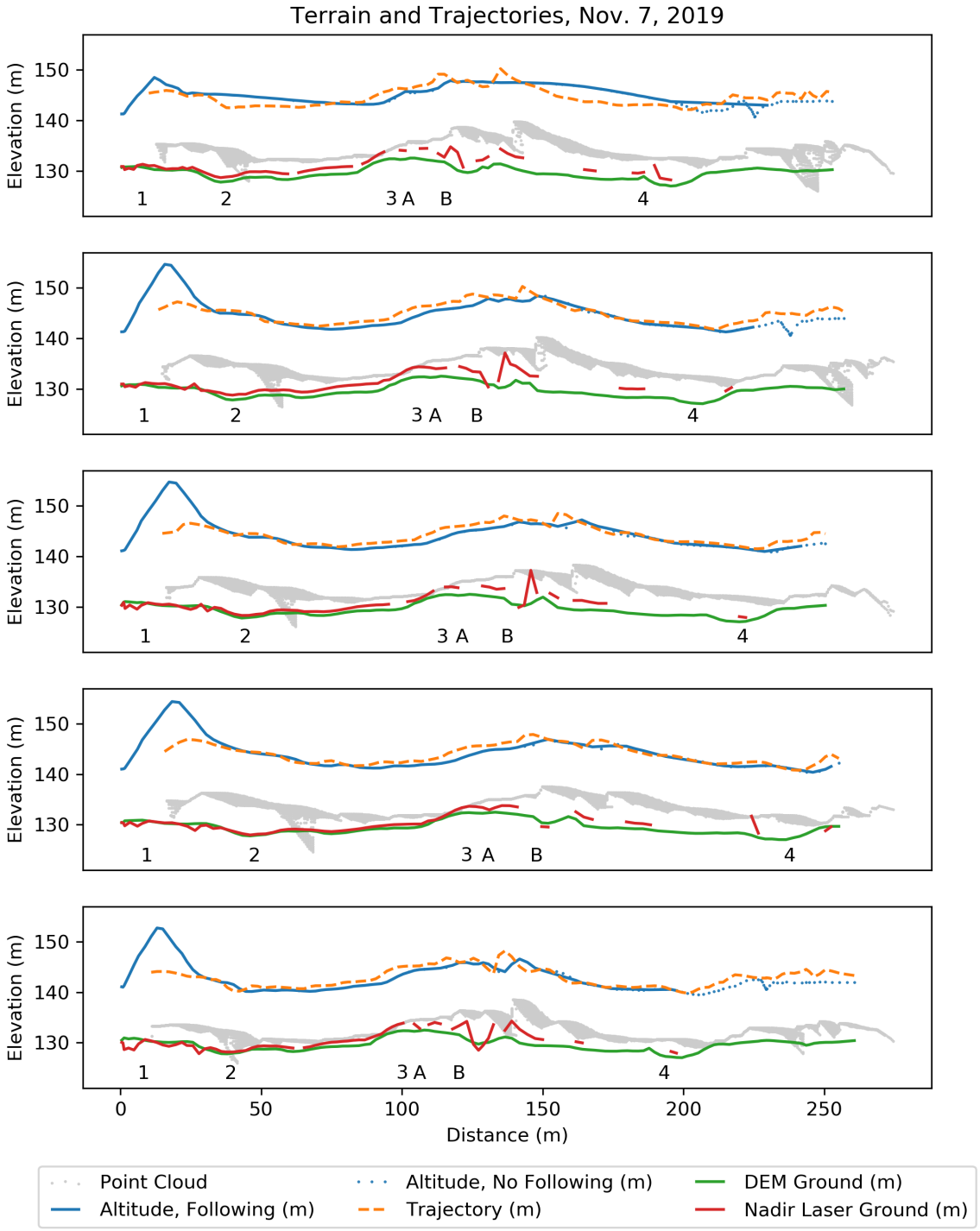


Figure 5.1: Altitude, trajectory, point cloud and ground for six flights on November 7

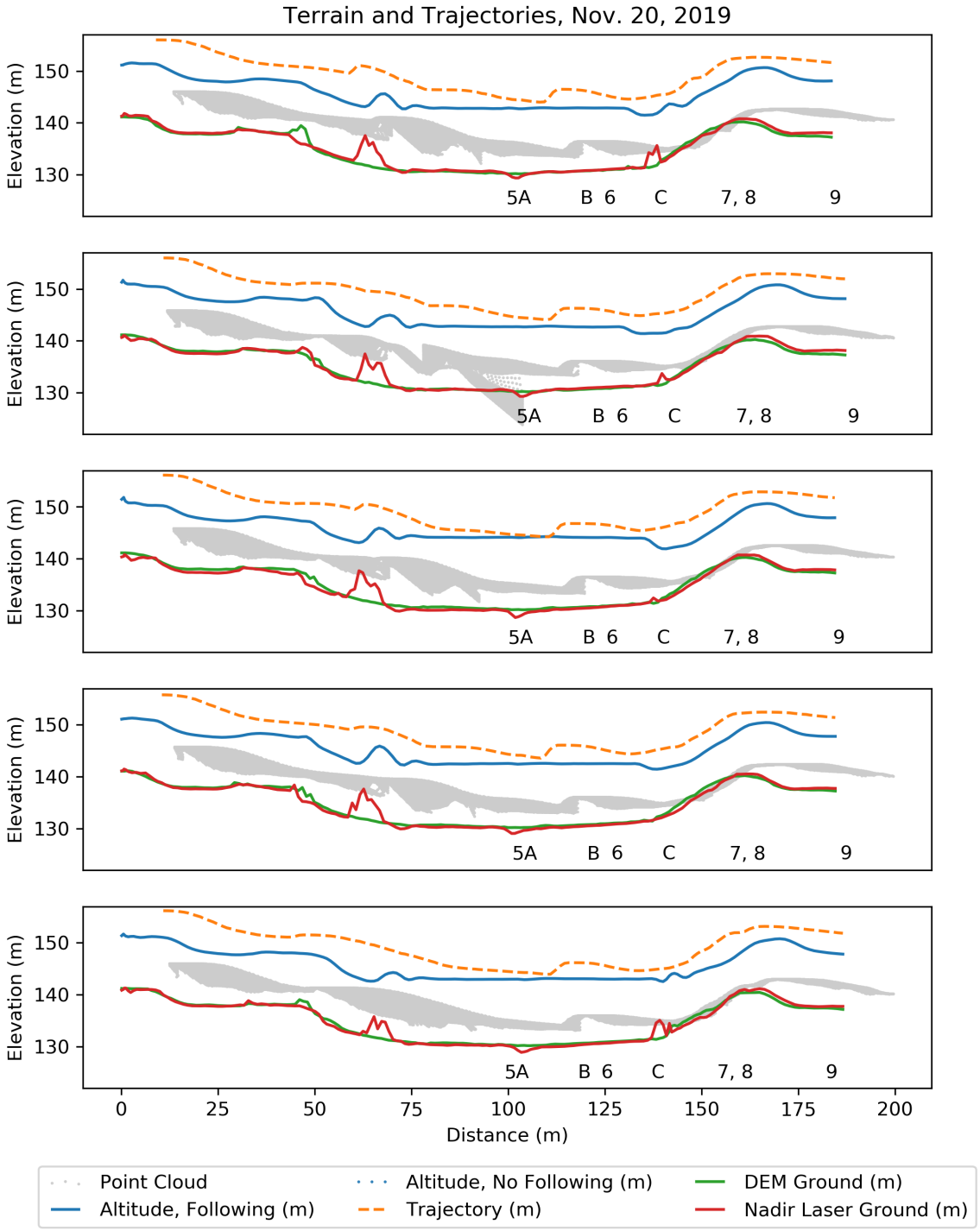


Figure 5.2: Altitude, trajectory, point cloud and ground for six flights on November 20

Issue 1

Clearly, the point cloud generated by the forward-facing laser is not in the correct location with respect to the surface. This is most likely due to an incorrect configuration: if the laser's declination angle is too small, points computed from the ranges will appear high and forward of their "true" location. The flight controller is capable of accurately measuring the forward laser declination angle (assuming the bore sight angle and translation parameters are correct), but not of updating the configuration, which must be done manually.

For the November 20 trials, the configuration was not updated. On November 7, the configuration was set, but the accuracy of the angle measurement was unsatisfactory. To the many possible sources of error in the point projection – INS accuracy, timing issues, bore sight angles and translations, etc. – one may add the difficulty of measuring the declination angle under field conditions; a flat, level surface is required.

Issue 2

Photogrammetry software works well with non-porous surfaces, but struggles with vegetated surfaces. The study site is dotted with clusters of small trees which are rendered as voids in the resulting DEM (see figure 4.2). These were filled by linear interpolation, leaving a smooth surface presumed to represent the ground beneath the vegetation. The flight paths were designed to avoid these clusters of vegetation, but due to misregistration of the maps and the inaccuracy of the vehicle's GPS – neither of which can be resolved, except by using more accurate real-time kinematic GPS – as well as the influence of wind and the vehicle's control response, the actual flight path frequently passed over vegetated areas. The nadir laser (red) records these features, while the DEM (green) does not.

There are two causes of infidelity in the point cloud (gray). The first was the

decision to project all laser ranges in a plane normal to the vehicle's y-axis, whereby the y-coordinate would be set to zero, and the x-coordinate to the vehicle's cumulative forward travel distance, plus the horizontal component of the laser range vector. This was done to simplify the process of extracting a surface efficiently. For reasons that are now obvious, this was a serious error: high-frequency noise in the vehicle position measurement tends to accumulate in the distance measurement, stretching the point cloud in the x-direction. This effect is clearly visible in all trials.

The second cause of infidelity occurs as a combination of the first, and of the tendency of the forward laser to sweep terrain outside of the flight path, especially during turns. Those features are re-projected *into* the flight path by the algorithm, where they become "phantom" obstacles which the trajectory algorithm nevertheless attempts to avoid.

Figures 5.3 to 5.6 demonstrate both phenomena. For each vehicle position (black squares), the entire surface point cloud is shown as a straight line of points of identical colour. These are the surfaces from which the trajectory would be extracted at each time step. (Note that points added to the surface are preserved across time steps: each line of points represents the terrain from the vehicle's present position to its future position; points are discarded when the vehicle has moved past them.) In figure 5.3 the vehicle is travelling from North (blue) to South (green) and then North (red); and in figure 5.5, the vehicle is travelling from Northeast (blue), to Southwest (green) and then to the East (red).

At locations C and D in figure 5.5, there are clusters of vegetation under the flight path to which the vehicle reacts, easily visible at position D in the oblique view (figure 5.6). At positions A and B on November 7, and A on November 20, there are a small hills and clusters of trees, visible in the orthophotos. As the vehicle rotates through turns, the laser records these objects. When the turns are complete and

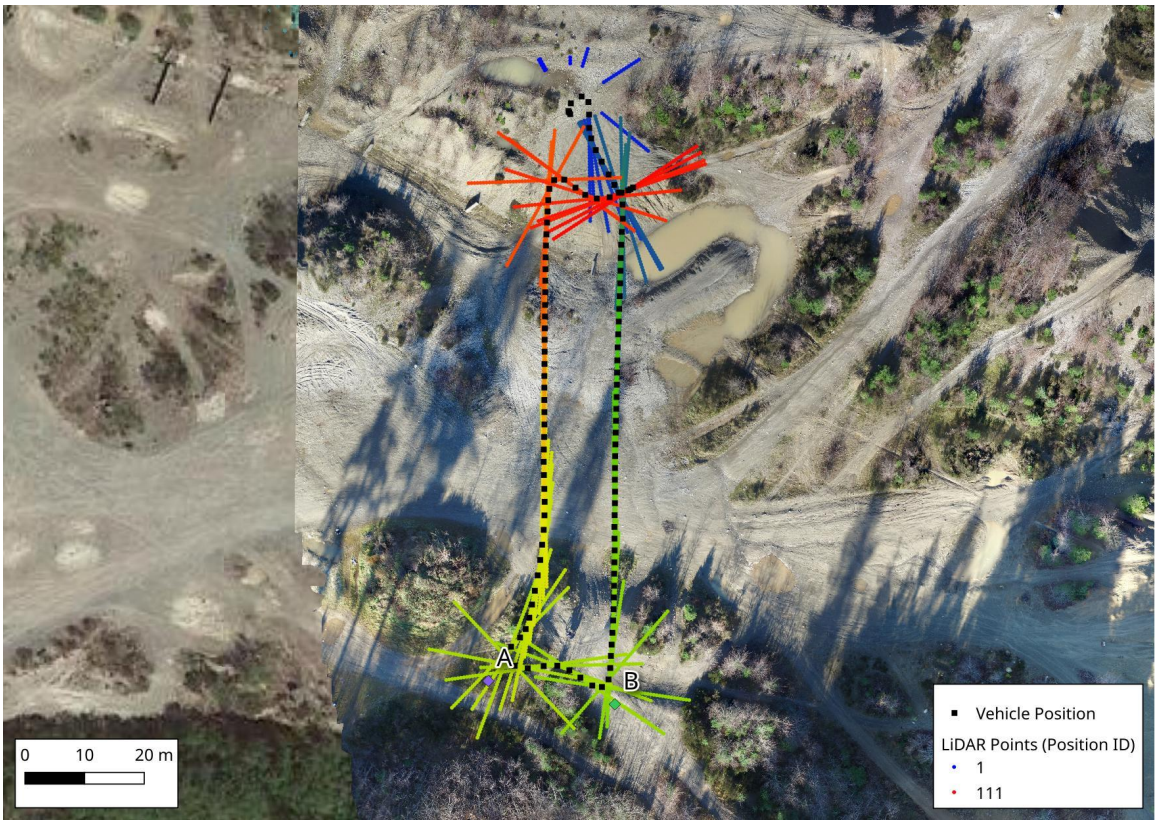


Figure 5.3: Top view of laser overshoot, flight 1, November 7.

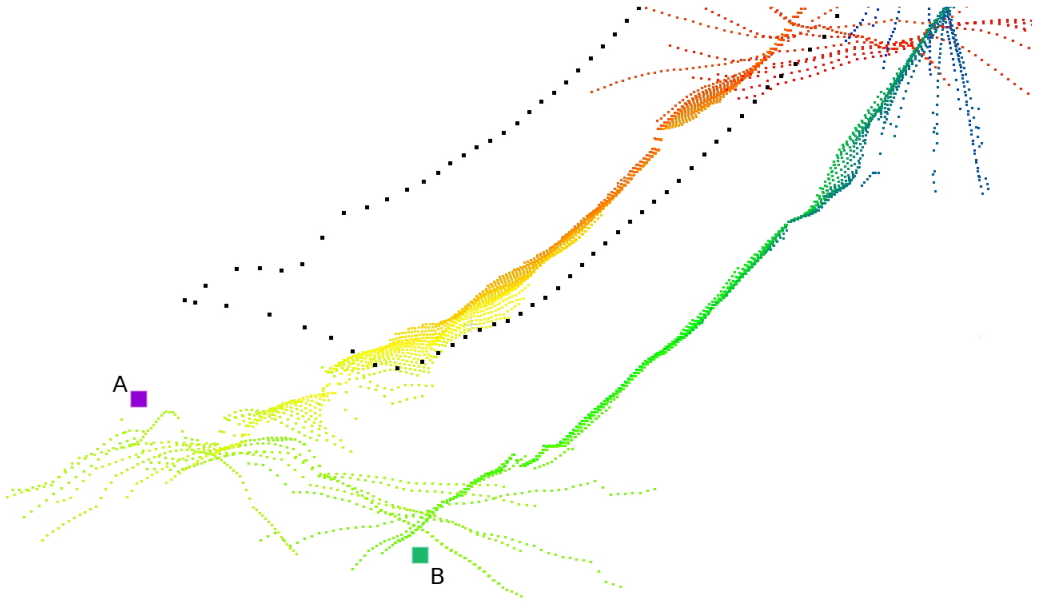


Figure 5.4: Oblique view of laser overshoot, flight 1, November 7.

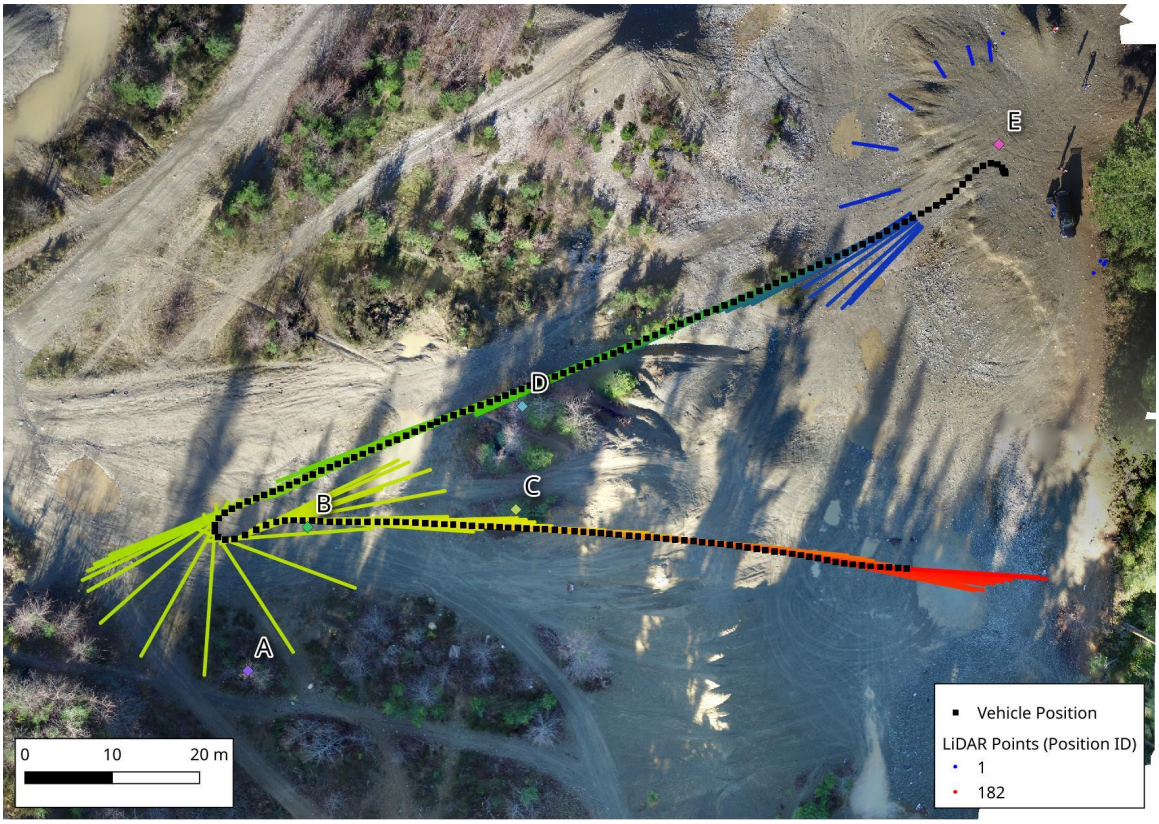


Figure 5.5: Top view of laser overshoot, flight 1, November 20.

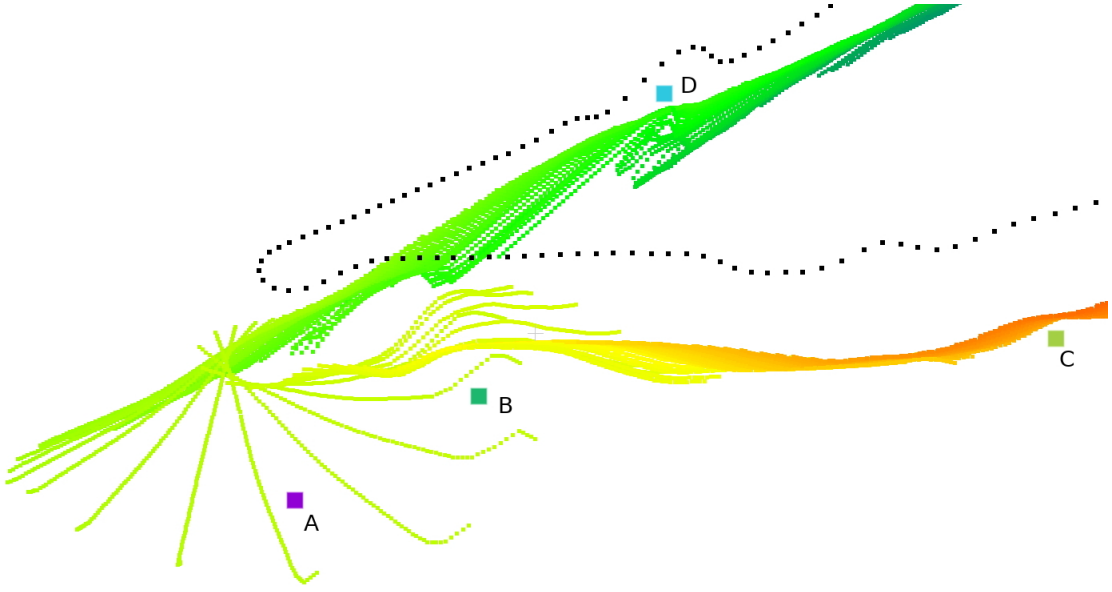


Figure 5.6: Oblique view of laser overshoot, flight 1, November 20.

the vehicle resumes forward flight, the "phantom" objects are still represented in the point cloud, easily visible at position B in figure 5.6.

Issue 3

The flight data records whether the control altitude was extracted from the trajectory function or from the nadir laser. On November 7, the trajectory function controlled the altitude for the duration of each flight. On November 20, the trajectory function was ignored. It is not clear why this occurred, nor was it noticed in the field.

Issue 4

It would seem intuitive to provide a minimum target altitude as a sanity check for the flight controller. Unfortunately, because the flight controller uses a barometric altimeter and there is no immediate way to calibrate the altimeter against the true (orthometric or ellipsoidal) altitude in real time, the vehicle altitude is normalized against the initial barometric altitude, which is set to zero. When the vehicle's home point is higher than the terrain, a target altitude may be less than zero. A sanity check in the flight controller was set to forbid target altitudes below zero, which forced the vehicle into level flight. This occurred in each of the trials on November 20.

Issue 5

Part of the motivation for this research is to implement a *predictive* terrain-following system, as current solutions using nadir-aligned laser rangefinders are purely reactive. During the November 20 trials, the predictive system failed (see issue 3), falling back on the nadir laser, which operated according to expectations.

Conversely, the November 7 trials show that the trajectory function (orange) anticipated terrain features represented in the point cloud. The terrain function still

shows a delay relative to the terrain, but this can be attributed to the x-coordinate stretch in the point cloud (see issue 2). It appears that a properly-calibrated point cloud would have enabled the trajectory function to correctly anticipate the terrain.

5.2 Imagery

The the `ffmpeg` program was employed to extract key frames from the videos and the resulting images were sorted into sets corresponding to each trial. The video for the two final trials on November 20 became corrupted and the data for these was lost. In total, 15 trials along three flight paths produced usable data, eight with terrain following and seven without.

It would have been time-consuming and subjective to manually digitize the targets in the imagery. Instead, the Python port of the OpenCV library was employed to detect and measure targets over the full image set, with visual verification of the result using images with the extracted outlines drawn over the targets (figure 5.7).

To compensate for lens distortion, the camera matrix and distortion coefficients were derived using a checkerboard target and calibration routines provided by the OpenCV project. The calibration program consumes a video or series of photos of the checkerboard within the camera's field of view and produces an XML document containing the calibration coefficients. These are then provided to the OpenCV `undistort` method, which transforms the images.

While white targets could be identified using digital number alone, dark targets were identified by computing the blue-red band ratio and scaling the result to 8-bit resolution (0-255). Each image was then blurred slightly with a Gaussian kernel and converted to a binary image using a threshold of 150 for dark targets and 180 for light. The binary images were subjected to the Canny algorithm for edge extraction

(Canny, 1986), then dilated and eroded by one pixel to consolidate the edges. The contours were extracted and simplified using the Ramer-Douglas-Peuker algorithm (Douglas & Peucker, 1973; Ramer, 1972) and filtered for "squareness." Squareness was given when:

- a geometry contained 4 vertices;
- each angle was within 10% of perpendicular;
- the ratios between the lengths of pairs of opposing sides were within 20%;
- and the total area was greater than 4000 pixels (arbitrarily selected to exclude random noise in the scene).

All images in which a target was not detected were discarded. The filename, distance from the image center and area of each target were written to a comma-separated values (CSV) file. As a single target could appear in multiple frames over the course of a single trial, the remaining images were examined manually and, for each trial and target, the frame in which a target was nearest the image centre was marked for preservation in the spreadsheet. The centre-most target was selected because, while the `undistort` method removes camera distortion, it does not address scale distortion; the image nearest the centre of the frame exhibits the least scale distortion.

The CSV file was imported into a relational database (PostgreSQL) for where the mean, standard deviation and coefficient of variance of the square root of the area of the targets were calculated. The square root was used to re-linearize the relationship between that platform altitude and the sample values (the average side length would also have sufficed). Results are shown in tables 5.1 and 5.2. It is clear from the table that the trials with terrain following exhibit the least variability in image scale.

Figure 5.7: Light and dark target extraction results.

Flight	Coefficient of Variation	Std. Dev. (m)	Mean (m)	Following
6	0.19	26.25	135.47	Y
1	0.21	24.54	115.31	Y
8	0.21	27.46	131.57	Y
3	0.24	26.77	112.20	Y
4	0.28	39.85	144.21	N
2	0.29	41.44	144.29	N
5	0.29	44.46	150.82	N
7	0.31	44.20	142.39	N

Table 5.1: Coefficient of variation, mean and standard deviation of target area, ordered by CoV. November 7, 2019.

To prove that the terrain-following system truly acheives the objectives of this research, a significant difference must be found between the following and no-following sample sets on each date. The F-test performs this function but is highly sensitive to the normality of the sample distribution (Box, 1953; Markowski & Markowski, 1990). The Shapiro-Wilk test (Shapiro & Wilk, 1965) provides a way of verifying the normality of datasets as small as $n = 3$. The test statistic, W, is found by,

Flight	Coefficient of Variation	Std. Dev. (m)	Mean (m)	Following
2	0.12	17.13	142.00	Y
3	0.13	19.34	150.24	Y
7	0.15	22.09	145.47	Y
6	0.23	32.72	141.26	Y
1	0.32	32.66	102.03	N
4	0.33	35.76	107.68	N
5	0.34	35.51	105.84	N

Table 5.2: Coefficient of variation, mean and standard deviation of target area, ordered by CoV. November 20, 2019.

$$W = \frac{(\sum_{i=1}^{n} a_i x_{(i)})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, \tag{5.1}$$

where $x_{(i)}$ is the i-th order sample value, $\bar{x}$ the sample mean, and a_i a constant calculated from the parameters of the ideal distribution and the sample size. See Shapiro and Wilk (1965) for tables of constants and W-threshold values for $n = 3$ to $n = 50$. The W-statistic is compared to the W-threshold from the table for the appropriate α level. If the W-statistic is less than the W-threshold at the selected significance level, the null hypothesis is rejected and normality cannot be assumed. The W-statistics for both test dates are given in table 5.4. For three of the four trials, a very small p-value suggests that the null hypothesis should be rejected. For the remaining trial – with following on November 20 – the null hypothesis is not rejected, but the evidence is fairly weak. The histograms in figure 5.8 also do not inspire confidence in the suggestion that the results follow a normal distribution. In the absence of good evidence for normality, that assumption is abandoned.

Levene (1960) provides a non-parametric test for the analysis of variance using the sample mean, while the Brown-Forsyth test (Brown & Forsythe, 1974) uses the median for improved robustness. Test statistic W is calculated by,

Date	Following	n	W	p	W-threshold (0.05)
Nov. 7, 2019	Y	19	0.912	0.081	0.901
Nov. 7, 2019	N	20	0.809	0.001	0.905
Nov. 20, 2019	Y	20	0.965	0.642	0.905
Nov. 20, 2019	N	15	0.830	0.009	0.881

Table 5.3: Results of Shapiro-Wilks test, both dates, following and no following. The null hypothesis (normality) is not rejected for the trials with following.

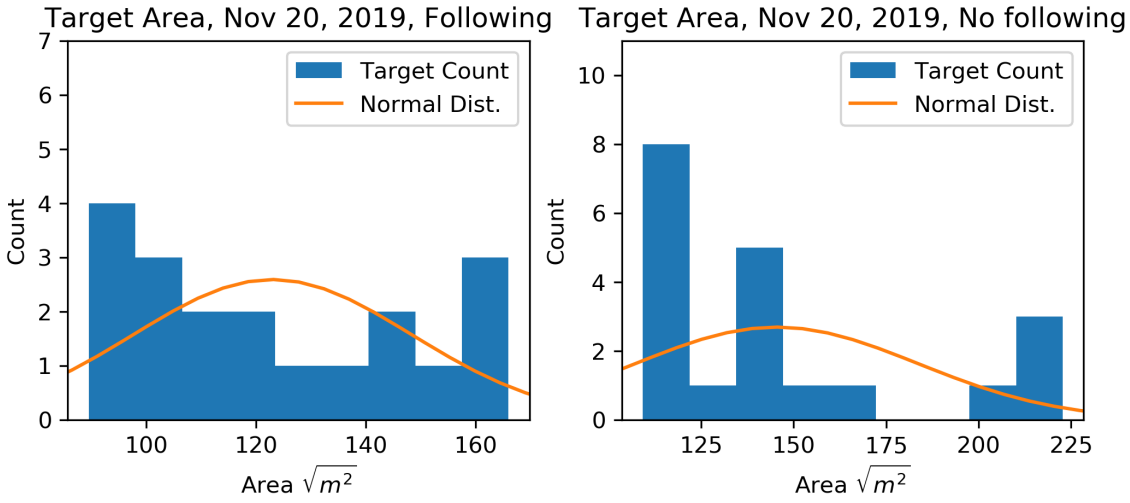
$$W = \frac{(N-k)}{(k-1)} \frac{\sum_{i=1}^{k} N_i (Z_i - Z_{..})^2}{\sum_{i=1}^{k} \sum_{j=1}^{N_i} (Z_{ij} - Z_{i.})^2}, \tag{5.2}$$

where k is the number of groups (2); N_i is the number of cases in group i; N is the number of cases; Z_{ij} is the absolute difference of the i^{th} element of the j^{th} group and the group mean; $Z_{i.}$ is the mean of Z_{ij} for each group; and $Z_{..}$ is the mean of all Z_{ij}. Z_{ij} may use the mean (Levene's test) or median (the Brown-Forsyth test).

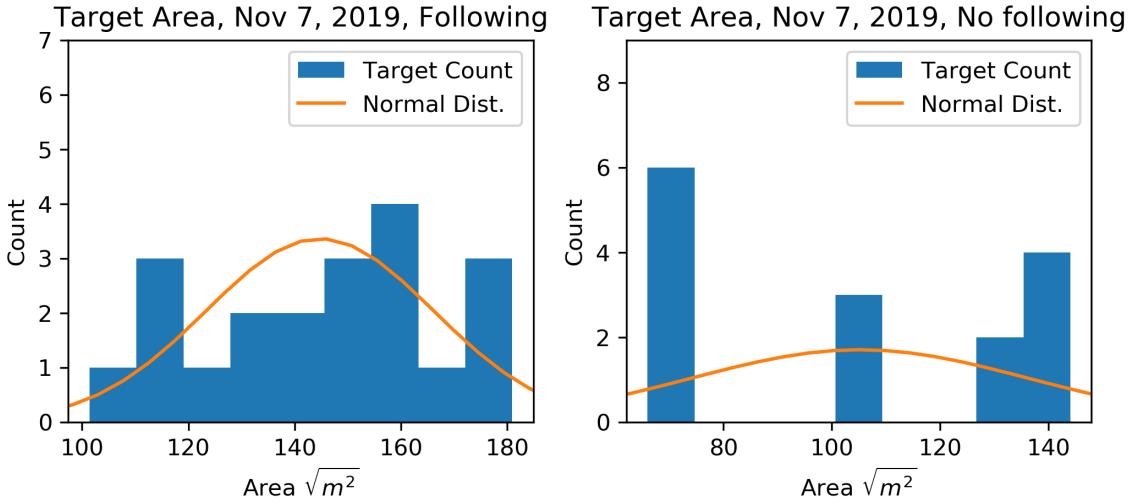
Table 5.4 shows the results of a comparison between the following and non-following trials on each date. Here, the null hypothesis is that the variances between the following and non-following trials are equal. On November 20, the W-statistic exceeds the critical value in both tests with $\alpha = 0.028$ suggesting that the variances are not equal. On November 7, the null hypothesis cannot be rejected. Possibly, this is due to the lower terrain relief of the November 7 trials, or the small number of test cases. Of course, it is also possible that the strength of the effect simply did not rise to significance on that day.

5.3 Modelling the Ideal Trajectory

Though it was not possible to perfect the performance of the terrain following system in the field, the flight trials and instrumentation did provide enough data and in-



(a) Target area histograms, November 7, 2019.



(b) Target area histograms, November 20, 2019.

Figure 5.8: Target area histograms, with following and without. With hypothetical normal distribution.

sight to model the performance of the system. In general, there are three competing interests in parameter selection:

1. to minimize the root mean squared error between the trajectory and the imaged surface;

2. to keep the vertical velocity within bounds; and

Date	k	N	Test	W	p	F crit.
Nov. 7, 2019	2	39	Levene	1.68	0.203	4.11
Nov. 7, 2019	2	39	Brown-Forsyth	1.08	0.305	4.11
Nov. 20, 2019	2	35	Levene	5.59	0.028	4.14
Nov. 20, 2019	2	35	Brown-Forsyth	5.28	0.028	4.14

Table 5.4: Results of Levene's and Brown-Forsyth tests, with critical values from F-distribution.

3. to keep the vertical acceleration within bounds.

The A3 controller limits the vertical velocity to $\pm 5\,\mathrm{ms}^{-2}$. At the maximum vehicle weight of 15.1 kg, the maximum positive acceleration of the Matrice 600 is $9.873\,\mathrm{ms}^{-2}$ and the minimum is $-g$, or approximately $-9.807\,\mathrm{ms}^{-2}$. Figures 5.9 and 5.10 show the effects of parameter selection on the three constraints. For each plot, the vehicle's forward velocity is set to $4\,\mathrm{ms}^{-1}$, and the segmentation distance to 0.25 m. The dominant factor affecting RMSE is the α distance, or bin size, which tends to smooth the surface of the point cloud prior to filtering. Window size and α both play a role in reducing the velocity and acceleration requirements, by increasing the smoothing effect of the Savitzky-Golay filter. The velocity and acceleration converge asymptotically towards zero as both parameters are increased.

The problem of determining an ideal parameter set for the terrain following system is known as a multi-objective optimization problem, and the set of outputs that satisfy the constraints as the solution space. The frontier that divides the solution space from all non-solutions is the Pareto front. These optimization problems are difficult, and may have no single solution. This thesis will not explore the issue in depth, however the literature provides some direction for reducing the complexity of the problem. In particular, the ϵ-constraint method (Yang, 2014) divides the solution space such that a solution at the location of ϵ becomes a global optimum (within a new domain).

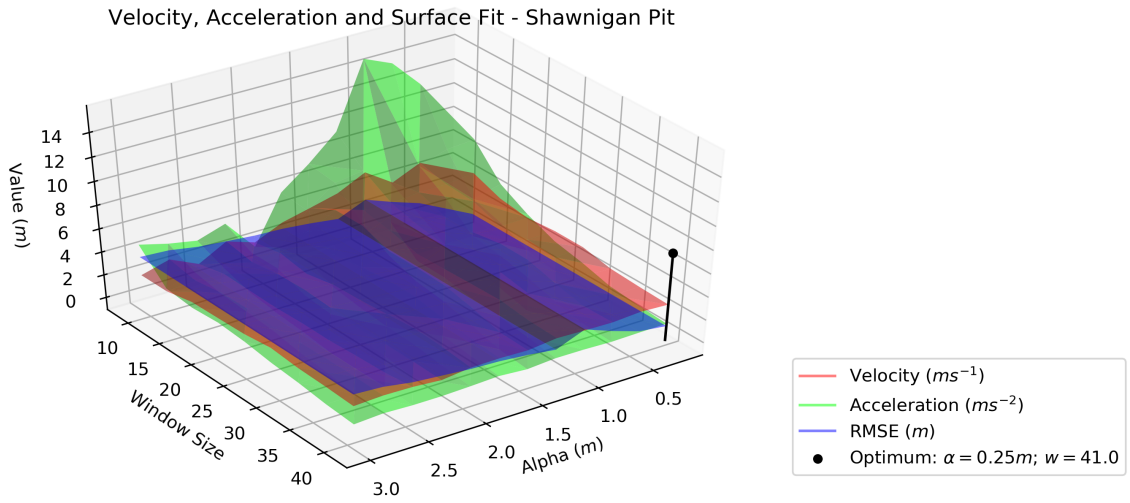


Figure 5.9: 3D plot of acceleration, velocity and RMSE for various parameter combinations on the bare ground at the Shawnigan pit study site.

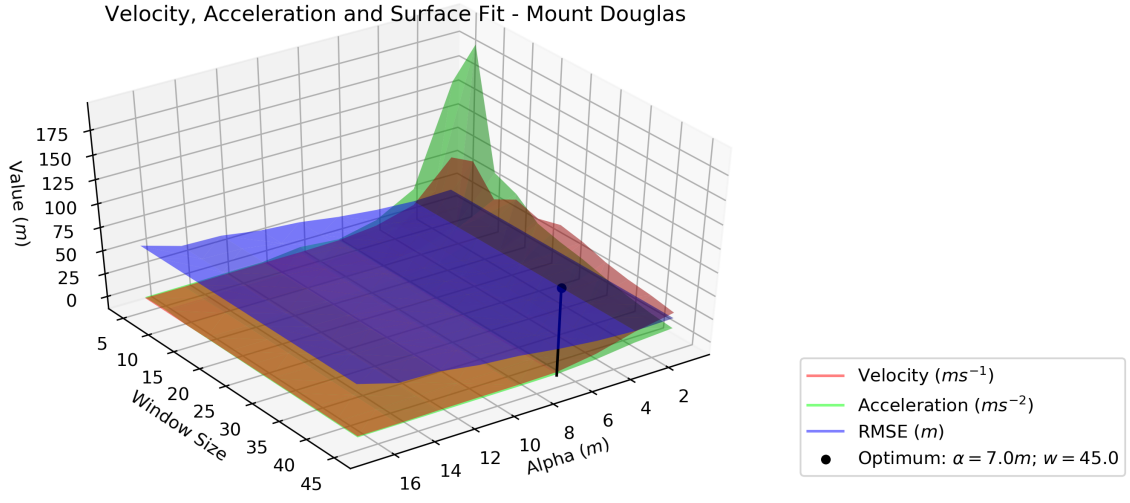


Figure 5.10: 3D plot of acceleration, velocity and RMSE for various parameter combinations on the mature Douglas fir canopy at Mount Douglas.

In the present case, if the solution space is split where vertical velocity is less than $5\,\mathrm{ms}^{-1}$. With this constraint, at no point does the acceleration exceed its limit and the only remaining free variable is the RMSE. The user can then select parameters that minimize the RMSE without exceeding the other limits.

The "Optimum" markers in figures 5.9 and 5.10 show the parameters which meet

the velocity threshold and minimize the RMSE. The markers are shown at the edges of the plots because the velocity and acceleration decline indefinitely as the window size increases. In both instances the window size could be safely decreased (which improves computational performance and fidelity to the surface) without exceeding the constraints. Naturally, a safety margin should be added to the velocity and acceleration constraints, however the parameter selection is so dependent on the surface characteristics and other mission objectives that the operator will have to use their experience and intuition to divine the appropriate configuration until such time as a rubric or autonomous intelligence is developed for that purpose.

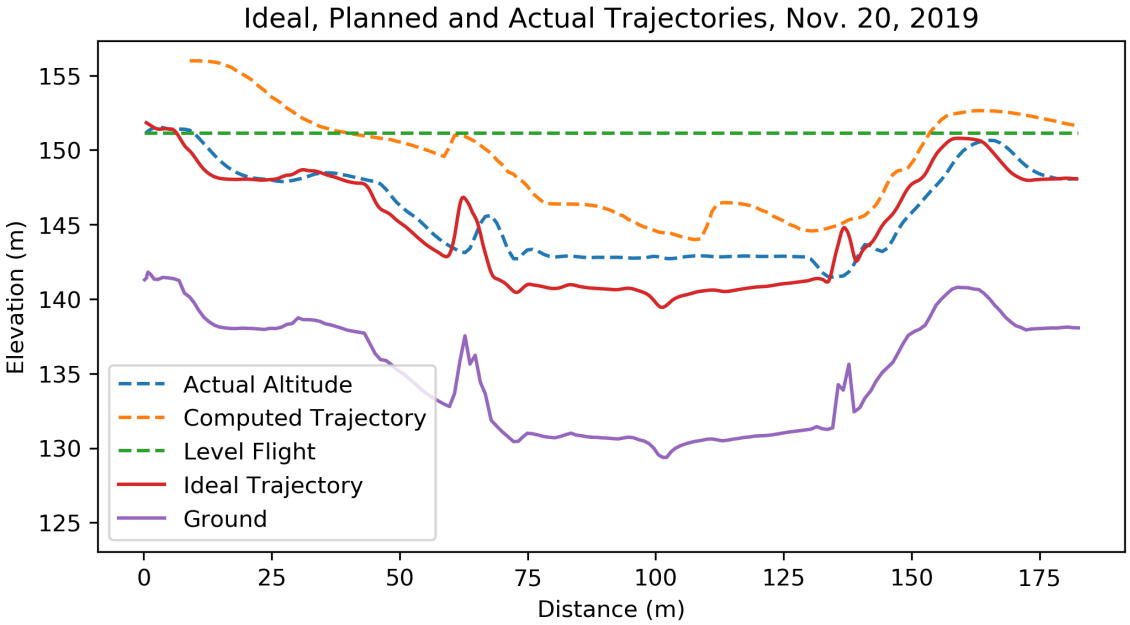


Figure 5.11: Comparison of actual, planned and idealized trajectories.

For five trials, the RMSE values given in table 5.5 give some idea of the fit of the actual, computed, and idealized trajectories to the terrain, as well as a comparison to level flight. The range of velocity and acceleration are given for the computed and ideal trajectories. Note that the This exercise was not performed on the November 7 data due to intermittent recording of the nadir laser ranges.

Trial	Type	RMSE	n	Velocity (ms^{-1})		Acceleration (ms^{-2})	
				Min.	Max.	Min.	Max.
1	Level	7.13	178	-	-	-	-
1	Ideal	0.22	145	-2.03	2.24	-1.56	1.36
1	Computed	4.87	166	-0.161	0.211	-0.0397	0.0318
1	Actual	1.69	178	-	-	-	-
2	Level	7.43	174	-	-	-	-
2	Ideal	0.207	140	-2.23	2.3	-1.77	1.26
2	Computed	4.99	163	-0.166	0.169	-0.0155	0.0264
2	Actual	1.73	174	-	-	-	-
3	Level	7.82	179	-	-	-	-
3	Ideal	0.176	141	-1.64	2.5	-2.07	0.911
3	Computed	5.32	164	-0.152	0.205	-0.0211	0.0397
3	Actual	2.62	179	-	-	-	-
4	Level	7.54	174	-	-	-	-
4	Ideal	0.179	144	-1.81	2.46	-1.88	1.09
4	Computed	4.85	163	-0.174	0.187	-0.0208	0.0244
4	Actual	1.8	174	-	-	-	-
5	Level	7.61	181	-	-	-	-
5	Ideal	0.266	149	-1.82	2.21	-1.44	1.06
5	Computed	4.97	166	-0.142	0.163	-0.0124	0.0181
5	Actual	1.9	181	-	-	-	-

Table 5.5: Comparison of RMSE, velocities and ranges of level flight versus ideal, computed and actual trajectories.

5.4 Summary

A subjective examination of the flight data reveals that the system did *work* in some sense: during the November 7 trials, the system computed and navigated a trajectory using a degenerate, unrepresentative point cloud; on November 20, the system failed to compute a trajectory, yet still followed the terrain in failure mode using the nadir laser. The statistical analysis of the target imagery and the trajectory comparison reveal that the data-quality objectives were met during both trials.

To the extent that the system failed, it does not appear to have been a conceptual failure, but rather a failure in engineering. The comparison of trajectories shows

that, though the computed and actual trajectories both achieve the remote-sensing objective to an extent, the ideal modelled trajectory radically reduces the divergence of the trajectory with respect to the terrain and would further improve data quality.

An analysis of the interplay between the system parameters and mission constraints (RMSE, velocity and acceleration) shows that the selection of appropriate parameters can be approached systematically. At no point did the vehicle exceed its dynamic limitations – i.e., it did not crash – in spite of the failures that occurred. This is likely due to the low terrain relief and gradient, and the low horizontal velocity of the vehicle; but this analysis suggests a starting point for the formulation of a strategy for determining the appropriate configuration during the mission planning stage. It would also prove invaluable to implement a flight data record of the vertical velocities and accelerations experienced by the aircraft, to validate the parameter-selection model empirically.

Chapter 6

Conclusions

RPAS represent an exciting new way for researchers to perform remote sensing surveys with lower financial and regulatory overhead, and at levels of detail previously unknown. The technological tributaries of RPAS research are evolving rapidly in terms of both the hardware and software, but research into autonomous control has tended to focus on artificial intelligence and biorobotic approaches that mimic living beings and their goals – navigating three-dimensional, dynamic spaces. Much of the relevant research had been performed in the 20^{th} century for military uses, and in the automated underwater vehicle (AUV) field.

This research has attempted to address the need for control system that satisfies remote sensing objectives. This is a highly constrained problem space: the vehicle's path over a surface is predetermined; it has only to find an optimal vertical trajectory that obeys the vehicle's dynamic limitations and minimizes the variance in specific measures of data quality. But current terrain following systems are either reactive and cannot anticipate terrain variations, or require a pref-flight or the pre-existence of a surface model which may be inaccurate or out-of-date.

The solution has been envisioned signal processing terms: if the terrain as con-

ceived as a signal, signal-processing techniques can extract a representative function to serve – after translation – as a trajectory. If the function is smooth and differentiable, its first (vertical velocity) and second (vertical acceleration) derivatives can be checked against the aircraft's limitations. This ensures both a safe flight and the maintenance of data quality.

After consideration of a variety of methods for extracting the terrain signal, the Savitzy-Golay filter was selected. This is a convolution kernel that pre-computes coefficients for a selected degree (cubic, in this case), derivative and window size. For any point in an evenly-spaced point set, the function value, first and second derivatives can be calculated in constant time.

The Savitzky-Golay filter was implemented in a terrain-following flight controller to consume points produced by a forward-facing laser rangefinder and control a RPAS in the field. The controller was able to load a user-created flight plan and navigate autonomously by receiving telemetry from and sending commands to a DJI Matrice 600 hexacopter. System performance was gauged by a statistical analysis of imagery collected during the flight trials and by comparing telemetric data to the terrain model.

Though it proved difficult to transform laser ranges into an accurate Cartesian surface representation, the system achieved its objectives in both predictive and reactive (failure) modes. A wealth of insight was gathered from flight data recorded by the controller, pointing the way towards improvements that would yield an effective real-time terrain following solution. The test flights produced enough data to model the ideal Savitzky-Golay trajectory and demonstrate a substantial improvement over both the actual trajectory and level-flight surveys.

Future Work

The two points of failure in this work have been the transformation of LiDAR range data into point clouds, and the projection of points into two dimensions. The first problem is resolved by tighter coupling between the inertial sensor and the projection algorithm; the second by the projection of into three dimensions before surface trajectory extraction. Both problems can be resolved by better application of existing engineering techniques, in particular by careful synchronization of the laser pulses and IMU measurements.

To the extent that the timing issue is hardware-induced, it could be resolved either by connecting the lasers directly to the GPIO pins of the Raspberry Pi, or through the use of a pulse per second (PPS) signal sent to the Teensy to update its internal clock. The latter solution would be preferable, as the Teensy is able to collect ranges in real time, and having a separate instrument controller preserves the modularity of the system.

A larger challenge concerns the selection of system parameters in relation to the characteristics of the surface and the mission objectives. The vineyard example is fairly trivial and consists of the interaction of two signals: the regular, high-frequency spacing of rows, and the lower-frequency relief of terrain. Other surfaces are both more complex and vary at a much larger magnitude relative to the platform altitude.

In the vineyard example, the magnitude of vine-row signal would be approximately 2 m, or about 7% of the typical 30 m platform altitude, and the row frequency – approximately 4 m – is high enough that multiple cycles can be convolved together in one pass of the kernel. A mixed-age Douglas fir canopy (figure 6.1), by contrast, presents a signal with a magnitude up to 30 m, or 25% of the maximum altitude of an RPAS under Canadian law without a special flight operations certificate (SFOC). The multi-spectral nature and larger scale of a forest canopy may prove challenging

Figure 6.1: Mixed-age second-growth Douglas fir canopy.

for the convolution routine due to the limited range of the laser and the resulting limit on the spatial extent of the kernel. It could prove difficult, with the equipment used in this research, to extract a sufficiently smooth trajectory from such a surface. An old-growth coastal rain forest canopy – with a magnitude up to 90 m and lower frequency in proportion – may well be impossible to negotiate using this technology.

Further research would seek a systematic approach to determining the optimal set of parameters for a variety of surface characteristics, including the forward velocity, laser angle, α or bin size and Savitzky-Golay kernel window size. This process should, perhaps more importantly, reveal the theoretical limits of the system and provide the operator an objective guide as to its suitability for specific research objectives.

Acronyms

AI aritificial intelligence. 26

ANN artificial neural network. 25

API application programming interface. 45

AUV automated underwater vehicle. 25, 26, 113

CMOS complementary metal-oxide-semiconductor. 8

CSV comma-separated values. 103

DEM digital elevation model. 3, 94, 97

DTM digital terrain model. 3

GLONASS global navigation satellite system. 45

GPIO general-purpose input/output. 51, 115

GPS global positioning system. 45, 73, 81, 94, 97

HLRG Hyperspectral-LiDAR Research Group. 53

IDE integrated development environment. 49

IMU inertial measurement unit. 24, 115

INS inertial navigation system. 25, 73, 97

IPC inter-process communication. 82

LiDAR light detection and ranging. viii, 2, 3, 7, 8, 13, 14, 17, 18, 31, 115

MEMS microelectromechanical sensor. 2

ML machine learning. 26

PFA probability of false alarm. 16

PID proportional integral derivative. viii, 26, 29, 30, 81, 82

PPS pulse per second. 115

PSP probability of single pulse detection. 16

PTAM panel tracking and mapping. 23

PWM pulse width modulation. 51

RADAR radio detection and ranging. 7, 19

RMS root mean square. 61

RMSE root mean square error. x, 108–110, 112

RPAS remotely piloted aerial system. iii, 1–3, 6, 8, 10, 11, 15, 18, 20, 21, 23, 25–27, 36, 45, 113–115

RPM revolutions per minute. 28

RTK real-time kinematic GPS. 73, 81

SFOC special flight operations certificate. 115

SG Savitzky-Golay. viii, ix, 40–42, 69–71, 91

SLAM simultaneous location and mapping. 23

SNR signal-to-noise ratio. 3, 9, 10, 12, 14, 16

SoC system-on-a chip. 2

SONAR sound navigation and ranging. 22, 23

UART universal asynchronous receiver/transmitter. 50

UAV unmanned aerial vehicle. 1

USB universal serial bus. 49, 50

Bibliography

Amenta, N., Bern, M., & Eppstein, D. (1998). The Crust and the β-Skeleton: Combinatorial Curve Reconstruction. *Graphical Models and Image Processing*, *60*(2), 125–135. https://doi.org/10.1006/gmip.1998.0465

Andrew, A. (1979). Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, *9*(5), 216–219. https://doi.org/10.1016/0020-0190(79)90072-3

Artale, V., Collotta, M., Milazzo, C., Pau, G., & Ricciardello, A. (2016). An Integrated System for UAV Control Using a Neural Network Implemented in a Prototyping Board. *Journal of Intelligent & Robotic Systems*, *84*(1-4), 5–19. https://doi.org/10.1007/s10846-015-0324-x

Asaeedi, S., Didehvar, F., & Mohades, A. (2017). α -Concave hull, a generalization of convex hull. *Theoretical Computer Science*, *702*, 48–59. https://doi.org/10.1016/j.tcs.2017.08.014

Asseo, S. J. (1988). Terrain following/terrain avoidance path optimization using the method of steepest descent. *Aerospace and Electronics Conference, 1988. NAECON 1988., Proceedings of the IEEE 1988 National*, 1128–1136 vol.3. https://doi.org/10.1109/NAECON.1988.195148

Aubrecht, C., Höfle, B., Hollaus, M., Köstl, M., Steinnocher, K., & Wagner, W. (2010). Vertical roughness mapping - ALS based classification of the vertical vegetation

structure in forested areas. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *38*(Part 7B), 35–40.

Avadhanula, S., Wood, R., Campolo, D., & Fearing, R. (2002). Dynamically tuned design of the MFI thorax, In *Proceedings 2002 ieee international conference on robotics and automation (cat. no.02ch37292)*, IEEE. https://doi.org/10.1109/ROBOT.2002.1013338

Avery, T. E., & Berlin, G. L. (1992). *Fundamentals of Remote Sensing and Airphoto Interpretation* (5th). New York, NY, Macmillan.

Bagherian, M. (2018). Unmanned Aerial Vehicle Terrain Following/Terrain Avoidance/Threat Avoidance trajectory planning using fuzzy logic. *Journal of Intelligent and Fuzzy Systems*, *34*(3), 1791–1799. https://doi.org/10.3233/JIFS-161977

Barbosa, J. M., Broadbent, E. N., & Bitencourt, M. D. (2014). Remote Sensing of Aboveground Biomass in Tropical Secondary Forests: A Review. *International Journal of Forestry Research*, *2014*arXiv 715796, 1–14. https://doi.org/10.1155/2014/715796

Barducci, A., Guzzi, D., Marcoionni, P., & Pippi, I. (2007). Assessing noise amplitude in remotely sensed images using bit-plane and scatterplot approaches. *IEEE Transactions on Geoscience and Remote Sensing*, *45*(8), 2665–2675. https://doi.org/10.1109/TGRS.2007.897421

Behroozpour, B., Sandborn, P. A. M., Wu, M. C., & Boser, B. E. (2017). Lidar System Architectures and Circuits. *IEEE Communications Magazine*, *55*(10), 135–142. https://doi.org/10.1109/MCOM.2017.1700030

Ben-Dor, E., Malthus, T., Plaza, A., & Schläpfer, D. (2012). Hyperspectral Remote Sensing. *Airborne Measurements for Environmental Research: Methods and Instruments*, 413–454.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, *18*(9), arXiv arXiv:1011.1669v3, 509–517. https://doi.org/10.1145/361002.361007

Berger, M., Alliez, P., Tagliasacchi, A., Seversky, L. M., Silva, C. T., Levine, J. a., & Sharf, A. (2014). State of the Art in Surface Reconstruction from Point Clouds. *Eurographics STAR (Proc. of EG'14)*, 161–185. https://doi.org/dx.doi.org/10.2312/egst.20141040

Berger, M., Tagliasacchi, A., Seversky, L. M., Alliez, P., Guennebaud, G., Levine, J. A., Sharf, A., & Silva, C. T. (2017). A Survey of Surface Reconstruction from Point Clouds. *Computer Graphics Forum*, *36*(1), 301–329. https://doi.org/10.1111/cgf.12802

Beyeler, A., & Floreano, D. (2009). optiPilot : control of take-off and landing using optic flow. *European Micro Air Vehicle Conference and Competition 2009 (EMAV 2009)*.

Bovio, E., Cecchi, D., & Baralli, F. (2006). Autonomous underwater vehicles for scientific and naval operations. *Annual Reviews in Control*, *30*(2), 117–130. https://doi.org/10.1016/j.arcontrol.2006.08.003

Box, G. E. P. (1953). Non-Normality and Tests on Variances. *Biometrika*, *40*(3/4), 318. https://doi.org/10.2307/2333350

Brasington, J., Vericat, D., & Rychkov, I. (2012). Modeling river bed morphology, roughness, and surface sedimentology using high resolution terrestrial laser scanning. *Water Resources Research*, *48*(11), 1–18. https://doi.org/10.1029/2012WR012223

Brisco, B., Murnaghan, K., Chichagov, A., & White, L. (2015). *Monitoring Terrestrial Wetlands and Watershed Basins with RCM Rapid Revisit and CCD Mode*

(tech. rep.). Canadian Space Agency
NULL.

Brown, M. B., & Forsythe, A. B. (1974). Robust Tests for the Equality of Variances. *Journal of the American Statistical Association*, *69*(346), 364–367. https://doi.org/10.1080/01621459.1974.10482955

Burns, H. N. (1991). System design of a pulsed laser rangefinder. *Optical Engineering*, *30*(3), 323. https://doi.org/10.1117/12.55801

Burrough, P. A. (1981). Fractal dimensions of landscapes and other environmental data. *Nature*, *294*(5838), 240–242. https://doi.org/10.1038/294240a0

Burrough, P. A. (1983). Multiscale sources of spatial variation in soil. I. The application of fractal concepts to nested levels of soil variation. *Journal of Soil Science*, *34*(3), 577–597. https://doi.org/10.1111/j.1365-2389.1983.tb01057.x

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-8*(6), 679–698. https://doi.org/10.1109/TPAMI.1986.4767851

Chakravorty, P. (2018). What Is a Signal? [Lecture Notes]. *IEEE Signal Processing Magazine*, *35*(5), 175–177. https://doi.org/10.1109/MSP.2018.2832195

Clark, M., & Roberts, R. C. (2017). Autonomous quadrotor terrain-following with a laser rangefinder and gimbal system. *Proceedings of IEEE Sensors*, *2017-Decem*, 1–3. https://doi.org/10.1109/ICSENS.2017.8234268

Cook, K. L. (2007). The silent force multiplier: The history and role of UAVs in warfare. *IEEE Aerospace Conference Proceedings*. https://doi.org/10.1109/AERO.2007.352737

Côté, J.-F., Fournier, R. A., & Egli, R. (2011). An architectural model of trees to estimate forest structural attributes using terrestrial LiDAR. *Environmental*

Modelling & Software, *26*(6), 761–777. https://doi.org/10.1016/j.envsoft.2010.12.008

Côté, J.-F., Fournier, R. A., Frazer, G. W., & Olaf Niemann, K. (2012). A fine-scale architectural model of trees to enhance LiDAR-derived measurements of forest canopy structure. *Agricultural and Forest Meteorology*, *166-167*, 72–85. https://doi.org/10.1016/j.agrformet.2012.06.007

Couteron, P., Pelissier, R., Nicolini, E. A., & Paget, D. (2005). Predicting tropical forest stand structure parameters from Fourier transform of very high-resolution remotely sensed canopy images. *Journal of Applied Ecology*, *42*(6), 1121–1128. https://doi.org/10.1111/j.1365-2664.2005.01097.x

Cryderman, C., Bill Mah, S., & Shufletoski, A. (2015). Evaluation of uav photogrammetric accuracy for mapping and earthworks computations. *Geomatica*, *68*(4), 309–317. https://doi.org/10.5623/cig2014-405

Da, T. K. F., Loriot, S., & Yvinec, M. (2018). {3D} Alpha Shapes. In {*cgal*} *user and reference manual* (4.13). CGAL Editorial Board. https://doc.cgal.org/4.13/Manual/packages.html#PkgAlphaShapes3Summary

de Berg, M., Cheong, O., van Kreveld, M., & Overmars, M. (2008). *Computational Geometry*. Berlin, Heidelberg, Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-77974-2

de Boor, C. (1978). *A Practical Guide to Splines* (F. John, J. P. LaSalle, L. Sirovich, & G. B. Whitham, Eds.; Vol. Volume 27). New York, NY, Springer-Verlag. https://doi.org/10.2307/2006241

de Boor, C. (2001). Calculation of the smoothing spline with weighted roughness measure. *Mathematical Models and Methods in Applied Sciences*, *11*(01), 33–41. https://doi.org/10.1142/S0218202501000726

de Boor, C. (2006). A draftsman's spline. http://pages.cs.wisc.edu/~deboor/draftspline.html

Dierckx, P. (1979). Surfit.f. http://www.netlib.org/dierckx/

Dierckx, P. (1993). *Curve and Surface Fitting with Splines*. New York, NY, USA, Oxford University Press, Inc.

DJI. (2017). MATRICE 600 Pro. *Shenzhen, Guangdong*, DJI. https://www.dji.com/es/matrice600-pro

DJI. (2018a). DJIFlightPlanner Software. Retrieved January 27, 2019, from https://www.djiflightplanner.com/

DJI. (2018b). Telemetry Topics. Retrieved October 7, 2019, from https://developer.dji.com/onboard-api-reference/group__telem.html

Douglas, D. H., & Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, *10*(2), 112–122. https://doi.org/10.3138/FM57-6770-U75U-7727

Drakos, N., & Moore, R. (2002). Cubic Spline Interpolation. Retrieved March 19, 2019, from http://www.physics.utah.edu/~detar/phys6720/handouts/cubic_spline/cubic_spline/node2.html

Du Preez, C. (2014). A new arc–chord ratio (ACR) rugosity index for quantifying three-dimensional landscape structural complexity. *Landscape Ecology*, *30*(1), 181–192. https://doi.org/10.1007/s10980-014-0118-8

Duckham, M., Kulik, L., Worboys, M., & Galton, A. (2008). Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern Recognition*, *41*(10), 3224–3236. https://doi.org/10.1016/j.patcog.2008.03.023

Edelsbrunner, H., & Mücke, E. P. (1994). Three-dimensional alpha shapes. *ACM Transactions on Graphics*, *13*(1), arXiv 9410208, 43–72. https://doi.org/10.1145/174462.156635

Feijun Song, & Smith, S. (2000). Design of sliding mode fuzzy controllers for an autonomous underwater vehicle without system model, In *Oceans 2000 mts/ieee conference and exhibition. conference proceedings (cat. no.00ch37158)*, IEEE. https://doi.org/10.1109/OCEANS.2000.881362

Fletcher, J. (1982). An Arithmetic Checksum for Serial Transmissions. *IEEE Transactions on Communications*, *30*(1), 247–252. https://doi.org/10.1109/TCOM.1982.1095369

Fu, C., Olivares-Mendez, M. A., Suarez-Fernandez, R., & Campoy, P. (2014). Monocular Visual-Inertial SLAM-based collision avoidance strategy for Fail-Safe UAV using Fuzzy Logic Controllers: Comparison of two Cross-Entropy Optimization approaches. *Journal of Intelligent and Robotic Systems: Theory and Applications*, *73*(1-4), 513–533. https://doi.org/10.1007/s10846-013-9918-3

Funk, J. (1976). Optimal-path precision terrain following system, In *Guidance and control conference*, Reston, Virigina, American Institute of Aeronautics; Astronautics. https://doi.org/10.2514/6.1976-1958

Gibiansky, A. (2012). Quadcopter Dynamics , Simulation , and Control Introduction Quadcopter Dynamics, 1–18. http://andrew.gibiansky.com/blog/physics/quadcopter-dynamics/

Gilmore, M. S., Wilson, E. H., Barrett, N., Civco, D. L., Prisloe, S., Hurd, J. D., & Chadwick, C. (2008). Integrating multi-temporal spectral and structural information to map wetland vegetation in a lower Connecticut River tidal marsh. *Remote Sensing of Environment*, *112*(11), 4048–4060. https://doi.org/10.1016/j.rse.2008.05.020

Goheen, K. R., & Jefferys, E. R. (1990). Multivariable Self-Tuning Autopilots for Autonomous and Remotely Operated Underwater Vehicles. *IEEE Journal of Oceanic Engineering*, *15*(3), 144–151. https://doi.org/10.1109/48.107142

Gupta, R. P. (2018). *Remote Sensing Geology* (Vol. 18). Berlin, Heidelberg, Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-55876-8

Hancock, S., Anderson, K., Disney, M., & Gaston, K. J. (2017). Measurement of fine-spatial-resolution 3D vegetation structure with airborne waveform lidar: Calibration and validation with voxelised terrestrial lidar. *Remote Sensing of Environment*, *188*, 37–50. https://doi.org/10.1016/j.rse.2016.10.041

Hancock, S., Armston, J., Li, Z., Gaulton, R., Lewis, P., Disney, M., Mark Danson, F., Strahler, A., Schaaf, C., Anderson, K., & Gaston, K. J. (2015). Waveform lidar over vegetation: An evaluation of inversion methods for estimating return energy. *Remote Sensing of Environment*, *164*, 208–224. https://doi.org/10.1016/j.rse.2015.04.013

Hanley, J. T. (1977). Fourier analysis of the Catawba Mountain knolls, Roanoke county, Virginia. *Journal of the International Association for Mathematical Geology*, *9*(2), 159–163. https://doi.org/10.1007/BF02312510

Hart, P., Nilsson, N., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, *4*(2), 100–107. https://doi.org/10.1109/TSSC.1968.300136

Headwall Photonics Inc. (2017). Headwall Nano-Hyperspec VNIR® & DJI MATRICE™ 600 Pro. *Bolton, MA*, Headwall Photonics, Inc. http://files.constantcontact.com/8abb5327be/b98849a7-3009-4d8b-9d59-67e3d1c475c1.pdf

Healey, A., & Lienard, D. (1993). Multivariable sliding mode control for autonomous diving and steering of unmanned underwater vehicles. *IEEE Journal of Oceanic Engineering*, *18*(3), 327–339. https://doi.org/10.1109/JOE.1993.236372

Hérissé, B., Hamel, T., Mahony, R., & Russotto, F. X. (2010). A terrain-following control approach for a VTOL Unmanned Aerial Vehicle using average optical flow. *Autonomous Robots*, *29*(3-4), 381–399. https://doi.org/10.1007/s10514-010-9208-x

Hoare, C. a. R. (1961). Quicksort: Algorithm 64. *Communications of the ACM*, *4*(7), 321. https://doi.org/10.1145/366622.366647

Hoffmann, G. M., Huang, H., Waslander, S. L., & Tomlin, C. J. (2007). Quadrotor helicopter flight dynamics and control: Theory and experiment. *American Institute of Aeronautics and Astronautics*, 1–20. https://doi.org/10.2514/6.2007-6461

Hopkinson, C., Lovell, J., Chasmer, L., Jupp, D., Kljun, N., & van Gorsel, E. (2013). Integrating terrestrial and airborne lidar to calibrate a 3D canopy model of effective leaf area index. *Remote Sensing of Environment*, *136*, 301–314. https://doi.org/10.1016/j.rse.2013.05.012

Huang, M.-j., Shyue, S.-w., Lee, L.-h., & Kao, C.-c. (2008). A Knowledge-based Approach to Urban Feature Classification Using Aerial Imagery with Lidar Data, *74*(12), 1473–1485.

Jalving, B. (1994). The NDRE-AUV Flight Control System. *IEEE Journal of Oceanic Engineering*, *19*(4), 497–501. https://doi.org/10.1109/48.338385

Juul, D. L., Mcdermott, M. E., Nelson, E. L., Barnett, D. M., & Williams, G. N. (1994). Submersible Control Using the Linear uadratic Gaussian with Loop Transfer ecovery Metha.

Klein, G., & Murray, D. (2007). Parallel Tracking and Mapping for Small AR Workspaces, In *2007 6th ieee and acm international symposium on mixed and augmented reality*, IEEE. https://doi.org/10.1109/ISMAR.2007.4538852

Kosari, A., Maghsoudi, H., Lavaei, A., & Ahmadi, R. (2015). Optimal online trajectory generation for a flying robot for terrain following purposes using neural network. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, *229*(6), 1124–1141. https://doi.org/10.1177/0954410014545797

Krachmanlick, F. M., Vetsch, G. J., & Wendl, M. J. (1968). Automatic flight control system for automatic terrain-following. *Journal of Aircraft*, *5*(2), 168–175. https://doi.org/10.2514/3.43925

Krislock, A., & Krislock, N. (2014). Resolving Histogram Binning Dilemmas with Binless and Binfull Algorithms, arXiv 1405.4958, 1–19. http://arxiv.org/abs/1405.4958

Kröger, T. (2010). *On-Line Trajectory Generation in Robotic Systems* (Vol. 58). Berlin, Heidelberg, Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-05175-3

Lancaster, P., & Šalkauskas, K. (1986). *Curve and Surface Fitting: An Introduction*. Academic Press. https://books.google.ca/books?id=eZIQAQAAIAAJ

Levene, H. (1960). Robust Tests for Equality of Variances. In I. Olkin & H. Hotelling (Eds.), *Contributions to probability and statistics: Essays in honor of harold hotelling* (pp. 278–292). Palo Alto, CA, Stanford University Press.

Li Qing. (1997). Aircraft route optimization using genetic algorithms, In *Second international conference on genetic algorithms in engineering systems*, IEE. https://doi.org/10.1049/cp:19971212

LightWare Optoelectronics. (2019). SF30 Accelerated laser rangefinder product manual. *Gauteng, South Africa*. http://documents.lightware.co.za/SF30%20-%20Laser%20Altimeter%20Manual%20-%20Rev%209.pdf

Livshitz, A., & Idan, M. (2018). Low-Cost Laser Range-Measurement-Based Terrain-Following Concept and Error Analysis. *Journal of Guidance, Control, and Dynamics*, *41*(4), 1006–1014. https://doi.org/10.2514/1.G002565

Livshitz, A., & Idan, M. (2019). Preview Control Approach for Laser-Range-Finder Based Terrain-Following. *IEEE Transactions on Aerospace and Electronic Systems*, *9251*(100), 1–1. https://doi.org/10.1109/TAES.2019.2933955

Lougovski, P., & Pooser, R. (2014). An Observed-Data-Consistent Approach to the Assignment of Bit Values in a Quantum Random Number Generator, arXiv 1404.5977. http://arxiv.org/abs/1404.5977

Lovell, J., Jupp, D., Newnham, G., Coops, N., & Culvenor, D. (2005). Simulation study for finding optimal lidar acquisition parameters for forest height retrieval. *Forest Ecology and Management*, *214*(1-3), 398–412. https://doi.org/10.1016/j.foreco.2004.07.077

Lu, P., & Pierson, B. L. (1995). Optimal aircraft terrain-following analysis and trajectory generation. *Journal of Guidance, Control, and Dynamics*, *18*(3), 555–560. https://doi.org/10.2514/3.21422

Mandelbrot, B. (1967). How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension. *Science*, *156*(3775), 636–638. http://www.jstor.org/stable/1721427

Markowski, C. A., & Markowski, E. P. (1990). Conditions for the Effectiveness of a Preliminary Test of Variance. *The American Statistician*, *44*(4), 322. https://doi.org/10.2307/2684360

Martín-alcón, S., Coll, L., Cáceres, M. D., Guitart, L., Cabré, M., & Just, A. (2015). Combining aerial LiDAR and multispectral imagery to assess postfire regeneration types in a Mediterranean forest. *Canadian Journal of Forest Research*, *866*(February), 856–866. https://doi.org/10.1139/cjfr-2014-0430

McKerrow, P. (2004). Modelling the Draganflyer four-rotor helicopter, In *Ieee international conference on robotics and automation, 2004. proceedings. icra '04. 2004*, IEEE. https://doi.org/10.1109/ROBOT.2004.1308810

Menon, P. K. A., Cheng, V. L., & Kim, E. (1991). Optimal trajectory synthesis for terrain-following flight. *Journal of Guidance, Control, and Dynamics*, *14*(4), 807–813. https://doi.org/10.2514/3.20716

Mian, A. A., & Wang, D.-b. (2008). Dynamic modeling and nonlinear control strategy for an underactuated quad rotor rotorcraft. *Journal of Zhejiang University-SCIENCE A*, *9*(4), 539–545. https://doi.org/10.1631/jzus.A071434

Mwakibinga, T., & Lee, J. (2005). *Altitude Control of an Unmanned Aerial Vehicle Using a Binaural Bat Echolocation System* (tech. rep.). Rensselaer Polytechnic Institute. Troy, NY. https://ece.umd.edu/merit/archives/merit2005/merit05_tech_reports/RITE_Mwakibinga_Lee.pdf

Nakassis, A. (1988). Fletcher's error detection algorithm: how to implement it efficiently and how toavoid the most common pitfalls. *ACM SIGCOMM Computer Communication Review*, *18*(5), 63–88. https://doi.org/10.1145/53644.53648

Netter, T., & Franceschini, N. (2002). A robotic aircraft that follows terrain using a neuromorphic eye. *2002 Ieee/Rsj International Conference on Intelligent Robots and Systems, Vols 1-3, Proceedings*, 129–134. https://doi.org/10.1109/IRDS.2002.1041376

Niemann, K. O., Frazer, G., Loos, R., Visintini, F., & Stephen, R. (2007). Integration of first and last return LiDAR with hyperspectral data to characterize forested environments., In *2007 ieee international geoscience and remote sensing symposium*, IEEE. https://doi.org/10.1109/IGARSS.2007.4423102

Niemann, K. O., Goodenough, D. G., Loos, R., Quinn, G., & Visintini, F. (2011). Remote sensing of forested environments: The effects of a radiometrically porous

and structurally complex surface, In *2011 3rd workshop on hyperspectral image and signal processing: Evolution in remote sensing (whispers)*, IEEE. https://doi.org/10.1109/WHISPERS.2011.6080964

Niemann, K. O., Quinn, G., Goodenough, D. G., Visintini, F., & Loos, R. (2012). Addressing the effects of canopy structure on the remote sensing of foliar chemistry of a 3-dimensional, radiometrically porous surface. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *5*(2), 584–593. https://doi.org/10.1109/JSTARS.2011.2179637

Niemann, K. O., Quinn, G., Stephen, R., Visintini, F., & Parton, D. (2015). Hyperspectral Remote Sensing of Mountain Pine Beetle with an Emphasis on Previsual Assessment. *Canadian Journal of Remote Sensing*, *41*(3), 191–202. https://doi.org/10.1080/07038992.2015.1065707

Nikolos, I. K., Zografos, E. S., & Brintaki, A. N. (2007). UAV Path Planning Using Evolutionary Algorithms. In J. S. Chahl, L. C. Jain, A. Mizutani, & M. Sato-Ilic (Eds.), *Innovations in intelligent machines - 1* (pp. 77–111). Berlin, Heidelberg, Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-72696-8_4

Orfanidis, S. J. (1996). *Introdution to Signal Processing*. Upper Saddle River, N.J., Prentice-Hall International. https://www.ece.rutgers.edu/~orfanidi/intro2sp/

Pearson, K. (1900). X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *50*(302), 157–175. https://doi.org/10.1080/14786440009463897

Perron, J. T., Kirchner, J. W., & Dietrich, W. E. (2008). Spectral signatures of characteristic spatial scales and nonfractal structure in landscapes. *Journal of Geophysical Research*, *113*(F4), F04003. https://doi.org/10.1029/2007JF000866

Rahim, M., & Móhammad-Bagher Malaek, S. (2011). Aircraft terrain following flights based on fuzzy logic. *Aircraft Engineering and Aerospace Technology*, *83*(2), 94–104. https://doi.org/10.1108/00022661111120980

Ramer, U. (1972). An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, *1*(3), 244–256. https://doi.org/10.1016/S0146-664X(72)80017-0

RCA Corporation. (1974). *RCA electro-optics handbook* (2nd ed.). Harrison, NJ, RCA/Commercial Engineering.

Reinsch, C. H. (1967). Smoothing by spline functions. *Numerische Mathematik*, *10*(3), 177–183. https://doi.org/10.1007/BF02162161

Rogaß, C., Spengler, D., Bochow, M., Segl, K., Lausch, A., Doktor, D., Roessner, S., Behling, R., Wetzel, H. U., & Kaufmann, H. (2011). Reduction of radiometric miscalibration-applications to pushbroom sensors. *Sensors*, *11*(6), 6370–6395. https://doi.org/10.3390/s110606370

Ruffier, F., & Franceschini, N. (2004). Visually guided micro-aerial vehicle: automatic take off, terrain following, landing and wind reaction, In *Ieee international conference on robotics and automation, 2004. proceedings. icra '04. 2004*, IEEE. https://doi.org/10.1109/ROBOT.2004.1307411

Ruyten, W. (1999). Smear correction for frame transfer charge-coupled-device cameras. *Optics Letters*, *24*(13), 878–880. https://doi.org/10.1364/OL.24.000878

Sánchez, J., Visioli, A., & Dormido, S. (2012). *PID Control in the third millenium*. https://doi.org/10.1007/978-1-4471-2425-2

Savitzky, A., & Golay, M. J. E. (1964). Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry*, *36*(8), 1627–1639. https://doi.org/10.1021/ac60214a047

Schott, J. R. (2007). *Remote sensing: the image chain approach* (2nd). New York, Oxford University Press.

Shaler, N. S. (1899). Spacing of rivers with reference to hypothesis of baseleveling. *Geological Society of America Bulletin*, *10*(1), 263–276. https://doi.org/10.1130/GSAB-10-263

Shapiro, S. S., & Wilk, M. B. (1965). An Analysis of Variance Test for Normality (Complete Samples). *Biometrika*, *52*(3/4), 591. https://doi.org/10.2307/2333709

Shoemake, K. (1985). Animating rotation with quaternion curves, In *Proceedings of the 12th annual conference on computer graphics and interactive techniques - siggraph '85*, New York, New York, USA, ACM Press. https://doi.org/10.1145/325334.325242

Shook, K., Pomeroy, J. W., Spence, C., & Boychuk, L. (2013). Storage dynamics simulations in prairie wetland hydrology models: Evaluation and parameterization. *Hydrological Processes*, *27*(13), 1875–1889. https://doi.org/10.1002/hyp.9867

Smith, C. G. M., Boyd, D. S., & Matthews, J. A. (2014). Encyclopedia of environmental change. *Choice Reviews Online*, *51*(12), 51–6500–51–6500. https://doi.org/10.5860/CHOICE.51-6500

Solberg, S., Næsset, E., Hanssen, K. H., & Christiansen, E. (2006). Mapping defoliation during a severe insect attack on Scots pine using airborne laser scanning. *Remote Sensing of Environment*, *102*(3-4), 364–376. https://doi.org/10.1016/j.rse.2006.03.001

Starling, R., & Stewart, C. (1971). The Development of Terrain Following Radar. *Aircraft Engineering and Aerospace Technology*, *43*(4), 13–15. https://doi.org/10.1108/eb034756

Thornley, D. J. (2006). Anisotropic Multidimensional Savitzky Golay kernels for Smoothing, Differentiation and Reconstruction. *Differentiation*, *2006/8*, 1–12.

Twigg, S., Calise, A., & Johnson, E. (2003). On-Line Trajectory Optimization for Autonomous Air Vehicles, In *Aiaa guidance, navigation, and control conference and exhibit*, Reston, Virigina, American Institute of Aeronautics; Astronautics. https://doi.org/10.2514/6.2003-5522

Valavanis, K. P. (Ed.). (2007). *Advances in Unmanned Aerial Vehicles*. Dordrecht, Springer Netherlands. https://doi.org/10.1007/978-1-4020-6114-1

Valavanis, K. P., & Vachtsevanos, G. J. (2015). *Handbook of Unmanned Aerial Vehicles* (K. P. Valavanis & G. J. Vachtsevanos, Eds.). Dordrecht, Springer Netherlands. https://doi.org/10.1007/978-90-481-9707-1

Villafranca, A. G., Corbera, J., Martín, F., & Marchán, J. F. (2012). Limitations of Hyperspectral Earth Observation on Small Satellites. *Journal of Small Satellites*, *1*(1), 19–29.

Waldock, M. (1995). Terrain following control of an unmanned underwater vehicle using artificial neural networks, In *Iee colloquium on 'control and guidance of remotely operated vehicles'*, IEE. https://doi.org/10.1049/ic:19950800

Wegman, E. J., & Wright, I. W. (1983). Splines in Statistics. *Journal of the American Statistical Association*, *78*(382), 351–365. https://doi.org/10.1080/01621459.1983.10477977

Xinyan Deng, Schenato, L., & Sastry, S. (2003). Model identification and attitude control for a micromechanical flying insect including thorax and sensor mod-

els, In *2003 ieee international conference on robotics and automation (cat. no.03ch37422)*, IEEE. https://doi.org/10.1109/ROBOT.2003.1241748

Yang, X.-S. (2014). Multi-Objective Optimization. In *Nature-inspired optimization algorithms* (pp. 197–211). Elsevier. https://doi.org/10.1016/B978-0-12-416743-8.00014-2

Zhao, K., & Popescu, S. (2009). Lidar-based mapping of leaf area index and its use for validating GLOBCARBON satellite LAI product in a temperate forest of the southern USA. *Remote Sensing of Environment*, *113*(8), 1628–1645. https://doi.org/10.1016/j.rse.2009.03.006