Automated flight planning for roof inspection using a face-based approach

by

Paul Sponagle

B.Eng. Carleton University, 2001

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in the Chester F. Carlson Center for Imaging Science
College of Science
Rochester Institute of Technology

July 17, 2017

Signature of the Author _____

Accepted by _____
Coordinator, M.S. Degree Program Date

CHESTER F. CARLSON CENTER FOR IMAGING SCIENCE
COLLEGE OF SCIENCE
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

M.S. DEGREE THESIS

The M.S. Degree Thesis of Paul Sponagle
has been examined and approved by the
thesis committee as satisfactory for the
thesis required for the
M.S. degree in Imaging Science

Dr. Carl Salvaggio, Thesis Advisor

Dr. Jan van Aardt

Dr. Jeff Pelz

Nina Raqueno

Date

Automated flight planning for roof inspection using a face-based approach

by

Paul Sponagle

Submitted to the
Chester F. Carlson Center for Imaging Science
in partial fulfillment of the requirements
for the Master of Science Degree
at the Rochester Institute of Technology

Abstract

The rapid proliferation of consumer small unmanned aerial systems (sUASs) has expanded ownership to include amateurs and professionals alike. These platforms in combination with numerous open source and proprietary applications tailored to gather aerial imagery and generate 3D point clouds and meshes from aerial imagery, have made 3D modeling available to anyone who can afford an entry-level sUAS. These flight plans force the sensor to remain at greater distances from their targets, resulting in varying spatial resolution of sloped surfaces. The work described here explains the development of a variety of 3D automated flight plans to provide vantage points not achievable by constant-altitude, nadir-looking imagery. Specifically, the issue of roof inspection is addressed in detail. This work generates an automated flight plan that positions the sUAS and orients its sensor such that the focal plane array is parallel to the roof plane based on a priori knowledge of the roof's geometry, greatly reducing single- or two-point perspective. This a priori knowledge can come from a variety sources including databases, a site survey, or data extracted from an existing point cloud. Still images or video from orthogonal flight plans can be used for visual inspection, or the generation of dense point clouds and meshes. These products are compared to those generated from nadir imagery. This novel flight planning approach permits the aircraft to fly the orthogonal flight plans from start to finish without intervention from the remote pilot. This work is scalable to similar sUAS-based tasks including aerial-based thermography of buildings and infrastructure.

Acknowledgements

First and foremost, I would like to thank the Canadian Armed Forces for the opportunity to study at RIT. I will be eternally grateful for the support of my chain of command in the pursuit of my Masters of Imaging Science degree. This would not have been possible without the support and inspiration of Major Kristin Strackerjan who supported my application.

To my advisor, faculty, and staff at RIT, the education I have received here far exceeds the lessons learned in the classroom or in the lab. The opportunity to put my hands on a variety of equipment has triggered my curiosity, and has helped me to identify and solve technical problems from a whole new angle. I am very appreciative of the assistance of Anna Lang who modified Sun's building extraction code and helped me troubleshoot this work.

I would like to express my appreciation to the various sponsors of the Chester F. Carlson Center for Imaging Science (CIS). The research being conducted within CIS has will continue to contribute to society on many levels. I would also like to thank the Genesee Country Village and Museum for allowing me to conduct my research on their premises. Not only have I completed my experimental goals, I was able to do so in a beautifully kept environment surrounded by historical buildings. I looked forward to every trip I made to your grounds.

To my new friends and counterparts that I have made here in the past two years, I would like to express my gratitude for your friendship, motivation, and assistance throughout the past two years. I hope that I was able to give back as much as I have received. From squash, to concerts, to trivia, you have helped me enjoy my temporary home here in Rochester.

To my parents and family, your love and encouragement has got me through a lot in this life. Growing up on a farm left me with a strong work ethic, unbridled determination, and dedication necessary to succeed in life. You have given me the support I needed to follow my path, even if you have not always agree with the direction. You gave me the room to make my mistakes, and the lessons I have learned have made me a better person.

To all of you, thank you.

Contents

Chapter 1

Introduction

RIT has recently invested heavily in small unmanned aerial systems (sUASs), acquiring multiple sUAS platforms and sensors. To date, most of the work done with these aircraft was based on downward looking (i.e. nadir) imagery, or manually adjusting a gimbal to point the sensor. While most flights are conducted using an autopilot, there are only a limited number of flight profiles that can be found on most ground control software applications that permit users to execute flight plans from a laptop or mobile device. The first part of this work is the development of a process for generating '.xml' flight plans used by a ground control application. This process creates a script, allowing a user to convert tabulated positions and sensor parameters into a flight plan. This permits flight plans to be rapidly generated for repeatable flights. To validate the process, I have generated three examples of flight plans that I felt would offer users greater flexibility. This work was presented at the Society of Photo-Optical Instrumentation Engineers's (SPIE's) Commercial and Defense Sensing 2017 [55] and is updated for this thesis. This work was expanded upon for other tasks such as roof inspection.

Roof inspection is becoming a popular task for these platforms, but imagery is often taken from nadir, above any trees close to the building for safety purposes. Alternatively, the aircraft and gimbal are controlled manually, often resulting in oblique imagery of the roof. Witnessing this problem, I have created a method of generating flight plans from building geometries for the purpose of acquiring imagery that is orthogonal to each roof facet of a building. Throughout this process, the aircraft maintains a constant distance from the roof facet. The bulk of this thesis deals with the comparison of dense point clouds generated from structure-from-motion using orthogonal and nadir imagery.

Thinking forward, the use of an autopilot to position the aircraft and sensor in complex patterns is ideal for infrastructure inspection, such as pipelines. With advance thermal sensors, these pipelines can be inspected using thermographic techniques [34]. By leverag-

ing automated flight plans to control a sUAS with an infrared sensor, this technology has potential to be applied to natural gas pipelines. I provide a brief overview of some thermographic techniques used in the hope that the work performed thus far will be applied to sUAS applications.

Chapter 2

Automated mission planning

2.1 Abstract

The rapid advancement and availability of small unmanned aircraft systems (sUAS) has led to many novel exploitation tasks that utilize this unique aerial imagery data. This work describes novel flight planning algorithms to better support tasks such as: Structure-from-motion missions to minimize occlusions, periodic overflight of calibration panels to better assess surface reflectance, and to study optical properties such as the bidirectional reflectance distribution function of a fixed target. Collection of these unique data requires novel flight planning methods to accomplish these tasks. These methods will provide scientists with additional tools to meet their future data collection needs.

2.2 Introduction

The small unmanned aerial system (sUAS) market has boomed in recent years. Estimates suggest that the global sUAS market will be worth between $6 and $8.4 billion by 2019 [9]. A variety of payloads are available for the sUAS market to meet an ever-increasing demand for data. Scientific payloads can include traditional cameras for photography or videography, multi- and hyper-spectral sensors, light detection and ranging (LiDAR), thermal cameras, and many other imaging modalities. sUASs are controlled manually by a remote pilot, autonomously by uploading a sequence of waypoints, or with an on-board computer that will react to changing conditions. Commercial ground control and mission planning software applications are available to the consumer. These systems are customized to meet the requirements of the aerial photography community and lack features required for more specific scientific tasks. Remote pilots wishing to fly complex flight profiles to meet technical requirements will likely find themselves struggling to enter

many waypoints manually, and a researcher may need to fly similar flight profiles at various different locations. This work documents algorithms that produce three specific flight profiles, and provides users with a way to create scripts, generating flight plans particular to their research.

2.3 Background

The sUAS technology boom has triggered a surge of mobile applications that can be used to control a variety of unmanned aircraft, such as: Litchi, Pix4D drone mapping software, Universal Ground Control Software (UgCS), and PrecisionHawk's Precision Flight. The majority of these applications enable a user to enter waypoints on a touch screen, permitting the aircraft to fly a preprogrammed mission. Many applications permit the user to specify a polygon on a touchscreen, input altitudes, and specify image overlap requirement. The output of these include a flight plan to acquire imagery of the entire area inside the polygon. While these tools are useful, they fall short in several areas required by imaging scientists:

1. Many imaging tasks require a very large number of waypoints to be entered manually. A small modification to the flight plan, like aircraft altitude, may require all of the points to be adjusted, which is excessively time consuming and error prone. The positional accuracy requirement of these systems require latitude and longitude coordinates to be specified using up to seven decimal places.

2. Existing photogrammetry flight profiles in commercial mission planning applications do not permit the user to specify a location for calibration targets. These targets may need to be overflown multiple times to accommodate for changing lighting conditions. These calibration panels are required to calculate surface reflectance values using the Empirical Line Method (ELM) [53]. Reflectance data are often more useful than radiance values or raw digital count in remote sensing applications. This is because reflectance is a target-specific property that is independent of atmospheric and illumination conditions under which the data were collected. Imaging scientists can apply various reflectance values to apply indices to perform a variety of tasks. One such index is the normalized differential vegetation index (NDVI), which is used to assess vegetation [50, p. 312]. Reflectance values also permit the application of anomaly detection and sub-pixel target detection algorithms [21, pp. 617-714].

3. Typical tools within applications are either limited to terrain following or constant altitude flights, and predominantly used with the camera fixed and pointing in the nadir direction. These profiles can be problematic due to self-occlusions present when

viewed from nadir. This results in a reduced number of points in a dense point cloud generated using structure-from-motion (SfM) techniques. If wanting to inspect an occluded structure, a sUAS would need to fly at various altitudes and camera angles. Fig 2.1 shows a pipebridge with many trusses that generate occlusions. Varying the view angle is also necessary when attempting to make an in-field bidirectional reflectance distribution function (BRDF) [50, p. 123] measurements. During such measurements, it is necessary to stare at a fixed position, regardless of platform location [15]. Inspection of wind turbine blades also requires precise flight plans to prevent damaging expensive and sensitive turbine components [23].

Figure 2.1: A pipebridge crossing the Cayuga-Seneca Canal illustrates how trusses can occlude features of a bridge.

sUASs are becoming more common for infrastructure inspection. Long stretches of gas pipelines can be inspected for structural anomalies from the inside of the pipe using a pipeline inspection gauge [56]. A sUAS operating near the pipeline also has the potential to detect anomalies using infrared sensors, for which customized flight plans would be required. These sUAS platforms have the potential to conduct rapid inspection, while minimizing risk to inspectors. The need is even more necessary when an inspection of a pipe's exterior requires accessing hard-to-reach locations, such as river crossings. Another advantage of inspecting the pipe's integrity from the exterior also means that the inspection will not impede the transportation of natural gas.

A variety of applications are available for iOS and Android mobile platforms that can operate several commercially available sUAS platforms. The ground control system used for planning by Rochester Institute of Technology's (RIT) UAS Research Laboratory is UgCS [4, 6]. The UAS Research Laboratory is equipped with the Da-Jiang Innovations'

(DJI) Spreading Wing S1000, Matrice 100, and Matrice 600, all of which are compatible with UgCS. The S1000 is capable of carrying payloads of up to 5 kg [18], but gimbal control is yet to be automated. The Matrice 100 is equipped with the Zenmuse X3 camera that provides full gimbal control manually or through UgCS. The Matrice 600 platform will serve as the testbed for new payloads under development. RIT also has access to inexpensive platforms, such the Parrot AR Drone 2.0, for algorithm testing to ensure that flight plans can be successfully uploaded and flown.

UgCS flight plan segments are similar to those found in similar applications used for sUAS control. Available UgCS segments include:

1. Takeoff and landing. Depending on the platform, UgCS permits automatic takeoff and landing.

2. User-specified waypoints. The ability to specify the latitude, longitude, and altitude of multiple points that the user requires. Some mobile applications will allow the user to enter waypoints rapidly using a touch screen, however, entering evenly spaced points is a cumbersome process due to the required precision. The need to enter a large number of points, required for longer missions, amplifies this problem.

3. Photogrammetry tool. This feature determines waypoints necessary to take photos at a specified ground sampling distance (GSD) within a user-specified polygon. GSD is calculated using Eq 2.1 and represents the size of a single pixel projected on the ground, as shown in Fig 2.2 [50, p.606]. Based on the users GSD requirements, the application will adjust the aircraft's height above ground level. A digital elevation model (DEM) is incorporated into UgCS to determine the aircraft's height above ground based on the aircraft's current GPS location. Waypoint spacing is determined based on user-specified overlap. Fig 2.3 shows fields required to generate this flight profile, along with a visualization of this segment.

$$GSD = z * w/f \tag{2.1}$$

   - GSD is the Ground sampling distance [m]
   - z is the distance above ground [m]
   - w is the width of a pixel [mm]
   - f is the focal length [mm]

4. Area scan. This feature is similar to photogrammetry tools, except the aircraft flies at a specified altitude above mean sea level (AMSL), permitting the GSD to vary.

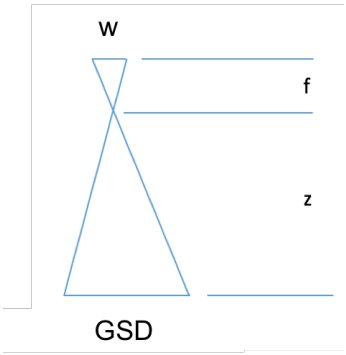

Figure 2.2: Relationship of GSD, focal length, height, and pixel size.

5. Circles. The user specifies the center, radius, number of points and altitude, from which the application determines the waypoint locations to approximate a circle.

6. Perimeter. The user specifies a polygon and altitude, and the application will control the aircraft, such that it follows the edge of the polygon.

Camera actions permit the autopilot to control when the camera acquires an image, the zoom level, orientation, time delay, as well as a variety of other features. These actions are associated with one or more of segments in the flight plan. The UgCS graphical user interface (GUI) uses the icons displayed in Fig 2.4 to permit the user to append camera actions to segments.

Even during a brief flight, the lighting conditions may vary rapidly due to cloud movement. Low clouds, such as cumulus, can produce dramatic effects, casting a shadow on the scene with a clearly visible edge. High clouds are subject to high velocity winds and tend to diffuse light. An example of a high cloud affecting illumination would be an airliner's contrail being blown under the sun. In a matter of a seconds, the scene will shift from being primarily illuminated by direct solar radiance to a diffuse downwelling radiance [50, pp. 57-62]. The photogrammetry tool is convenient during consistent lighting conditions, but for agricultural purposes, sUAS platforms are required to fly over calibration panels so that surface reflectance values can be calculated using ELM [53, 13]. Variable lighting conditions introduce errors when calculating reflectance values. If a flight occurs under variable lighting conditions, then it is often necessary to revisit the calibration panels to compensate for these conditions several times during a given flight. The existing method of visiting these calibration panels involves the remote pilot assuming manual control of the aircraft, flying to the calibration panels and then resuming the flight [13]. Ensuring

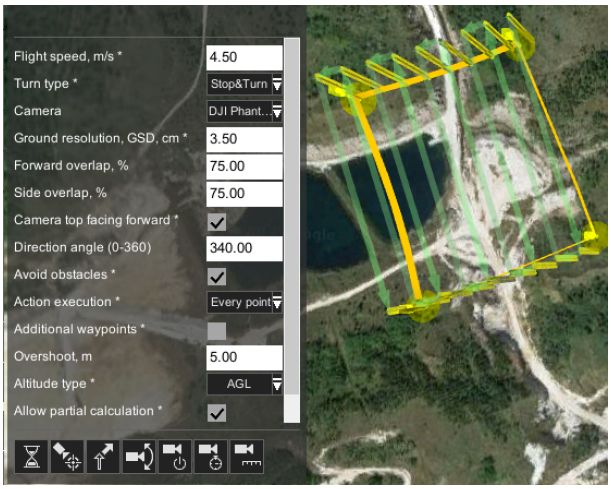

Figure 2.3: Image captured from UgCS shows an example of the photogrametry tool with necessary user inputs. Icons at the bottom of the screen show various camera actions available to the user.

Figure 2.4: UgCS offers seven camera actions from left to right: wait, point of interest tool, yaw, set camera angles, still/video, time interval between exposure, and distance between exposures.

the target is centered in the frame is not always an easy task for the remote pilot, as the pilot may not have a live video feed from the aircraft.

Another useful pattern absent from these software applications is a hemispherical pattern. A hemispherical pattern is useful for observing the effects of the BRDF [50, p. 123], as well as creating dense point clouds of intricate structures using SfM software. For this pattern to be useful, the lookdown angle of the sensor must be adjusted while the aircraft is continuously oriented towards the center of the image. These flight profiles have been used during previous experiments for the purposes of creating a UAS-based goniometer where 'mAngle' software permits flights to be planned and executed in 30 minutes, of which, 20 minutes were allocated for preparation [15]. Increased automation of the flight planning process will help reduce this preparation time further. To prevent blurring, the

aircraft should be kept stationary during image acquisition, but this is not achievable due to vibration and aircraft motion. Experience with RIT's DJI Matrice 100 has shown that attempting to upload many waypoints to the aircraft is time consuming and may generate a 'timed out' error. There is also a limit to how many waypoints the aircraft's autopilot can store. Exceeding this value requires that the flight plan be divided into two or more segments.

Bridge inspection is another frequent task required of sUASs for which commercial ground control applications are not optimized. Inspecting a bridge involves a repetitive back and forth motion, and altitude adjustments during imagery acquisition. Manually flying close to a bridge adds risk, using automated flight control can significantly reduce the risk of collision with the structure. Images captured during a bridge inspection can be used to create a dense point cloud using SfM [17], or taking several still images from multiple angles. As with the hemispherical pattern, the camera's gimbal must be adjusted to ensure that the bridge is always in frame.

2.4 Methods

UgCS was the primary ground control application used for this work. UgCS saves flight plan files in '.xml' format. By analyzing the '.xml' file line by line, the necessary fields can be replicated using '.xml' capabilities residing within MATLAB [3] to create the '.xml' document, node structure, and administrative parameters.

UgCS's '.xml' flight plan format consists of header and footer nodes. These sections describe data critical to the flight planning process. Home location and altitude, altitude when returning to home, and preferred altitude type (height AGL vs AMSL), are examples of data stored here. Once these are created, segment nodes are appended to the structure. Each segment may also have camera action nodes appended to them.

MATLAB [3] functions were created to generate a '.xml' structure to include each UgCS segment and action. This was accomplished by iteratively creating files using UgCS to better understand the file format. These functions create a document node and append header, footer, and segment nodes to the main document node. When a camera action node is required, it is created and appended to a segment node before the segment node is attached to the main structure. Fig 2.5 provides a visualization of this structure.

To verify the functionality of the MATLAB functions, I created three flight profiles that could be useful to imaging scientists: agricultural, hemispherical, and bridge inspection.

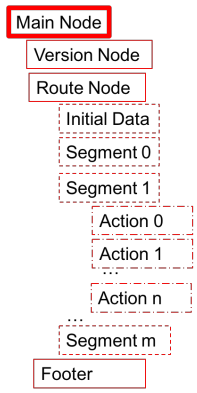

Figure 2.5: Functions created with MATLAB build a flight plan by first creating a header, adding segments and adding actions to segments as required.

2.4.1 Agricultural flights

Typical agricultural flights, as shown in Fig 2.3, require an aircraft to fly a raster-like pattern across a user-specified area while taking images at nadir (i.e. directly below the aircraft). Since imaging scientists are often concerned with surface reflectance, it may be necessary to fly over and capture images of calibration panels a number of times over the duration of the collection. Once the images are obtained, surface reflectance values can be calculated by applying the empirical line method [53]. The following algorithm accomplishes the task of collecting images while the aircraft is shooting at user-specified intervals set on the camera:

- Find georeferenced imagery. National Agricultural Imagery Program (NAIP) imagery is available from '`http://earthexplorer.usgs.gov`' and offers one meter resolution over a large area. This site permits users to specify a region of interest by clicking on an image as opposed to tabulating latitude and longitude coordinates. The downloaded images are provided in a 'geo TIFF' format, which includes georeferenced data.

- Re-project the image. Project the image onto a Mercator projection [54]. Harris Geospatial Solutions' ENVI 5.3 software package [2] provides a variety of tools for image analysis, including georeferenced images. One such tool permits georeferenced images to be reprojected onto another georeferenced system. This projection process re-projects the 'geo TIFF' image, such that all lines of longitude are vertical and all lines of latitude are horizontal. This simplifies calculations when selecting points on an image.

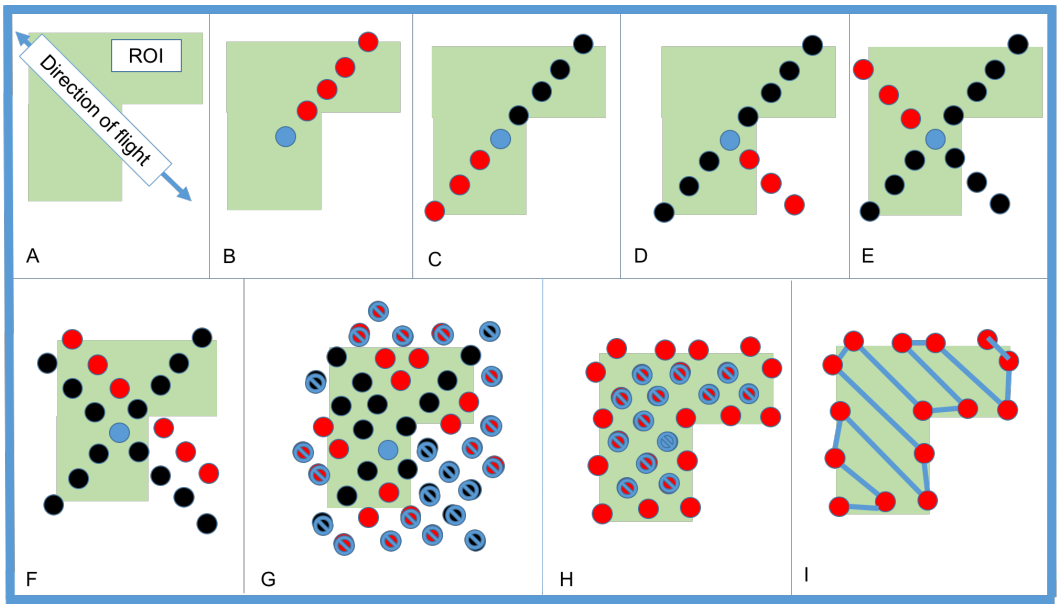

Figure 2.6: A - Determine direction of flight. B - Find center point, and build points orthogonal to direction of flight. C - Repeat step B in opposite direction. D - Starting at each point along orthogonal line, add imaging points in the direction of flight. E - Repeat Step D in opposite direction. F - Continue to populate the entire ROI with imaging points, exceeding the limits of the ROI. G - Remove all points outside of the ROI. H - Keep only the end points along direction of flight. I - Sort the remaining points in a logical order.

- Reduce size. Full NAIP images are not convenient to use since the coverage area is much greater than what a commercial sUAS can cover. A subset of the image was selected. The latitude and longitude of all four corners were noted as this was necessary to convert from row/column to latitude/longitude. The image was then saved as a standard '.tiff' format.

- Import image into MATLAB. Use the 'imread' command to read in the image and store latitude and longitude of the top, bottom, left and right of the image as variables.

- Select the region of interest (ROI). This can be done by either clicking on the image or specifying the latitude and longitude of each polygon vertex of each pixel. An example of a non rectangular ROI is shown in Fig 2.6-A.

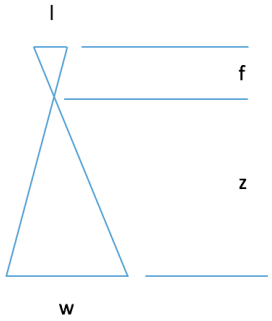

Figure 2.7: Similar triangles are used to calculate the projected swath width of the image, assuming that the ground is level. Once the swath width is known, the leg spacing can be determined based on the required overlap. f is the focal length [mm], l is a sensor width or height [mm], z is the altitude above ground [m], and w is the swath width of the image [m].

- Direction. By default, the algorithm applies principal component analysis (PCA) to determine the prominent direction of the ROI [38]. If the ROI is an elongated rectangle then by default, the aircraft should fly flight paths parallel to the long edges of the rectangle. If the user requires specific track angles, then this step can be commented out and specified manually. PCA is a tool used in pattern recognition and image analysis. For pattern recognition, when provided with a list of Cartesian coordinates, such as the latitude and longitude of a region of interest, the directionality of the points can be found by creating a two-dimensional array consisting of a column of 'x' values and a column of corresponding 'y' values. Since this two-dimensional array consists of 2 columns, there are two eigenvectors. The first eigenvector, or Principal Component (PC), describes the principal direction of the points and the second PC is orthogonal to the first PC. The first PC corresponds to the vector with the highest eigenvalue. Each PC can be represented on a 2-dimensional plane as a vector. A heading can be measured counterclockwise from the vertical axis and expressed in degrees from true north, minimizing the number of legs required to fly.

$$\Delta y = Xw = \frac{Xlz}{f} \tag{2.2}$$

- Δy is the separation [m]
- X is the user-specified overlap [-]

- w is the swath width [m]
- l is the edge length of sensor [mm]
- z is the height above ground [m]
- f is the focal length [mm]

- Plot initial points. As shown in Fig 2.6-B, find a point near the center of the ROI and create points orthogonal to the direction of flight where images are to be taken. Then return to the center and repeat in the opposite direction as shown in Fig 2.6-C. Coordinates for the image locations are found using Eq 2.3 and Eq 2.4 [60] [1]. Ensure these points are stored in a logical sequence. From each of these points, add additional imaging locations in the direction of flight by applying Eq 2.3 and Eq 2.4 [60] again. A grid of imaging points is built up to include points well outside of the ROI before removing points outside of the ROI as shown in Fig 2.6. Extending points outside the ROI is important for more intricately shaped ROIs. All of the imaging points then need to be arranged in sequence.

$$lat_{new} = sin^{-1}(sin(lat_{current}) * cos(\frac{d}{R})) + cos(lat_{current})sin(\frac{d}{R})cos(\theta)) \tag{2.3}$$

$$long_{new} = long_{current} + tan^{-1}[cos(\frac{d}{R}) - sin(lat_{new})), sin(\theta)sin(\frac{d}{R})cos(lat_{current})] \tag{2.4}$$

  - $lat_{current}$ is the latitude of the current point [radians]
  - lat_{new} is the latitude of new point [radians]
  - d is the distance to travel [km]
  - R is the radius of the Earth [km]
  - θ is the bearing [radians]
  - $long_{current}$ is the longitude of original point [radians]
  - $long_{new}$ is the longitude of new point [radians]

- Populate ROI with imaging points. As shown in Fig 2.6-D, starting from the orthogonal points just created, create points in the direction of flight. This is repeated in the reverse direction as shown in Fig 2.6-E, and repeated again as shown in Fig 2.6-F.

[1] 'tan^{-1}' represents the four quadrant inverse tangent in these equations.

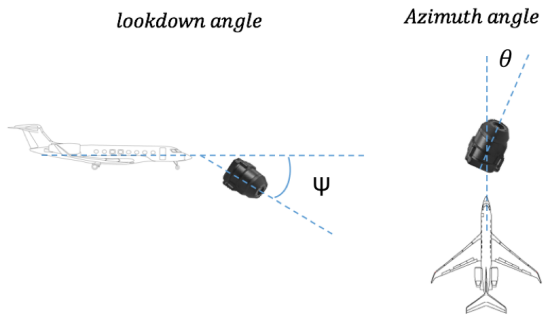

Figure 2.8: Lookdown and azimuth angles [images from dji.com and debonair.co.uk.com have been annotated.].

- Remove points outside of ROI. Once the ROI is filled, remove excess points as shown in 2.6-G. MATLAB's 'inpolygon' function removes all of the points outside of the ROI. At this juncture, all points can be expressed in either Cartesian or latitude/longitude coordinates. Functions for changing between coordinate systems were derived based on linear interpolation using Eq 2.5 and Eq 2.6.

$$x = \left(\frac{long - A}{B - A}\right) w \tag{2.5}$$

$$y = \left(\frac{lat - D}{C - D}\right) h \tag{2.6}$$

  - x is the x-coordinate on the image measured from left to right [pixels]
  - long is the longitude of a point [degrees]
  - A is the line of longitude corresponding to the left side of the image [degrees]
  - B is the line of longitude corresponding to the right side of the image [degrees]
  - w is the width of the image [pixels]
  - y is the y-coordinate on the image measured from bottom to top [pixels]
  - lat is the latitude of a point [degrees]
  - C is the line of latitude across the top of the image [degrees]
  - D is line of latitude across the bottom of the image [degrees]

  - h is the height of the image [pixels]

- Reduce points. There are a finite number of waypoints that can be uploaded to a sUAS. For the Matrice 100, the limit is 99 points [4]. To reduce the number of waypoints, only the endpoints of each leg are needed, so all points between these points can be removed.

- Calibration panels. The user will specify the number of visits to the calibration panels required. If only two trips to the calibration panels are required, the aircraft will takeoff and immediately fly to the calibration panels before starting the route, and then revisit the panels after imaging of the ROI is complete. If additional visits to the panels are required, then the algorithm calculates the total time required over the entire ROI, creating breakpoints to permit the aircraft to image the calibration panels. If only one extra trip is required then the accumulated flight time to fly over the ROI can be split into two roughly equal portions. Once the calculated interval is exceeded, the aircraft will revisit the panels at the end of that leg and return to the start of the next leg.

2.4.2 Hemisphere

A hemispherical pattern is particularly useful for 3D reconstruction using SfM software, such as Pix4Ddesktop [5], and for observing BRDF effects [50, p. 123]. In either case, the target must be centered in each frame in order to extract features of the desired target. The following algorithm generates a hemispherical flight plan:

- User specified data. The latitude, longitude, and the elevation of the target must be known. The elevation can be determined using the aircraft's GPS by setting it on level terrain near the target, since digital elevations are based on averages over a wider area. The speed must be specified, but it is not critical. Faster speeds make stopping and positioning itself accurately at a waypoint more difficult. The radius of the hemisphere is also user-specified.

- Determine a safe waypoint. Obstacles may be present between the takeoff point and the target's location. It is a good practice to climb to a specified altitude and position before starting the hemispherical pattern.

- Start at the top. All altitudes of waypoints in the sphere must be calculated in height above mean sea level, since the surrounding terrain may not be level. The first point is directly above the target at an altitude that is calculated by adding the radius of the hemisphere to the elevation of the target. Set the camera lookdown angle to 90 degrees.

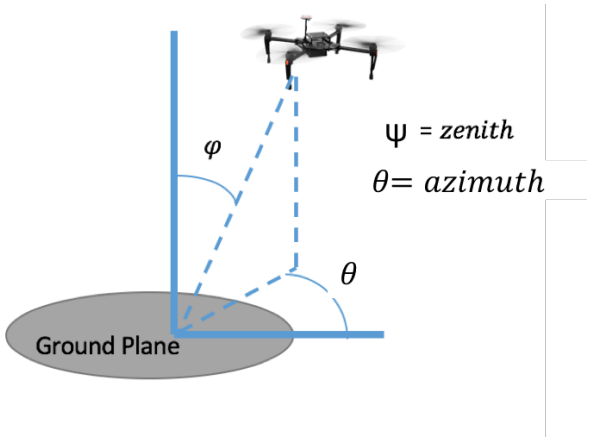

Figure 2.9: Image shows the zenith angle measured from zero and azimuth angle measured from an arbitrary direction [Image from dji.com has been annotated.]

- Fly concentric circles. Fly north to the next waypoint. Adjust the altitude to the required zenith angle and use the point of interest (POI) action to yaw the aircraft towards the target, and then take a single frame. Fly concentric circles until the target has been imaged from the specified zenith and azimuth angles, as shown in Fig 2.9. Adjust the camera lookdown angle at the start of each circle. The lookdown angle is the angle of the sensor below the horizon, as shown in Fig 2.8.

- Split flight plan. It may be necessary to generate multiple files, depending on the number of waypoints required. The algorithm permits the user to specify the number of points that can be uploaded to their specific aircraft. If the aircraft's capability is exceeded, then the flight plan must be split into two or more parts.

- Manual control. At the end of the flight, the aircraft will enter a hover. This permits the remote pilot to assume manual control and safely land the aircraft. This reduces the potential for the aircraft to hit an obstruction between the target and the landing site. The user can add additional waypoints as required for safety purposes.

When completed, this algorithm produces waypoints and camera angles as illustrated in Fig 2.10.

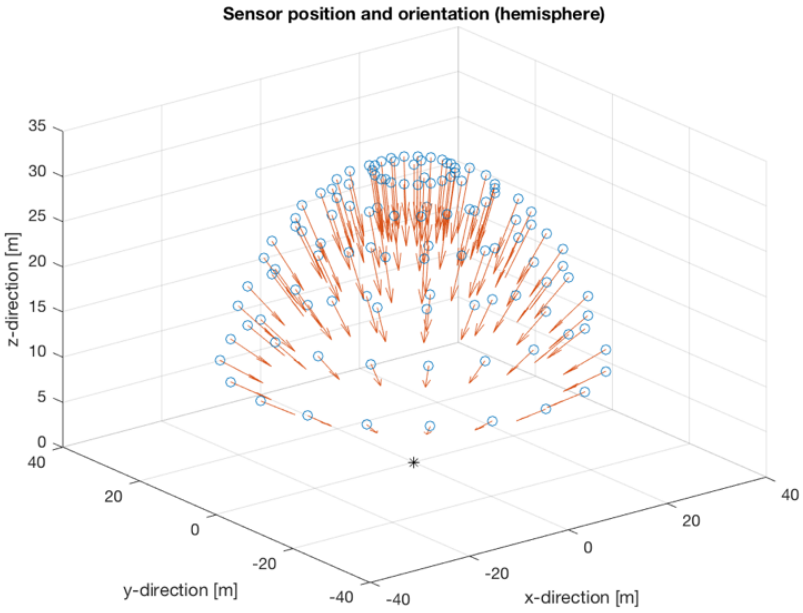

Figure 2.10: UAS position and gimbal angles always keep the sensor pointed at the base of the hemisphere

2.4.3 Bridge Inspection

Many bridges have a truss-type construction where the trusses occlude much of the bridge as shown in Fig 2.1. To resolve these occlusions, images must be taken from multiple vantage points. It is important for the camera's optical axis to be oriented towards the bridge. Any change to camera azimuth, camera lookdown angle, or aircraft yaw will cause the bridge's resolution to be inconsistent across the frame. If attempting to fly below a bridge, the autopilot of the Matrice 100 will not accept camera angles that look above the horizontal plane, so any negative lookdown angles must be set to zero. If the aircraft does fly below the level of the bridge, the camera will need to rely on it's field of view to resolve occlusions, as opposed to adjusting the camera lookdown angle. Attempting to extend the profile beyond a semi-circle risks interference of the GPS signal by the bridge.

A flight profile, resembling a semi-circular tunnel surrounding the bridge (see Fig 2.14) that adjusts the camera lookdown angle, meets these requirements and is created using the following algorithm:

- User-specifications. The user must specify the endpoints, ground speed, angles, elevation of the bridge, and the radius of the flight path. The overlap can also be specified if taking still imagery.
- Calculate end points. While many camera positions may be calculated along several linear legs, only the end points need to be kept. An example of the end points and the flight path are displayed in Fig 2.11.

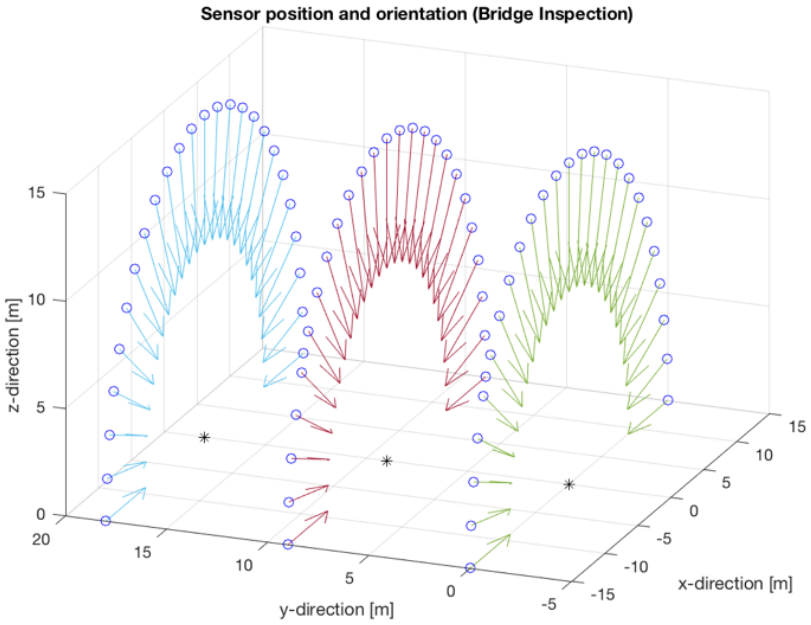

Figure 2.11: Aircraft position and gimbal angles. The sensor always points along a linear path denoted by black asterisks.

- Calculate lookdown angle. The lookdown angle is simply 90 degrees less than the zenith angle. This needs to be calculated for each waypoint.
- Calculate yaw angle. The aircraft must orient itself so that the frame is parallel to the direction of the span. The yaw angle will be either 0, 90, 180, or 270 degrees, as measured from the aircraft's track. The angle can be selected by observing a pattern of the necessary yaw angles. When passing along the apex, the yaw angle should

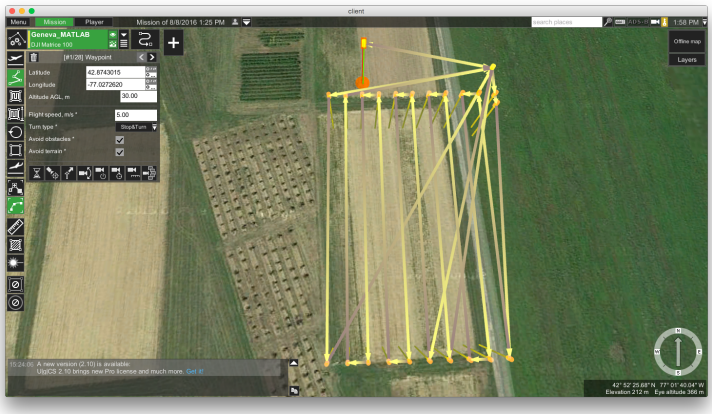

Figure 2.12: Screen capture of UgCS showing a calibration panel flight profile. Calibration panels would be situated in the top right corner where several arrows converge.

be either 90 or 270 degrees, such that bridge is parallel to the longer edge of the camera's frame.

2.5 Results

All flight profiles discussed have successfully been loaded into UgCS, as can be seen in Figs 2.12, 2.13, and 2.14. Both the hemispherical, and bridge inspection flight profiles, have been successfully executed.

All functions used to create these flight plans along with sample scripts are available on Github (`https://github.com/paulSponagle/UGCS_MATLAB_scripts`) and available for public use. The code is caveatted as experimental and the remote pilot is ultimately responsible for verifying a safe flight path and the safe operation of their aircraft.

Figs 2.15 and 2.16 are examples of two point clouds generated using the hemispherical and bridge inspection algorithms. Fig 2.16 displays intricate details that can be seen along the side of the pipebridge structure, whereas the point cloud in Fig 2.17 shows less detail along vertical surfaces. Fig 2.17 also shows gray points that appear to be attached to the structure. These points represent background that often get attached to thin members, which can be removed by editing the point cloud. Fig 2.16 was processed significantly to remove much of this type of noise. There are several reasons that the reconstruction created from images taken at nadir was worse than the one created using the algorithm.

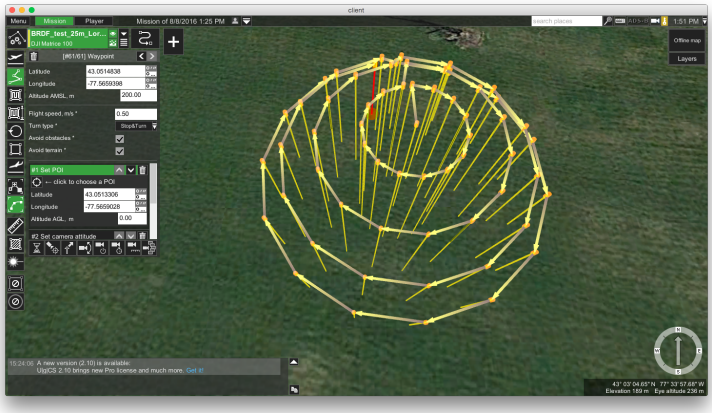

Figure 2.13: Screen capture of UgCS showing a hemispherical flight profile.

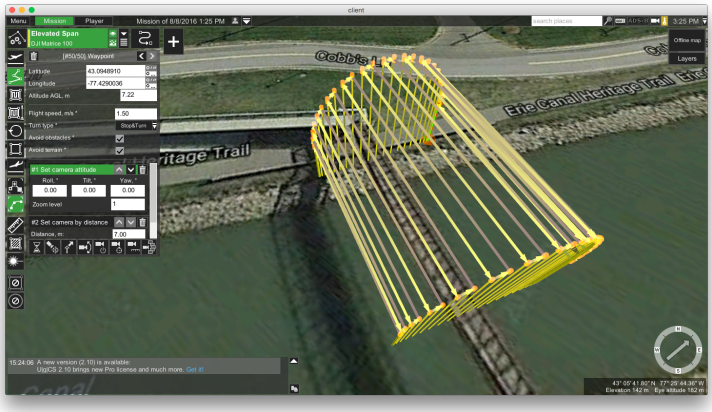

Figure 2.14: Screen capture of UgCS showing a bridge inspection flight profile.

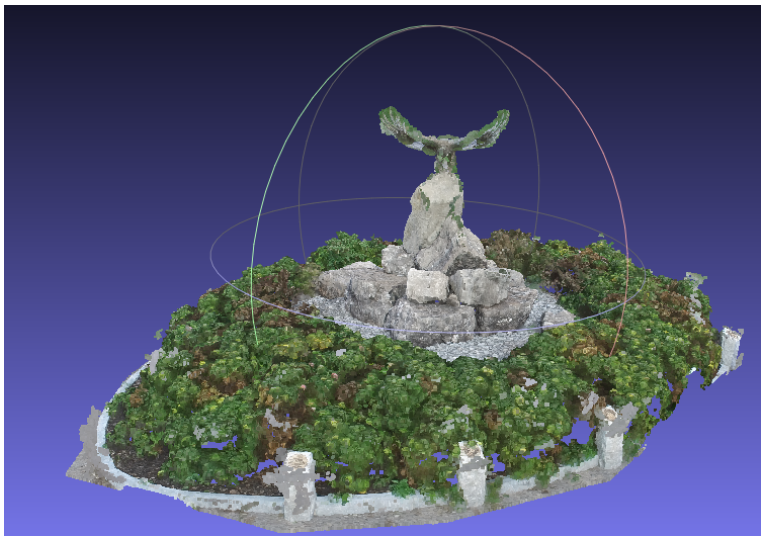

Figure 2.15: Point cloud reconstruction of a statue with many self-occlusions.

The primary deficiency was caused by the pipebridge's self-occlusions that could not be resolved when imaging at nadir. The nadir reconstruction was also taken from a further distance away from the bridge, due to the presence of obstructions, resulting in a lower GSD and fewer points within the point cloud.

From experience, it is also interesting to note that flying closer to a structure that has a repetitive pattern is not always better. Points may be attached to the wrong instance within the pattern. This minimum altitude restriction is a common issue when flying constant altitude flights. The nadir point cloud was also constructed using fewer images at an overlap setting of 80%, in order to mimic typical collections from nadir. While additional photos would have improved results, the self-occlusions would still be present. Imaging from an angle orthogonal to a surface allows the greatest number of tie points to be generated, resulting in a better point cloud. A constant altitude flight looking at nadir will typically have fewer tie points on vertical surfaces compared to level surfaces. Each point cloud was also taken with different 4K cameras, but this is comparatively minor issue in comparison to the other discrepancies.

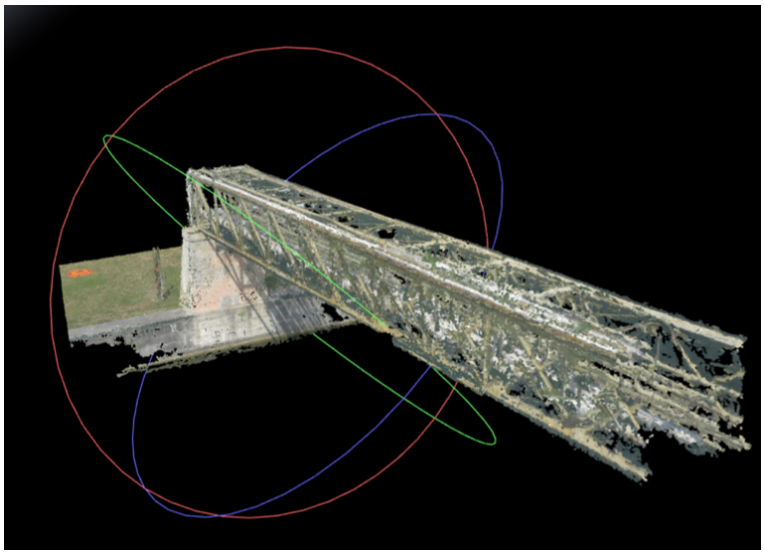

Figure 2.16: Point cloud reconstruction of a pipebridge using the bridge profile algorithm.

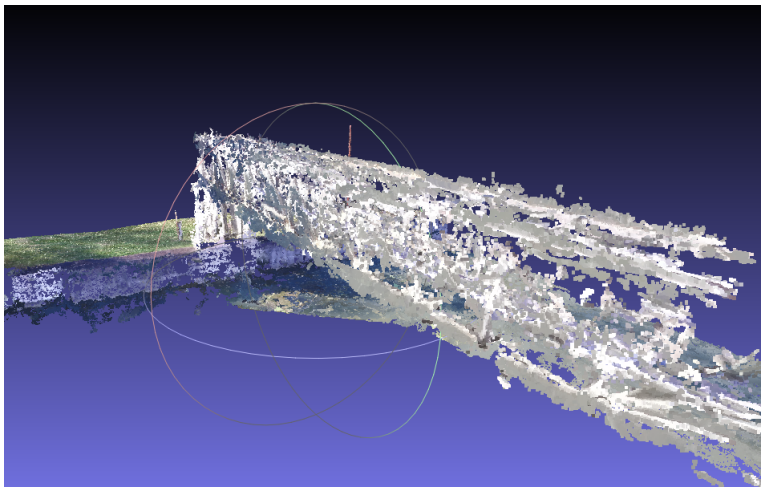

Figure 2.17: Point cloud reconstruction of a pipebridge using images collected at nadir.

2.6 Conclusion

The completed work provides fellow researchers with three specific flight plans that may be of interest to their research. These flight plans address several deficiencies amongst popular ground control applications. More importantly, if a user can describe a flight pattern and camera angles in a Cartesian coordinate space, they can generate their own flight profiles and read them into UgCS to fly pre-programmed missions.

2.7 Future Work

The efforts presented document the work required to create custom flight plans for use on readily available software. Future work should include:

- Testing of the agricultural flight plan in an operational environment and perform any necessary debugging.
- Creating a graphical user interface that permits customized flight plans to be created by filling out the necessary fields.
- Create periodic updates of the code, ensuring it is compatible with new versions of UgCS.
- Leverage DJI's 'Onboard Software Development Kit' to update flight plans in real time. This technology would lend itself well to variable lighting conditions. The changing conditions could be monitored to determine when images of calibration panels are required. It could also be useful for building point clouds of structures by anticipating cloud movement and determining when the lighting will become more diffuse.

Chapter 3

Roof inspection

3.1 Abstract

The rapid proliferation of consumer small unmanned aerial systems (sUAS) has expanded ownership to amateur and professional pilots alike. These platforms, in combination with numerous software applications that can generate 3D point clouds and meshes from aerial imagery, have made 3D modeling available to anyone who can afford an entry-level sUAS. Traditional, constant-altitude flight patterns are frequently used to generate 3D models, but these flight plans force the sensor to remain at greater distances from their targets. Detailed rooftop inspections require better resolution to make sound assessments. This work leverages existing 3D modeling techniques to generate dense point clouds from which a separate automated flight plan is created. This permits imagery to be gathered with improved spatial resolution. This process also aligns the optical axis perpendicular to a roof face, such the image of the roof face maintains a constant resolution across the frame. The improved spatial resolution permits better photos and videos to be captured. These improved photos provide the basis for production of optimized dense point clouds and meshes.

3.2 Introduction

A common use of rapidly proliferating small unmanned aerial systems (sUASs) is to create three dimensional models by creating point clouds and three-dimensional meshes generated from multiple still images using structure-from-motion (SfM) techniques [17]. These images are typically collected at constant altitude, using an autopilot, while looking directly beneath the aircraft (i.e. nadir) [17]. Alternatively, the aircraft position and sensor angles can be oriented manually by the pilot. Another option is that a pilot can manually

calculate the proper positions and angles, and then enter these data into a ground control application and execute an automated flight plan. This would be time consuming for complex flight plans. Flying manually also poses the problem of repeatability as it might be necessary to revisit the site and image from the same positions on multiple occasions. An automated flight plan, that includes gimbal inputs, would address these issues.

The inspection of rooftops is becoming a popular task for sUAS platforms [7] [41]. Images from this type of collection are often highly distorted as the sensor's focal plane array is rarely oriented parallel with the portion of the roof being imaged. This work intends to generate automatic flight plans from initial point cloud data for the purpose of imaging major planar surfaces of a roof, while keeping the focal plane array (FPA) parallel to each roof plane of interest. This point cloud can be derived from SfM, light detection and ranging (LIDAR) [22], an existing database, or determined manually from a site survey. From these flight plans, additional images will be collected and a dense point cloud and mesh will be created. These products will be compared to data collected by overhead flights.

3.3 Background

One of the many tasks typically accomplished using sUASs is to gather imagery. Individual images on their own have value for inspection purposes, but with sufficient overlap, they can also be used to generate 3D models using a technique called structure-from-motion (SfM). A patent has been published that describes a sUAS that will accomplish this task autonomously [51]. This work aims to provide the necessary steps that will calculate the camera locations and angles that a similar sUAS might use.

3.3.1 Point cloud generation

Generating flight plans for rooftop inspections involves a number of steps, each of which can be carried out using a number of approaches. The basic steps used within this work include: point cloud generation, extracting buildings from point clouds, finding buildings within a point cloud, finding roof planes and their directionality, and generating camera positions from which to collect imagery of the roof surface. Once sensor position and orientations are determined, these data need to be placed into a format that is readable by a ground control software.

One method of generating point cloud data is to use a LIDAR to build a point cloud of a scene consisting of many points. The device is tied to an inertial measurement unit (IMU) and a GPS to accurately geolocate points. The output of this technique generates a point cloud with georeferenced coordinates [33].

Alternatively, a point cloud can also be generated using SfM. SfM involves generating a 3D point cloud using photogrammetry taken from several vantage points [32]. Ideally, the images are shot without adjusting the focus or aperture, and only the exposure time and ISO are varied to accommodate illumination. This allows the software to do an internal calibration of the sensor to account for the sensor's geometry and lens distortion. As a general rule of thumb, these images should have high overlap, such that each point of interest within the object being modeled is imaged several times [17]. Some objects may be occluded by other objects in the frame. Increasing the overlap ensures occluded points are imaged in more frames, providing more reference points to better triangulate each feature.

Imaging a point on a plane, while translating the sensor parallel to the plane at 80% frame overlap will result in each point being imaged at least 16 times as shown in Fig 3.1. Assuming that the plane is perfectly flat, there is no lens distortion, and the optical plane is aligned perpendicular to the roof plane, the point at the center of the figure may also appear along the edge of an additional 20 images [52]. To aid visualization, bars corresponding to the frames' horizontal and vertical positions are indicated by blue bars. The dashed line represents the frame boundaries of the additional 20 images where the point would rest on the edge or corner of a frame.

Fig 3.2 shows how the spacing between capture points increases with an increase in altitude. Increasing the altitude expands the swath width of the sensor, so for a constant overlap, the distance of sensor travel between imaging positions increases. This means that if the same region of interest is imaged at two different altitudes and constant overlap, there will be fewer images taken at the higher altitude. If the swath width of a frame is much larger than the region of interest, then the number of times each point is imaged is reduced. Likewise, points on the edge of a region of interest are imaged fewer times.

Commercial SfM applications, such as Agrisoft's 'Photoscan' and Pix4D's 'Mapper Pro Desktop', enable users to generate 3D models by inputting images taken from a camera [26]. The software then uses any of a number of proprietary or open source algorithms to extract features from each image and relates these features to one another to extract points corresponding to other photos. From these correspondences, the coordinates of the corresponding features can be triangulated [17]. SfM applications are able to make use of metadata attached to the file such as latitude, longitude, height above mean sea level (MSL), f/stop, and exposure time to better initialize data for improved results [17]. Examples of features extracted from an image using Pix4D are shown in Fig 3.3.

Common feature extraction algorithms commonly used to generate three dimensional point clouds include:

- **Scale-invariant feature transform (SIFT)** SIFT is a proprietary algorithm that generates corresponding features between images using a staged filter approach.

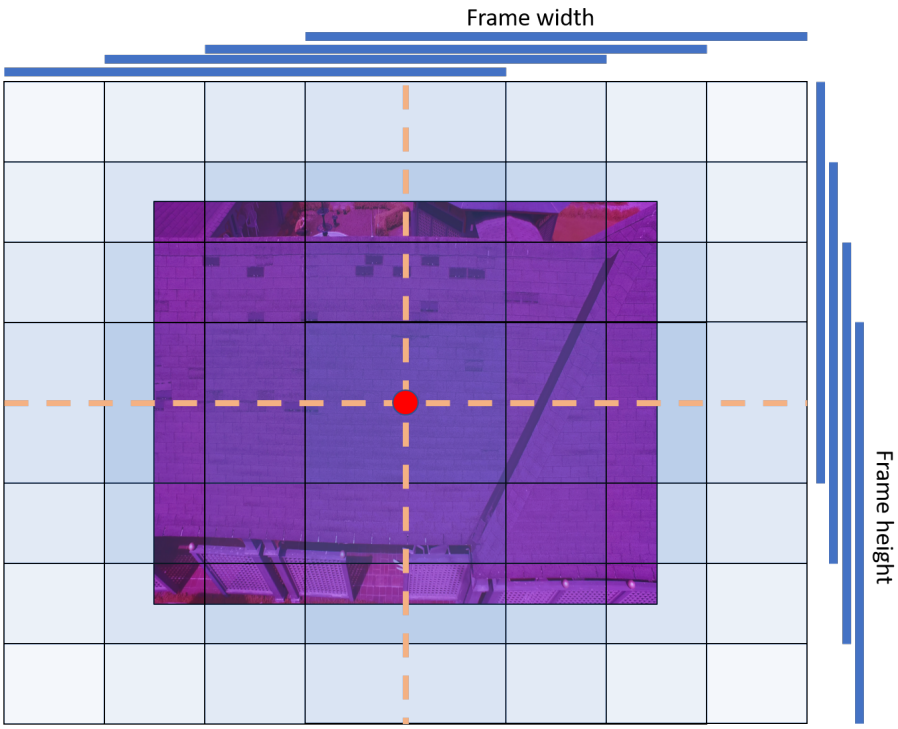

Figure 3.1: Frame distribution with 80% overlap.

SIFT is invariant to rotation and scaling, while being partially invariant to illumination, camera viewpoints, and occlusions [35]. Fig 3.16 provides an example of occlusions along a wall created by a roofline. SIFT features are found using a difference of Gaussian filter to approximate a Laplacian of Gaussian, while rotation invariance is achieved by determining the major orientation of local neighborhoods and aligning them with the major orientation. SIFT has been used as the basis for several other models such as SIFT-PCA, where principal component analysis is used to reduce the dimensionality of the rotational space, and the speeded up robust features (SURF) [31].

- **Speeded up robust features (SURF)** The SURF method [11] can outperform SIFT in many situations, but not all [31]. Its primary advantage is that it gains computational efficiency by deemphasizing rotational invariance [11]. SURF is a two-step process: first producing a fixed reproducible orientation; second, constructing a

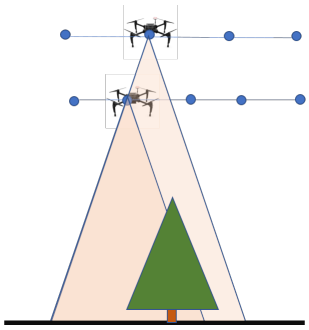

Figure 3.2: Effects of altitude on swath width and number of images for a constant overlap. [Image from dji.com has been annotated.]

square region aligned to the selected orientation from which to extract features [11].

- **Harris Corner Detector** The Harris corner detector generates features based on gradient images. The significant drawbacks of this detector is that it is not scale-invariant, and produces fewer features than SIFT or SURF [11, 27].

Camera calibration, location, and orientation

Regardless of which method is used to generate features from the images, an iterative matching algorithm matches features from one photo with every other photo based on common features. From these matches, the application can determine the position and angle of the sensor through triangulation [10]. Features extracted from images can range in number from tens of points to thousands. The process of finding correspondences can generate a high percentage of mismatches (i.e. outliers) that can be reduced by using methods such as random sampling and consensus (RANSAC) [28, 57].

This process of finding correspondences can be aided by manual intervention by marking a common feature on several photographs and then repeating this for several features. Each marked feature is known as a two-dimensional ground control point (GCP). GCPs can be used to correct camera positions and orientation that were misaligned during initial calibration.

A sparse point cloud is generated by triangulating the position of features that appear in multiple photographs to provide Cartesian coordinates. The resulting points are known as automatic tie points. If the images have geospatial metadata attached, then each point in the sparse point cloud will be georeferenced. Once these points are created, three-dimensional GCPs can be added to improve accuracy if coordinates of specific objects in

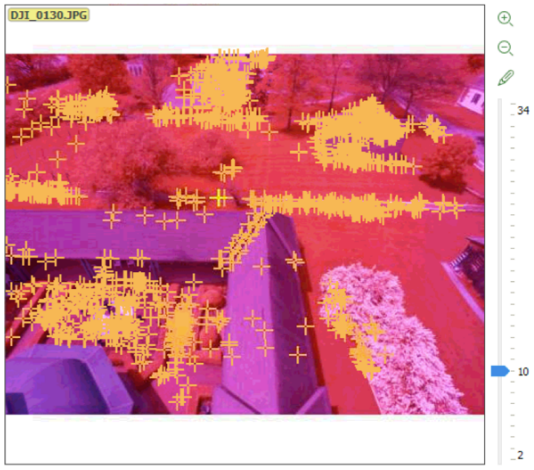

Figure 3.3: Examples of features extracted from an image within Pix4Ddesktop.

the scene are known. A dense point cloud is generated by applying a densification process that finds new points between tie points found in the sparse point cloud.

The output of these SfM applications are dense point clouds that can be referenced in absolute or georeferenced coordinates [5]. The two referencing systems of primary interest in this work are the Universal Transverse Mercator (UTM) system and the World Geodetic System 1984 (WGS84) system. An extension of this work also uses the traditional Mercator projection. Each of these systems can have their own advantages, but have the common goal of positioning a specific point of the Earth's surface. Conversion from UTM to WGS84 was done using a script slightly modified from mathworks.com [43]. UTM and traditional Mercator projections are ways of projecting the Earth's round surface onto a planar surface.

- WGS84 uses two coordinates to express horizontal position and a third coordinate represents the elevation above sea level. This system accounts for the Earth not being a perfect sphere. This system is used by the Global Positioning System (GPS) to permit GPS instruments to determine their position. The first coordinate is the latitude, which is the angle of a point North or South of the equator. The second coordinate is the longitude, which measures the angle of point east or west of the prime meridian. The format of the system can be in degrees, minutes, and seconds, or as decimal degrees (DD). DD will be used in this work to reduce the number of calculations. Northern and easterly directions will be defined as positive throughout this work [30].

- Mercator projections stretch the Earth's surface so that all lines of longitude are parallel. This type of projection distorts the view of Earth such that areas near the poles are stretched out more than near the Equator. This distortion makes measuring distances more complicated [54]. On a positive note, projecting the Earth onto a flat surface in this fashion aligns it with a Cartesian coordinate system, which simplifies location of points using a cursor when coding from scratch.

- Universal Transverse Mercator (UTM) projections project a local Cartesian grid onto a region of the Earth's surface, reducing the distortion found in the traditional Mercator projection [25]. UTM coordinates expresses horizontal coordinates in three parts: zone, easting, and northing. The area being surveyed in this work is confined to zone '18 N'. The '18' value represents a range of longitudes and the 'N' value specifies that it is in the northern hemisphere. The next coordinate is the easting, which measures the distance in meters from a local north/south datum. The final coordinate is the northing, which measures the distance north or south of the equator. A point cloud referenced in UTM offers a convenient Cartesian system to work with, which greatly simplifies calculations. Elevations are typically expressed as distance above mean sea level [54].

3.3.2 Sensitivity to perspective

Feature extraction algorithms used in SfM applications rely on the quality of the collected imagery, which is used to extract features. An image taken at an oblique angle will demonstrate a single-point (or two-point if both yaw and lookdown angle are not correct) perspective where parallel lines will converge onto a single point, as shown in the left image of Fig 3.4. An image of a roof taken at a smaller oblique angle to the surface will show more evenly spaced shingles throughout the image, as shown in the image to the right of Fig 3.4. Shingles appearing near these vanishing points in the image will appear noticeably smaller than closer shingles, despite having the same physical dimensions. If there is no lens distortion, the number of pixels that represent a flat object should decrease as the optical axis of the camera is moved away from the perpendicular position, until the sensor's FPA is parallel with the flat object. This will result in a loss of features due to the reduced resolution near the point of convergence [49].

3.3.3 Sampling distance

Another important factor in feature extraction is the distance from the target. As a target's distance is increased from the sensor, the number of pixels representing that target is reduced. Typically, the term ground sampling distance (GSD) is used to represent

Figure 3.4: Left - Non-orthoginal imagery with single-point perspective. Right - As the vantage point moves toward an orthogonal position, the perspective is reduced.

the distance each pixel represents on the ground. GSD is calculated using Eq 3.1 and illustrated in Fig 3.5 [50, p. 606]. When imaging a 3D target at close range, this term loses relevance since the GSD decreases closer to the sensor, where a 3D target's distance to the sensor will vary at different points. Ignoring distortion, if all imagery of a roof face is orthogonal to the roof face, the sampling distance along the roof plane should be constant, however, any out-of-plane objects will have a different spatial resolution. Since only roof planes are of interest, the term GSD can be misleading, so spatial resolution of the roof will be referred as roof sampling distance (RSD). Instead of using altitude above ground level to calculate GSD, the roof offset distance will be substituted to provide the RSD. Any point other than those on the roof plane being imaged (eg. walls, a different roof surface, the ground) will have a different resolution. A smaller RSD will result in more features, which provides SfM software additional features to compare. From the author's experience using SfM applications, imaging too close to homogeneous surfaces, like shingles, might result in a somewhat repetitive pattern which could interfere with sensor location calculations and cause mesh tiles to be misaligned or multiple instances of an object being created in the point cloud.

$$GSD = z * w/f \tag{3.1}$$

- GSD is the ground sampling distance [m]
- z is the distance above ground [m]
- w is the width of a pixel [mm]
- f is the focal length [mm]

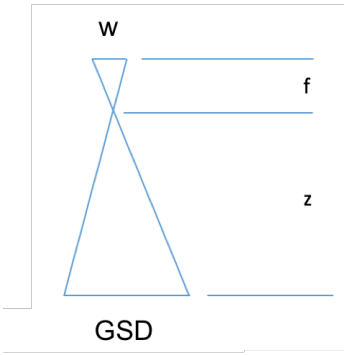

Figure 3.5: Relationship of GSD, focal length, height, and pixel size.

3.3.4 Building feature extraction

Commercial computer vision-based photogrammetry software applications output dense point clouds from which a texture mesh can be generated. This amount of data can be overwhelming as it could consist of several million points. In order to generate a flight plan to fly parallel to a roof face, points corresponding to the roof must be extracted. The following steps describe the general process of identifying individual roof faces.

There are several building extraction algorithms available. For this work, the solution implemented by Sun [58] was explored. Sun used data from a LIDAR point cloud, but the principle is equally applicable to point clouds generated from a SfM application.

This automated process first extracts trees and shrubs. This is done by measuring the flatness of small local neighborhoods. Comparing the quotient of the first eigenvalue and the sum of all eignevalues in the neighborhood provide a sense of flatness, as seen in Eq 3.2. If perfectly flat and horizontal, the third eigenvalue will be zero due to a lack a of variability in the orthogonal direction. Comparing the variation of normal vectors in a local neighborhood, as defined in Eq 3.3, is also considered. Points representing foliage will have a somewhat random distribution of normal unit vectors, whereas a roof surface is planar in nature and normals will be relatively consistent as long as a large enough sample is collected. Graph cuts is then applied to classify points as 'vegetation' or 'other'[57] based on these statistics.

$$F_f = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3} \tag{3.2}$$

- F_f is the Flatness estimation

- λ_n is the eigenvalue of a local neighborhood of points.

$$C_p^n = \frac{1}{k} \sum_{q \in N^k} \hat{\mathbf{n}}_\mathbf{q}^\mathbf{T} \cdot \hat{\mathbf{n}}_\mathbf{q} \tag{3.3}$$

- $\hat{\mathbf{n}}_\mathbf{q}$ is the normal vector
- C is the Variation of normal directions
- k is the number of points in local neighborhood, which is in the set of k natural numbers.
- q is the an arbitrary index representing the normal vector
- p is the index representing the local neighborhood

With the trees extracted, the remaining scene can be categorized into buildings and terrain. These can be separated by Euclidean clustering, since there is an elevation change between the surface and rooftops. When the terrain is removed, only buildings remain. The boundary of these points will reveal the footprint of the building and plane fitting will permit a simplified 3D representation of the buildings. Roof features, such as air conditioning units can be identified by assessing the curvature and normal vectors within a specified tolerance [57].

The vertices of the building can now be determined, providing enough information to produce an '.obj' mesh file that contains a watertight triangular mesh of the building consisting of only the main vertices of the building. Roof faces in an '.obj' mesh can be extracted by finding vertices on the same plane that are on the same surface.

Both roofs and meshes have flat surfaces. To avoid confusion, I will refer to triangular mesh components as mesh facets, and roof surfaces will be referred to as roof faces. A roof face will likely consist of two or more mesh facets.

Other methods of obtaining roof vertex coordinates include conducting a scene survey of the building, importing coordinates from a database, or manual selection from an existing point cloud. The automated process has its advantages, but requires iterative adjustment of parameters to successfully extract the building. Depending on the complexity of the roof structure, automatic extraction may or may not be a desirable method. Conversely, a successful survey also requires access to equipment to accurately geolocate building vertices.

3.3.5 Roof face feature extraction

Sun's automated building feature extraction process [57] extracts individual roof faces within his code, but they are not part of his output. Also, roof vertices may be available

through other means, therefore, a way to extract individual roof faces from an '.obj' mesh is still required. All points on a plane can be expressed by Eq 3.4, but it is important to note that just because two points are on the same plane, they are not necessarily on the same roof face. For example, one construction design commonly found in townhomes and low-rise buildings involves staggering nearly identical units fore, or aft, along the length of a building as shown in Fig 3.6. This must be taken into account when extracting individual roof faces.

$$ax + by + cz + d = 0 \tag{3.4}$$

- x,y, and z are Cartesian coordinates
- a,b,c, and d are coefficients describing plane position and orientation

Figure 3.6: Townhouse with multiple, parallel roof faces.

A roof face's normal vector is significant because it can be used to generate parallel planes, determine the orientation of a roof face, and serve as the basis for orienting the direction of evenly spaced points across a plane. This plane is where the sensor needs to be placed for orthogonal imagery to be acquired at a constant offset distance from the roof face. The direction of the normal vector of a triangular mesh facet can be calculated by determining the cross product of any two vectors connecting points in a triangle, as highlighted in Eq 3.5. If many points from a point cloud are close to a common plane, the normal vector can also be determined using principal component analysis (PCA). The first and second principal components (PCs) will be in-plane and the third PC will be orthogonal to the first two PCs. The third PC is the unit normal vector. If all points

are perfectly aligned along a plane, then the normal vector (i.e. third PC) will have zero length [24, pp. 594]. In this case, the direction of the normal vector can be taken from the cross product of the first two PCs. Intuitively, all surfaces on one face of a roof should be on the same plane with the exception of variations caused by shingles, solar panels, etc. When an '.obj' mesh file is rendered, one roof face may consist of several triangular mesh facets, which may not be perfectly aligned, but some variation can be dealt with by applying a threshold to the components of the normalized normal vector. Regardless of which technique is used, the normal vector may point upwards or downwards when calculated. To ensure that a normal vector is oriented upwards, the unit normal vector can be scaled by negative one if its z-component is negative.

$$\mathbf{n} = \mathbf{p_{12}} \times \mathbf{p_{23}} \tag{3.5}$$

- $\mathbf{n}$ is a '3 x 1' normal vector (non-normalized)
- $\mathbf{p_{12}}$ is a '3 x 1' vector connecting two points 1 and 2
- $\mathbf{p_{23}}$ is a '3 x 1' vector connecting two points 2 and 3

The normal vector can be used to identify and disregard walls by rejecting faces that have a unit normal vector with a z-value of approximately zero. The floor of the building can also be rejected by finding the lowest building face where the normal vector's z-component is approximately one.

Several criteria of mesh facet properties can be used to determine if mesh facets belong to the same roof face or not. To track these in an efficient way, logical matrices can be used for each criterion. If a mesh contains 'n' mesh facets, then a logical matrix would be of size 'nxn', containing ones and zeros. A '1' would indicate that the criterion has been met between the mesh facets that correspond to the row and column indices, and a '0' would indicate that it has not been met. Each diagonal element of the matrix must be a '1', since it is being compared to itself. Each logical matrix will also be symmetrical. When the logical matrices are multiplied in an elementwise fashion (Hadamard-Schur product), denoted as '$\odot$', in Eq 3.6, mesh facets that meet all of the criteria can be identified by a '1', and if one or more criteria are not met, the cell would be a '0', as shown in Fig 3.7 [45].

$$M = C_1 \odot C_2 \odot ... \odot C_k \tag{3.6}$$

- M is the logical matrix for all criteria
- C_k is the logical matrix for criterion 'k'

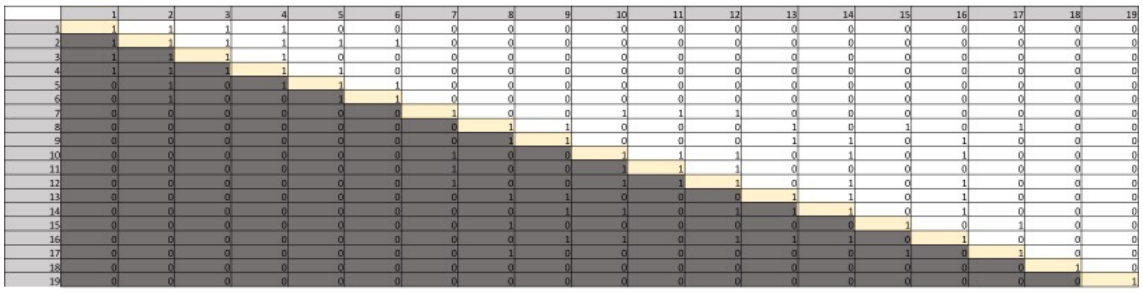

Figure 3.7: Example of a logical matrix showing faces with common properties. In this case, all facets from an '.obj' mesh file that share a common normal vector within a specified threshold are indicated by a '1' in the appropriate row and column to form a symmetrical matrix. This process is repeated for facets that share at least one common vertex.

3.3.6 Roof face orientation

The x- and y-components of the normal vector indicate the orientation of a roof face. A four quadrant inverse tangent function (i.e. atan2d in MATLAB) provides the orientation of the roof, as shown in Eq 3.7 and Fig 3.8 [19]. It is important to note here that all angles will be determined in polar coordinates, i.e. angles will be measured counter-clockwise from the x-axis, where the x-axis will point eastward, and the y-axis will point northward. Navigational angles are determined clockwise from north. This work adopts the convention of working in polar coordinates and only converting to navigational angles when required to do so.

$$\theta = tan^{-1}\left(\frac{n_y}{n_x}\right) \tag{3.7}$$

- θ is the angle measured counterclockwise from the x-axis (east)[1]
- n_x is the x-component of the normal vector
- n_y is the y-component of the normal vector

3.3.7 Creation of a plane that is offset and parallel

The normalized unit vector for each plane can be used to create a plane that is parallel and offset by multiplying it by the required distance and adding this result to each point

[1] 'tan^{-1}' represents the four quadrant inverse tangent

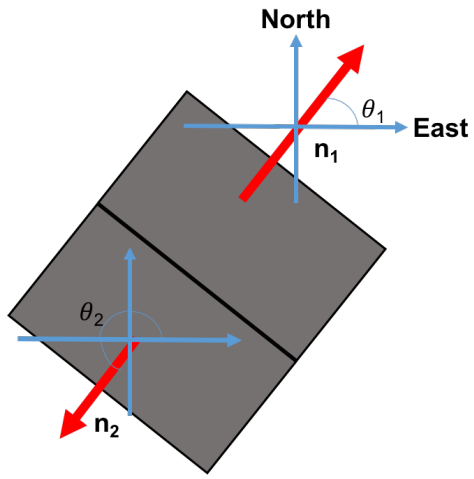

Figure 3.8: Roof face orientations of a gabled roof using the normal vector of a roof face

in a plane as shown in Eq 3.8 and Fig 3.9. This offset and parallel plane is where the aircraft's sensor needs to be positioned to maintain a constant distance from the roof, and the sensor's optical axis is aligned parallel to the normal vector of the roof face.

$$\mathbf{p_2} = d\hat{\mathbf{n}} + \mathbf{p_1} \tag{3.8}$$

- d is the distance between the old plane and the new plane
- $\mathbf{p_1}$ is the point on the roof plane
- $\mathbf{p_2}$ is the point belonging to the offset plane

3.3.8 Determine capture locations

Now that the offset plane is created, the points on this plane, where the images are to be taken, need to be determined. The spacing will be based on the overlap as previously discussed, but their orientation can be based on PCA [16].

Orientation of image capture locations

The first two PCs of a triangular plane's vertex coordinates will be in-plane. If all vertices lie exactly on this plane, then the third PC will be zero. Otherwise, the third PC will be the normal vector of the plane [16].

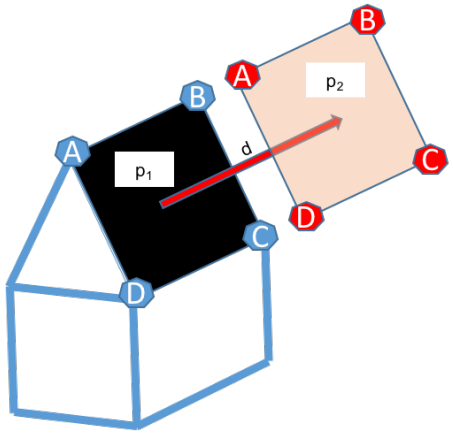

Figure 3.9: Creating a plane parallel to a roof face

These PCs can be used to generate the direction of a grid of desired sensor positions, but the orientation of one of these vectors should be aligned horizontally to minimize climbing, since climbing is the least efficient flight mode for an aircraft [46]. Knowing this, a logical approach is to navigate the aircraft from side-to-side on a horizontal leg to image along a path that is parallel with the roof face, and then climbing parallel to the roof to conduct the next leg, as shown in Fig 3.10. The direction of the legs will be called the leg vector, $\hat{\mathbf{a}}$, and the advance vector, $\hat{\mathbf{b}}$. The z-component of the leg vector must be zero, because it is horizontal by design. All components of the normal vector are known. The x- and y-components of the leg vector can be found once constraints are applied to its length and orientation. Since the z-component is zero, and the length is unity, Pythagoras' theorem can be applied to generate one constraint (Eq 3.9), and the second constraint is that the dot product of the normal vector and leg vector is zero, due to their orthogonality (Eq 3.10). This leaves two unknown variables and two equations, making the problem solvable by substitution. The advance vector can be determined by taking the cross product of the normal and advance vectors (Eq 3.11). Should the z-component of the advance vector be negative, then the advance vector can be scaled by negative one to ensure that it points upwards.

$$\hat{\mathbf{a}} = \begin{bmatrix} d \\ e \\ 0 \end{bmatrix}, \hat{\mathbf{b}} = \begin{bmatrix} g \\ h \\ i \end{bmatrix}, \hat{\mathbf{n}} = \begin{bmatrix} j \\ k \\ l \end{bmatrix}$$

- $\hat{\mathbf{a}}$ - leg vector with unknown xy-coordinates (d,e), z- coordinate is 0.

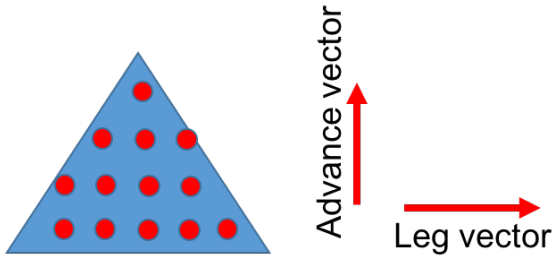

Figure 3.10: Spacing of camera locations along a triangular plane along the leg vector and advance vector.

- $\hat{\mathbf{b}}$ - advance vector with unknown xyz-coordinates (g, h, and i).
- $\hat{\mathbf{n}}$ - normal unit vector with known xyz-coordinates (j, k, and l).

$$1 = d^2 + e^2 \tag{3.9}$$

$$\hat{\mathbf{a}} \cdot \hat{\mathbf{n}} = \mathbf{0} \tag{3.10}$$

$$\hat{\mathbf{b}} = \hat{\mathbf{a}}\mathbf{x}\hat{\mathbf{n}} \tag{3.11}$$

Another way of looking at this problem is that the leg and advance vectors are, in reality, the first two PCs rotated about the normal axis until one of them is horizontal.

Sensor position spacing

The spacing between image capture locations is dependent on the required overlap and distance from the roof and the swath width and height. The swath width is determined from the dimensions of the FPA and the focal length of the lens, as per Eq 3.12, which is derived from similar triangles [50, p. 606]. The step size in the direction of the leg and the forward direction are determined using Eq 3.13 and applied along the leg vector and the advance vector.

Images can be taken at nearly any orientation, but a simple approach is to assume that all images are taken in a landscape fashion such that the longer edge of the sensor is parallel to the ground. This dimension, projected onto the ground, will be referred to as a swath width and the opposite direction is the swath height. The distance between images will be referred to as the step distance. The step distance along the direction of the advance vector will indicate the separation between imaging legs. The step distance in the direction of the leg vector will determine the distance between exposures required

by UgCS. Fig 3.11 shows that increasing the distance from the roof decreases the number of photos taken at a constant overlap due to the increased step size.

$$l = zw/f \tag{3.12}$$

$$s = (1 - X)l \tag{3.13}$$

- l is the swath width/height [m]
- z is the distance from roof plane [m]
- w is the width/height of the sensor [mm]
- f is the focal length [mm]
- s is the step distance [m]
- X is the overlap fraction in desired direction [-]

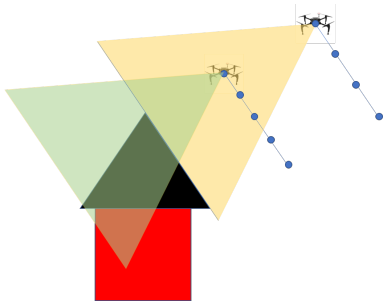

Figure 3.11: Effect of increasing the offset distance from the roof on distance between images for a constant overlap. [Image annotated from dji.com]

Starting at the lowest point of the projected plane, a grid can be built of desired image capture locations. Since roof lines may not be perfectly parallel, the first leg should be slightly higher than the lowest point on the roof face. An example of capture locations and flight path are projected on the xy-plane in Fig 3.10.

3.3.9 Sensor orientation

The sensor must be oriented along the normal vector of the plane, pointing towards the roof. Since the desired end-result of this flight plan is to collect orthogonal images, it is important for the aircraft to be oriented towards the roofplane at all times. This requires knowing the yaw and lookdown angles.

Lookdown angle

The lookdown angle (ψ) is the angle that the sensor's optical axis is oriented below the xy-plane, as can be seen on the left of Fig 3.12. A proper lookdown angle is necessary to ensure that a sensor's FPA is parallel to a roof face. This angle can be determined from either the normal vector or from the advance vector. Eq 3.14 shows the calculation of the lookdown angle based on the normal vector, $\hat{\mathbf{n}}$, as previously defined.

$$\psi = tan^{-}1\left(\frac{l}{\sqrt{j^2 + k^2}}\right) \tag{3.14}$$

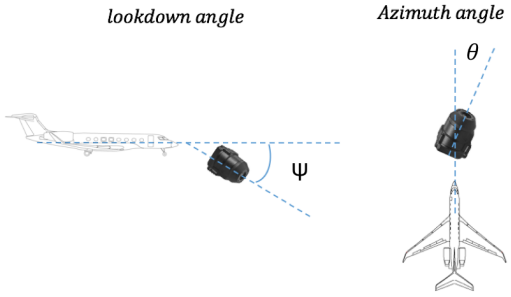

Figure 3.12: Sensor lookdown angle and azimuth angle. [Image from liteye.com and debonair.co.uk.com were used in this figure.]

Yaw

The term 'yaw' is defined within UgCS as the angle between the direction of flight and the direction the aircraft is pointing, however, yaw is actually the rotation of an aircraft about its normal axis. The proper term for the angle of relative motion is actually 'drift angle' [8]. The term 'yaw angle' will be used to remain consistent with DJI and UgCS terminology, as shown in Fig 3.13, whereas 'yaw' will refer to the rotation of the aircraft about its normal axis.

As with the lookdown angle, a proper yaw angle is necessary to ensure that the FPA is parallel with a roof face. The yaw angle is determined based on the building orientation and the direction to the aircraft's next point. This yaw angle orients the aircraft towards the roof face, regardless of the building orientation. Adjusting the sensor's azimuth instead

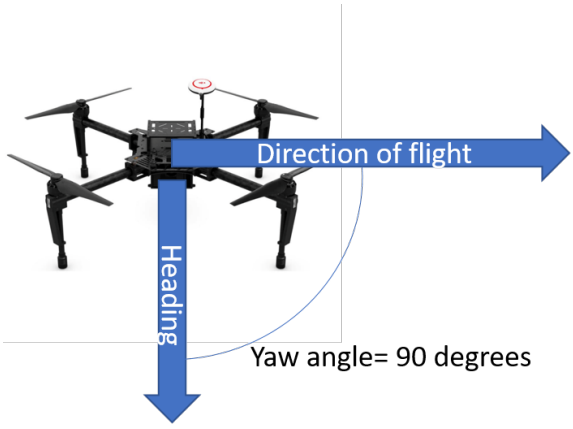

Figure 3.13: Yaw angle, as defined by UgCS. [Aircraft image from dji.com]

of yawing the aircraft would have also accomplished the same effect, but the aircraft's landing legs may have entered the camera's field of view at certain yaw angles, which is not desirable.

Aircraft limitations

Commercial-off-the-shelf sUASs have a limited number of waypoints that can be uploaded to the aircraft. For example, the DJI Matrice-100 is limited to 99 waypoints. Imaging a large building at close range could easily exceed this limit. This can be problematic when planning an intricate flight using an autopilot. Care must be taken to ensure a limited number of waypoints are passed to the aircraft.

It is also important to note that aircraft have a waypoint acceptance distance, which the aircraft must be within in order for the autopilot to advance to the next waypoint. These values can be specified in the UgCS application. The default for this value is approximately two meters.

Exporting to ground control software

Once a list of camera locations, yaw angles, camera angles, and image spacings has been created, these data must be put into a format readable by the ground control station. UgCS permits flight plans to be imported using an '.xml' structure. To transition from a matrix with various parameters, a script needs to be generated. UgCS requires initial data to determine where home is, and a variety of other safety parameters. Waypoint segment

nodes are then created. Any required camera actions are entered into camera action nodes and attached to the applicable waypoint node. A script can be used to convert the data into an '.xml' file, as implemented in previous work [55]. Fig 3.14 outlines the '.xml' structure used by UgCS.

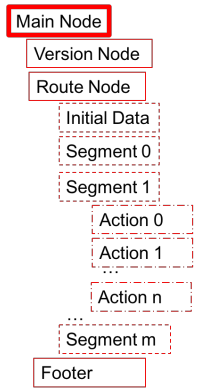

Figure 3.14: Example of an '.xml' structure that can read into UgCS.

3.3.10 Point cloud comparison

There are two main factors in determining the quality of a point cloud. First, the completeness of the point cloud addresses variations in the surface, as well as any occlusions that might exist. Secondly, position accuracy determines how well each point is located. This positioning error can be relative, by comparing the variations in points relative to each other, or absolute, by comparing a point to a known, geolocated position [48].

Comparing the accuracy of georeferenced point clouds is somewhat difficult, because it is difficult to determine precise measurements without surveying equipment. GPS.gov claims that the GPS hardware used in most cell phones provide horizontal position accuracy of approximately 5.0 m. More advanced equipment, such as dual frequency receivers, claim accuracy of up to approximately 5.0 cm with differential post-processing [47]. These accuracies can be considerably worse near buildings and trees as GPS signals that can cause a multi-path error. While geopositioning accuracy will be briefly addressed, focus will be on comparing overall dimensions with measurements taken with a measuring tape serving as reference data. Even a measuring tape will be subject to error due to sag, and stretching.

GPS errors can be reduced by differential post-processing. This is accomplished by logging GPS data at a point with a portable GPS for a period of time, and then com-

paring the data against GPS measurements data from a known GPS reference station. The reference station provides a correction based on the difference between its measured GPS position and the known location. Reference stations are located at multiple locations around the Earth and take real-time ionospheric measurements using dual-frequency carrier GPS receivers and determining real-time orbit positions. The correction is then applied to the portable GPS position to provide sub-decimeter accuracy [42].

Within the dense point cloud, it may be difficult to consistently find the specific point that corresponds to a point in the real world due to the shear number of points in the point cloud. This is highlighted by Fig 3.15, which shows a dense point cloud, centered near a roof vertex. There are several points that could be selected as the point representing the vertex, thus requiring a degree of subjective selection.

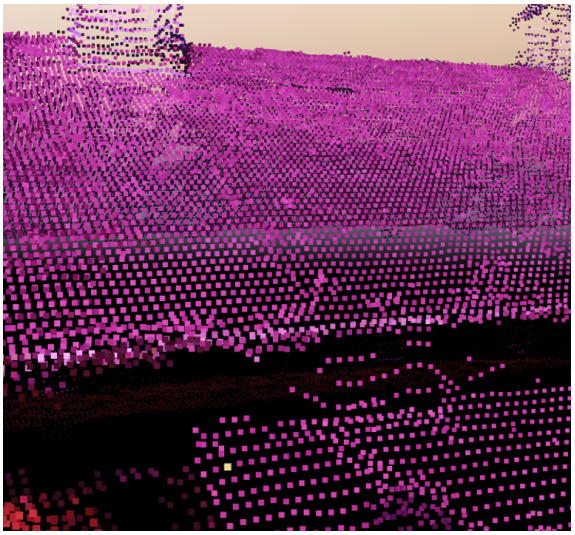

Figure 3.15: Initial dense point cloud of the Carriage Museum showing the distribution of points around a roof vertex.

In addition to finding the relative error between points in different point clouds, and comparing distances to measured lengths, off-plane deviation can show how well points are assigned to known planes. To do this, all points of a plane are projected onto the normal unit vector (i.e. the third PC) using Eq 3.15, and determining the standard deviation of these values using Eq 3.16 [20, pp. 72-75].

$$\mathbf{v} = \mathbf{P}\hat{\mathbf{n}} \tag{3.15}$$

- **v** is the 'n x 1' vector containing the projection of every point onto normal vector
- **P** is the 'n x 3' matrix of point cloud coordinates
- $\hat{\mathbf{n}}$ is the '3 x 1' unit normal vector

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(v_i - \mu)^2} \tag{3.16}$$

- σ is the standard deviation
- N is the number of points in the point cloud
- v_i is the i^{th} point of the point cloud projected onto the normal unit vector
- μ is the mean value of v

The slope of a roof face is determined using components of the normal vector as per Eq 3.17.

$$\alpha = tan^{-1}\left(\frac{n_z}{\sqrt{n_x^2 + n_y^2}}\right) \tag{3.17}$$

- α is the slope angle
- $n_{x,y,z}$ are the components of the normal vector

Point density was calculated by dividing the number of points by the area of the surface, as measured as per Eq 3.18.

$$density = \frac{A_{xy}N}{cos(\alpha)} \tag{3.18}$$

- $density$ is the number of points on a surface $[points/m^2]$
- A_{xy} is the area of surface projected onto the xy-plane $[m^2]$
- N is the number of points within the point cloud
- α is the slope angle of the roof face as determined by the normal vector [deg]

3.4 Hypotheses and objectives

This work was built around the central premise that orthogonal imagery would be better suited for point cloud reconstruction of rooftops in comparison with imagery taken at nadir. Fig 3.16 shows that occlusions, caused by the roofline, can be reduced when flying an orthogonal flight plan in comparison to a constant altitude flight. The orthogonal flight profile enables the sensor to be positioned at a shorter distance from the roof than a constant altitude flight, while maintaining a roughly constant distance from the roof throughout the flight. The downside is that greater positional accuracy is required to avoid contact with ground obstacles or the roof itself.

3.4.1 Hypothesis 1

A constant altitude flight will generate imagery to form a dense point cloud from which a safe, orthogonal flight plan can be generated.

Objective 1a

Generate and successfully execute orthogonal flight plans, based on an initial dense point cloud geometry, built using imagery taken from nadir.

Objective 1b

Assess the suitability of dense point clouds, generated from nadir imagery, using the coarsest settings available, to produce consistent vertex locations.

3.4.2 Hypothesis 2

Imagery generated from an orthogonal flight plan can generate a higher quality dense point cloud than imagery from a constant altitude flight.

Objective 2

Using the highest settings available within an SfM application, evaluate dense point cloud density, slope angle, and off-plane deviation of common roof faces. From these metrics, determine which method is superior.

3.4.3 Hypothesis 3

Using orthogonal imagery to produce a dense point cloud or mesh will produce superior results of vertical surfaces than overhead imagery.

Objective 3

Assess the qualitative differences between point clouds and meshes generated using nadir and orthogonal imagery.

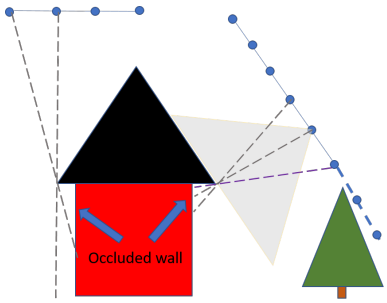

Figure 3.16: Effect of roofline on occlusions of the wall as seen from nadir and orthogonal imagery.

3.5 Methodology

The workflow shown in Fig 3.17 starts by building a watertight mesh of the building by either generating an initial dense point cloud, surveying an area, or using vertices already recorded in a database. From this watertight mesh, roof faces and datum planes of the same shape and size as each roof face were created above each roof face. Evenly spaced capture locations along this datum plane were determined from camera parameters and user-specified overlap. Based on the geometry of the roof, the sensor's optical axis was aligned to be perpendicular to the roof face. Images collected at these points and angles ensured objects of similar size would consist of approximately the same number of pixels across the frame.

Imagery for this collection was gathered over two buildings located at the Genesee Country Village and Museum near Rochester, New York using two sUASs. The specifications for these cameras and sensors are listed in Table 3.1. The DJI Matrice-100 uses a Zenmuse X3 sensor to control the sensor's azimuth, roll, and lookdown angle. The camera and UAV are controlled using the Universal Ground Control Software (UgCS) application created by SPH engineering [4]. A functional red, green, and blue (RGB) gimbaled camera was not serviceable for the Matrice-100, so a gimbaled sensor with red, green, and near infrared (RG-NIR) bands was used to generate false color imagery. A DJI Phantom 3-4K with stock sensor provided RGB imagery. Both cameras used the same framing sensor,

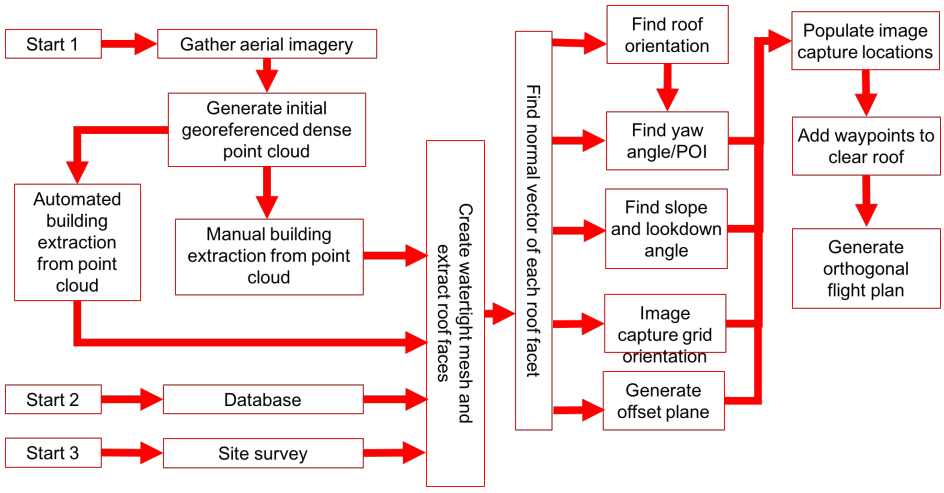

Figure 3.17: Workflow to generate point clouds from an orthogonal flight.

but have different gimbals and optics. The Phantom 3-4K platform is not compatible with UgCS, but was controlled manually to collect imagery from nadir. Fig 3.18 highlights the two imaged buildings: The Garden Restaurant (GR) and the Carriage Museum, while Fig 3.19 shows greater detail of the Carriage Museum (left), and the Garden Restaurant (right).

3.5.1 Flights

All collections were conducted over four separate days. Parameters from each flight are tabulated in Table 3.2.

3.5.2 Initial dense point cloud generation

Point clouds were generated using Pix4D Mapper Pro Desktop on a desktop computer. Images used during the point cloud generation were all 4000 X 3000 'JPEG'. The file sizes of the images from both platforms ranged from 4.8 to 6.1 MB. Images were not modified between downloading the images from the aircraft and uploading them into the Pix4D application.

The first goal of this process was to generate an initial dense point cloud as quickly as possible. To accomplish this, the image scale setting during the sparse point cloud generation was set to be as small as possible. On occasion, the software failed to generate

Table 3.1: Technical specifications for the Matrice-100 and Phantom3-4K. Data transcribed from DJI.com.

	Phantom 3 -4K	Matrice-100
mass (g)	1280	2838
endurance (min)	25	35
filter	RGB	RG-NIR
Sensor	Sony EXMOR 1/2.3" CMOS	
width (mm)	6.17	
height (mm)	4.55	
pixels	4000 x 3000	
pixel size (um)	1.54	
focal length (mm)	3.57	
field of view (deg)	94	

Figure 3.18: Location of buildings imaged.

Figure 3.19: Image of the Carriage Museum (left) and Garden Restaurant (right).

Table 3.2: Flight Parameters.

Flight	Building	Height (m)	Height (m)	Sensor	angle
A	CM	23	23	NIR	nadir
B	CM	23	23	RGB	nadir
C	CM	30	30	NIR	nadir
D	CM	30	30	RGB	nadir
E	GR	23	23	NIR	nadir
F	GR	23	23	RGB	nadir
G	GR	30	30	NIR	nadir
H	GR	30	30	RGB	nadir
I	GR	5	5	NIR	ortho_partial
J	GR	10	10	NIR	ortho_partial
K	GR	20	20	NIR	ortho_full
L	GR	30	30	NIR	ortho_full

a sparse point cloud, due to an error stating an inability to calibrate cameras. This error indicates that the position of the camera could not be found and was resolved by increasing the initial scaling factor incrementally until a sparse point cloud could be successfully rendered. Once complete, a dense point cloud was generated using a one-eighth scaling factor. The information from the dense point cloud was used to generate the flight plan. For later comparison, another dense point cloud was generated using maximum scaling options for the sparse and dense point cloud renderings. Subsequent point clouds generated from the orthogonal flight plan were generated using the maximum possible scaling options. All other parameters within Pix4D were not changed from their default values.

Following sparse point cloud generation, a processing area was manually selected to exclude much of the area outside of the building. This was done to reduce the overall runtime, since all points outside of the footprint of the processing area are ignored by Pix4D software during point cloud densification.

Generation of the sparse point clouds sometimes resulted in misaligned tie points, creating multiple instances of the scene that were oriented in different directions. Whenever this occurred, 2D GCPs were added to the photos and generation of the sparse point cloud was re-started. It would have been preferable to avoid this step, since this could have artificially improved the accuracy.

3.5.3 Roof face - feature extraction

The Carriage Museum has a gabled roof consisting of two roof faces connected along the peak, so they are interconnected, but the two faces do not share the same normal vector. Conversely, the overhang over the entry might appear parallel to one of the two main roof facets, but it is not adjacent to any other roof face. Roof faces typically consist of multiple mesh facets that are contiguous, and share a normal vector.

An '.obj' mesh file consisting of the vertices can be generated from an automated building extraction tool, a database, or manual entry based on point cloud data. Sun's automatic roof extraction algorithm does this, but it is not a direct output. The process of labeling the surfaces was easier to implement from scratch than to find the necessary algorithms within his code.

An '.obj' mesh file labels vertices, and each of the 'N' triangular facets, and indicates which vertices belong to which facet. Visually, a user can look at the building and label each roof, but an algorithm can also be used to identify mesh facets belonging to each roof face.

From the mesh file, the normal vector of each facet was calculated and placed in 'N x N' commonality matrices. This was also done for each mesh facet that shares at least one vertex. The elementwise product was then calculated to produce matrix 'M', which disassociates facets that are adjacent, but do not have the same unit normal vector. Mesh

facets that corresponded to the first roof face were identified by starting with the first mesh facet (i.e. the first row) and listing all other mesh facets in 'M' the share the same features. These mesh facets were also searched for other common facets. This process was repeated until all appropriate mesh facets were assigned to the first roof face. When the first roof face was completed, the remaining roof faces were assigned mesh facets in the same manner.

3.5.4 Automated flight plan generation

The creation of a flight plan based on an initial point cloud requires that a suitable safety distance is maintained between the aircraft and the structure. The aircraft was subject to its own GPS error, as well as any error in the geolocation of the initial dense point cloud. Once the faces have been extracted, the camera locations were determined. For safety reasons, no flights closer than five meters from the roof surface were attempted. This limitation was based on the horizontal position accuracy of most GPS receivers.

Camera locations

Once each roof face was extracted, parallel planes were created based on the required offset distance. Camera positions were then determined based on user-specified overlap, as shown in Fig 3.20. These camera points are aligned horizontally for efficiency along the leg vector. This was repeated for additional legs at a specified overlap in the direction of the advance vector until the plane was filled with the desired sensor locations. Lookdown and yaw angles were then calculated. At this point, it is important to note that using these yaw angles may not be possible. During initial orthogonal collections, the Matrice-100 did not react consistently when given yaw instructions, sometimes continuously yawing while moving from waypoint to waypoint. This was remedied for each roof face by finding a distant point of interest, 100 km away, for the aircraft to orient itself towards while moving along the imaging plane.

Roof collision avoidance

Transitioning from one roof face to another could be potentially hazardous, since some roof planes may be lower than the peak, potentially causing the aircraft to collide with the roof when transitioning from one plane to the next. To remedy this, waypoints were added when changing between imaging planes. For each transition, the aircraft climbed vertically over the last point on that plane, moved directly over the first point on the next plane, before descending and then continued to capture imagery.

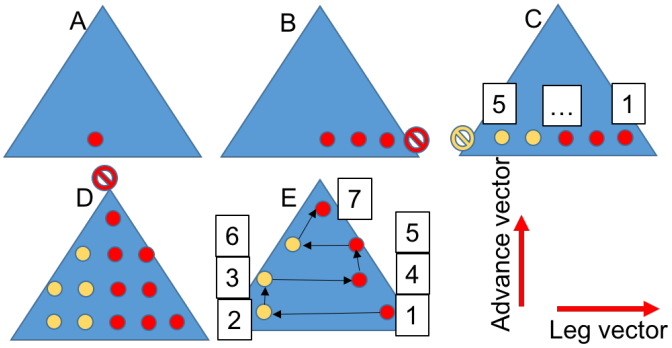

Figure 3.20: A – An initial point is found between the left and right extremes, slightly above the lowest point. B – Moving along the leg vector additional capture locations are determined until outside the triangle. C – Step B is repeated in the opposite direction and the points are arranged in sequence. D – Steps B and C are repeated by incremental distance in the direction of the advance vector until the region has been filled. E – Remove capture points, leaving only the end points for each leg and reorder the waypoints in a logical order.

Vertical offset

While generating the initial sparse point cloud, the ground elevation was different than data from UgCS's DEM indicated by an excess of 30 m. This difference was not consistent between the different initial dense point clouds. This error was corrected by subtracting the elevation of the takeoff point, as measured within the DEM, from the elevation from a surface point in the initial sparse point cloud. The need for this offset may not be required if proper ground control points are determined and entered into the SfM application immediately prior to point cloud densification, which will be discussed later. A significant part of this offset could also be explained by inconsistent usage of height above ellipse (HAE) altitude and above mean sea level (MSL) altitude between applications.

XML flight plan

The tabulated waypoints were converted into latitude and longitude using a script found online [43]. These waypoints and camera actions were converted into an UgCS-readable '.xml' file using an iterative script [55]. These flight plans ensured that the sensor's FPA was aligned parallel with a roof face at a specified distance, thus ensuring orthogonal imagery.

3.5.5 Point cloud analysis

The quality of the point clouds was compared using a variety of metrics. Pix4D permits sections of the point cloud to be labeled and isolated for a face-by-face analysis. Since the entire Garden Restaurant could not be imaged for each distance, only features on three faces on the south facade were assessed. Point density, off-plane variation, and roof slope angles were assessed for each of the flight profiles. Points within each dense point cloud were assigned to be part of the roof face being analyzed by their proximity to significant features that could be identified in each model. Edges were also selected on the basis of accessibility that could be referenced against real world measurements, as shown in Fig 3.21.

Figure 3.21: Identification of Garden Restaurant roof edges used for length measurement comparisons.

For each dense point cloud generated, each roof face was individually selected and stray points were subjectively removed. The roof faces were labeled as shown in Fig 3.22. The remaining points were exported from the Pix4D application for separate analysis. It was not always clear from visual observation which points belonged to which surface and what points were noise. A subjective call had to be made as to which points to exclude from a surface. As a general rule of thumb, points with improper color were removed when possible. These points were artifacts generated when an edge, or thin object, like a tree branch, overlapped a homogeneous scene like grass, or gravel. Points that were not connected to any surface, or points generated inside of a building were removed, because

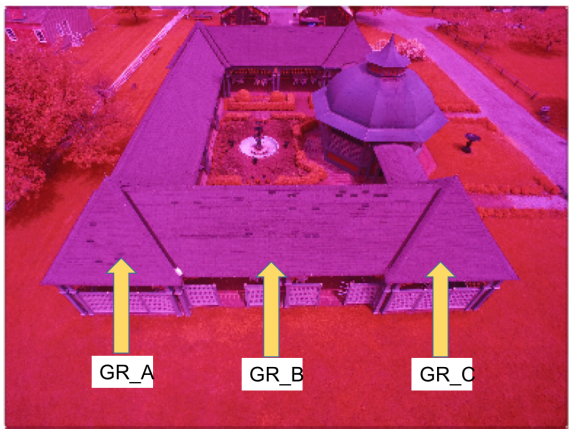

Figure 3.22: Screen capture of Pix4D showing the three roof sections used for comparison of dense point clouds.

they do not belong to the surface of interest.

Due to safety considerations, each area calculation was conducted using the area determined from each dense point cloud model. This removed the need for someone to take measurements on the roof. Using the density over an arbitrary area provides a normalized comparison for point cloud density comparison. Determining which points to include in a surface, was a subjective decision based on finding vertices that stood out in each point cloud model. The area used for the density calculation was determined using the area of the boundary of the points projected onto the xy-plane. Knowing the slope, the area of the surface was determined by dividing by the cosine of the slope.

Off-plane deviation was determined by projecting all points deemed to be part of a face onto the normal axis (i.e. the third PC). Since building surfaces were generally covered with uneven features including shingles, trim, windows, doors, etc., some deviation can be expected. A larger standard deviation is a likely indicator that the placement of points on that plane are not accurately placed.

Error analyses of slope and off-plane deviation, taken from point clouds, were conducted by randomly sampling each roof face for each dense point cloud 100 times and determining the standard deviation. This was repeated 100 times and the mean of these standard deviations was computed. Error bars were then created using plus or minus one standard deviation.

Effect of Ground Control Points

Ground control points (GCPs) can be added when rendering point clouds in Pix4D. A GCP permits a user to intervene and specify the precise Cartesian coordinates of any point in the sparse point cloud after initial processing (3D GCP), or in multiple photos prior to processing. Easily identifiable GCPs were assigned in the space surrounding the Garden Restaurant, as shown in Fig 3.23. The UTM coordinates of these points were measured using a Trimble Pro 6h GPS receiver (Fig 3.24) and differential post-processing was applied. Locations for these GCPs were selected to try to minimize GPS error due to the proximity of trees and buildings. Dense point clouds were re-generated several times for the 20 m orthogonal flight path, incrementing the number of 3D GCPs to determine the effect on the position of three ground reference points (GRPs) in reference to their positions measured with GPS and differential post-processing.

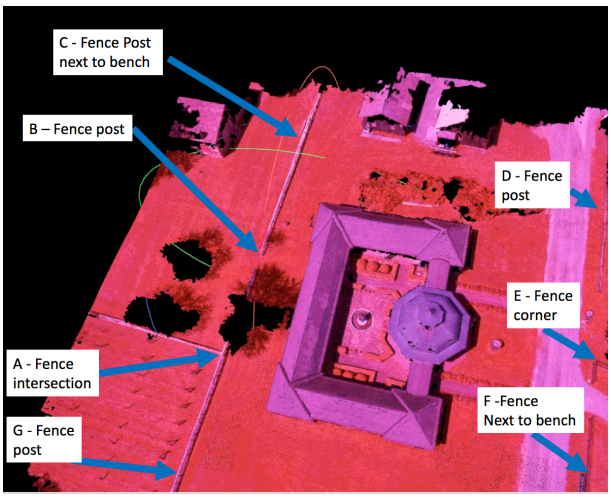

Figure 3.23: Locations of selected ground control points surrounding the Garden Restaurant. Points A, C, and E were used to evaluate changes resulting from incrementing the number of GCPs.

3.6 Results

The approach of generating flight plans directly from initial point cloud data has been tested for functionality, but the comparison of key metrics of point clouds is necessary to

Figure 3.24: Image of reference point E being measured using a Trimble Pro 6h. [Photo provided by Brittany Ambeau.]

determine if there is any advantage of conducting an automated flight in close proximity to the building.

3.6.1 Initial point cloud

Constant-altitude collections over the Garden Restaurant and the Carriage Museum at 23 and 30 m, produced images of sufficient quality to generate initial dense point clouds, as displayed in Fig 3.25.

Table 3.3 shows the cumulative averages of the standard deviations of the easting, northing, and elevation coordinates of roof face vertices broken down by altitude, and sensor type. The roof vertices within the dense point clouds were consistently placed within one meter of each other in the easting and northing coordinates. The vertical positions of the roof vertices were located within 1.8 m when corrected for the offset difference between the elevation of the takeoff location specified within the initial point cloud, and the DEM. This indicates that a vertical offset is critical before attempting to fly in order to keep the roof in frame, and to prevent a crash. Easting, northing, and corrected elevation errors were much smaller than the advertised 5 m horizontal and 10 m vertical accuracies expected of GPS readings [47]. Without the vertical offset, the elevation was

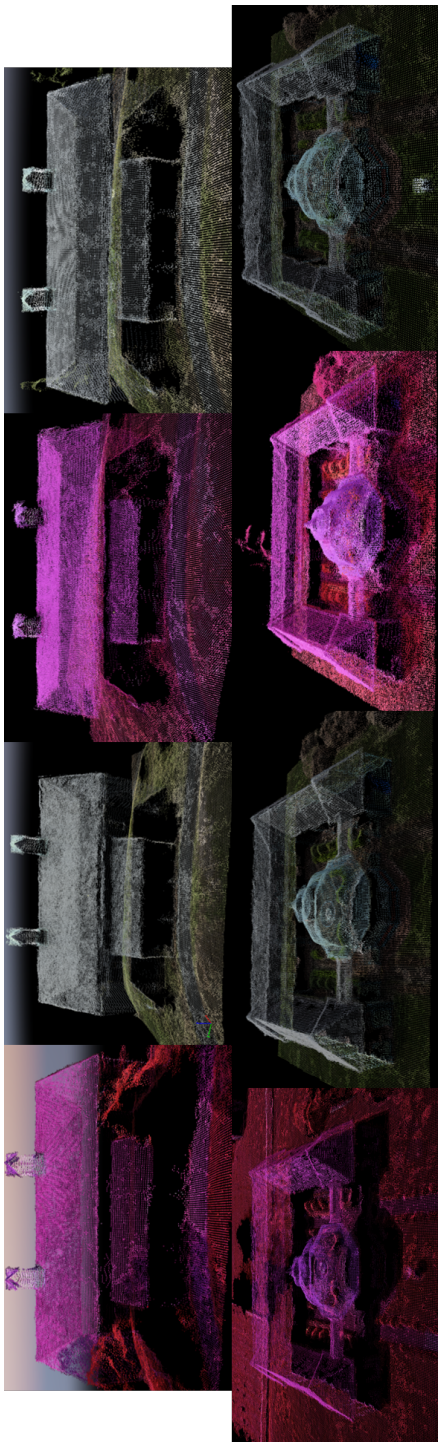

Figure 3.25: Screen captures of the coarsest point clouds generated from each of the eight initial flights. From left to right (23 m rg-nir, 23 m rgb, 30 m rg-nir, 30 m rgb.

Table 3.3: Standard deviations of each roof vertex calculated from flights at 23 m and 30 m, with the RGB and RG-NIR sensors, and for all flights.

Mean standard deviation [m]				
Metric	Easting	Northing	z	z-offset
23 m	0.42	0.57	9.43	0.89
30 m	0.62	0.40	2.56	0.86
Matrice-100	0.10	0.53	5.84	0.84
Phantom 3-4K	0.48	0.55	4.03	1.80
overall	0.98	0.60	7.28	1.38

underestimated between 7.4 and 36.7 m, which is unsuitable for mission planning purposes. This was recognized in the field when generating new flight plans. Easting and northing values were not compensated for horizontal position error, since there was no georeferenced position available at the time of flight.

Fig 3.26 provides a visualization of the position of each vertex measured from each of the flights. For each building, a single set of measured vertices were connected to assist in visualizing the building. Visually, one can see that the differences are not completely random as there appears to be a translation of the building between each collection flight. This suggests that the accuracy can be improved by translating easting and northing measurements by constant values. A person wishing to conduct a roof inspection will not likely have access to an accurate GPS system to apply a correction, however, the person will have access to digital elevation model (DEM) data available in most ground control system applications. For each flight, the error appears to be relatively constant over the duration of the flights, so a horizontal correction may not provide any advantage if the time lapse between the overhead and orthogonal flight can be minimized, but the dataset is too small to be certain. Regardless, the effects of horizontal correction are explored later by employing 3D GCPs measured using an accurate GPS.

Table 3.4 shows three measurements taken using a measuring tape in comparison with the relative distances between vertices in the dense point clouds. These data show that for each measurement of the four point clouds generated, each of the three edges were all shorter than their measured length by 0.07 m to 0.45 m. This is likely due to the sampling distance used within Pix4D during generation of the point cloud. This means that when the flight plan is generated, it will consist of legs that are slightly shorter than reality. This is a positive indication from a safety aspect in that the legs will be slightly shorter and the aircraft is less likely to bump into obstacles. From an imaging aspect, this could reduce the number of photos being collected by up to one per leg, possibly causing some degradation in the final point cloud representation.

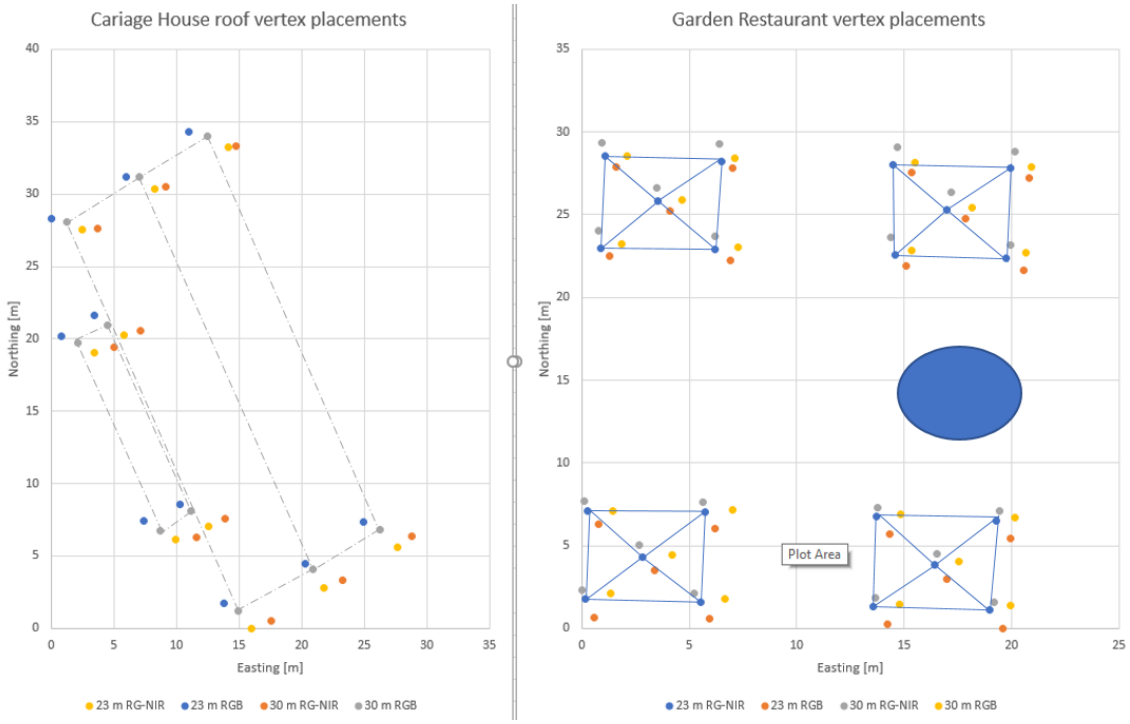

Figure 3.26: Location of roof vertices of the Carriage Museum (left) and Garden Restaurant (right) within point clouds generated from multiple flights. To aid in visualization, the blue circle denotes the Garden Restaurant's dome.

Table 3.4: Comparison of three roof edges as measured from a measuring tape and from multiple flights

Edge	Measured [m]	23 m rg-nir	Diff [m]	% diff % diff	23 m rgb	Diff [m]	% diff % diff	30 m rg-nir	Diff [m]	% diff % diff	30 m rgb	Diff [m]	% diff % diff
B12	5.64	5.37	-0.27	-4.7	5.37	-0.27	-4.8	5.21	-0.43	-7.5	5.29	-0.35	-6.2
A12	5.61	5.38	-0.23	-4.0	5.39	-0.22	-0.04	5.54	-0.07	-1.3	5.16	-0.45	-8.0
A14	5.64	5.39	-0.25	-4.4	5.47	-0.17	-3.0	5.48	-0.16	-2.8	5.39	-0.25	-4.4

3.6.2 Roof face extraction and identification

The automated building extraction code required considerable adjustment of the parameters to generate a reasonable representation of the Carriage Museum. The mesh generated in the automated roof extraction was also visually imperfect, as shown in the left of Fig 3.27. As a result, vertices were selected manually from the initial point cloud and a mesh was manually created from these coordinates. Automatic roof extraction from point clouds of the Garden Restaurant was not attempted due to a recent update of local servers that caused the code to stop working. Given either the auto-generated, or manually generated mesh, each roof face was uniquely identified by using automated roof face extraction that was described earlier. The results of the individual roof face extraction are shown in Fig 3.21. The image on the left is a watertight mesh, and the image on the right identifies each roof face as a separate colour. The east side of the Garden Restaurant was removed in these figures, because a dome is not well suited for the face-based approach being explored here.

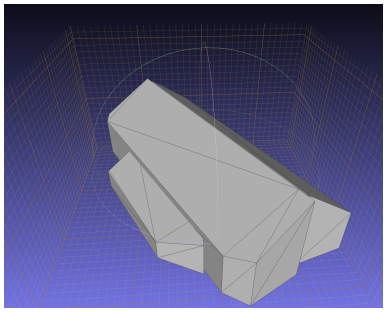

Figure 3.27: Image of the '.obj' mesh file of the Carriage Museum generated from an automated feature extraction algorithm based on a point cloud rendered from imagery acquired during a constant altitude flight.

Due to the imperfect shape of the mesh generated from the automatic roof extraction, meshes were generated manually using the coordinates of the Carriage Museum's roof face vertices from the dense point cloud generated from the lower altitude flight with the Matrice 100. This was also done for the north, south, and west sections of the Garden Restaurant, as shown on the left side of Fig 3.28. Flight plans were generated from these meshes and successfully imported into the UgCS application. On the right of Fig 3.28, roof faces are identified using a color coding system through an automated process.

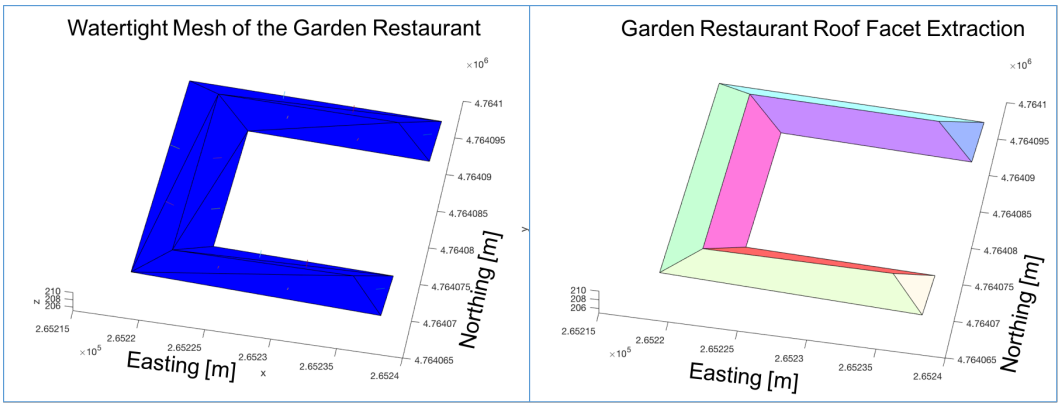

Figure 3.28: The image on the left was generated manually, using point cloud coordinates from the south, west, and north sides of the Garden Restaurant. The image on the right highlights each extracted roof face.

3.6.3 Flight plan generation and orthogonal imagery

An example of an orthogonal flight plan around the Garden Restaurant is shown in Fig 3.29. In this figure, the aircraft is offset 30 m from each roof plane. At this distance, the aircraft clears all obstacles and the entire flight plan was executed. Trees to the west of the Garden Restaurant and the dome prohibited the full orthogonal flight plan at 5.0 and 10 m, but full flights were conducted at 20 and 30 m. When attempting to fly an orthogonal flight plan around the Carriage Museum, the presence of obstacles forced the ends of some flight legs to be manually shortened within UgCS. The resulting imagery was affected by this adjustment, as the aircraft's yaw angle was less perpendicular to the roof. Flights could also not be conducted at 20 and 30 m from the Carriage Museum's roof faces due to further obstructions, forcing the use of the Garden Restaurant as the primary structure for dense point cloud analysis. In order to determine the effectiveness of the orthogonal flight plan, the south side of the Garden Restaurant was imaged at 5.0, 10, 20, and 30 m.

Fig 3.30 contains images taken using each of the automated orthogonal flight plans over the Garden Restaurant. The RSD increased for each time the distance from the roof plane increased from 5.0 m to 30 m. There is no significant single-point perspective noticeable along any of the three roof faces on the southern side of the roof.

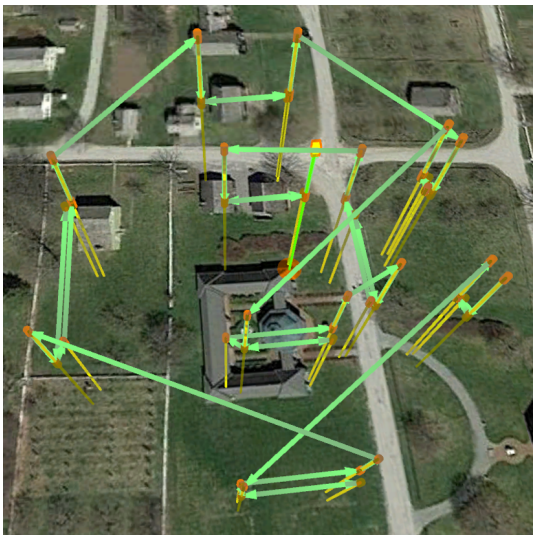

Figure 3.29: Screen capture of UgCS showing a successfully generated flight plan around the Garden Restaurant.

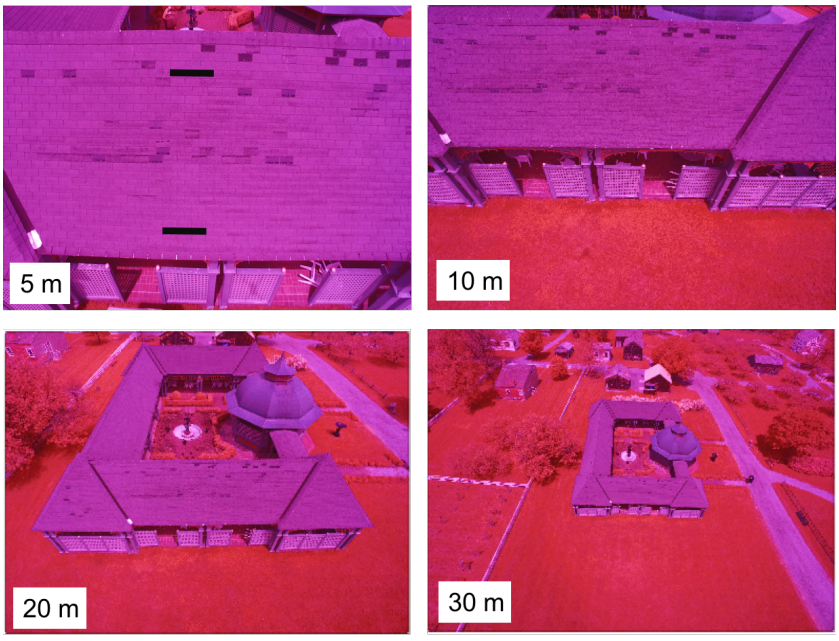

Figure 3.30: Imagery of the Garden Restaurant from varying distances.

3.6.4 Comparison of 3D models

As mentioned previously, the south side of the Garden Restaurant was the principal structure used for analyzing image quality. The three roof faces assessed are labeled in Fig 3.22. Due to safety considerations, measurements were limited to data that can be collected using a small ladder. During point cloud generation orthogonal imagery produced multiple instances of the scene. To resolve this, 2D GCPs were applied to the images prior to the sparse point cloud generation.

While substantial effort went into generating automated flight plans for rooftop inspections, metrics of the dense point clouds were compared to see if this process is an improvement over models generated from overhead imagery.

3.6.5 Qualitative comparison

The quality of the lattice work and shingles are shown in Fig 3.31 and 3.32 to visualize the difference in quality between the dense point clouds. These images illustrate dense point cloud representations generated at 5.0, 10, 20, and 30 m distances from the roof plane. Note that the shingles are 32 x 12.8 cm, and the lattice members are 4.2 cm wide, and spaces are approximately 5.5 cm x 5.5 cm. Visually, the greatest detail could be seen in the point cloud generated at 10 m.

The point cloud of the shingles of the generated from the 20 m was less detailed than the 30 m plot. Visual inspection of the images used to generate the point cloud showed blurred pixels, as shown in Fig 3.33. An assessment of the roof was performed in the frequency domain by cropping the same area out of each of the four photos used in the 10, 20, and 30 m point cloud models. The 2D Fast Fourier Transform (FFT) of these points was taken, and the power (P) (i.e. the square of the magnitude) [20, p. 259] was calculated for each cropped image. A circular mask with a known radius (m_r) was applied to the power spectrum starting with a radius of one and increasing until the mask reached the edge of the plot. The mask size represented spatial frequencies increasing from 0 m-1 to 60 m-1, as per Eq 3.19. The relative power (P_{rel}) was calculated for each increment by ignoring the power within the zero frequency point (i.e. the direct current point), and dividing the sum of the power within each mask (P) by the total power (P_tot) of the image as shown in Eq 3.20. The frequencies corresponding to a relative power of 0.95 are tabulated in Table 3.5. The imagery taken at 20 m from the roof had a lower frequency for bands one (NIR) and three (Green) than either the 10 m or 30 m images. The red band had a similar frequency to the 30 m band. The lower frequencies denote a lack of sharpness which could have affected feature extraction from the 20 m plot, affecting the position of many points.

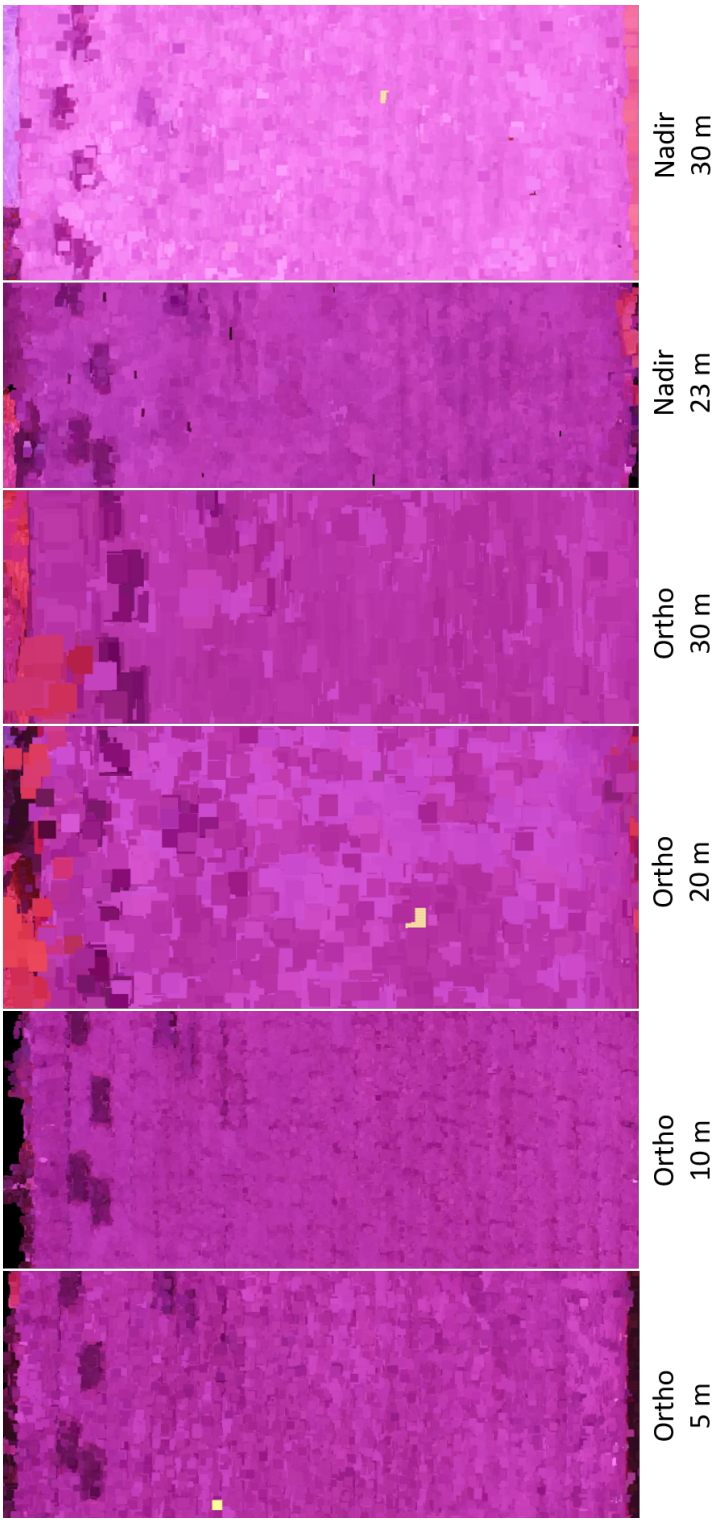

Figure 3.31: Screen captures of high resolution dense point clouds of the Garden Restaurant's shingles taken at 5, 10, 20, and 30 m from the roof plane, and 23 and 30 m nadir imagery.

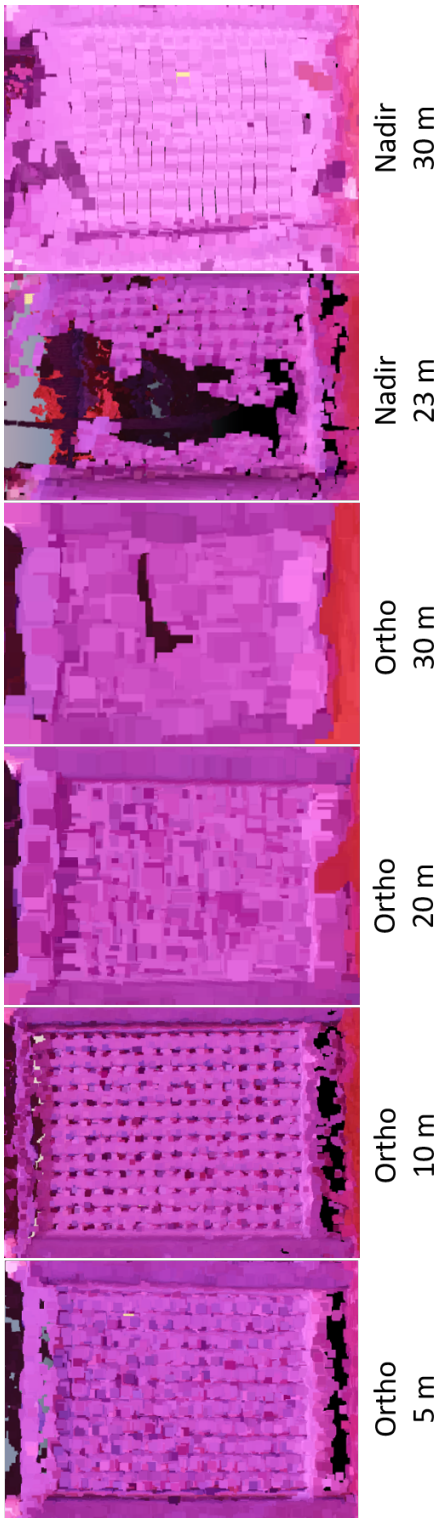

Figure 3.32: Screen captures of high resolution dense point clouds of the Garden Restaurant's lattice work taken at 5, 10, 20, and 30 m from the roof plane, and 23 and 30 m nadir imagery.

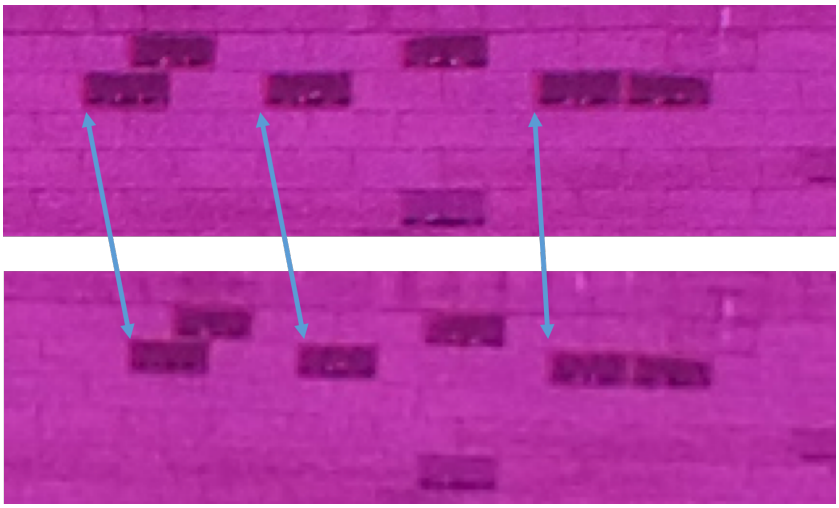

Figure 3.33: Top - Image from 20 m orthogonal flight showing a better resolution, but blur degrades the quality of the image, as denoted by the arrows. Right - Image from the 30 m orthogonal flight has a worse resolution, but blur is not as significant.

Table 3.5: Blur measurements for images taken at 10 m, 20 m, and 30 m

	Frequency corresponding to 95% of the relative power $[m^{-1}]$		
Distance from roof	Channel 1	Channel 2	Channel 3
10 m	36.87	42.54	35.67
20 m	35.9	40.71	34.3
30 m	36.49	40.33	35.05

$$\Delta u = \frac{1}{MRSD} \tag{3.19}$$

$$P_{rel}[u] = \frac{1}{P_{tot}} \sum P \odot m_r \tag{3.20}$$

The point cloud of the lattice generated from the 23 m orthogonal flight was has a large gap in the middle. The was the result of a lack of features detected within images that contained the lattice. These lack of features could have been from a lack of images containing the lattice, or a variation in illumination between the two flights. More images of the lattice can be obtained by expanding the collection area such that the lattice is on

the edge of the model, or using a higher overlap.

3.6.6 Slope gradient

The first metric that was assessed was slope. Slopes of georeferenced dense point cloud roof faces were calculated based on the face's normal vector (i.e. the third PC). Fig 3.34 shows the slope of the three faces of the south side of the Garden Restaurant, which were measured from dense point clouds generated from various flights. The slope of each roof face was also measured with an inclinometer in several locations. At each location, the slope was measured to be 45 +/-2 degrees.

For faces A, B, and C, the slopes measured from the dense point clouds were within five degrees of the measurement of the inclinometer for all of the orthogonal flight plans and the 30 m constant altitude flight. The lower constant altitude flight produced higher slope angles for all three surfaces than any of the other point clouds, which could not be explained.

3.6.7 Density

Point cloud densities were determined based on the number of points manually identified as part of the roof face. For the georeferenced point clouds, the boundary of points were projected onto the xy-plane and the projected area was calculated. The surface area was calculated by dividing the projected area by the cosine of the slope.

One of the principal hypotheses in this work was that orthogonal based imagery would improve point cloud quality. If valid, imagery taken closer to a target should intuitively generate denser point clouds due to the presence of features that would not be visible at greater distances. The data collected during these experiments support this hypothesis, but is not conclusive.

Fig 3.35 and Fig 3.36 show a trend where decreasing the orthogonal distance results in an increased number of points for a fixed area within the dense point clouds. These plots are similar in shape, but they should be scaled versions of each other, but the points were selected subjectively. This means that the area for each roof face in the various dense point clouds was never exactly the same.

Off-plane deviation

Off-plane deviation was assessed to give a sense of how well points were assigned to the appropriate roof face. Off-plane variation was determined by manually classifying points belonging to three roof faces for each point cloud model and projecting them onto the third PC, which is normal to the surface. This metric provides an idea of how well points

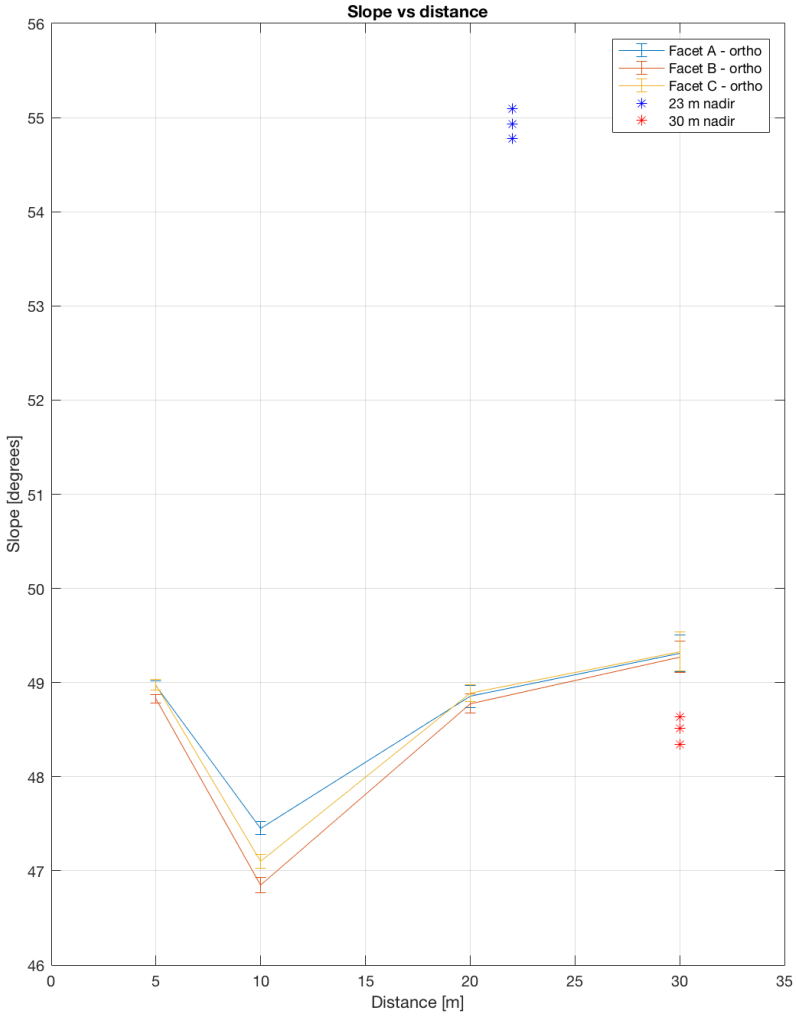

Figure 3.34: Slopes of Garden Restaurant roof faces.

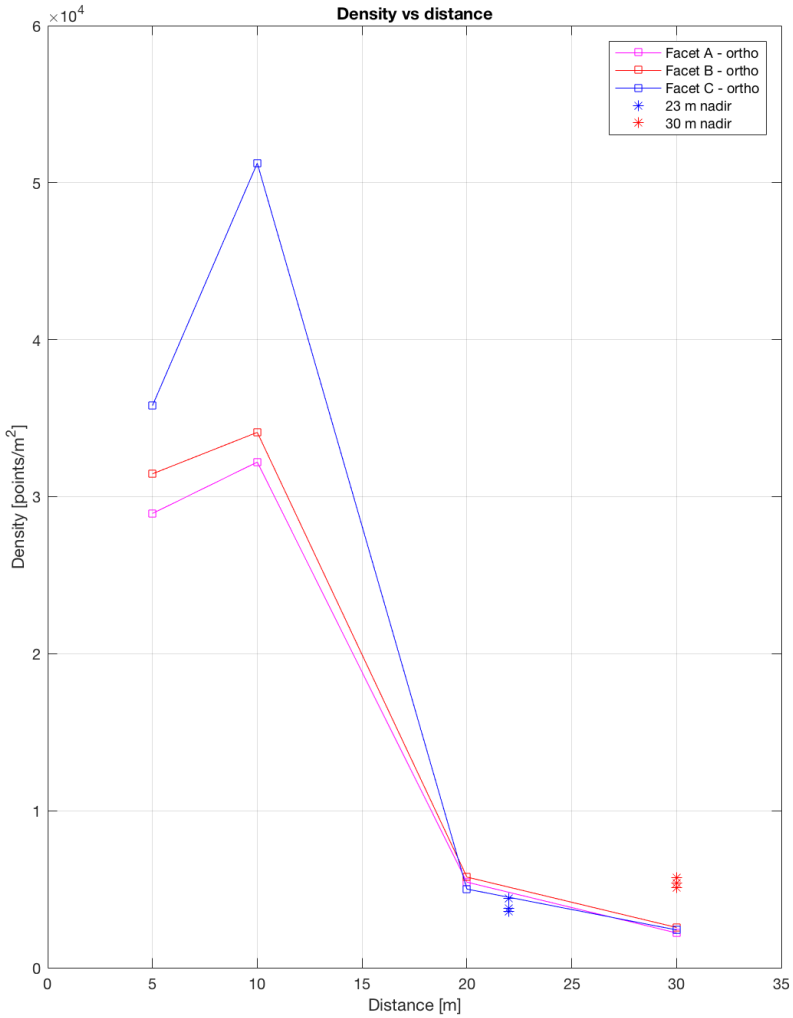

Figure 3.35: Plot showing the point cloud density generated from orthogonal and nadir imagery.

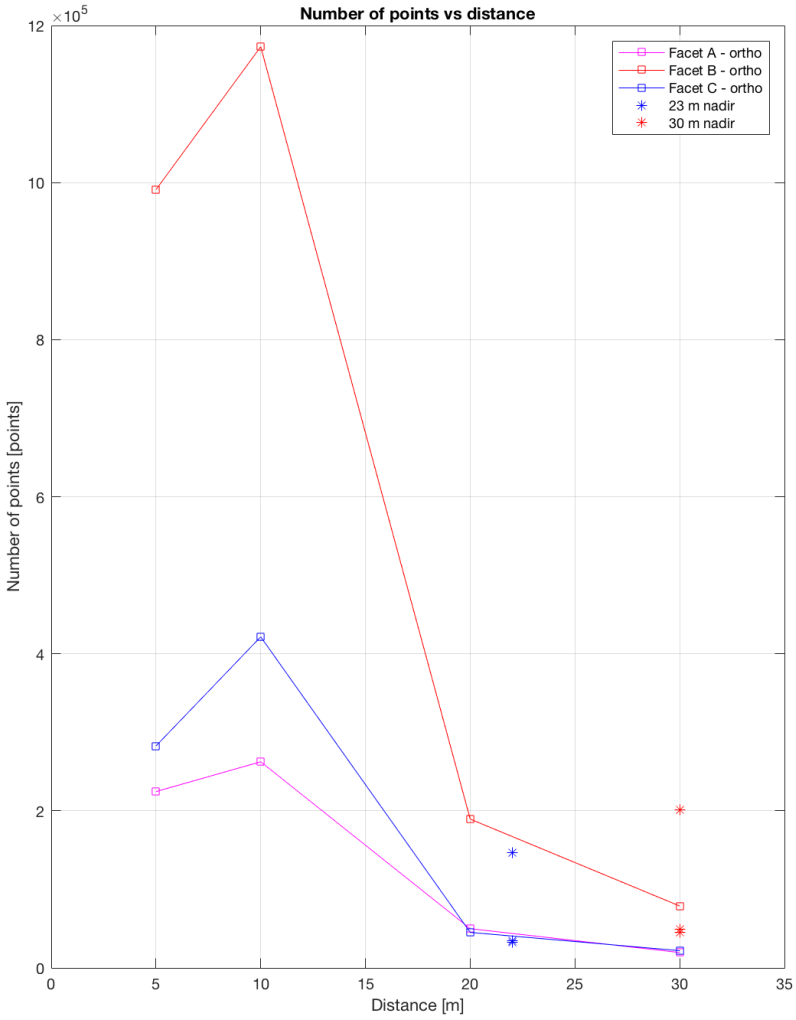

Figure 3.36: Plot showing the number of points in each roof face for orthogonal and nadir imagery.

are assigned to a plane. These standard deviations of off-plane distributions are shown in Fig 3.37.

Fig 3.37 shows a clear relationship between distance from the roof face and the off-plane deviation. This relationship appears linear, but lacks sufficient data points to confirm conclusively. If the linearity can be proven in future iterations, then it is likely due to the lower RSD.

Effect of ground control points

As explored previously, variations in elevation can be corrected based on accessible elevation data. Finding reliable easting and northing data to reference is more difficult than elevation data. While one could resort to using data from Google Earth or similar online mapping application, the level of detail can change by location. Instead, using a high accuracy GPS to determine the position of features surrounding the building can generate the necessary translation. Several distinctive points surrounding the Garden Restaurant are labeled A through G, as previously shown in Fig 3.23. Fig 3.38 shows the effect of increasing the number of GCPs on horizontal, vertical, and total difference measurements of GRPs A, C and E. The GRPs used as 3D GCPs are listed in Table 3.6. For each iteration, the number of GCPs was increased and the effect on two GRPs' positions were noted. As expected, the most significant effect was altitude, which validates the need to offset measurements from the initial dense point cloud.

Table 3.6: GCPs used with increasing GCPs

# of GCPs	GCPs used
0	-
1	B
2	BD
3	BDF
4	BDFG

Based on this very limited dataset, using two or three 3D GCPs has the potential to reduce horizontal position error to less than 20 cm, which is more than adequate for a roof inspection flight plan. A single GCP may help or or hinder horizontal position error, but will likely improve elevation results. Ideally, roof vertex locations would have been measured using an accurate GPS receiver, but this would have required significant ladder work, and there may have been additional error induced by GPS signal reflections from roof faces and trees.

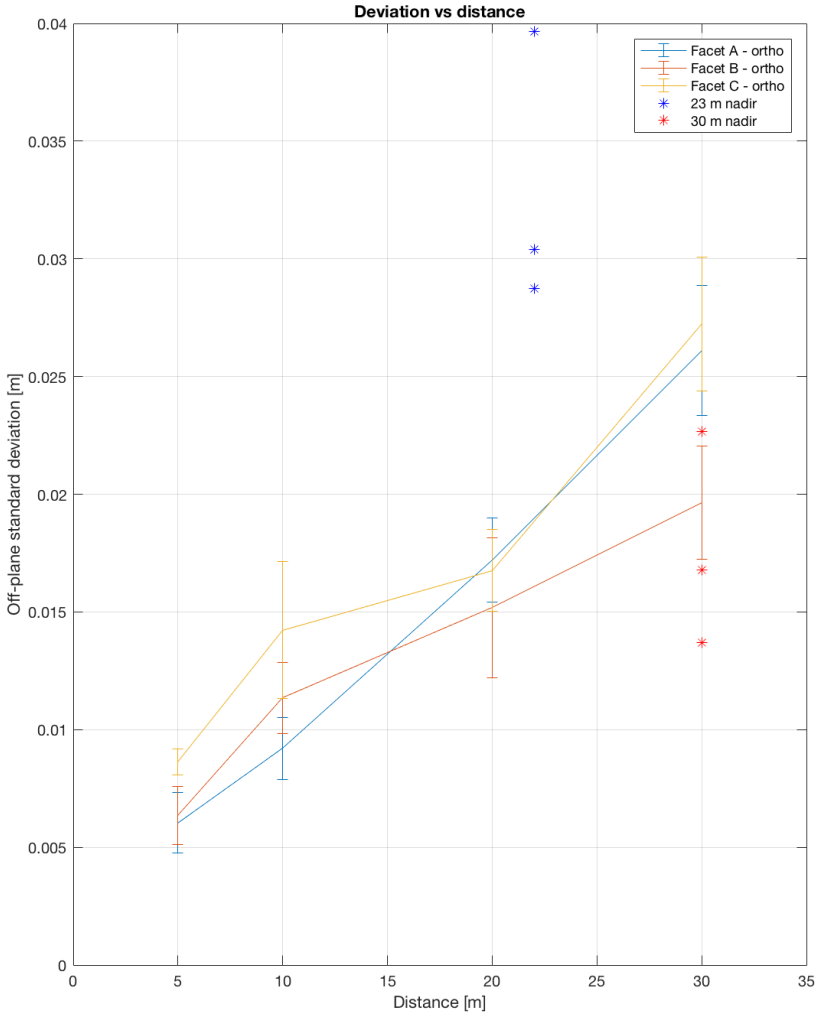

Figure 3.37: Plot showing the point cloud deviation from the roof face.

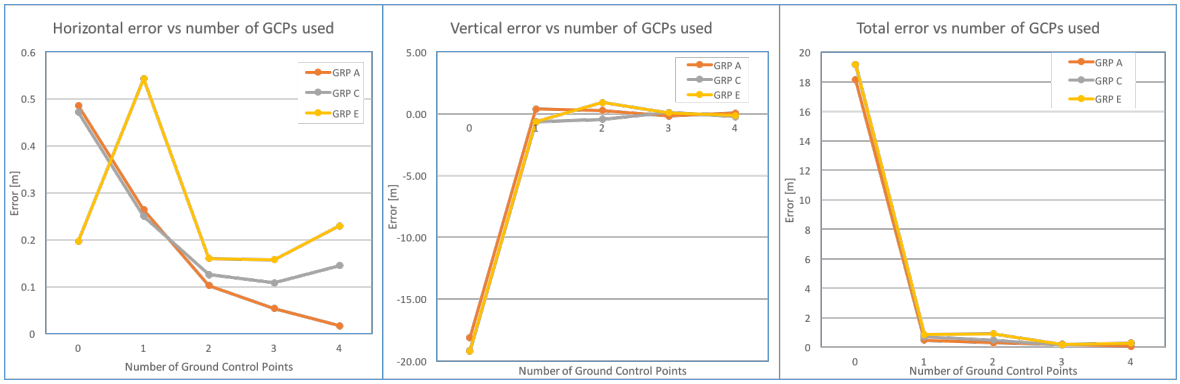

Figure 3.38: Horizontal (left), vertical (center), and total (right) difference between point cloud location and measurements made with the Trimble GPS after differential post processing.

Mesh examples

While the emphasis of this work has been the analysis of point clouds, it is important to investigate the effect of these point clouds on the resulting meshes generated by the Pix4D application.

Figs 3.39 and 3.40 show examples of meshes of the Garden Restaurant's south facade, generated at several distances. These meshes show variation in the appearance of repetitive patterns of different sizes. Visual evaluation of both figures shows that, for the Matrice-100's camera configuration, the best rendering was 5-10 m from the roof plane.

3.7 Conclusions

The key outcome of this work is that flight plans can be generated from point cloud data and safely executed, as long as a vertical correction is applied to point clouds generated using uncalibrated sUASs. Failure to do so could cause the desired target to be out of frame, or cause a crash, when conducting flights based on the geolocation of the initial dense point cloud generated from nadir imagery. Horizontal corrections may also be required if the aircraft gets too close to the target, especially as time passes following the nadir flight. Regardless, the remote pilot must be ready to take over controls when flying close to the roof, especially at distances of less than five meters. Horizontal and vertical corrections can be performed using 3D GCPs if suitable reference points and measuring equipment are available. A single GCP can provide the necessary accuracy necessary to

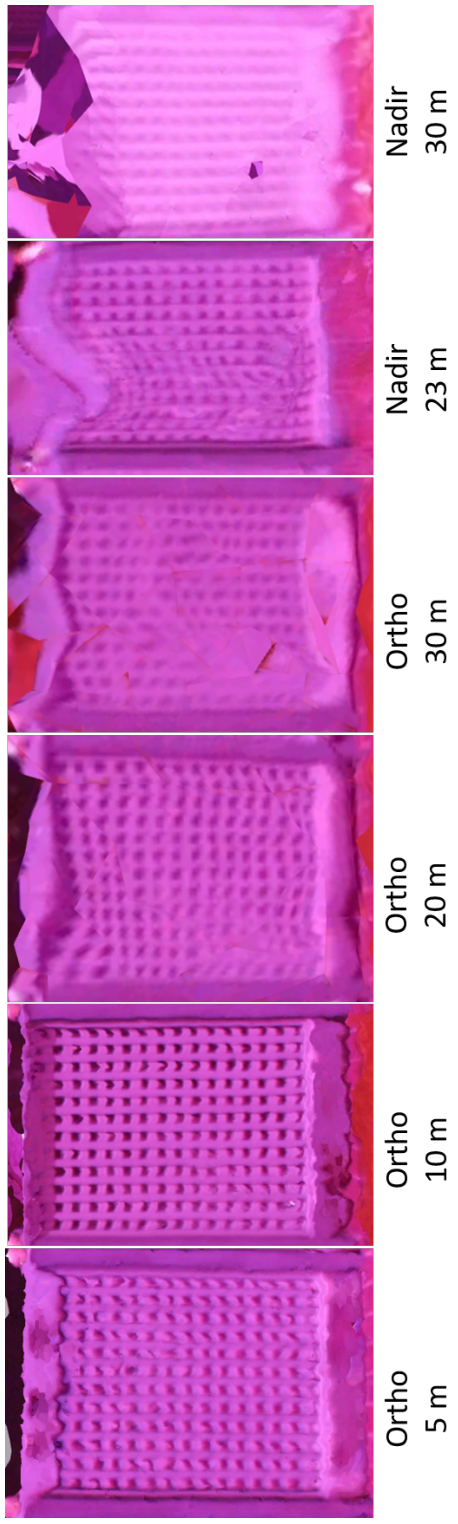

Figure 3.39: Meshes of the Garden Restaurant's lattice work taken from 5, 10, 20, and 30 m orthogonal from the roofs surface, and from 23 and 30 m AGL looking at nadir.

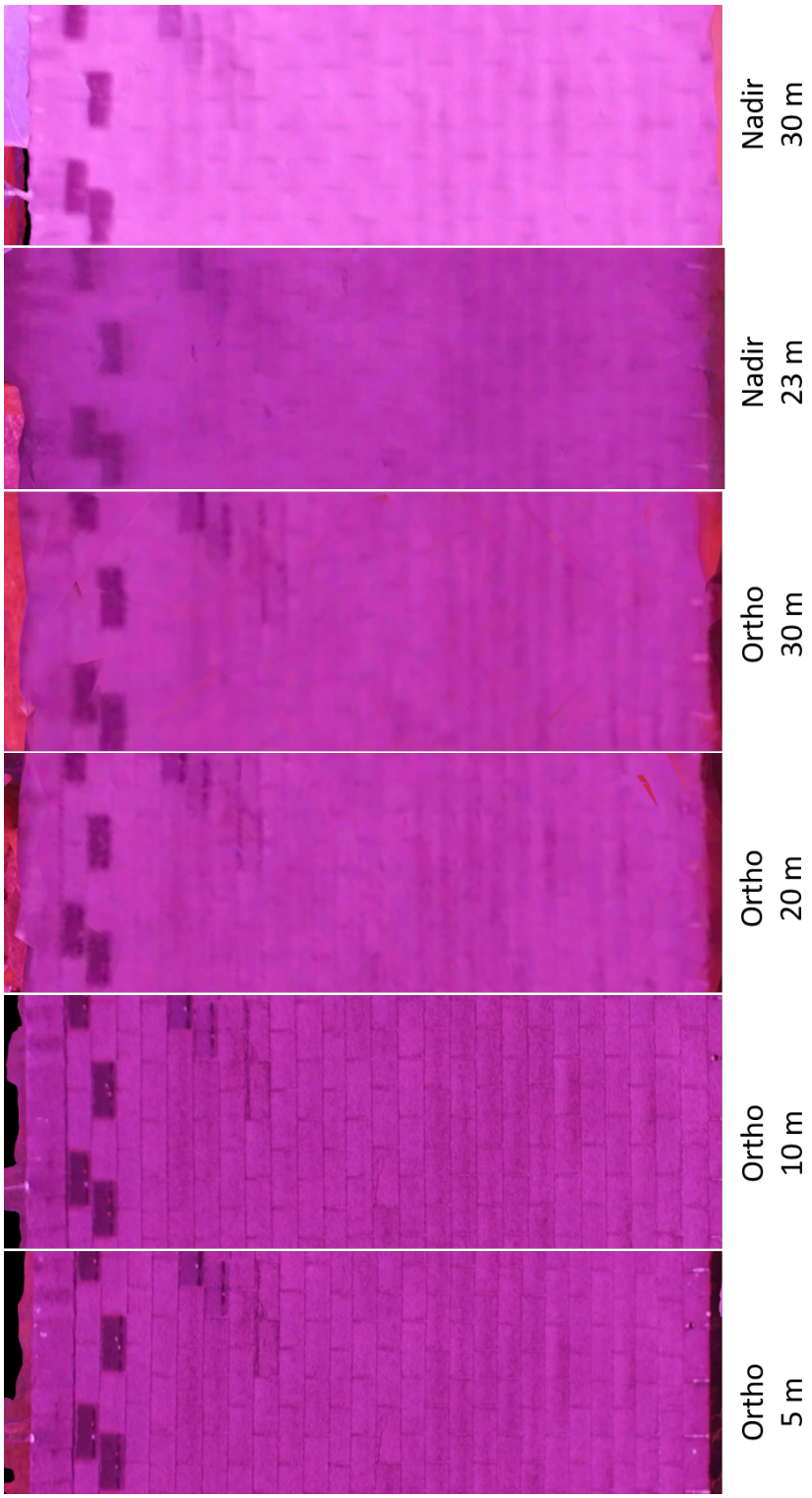

Figure 3.40: Meshes of the Garden Restaurant's shingles taken from 5, 10, 20, and 30 m orthogonal from the roof's surface, and from 23 and 30 m AGL looking at nadir.

produce flight plans from an initial point cloud, but more GCPs are better.

The results of these experiments did not contradict the stated hypotheses, but additional data collection should be conducted before confirming them as factual. In particular, the effect of distance from the roof face on slope requires special attention due to the distribution of points.

3.8 Future work

During the data collections, conditions were constantly changing over the various flights, resulting in position error due to wind gusts and variable lighting conditions. In order to determine what distance generates the best point clouds, these experiments should be run many times for many different distances, and they should be assessed by several independent observers.

There are several areas that could be explored to either improve or expand upon this work:

3.8.1 Speed optimization

The customized orthogonal flight plans were limited in this experiment to less than 2.0 m/s for safety considerations. Faster speeds could easily be implemented, but image blur could become an issue, especially during low lighting conditions, and the time required to save an image may exceed the interval between image captures. When flying at different distances, it may be convenient to ensure that the blur distance is consistent across all flights. Since most of the flight parameters are known, providing a recommended speed to fly based on distance from the roof, focal length, and acceptable blur length would be relatively easy to implement. Attention must also be paid to the camera's frame rate to ensure that it can gather still photographs. If the camera lacks the necessary buffer, then desired image locations may be missed if the speed is too high or the capture locations are too close.

The speed of the aircraft and a vibrating sensor can cause blur. This can be measured by flying a sUAS parallel to a textured wall and capturing several images as the aircraft moves from left to right while imaging the wall. A tribar target fixed to the wall will provide information on how much edge spread occurs. A combination of different ground speeds and exposure times should be used in consistent conditions. This experiment should be conducted in a laboratory where illumination settings can be controlled. Experimentation can also be conducted outdoors, but care must be taken to choose a day with consistent illumination conditions. The variability of lighting conditions may require the experiment to repeated several times. Another variable that should be assessed is the effect of focal

length. A longer focal length will result in more blur at a given distance, but provides a smaller sampling distance. To judge the effect of jitter on image quality, imagery should be captured in a hover at multiple distances from the target to determine the amount of blur in each frame. This should be repeated at multiple focal lengths as well.

3.8.2 GPS system upgrade

One issue encountered in this work was the accuracy of the GPS system. Improved guidance systems can provide better accuracy using a digital real-time kinetic (D-RTK) system that claims sub-centimeter accuracy. This would improve point clouds by providing a better initial estimation of the camera locations, while reducing the chance of collision with surrounding objects.

3.8.3 Collision avoidance

As part of building extraction work performed by Sun [58], tree extraction was conducted. By knowing the position of trees within a scene, a safety buffer can be applied to ensure that the aircraft remains at a specified distance from the obstacle. This technique could also be applied for the location of utility wires attached to a building. For complex roof structures, a roof face may be close to other roof faces. Attempting to position the aircraft on a plane offset to the one roof face may cause a collision with another roof face. A safety buffer around the building may also prevent collisions.

3.8.4 Onboard decision making

DJI provides users with an onboard software development kit (SDK). This SDK permits the sUAS to autonomously maneuver in real time based on the changing environment. A guidance package is available for the Matrice-100 and Matrice-600 models. While the typical use for the guidance system is to avoid collisions with objects in the flight path, data from these cameras could also be used to measure the aircraft's distance from objects, such as a roof, to ensure that the imagery being collected is at a constant offset and orientation.

Additional on-board applications could also aid the measurement of surface reflectance. Surface reflectance is typically measured using the Emprical Line Method (ELM) [53]. This method involves the imaging of at least two calibration panels with known reflectance values. Since sky conditions change rapidly throughout a given day, additional images of calibration panels need to be collected. An upward looking sensor could be integrated into the aircraft to monitor changes in illumination, triggering the aircraft to move to the calibration panels and take a picture.

Coaxial laser altimeter

A gimbaled camera with a co-axial laser range finder could be employed to ensure that the aircraft maintains a constant distance from the roof plane. Successful implementation of such a system could be incorporated into the onboard decision making system to ensure that the aircraft is at a specified distance away from the target when capturing an image. The anticipated result would have made for a better comparison when assessing the quality of point clouds generated at different distances from the roof.

Harris corner detector

Thus far, this work has focused on a collection flight, off-aircraft processing, and then uploading a new flight plan to the aircraft. Rather than using commercial applications to generate the initial model of the building, a less robust method could be implemented to reduce the computational expense and time. The Harris corner detector [27] may prove ideal for detecting corners of roofs, vinyl siding, shingles, and bricks. The result would be a coarse point cloud that would avoid many additional calculations. This would permit the aircraft to detect planar surfaces and then image them based on a few user-specified parameters. A trade-off study would have to be conducted to determine if it is more energy efficient to transmit the imagery to the ground to a personal computer for processing, or to process onboard the aircraft. Processing on board the aircraft will consume power, and shorten the aircraft's endurance.

3.8.5 Roof sag

One potential problem with the face-based approach is the underlying assumption that a building's roof consists of flat planes. While no object is perfectly flat, the techniques described are robust enough to deal with some imperfections. The algorithm would likely fail if the structural integrity of the roof was severely compromised, due to significant roof sag. The face-based could still be used by selecting additional points along a sagging roof to serve as the vertices. Images from the flight could be used to create 3D models to see if the roof sag affects the quality of the point cloud.

3.8.6 Circumferential approach

The implemented solution consisted of a face-by-face approach. The resulting flight path involves the aircraft performing multiple climbs to clear the building. Flying around the entire building, while adjusting the lookdown angle to keep the FPA parallel with the roof's surface, eliminates the need for multiple climbs and descents. Coordinates for the aircraft could be generated from the digital elevation model generated by Pix 4D and can

be displayed as contour lines. This circumferential approach would reduce the number of climbs for the aircraft and extend battery life, since climbing is the least efficient segment of any flight [46]. This circumferential approach would be less viable for a roof with only two faces, since there would be two sides of the building with no walls and would be non-imaging legs.

3.8.7 Increase size of parallel planes

The planes where images were taken were made to be the same size as the corresponding roof face. Increasing the size of this imaging plane would produce more photos of each roof face from more angles. This might improve the results and should be investigated. Care needs to be taken to ensure that the aircraft stays above a certain safety altitude to prevent collision with the ground or other obstacles.

3.8.8 Impact of scene and spectral band on tie point generation

It is worth investigating the use of different spectral bands to find additional tie points. Different spectral bands might result in more, or fewer features being detected. The images used to generate dense point cloud data were all taken with a silicon-based sensor using three bands and using a Bayer filter pattern. A multispectral sensor could be equipped with filters to optimize feature detections on different types of surfaces. This poses the problem that exposure time will be higher than an unfiltered sensor which would cause additional blur. Regardless, ingesting individual bands of the multi-spectral images into Pix4D should determine if the point cloud is affected by the choice of which band is used.

3.8.9 Simulated images

Rather than using a sUAS system that is subject to position error, images could be generated using a simulated scene. Simulated cameras could be precisely positioned and oriented to generate imagery from evenly spaced points, perpendicular to the roof face. The resulting images could be fed into a photogrammetric software's workflow to generate a point cloud. These simulated images would remove error created by the constantly changing illumination, camera limitations such as time between exposures and pointing accuracy, and GPS error.

3.8.10 Improved camera

Ground obstacles and roof surfaces were potential collision hazards throughout the experiment. Increasing the distance between the aircraft and the roof would provide remote pilots with an additional sense of safety. This can be accomplished by using a camera with

a longer focal length. This could be accomplished using commercial-off-the-shelf sensors, however, longer focal lengths are also susceptible to additional blur. A trade-off study to compare the advantages of a longer focal length against the additional blur would be useful. During these experiments, several frames were missed because the system was too busy to record the frame. While not confirmed, I believe this was due to the time required to save the image to the memory card. If this is the case, a suitable memory buffer will resolve this issue.

3.8.11 Accurate roof vertex locations

Much of this work required improvisation to compensate for the fact that only a few roof measurements were recorded. Accurate GPS readings of vertex locations should be gathered for a better comparison. A boom truck or cherry picker and operator should be hired to assist in these measurements.

3.8.12 Use of raw imagery

The effect of using raw images should be assessed. Several sUAS cameras have the capability of recording images in raw (.dng) format or save directly into a JPEG format. JPEG images are compressed, causing blocking artifacts to occur, which may impact on feature extraction. An example of an uncompressed image and a JPEG image are compared in Fig 3.41. The images are of a shingle test sample within a laboratory. The images were captured using a feature of the camera that saves all frames as a raw and JPEG file. The raw image on the left clearly shows more detail which should lead to more features be detected. To further assess the impact of compression on the rendering of 3D point clouds, a flight could be flown over a target capturing several photos in both formats simultaneously. The raw images could also be converted into JPEG images with varying levels of compression. Each set of images could be ingested into a SfM application to determine the affects on the resulting point cloud for each level of compression. An analysis could also be done by running feature detection algorithms on each of the different levels of compression to determine the effect of compression on the number of features. The additional features should result in more features being detected.

Figure 3.41: Two consecutive pictures were taken from the Phantom 3-4K to show the effects of compression: Left - Uncompressed image. Right - Compressed JPG.

Chapter 4

Final remarks

Over the past year, my goal has been to bring additional capabilities to RIT's sUAS lab. The most significant contribution has been the development of scripts to convert tabled coordinates into a format that can be read by ground control software to automatically conduct imaging tasks. Specifically, the incorporation of gimbal controls provides users the ability to create flight plans for a multitude of tasks that might be required by imaging scientists. While the flight patterns that I have developed address current needs, the true utility of the script enables users to customize it, rapidly producing their own flight plans.

While the work I have performed shows the advantages of using a watertight mesh to inspect roof surfaces, the same mesh could have allowed someone else to approach the problem in other ways. Regardless of their approach, the automated script will save them time in converting their coordinates into a format that is readable by UgCS. Neither the script, nor the unlicensed version of UgCS, have any cost associated with them, allowing for an economical implementation. However, for many applications, the licensed version of UgCS will be required.

Moving forward, I hope that the products that I have created will permit RIT's UAS Laboratory to exploit thermographic techniques. This will involve the mounting of a high-end thermal camera to a sUAS and then employing various thermographic techniques to this imagery. Appendix A contains a summary of my initial background research into thermography. This work was undertaken with the intent of eventually applying it to aerial inspection of pipeline infrastructure, but these techniques have the potential to provide subsurface visualization through a variety of materials [34, 23].

Appendix A

Thermography

A.1 Introduction

Thermography is a method of non-destructive testing and evaluation of changes in temperature across a surface using a thermal camera [29]. A variety of techniques can be used to determine the thickness of a target based on how the apparent temperature changes with time. As long- and mid-wave thermal cameras continue to become more capable, their potential to be paired with a UAV for inspection purposes increases. The initial steps to apply thermographic techniques to pipeline inspection are investigated in this chapter with the hope that a sUAS-based thermal camera will enable thermographic measurements to be collected.

A.2 Background

In order to fully understand how thermography relates to the thickness of a pipe, one must first look at basic radiometric and thermodynamic theories. The thermal camera measures irradiance at the focal plane array (FPA) for a given bandwidth. Each element of the FPA is heated or cooled by the scene radiance, resulting in a change in resistance, and a digital number being applied to the corresponding pixel [61]. Within the thermal camera's frame, if there is a blackbody, then the sensor will be subjected to blackbody radiation according to Planck's Equation for spectral exitance, Eq A.1 [50, pp. 72-75]. An increase in temperature of the blackbody will result in more energy being emitted. Photon's reaching the sensor within its bandwidth will cause an increase in temperature of the sensor, which causes the resistance of the sensor to increase. This increase of resistance is then converted into a digital count [59].

$$M_\lambda(T) = \frac{2\pi hc^2}{\lambda^5(e^{\frac{hc}{\lambda kT}} - 1)} \tag{A.1}$$

- M_λ (T) is the spectral exitance $[\frac{W}{m^2 \mu m}]$
- h is Planck's constant $[6.626X10^{-34}Js]$
- c is the speed of light $[2.998X10^8 m/s]$
- λ is the wavelength $[m]$
- k is Boltzman's gas constant $[1.38X10^{-23}\frac{J}{K}]$
- T is the temperature $[K]$

While most objects are not true blackbodies, an increase in temperature will still result in an increase in exitance (i.e. energy leaving the object) based on the material's emissivity. This relationship is defined by Eq A.2.

$$\epsilon_\lambda = \frac{M_{\lambda-e}}{M_{\lambda-bb}} \tag{A.2}$$

- ϵ_λ is the emissivity [-]
- $M_{\lambda-e}$ is the spectral exitance emitted from an object $[\frac{W}{m^2 \mu m}]$
- $M_{\lambda-bb}$ is the spectral exitance emitted from a blackbody at the same temperature $[\frac{W}{m^2 \mu m}]$

Conduction, convection, and radiation all work to transfer heat within a pipe or similar object [40, pp. 46-53]. Heat is conducted within the pipe's structure from hot to cold regions. Heat is transferred to the pipe from radiant sources such as the sun, or a local heat source. The spectral absorption of the pipe dictates how much energy is absorbed by the pipe at each wavelength. Internal and external fluids cause air particles to change temperature and density, causing air to move and be replaced with new particles, thus transferring energy via convection. The rate of heat transfer increases in instances where the fluid is forced through or around a pipe [12]. This forced flow is relevant in both laboratory and real-world environments. In the laboratory environment, a building's ventilation will move some air over the external surface of the pipe, inducing heat transfer to or from the pipe. A pump or fan can be used to force a fluid over the external or internal surface of a pipe. Outdoors, wind acts to force airflow over a pipe's exterior surface, increasing the rate of heat transfer. The contents of a pipeline are also forced by way of a pump. If the pipeline is not running at full capacity, then the flow rate will vary based on demand, affecting the rate of heat transfer [12].

Thermography can be broken down into passive, and active thermography [39]. Passive thermography analyzes imagery based on normal operating conditions. Active thermography is typically conducted within a lab setting, using a heat source (such as a laser) to heat the surface of a target. Since the heat stored within a volume of space is linearly proportional to its mass, as indicated in Eq (A.3), a thinner area has less mass and cools faster than thicker areas, as there is less excess heat to be transferred from or to the surrounding air. Thermography can be used to determine in-depth and in-plane diffusion [39].

$$Q = mC\Delta T \tag{A.3}$$

- Q is heat [J]
- m is mass [kg]
- C is specific heat capacity of the material. [J/kg°C]
- ΔT is change in temperature [°C]

Thermography has proven successful in measuring pitting corrosion of a ductile iron pipe in a laboratory setting [34], and detecting flaws in composite wind turbine blades [23] using a sUAS-based thermal camera. Combining these two concepts, a thermographic payload has the potential to be integrated onto a sUAS to measure thinning in a pipeline.

Active thermography consists of several methods, but only a few are of significance for this thesis. A constant heat source can be used to activate the flow of heat from the part, so temperature change can be observed. Conversely, chilling a specimen and observing how heat is absorbed can provide information as to the thickness of the specimen. Pulse thermography (PT) employs a single or multiple pulses of energy that causes a slight rise in the surface temperature of the part [37]. Thinner portions of the target will result in more rapid dissipation of the heat. The temperature following a pulse is defined in Eq (A.4). A chopper wheel or flash strobes can also be used to generate multiple pulses.

$$\Delta T = \frac{Q}{e\sqrt{\pi t}} = \frac{Q}{\sqrt{k\rho C\pi t}} \tag{A.4}$$

- ΔT is change in temperature [°C]
- Q is heat [J]
- e is thermal effusivity $[\frac{Ws^{1/2}}{^{o}Cm^{2}}]$
- t is time [s]
- k is the thermal conductivity $[\frac{W}{^{o}Cm}]$
- ρ is the density $[kg/m^3]$
- C is the specific heat capacity of the material. [J/kg°C]

A.2.1 Image processing

Two relatively successful methods for measuring pitting corrosion in ductile iron pipes are principal component thermography (PCT) and pulsed thermography (PT) [34]:

Principal component thermography

The (PCT) process [44] is similar to principal component analysis (PCA) used to analyze hyperspectral imagery (HSI) [21]. Where PCA conducted on HSI requires building a hypercube from many spectral bands, PCT uses frames from a video to produce temporal bands. Once the hypercube is built, the image is processed as follows:

- Reshape the hypercube into a matrix, M, where each frame from the video is reshaped into a column vector. This is repeated until all desired frames are used. The resulting matrix consists of n rows, where n corresponds to the number of pixels in the array, and t columns, where t corresponds the number of frames selected.
- Singular value decomposition (SVD) generates the eigenvectors of M in a 'n x t' matrix, U. Each eigenvector in U represents a principal component (PC) of the selected frames.
- Reshaping each PC (i.e. column vector) of U into the original image dimensions will reform an image for analysis. The first PC corresponds to the eigenvector with the highest eigenvalue. Each image can be normalized for a selected region of the image to highlight defects.

Pulse thermography

For each pixel, the camera's response is recorded for each frame. A Fast Fourier Transform (FFT) [14] is then applied to the decaying response following a pulse from a lamp. Since the result of the FFT is complex, the real and complex values of each pixel can be calculated using Eq A.5 [34]. From this, the phase of the response can be found using Eq A.6 and the magnitude can be found using Eq A.7. [36].

$$F[u] = \frac{1}{N}\sum_{N-1}^{n=0} f[t]e^{\frac{-2\pi uti}{N}} = R[u] + I[u]i \tag{A.5}$$

$$\Phi[u] = tan^{-}1(\frac{I[u]}{R[u])}) \tag{A.6}$$

$$M[u] = \sqrt{R[u]^2 + I[u]^2} \tag{A.7}$$

- $F[u]$ is the complex discretized function in frequency space [Hz]
- N is the number of samples [-]
- $f[t]$ is the discretized function in time space [K]
- u is the frequency [Hz]
- t is time [s]
- i is the imaginary unit [-]
- $R[u]$ is the real portion of F[u] [Hz]
- $I[u]$ is the imaginary part of F[u] [Hz]
- $\Phi[u]$ is the phase of F[u] [radians]

Images can be generated for each phase and magnitude value [34], but the best depth measurements can be achieved by finding the maximum phase value for each pixel and creating an image based on these values [36].

A.3 Hypothesis and objectives

- Hypothesis. sUAS-based active thermography can be used to detect anomalies in gas pipeline thicknesses that could represent a safety hazard.
- Objective. Design and execute laboratory experiments that will assess suitability of active thermography to detect defects in gas pipelines and at what distance the aircraft must be from the pipeline to detect defects.

A.4 Methods

A variety of techniques have been used to process thermal imagery. Since the materials being investigated are not yet known, it is prudent to assess thermographic techniques by attempting to reproduce thickness measurements of known defects in a laboratory environment, before attempting this with aerial imagery.

A.4.1 Hardware

While there are regulatory issues associated with operating lasers from sUASs, a single longwave infrared (LWIR) source or an array of LWIR sources, in conjunction with a chopper, may be a suitable substitute. An Electro Optical Components Inc. EK-3431, shown in Fig A.1, was selected as an initial IR source due to its compact size, a relatively high output (12 W) amongst similar devices, and the presence of a reflector and window that provides some columnation while protecting the element. If this source proves sufficient to heat a target specimen at a distance of 5 m, then it could easily be integrated onto a

Figure A.1: EK-3431 IR Source with sapphire window

Figure A.2: Liteye Aquila Micro 320

sUAS as an active source. The use of a shutter, or a chopper wheel, can provide the effect of providing multiple pulses. If this source is insufficient, then an array of multiple sources could be used to boost the total output.

An active thermography system mounted to a sUAS may require excessive power. A tether may be required to supply enough power to the aircraft to meet power requirements without compromising the aircraft's endurance. However, a tether has the potential to reduce stability and mobility of the aircraft [39]. While this work focuses on detecting defects in pipelines, sUASs have previously implemented thermography techniques to successfully detect defects in wind turbine blades [23].

Table A.1: Liteye Micro Aquila 320 specifications

Pixel count	320 X 240
Pixel size	17 μm
Frame rate	60 fps
Format	NTSC/PAL
Sensor type	uncooled vanadium oxide microbolometer
Mass	95 g
Digital Zoom	1x, 2x, or 4x
NEΔT	50 mK

The primary sensor being used for this work was the Liteye Aquila Micro 320, illustrated in Fig A.2. This system is a modification of a DRS Technologies' Tamarisk 320 camera. The modifications were made by Liteye to permit camera settings to be adjusted by using pulse width modulation, used in many recreational radio controllers. Table A.1 contains the camera specifications provided by the manufacturer. A Tamarisk 640 was also used to conduct some experiments. This system has the same specifications as the Aquila Micro 320, except that each dimension of the FPA is doubled. The Tamarisk 640 can also provide raw data, which is a significant advantage over the Aquila Micro 320.

The Aquila micro 320's camera settings were controlled by using pulse width modulation (PWM) to three leads, marked as channels 7, 8, and 9. A 1.0 ms pulse width and a period of 16 ms, shown in Fig A.3, provided by a Futaba transmitter, engages 'manual' mode. Alternatively, a Raspberry Pi was also used as an optional method to generate the signal, providing the operator with the flexibility to use either a radio transmitter or the Raspberry Pi to modify camera settings. Once in manual mode, a serial connection is made between a computer and the camera using DRS Technologies' Camera Control Software [1]. This enables the software to adjust the gain and bias of the camera, switch polarities from white-is-hot to black-is-hot, and change between zoom settings. When disconnected from the Raspberry Pi, the camera stays in the last mode selected. There was a problem noted with the camera, where slight movement of the leads caused changes to the zoom, polarity, and gain control modes. While conducting experiments, all three leads should be plugged into the receiver to reduce the chance of accidentally switching modes.

Raw data were captured from the camera using an Elgato Video Capture device. This device takes the NTSC signal from the camera and upscales the image to 480 X 640 at a frame rate of 29.97 frames per second, and reduces the signal to 8-bits. Neither the dimensions of the frame, nor the frame rate match the specifications of the camera; this is expected to affect the relative edge response of the camera due to an unknown interpolation process applied to the video signal when the Elgato software saves videos in '.mov' format. No information is available on how the Elgato device processes data before saving. Raw data from the camera may be subject to a variety of processing techniques, such as compression and contrast stretching. While these issues will impact the image read into MATLAB, no corrections have been made to the frames to account for this, since the ultimate objective is to determine the feasibility of a sUAS-based thermographic imaging. Once captured, individual frames are extracted from the video for analysis.

A.4.2 Thermal targets

Test samples with known defects have been created. Ideally, these targets should be made of similar materials and dimensions as gas pipelines. Since these data have not been

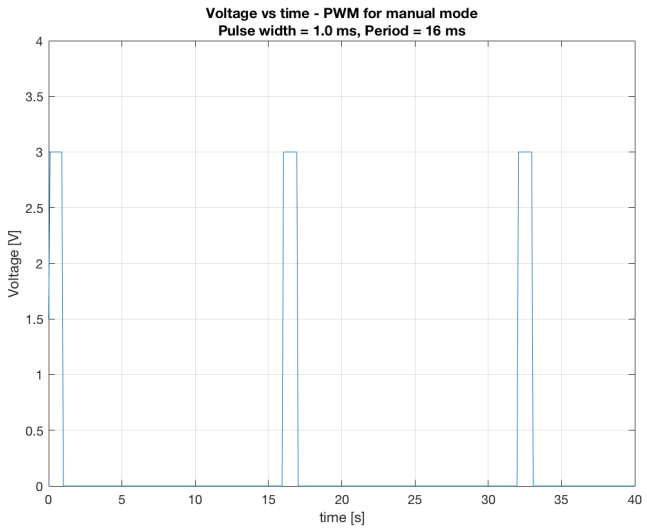

Figure A.3: PWM required to enable manual modes on the microbolometer

provided, initial testing was conducted on a 3D-printed target, before acquiring a steel and aluminum target.

The planar thermal target is pictured in Fig A.4 and consists of four quadrants. The first quadrant consists of two grooves. One groove is narrow and deep, and the second is wider and shallower than the first. This quadrant is intended to determine at what point the removed material can be detected. The second quadrant consists of concentric squares of different widths and increasing depth. The intent of this quadrant is to determine the necessary camera gain required to detect defects of different depths. The third quadrant consists of several holes with two different diameters and at varying depths. The final quadrant was left blank so that it could be used either as a way to manipulate the target or as a reference area. This blank area could also be used in the future to provide an additional defect.

Fig A.5 shows the key dimensions of the steel and aluminum targets. Due to constraints of the machining process, the recesses and slots were milled with a hog-nose (i.e. rounded) bit. The result is that internal corners have a 3.2 mm round, whereas the 3D printed target has internal corners that are square. This difference in geometry will affect the transfer of heat through the target.

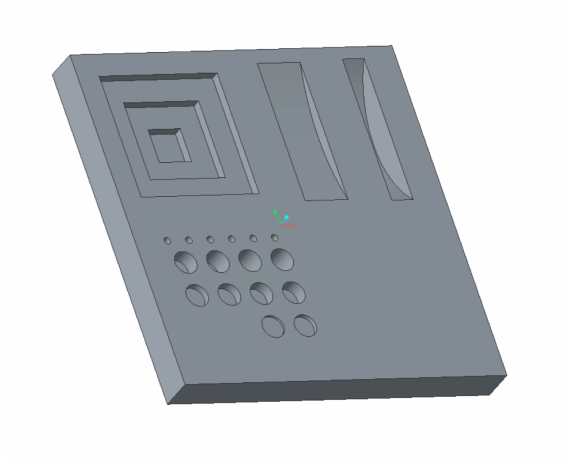

Figure A.4: Thermal target. 9 cm x 9 cm x 1 cm planar target used for initial testing.

A.5 Initial results

A planar thermal target was manufactured using 3D printing, but has an unintended defect in one corner. This defect should not significantly affect depth measurements due to the symmetry of that portion of the target. Initial imagery was captured of the flat side of the thermal target, while warming up after being removed from a freezer.

Pulsed phase thermography was applied by collecting thermal imagery after pulsing a photographic strobe light ten times to warm the surface of the target while a microbolometer recorded the process. PCT (left), magnitude (center), and phase (right) images of this process are shown in Fig A.6. The digital counts of two arbitrarily selected pixels over time are shown in Fig A.7. There appears to be multiple horizontal lines for each plot, but this is noise. Frames were gathered at a frame rate of 30 frames per second. On closer inspection of the plot, the digital count varies between plus or minus three digital counts from frame to frame. It is also worth noting that every seventh digital count is blank. The processes of how the video signal is converted from raw data from the camera to an analog output into the Elgato device and then back into a digital signal has not been determined, but this process could be responsible for the uniform gaps in the digital count. This is a moot point, since the availability of the Tamarisk 640 has enabled the analysis of raw data. The plot also shows the upper plot reaching a level plateau prior to decreasing. The thin nature of the plateau suggests that the sensor is saturated, indicating that ten pulses were excessive for the experimental configuration.

Initial work done with the EK-3431 IR source showed some potential in a laboratory

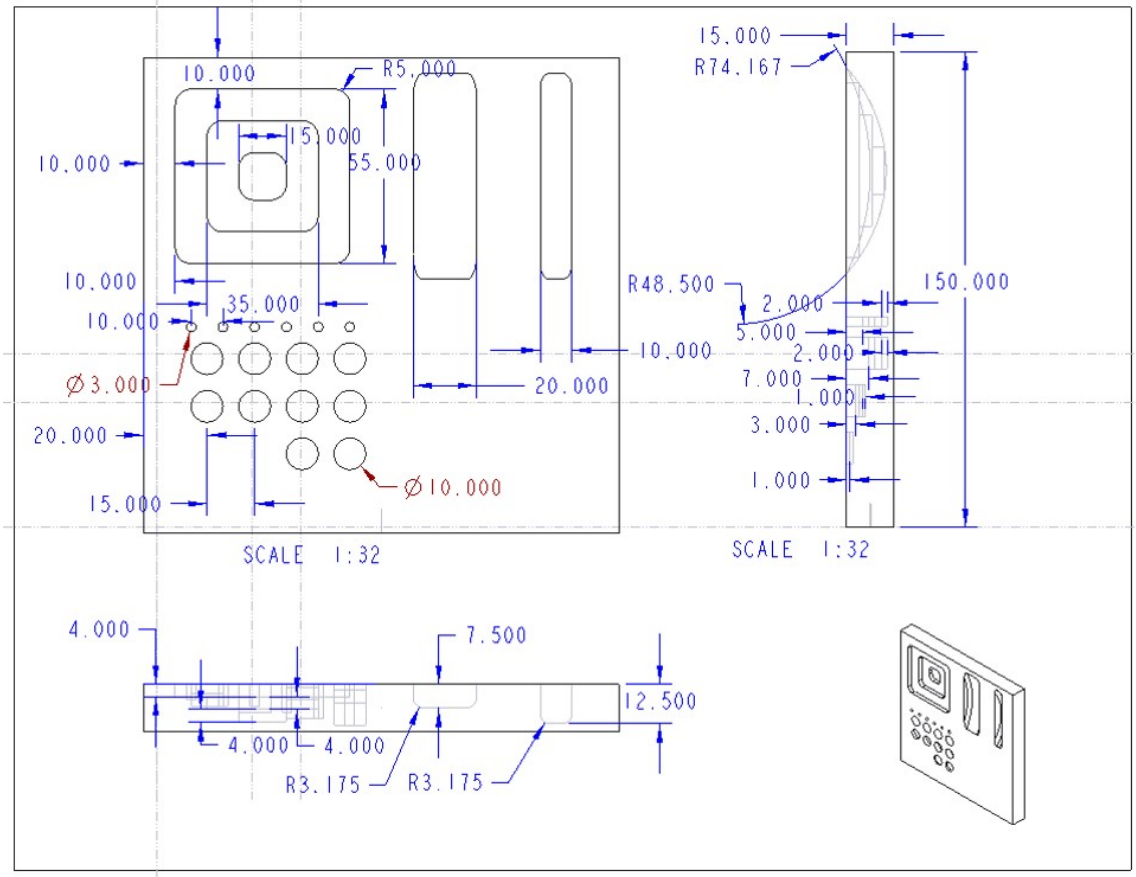

Figure A.5: Engineering drawing of the aluminum and steel thermal targets. All dimensions are in millimeters

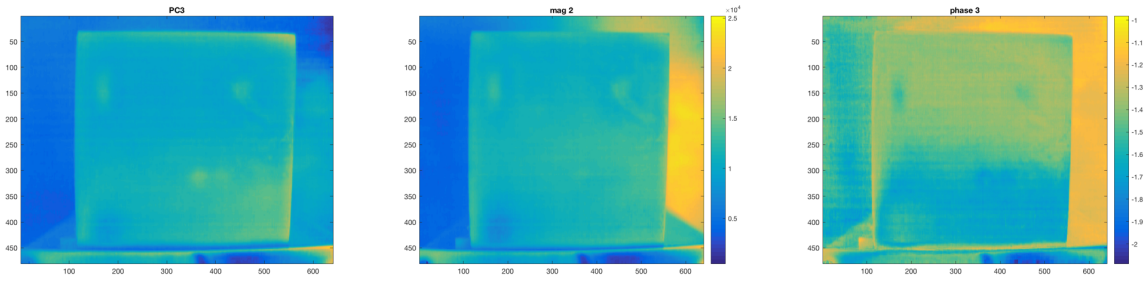

Figure A.6: Examples of PCT, magnitude, and phase imagery of the heating and cooling of a thermal target using a strobe light as the activation source.

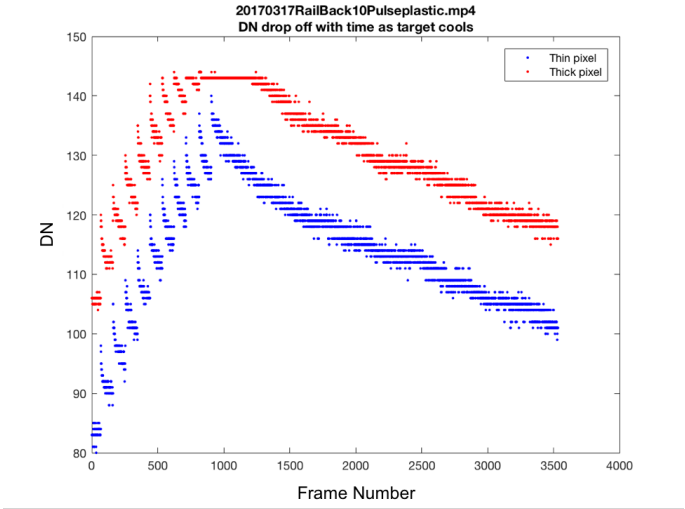

Figure A.7: The rise and fall of digital count over time.

environment, however, a single unit is not able to provide uniform illumination. Considering the uneven illumination and the power consumption required by multiple units, this IR source is not suitable as a source for thermal illumination aboard an airborne sensor.

A.6 Future work

It is yet to be proven whether sUAS-based thermography for pipeline inspection is practical or not, but previous work performed on wind turbines indicates this is worth further evaluation. Further experimentation is required to assess this. The following items should be explored in further detail:

- The noise present in the uncooled microbolometers resulted in significant error. With the acquisition of a cooled camera system, the initial laboratory work can be repeated, such that results from published work can be reproduced.
- Access to the Tamarisk 640 provides a means to analyze raw data, which was not possible with the Aquila Micro 320. Experiments using the Tamarisk 640's raw data should be assessed more closely for its potential use in thermography. Further experimentation should also be done using a cooled thermal camera.
- Each side of each thermal target should be imaged and assessed, since both sides of a pipeline are subject to damage.

- The aluminum target requires an anodic coating. The current state is nearly a perfectly reflective mirror. The anodic coating should remove this effect.
- The steel target has initial signs of oxidation. This should be polished to remove the rust and coated with oil to preserve it.
- Mounting a xenon flash system to a sUAS might be cumbersome and add too much weight to the aircraft to be effective. Alternate methods of activating heat transfer should be explored.
- So far, only magnitude and phase images have been generated. Maximum phase images were not produced. Methods to obtain depth measurements from each of these techniques require further exploration.

Bibliography

[1] Camera control software (1.9.3), http://www.drsinfrared.com/products/tamarisk640.aspx.

[2] ENVI (5.3), https://www.harris.com/what-we-do/geospatial-solutions.

[3] MATLAB (2016b), https://mathworks.com, 2016.

[4] UgCS (2.9), https://www.ugcs.com, 2016.

[5] Pix4Ddesktop (3.1.23), https://pix4d.com, 2017.

[6] UgCS (2.10), https://www.ugcs.com, 2017.

[7] S Adams, C Friedland, and M Levitan. Unmanned aerial vehicle data acquisition for damage assessment in hurricane events. In *Proceedings of the 8th International Workshop on Remote Sensing for Disaster Management, Tokyo, Japan*, volume 30, 2010.

[8] WS Alexander. The relationship of wind correction angle to drift angle. *Journal of the Aeronautical Sciences*, 8(11):409–412, 1941.

[9] Clint R Balog, Brent A Terwilliger, Dennis A Vincenzi, and David C Ison. Examining human factors challenges of sustainable small unmanned aircraft system (suas) operations. In *Advances in Human Factors in Robots and Unmanned Systems*, pages 61–73. Springer, 2017.

[10] L Barazzetti, F Remondino, M Scaioni, and R Brumana. Fully automatic uav image-based sensor orientation. In *Proc. ISPRS Commission I Mid-Term Symposium'Image Data Acquisition-Sensors & Platforms*, volume 12, 2010.

[11] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *Computer vision–ECCV 2006*, pages 404–417, 2006.

[12] Theodore L Bergman and Frank P Incropera. *Introduction to heat transfer*. John Wiley & Sons, 2011.

[13] Elizabeth Bondi, Carl Salvaggio, Matthew Montanaro, and Aaron D Gerace. Calibration of uas imagery inside and outside of shadows for improved vegetation index computation. In *SPIE Commercial+ Scientific Sensing and Imaging*, pages 98660J–98660J. International Society for Optics and Photonics, 2016.

[14] E Oran Brigham. The {F} ast {F} ourier {T} ransform and its applications. 1988.

[15] Andreas Burkart, Helge Aasen, Luis Alonso, Gunter Menz, Georg Bareth, and Uwe Rascher. Angular dependency of hyperspectral measurements over wheat characterized by a novel uav based goniometer. *Remote sensing*, 7(1):725–746, 2015.

[16] George Dallas. Principal component analysis 4 dummies: Eigenvectors, eigenvalues and dimension reduction. 2013.

[17] Jonathan P Dandois, Marc Olano, and Erle C Ellis. Optimal altitude, overlap, and weather conditions for computer vision uav estimates of forest structure. *Remote Sensing*, 7(10):13895–13920, 2015.

[18] DJI. Dji.com, 2002-2017.

[19] Fletcher Dunn and Ian Parberry. *3D math primer for graphics and game development*. CRC Press, 2015.

[20] Roger L Easton Jr. *Fourier methods in imaging*. John Wiley & Sons, 2010.

[21] Michael Theodore Eismann. Hyperspectral remote sensing. SPIE Bellingham, 2012.

[22] Mark A Fonstad, James T Dietrich, Brittany C Courville, Jennifer L Jensen, and Patrice E Carbonneau. Topographic structure from motion: a new development in photogrammetric measurement. *Earth Surface Processes and Landforms*, 38(4):421–430, 2013.

[23] C Galleguillos, A Zorrilla, A Jimenez, L Diaz, ÁL Montiano, M Barroso, A Viguria, and F Lasagni. Thermographic non-destructive inspection of wind turbine blades using unmanned aerial systems. *Plastics, Rubber and Composites*, 2015.

[24] S Gottwald. *The VNR concise encyclopedia of mathematics*. Springer Science & Business Media, 2012.

[25] E Grafarend. The optimal universal transverse mercator projection. In *Geodetic Theory Today*, pages 51–51. Springer, 1995.

[26] John W Gross. A comparison of orthomosaic software for use with ultra high resolution imagery of a wetland environment. *Center for Geographic Information Science and Geography Department, Central Michigan University, Mt. Pleasant, MI, USA. Available from: http://www. imagin. org/awards/sppc/2015/papers/john_gross_paper. pdf*, 2015.

[27] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Manchester, UK, 1988.

[28] Fangning He and Ayman Habib. Automated relative orientation of uav-based imagery in the presence of prior information for the flight trajectory. *Photogrammetric Engineering & Remote Sensing*, 82(11):879–891, 2016.

[29] Clemente Ibarra-Castanedo, Jean-Marc Piau, Stéphane Guilbert, Nicolas P Avdelidis, Marc Genest, Abdelhakim Bendada, and Xavier PV Maldague. Comparative study of active thermography techniques for the nondestructive evaluation of honeycomb structures. *Research in Nondestructive Evaluation*, 20(1):1–31, 2009.

[30] National Imagery and Mapping Agency. Department of defense world geodetic system 1984 - it's definition and relationships with local geodetic systems. 1984.

[31] Luo Juan and Oubong Gwun. A comparison of sift, pca-sift and surf. *International Journal of Image Processing (IJIP)*, 3(4):143–152, 2009.

[32] Jan J Koenderink and Andrea J Van Doorn. Affine structure from motion. *JOSA A*, 8(2):377–385, 1991.

[33] Wenkai Li, Qinghua Guo, Marek K Jakubowski, and Maggi Kelly. A new method for segmenting individual trees from the lidar point cloud. *Photogrammetric Engineering & Remote Sensing*, 78(1):75–84, 2012.

[34] Zheng Liu, Marc Genest, and Dennis Krys. Processing thermography images for pitting corrosion quantification on small diameter ductile iron pipe. *NDT & E International*, 47:105–115, 2012.

[35] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[36] Xavier Maldague and Sergio Marinetti. Pulse phase infrared thermography. *Journal of applied physics*, 79(5):2694–2698, 1996.

[37] Xavier PV Maldague. Introduction to ndt by active infrared thermography. *Materials Evaluation*, 60(9):1060–1073, 2002.

[38] Kanti V Mardia and Peter E Jupp. *Directional statistics*, volume 494. John Wiley & Sons, 2009.

[39] LE Mavromatidis, JL Dauvergne, R Saleri, and JC Batsale. First experiments for the diagnosis and thermophysical sampling using impulse ir thermography from unmanned aerial vehicle (uav). In *Qirt conference*, 2014.

[40] J Moran Michael and N Shapiro Howard. Fundamentals of engineering thermodynamics third edition. 1995.

[41] G Morgenthal and N Hallermann. Quality assessment of unmanned aerial vehicle (uav) based visual inspection of structures. *Advances in Structural Engineering*, 17(3):289–302, 2014.

[42] K Tysen Mueller, Peter VW Loomis, Rudolph M Kalafus, and Leonid Sheynblat. Networked differential gps system, June 21 1994. US Patent 5,323,322.

[43] Rafael Palacios. https://www.mathworks.com/matlabcentral/fileexchange/10914-utm2deg, accessed 10 july 2017, 2006.

[44] Nikolas Rajic. Principal component thermography for flaw contrast enhancement and flaw depth characterisation in composite structures. *Composite Structures*, 58(4):521–528, 2002.

[45] C Radhakrishna Rao and M Bhaskara Rao. *Matrix algebra and its applications to statistics and econometrics*. World Scientific, 1998.

[46] Daniel P Raymer. Aircraft design: A conceptual approach, american institute of aeronautics and astronautics. *Inc., Reston, VA*, page 21, 1999.

[47] Brent A Renfro, Josh King, Audric Terry, Jeff Kammerman, David Munton, and Johnathan York. An analysis of global positioning system (gps) standard positioning system (sps) performance for 2013. 2015.

[48] Tomi Rosnell and Eija Honkavaara. Point cloud generation from aerial image data acquired by a quadrocopter type micro unmanned aerial vehicle and a digital still camera. *Sensors*, 12(1):453–480, 2012.

[49] Carsten Rother. A new approach to vanishing point detection in architectural environments. *Image and Vision Computing*, 20(9):647–655, 2002.

[50] John R Schott. *Remote sensing: the image chain approach*. Oxford University Press on Demand, 2007.

[51] Stephen L Schultz and John Monaco. Unmanned aircraft structure evaluation system and method, April 4 2017. US Patent 9,612,598.

[52] Shishir Shah and JK Aggarwal. A simple calibration procedure for fish-eye (high distortion) lens camera. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 3422–3427. IEEE, 1994.

[53] Geoffrey M Smith and Edward J Milton. The use of the empirical line method to calibrate remotely sensed data to reflectance. *International Journal of remote sensing*, 20(13):2653–2662, 1999.

[54] John Parr Snyder. *Map projections–A working manual*, volume 1395. US Government Printing Office, 1987.

[55] Paul Sponagle and Carl Salvaggio. Automatic mission planning algorithms for aerial collection of imaging-specific tasks. In *SPIE Commercial+ Scientific Sensing and Imaging*, pages 1021807–1021807. International Society for Optics and Photonics, 2017.

[56] J Sullins, D Warren, and W Wiers. Pipeline inspection pig, January 22 1974.

[57] Shaohui Sun. Automatic 3d building detection and modeling from airborne lidar point clouds (doctoral dissertation), Dec 2013.

[58] Shaohui Sun and Carl Salvaggio. Aerial 3d building detection and modeling from airborne lidar point clouds. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(3):1440–1449, 2013.

[59] D Sabuncuoglu Tezcan, Selim Eminoglu, and Tayfun Akin. A low-cost uncooled infrared microbolometer detector in standard cmos technology. *IEEE Transactions on electron devices*, 50(2):494–502, 2003.

[60] Chris Veness. Calculate distance, bearing and more between latitude/longitude points, http://www.movable-type.co.uk/scripts/latlong.html, accessed 23 june 2017, 2002-2017.

[61] RA Wood. Monolithic silicon microbolometer arrays. *Uncooled Infrared Imaging Arrays and Systems*, 47:43, 1997.