3D RECONSTRUCTION OF LAKE SURFACE USING CAMERA AND LIDAR SENSOR FUSION

BY

SHAHRUKH KHAN

Submitted in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical and Aerospace Engineering
in the Graduate College of the
Illinois Institute of Technology

Seebany Datta-Barua

Digitally signed by Seebany Datta-Barua
DN: cn=Seebany Datta-Barua, o=Illinois Institute of Technology, ou, email=sdattaba@iit.edu, c=US
Date: 2020.07.17 12:30:34 -06'00'

Approved ______________________
Advisor

Chicago, Illinois
July 2020

ACKNOWLEDGMENT

Firstly, I would like to thank the universe for existing. If events did not play out exactly as they did, I may not have existed to be able to accomplish all this. I must also express my very profound gratitude to my parents, my siblings and family, for providing me with unfailing support through the ups and downs of life.

I would like to thank my thesis advisor Dr. Seebany Datta-Barua. She consistently steered me in the right the direction, without which I would have been woefully lost. I would also like to thank my thesis committee members who advised me on this research project: Dr. Boris S. Pervan and Dr. Matthew Spenko. Their input was paramount to the completion of this research work. I would like to express my gratitude to the IIT faculty and Staff, especially Ms. Jessica Nicholson, who were the cornerstone to everything I achieved here.

I am grateful to Kiran Kumar, Heli Trivedi, Tarang Vaidya, Prakarsh Gurmule, Omar Takadoum, Yuttana Jamjumrus, Xiang Yim, Vineeth Kannan, Aleksander Woody, Charlie Cao, Yihe Chen, Sankalp Sanand and all of the friends I made since coming to IIT. A very special gratitude goes out to David Stuart, Heejin Kim, Li Pan, Roohollah Parvizi and all of my Space Weather Lab colleagues. I am gratefully indebted to them for their help and support throughout my time at IIT.

I would like to extend a sincere thanks to my close friends Faran Ali Khan, Salman Waraich, Fahad Tariq Gill, Hassan Tahir Shah, Talha Shah and Usama Azad, who always encouraged me on. We accomplished a lot during our undergrad days and I still believe we can achieve more. Perhaps, one might say even pierce the Heavens.

Finally, this work was supported by the National Aeronautics and Space Administration (NASA) award NNX15AV01G. Thanks to the Chicago Park District for providing the Research Access Permit for the field tests.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

Symbol	Definition
mm	millimeter
nm	nanometer
m	meter
$\textcircled{C}$	Camera coordinate frame C
C_0	Origin of Camera coordinate frame C
$\hat{x}_C$	x-axis of Camera coordinate frame C
$\hat{y}_C$	y-axis of Camera coordinate frame C
$\hat{z}_C$	z-axis of Camera coordinate frame C, positive in forward direction of camera (Optical Axis)
X_C	coordinate component along the $\hat{x}_C$ axis (mm)
Y_C	coordinate component along the $\hat{y}_C$ axis (mm)
Z_C	coordinate component along the $\hat{z}_C$ axis (mm)
$\vec{r}_H^{\,J/K}$	Position vector of point J relative to point K, in the H coordinate frame (Generalized Notation)
$\textcircled{L}$	Lidar coordinate frame L
L_0	Origin of Lidar coordinate frame L
$\hat{x}_L$	x-axis of Lidar coordinate frame L
$\hat{y}_L$	y-axis of Lidar coordinate frame L, positive in forward direction of lidar
$\hat{z}_L$	z-axis of Lidar coordinate frame L, positive normal to lidar mounting surface
X_L	coordinate component along the $\hat{x}_L$ axis (mm)
Y_L	coordinate component along the $\hat{y}_L$ axis (mm)

Z_L	coordinate component along the $\hat{z}_L$ axis (mm)
f	Focal length (mm)
O	Camera optical center
Q	Image of Point P on the sensor plane?
P	Point of interest on object
k_u	Pixel density along $\hat{x}_C$ direction on Sensor (pixels per mm)
k_v	Pixel density along $\hat{y}_C$ direction on Sensor (pixels per mm)
u	Horizontal pixel coordinate
v	Vertical pixel coordinate
Res_H	Horizontal image resolution (no. of pixels)
Res_V	Vertical image resolution (no. of pixels)
θ_H	Horizontal Field of View (degrees)
θ_V	Vertical Field of View (degrees)
s	Skew parameter
α	Skewness angle (degrees)
k_n	Radial distortion parameter, where n=1,2,3
p_n	Tangential distortion parameter, where n=1,2
r	Radial distance from principal point on the sensor plane
x	Horizontal coordinate on sensor plane (mm)
y	Vertical coordinate on sensor plane (mm)
Pm	Perspective Projection matrix
S_0	Principal point on sensor plane (Sensor plane origin)

I_0	Pixel coordinates origin
$\hat{u}$	Horizontal axis of image
$\hat{v}$	Vertical axis of image
K	Camera Calibration matrix
d_L	Distance of object from the lidar
$\circ$	Degree
Ⓑ	Boom coordinate frame
$\hat{t}$	x-axis of the Boom coordinate frame
$\hat{b}$	y-axis of the Boom coordinate frame, positive in the direction of the boom
$\hat{u}$	z-axis of the Boom coordinate frame, positive normal to ground
Ⓐ	Intermediate coordinate frame
$\hat{a}_1$	x-axis of Intermediate coordinate frame A
$\hat{a}_2$	y-axis of Intermediate coordinate frame A
$\hat{a}_3$	z-axis of Intermediate coordinate frame A
$\text{Ⓒ}'$	Camera coordinate frame with misalignment
$\hat{x}_{C'}$	x-axis of Camera coordinate frame C′
$\hat{y}_{C'}$	y-axis of Camera coordinate frame C′
$\hat{z}_{C'}$	z-axis of Int Camera coordinate frame C′
el_L	Elevation angle (degrees) of lidar. Elevation angle of $\hat{y}_L$ axis from $\hat{b}$ axis
az_L	Azimuth angle (degrees) of lidar. Azimuth angle of $\hat{y}_L$ axis from $\hat{b}$ axis

el_C	Elevation angle (degrees) of Camera. Elevation angle of $\hat{z}_C$ axis from $\hat{b}$ axis
az_C	Azimuth angle (degrees) of Camera. Azimuth angle of $\hat{z}_C$ axis from $\hat{b}$ axis
cr_C	Misalignment angle (degrees) between $\hat{z}_C$ and the $\hat{y}_{C'}$ axes
u_L	Horizontal pixel coordinate for lidar point projected onto image plane
v_L	Vertical pixel coordinate for lidar point projected onto image plane
Ⓔ	East-North-Up (ENU) coordinate frame
$\hat{e}$	x-axis of the ENU coordinate frame, positive towards East direction
$\hat{n}$	y-axis of the ENU coordinate frame, positive towards North direction
$\hat{u}$	z-axis of the Boom coordinate frame, positive normal to ground
μ	Boom Heading angle (degrees). Angle of the $\hat{b}$ axis with respect to $\hat{n}$ axis and is positive east of north
An_0	Origin of the reflected antenna
W	Watts
V	Volts
s	Seconds
μs	Micro-seconds
ft	Feet
°C	Degrees Celcius
Hz	Hertz
g	Grams

l_H	Horizontal length of the checkerboard pattern visible to the camera (mm)
l_V	Vertical length of the checkerboard pattern visible to the camera (mm)
d_C	Perpendicular distance from the checkerboard pattern to the camera (mm)

LIST OF ABBREVIATIONS

Abbreviation	Definition
AGC	Automatic Gain Control
B/W	Black and White
BLC	Backlight compensation
CBR	Constant bit rate
CCTV	Closed-circuit television
CMOS	Complementary metal–oxide–semiconductor
DDNS	Dynamic Domain Name Server
DHCP	Dynamic Host Configuration Protocol
DNR	Digital Noise Reduction
DNS	Domain Name Server
ENU	East North Up
FOV	Field of View
FOV	Field of View
FPS	Frames per Second
FTP	File Transfer Protocol
GNSS	Global Navigation Satellite System
GNSS-R	Global Navigation Satellite System Reflectometry
GPS	Global Positioning System
HFOV	Horizontal Field of View
HTTP	Hypertext Transfer Protocol
IE	Internet Explorer

IEC	International Electrotechnical Commission
IIT	Illinois Institute of Technology
IP camera	Internet Protocol camera
IP rating	Ingress Protection rating
IR	Infrared
JPL	Jet Propulsion Laboratory
Kbps	Kilobits per second
LED	Light-Emitting Diode
LiDAR	Light Detection and Ranging
Mbps	Megabits per second
MP	Mega-Pixels
NFS	Network File System
NMEA	National Marine Electronics Association
NTP	Network Time Protocol
ONVIF	Open Network Video Interface Forum
OSD	On-Screen Display
P2P	Peer to Peer
POE	Power over Ethernet
QVGA	Quarter Video Graphics Array
RGB	Red Green Blue
RH	Relative Humidity
RTSP	Real Time Streaming Protocol

SMD	Surface Mounted Devices
SMTP	Simple Mail Transfer Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
uPnP	Universal Plug and Play
URL	Uniform Resource Locator
VBR	Variable bit rate
VFOV	Vertical Field of View
VGA	Video Graphics Array
WDR	Wide Dynamic Range

ABSTRACT

Global Navigation Satellite System Reflectometry (GNSS-R) relies upon detecting the GNSS signals reflected off a surface and then analyzing the reflected signal to obtain surface characteristics. GNSS-R has become one of the many additional applications of the readily available GNSS signals [1], alongside more traditional remote sensing of ionospheric monitoring [2], beyond the intended GNSS purposes of providing position, navigation, and timing estimation.

In previous work, GPS signals reflected off Lake Michigan in Chicago have been collected using a specially designed portable sensor suite [3]. The data collected is then analyzed to differentiate between surface ice and water conditions, as well as obtain other characteristic information such as surface reflectivity [4]. The goal is to provide a way for remote sensing of seasonal ice formation beyond just satellite imagery which can be affected by cloud cover. To confirm the validity of the GNSS-R results there needs to be a separate reference against which to compare.

This work demonstrates the sensor fusion between camera and lidar to reconstruct the lake surface, to provide that truth reference for comparison against the results of the GPS reflectometry signal processing. For this setup, the camera provides visual information about the lake surface, while the lidar provides distance information with respect to the sensor suite. Combining the data from the two sensors allows backward projection of the camera image to reconstruct the lake surface and its features. The backward projection relies upon knowledge of the camera's intrinsic properties alongside distance information of the features captured by the camera. Each pixel of the camera image is then transformed to its 3D position relative to the sensor system. This produces a 3D map of the lake surface, as captured by the sensors. The estimated point at which the GPS signal reflects off the surface, the specular point, is calculated by the satellite position at the time of interest and

the receiver location. This point is then mapped onto the reconstructed surface to identify the exact location where the signal reflected and compare the surface visually to the results from the signal analysis.

Time-varying camera-lidar-specular-point maps of the data campaigns conducted for this project are created for comparison with the GPS signal analysis. Multiple data campaigns were performed during which the Lake Michigan surface had surface ice, water or a mixture of the two. The lake surface is reconstructed for different timestamps, using the appropriate image frame and lidar frame. Combining chronologically, the changes in the lake surface can then be observed along with the movement of the specular point, due to the movement of the GPS satellites. Any satellites passing over a boundary between water and ice on the lake surface are identified and time stamped, to then be compared to the GPS signal analysis results [3].

CHAPTER 1
INTRODUCTION

Global Positioning System (GPS), originally known as the NAVSTAR GPS, is one of many constellations of Global Navigation Satellite Systems (GNSS) used for geolocation and timing information. All that is required to make use of its capabilities is a GPS receiver which over the years has made its way onto our phones, cars and planes. However, with such a vast array of satellites transmitting data independently without the need for the user to transmit any data back, they have been considered for projects outside of their originally intended role. One such application is in remote sensing using reflected signals, which when using GNSS satellites is known as Global Navigation Satellite System Reflectometry (GNSS-R).

The idea behind this is simple: to use the already available GPS satellites as transmitters and receive a GPS signal after it reflects off the surface of the Earth. This signal is used to obtain any characteristic information required of the surface that it reflected off. For this work, as part of the GNSS-R project at the IIT Space Weather Lab, a mobile sensor suite was designed [5] to collect data comprising of GPS antennae, a weather station, cameras, a lidar and supporting electronics. The camera and lidar, although not dealing with the GPS signals directly are helpful in providing supporting data on the surface, explained further in 1.1. Methods for acquisition of the weak reflected GPS signals [6][7] and their processing to obtain surface characteristics was performed.

1.1 Motivation

The GNSS-R project is based on using GPS signals to differentiate between ice and water on the lake surface [4]. The GPS signal reflects off a particular point on the surface, known as the specular point. This point of reflection can be determined using the position of the GPS satellite in its current orbit and the location of GPS receiver, picking up the signal after it reflects off the surface. Analyzing the signal obtained could potentially provide characteristics of the point on the surface this signal reflected off of.

However, in all this there is no real visual way to prove the results. There is a necessity to have a “truth” reference to compare against, which is where a camera would help. Previous work done in this project used the lidar point cloud overlaid with the GPS specular points map to obtain a sense of the surface conditions for a truth reference [8][9]. While this does work, it is hard to decipher exact surface conditions with the point cloud alone, which is why the inclusion of a camera is beneficial.

The camera can save visual information about the surface and be used to see where the specular point actually is and the surface features there. Reconstructing the surface and then plotting the specular point on top would allow visual confirmation of the lake surface conditions at the point of interest. This can then be compared to the results obtained from analysis of the reflected GPS signal.

However, a camera alone is not enough to accurately identify the specular point location in the image. This is because the 2D visual information from camera is not enough to determine where the visible features are actually located in 3D space. Reconstructing the lake surface from camera images would require additional depth information. In this thesis it is done using a lidar. The lidar provides a point cloud of the environment giving 3D coordinates for features detected. The only thing left then is to combine the two and obtain visual data of the camera in 3D coordinates.

1.2 Prior Research

The general concept of camera-lidar sensor fusion is not new; it has been used in the robotics and automotive fields among others for position and pose estimation. Each sensor fills in for weaknesses of the other, the camera lacking depth information but providing higher resolution visual data and the lidar with lower resolution data but with depth perception. The sensors together are used for better feature detection [10] and improved environment perception [11]. Backward projection using lidar and camera fusion has been applied to indoor environment reconstruction [12]. Autonomous vehicles with camera-lidar setups also perform similar reconstructions of their surrounding environment, especially for solid object feature detection [13].

Rather than navigation and attitude sensing, this project application involves sensing the natural environment, particularly the lake surface. Previously studies have been done to topographically map large water bodies. One such work used an airborne lidar scanning the ground below [14]. The paper presented the problem of lidar lasers not having enough reflectance off calm water bodies. The water bodies were identified by the lack of lidar returns [15] and then an estimated water surface was generated. A similar study, also using an airborne lidar for Water Surface Mapping used a ratio index to identify water bodies. This paper identified the low reflectivity of lidar off the water as well, using a spectral library created through tests by studies done at JPL in California [16].

Previous studies have also been done to use only a camera for remote sensing water using spectral analysis of the retrieved airborne camera images [17]. In previous study, the researchers proposed a cheaper solution to multi-sensor or even expensive sensor solutions as alternatives to satellite imagery. Where satellite imagery can be influenced by cloud cover, the potential solution detailed in the paper mentioned flying cameras in a plane, over the surface to be mapped.

All of these studies sensed environments consisting of primarily solid surfaces (e.g., urban streets in the case of autonomous vehicles) or primarily liquid surfaces near solid ground with a defined boundary (e.g., river water surface mapping). In this thesis work, we are interested in sensing a natural body of water whose surface phase of matter varies spatially.

1.3 Contributions

The objective of this work is to provide a method to reconstruct the lake surface in 3D, providing a truth reference to compare against the reflected GPS signal processing results. The contributions made are as follows:

1. Developed a method to backward project a camera image, using lidar, to reconstruct the imaged environment in 3D coordinates

2. Demonstrated the camera-lidar projection method to reconstruct the lake surface during surface ice conditions

3. Reconstructed lake surface during surface water conditions

4. Reconstructed lake surface during mixed surface ice and water conditions

Chapter 2 reviews the background principles of the camera and the lidar sensors, and the backward projection concept. Chapter 3 describes the algorithm developed for computing the backward projection, explaining the processing performed on the sensor data. Chapter 4 describes the lab tests performed when testing out this algorithm. Chapter 5, Chapter 6 and Chapter 7 review the results from applying the algorithm to data collected in field tests with lake surface ice, surface water and mixed surface ice-and-water conditions, respectively. Finally, Chapter 8 summarizes the results and describes future work.

CHAPTER 2
BACKGROUND

2.1 Pinhole camera model

To perform the sensor fusion, the camera needs to be properly understood. The camera is a device used to capture images of the real world. This is the most important piece of the reconstruction process as it provides the visual information of the environment that needs to be transformed. The camera is clearly a complex device, but for the sake of the reconstruction, will be treated as a pin hole camera. A simple pin hole camera model can be seen in Figure 2.1. According to this model there is only a tiny aperture for light rays to pass through to project an image on the sensor.

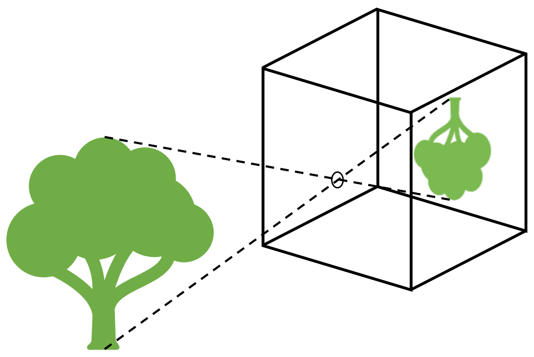


Figure 2.1. Simple Pinhole camera

Expanding on this model, Figure 2.2 shows the projection of a light ray from some point of interest on an object, on to the sensor plane, in the camera coordinate frame. According to this system, $\hat{z}_C$ axis along the optical axis of the camera, positive toward the direction the camera is facing. The $\hat{x}_C$ axis and $\hat{y}_C$ axis defining the camera's horizontal and vertical field-of-view directions respectively, which are parallel to the sensor plane. The position vector $\vec{r}^{P/O}$ is the point of interest on an object, represented as P, from the camera optical center O and position vector $\vec{r}^{Q/O}$ is the image of the point P on the sensor plane, represented as Q, from the from the camera optical center O. The optical center is essentially the camera's "pinhole". It is important to note that the sensor plane is actually in the -$\hat{z}_C$ direction at a distance f behind the optical center O, with the image inverted. For ease of understanding, the sensor plane is being projected in the same plane as the object, which would technically make this the virtual image. However, the relationships and properties remain the same while making this visually easier to interpret.

Some of the Intrinsic properties of the camera, also known as the camera parameters, can be seen in Figure 2.2. The properties include the following:

1. **Camera optical center.** The camera optical center is the point where all the light rays pass straight through with angle of incidence and angle of emergence about this point being the same. In a pinhole camera the aperture, or simply the hole, is the optical center. In more complex cameras with multiple lenses it becomes more complex to locate the exact location. For this work, it is considered to be at the center of the front face of the camera. The optical center is also the origin of the camera coordinate system.
2. **Optical axis.** The $\hat{z}_C$ is also known as the optical axis.
3. **Focal length.** The focal length f is the distance of the sensor plane away from the camera optical center, along the optical axis.
4. **Principal point.** This is the point on the sensor plane which intersects with the optical axis. As seen in Figure 2.2, the x and y coordinates of point Q on the sensor plane are with respect to this point. Usually when dealing with points on the sensor plane, it is treated as a 2D plane (as all points on the plane

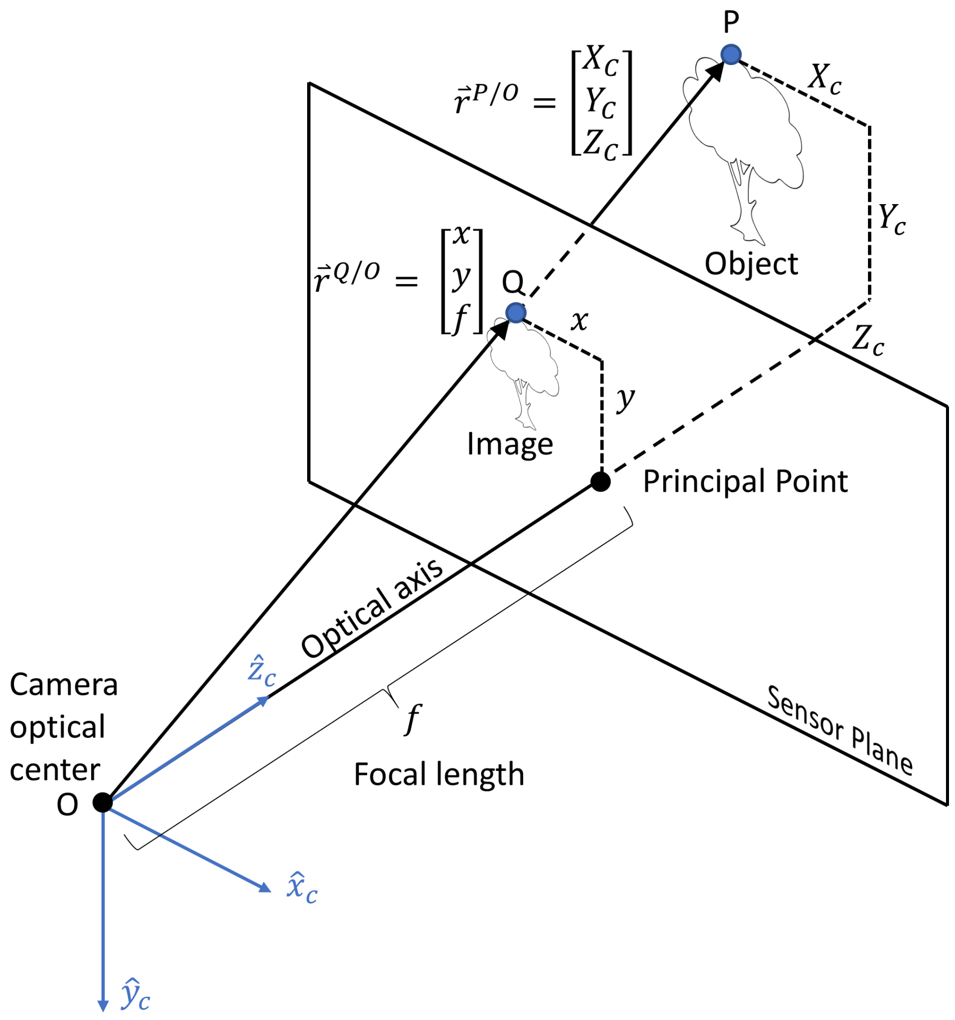


Figure 2.2. Object projected onto sensor plane

have the same third dimension with respect to the optical center, being f), the principal point is considered the origin.

5. **Pixel density.** It is the number of pixels generated along each dimension of the sensor. k_u and k_v represent the pixel density along the $\hat{x}_C$ and $\hat{y}_C$ directions on the sensor.

6. **Image resolution.** The image generated has a certain number of pixels along its horizontal dimension u referred to as Res_H and vertical dimension v referred

to as Res_V, dependant on the sensor pixel density.

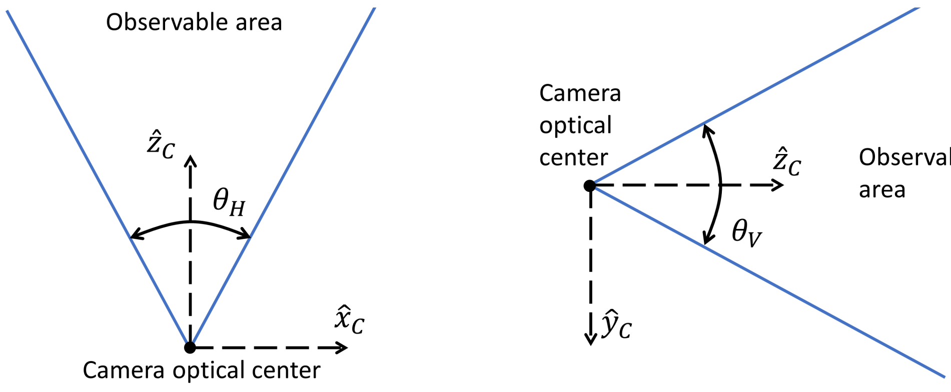
7. **Distortion coefficients.** Compared to an ideal pinhole camera, a modern camera can have multiple sources of distortion. Thus, to model the camera as a pinhole camera the distortion effects in the image must be removed. The different types of distortion are modelled using distortion coefficients

These camera parameters are estimated using a process known as 'geometric camera calibration' or simply 'camera calibration', also known as camera resectioning. This is done using the Camera calibrator app further explored in Section 3.1.

There are other properties of a camera, such as the field of view (FOV), can be determined from the intrinsic properties. The FOV is what viewing angle is observable to the camera. This is generally broken down into the horizontal FOV (HFOV) and vertical FOV (VFOV). The HFOV relates to the observable area visible to the camera horizontally, measured as the angle θ_H which is the angular extent visible to the camera in the $\hat{z}_C$ $\hat{x}_C$ plane. Similarly, the VFOV relates to the observable area visible to the camera vertically, measured as the angle θ_V which is the angular extent visible to the camera in the $\hat{z}_C$ $\hat{y}_C$ plane. Figure 2.3 shows this concept.

2.2 Image Distortion

To remove the distortion and represent an ideal pinhole camera, its necessary to understand the types distortion. Figure 2.4 shows an image with distortion. The distortion in the image is apparent when looking at the shelf to the left which seems curved and even the tiles on the floor seem to bend more towards the edges of the image, when in reality they are both straight. Cameras tend to have a mix of these types of distortion. The distortions are modelled using distortion coefficients. These model the pixel locations in the image with distortion and can be used to obtain the undistorted pixel locations. For this work its important to note that the distortion



(a) Horizontal FOV (b) Vertical FOV

Figure 2.3. Field of View of the camera

removal was performed by a built-in Matlab funtion, called ‘undistortImage’, as part of the image processing toolbox. The distortion models, detailed below, are used by the function to undistort the image. They are explained for the sake of understanding the background behind the undistortion but are not a focus of this work.

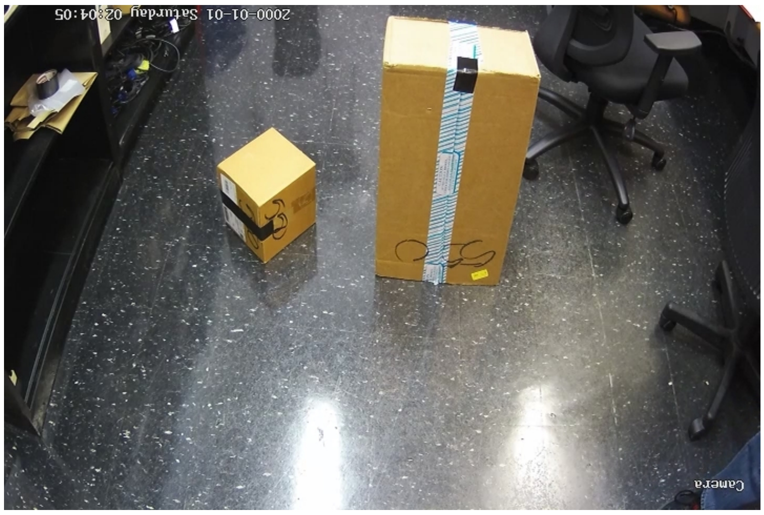


Figure 2.4. Image showing lens distortion

2.2.1 Radial distortion. A pinhole camera does not have any lenses but most modern cameras do to further provide better images with sharper quality and wider field of view (FOV). This comes at a cost as lens distortion is introduced into the image. This occurs when light rays entering the camera bend more near the edges of a lens than they do at its optical center, as can be seen in Figure 2.5. This is common for camera lenses which are used to increase or decrease the visual FOV of the camera.

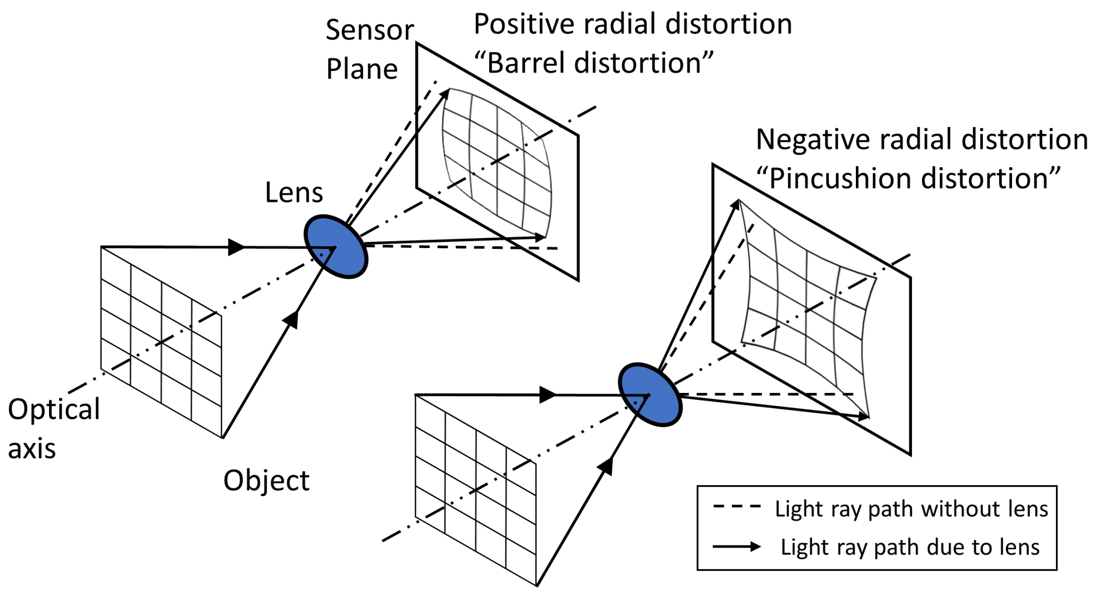

Sensor Plane
Positive radial distortion "Barrel distortion"
Negative radial distortion "Pincushion distortion"
Lens
Optical axis
Object
Light ray path without lens
Light ray path due to lens

Figure 2.5. Radial distortion

Radial distortion can be modelled using radial distortion coefficients as shown by Equations 2.1 and 2.2 [18] [19]. Usually two coefficients suffice, however, in the case of wide angle lenses which cause sever distortion, three coefficients can be included.

$$x_{\text{distorted}} = x(1 + k_1\, r^2 + k_2\, r^4 + k_3\, r^6) \tag{2.1}$$

$$y_{\text{distorted}} = y(1 + k_1\, r^2 + k_2\, r^4 + k_3\, r^6) \tag{2.2}$$

Where, x and y are the undistorted pixel locations of the image on the sensor plane, along the $\hat{x}_C$ and $\hat{y}_C$ axis respectively. k_1, k_2, and k_3 are the radial distortion coefficients of the lens. Also $r = \sqrt{x^2 + y^2}$, is the radial distance from the principal point.

2.2.2 Tangential distortion. This occurs when the lens and the sensor plane are not oriented parallel to each other, as seen in Figure 2.6. This is generally caused by defects in the camera assembly.

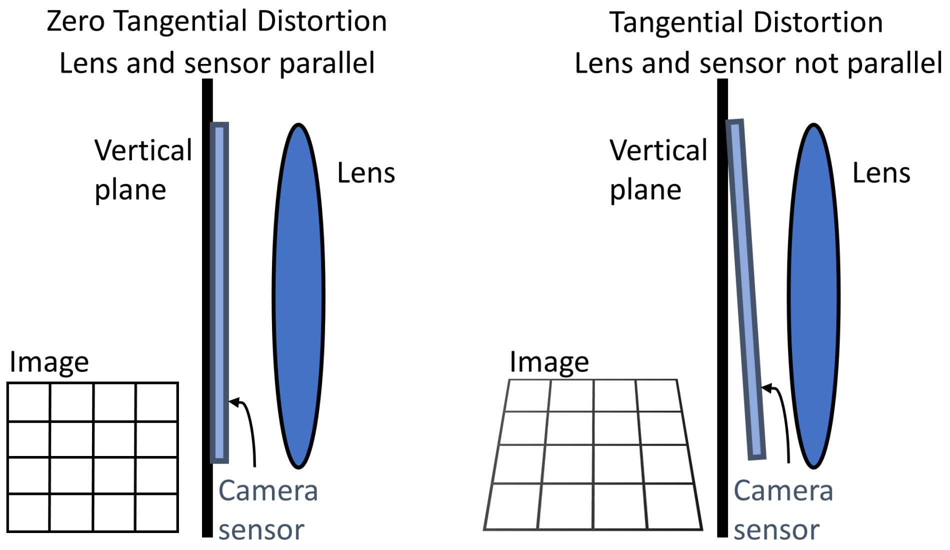


Figure 2.6. Tangential distortion

Tangential distortion, similar to radial distortion, can be modelled using tangential distortion coefficients as shown by Equations 2.3 and 2.4 [18] [19].

$$x_{distorted} = x + [2\, p_1\, x\, y + p_2\, (r^2 + 2\, x^2)] \tag{2.3}$$

$$y_{distorted} = y + [p_1\, (r^2 + 2\, y^2) + 2\, p_2\, x\, y] \tag{2.4}$$

Where, x and y are the undistorted pixel locations of the image on the sensor plane, along the $\hat{x}_C$ and $\hat{y}_C$ axis respectively. p_1, and p_2 are the tangential distortion coefficients of the lens. Also as before, $r = \sqrt{x^2 + y^2}$, is the radial distance from the principal point.

2.2.3 Shear distortion. Camera sensors can tend to have defects which can sometimes cause the axes of the image to not be perpendicular, also known as shear distortion shown in Figure 2.7(b). This is modelled using a skew parameter s. If the skew parameter is zero then there is no skewness and the pixels are rectangular, however, if the skew parameter is non-zero then the pixels in the image are assumed to be more of a parallelogram shape with some skewness angle α shown in Figure 2.7(a). For this work the skewness is assumed to be negligible.

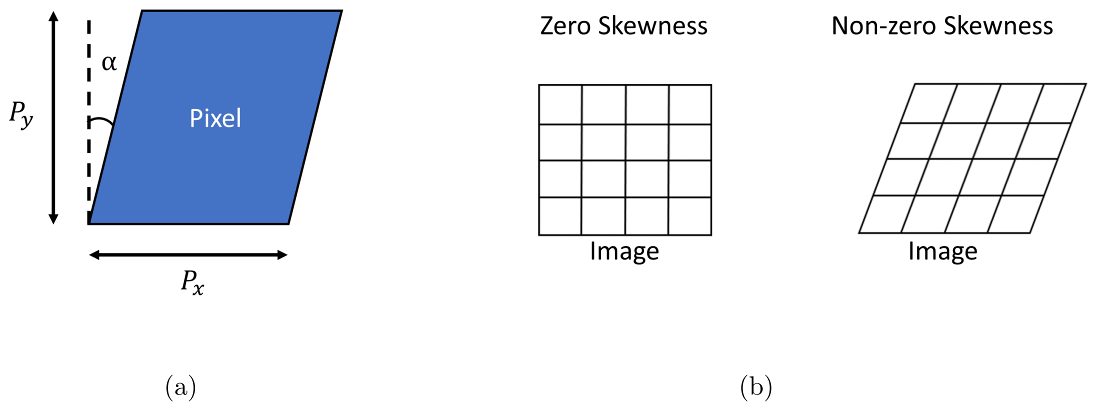


Figure 2.7. (a) Skewed pixel, (b) Shear distortion in image

2.3 Forward projection

Before being able to transform a camera image into a reconstructed environment, it is necessary to first understand how a camera obtains its image from the environment. This process is known as forward projection, where light rays from the environment are captured by the camera to create an image. This process can be broken down into multiple steps. The light rays from the environment are first captured onto the camera sensor, before the sensor converts them into image pixels.

Looking back at Figure 2.2, the vectors $\vec{r}^{P/O}$ and $\vec{r}^{Q/O}$ are collinear and can have their components related, as shown by Figure 2.8, using similar triangles, providing Equations 2.5 and 2.6.

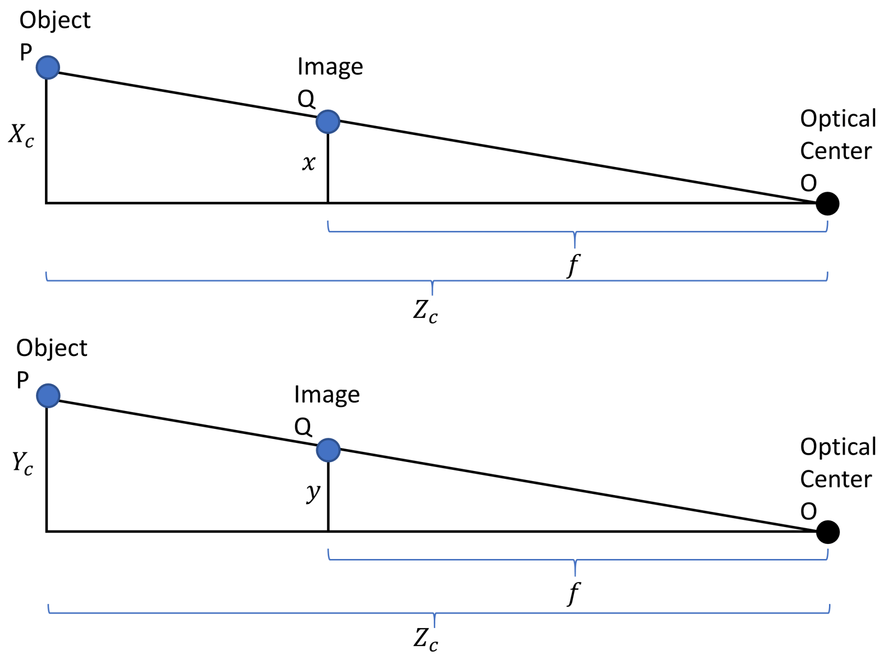


Figure 2.8. Similar triangles relating components of Point P and Image Q

$$x = f\,\frac{X_C}{Z_C} \tag{2.5}$$

$$y = f\,\frac{Y_C}{Z_C} \tag{2.6}$$

These equations can be formulated into matrix form, as shown in Equation 2.7, where the Perspective projection matrix Pm transforms the camera coordinates X_C, Y_C, Z_C into sensor plane coordinates x, y. Factoring out $\frac{1}{Z_C}$, it can also be written in the form shown in Equation 2.8, where the camera coordinates, position vector $\vec{r}^{P/O}$, are transformed into the sensor coordinates. the third row is included to make a square matrix that is invertible.

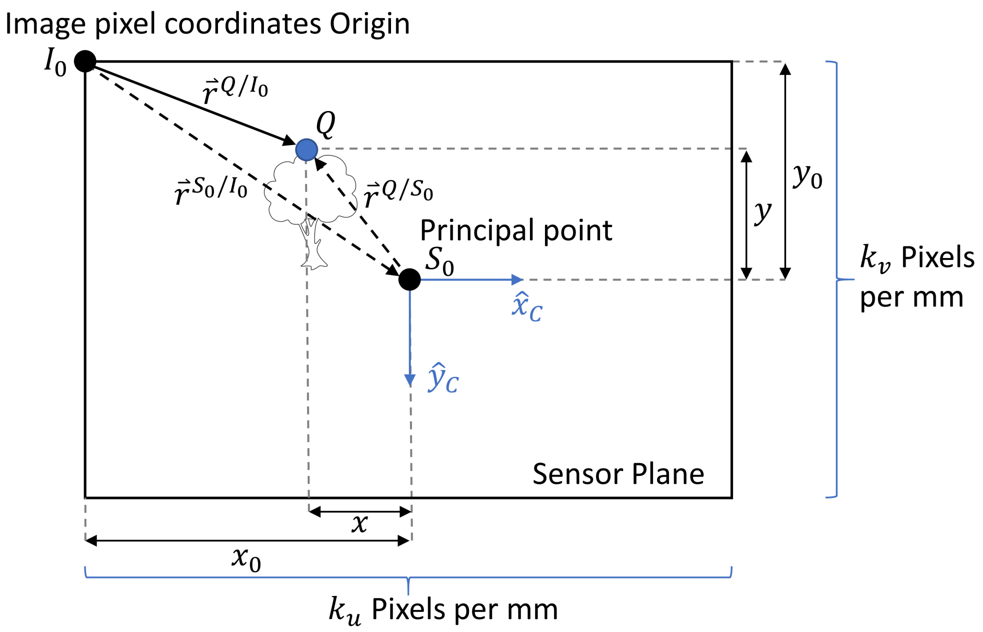
$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{Pm} \begin{bmatrix} \frac{X_C}{Z_C} \\ \frac{Y_C}{Z_C} \\ 1 \end{bmatrix} \tag{2.7}$$

$$= \frac{1}{Z_C} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} \tag{2.8}$$

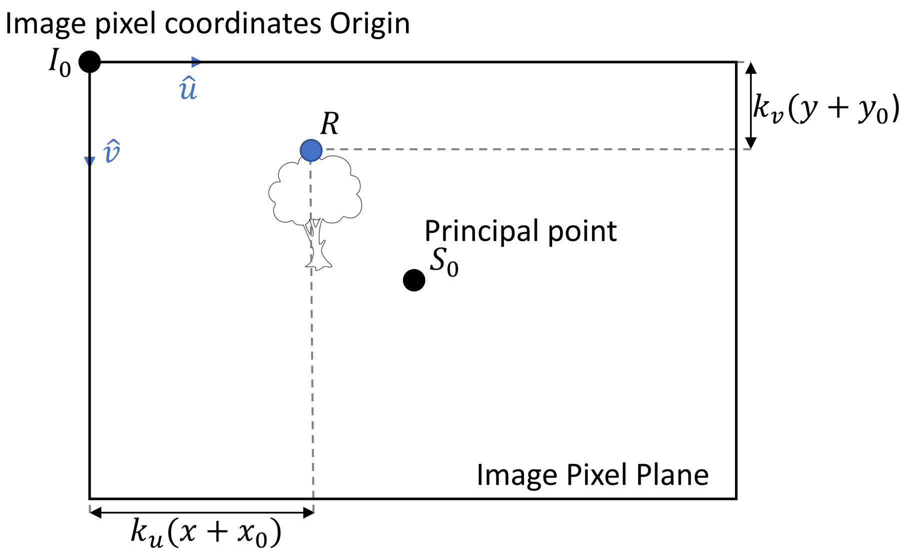
Image pixels have their coordinate system origin at the top left of an image. Therefore before converting the Image from sensor coordinates into image pixel coordinates, the origin has to be translated as shown in Figure 2.9(a) and formulated in Equation 2.9. Position vector $\vec{r}^{Q/S_0}$ are the 2D coordinates of the image point Q on the sensor plane from the principal point S_0. Position vector $\vec{r}^{Q/I_0}$ are the coordinates of the Image point Q from the pixel coordinates origin I_0. Position vector $\vec{r}^{S_0/I_0}$ shows the the principal point S_0 from the pixel coordinates origin I_0.

$$\begin{aligned} \vec{r}^{Q/I_0} &= \vec{r}^{P_0/I_0} + \vec{r}^{Q/P_0} \\ &= \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix} \\ &= \begin{bmatrix} x_0 + x \\ y_0 + y \end{bmatrix} \end{aligned} \tag{2.9}$$

From here converting to the image plane in pixels depends on knowledge of the sensor pixel resolution. The camera sensor contains light-sensitive spots called photosites, which record light information that falls on them. These are laid out in a



(a) Sensor plane in world coordinates (usually millimeters)



(b) Image plane in pixels

Figure 2.9. Sensor plane coordinates converted into Image pixel coordinates

grid and correspond to pixels in the image generated. The coordinates of these pixels is with respect to the $\hat{u}$ axis which is aligned horizontally, pointing to the right of the image, and $\hat{v}$ axis which is aligned vertically, pointing to the bottom of the image, as shown in Figure 2.9(b). The pixel density, designated by k_u and k_v along the $\hat{u}$ and $\hat{v}$ directions respectively, is the number of pixels per mm along each length of the sensor. Thus, knowing the position of the image projection on the sensor and this density allows conversion into pixels. This is shown in Figure 2.9(b) where point R is the projected point of interest Q transformed into the image pixel plane, in pixel coordinates u and v. Building upon Equation 2.9, the transformation to image pixels is shown in Equation 2.10.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} k_u\,(x_0 + x) \\ k_v\,(y_0 + y) \end{bmatrix} \tag{2.10}$$

Separating Equation 2.10 into individual matrices and adding an extra dimension to create a square matrices, the result is Equation 2.12 which shows the conversion from sensor plane coordinates x, y into image pixel coordinates u, v.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_u & 0 & 0 \\ 0 & k_v & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.11}$$

$$= \begin{bmatrix} k_u & 0 & k_u\,x_0 \\ 0 & k_v & k_v\,y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.12}$$

Combining Equations 2.7 and 2.12 the complete transformation from camera coordinates to pixel coordinates is attained, also known as forward projection, as shown by Equation 2.14. Matrix K is known as the Calibration matrix, made up of the intrinsic properties of the camera. The third row is added here to make matrix K invertible.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_u & 0 & k_u x_0 \\ 0 & k_v & k_v y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{X_C}{Z_C} \\ \frac{Y_C}{Z_C} \\ 1 \end{bmatrix} \tag{2.13}$$

$$= \underbrace{\begin{bmatrix} k_u f & 0 & k_u x_0 \\ 0 & k_v f & k_v y_0 \\ 0 & 0 & 1 \end{bmatrix}}_{K} \begin{bmatrix} \frac{X_C}{Z_C} \\ \frac{Y_C}{Z_C} \\ 1 \end{bmatrix} \tag{2.14}$$

If skew distortion was considered then an s term would be added to the calibration matrix K as shown in Equation 2.15 and f would no longer be the same in the x and y directions on the sensor, introducing f_x and f_y instead. However, for this work, it is assumed that there is zero skew in the camera sensor.

$$K = \begin{bmatrix} k_u f_x & s & k_u x_0 \\ 0 & k_v f_y & k_v y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.15}$$

Equation 2.14 can be simplified by factoring out $\frac{1}{Z_C}$, obtaining Equation 2.16 with vector P representing 3D camera coordinates of a point of interest being projected and vector R representing the pixel coordinates of the point.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{Z_C} \begin{bmatrix} k_u f & 0 & k_u x_0 \\ 0 & k_v f & k_v y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} \tag{2.16}$$

$$\begin{bmatrix} R \end{bmatrix} = \frac{1}{Z_C} \begin{bmatrix} K \end{bmatrix} \begin{bmatrix} P \end{bmatrix} \tag{2.17}$$

2.4 Backward projection

For backward projection the goal is to do the opposite of the forward projection, where the pixel coordinates are known and the 3D camera coordinates must be determined. For this, inverting Equation 2.16 produces the required backward projection, shown in Equation 2.19.

$$\begin{bmatrix} P \end{bmatrix} = Z_C \begin{bmatrix} K \end{bmatrix}^{-1} \begin{bmatrix} R \end{bmatrix} \tag{2.18}$$

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} = Z_C \begin{bmatrix} k_u f & 0 & k_u x_0 \\ 0 & k_v f & k_v y_0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{2.19}$$

Its important to note that to obtain all the Camera coordinates, backward projection still requires one of the camera coordinates to be known, namely the Z_C depth component. The calibration matrix K is built with the known intrinsic properties of the camera and the image pixel coordinates are obtained from the image.

2.5 LiDAR

The lidar, abbreviated from Light Detection and Ranging, is a device used in remote sensing applications for range detection. A laser is emitted, which bounces off an object in the environment back to the sensor. Knowing the speed of light and the time it took from emitting the laser to detecting it, the exact distance of the object can be determined. This is shown in Equation 2.20 where d_L is the distance of the object from the lidar, c is the speed of light in air and t is the time of flight of the pulse. The reason its divided by 2 is because the light travels twice the distance, from the lidar to the object then back. In most commercial lidar sensors, there are usually multiple pulsating lasers and rotating parts to allow for more area to be covered, being the effective FOV of the lidar. Figure 2.10 shows the concept behind the working of a lidar.

$$d_L = \frac{c\,t}{2} \tag{2.20}$$

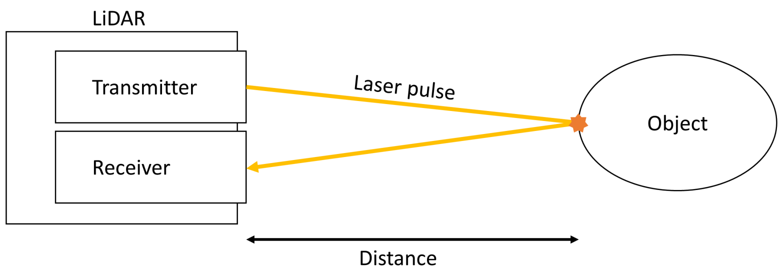


Figure 2.10. The concept behind LiDAR

The lidar sends out many laser pulses as it sweeps its FOV. This produces a point cloud, which is just a 3D map of all the laser returns plotted at the distance they were calculated to be reflected back from. This can be used to map the environment in 3D world coordinates. Figure 2.11 shows a point cloud generated by a lidar attached to the roof of a car. The color of the points can be set by the user to show distance, return intensity, object identity or other variables.

The lidar has its own coordinate system about which it provides its point cloud data. This lidar reference frame has its origin at the center of the lidar with the $\hat{y}_L$ axis, pointing forward with respect to the sensor, along which the azimuth and elevation angles of any returns are 0. The $\hat{z}_L$ axis points normal to the sensor mounting surface and the $\hat{x}_L$ axis, relatively speaking points right, completing the right handed coordinate system. For this work, the lidar is used to obtain the depth information required for backward projection. The Z_C values, being the depth information in question, are obtained through the use of the lidar, even though there is more processing of the lidar data required before obtaining the Z_C, which is explained further in Chapter 3.

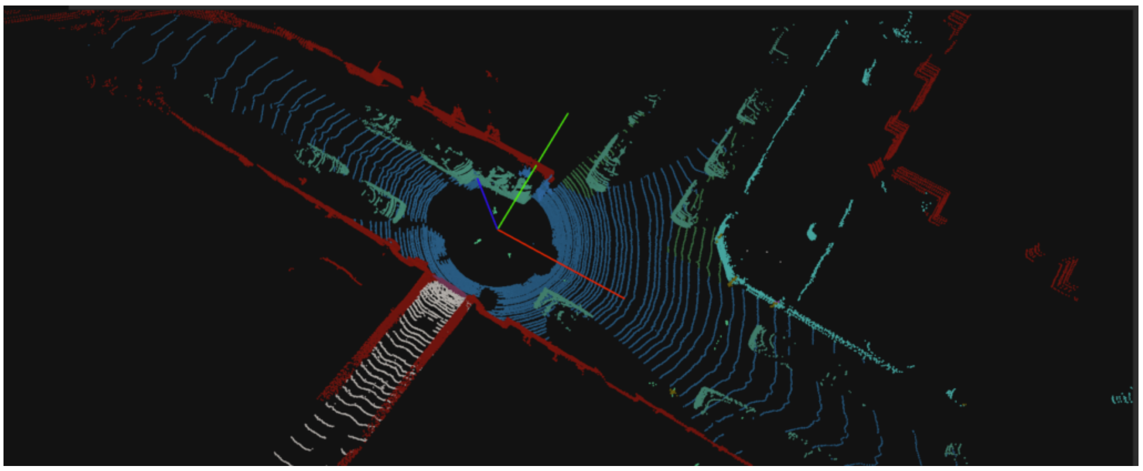
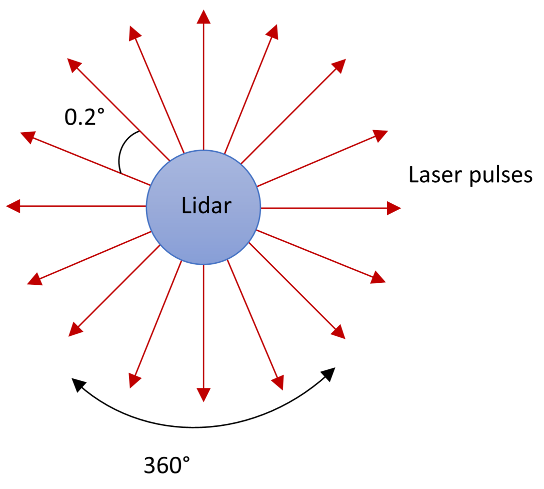


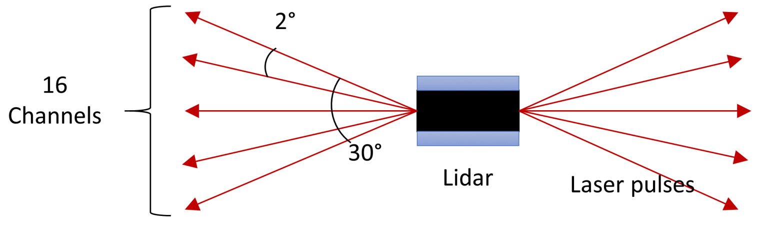
Figure 2.11. A point cloud generated from lidar mounted on top of a car (at the origin with red, green, and blue body axes) as viewed from above the vehicle. [20]

The lidar used for this work, operated at a 900 nm wavelength, making it infrared light. It is invisible and not dangerous to the naked eye, making it easy to work with. The exact specifications of the lidar used can be seen in Appendix B. This lidar is designed for modelling environments with solid structures such as cars, poles and walls. However, it struggles to obtain returns from liquid water surfaces, as the reflectively of light at this wavelength is low on such surfaces [16]. The effect of this is shown later in some of the field tests with lake water conditions. This interaction makes it good for identifying water surface conditions but bad for surface reconstruction. A little creativity was necessary to be able to still perform the backward projection regardless of the scarcity of lidar returns, further detailed in the water surface field test in Chapter 6.

This lidar in question also not only sweeped 360° around itself, giving it a 360° horizontal FOV, but also had multiple laser channels a couple degrees apart vertically, giving it a 30° vertical FOV. Figure 2.12 shows a top view, being the horizontal FOV, and a side view, being the vertical FOV, diagram of the laser pulses emitted by the lidar. The angular separation between the laser pulses is smaller horizontally as compared to vertically. This makes it have a denser resolution horizontally and gives the point clouds their iconic line-like features.



(a) Top view



(b) Side view

Figure 2.12. Lidar laser pulses defining its FOV

CHAPTER 3
BACKWARD PROJECTION ALGORITHM

The concept behind the backward projection, as explained in Chapter 2, requires more processing of the actual data obtained from the sensors. Steps must be taken before eventually using the data for the surface reconstruction. Equation 2.19 shows the matrix manipulation to convert each individual pixel of a camera into its respective 3D coordinate in the Camera reference frame. A flowchart of the process of performing this on an entire camera image can be seen in Figure 3.1

3.1 Matlab camera calibrator app

The camera image obtained cannot be used immediately and requires some post-processing. To accomplish this the camera properties must be known. As shown in the first column of the flowchart in Figure 3.1, the Camera Calibrator app, part of the Image Processing Toolbox in Matlab, is used. These parameters, which include distortion parameters, focal length, principal point and image resolution, are used to remove distortions as well as provide the camera calibration matrix K to perform the forward and backward projections.

To obtain the aforementioned parameters, calibration must be performed using the application. This requires using a checkerboard pattern as a calibration target. In this case the checkerboard pattern printed out is taped onto a large piece of cardboard for support. Multiple images are input with the checkerboard pattern in view of the camera, at different angles along with the size of the checkerboard squares. This is seen in Figure 3.2, where the application identifies the intersecting points of the checkerboard squares and comparing the real world distance between these points (provided as an additional input) to the pixel distances in the images, calculates the

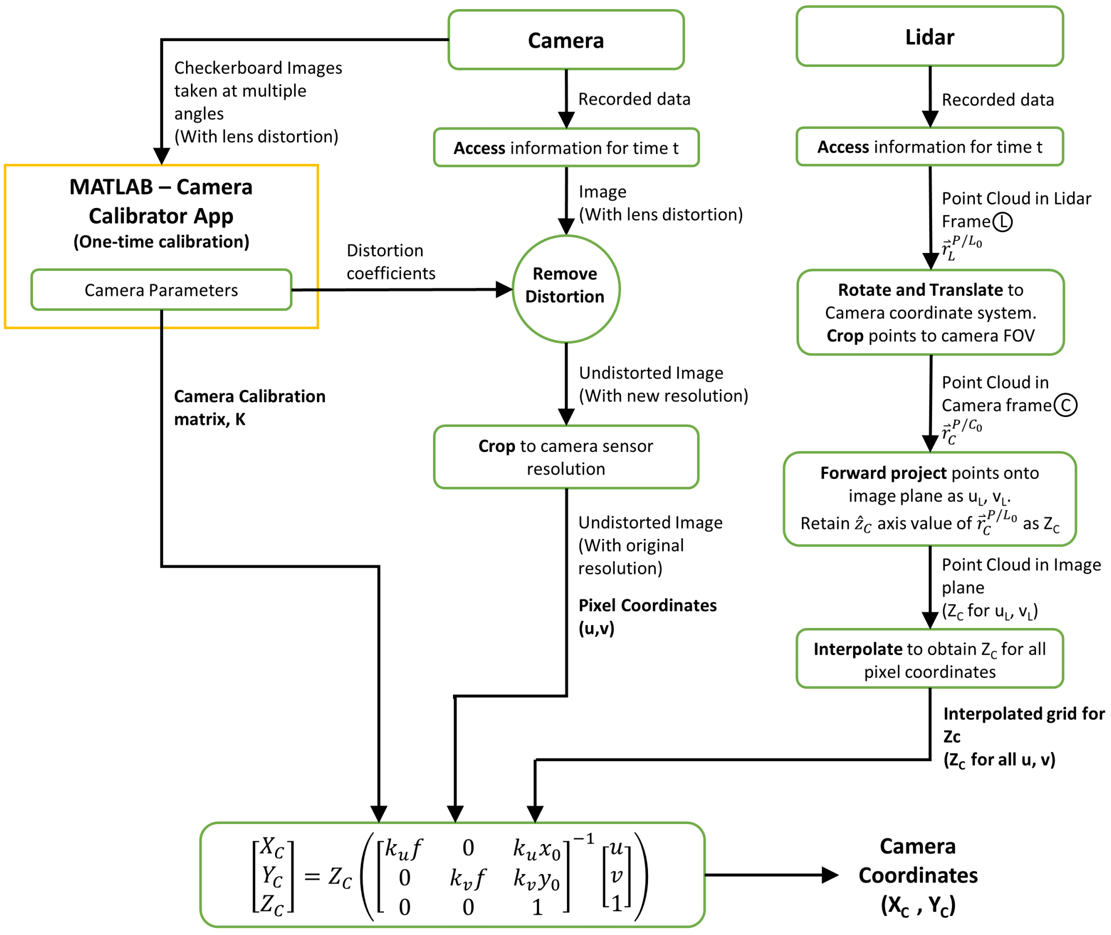

Camera
Lidar
Checkerboard Images taken at multiple angles (With lens distortion)
Recorded data
Recorded data
Access information for time t
Access information for time t
MATLAB – Camera Calibrator App (One-time calibration)
Camera Parameters
Image (With lens distortion)
Point Cloud in Lidar Frame Ⓛ $\vec{r}_L^{\,P/L_0}$
Distortion coefficients
Remove Distortion
Rotate and Translate to Camera coordinate system. Crop points to camera FOV
Camera Calibration matrix, K
Undistorted Image (With new resolution)
Point Cloud in Camera frame Ⓒ $\vec{r}_C^{\,P/C_0}$
Crop to camera sensor resolution
Forward project points onto image plane as u_L, v_L. Retain $\hat{z}_C$ axis value of $\vec{r}_C^{\,P/L_0}$ as Z_C
Undistorted Image (With original resolution)
Pixel Coordinates (u,v)
Point Cloud in Image plane (Z_C for u_L, v_L)
Interpolate to obtain Z_C for all pixel coordinates
Interpolated grid for Zc (Z_C for all u, v)
$\begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} = Z_C \left(\begin{bmatrix} k_u f & 0 & k_u x_0 \\ 0 & k_v f & k_v y_0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \right)$
Camera Coordinates (X_C , Y_C)

Figure 3.1. Flowchart of the full backward projection algorithm

desired parameters [21]. As shown in the flowchart in Figure 3.1, the calibration to obtain camera parameters is a one-time action. These are properties of the camera itself so do not change. They are then used when processing images from the camera obtained during lab tests and field data campaigns.

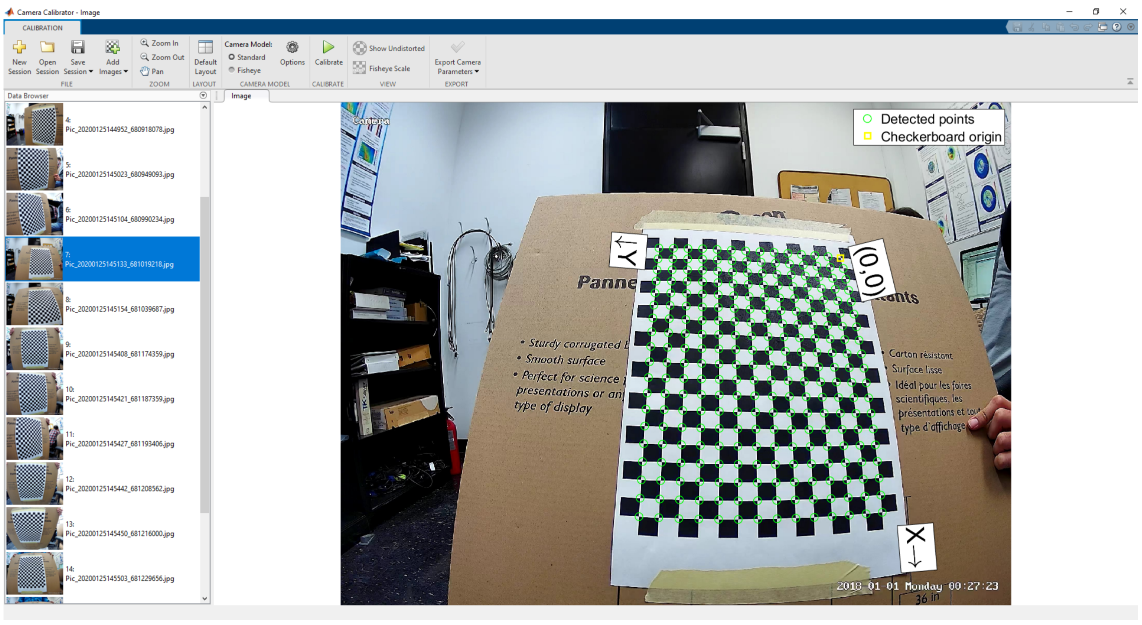
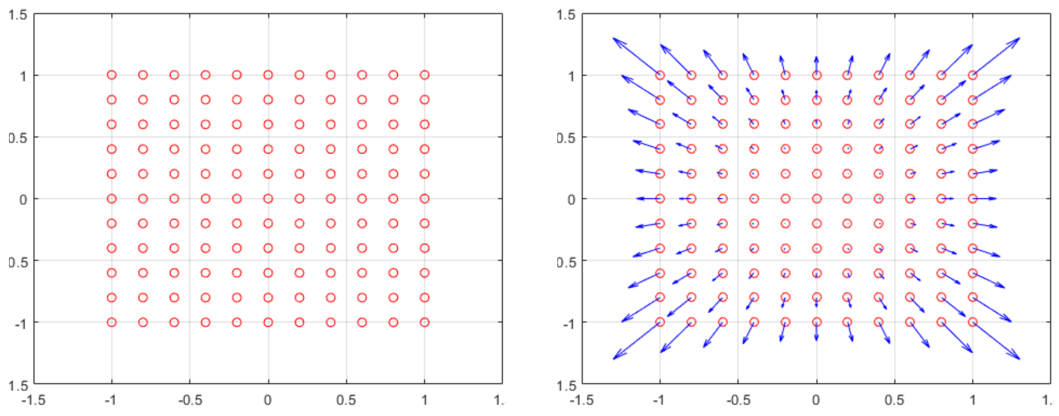

Camera Calibrator - Image
CALIBRATION
New Session
Open Session
Save Session
Add Images
FILE
Zoom In
Zoom Out
Pan
ZOOM
Default Layout
LAYOUT
Camera Model:
Standard
Fisheye
Options
CAMERA MODEL
Calibrate
CALIBRATE
Show Undistorted
Fisheye Scale
VIEW
Export Camera Parameters
EXPORT
Data Browser
Image
4: Pic_20200125144952_680918078.jpg
5: Pic_20200125145023_680949093.jpg
6: Pic_20200125145104_680990234.jpg
7: Pic_20200125145133_681019218.jpg
8: Pic_20200125145154_681039687.jpg
9: Pic_20200125145408_681174359.jpg
10: Pic_20200125145421_681187359.jpg
11: Pic_20200125145427_681193406.jpg
12: Pic_20200125145442_681208562.jpg
13: Pic_20200125145450_681216000.jpg
14: Pic_20200125145503_681229656.jpg
Detected points
Checkerboard origin
(0,0)
Y
X
2018-01-01 Monday 00:27:23

Figure 3.2. The Camera Calibrator App in Matlab

3.2 Remove distortion

Following along in the the second coloumn of the flowchart in Figure 3.1, the image must be processed as well. To model the camera as a pinhole camera requires removing the distortion of the image caused by the lens and other sources. As explained in Section 2.2, this distortion is modelled using distortion parameters obtained through the camera calibration app and is used predicts where the undistorted pixels coordinates should have been without the distortion. The distortion is removed using Matlab function ‘undistortimage’, part of the image processing toolbox. The distortion coefficients are provided to the function alongside the distorted image and it undistorts the image using the built-in distortion models which were described in the previous chapter. It is still worth looking at the process of distortion removal.

While the models operate in physical distance units, they can still be used in images with pixel coordinates, as the resolution of the image is proportional to the sensor dimensions. Figure 3.3(a) shows hypothetical pixel locations of a distorted image. The image is assumed to have Barrel distortion, so in this situation the pixels should have been generated by light further out if the lens had not refracted the light. So based on the undistorted coordinates proposed by the model using the distortion parameters, the pixels are moved to new locations, shown by Figure 3.3(b), where the blue arrows represent the shifting of the pixels to their new coordinates.



(a) (b)

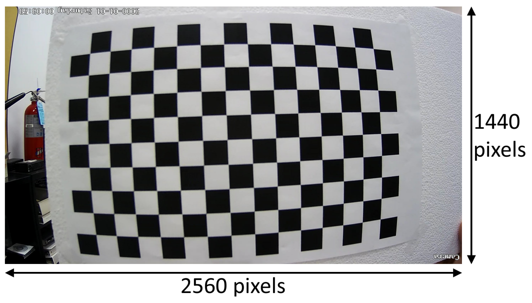
Figure 3.3. Shifting pixels to undistort image with barrel lens distortion [22]

A good example of this is an image taken in the lab of a checkerboard pattern, shown in Figure 3.4(a). The image shows the effect of lens distortion with the checkerboard pattern being warped. The way that it warps the image shows that it is barrel distortion. Thus, using the distortion parameters the image pixels are remapped to new locations to undistort the image, as shown in Figure 3.4(b). Notice how the image has black areas which are regions where there is no light information. The light rays for these parts never fell onto the sensor. Also note that the resolution of the image also changed. This is because the barrel distortion occurs when the lens of a camera increases the FOV refracting light from wider FOV inwards onto the sensor. The pixels generated would have otherwise fallen further outside as indicated after shifting the pixels and interpolating in between to fill in the he newly formed gaps. Needless to say the actual sensor does not change size and its resolution remains what it originally was. Therefore, the image must be cropped out to the original resolution, where those pixels that would fall outside the sensor are removed. This is shown in Figure 3.4(c) with the red box indicating the cropped image, which is seperately shown in Figure 3.4(d).

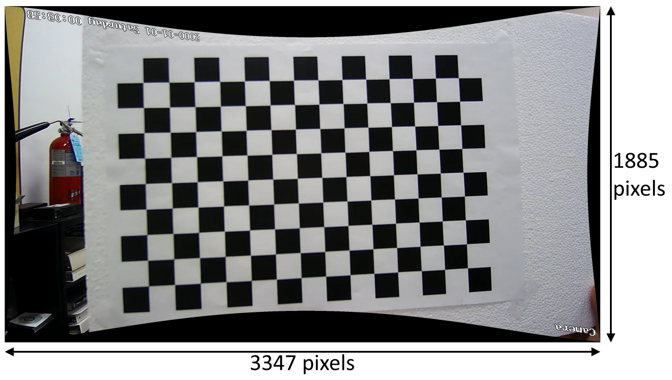
The image is then successfully undistorted and should be an accurate representation of one generated from an ideal pinhole camera. The pixel information can then be extracted from the image. Each pixel would have some u and v coordinate identifying its location in the image and RGB values identifying its color. RGB stands for Red Green Blue which are the primary colors of light. The RGB identifies the percentage of Red Green and Blue to make the color for the pixel at the location of interest.

3.3 Transform lidar point cloud to camera coordinate system

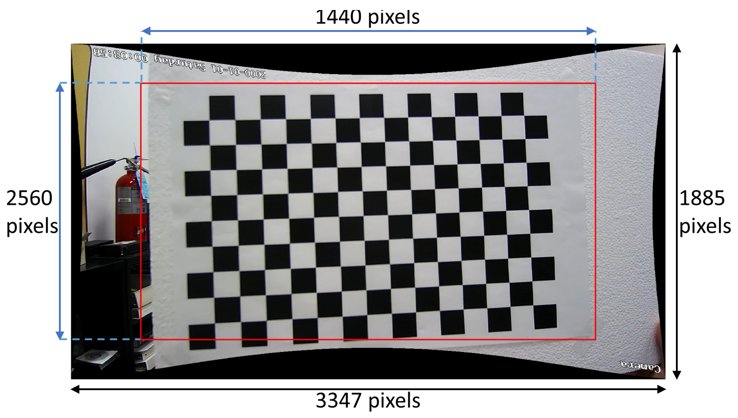
Finally, the third column in the flowchart in Figure 3.1 deals with lidar point cloud processing. Since the lidar point cloud obtained is in the lidars own coordinate system, it is necessary to transform the obtained point cloud to the camera coordinate system before being able to use both, the camera and lidar data, together. Figure 3.5 shows the coordinate frames on the sensor suite of the GNSS-R equipment. The camera and lidar can be seen in the center of the mast, while two GPS receivers are seen attached to the end of the boom on top of the mast, protruding out over the lake.



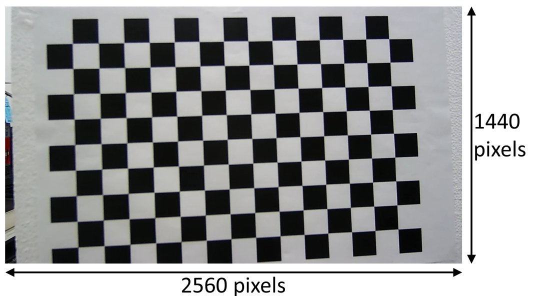
(a)



(b)



(c)



(d)

Figure 3.4. Removing barrel distortion from an image

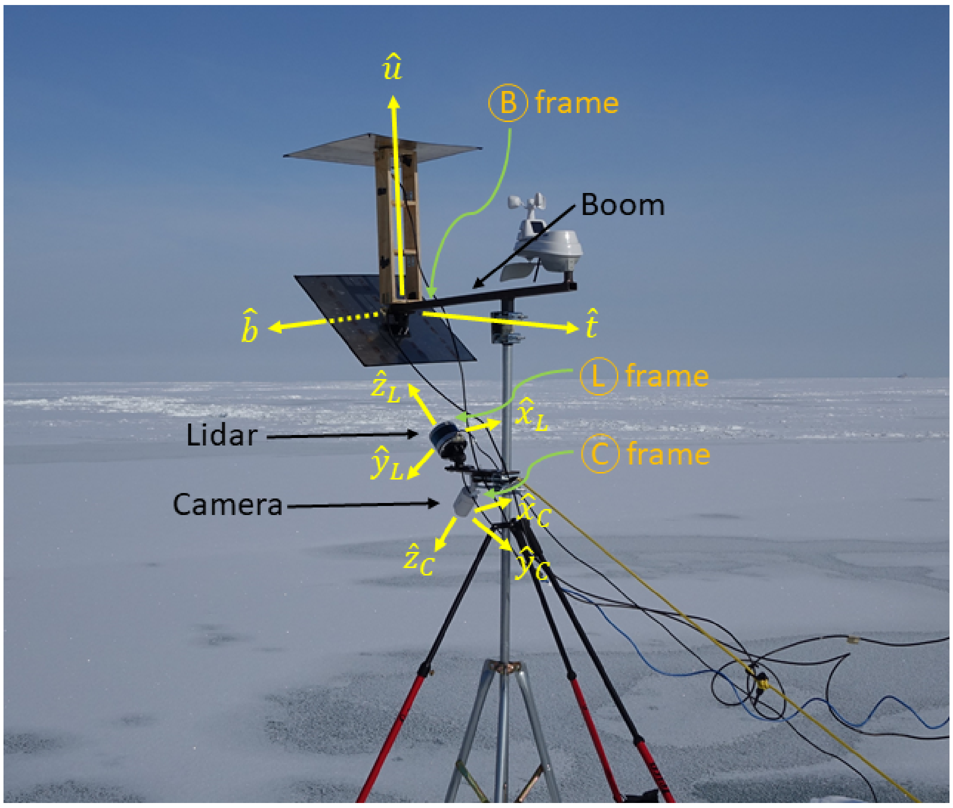


Figure 3.5. The different coordinate frames for the GNSS-R sensor suite

The boom coordinate frame Ⓑ consists of unit vectors $\hat{t}, \hat{b}, \hat{u}$, in which the $\hat{b}$ axis is along the direction of the boom, pointing horizontally out towards the lake, the $\hat{u}$ axis pointing straight up, perpendicular to the ground, and the $\hat{t}$ axis representing the transverse direction completing the right handed coordinate system. The lidar body coordinates Ⓛ consist of the $\hat{y}_L$ axis along the forward direction of the lidar, $\hat{z}_L$ the axisymmetric direction, positive away from the surface the lidar is mounted on, and $\hat{x}_L$ axis completing the right handed system. The camera coordinate frame Ⓒ, as also explained earlier, consists of the $\hat{z}_C$ axis along the optical axis of the camera, positive toward the viewing target, and the $\hat{x}_C$ axis and $\hat{y}_C$ axis defining the camera's horizontal and vertical field-of-view directions, pointing to the right and down respectively.

The lidar point cloud obtained is initially in the lidar coordinate frame Ⓛ relative to the lidar origin L_0 and must be transformed to the camera coordinate frame Ⓒ and expressed with respect to camera origin C_0. This is done through a translation and a series of rotations, as follows.

3.3.1 Translation from lidar origin L_0 to camera origin C_0.

Once the coordinate system is rotated, all that is left is translating the origin. The lidar point cloud originally has its origin at the center of the lidar, L_0. This must be translated to the origin of the camera C_0, being its optical center, as shown as point O in Figure 2.2. Since the camera is being assumed as a pinhole camera, the exact position of this optical center is assumed to be the center of front face of the camera.

The translation of the position vector $\vec{r}_C^{P/L_0}$ of point P relative to the lidar origin L_0 to the position vector $\vec{r}_C^{P/C_0}$ of point P relative to the camera origin C_0 is shown in equation 3.1, where $\vec{r}_C^{L_0/C_0}$ is the relative position vector of the lidar from the camera. This can be seen in Figure 3.6.

$$\vec{r}^{P/C_0} = \vec{r}^{P/L_0} + \vec{r}^{L_0/C_0} \tag{3.1}$$

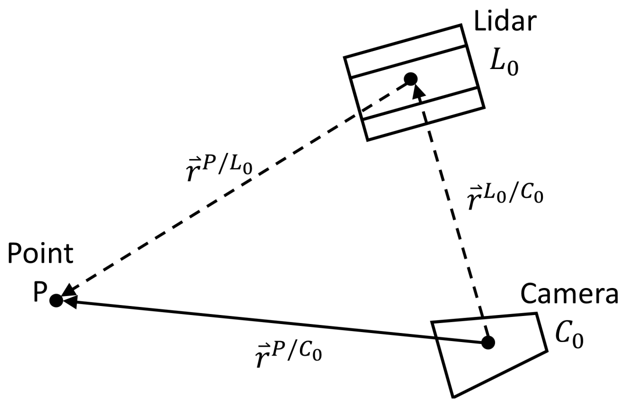


Figure 3.6. Translation of the Origin from Lidar to Camera

3.3.2 Rotation from Lidar frame Ⓛ to Boom frame Ⓑ.

The lidar has an elevation angle el_L with respect to the boom, defined as the angle between $\hat{b}$ and

$\hat{y}_L$ in Figure 3.7. For position vector $\vec{r}_L^{\,P/C_0}$ of an arbitrary point P relative to the Camera origin C_0, we rotate the $\hat{x}_L$,$\hat{y}_L$,$\hat{z}_L$ to $\hat{t}$,$\hat{b}$,$\hat{u}$ as shown in Figure 3.7 using the matrix ${}^{B}R^{L}$:

$$\vec{r}_B^{\,P/C_0} = {}^{B}R^{L}\,\vec{r}_L^{\,P/C_0} \tag{3.2}$$

$${}^{B}R^{L} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(el_L) & -\sin(el_L) \\ 0 & \sin(el_L) & \cos(el_L) \end{bmatrix} \tag{3.3}$$

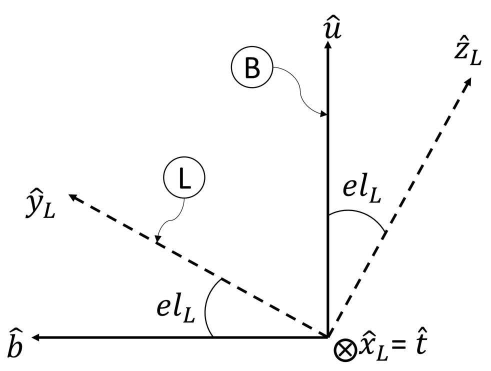


Figure 3.7. Rotation of the lidar frame L to boom frame B

3.3.3 Rotation from Boom frame Ⓑ to Camera frame Ⓒ. In general a rigid body attitude with respect to a reference coordinate system may be described by up to three Euler angles. The next sequence of rotations describes the transformation from the boom coordinate frame to any one camera's coordinate frame.

1. **Rotation from Boom frame Ⓑ to Intermediate frame Ⓐ.** Going from boom coordinate frame Ⓑ to Camera coordinate frame Ⓒ requires two rotation as the camera is rotated by an elevation angle el_C and an azimuth angle az_C,

with respect to the boom. Thus, one rotation must be made to an intermediate reference frame Ⓐ before rotating to the Camera reference frame Ⓒ. The boom reference frame Ⓑ is rotated by azimuth angle az_C to obtain the intermediate reference frame Ⓐ as shown in Figure 3.8.

The position vector $\vec{r}_B^{\,P/C_0}$ of point P relative to the camera origin C_0, is transformed from the $\hat{t}$,$\hat{b}$,$\hat{u}$ to $\hat{a}_1$,$\hat{a}_2$,$\hat{a}_3$ coordinates as shown in Figure 3.8 using the matrix ${}^AR^B$:

$$\vec{r}_A^{\,P/C_0} = {}^AR^B\,\vec{r}_B^{\,P/C_0} \tag{3.4}$$

$${}^AR^B = \begin{bmatrix} \cos(az_C) & -\sin(az_C) & 0 \\ \sin(az_C) & \cos(az_C) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

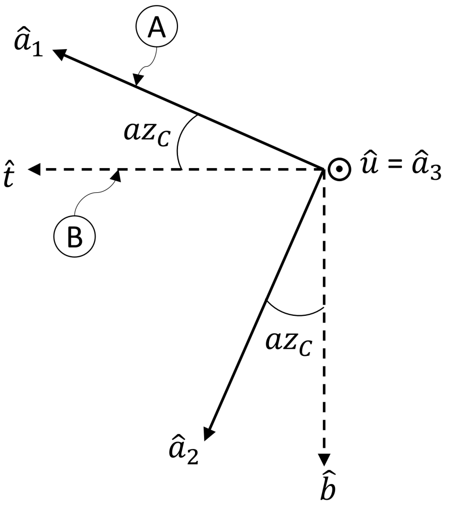


Figure 3.8. Rotation of the Boom frame B to Intermediate frame A

2. **Rotation from Intermediate frame Ⓐ to intermediate camera frame Ⓒ′.** A final rotation is required to rotate from the intermediate frame Ⓐ to camera frame Ⓒ′. This rotation is performed along the shared axis as shown in Figure 3.9 by the camera elevation angle el_C.

For the position vector $\vec{r}_A^{\,P/C_0}$ of point P relative to the camera origin C_0, we rotate the $\hat{a}_1$,$\hat{a}_2$,$\hat{a}_3$ to $\hat{x}_{C'}$,$\hat{y}_{C'}$,$\hat{z}_{C'}$, as shown in Figure 3.8 using the matrix ${}^{C'}R^A$:

$$\vec{r}_{C'}^{\,P/C_0} = {}^{C'}R^A \, \vec{r}_A^{\,P/C_0} \tag{3.6}$$

$${}^{C'}R^A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \sin(el_C) & -\cos(el_C) \\ 0 & \cos(el_C) & \sin(el_C) \end{bmatrix} \tag{3.7}$$

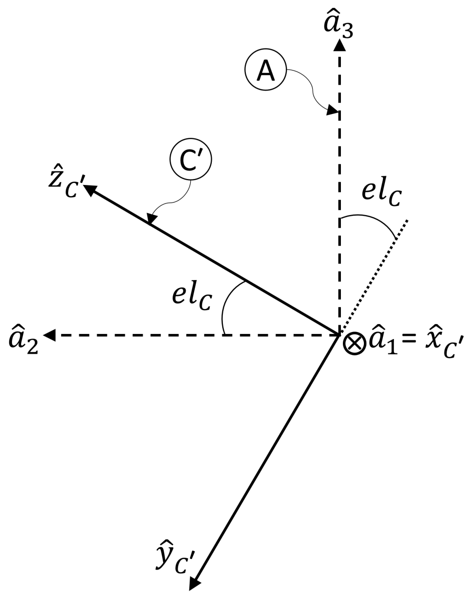


Figure 3.9. Rotation of the Intermediate frame A to Camera frame C′

3. **Rotation from intermediate frame Ⓒ′ to Camera frame Ⓒ.** From the GNSS-R hardware that was used during data collection, it was found that Camera 1 was slightly misaligned. This was determined to be a manufacturing defect of the camera mount used to attach the camera. This can be seen in Figure 3.10 where the sensor suite is seen from front, the -$\hat{b}$ axis direction. The vertical line indicates the plane created by the $\hat{b}$ and $\hat{u}$ axes and the blue

dashed line represents the optical axis. Camera 1, based on the hardware setup indicated in Appendix D, with az_C=0 and el_C=-90° should have its optical axis on the $\hat{b}$-$\hat{u}$ plane pointing in the $-\hat{u}$ direction. However, here it can be seen to be misaligned to the left. Figure 3.10(a) shows the camera, pointing down, with a -90° elevation with respect to the $\hat{b}$ axis. The optical axis should be in line with the $\hat{u}$ axis and perpendicular to the $\hat{b}$ axis. Figure 3.10(b) shows the camera pointing forward, with 0° elevation with respect to the $\hat{b}$ axis. Here the optical axis should be in line with the $\hat{b}$ axis and perpendicular to the $\hat{u}$ axis. In both cases it can be seen that the camera has some deviation. Upon close inspection the camera seems to be rotated slightly about its own $\hat{y}_C$ axis. This angular deviation of the camera about its $\hat{y}_C$ axis is designated as the misalignment angle δ.

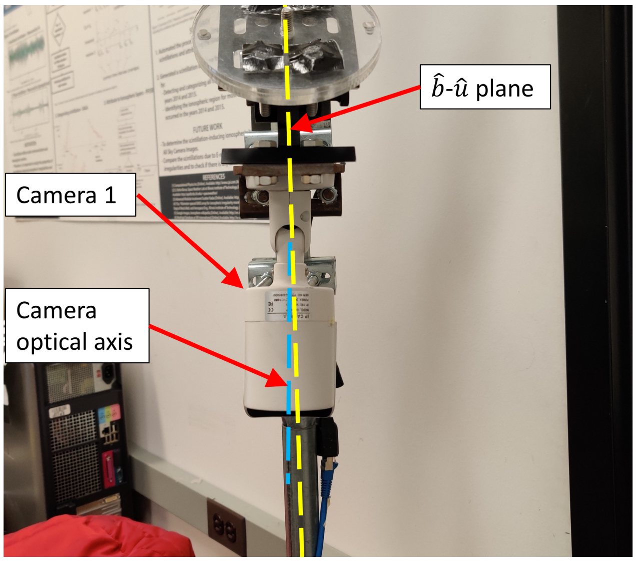
To correct for this, a final rotation is required to rotate from the intermediate camera frame $\textcircled{C'}$ to the camera frame $\textcircled{C}$. This rotation is about the $\hat{y}_{C'}$ axis by the misalignment angle δ between the $\hat{z}_C$ and the $\hat{z}_{C'}$ axes. We rotate the $\hat{x}_{C'}$,$\hat{y}_{C'}$,$\hat{z}_{C'}$ to $\hat{x}_C$,$\hat{y}_C$,$\hat{z}_C$, as shown in Figure 3.11 using the matrix ${}^{C}R^{C'}$:

$$\vec{r}_C^{\,P/C_0} = {}^{C}R^{C'}\, \vec{r}_{C'}^{\,P/C_0} \tag{3.8}$$

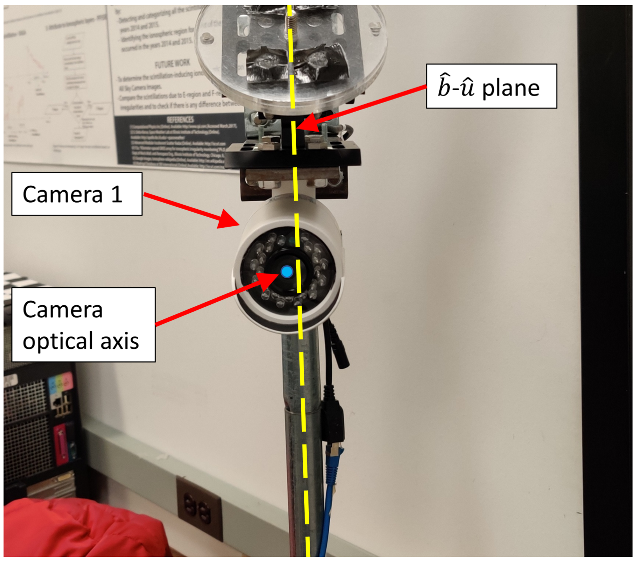
$${}^{C}R^{C'} = \begin{bmatrix} \cos(\delta) & 0 & -\sin(\delta) \\ 0 & 1 & 0 \\ \sin(\delta) & 0 & \cos(\delta) \end{bmatrix} \tag{3.9}$$

3.4 Crop lidar point cloud to camera FOV

The lidar point cloud can be merged together with the camera image at this stage. It is important to note that the lidar FOV may not be the same as the camera FOV. The points that are not visible to the camera can be cropped out. Figure 3.12 is an illustration of the camera FOV overlapping with lidar returns. The points (gray)



(a)



(b)

Figure 3.10. Camera 1 misalignment about its $\hat{y}_C$ axis

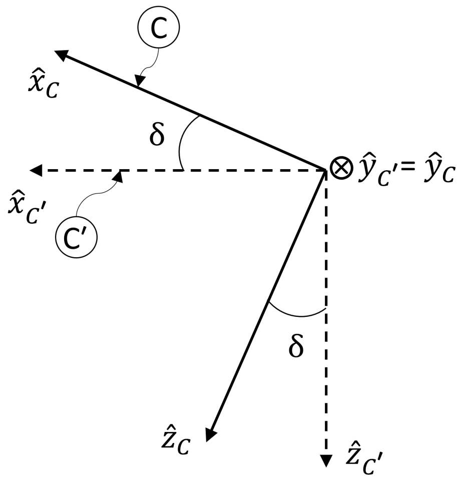


Figure 3.11. Rotation of the Camera frame C′ to Camera frame C

outside the FOV of the camera are cropped out leaving only the points inside the FOV (blue).

To determine which points lay in the FOV, the coordinates X_C, Y_C, Z_C of the point cloud points are used to obtain their azimuth and elevation angles with respect to the camera optical axis. The azimuth angle with respect to the $\hat{z}_C$ axis is determined using $\tan^{-1}\frac{X_C}{Z_C}$ and the elevation angle with respect to the $\hat{z}_C$ axis is determined using $\tan^{-1}\frac{Y_C}{Z_C}$. Points that do not satisfy the conditions provided in Equations 3.10 and 3.11 are cropped out. θ_H and θ_V are the horizontal FOV and vertical FOV angles, respectively. The θ_H and θ_V are bisected by the $\hat{z}_C$ axis and so the angle from the axis to the maximum FOV of the camera can be calculated as half the FOV for both horizontal and vertical cases. In both cases the magnitude of the azimuth and elevation angles must less than or equal to half the horizontal FOV or vertical FOV respectively, so the absolute values are compared. These conditions are shown visually in Figure 3.13(a) and Figure 3.13(b).

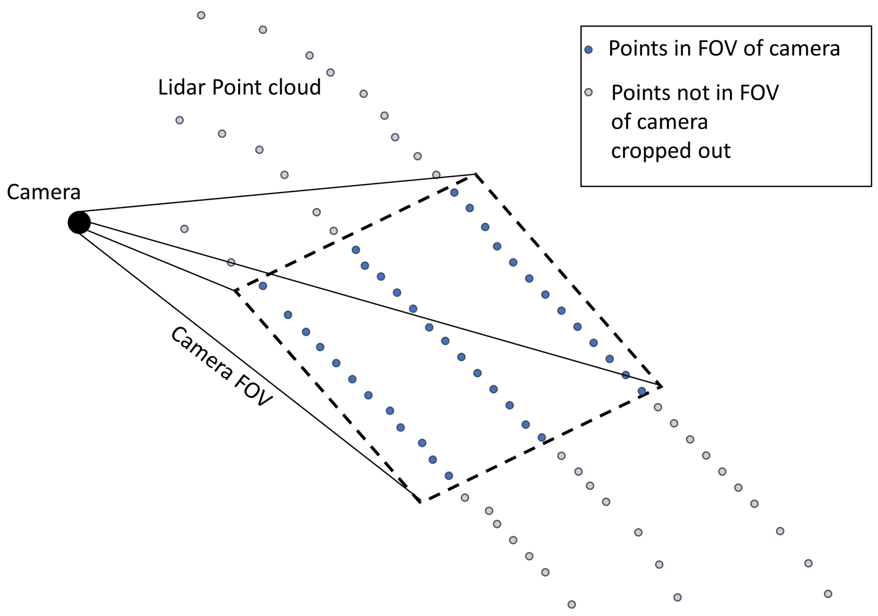

Lidar Point cloud
Camera
Camera FOV
● Points in FOV of camera
○ Points not in FOV of camera cropped out

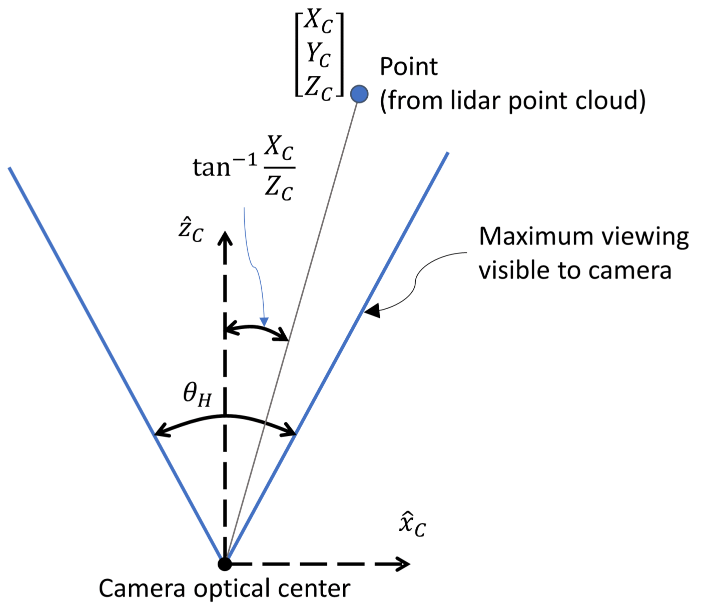
Figure 3.12. Crop lidar point cloud based on camera FOV

$$\left|\tan^{-1}\frac{X_C}{Z_C}\right| \leq \left|\frac{\theta_H}{2}\right| \tag{3.10}$$

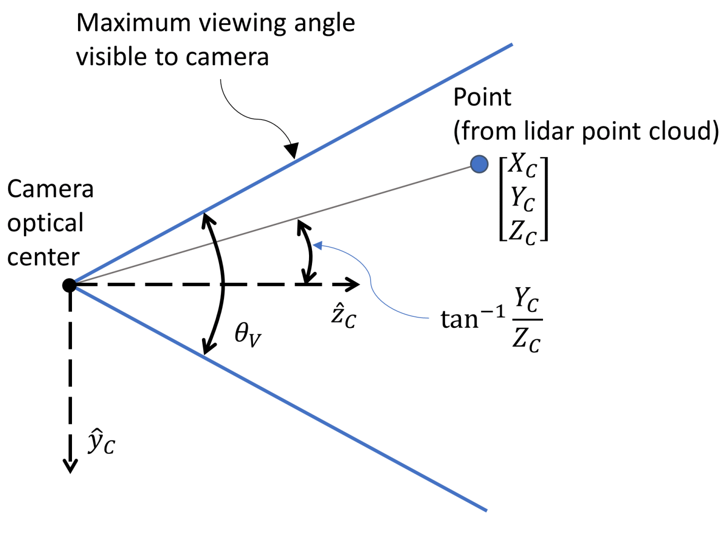
$$\left|\tan^{-1}\frac{Y_C}{Z_C}\right| \leq \left|\frac{\theta_V}{2}\right| \tag{3.11}$$

3.5 Overlay lidar point cloud onto image plane

Once the lidar point cloud is cropped, the points must be overlaid onto the image plane. The points must be transformed from their camera coordinates into pixel coordinates using forward projection. This requires using Equation 2.16 to perform the projection. However, testing the forward projection in the lab, using a set of boxes, resulted in a significant amount of error, as shown in Figure 4.5(a), where the projected lidar point cloud is overlaid on top of the image in pixel coordinates. The color bar shows the Z_C values of those points. The projected lidar points do not align well with the features in the image which is apparent when looking at the edges of the boxes. The edges of both boxes in the image do not line up with the edges of



(a)



(b)

Figure 3.13. Determining which lidar points are within the camera FOV

the same boxes in the lidar point cloud. This same issue can be seen in the directly behind the larger box on the right. The potential reasons for this error could be:

1. **Lidar translation to camera origin.** There is some uncertainty in the exact location of the lidar origin and more specifically camera origin. The camera origin is assumed to be in the center front face of the camera as it would be in case of a pinhole camera. The actual camera origin, the optical center, would be different due to the camera optics.

2. **Inaccurate K calibration matrix.** If the camera parameters making up the matrix were not good estimates, then the matrix K would not produce the correct projections.

3. **Inaccurate removal of image distortion.** The distortion coefficients obtained by the camera calibrator app may not have been accurate. This may cause the undistorted image to retain some of the warping from the distortion sources and not allow it to align well with the projected lidar point cloud.

Thus, a method using ratios was devised to approximate the projection of the points into the image plane. This method then approximates the projected pixel position using the ratio of the azimuth and elevation angles of the lidar points, being $\tan^{-1}\frac{X_C}{Z_C}$ and $\tan^{-1}\frac{Y_C}{Z_C}$, to the horizontal FOV and vertical FOV of the camera respectively. Going into the details, for the transformation of the lidar points to their projected u_L coordinate along the $\hat{u}$ direction in the image, the ratio between the azimuth angle of the point and half the horizontal FOV angle is taken. The FOV angle is divided by half as the ratio is from the $\hat{z}_C$ axis to the edge of the FOV. This ratio is used on the horizontal resolution Res_H, basically the number of pixels along the $\hat{u}$ direction, to determine where it would lie, as shown in Figure 3.14 and Equation 3.12. Similarly, for the transformation of the lidar points to their projected v_L

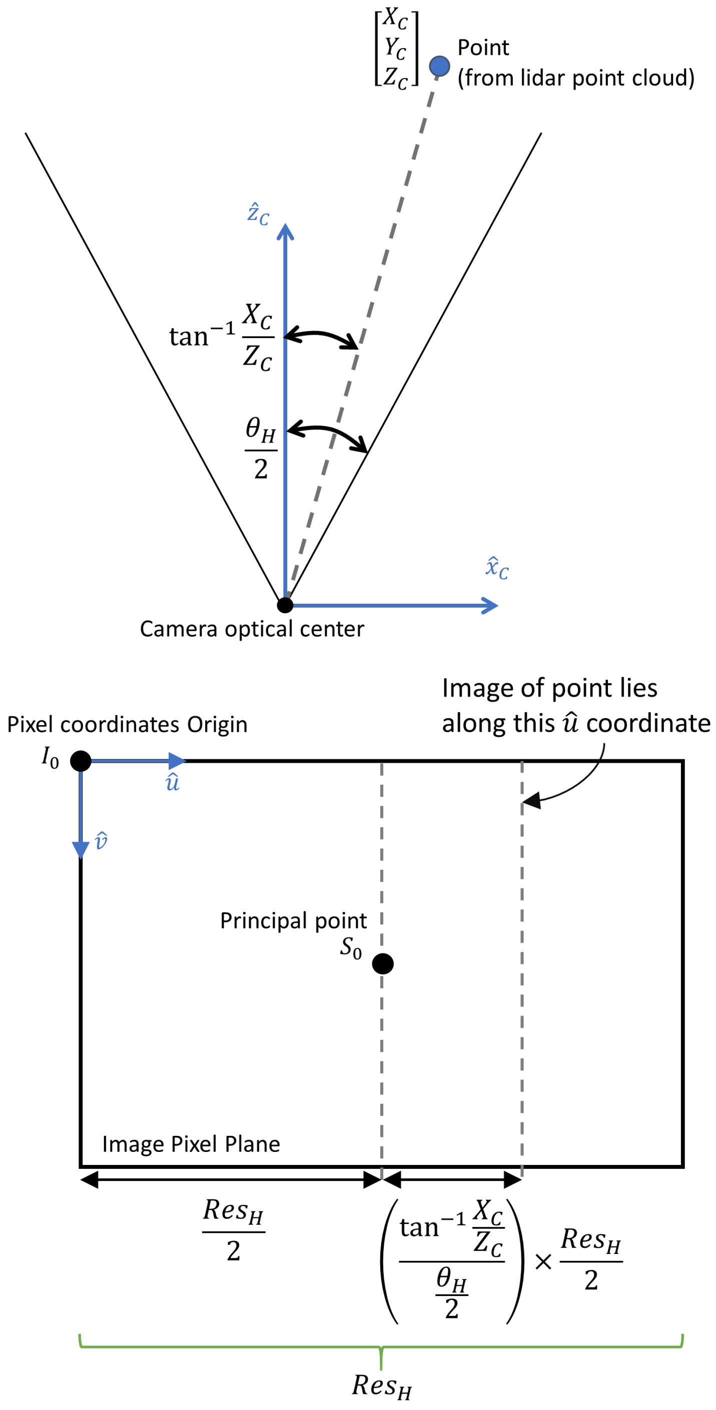
coordinate along the $\hat{v}$ direction in the image, the ratio between the elevation angle of the point and half the horizontal FOV angle is taken. The ratio determines where the pixel will lie based on the vertical resolution Res_V, the number of pixels along the $\hat{v}$ direction, shown in Figure 3.15 and Equation 3.13. The closer the point is to the maximum FOV of the camera, the closer this point will be to the edges of the image and vice versa.

The results of this new method can be seen to be better, in Figure 4.5(b). The lidar point cloud overlaid onto the image seems to align better with the features in the image. The points closer to the camera can be identified by their Z_C values, indicated by the colorbar, can be seen to overlap the image of the box well. The edges of the box are identified in the point cloud by the sudden change of color where the Z_C values change from being closer to the camera, i.e. on the box surface, to being further away, i.e on objects behind the box, right at the edge of the image of the box. The edge of the box in the point cloud coincides with the edge of the box in the image it is overlaid onto.

$$u_L = \left(\frac{\tan^{-1} \frac{X_C}{Z_C}}{\frac{\theta_H}{2}} \times \frac{Res_H}{2} \right) + \frac{Res_H}{2} \tag{3.12}$$

$$v_L = \left(\frac{\tan^{-1} \frac{Y_C}{Z_C}}{\frac{\theta_V}{2}} \times \frac{Res_V}{2} \right) + \frac{Res_V}{2} \tag{3.13}$$

This method is an alternate to the forward projection and only provides an approximation of the pixel location. That is because the assumption here is that the angle of light rays from the optical axis is directly proportional to the radial distance of the projected image from the principal point, on the sensor plane. This would only be true if the sensor plane were curved with a radius of f about the optical center. However, since the sensor plane is actually flat, the actual radial distance of the projected image from the principal point is not linearly related to the angle



$\begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix}$
Point
(from lidar point cloud)
$\hat{z}_C$
$\tan^{-1}\frac{X_C}{Z_C}$
$\frac{\theta_H}{2}$
$\hat{x}_C$
Camera optical center
Pixel coordinates Origin
I_0
$\hat{u}$
$\hat{v}$
Image of point lies along this $\hat{u}$ coordinate
Principal point
S_0
Image Pixel Plane
$\frac{Res_H}{2}$
$\left(\frac{\tan^{-1}\frac{X_C}{Z_C}}{\frac{\theta_H}{2}}\right) \times \frac{Res_H}{2}$
Res_H

Figure 3.14. Determining horizontal pixel coordinate through alternate method to forward project lidar points into image

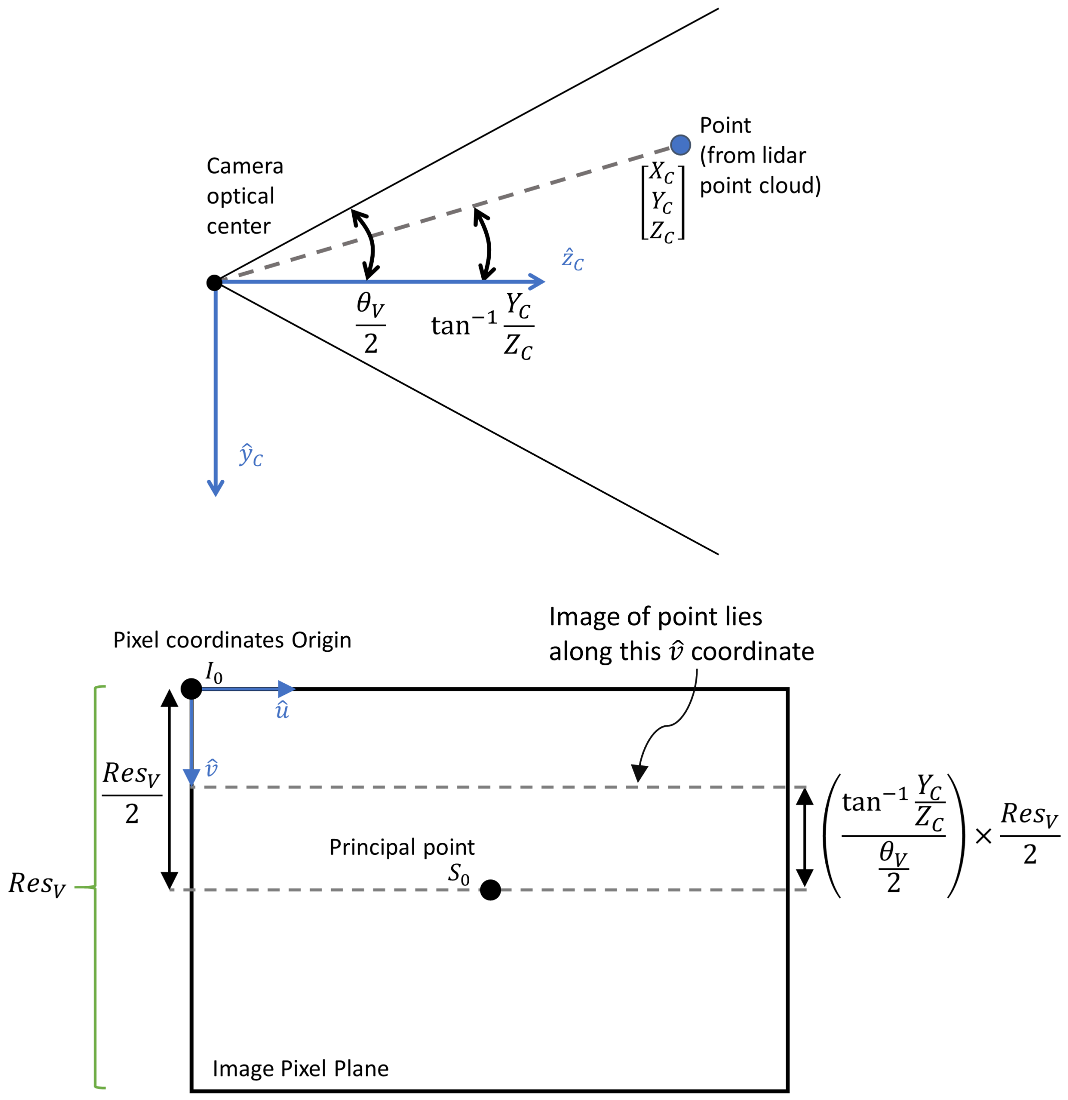

Camera optical center
Point (from lidar point cloud)
$\begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix}$
$\hat{z}_C$
$\hat{y}_C$
$\frac{\theta_V}{2}$
$\tan^{-1}\frac{Y_C}{Z_C}$
Pixel coordinates Origin
I_0
$\hat{u}$
$\hat{v}$
Image of point lies along this $\hat{v}$ coordinate
$\frac{Res_V}{2}$
Res_V
Principal point
S_0
$\left(\frac{\tan^{-1}\frac{Y_C}{Z_C}}{\frac{\theta_V}{2}}\right) \times \frac{Res_V}{2}$
Image Pixel Plane

Figure 3.15. Determining vertical pixel coordinate through alternate method to forward project lidar points into image

of light rays from the optical axis. Because of this assumption, this method would have more error the further away from the principal point the pixel needed to be calculated. Hence the pixels projected near the edges of the image would have the most error.

This seems to work better likely because it circumvents the need for the K matrix and any error it might have. In the rest of this work, this implementation is used instead of the K matrix for the forward projection. If the source of error is from the camera calibration then this method estimates the pixel locations with out the need to use the camera properties obtained from calibration.

3.6 Interpolate the overlaid points in the Image

The lidar points projected onto the image plane do not map one-for-one onto every pixel, so not every pixel in the image has a Z_C value. To cover the whole image the lidar points on the image must be interpolated over the entire resolution of the image. This interpolation must be performed over a grid with the same resolution as the image, allowing for each pixel to obtain a Z_C value.

For this the Matlab function ‘scatteredInterpolant’ is used, which interpolates over irregular data. The function generates an interpolant for the lidar points which is evaluated at various query points. The function is set to perform linear interpolation about the grid, but is also set to not extrapolate the data.

In this case, the projected lidar points do not lie on a neat grid pattern in the image and are ”scattered” about the image, with no specific order. In our case the query points are points on a grid with the same size as the image resolution, to produce interpolated values. This selection not to extrapolate means that the function will interpolate between the lidar points, but if the interpolation grid extends over to part of the image with no lidar coverage around, then there will be no interpolation performed here. Those pixels are simply left without a Z_C value. The reason for this choice was the extrapolation was not considered accurate enough and so disabled entirely.

3.7 Backward project Image

At this point all the necessary information is available to perform backward projection on each pixel. The camera calibration matrix K is built using the camera parameters obtained from the camera calibrator app. The undistorted image is used to obtain the pixels and their corresponding pixel coordinates. And after the interpolation, of the overlaid lidar points in the image plane, each pixel has a Z_C value. The backward projection can then be performed using Equation 2.19. Its important to note that this is a pixel-by-pixel process, where the backward projection is performed on each individual pixel to obtain its 3D camera coordinates X_C,Y_C,Z_C in the camera coordinate frame. Each pixel can then be plotted with these coordinates with its respective color (RGB) value. This produces a mapping of the pixels in camera coordinates.

3.8 Rotation of pixel point cloud to ENU coordinates

Once the pixel point cloud is generated, the environment has successfully been reconstructed, however, it is not of much use in camera coordinate frame. Thus, the point cloud must be rotated to a different coordinate frame. For the other sensors and equipment involved in this GNSS-R project, the Boom coordinate system $\hat{t}, \hat{b}, \hat{u}$ is an important one, as it is used as the main body-fixed coordinate system for the entire assembly. Moreover, when GPS satellites or other objects/systems, external to the sensor suite, need to be considered, the East-North-Up (ENU) coordinate system is far more useful. The ENU coordinate system consists of the $\hat{e}$ axis pointing towards

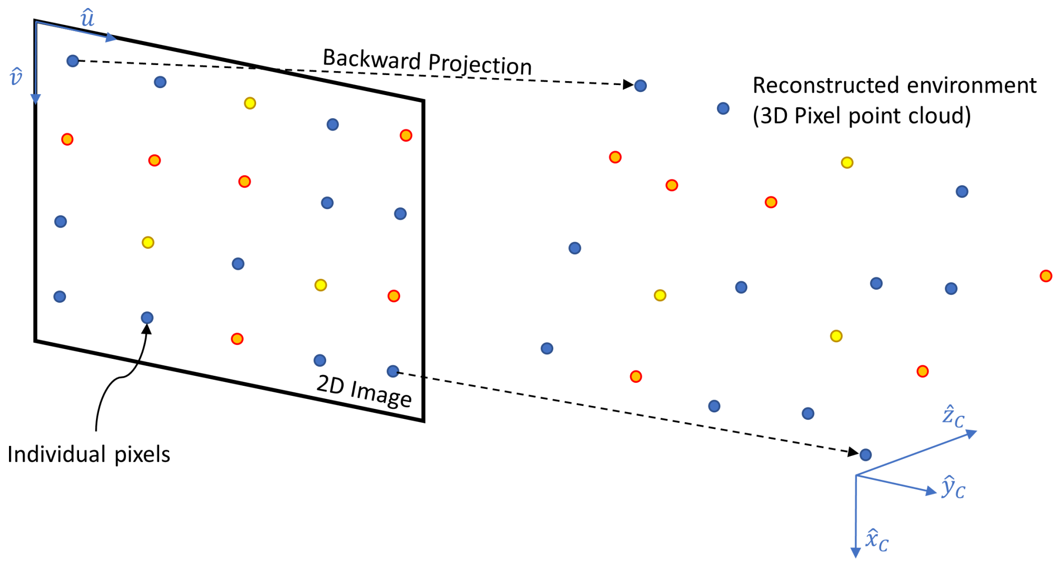


Figure 3.16. Backward projection of 2D image producing a 3D pixel point cloud

East, $\hat{n}$ axis pointing towards North and $\hat{u}$ pointing straight up.

3.8.1 Rotation from Camera frame Ⓒ to Boom frame Ⓑ. This involves the rotation matrices, devised earlier at Equations 3.7 and 3.7, between the Boom frame Ⓑ and the Camera frame Ⓒ. The position vector $\vec{r}_B^{\,P/C_0}$ of point P (arbitrary point from the pixel point cloud) relative to the camera origin C_0, is transformed from the $\hat{x}_C$,$\hat{y}_C$,$\hat{z}_C$ to $\hat{t}$,$\hat{b}$,$\hat{u}$.

$$\begin{aligned}
\vec{r}_B^{\,P/C_0} &= {}^{B}R^{C}\ \vec{r}_C^{\,P/C_0} && (3.14)\\
&= [{}^{C}R^{B}]^{-1}\ \vec{r}_C^{\,P/C_0} && (3.15)\\
&= [{}^{C}R^{C'}\ {}^{C'}R^{A}\ {}^{A}R^{B}]^{-1}\ \vec{r}_C^{\,P/C_0} && (3.16)\\
&= [{}^{A}R^{B}]^{-1}\ [{}^{C'}R^{A}]^{-1}\ [{}^{C}R^{C'}]^{-1}\ \vec{r}_C^{\,P/C_0} && (3.17)\\
&= [{}^{A}R^{B}]^{T}\ [{}^{C'}R^{A}]^{T}\ [{}^{C}R^{C'}]^{T}\ \vec{r}_C^{\,P/C_0} && (3.18)
\end{aligned}$$

Where ${}^{A}R^{B}$ was defined at Equation 3.5, ${}^{C'}R^{A}$ was defined at Equation 3.7 and ${}^{C}R^{C'}$

was defined at Equation 3.9.

3.8.2 Rotation from Boom frame Ⓑ to ENU frame Ⓔ.

Rotating from the boom coordinate frame Ⓑ to the ENU coordinate frame Ⓔ requires rotation by heading angle μ, which is measured as the angle of the boom $\hat{b}$ with respect to $\hat{n}$ and defined as positive east of north. The rotation is shown in Figure 3.17. Both the Boom and ENU frames share the same $\hat{u}$ axis about which this rotation is performed.

The position vector $\vec{r}_B^{\,P/C_0}$ of point P relative to the camera origin C_0, is transformed from the $\hat{t}$,$\hat{b}$,$\hat{u}$ to $\hat{e}$,$\hat{n}$,$\hat{u}$ coordinates as shown in Figure 3.17 using the matrix ${}^{E}R^{B}$.

$$\vec{r}_E^{\,P/C_0} = {}^{E}R^{B}\,\vec{r}_B^{\,P/C_0} \tag{3.19}$$

$${}^{E}R^{B} = \begin{bmatrix} \cos(\mu) & \sin(\mu) & 0 \\ -\sin(\mu) & \cos(\mu) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.20}$$

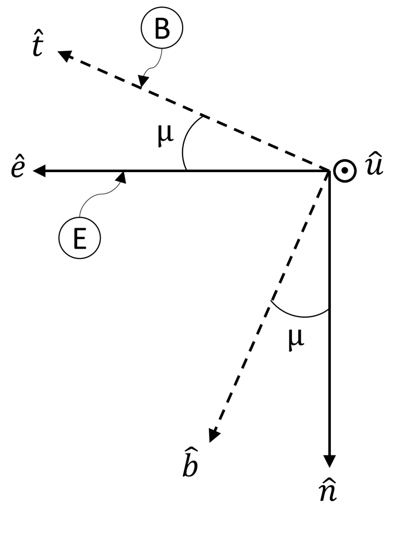


Figure 3.17. Rotation of the Boom frame B to ENU frame E

3.9 Translation of pixel point cloud to other origin

The pixel point cloud is also generated with the origin at the camera. The eventual goal of this work is to provide a truth reference against which to compare the GPS processing results, which means the information generated by this processing must be placed in the frame of reference. To ensure this, the pixel point cloud is translated to the same origin as used by the GPS results. Usually this is the reflected signal antenna, which is the one collecting the reflected GPS signal off the lake surface. The origin of the reflected antenna is An_0. This translation will not be shown in this work, but is provided here for completeness.

$$\vec{r}^{P/An_0} = \vec{r}^{P/C_0} + \vec{r}^{C_0/An_0} \tag{3.21}$$

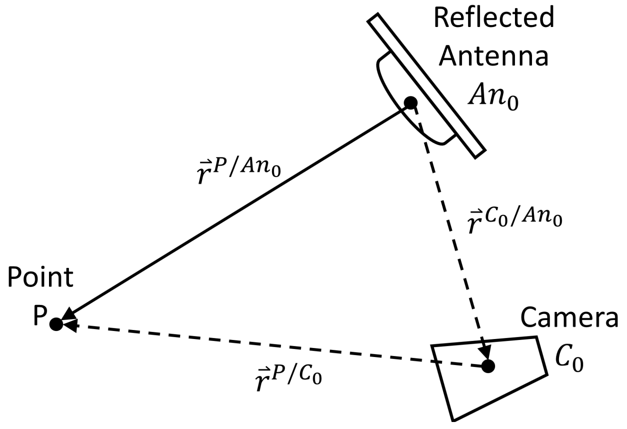


Figure 3.18. Translation of the origin from camera to reflected antenna

CHAPTER 4
TESTING BACKWARD PROJECTION IN LAB

Before the backward projection method was used on data obtained for the GNSS-R project, it was tested experimentally in the lab. Two types of test were conducted. The camera used for these tests was the GW 5037 IP, with more details on the camera in Appendix A, which was calibrated beforehand with all of the camera parameters already obtained. The purpose of the first test was to backward project individual pixels with the camera alone. This test was conducted in two parts with a checkerboard pattern which was was mounted on the wall with the camera facing it perpendicularly for the first part and then laid down on a table with the camera facing it at an angle for the second part. The method and results of this test are described in Section 4.1.

The purpose of the second test was to demonstrate the lidar point cloud overlay and full image backward projection. For this test, two cardboard boxes were placed on the floor in the FOV of the camera and the scanning FOV of the lidar. The method and results of this test are described in Section 4.2.

4.1 Camera-only test: backward projecting individual pixels

4.1.1 Checkerboard pattern perpendicular to optical axis. In this test, backward projection was performed on selected single pixel target points. The backward projection estimates of X_C, Y_C are compared to manual measurements of the target points in physical space.

The camera was pointed perpendicularly at a checkerboard pattern, i.e., the optical axis was normal to the checkerboard paper mounted on the wall. The physical checkerboard was manually marked with four target points of interest. The image of the pattern is seen in Figure 4.1, where four points are identified alongside the principal point. Since the camera optical axis $\hat{z}_C$ was normal to the wall, all points on the checkerboard pattern had the same Z_C coordinate, which was the distance from the lens of the camera to the principal point. The distance Z_C to the wall, and thus to the checkerboard pattern, was measured, using measuring tape, to be $Z_C = 1.06$ m. The pixel coordinates u, v of each of the target points were identified in the image and used to perform backward projection to obtain estimated camera coordinates $\hat{X}_C, \hat{Y}_C$ of each of the target points.

The estimated $\hat{X}_C, \hat{Y}_C$ were compared against "true" camera coordinates X_C, Y_C of the points. The true camera coordinates were obtained by manual measurement. The fact that the checkerboard pattern has squares of equal size, 38 mm on a side, means it can be used as a measuring grid. Table 4.1 lists the results along with the difference between the measured "true" and projection "estimated" camera coordinates for the points. The results show that the differences were all within 5 mm, corresponding to less than 5% of the "true" distance. Based on this, the backward projection of the points is judged to be adequately accurate.

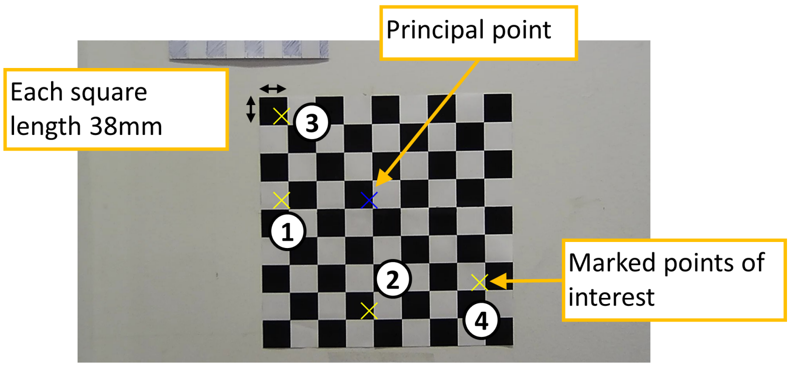


Figure 4.1. Checkerboard pattern perpendicular to optical axis

Table 4.1. Checkerboard perpendicular to optical axis test

Point	Pixels		Camera coordinates (mm)			Difference (mm)
				Measured "truth"	Projected "estimated"	
1	u	954	X_C	-114	-119	5
	v	612	Y_C	0	0	0
2	u	1171	X_C	0	-0	0
	v	904	Y_C	-152	-149	3
3	u	952	X_C	-114	-119	5
	v	399	Y_C	114	113	1
4	u	1462	X_C	152	150	2
	v	828	Y_C	-114	-111	3

4.1.2 Checkerboard pattern oblique to optical axis. The next step was test backward projection when the surface is not perpendicular to the optical axis. For this the camera was set at a -45° angle of elevation, pointed at a table with a checkerboard pattern. The Z_C values for each of the points marked on the pattern would be different since the checkerboard pattern is not perpendicular to the optical axis. The coordinates of the points were measured using measuring tape in Boom coordinates and then rotated to camera coordinates using Equations 3.4 and 3.6, where the el_c = -45° and the az_c = 0°.

Similar to the previous test, four target points were selected and manually marked on the checkerboard, to be projected from the image. Figure 4.2 shows the image of the pattern, by the camera in this test, with the results being plotted in Table 4.2. The difference between the measured and calculated camera coordinates for the points are again fairly small with the largest being about 8mm which is about 7% of the measured "truth" distance.

For both cases of perpendicular and obliques checkerboard patterns, the difference calculated were relatively low. Some of that error can also be chalked up to

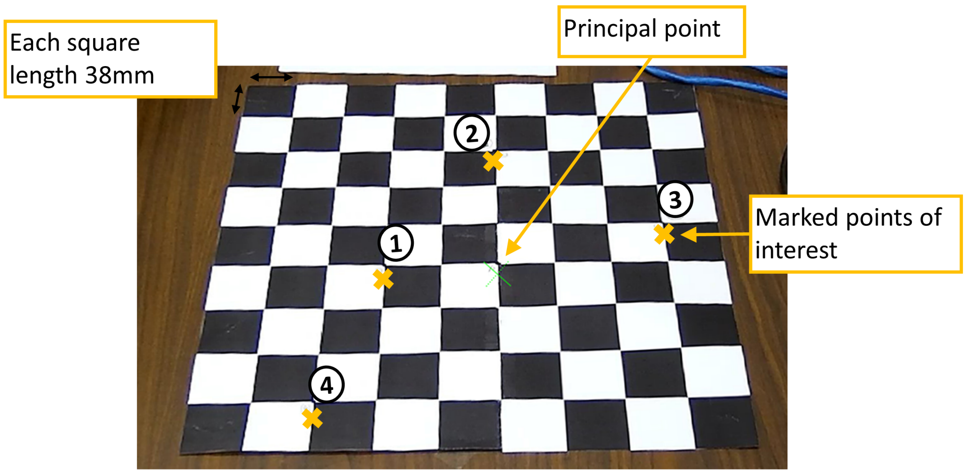


Figure 4.2. Checkerboard pattern oblique to optical axis

human error in the physical measurements as a measuring tape was used. However, the values show that this method is at least viable in projecting the pixels out to their camera coordinates at an acceptable level of accuracy.

4.2 Camera-lidar test: backward projecting boxes on floor

The next test projects each pixel in the image out instead of just one. The sensors were assembled in the configuration of Setup 1, as explained in Appendix D. The camera was calibrated as described in Section 3.1 and camera parameters saved. Since the camera parameters don't change, the same parameters obtained can be used with all future data from a given camera. Two cardboard boxes were placed on the floor. The white paper taped to the floor between the boxes has no purpose and can be ignored.

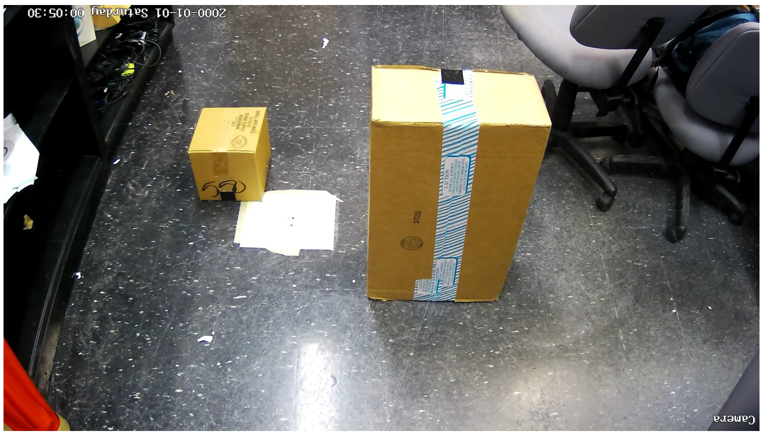
An image was taken with the camera, shown in Figure 4.3(a). Using the distortion coefficients and the built-in distortion removal function in Matlab as explained in Section 3.2, the image was undistorted to obtain image shown in Figure 4.3(b).

Table 4.2. Checkerboard oblique to optical axis test

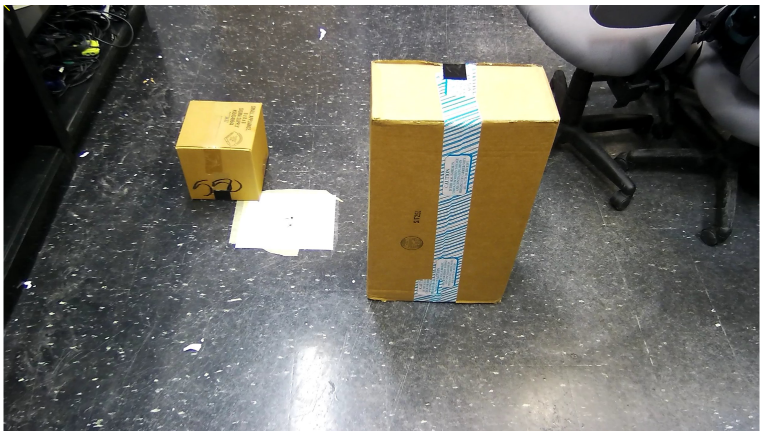
Point	Pixels		Camera coordinates (mm)			Difference (mm)
				Measured "truth"	Projected "estimated"	
1	u	1119	X_C	-78	-78	0
	v	712	Y_C	0	-3	3
2	u	1303	X_C	0	4	4
	v	538	Y_C	-81	-87	6
3	u	1561	X_C	114	122	8
	v	670	Y_C	-27	-22	5
4	u	993	X_C	-114	-122	8
	v	924	Y_C	80	83	3

Lidar data are collected at the same time as the camera image. The lidar point cloud was subsequently translated and rotated to the camera coordinate frame, producing the point cloud plot shown in Figure 4.4(a). The axes are the camera coordinates in meters and the blue dots are lidar return points. They reason they look like lines is because the lidar scans its surroundings at different vertical angles, so each line represents a lidar laser sweep. The area sensed by the lidar, as shown through the point cloud, includes not only the two boxes seen, at about where $Z_C = 2$m and around where $X_C = 0$, but also the surrounding objects such as a shelf in the positive $\hat{z}_C$ and negative $\hat{x}_C$ quadrant, chairs and desk in the positive $\hat{z}_C$ and positive $\hat{x}_C$ quadrant, and even part of the ceiling seen in the negative $\hat{z}_C$ direction. The point cloud was then cropped to the FOV of the camera as shown in Figure 4.4(b). Here the point cloud can only see the two boxes in the middle, with part of the shelf on the left and part of the chairs and desk on the right side. One thing to note is the blank pointless area behind the boxes. This is because the lidar can see the front of the boxes but has no information of the region behind it.

This lidar point cloud was then overlaid onto the camera image. As explained



(a) Distorted image with boxes

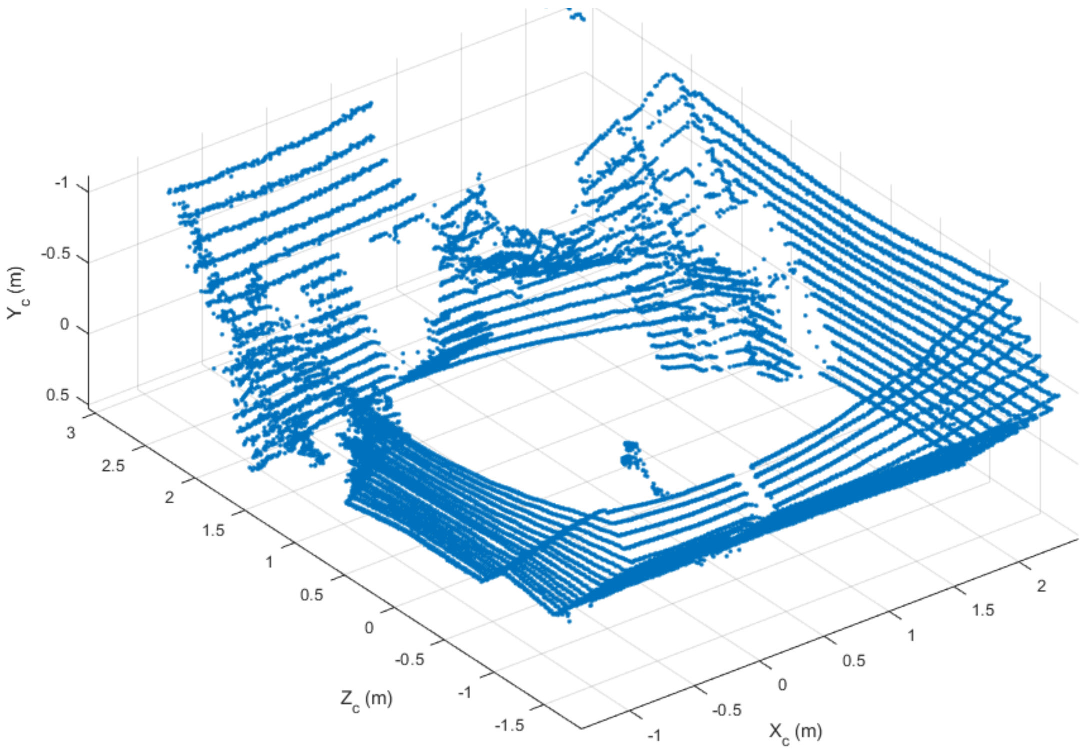


(b) Undistorted image with boxes

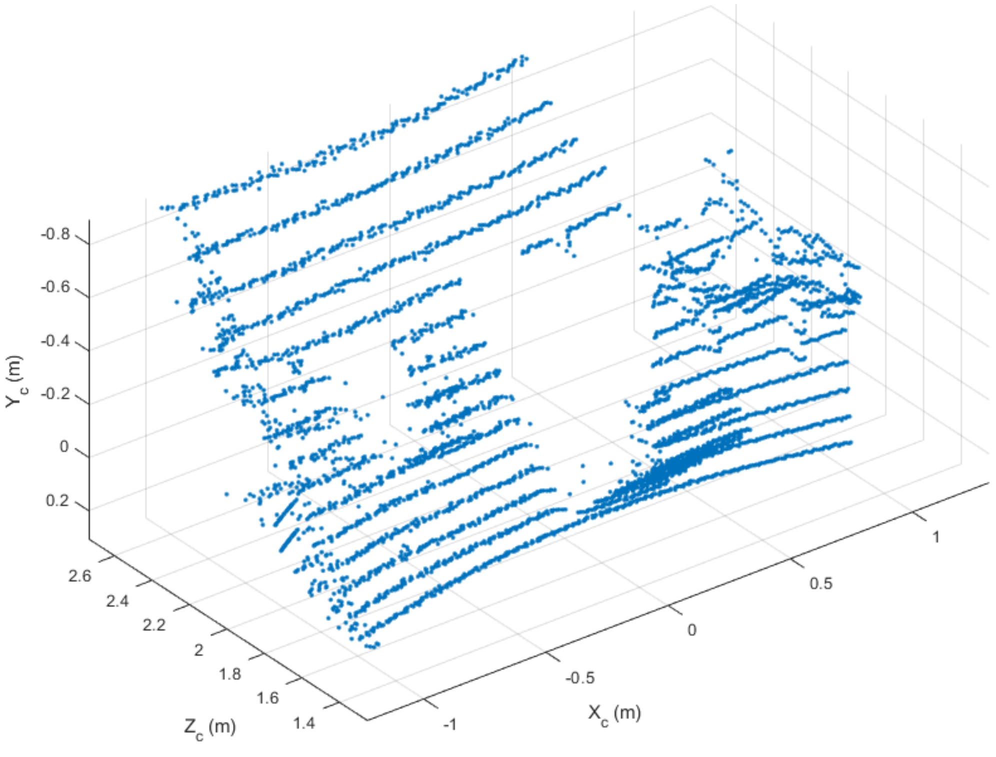
Figure 4.3. Image for lab test with boxes

in Section 3.5, the standard forward projection was performed to overlay the point cloud in Figure 4.4(b) onto the camera image 4.3(b). The results can be seen in Figure 4.5(a). The image from the camera is plotted along its pixel coordinates on the axes, which is overlaid with the colored points of the lidar point cloud. The colorbar gives information of the Z_C value of the lidar points in meters.

Examining the lidar points on top of the larger box, a problem can be identified immediately. Lidar points from behind the box (orange) are overlaid on top of the box. The physical points in space associated with those lidar returns are behind the



(a) lidar point cloud



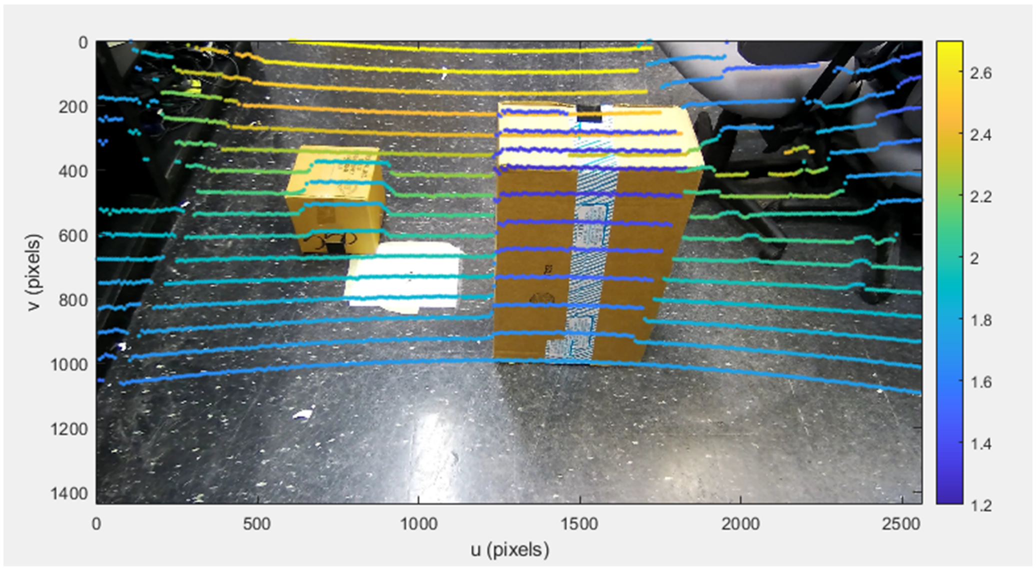
(b) Cropped lidar point cloud

Figure 4.4. Lidar point cloud for lab test with boxes

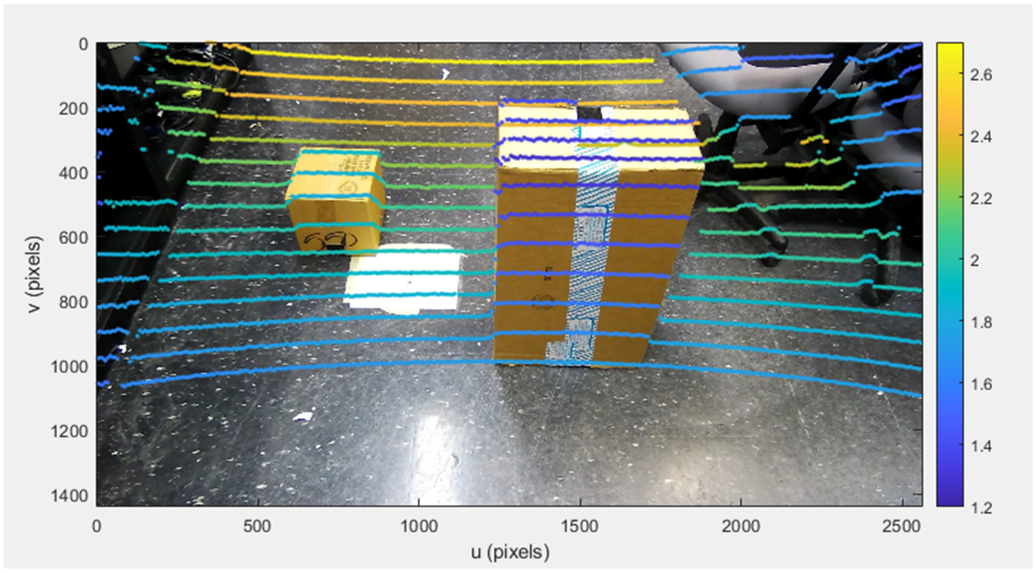
box but the reason for their visibility is because of the sensors' placement. The lidar can see objects from a different perspective than the camera, and so when translating the origin of the point cloud, the features visible to one sensor may not be visible to the other. This results in either blank spaces or, as seen here, overlapping of lidar points onto features. This is affected more by how high the protrusion is from the ground and how close to the sensors its located. Considering the lake surface, this was not considered a major problem as the lake surface is pretty flat with barely any protrusions high enough or close enough to cause this problem.

The lidar points in Figure 4.5(a) also do not align well with the features in the image. The reasoning for this was discussed more in Section 3.5. The alternative ratio method was then devised to approximate the forward projection. The results of the ratio method can be seen in Figure 4.5(b), where the lidar points and image align much better. Therefore, the ratio method was used for future data sets, including the field data campaigns, instead of the forward projection.

The lidar points in the image plane do not cover every pixel as can be seen in Figure 4.5(b) and thus, were interpolated to cover the image, as seen in Figure 4.6. This figure is similar to the image overlaid with the lidar point cloud, where the axes are the pixels in the u and v directions. The colorbar shows the Z_C value for the interpolated points in meters. These points now cover majority of the image except for one corner in the top left and the bottom which should go down to 1440 pixels according to the image size which the interpolation was performed to. The reason is that it would need to extrapolate the data in those regions due to lack of lidar point cloud coverage there. But due to bad extrapolation results which were very inaccurate, the extrapolation was not performed. These sections will be automatically omitted when forward projecting the pixels. Another detail to notice is the top of the larger box on the right, which shows two sets of points, the purple ones correctly



(a) Lidar overlay using forward projection



(b) Lidar overlay using ratio method

Figure 4.5. Lab test for projecting lidar point cloud onto image plane

showing the top of the box but also the orange ones which are actually from the floor behind the box. The only reason these are visible because the lidar can view those parts but not the camera, therefore once the lidar points are translated to the camera origin, the points behind the object overlap with the points in the front. Those orange points should be removed leaving only the purple points before performing the interpolation however since the lake surface, which is the actual objective of this project, does not have any major features protruding out high enough to cause this, it was not considered a major problem moving forward. It is something to consider for future work, in terms of improving the process.

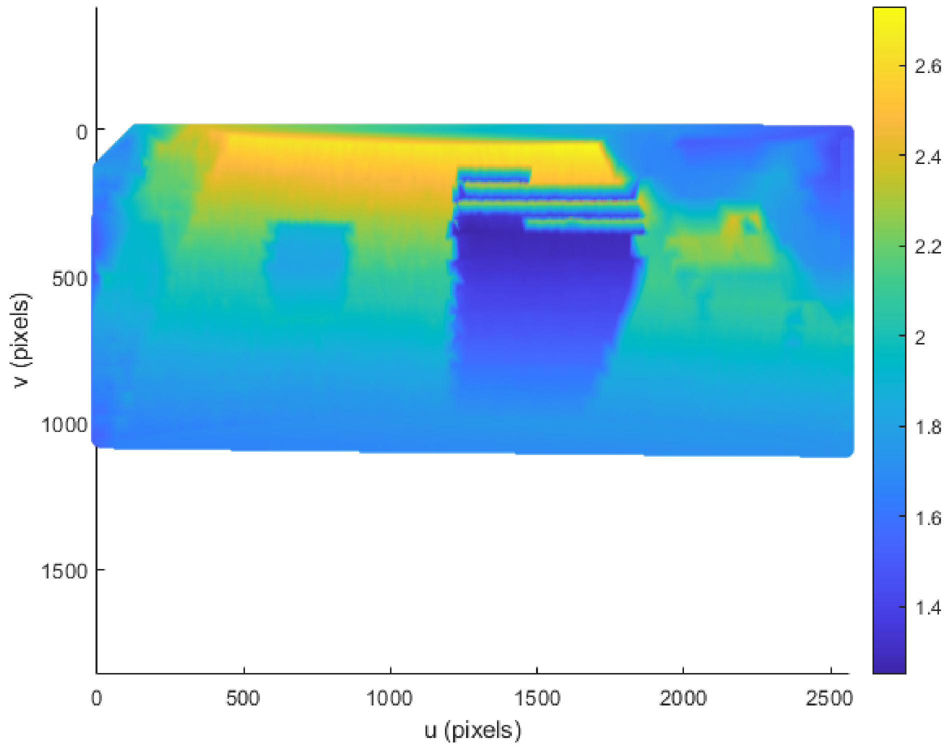


Figure 4.6. Interpolation of the lidar points in the image plane

With the Z_C values assigned to the pixels, the backward projection was performed on the image, one pixel at a time. The camera coordinates for each pixel were obtained which allowed a pixel point cloud to be generated with each pixel plotted at its corresponding coordinates. This is seen in Figure 4.7, in which the pixel point cloud is plotted in Boom coordinates. The boxes were successfully reconstructed alongside some of the surrounding objects such as the chairs and shelf.

There is some warping, with the reconstruction not being as smooth as the actual surfaces, for example the jagged edges of the boxes. This can be attributed to the interpolation not having enough points to interpolate between. The edges of the boxes which had no visual information had their pixels smudged as can be seen between the box on the right and the chair behind it, as the interpolation just assumes the pixels flow from one location to the other. Its happening because there's gap in the lidar point cloud between the box and the chair, seem as the abrupt change in the Z_C value, and so it's interpolating across those, from orange to blue. In the future, there may need to be a requirement to check the difference between adjacent pixel Z_C values before interpolating. Barring these shortcomings the reconstruction does a good job of recreating the environment.

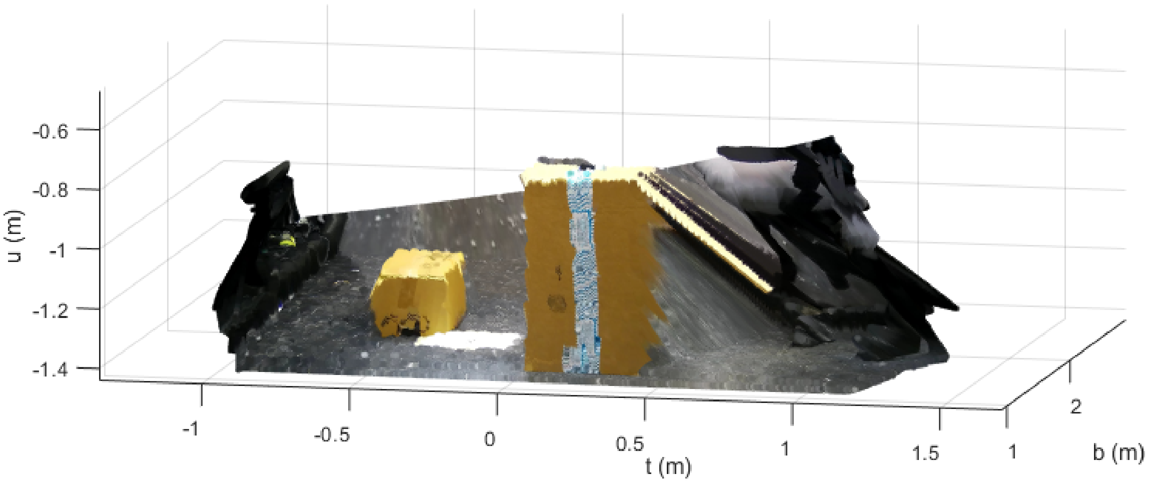


Figure 4.7. Reconstructed environment with boxes

CHAPTER 5
RECONSTRUCTING THE LAKE SURFACE WITH ICE CONDITIONS

With the lab tests demonstrating that the method works, it was then performed on the actual GNSS-R data. For the GNSS-R project the hardware was driven to a location next to Lake Michigan in Chicago. The sensor suite was assembled to face the lake surface and then data was collected for a couple of hours. The hardware was then packed and taken back to the lab, where all the collected data was processed. Each visit to the lakefront was designated as a “data campaign”, of which there have been many over the course of this project. These data campaigns were carried out to obtain data for different lake surface conditions, like ice and water, as well as after any changes were made to the hardware setup. More details as to the locations, dates and hardware setups of these campaigns can be found in Appendix E.

The crux of the GNSS-R project is to be able to differentiate between water and ice surface conditions using the GPS signals. Therefore to that point, data must be collected for both situations and ideally when both are present at the same time. For the 3D reconstruction side of the project, the ice conditions were the best scenario to perform the backward projection method. The lidar used would have good strong returns which mean a more accurate point cloud of the surface and its features. This would provide good data for the backward projection of the camera feed.

5.1 Backward projection of lake surface with ice conditions

Data campaign 7 was one of the field tests for which the Lake Michigan surface was frozen over. Figure 5.1 shows the sensor suite setup at the edge of a dock, facing the frozen lake surface. This test provides a good example to demonstrate the reconstruction process for ice surface conditions. One other reason this particular test is chosen as an example is that there is a large ice formation next to the dock in view of the camera and scanned by the lidar, which makes it a feature visible in the reconstruction process.

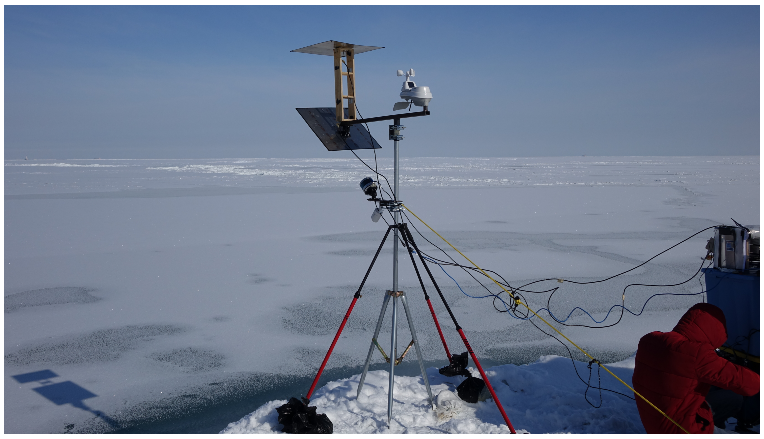


Figure 5.1. The hardware setup at the dock for Data campaign 7. Photo credit: David Stuart

Figure 5.2 shows an undistorted image from the field test. The surface of the lake can be seen as being frozen over with snow covering parts of it. There is also a large ice buildup seen at the bottom of the image, which is next to the dock's edge.

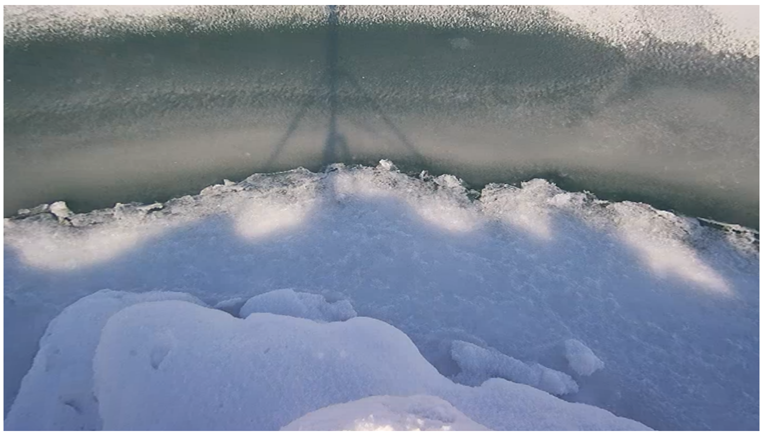


Figure 5.2. Camera image of ice surface conditions from Data campaign 7

The data was collected at the 31st street beach northern dock. A Google map of the location is shown in Figure 5.3, which is overlaid with the full lidar point cloud (all yellow and red points) with the origin being the camera, indicated as the black cross. The points in yellow are the lidar returns outside the FOV of the camera and the ones in red are those in the FOV. The lidar point cloud was rotated to the Boom coordinate system. It was resized and rotated on top of the Google maps image by hand, overlaying the features in the image with the points in the point cloud. The adjustments necessary to rotate the point cloud in order for the image features to coincide with the lidar points, such as the edge of the docks, the trees and the steps along the shore, could be due to the inaccuracy in measurements of the rotation angles. This figure then shows that the coverage of the lidar across the water surface is wide. However, the camera has a limited FOV so the lidar point cloud was cropped to the part of the point cloud shown in red. The cropped lidar point cloud is shown separately in camera coordinates in Figure 5.4.

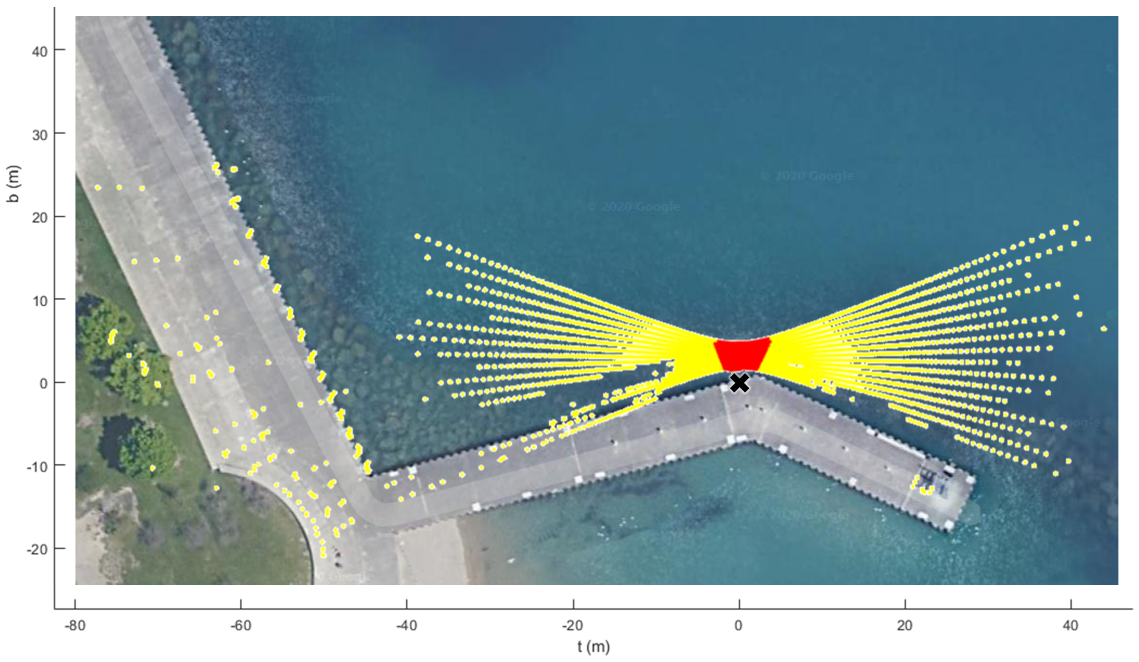


Figure 5.3. Lidar point cloud overlaid with Google map location of Data campaign 7

This cropped lidar point cloud was then projected on the image as shown in

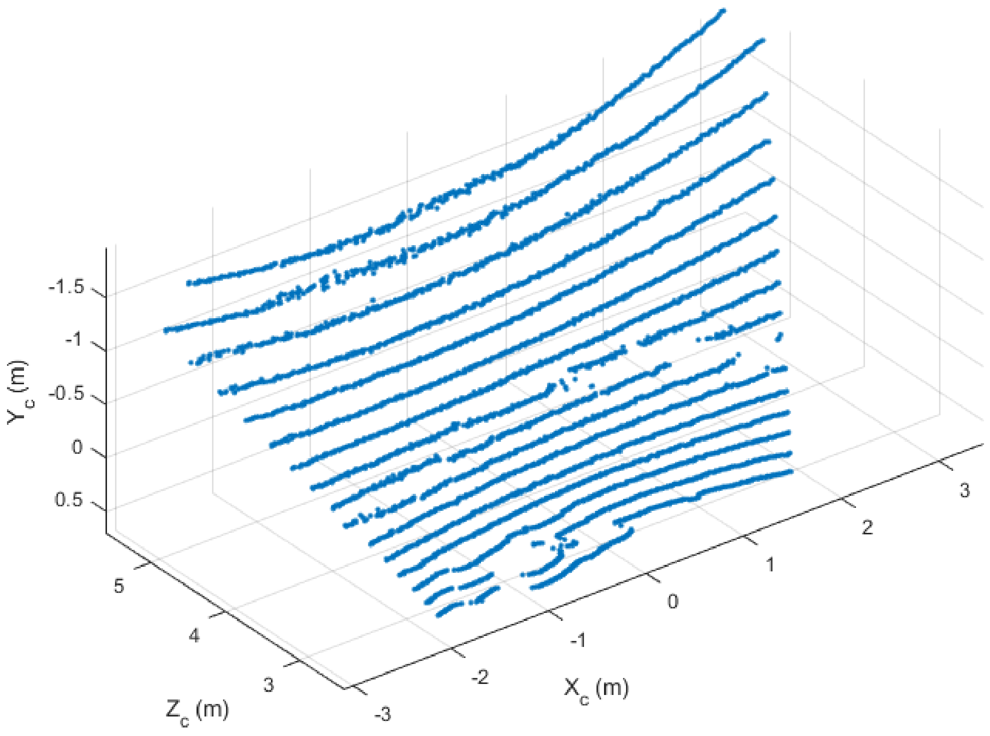


Figure 5.4. Cropped lidar point cloud in camera coordinates for Data campaign 7

Figure 5.5, using the alternative ratio method. The colorbar shows the lidar points Z_C values in meters. The features in the image seem to match well with the point cloud. It is not a perfect match but that is to be expected because the projection onto the image plane is making an approximation alongside other sources of error.

The overlaid lidar point cloud was then interpolated to cover the image to allow for backward projection of all the pixels. The image was then reconstructed and can be seen in Figure 5.6. The plot was rotated to a perspective that emphasizes the protruding ice feature. This shows that this reconstruction method works well on full ice surface conditions.

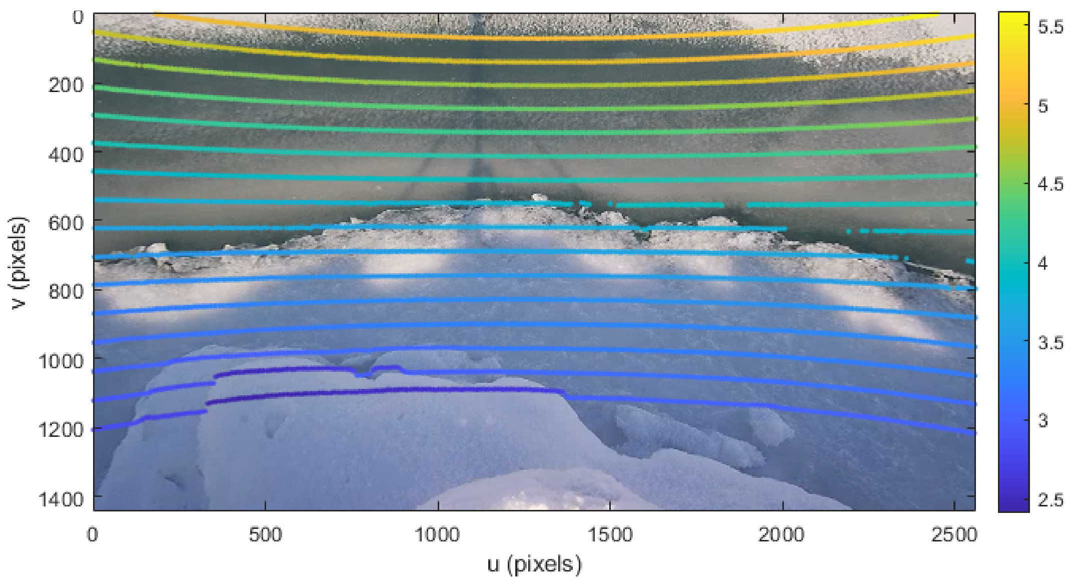


Figure 5.5. Lidar point cloud overlaid onto camera image for Data campaign 7

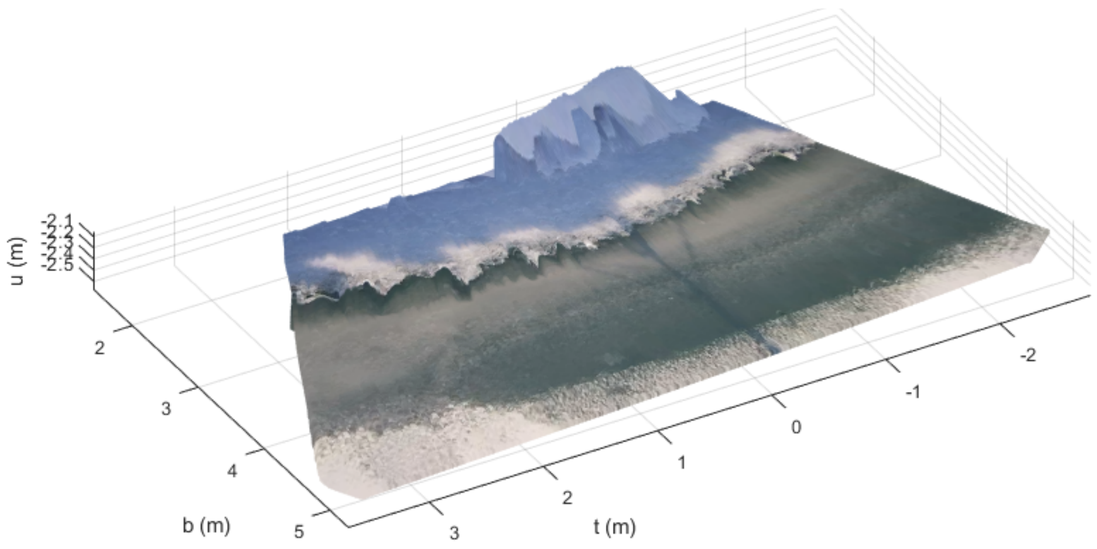


Figure 5.6. Reconstructed lake surface with ice conditions from Data campaign 7

CHAPTER 6
RECONSTRUCTING THE LAKE SURFACE WITH WATER CONDITIONS

As previously stated in Chapter 5, the differentiation between ice and water is the main goal of the GNSS-R project. When the lake is frozen over, the lidar returns are very strong and very consistent. This provides good data with which to perform the backward projection and reconstruct the lake surface. Having good lidar data from only ice conditions is not enough. Backward projection is required for surface water conditions as well. However, the lidar returns from water are rare.

Data campaign 5 was one of the field tests for which the Lake Michigan surface was completely water, with no surface ice. This test provides a good data set with which to develop a solution for the problem of backward projection with limited lidar data. Details of campaign 5 are provided in Appendix E. The hardware setup was similar to the field test for ice conditions, with a single camera and lidar (Setup 1 as explained in Appendix D). In this chapter, I present an alternative method for estimating the Z_c coordinate needed for backward projection from the lidar when the lidar point cloud data are sparse.

6.1 Backward projection of lake surface with water conditions

Data campaign 5 was conducted at the northern dock of Chicago's 31st Street beach. This field test was performed during the summer when the lake surface was completely devoid of any ice, thus providing surface water conditions. The sensor suite can be seen set up on the edge of the dock, pointing towards the lake, in Figure 6.1, with downtown Chicago visible in the background. The lake can be seen here to be completely unfrozen water.

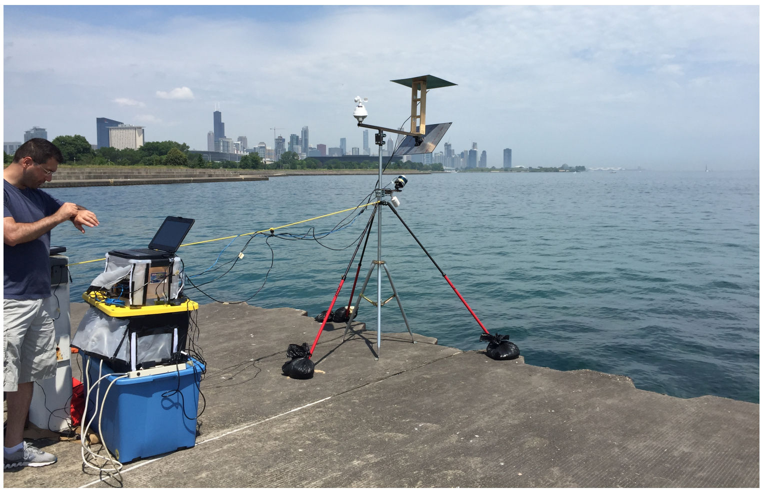


Figure 6.1. The hardware setup at the dock for Data campaign 5. Photo credit: Houshine Sabbagh Zadeh

The camera collected data alongside other sensors for a couple hours. Figure 6.2 shows an undistorted image taken by the camera at a single timestamp. The surface of the lake in view can clearly be seen to be completely water.

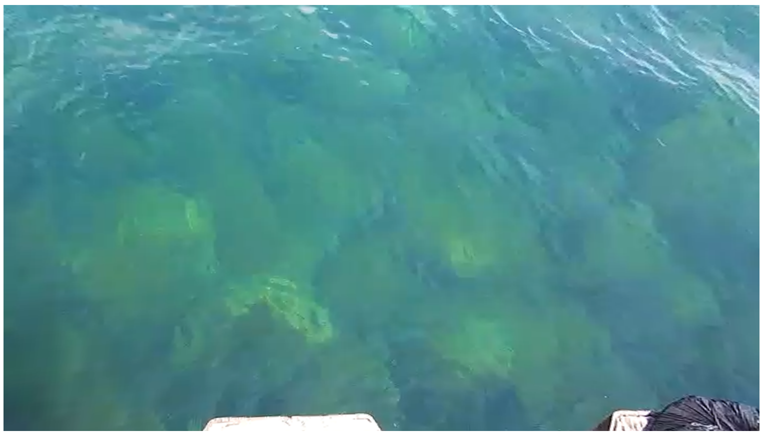


Figure 6.2. Camera image of water surface conditions from data campaign 5

While the camera provides good data, the lidar returns off the water are sparse and very low intensity. This can be seen in Figure 6.3, in which one frame of the full lidar point cloud, in Boom coordinates, is overlaid on top of a Google map of the test location. It is the same 31st Street beach northern dock as Data campaign 7 in the previous chapter. The yellow points are the lidar returns off objects not in the FOV of the camera, with the camera location indicated by a black cross. This figure was created by resizing and rotating the point cloud plot on top of the Google Maps image manually, overlaying the features in the image with the points in the point cloud. It provides a good visual reference to identify points from which the lidar returns are available. What is important to note here is that, compared to Figure 5.3, there are no red points. That's because for this instant there are no lidar points, in the FOV of the camera, from the water surface. There were occasional returns from the surface from time to time, but never enough points at any given timestamp to be able to perform the backward projection.

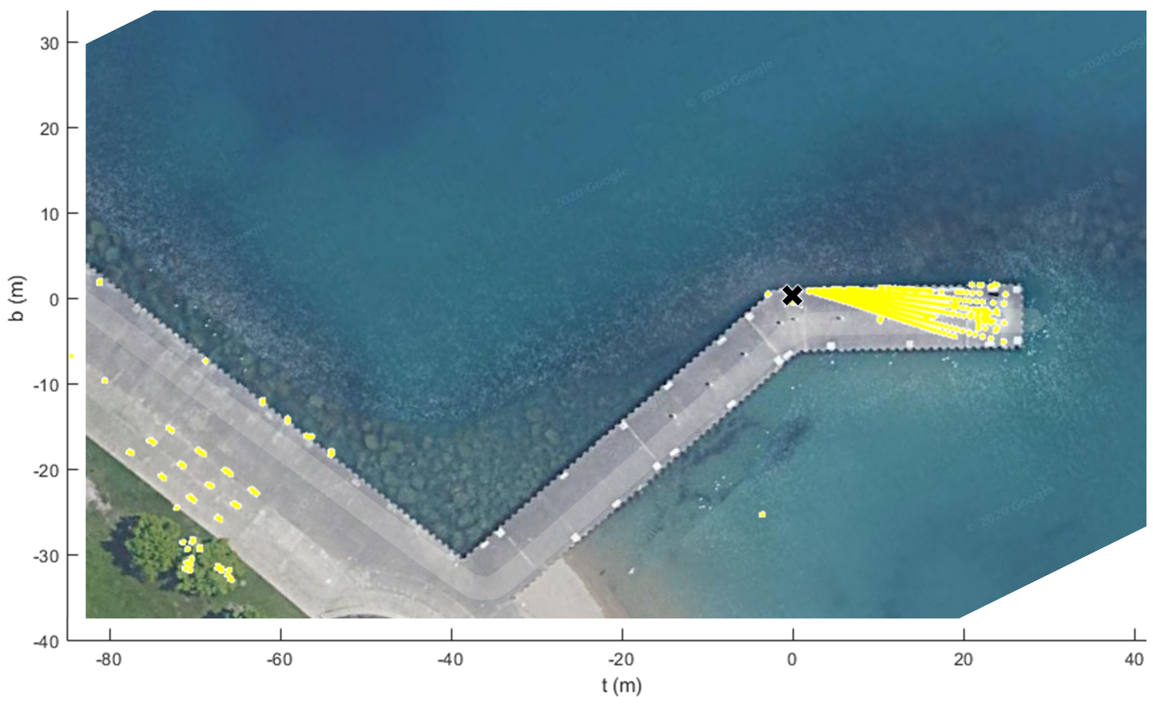


Figure 6.3. Lidar point cloud overlaid with Google map location of Data campaign 5

6.1.1 Generating a pseudo-lidar point cloud. Each frame, comprising one complete 360° scan, from the lidar point cloud did not have enough points. However, occasionally during the complete 20-minute data campaign there would be a return picked up from the water surface. Therefore, to estimate the water surface, all the returns obtained from the water over the course of the test were aggregated. Figure 6.4 shows all the returns plotted together in a single plot, in camera coordinates. These points were rotated to the Boom coordinate frame, where the mean $\hat{u}$ coordinate of the points over the entire time and camera FOV, was determined.

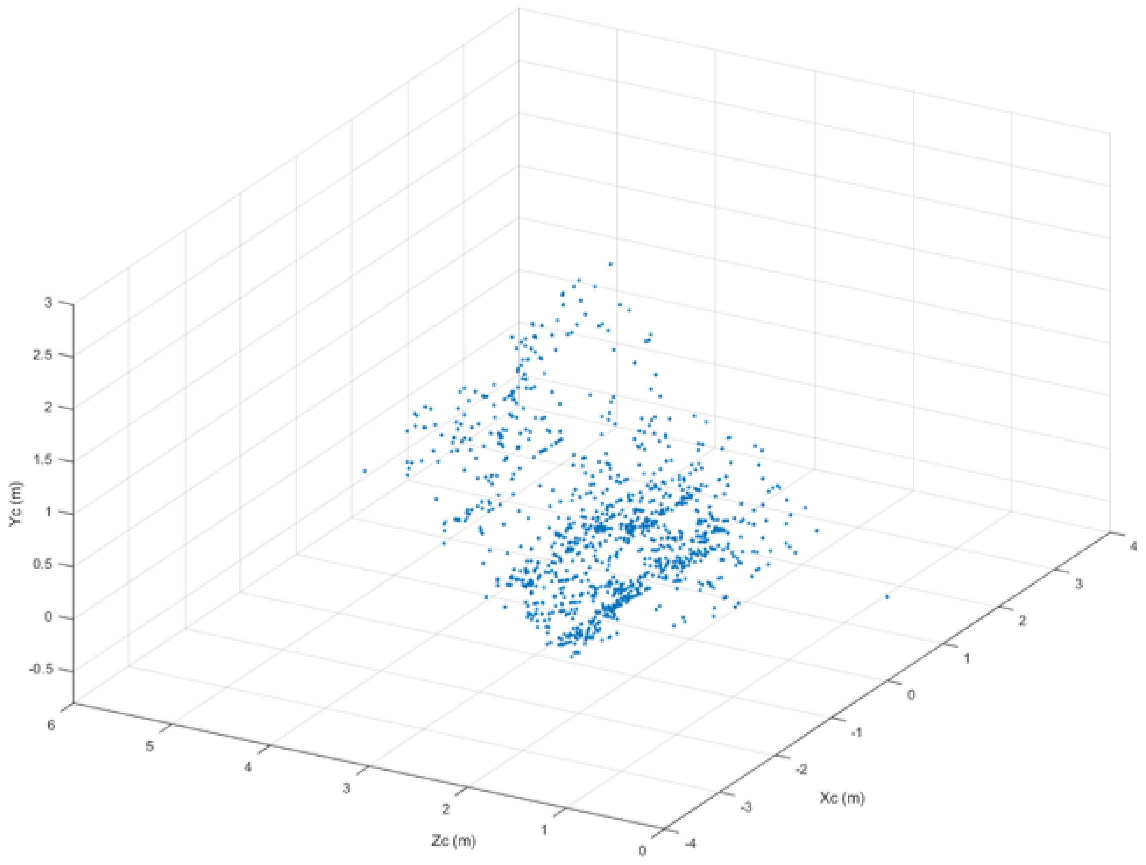


Figure 6.4. All lidar returns from water surface over the course of the entire test, plotted in camera coordinates

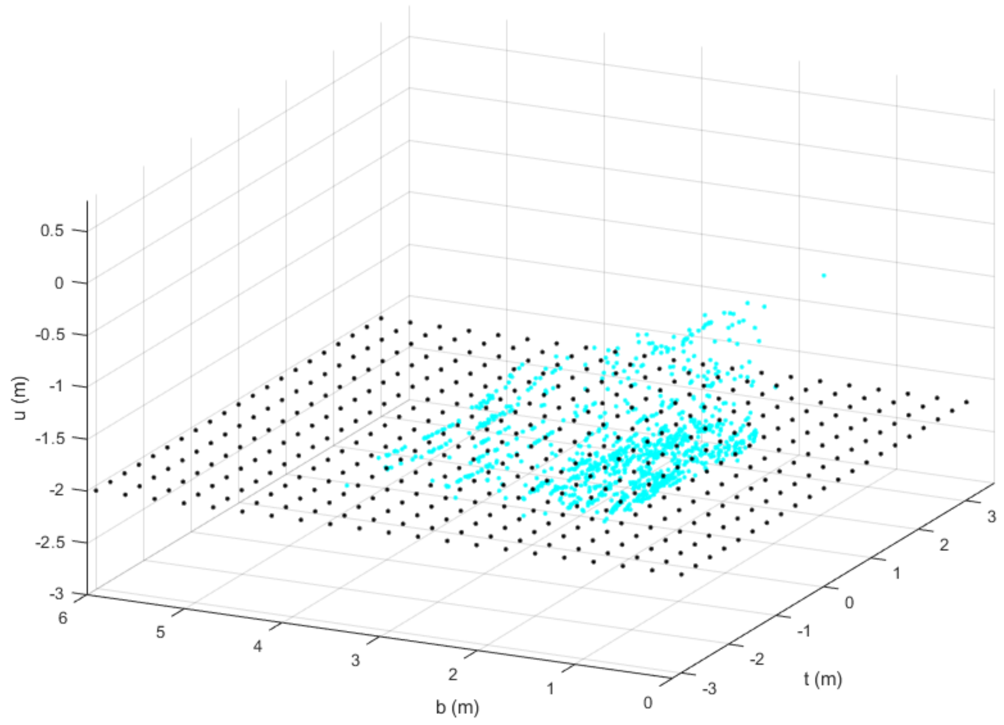
The surface of the water is then estimated to be at this mean level, and so a grid of points is plotted in Boom coordinates with this $\hat{u}$ coordinate. The grid is generated with points stretching out beyond what is visible by the camera, as it will be later cropped to the FOV. These points are dubbed "pseudo" lidar points because they will be used as though they are lidar returns from the water surface for the purposes of the backward projection. The pseudo-lidar point cloud can be seen in Figure 6.5 from two different views, in the Boom coordinates. The actual accumulated lidar returns from the water are plotted in blue, while the pseudo lidar points generated at the mean are plotted in black. Figure 6.5(b) provides a side view of the points plotted. It is interesting to note how spread out the actual lidar returns from the surface are. Likely this is due to waves on the surface. The reflection points above -1.5 m are unlikely to be water surface but instead stray particles in the air reflecting lidar energy randomly at different moments during the data campaign. For this work the outlier stray points were used in calculating the mean value. However, for future work, they should be removed.

6.1.2 Performing backward projection using the pseudo-lidar point cloud.

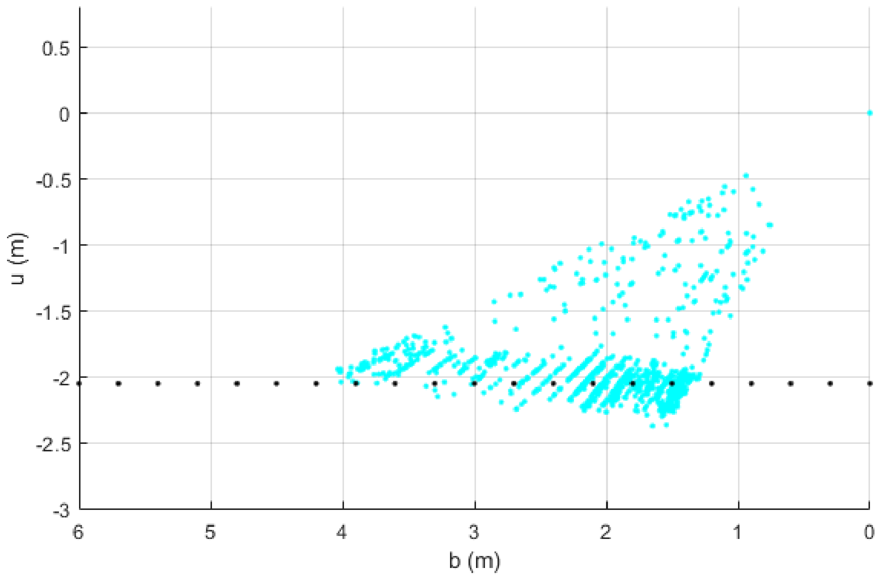
The pseudo-lidar point cloud is then substituted into the backward projection method, to be used instead of the actual lidar point cloud. The pseudo point cloud is rotated to the camera coordinate system, then cropped to the FOV of the camera, as with the method used for the actual lidar points. Figure 6.6 shows the pseudo-lidar point cloud plotted in camera coordinates and cropped to remove the points outside the camera FOV. The points in black are the pseudo-lidar points. The blue points are the actual lidar returns plotted for reference.

This pseudo-lidar point cloud is then projected onto the camera image, shown in Figure 6.7, using the alternative ratio method. The image is in pixel coordinates with the colorbar showing the Z_C values of the pseudo lidar points.

The rest of the process remains unchanged. The overlaid pseudo-lidar points don't cover the entire image and so are interpolated over the resolution of the image. Each pixel is then backward projected using its pixel coordinates and corresponding Z_C value. The lake surface is thus reconstructed in 3D coordinates as shown in Figure 6.8, in Boom coordinates. Note that while the camera image reflects sunlight



(a)



(b)

Figure 6.5. Pseudo-lidar point cloud (black) generated at the estimated water level based on the mean u coordinate of the time-aggregated lidar point cloud (blue), plotted in Boom coordinates.

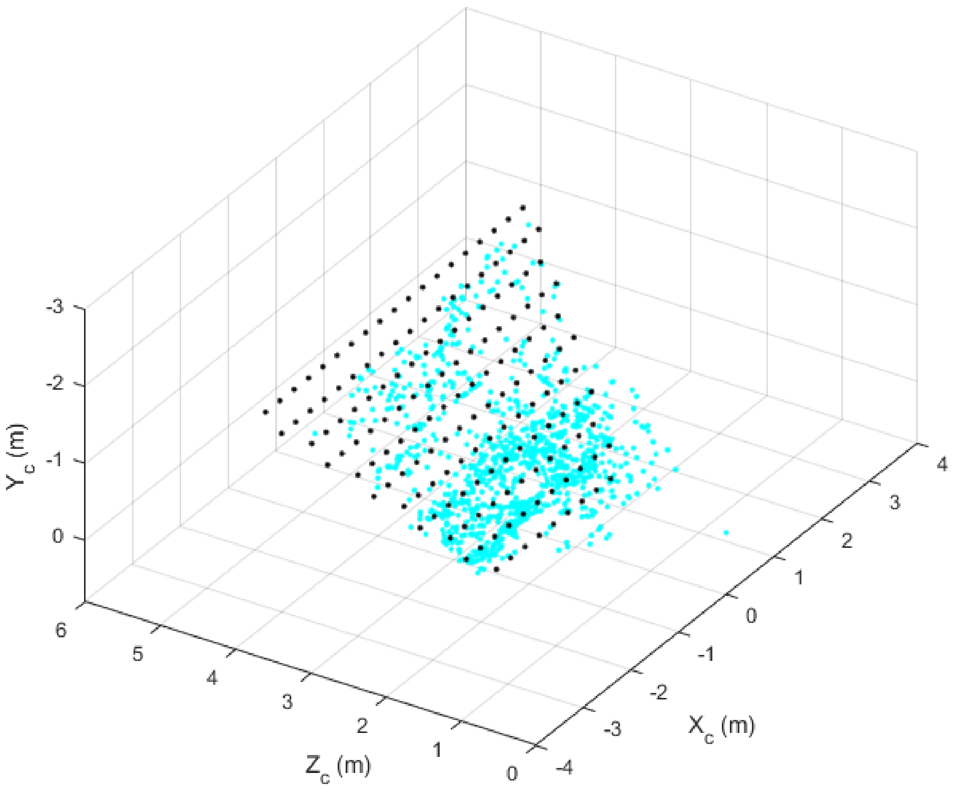


Figure 6.6. Cropped pseudo lidar point cloud (black) and measured lidar point cloud (blue) in camera coordinates for data campaign 5

at certain points, indicating variation in u coordinate, all camera pixels are assigned the same mean u value over the FOV and over time.

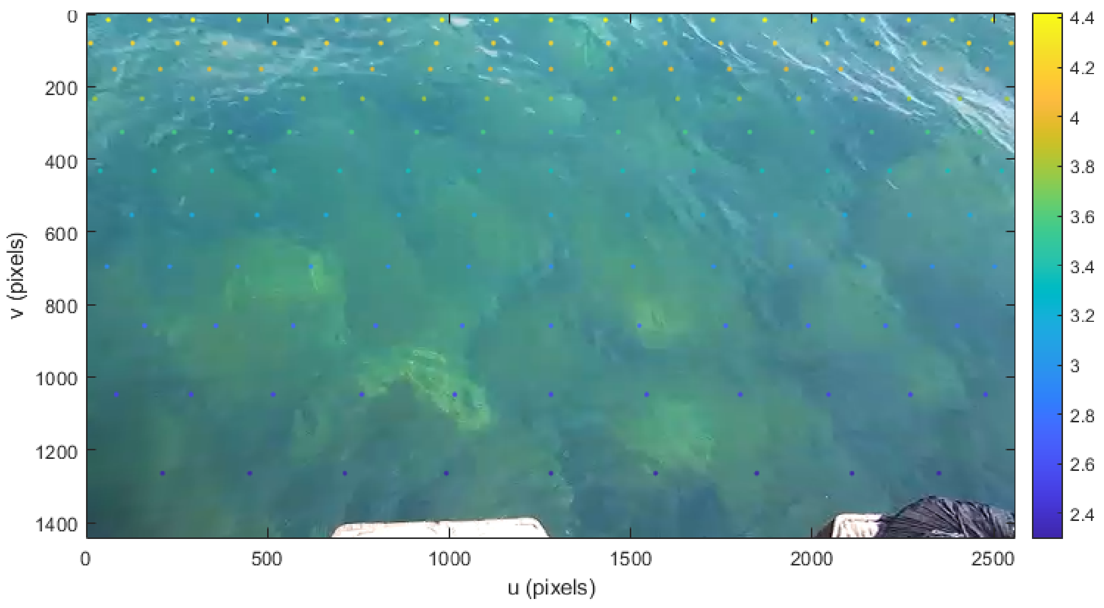


Figure 6.7. Pseudo-lidar point cloud overlaid onto camera image for data campaign 5. The Z_C values on the colorbar are in meters

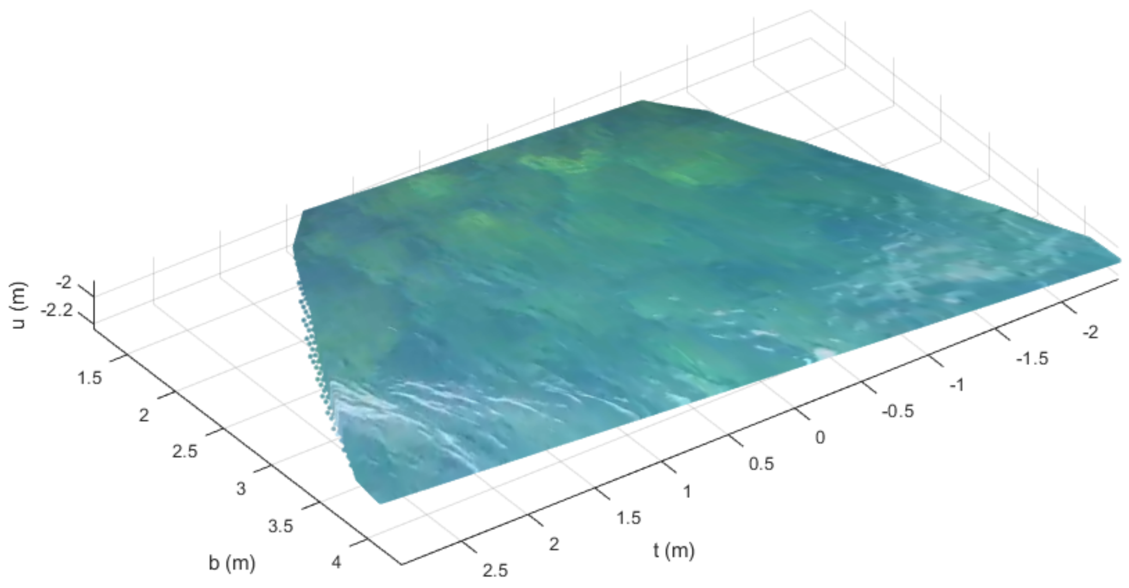


Figure 6.8. Reconstructed lake surface with surface water conditions from Data campaign 5

CHAPTER 7
RECONSTRUCTING THE LAKE SURFACE WITH MIXED ICE AND WATER CONDITIONS

The backward projection algorithm developed works for fully-ice surface conditions as explained in Chapter 5. For water surface conditions some additional post-processing must be performed to be able to backward project the camera image of the lake surface as explained in Chapter 6. Having performed the backward projection for entirely ice or entirely water surface conditions, the final step is to gather and then analyze data from a mixed surface condition in which both ice and water surface conditions are present.

It was also important to have data from mixed conditions as the goal of this GNSS-R project is to differentiate between ice and water using GPS reflected signals. The specular point of those reflected signals moves across the lake surface as the GPS satellites move across the sky. To have both ice and water in the same data campaign presents an ideal opportunity to detect GPS reflections when one such specular point moves over a boundary between ice and water. This can be visually confirmed from the reconstructed lake surface and further investigated though GPS signal processing. Change in certain signal characteristics such as surface reflectivity will be identified by the lake surface conditions change and then be used as metrics to distinguish between the types of surface conditions (beyond the scope of this thesis).

This chapter shows results of backward projection of camera images for mixed ice-and-water surface conditions. The hardware setup is changed from a single camera to triple camera setup, designated as Setup 2, further explained in Appendix D. Data from campaign 10, whose details are summarized in Appendix E are used in this demonstration.

7.1 Changing to a three camera setup

As previously seen in Figure 5.3, the camera FOV covers only a small portion of the lake surface. Taking the lidar point cloud from data campaign 7 as a reference, the red points are the only lidar returns from the lake surface that are visible to the camera. A large number of the surface lidar returns do not have corresponding visual images from a camera that can be reconstructed. In other words the lidar data and this backward projection method are underutilized given the domain over which GPS specular points may lie. Thus, we increased the camera coverage of the lake for all tests after Data campaign 9.

Two more cameras are added to the setup to achieve wider lake surface imaging coverage. Two new security CCTV cameras similar to the old camera are used. More information on the specifications can be found in Appendix A. Not only are these cameras inexpensive, their similarity to the old camera makes them easier to integrate into the existing hardware.

The processing involved with backward projecting the lake surface does not change. The data from each camera is backward projected separately using the lidar to produce three different lake surface reconstructions. Merging them to produce one singular lake surface map is beyond the scope of this work, and will be the subject of future study.

7.2 Backward projection of lake surface with mixed conditions

To test out the new hardware setup as well as obtain sensor data from mixed ice and water surface conditions, data campaigns 10 and 11 were carried out. Data campaign 10 is used in this work to show the results of these changes. Details of this test are provided in Appendix E. This test was performed at the 31st Street Beach harbour. Figure 7.1 shows the GNSS-R sensor suite setup facing the lake surface. The individual ship docks are seen in the background. The surface of the lake can be seen to possess both ice (to the right) and water (to the left) regions, where the sensor suite is setup pointing roughly in between the two.

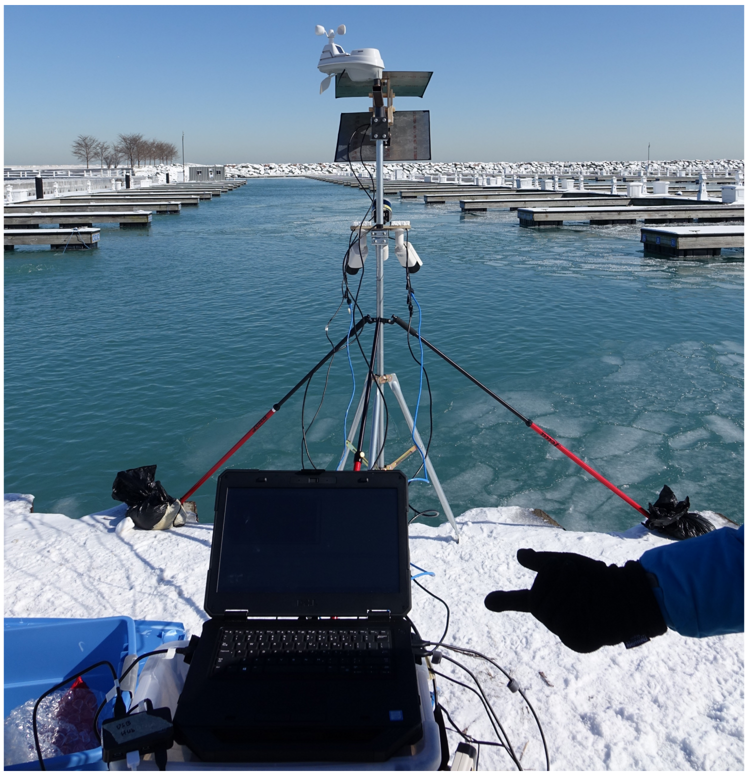
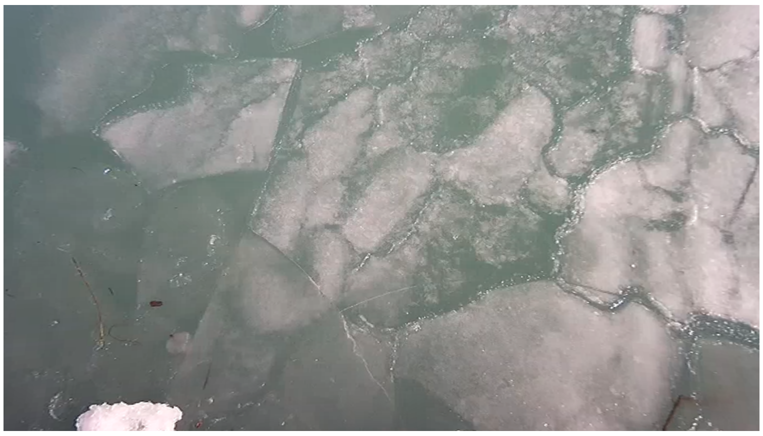
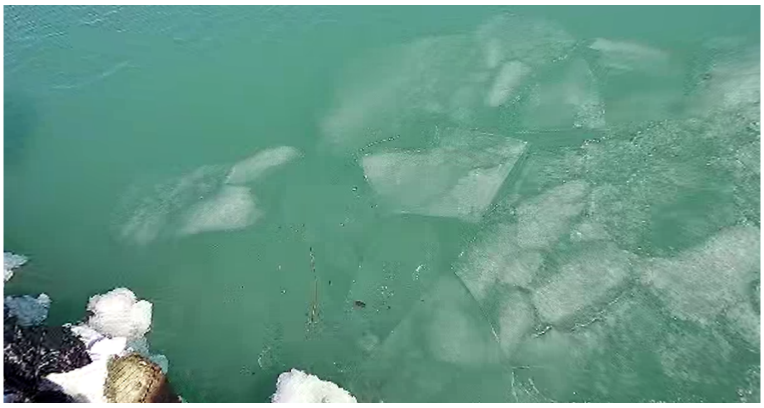


Figure 7.1. The hardware setup at the dock for Data campaign 10. Photo credit: David Stuart

Figure 7.2 shows each of the camera images of the lake surface. Camera 1 is the same camera used in the previous tests. Camera 2 and Camera 3 are the new cameras each attached with an azimuth angle with respect to the $\hat{b}$ axis, also known as the forward boom direction. This allows the cameras to cover the lake surface to the left and right of the original camera. The laptop used to run the sensor and collect data can be seen in the foreground.

Figure 7.3 shows the lidar point cloud plotted in Boom coordinates, overlaid onto the Google maps location for Data campaign 10. The black cross, the origin of


(a) Camera 1


(b) Camera 2

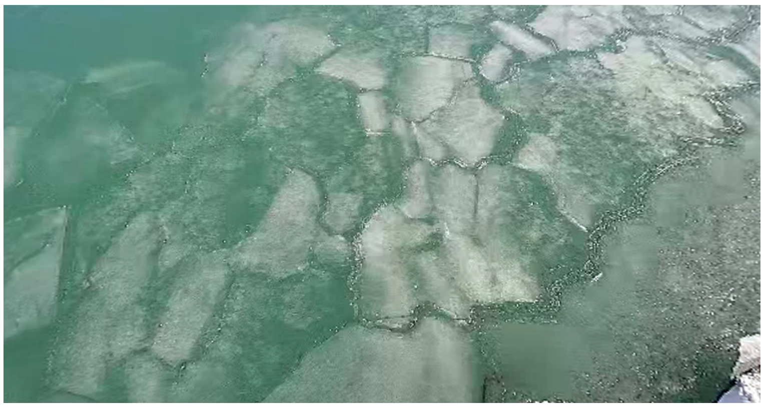

(c) Camera 3

Figure 7.2. Camera images of the mixed ice and water surface conditions from Data campaign 10

the plot, shows the location of camera 1. The yellow and red points plotted are all lidar returns, with yellow being those not in FOV of the cameras and the red being the ones in FOV of the cameras. Compared to the lidar point clouds from previous tests it can be seen that the 3 cameras allow for wider surface coverage.

The lidar point cloud also shows the same mixed surface conditions as seen in the camera images. The lake surface in front of and to the immediate right of the sensor setup is frozen, providing abundant lidar returns. The lake surface to the left is water and so there are almost no returns.

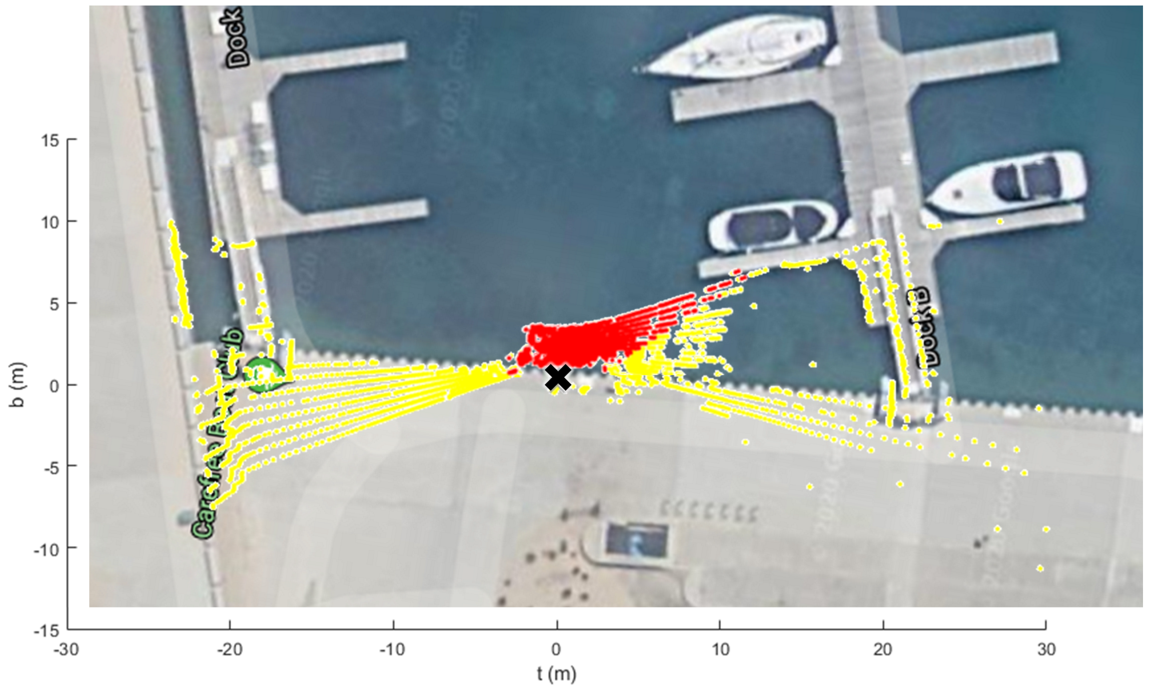


Figure 7.3. Lidar point cloud overlaid with Google map location of Data campaign 10

The lidar point cloud is cropped to the FOV of each camera separately and overlaid onto its respective image, using the alternative ratio method. Using the Camera-Lidar Setup 2 angles described in Appendix D, the lidar points did not line up well with their counterparts in camera image 2. This might be due to the orientation not being set perfectly, and there being error in the measured angles. Thus, the azimuth and elevation angles were changed through trial and error until the features in both matched up. For camera 2 the new angles were set as $az_C = -24°$ and $el_C = -39°$. The angles for camera 1 and 3 were not altered. The image-lidar overlays can be seen in Figure 7.4, Figure 7.5 and Figure 7.6. The plots are in pixel coordinates with the colorbars indicating the lidar points' Z_C values in meters.

The lidar points can visibly be seen to return from the ice portions of the lake surface. The lidar points follow the ice features of the surface quite accurately. The surface ice has many reflections, but there are portions in the ice devoid of any lidar returns due to the presence of water. There are areas of the images where the lidar either does not scan, such as the top portions of camera 2 and camera 3 images, or areas where the lidar does scan but there are no returns from the surface, such as the left most portion of camera 2.

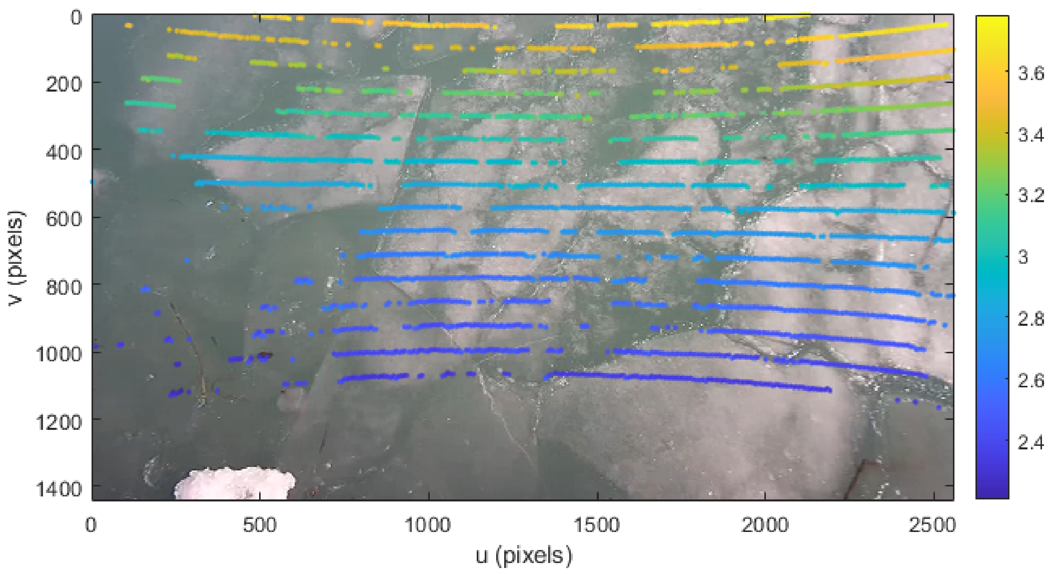


Figure 7.4. Lidar point cloud overlaid onto Camera 1 image for Data campaign 10

A pseudo-lidar point cloud is generated to fill in those regions. With mixed surface conditions it is much easier to generate the pseudo-lidar point cloud. The ice regions are simply used to obtain a mean value for the lake water level, i.e. a mean

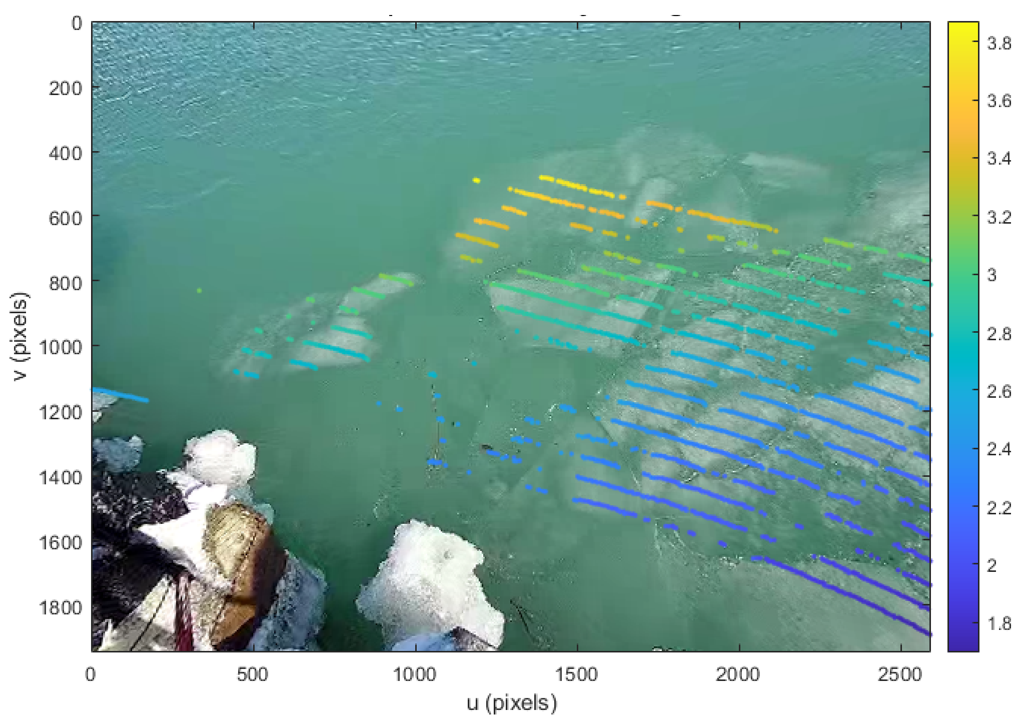


Figure 7.5. Lidar point cloud overlaid onto Camera 2 image for Data campaign 10

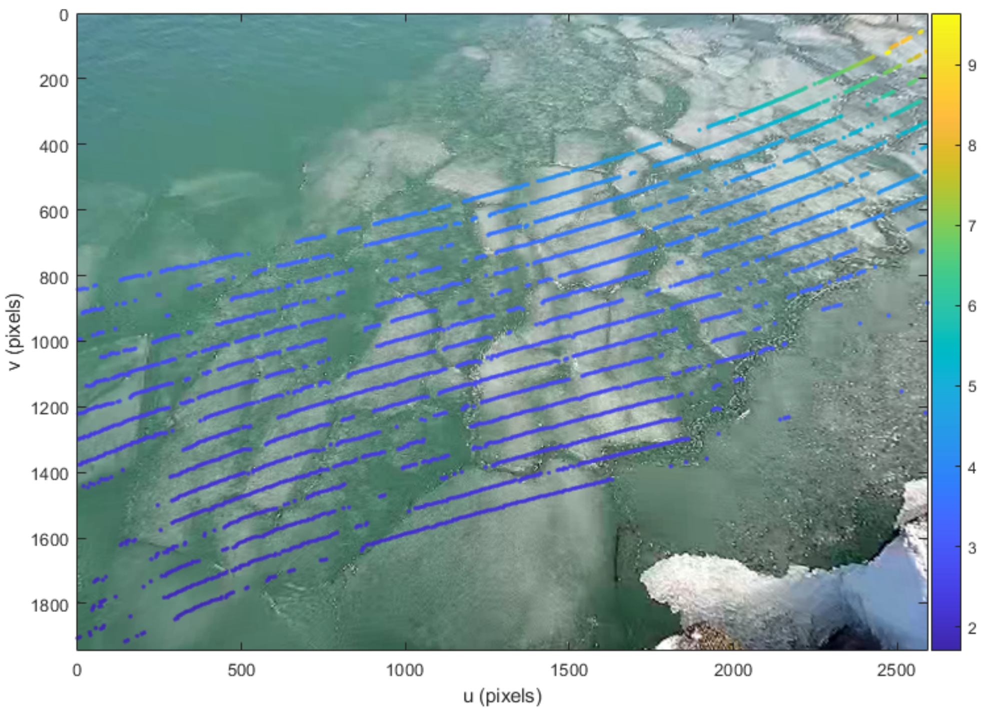


Figure 7.6. Lidar point cloud overlaid onto Camera 3 image for Data campaign 10

$\hat{u}$ value. The actual lidar returns from the ice, visible to each respective camera are used. Unlike the case with only water surface conditions in Chapter 6, if there are ice conditions in view of the cameras, the lidar points do not need to be aggregated over all time as each frame has sufficient returns to be used in calculating the mean surface level. A grid of pseudo-lidar points is created in the Boom coordinates that span a length of -20 to 20 m in the $\hat{t}$ direction, 0 to 19 m in the $\hat{b}$ direction and at the mean water level in the $\hat{u}$ axis. This mean weater level is calulated using the actual lidar points visible to each camera, so it generates one mean height per camera. This size of the grid is large so as to cover the FOV of all three of the cameras. This grid of pseudo-lidar points is rotated to the camera coordinate frame and then overlaid on to the image, alongside the actual lidar points, as previously explained in Section 3.5. Any pseudo-lidar points that are too close in proximity to actual lidar points in the image, i.e. within a 90-pixel radius, are removed. Thus, the pseudo-lidar points only cover the regions where there are either no lidar returns or no lidar coverage.

The camera images have the pseudo-lidar points added to their camera-lidar overlay, as shown in Figure 7.7, Figure 7.8 and Figure 7.9. The pseudo-lidar grid can be seen to cover the regions where the actual lidar returns were missing, when compared to Figure 7.4, Figure 7.5 and Figure 7.6 which show only the actual lidar points. The color of the actual lidar points is different from the previous figures because of the change in the maximum and minimum values in the colorbar, based on the addition of the pseudo lidar points.

The next steps are the same as in the previous chapters: all the lidar points are interpolated over each respective camera image, to obtain the necessary Z_C values for all pixel coordinates. Then the image pixels are all backward projected to produce the reconstructed surface. Since each camera is treated separately, there are three different lake surface maps generated. Figure 7.10, Figure 7.11 and Figure 7.12 show the

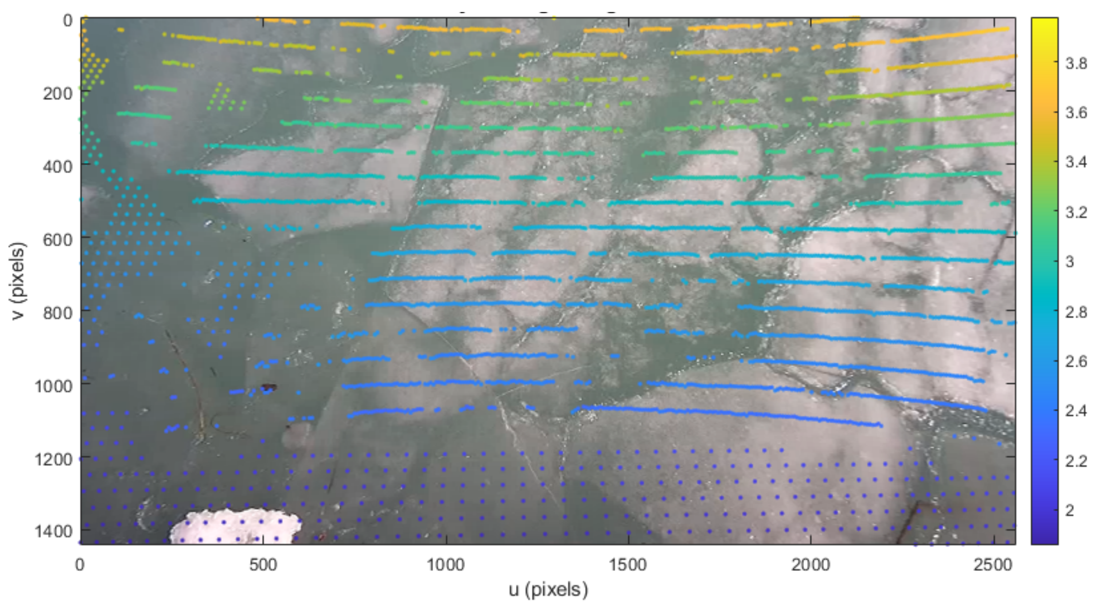


Figure 7.7. True lidar points and pseudo-lidar points overlaid onto Camera 1 image for Data campaign 10

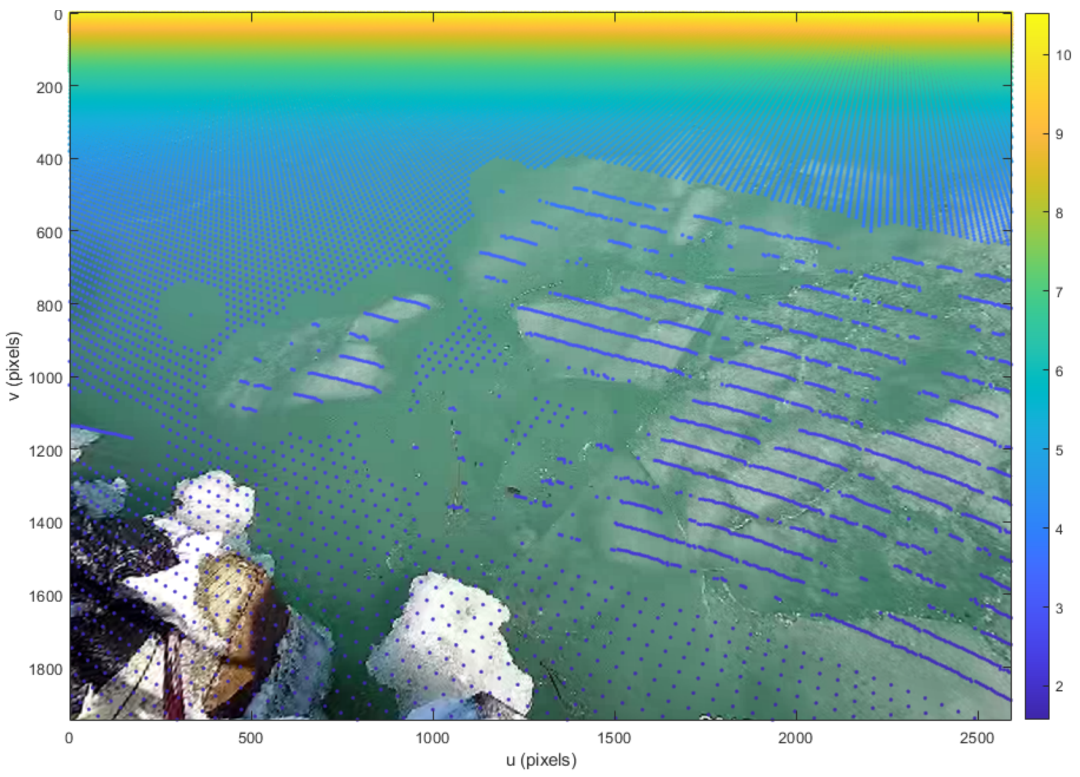


Figure 7.8. True lidar points and pseudo-lidar points overlaid onto Camera 2 image for Data campaign 10

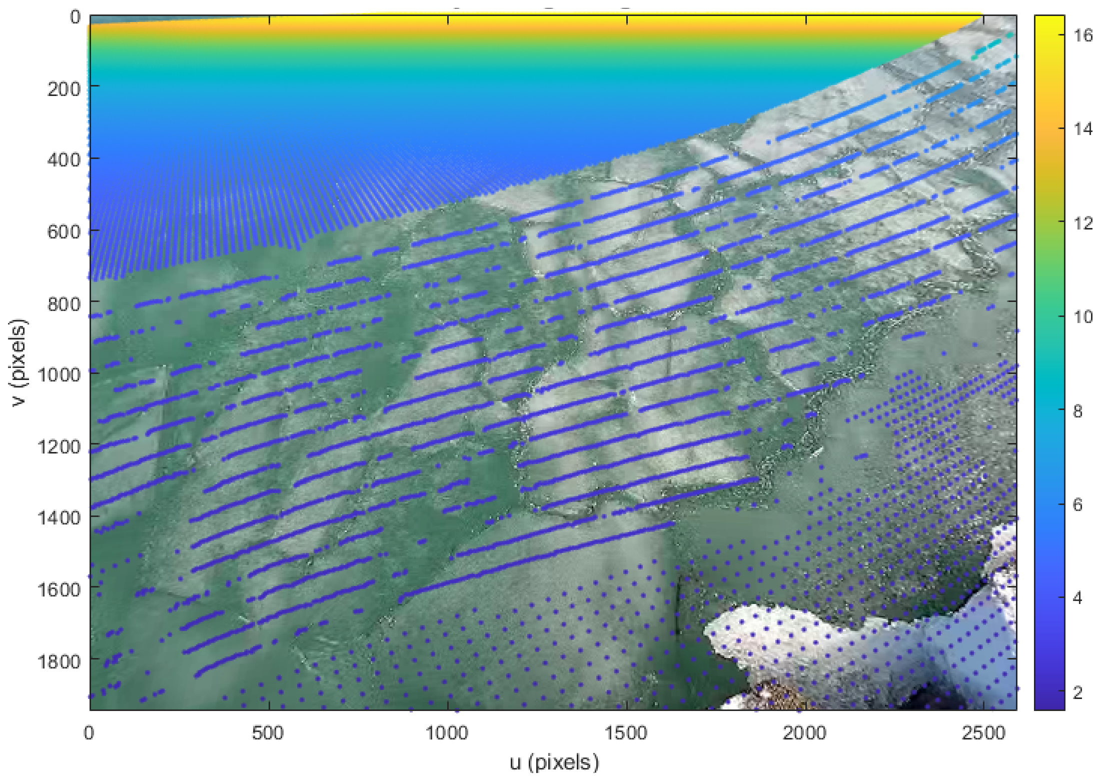


Figure 7.9. True lidar points and pseudo-lidar points overlaid onto Camera 3 image for Data campaign 10

reconstructed lake surfaces from each individual camera and are plotted in Boom coordinates. The water and ice surfaces have both been successfully reconstructed. The addition of the two new cameras allow more of the lake surface to be reconstructed. The ice and water surfaces can be visually identified as required by this project. This demonstrates the algorithm works in these mixed surface conditions. For future work the reconstructed surfaces would need to be stitched together to produce a singular map of the lake surface with GPS specular reflection points mapped on top.

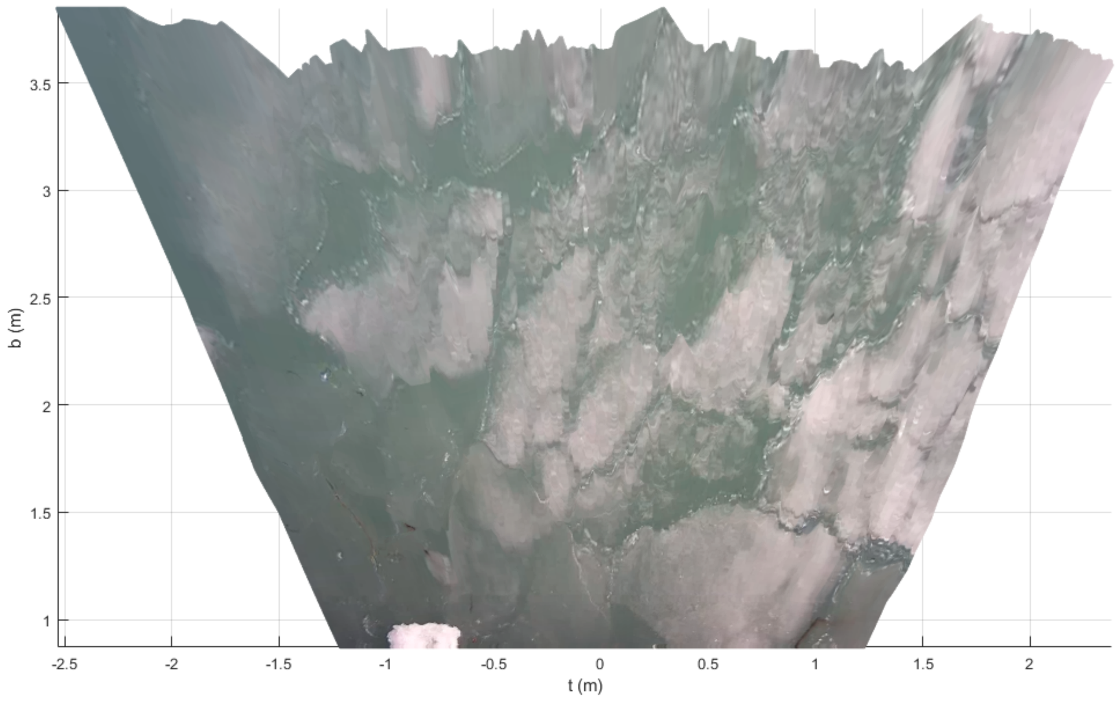


Figure 7.10. Reconstructed lake surface with mixed ice and water conditions for Data campaign 10 - Camera 1

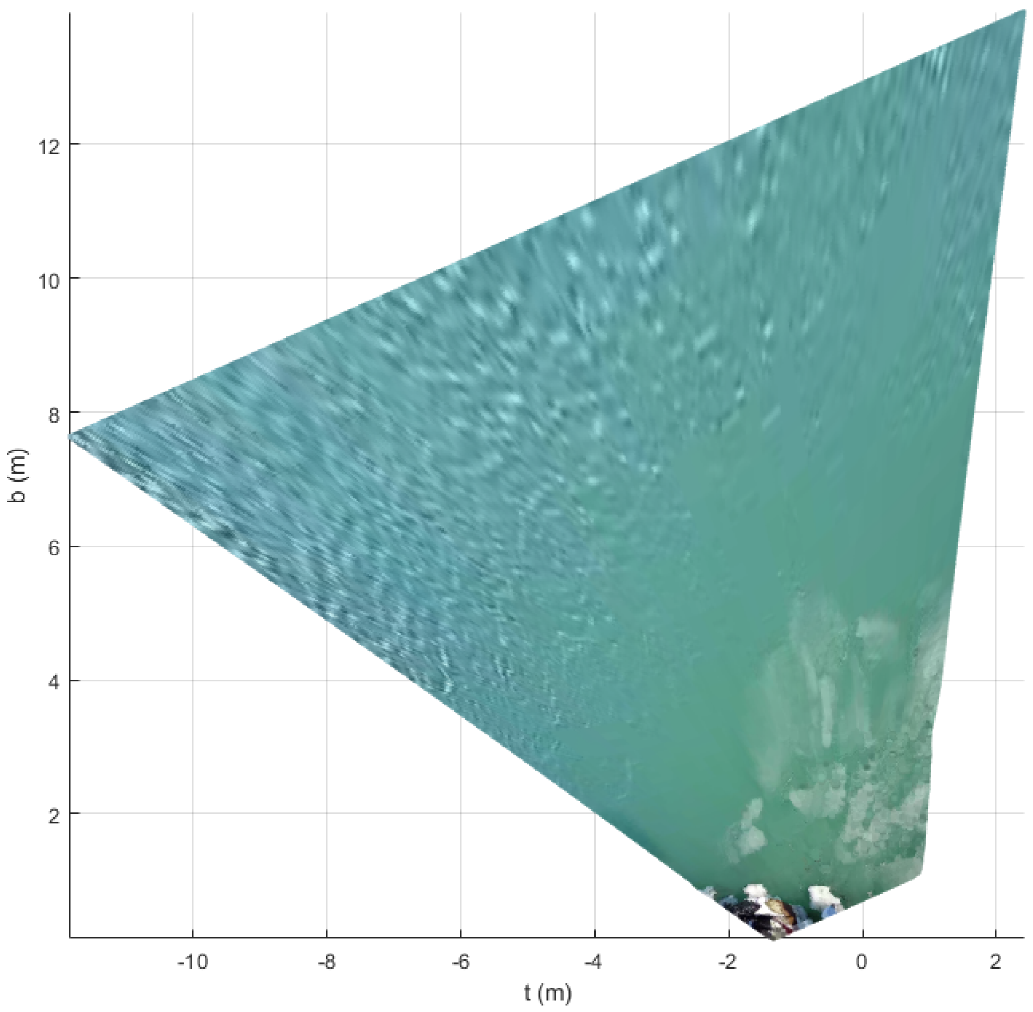


Figure 7.11. Reconstructed lake surface with mixed ice and water conditions for Data campaign 10 - Camera 2

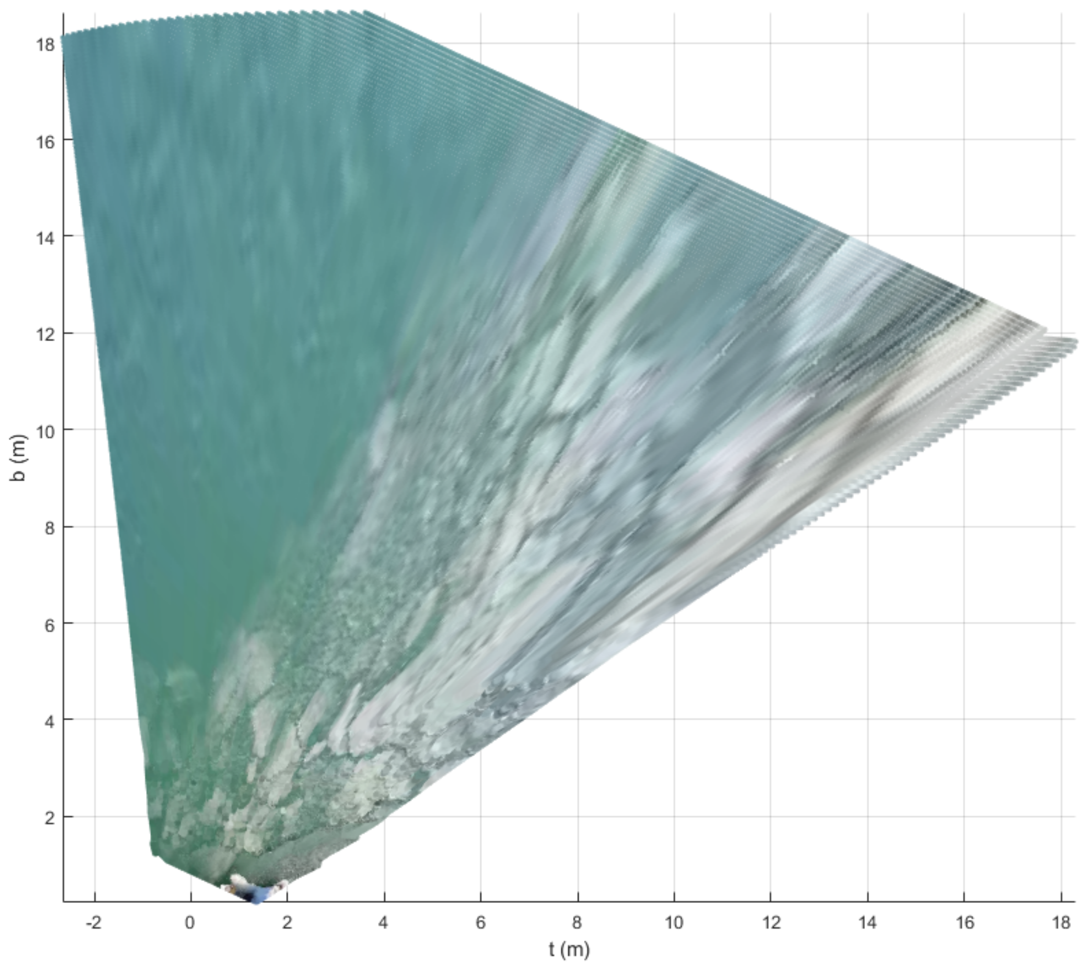


Figure 7.12. Reconstructed lake surface with mixed ice and water conditions for Data campaign 10 - Camera 3

CHAPTER 8
CONCLUSION

8.1 Summary

This work provided the basis to reconstruct a lake surface using a camera-lidar setup. The algorithm was developed based on backward projection of the camera image which required depth information provided by the lidar. The lidar point cloud was overlaid on the image and subsequently interpolated over the entire image. The image was then projected out into 3D coordinates to recreate the surface.

Since this method relied on lidar returns, which was good in the case of ice surface conditions, it was a challenge when there were little to no returns, as in the case of water surface conditions. This required aggregating the sparse returns of the lidar over time and space and using them to calculate mean surface height relative to the sensors. This mean height was used in backward projecting all pixels of the image for the water conditions.

The tests were performed for only ice surface conditions, only water surface conditions and mixed ice-water surface conditions. The results for all these cases were shown. The hardware setup of the experiments also changed from a single camera and single lidar to three cameras and single lidar, in an effort to expand the image field-of-view of the lake surface.

The results in this thesis show that this method can be used to reconstruct the lake surface for variable conditions, be it ice or water. The lidar turns out to be a good sensor to identify water from ice, simply from the presence or absence of a returned intensity point. There are almost no returns from regions with water, unless the water is choppy. The actual lidar points overlaid onto the camera images help visually locate the regions with water, especially if it was not immediately recognizable in the image.

Camera-lidar sensor fusion is a good way to reconstruct the surface assuming there are consistent lidar returns from the surface. As shown by the water surface test, it may sometimes be necessary to then estimate the water level; here we used pseudo-lidar points. This problem may be mitigated through other means of obtaining the Z_C value in the future, such as a stereo camera setup.

8.2 Future work

For future work improvements can be made to improve the accuracy of the results. The various sources of error are the obvious target for this:

- **Calibrating the camera accurately.** More care can be taken when performing the calibration process to obtain more accurate camera parameters. This can be done by taking more checkerboard pattern images, having sharper images, making sure the pattern covers majority of the image and others as stated in the documentation for the camera calibration app.
- **Identify error source for forward projection.** To avoid using an approximation the error in the forward projection must be eliminated. Since the exact reason is speculative at this point, it should be investigated more.
- **Translation vector to change lidar point cloud origin.** The relative position vector used to translate the lidar point cloud from the lidar origin L_0 to the camera origin C_0 can be measured more accurately, perhaps using a device with more accuracy than a measuring tape. The lidar itself might be reoriented temporarily for distance-measuring for this calibration step.
- **Angles measured more accurately.** For tests the angles of the sensors must

be measured more accurately. During the data campaigns, after setting up the sensors the angles were set using a digital angle finder. There is likely human error in the measurement due to the difficulty in measuring it when set up at the test site, especially during the cold winter tests. There were also tests where the dock was frozen over with ice and uneven, but during the processing of data, the assumption was made that the sensor suite was level because there was no measurement made of the tilt of the setup on this uneven surface. These measurements need to be made for future tests and be taken with a measuring device allowing for more accurate measurements.

Future work for the processing of the data include:

- **Improve interpolation.** Write own algorithm for interpolation instead of using built-in function, to more accurately interpolate the lidar data and cover the entire image.
- **Improve lidar point cloud translation.** As talked about in Section 4.2 the overlaid point cloud can sometimes have points from behind an object laid on top of points in front of the object. This must be corrected for and the points that should not be visible to the camera due to object blocking them, have to be removed.
- **Improving the water condition surface estimation.** A better method to estimate the mean surface from the sparse lidar returns off the water could be made. Perhaps instead of a standard average of the points, a weighted average could be taken or points too far out could be removed entirely.

In the longer term, the data sets collected may be used for new methods including

- **Use stereo camera setup.** Due to lack of lidar returns from the water surface, use a stereo camera setup to obtain depth information the perform the backward projection. This will not be affected by the presence of water or ice. The lidar can remain as an additional sensor which can be used to identify the ice and water from its returns or lack thereof respectively. This configuration will require the cameras to be placed sufficiently far apart to be resolve surface variations.
- **Create a timelapse of the surface.** Multiple frames from the data must be reconstructed to produce a time lapse of the surface while plotting GPS specular points on top. The points would move across the surface as satellites move across the sky. Time stamps could be identified for when the points pass over boundaries of ice and water to be used for the GPS processing.
- **Merging data from 3 camera setup.** The backward projection of the surface from each of the three cameras must be combined to create a single map of the entire surface in 3D coordinates.

This work, although currently meant to validate the GNSS-R signal analysis, could be used by itself for applications requiring water and ice differentiation. This could be the case for autonomous vehicles that need to obtain road surface conditions or even geographic mapping of large bodies of water. This work simply sets the foundation. The future possibilities are endless.

APPENDIX A

CAMERA SPECIFICATIONS

The cameras used for this were were IP cameras generally used for security purposes. GW 5037 IP was the central camera in the setup, designated as Camera 1, shown in Figure A.1(a). The specifications as provided by the manufacturer of this camera are provided in Table A.1 [23]. GW 5050 IP was the second model used, shown in Figure A.1(b). Two of this model were added to the sensor setup, designated as Camera 2 and Camera 3, to provide wider FOV coverage of the lake surface. Specifications as provided by the manufacturer of this camera are provided in Table A.2 [24].

(a) GW 5037 IP[25]

(b) GW 5050 IP [24]

Figure A.1. Photos of the Cameras used for this work.

Table A.1. GW 5037 IP Camera Specifications

Specification	Description
color	White
IP Rating	IP66
Video Encoding	H.264/H.265
Image Sensor	1/3” Progressive Scan CMOS
IP Camera Resolution	5 Megapixel
Video Bitrate	256Kbps 8192Kbps
Lens	3.6mm
Frame Rate	5MP @ 15FPS, 4MP @ 20FPS, 3MP @ 30FPS, 2MP @ 30FPS
Min Illumination	0.1Lux@(F1.2 AGC ON), 0 Lux With IR
IR Distance	75ft IR (18pcs 3rd Gen SMD LED 850nm)
Backlight Compensation	BLC
Wide Dynamic Range	Digital WDR
Elelctronic Shutter Speed	Auto/Manual 1/60s 1/100,000s
Built-In Audio	No
Two-way Audio	No
Network Interface	10/100M PoE
Network Protocol	DDNS, DHCP, DNS, FTP, HTTP, NTP, ONVIF, P2P, RTSP, SMTP, TCP/IP, uPnP, URL
Remote Software	Danale
Current Consumption	6W
Dimension	2.59” X 2.59” X 6.77”
Power Supply	Input 100 240V, 12V Optional

Table A.2. GW 5050 IP Camera Specifications

Specification	Description
Image sensor	1/2.8” 5MP IMX335 CMOS
Effective Pixels	2592(H) $\times$ 1944 (V)
Electronic Shutter	AUTO, 1/25s 1/100000s
Min. Illumination	0.01Lux@F1.2(AGC ON), 0Lux IR on
Day/Night	Auto/Color/(B/W)/Timing
WDR	Digital WDR
White Balance	Auto/Manual
AGC/BLC/HLC	Support
DNR	2D/3D DNR
Other Features	Multi-lines OSD, Motion Detection, Privacy Mask, Mirror
Video Standard	H.264/H.265
Video Resolution	Main Stream: 15fps @ 5MP (2592$\times$1944), 25fps @ 4MP/3MP, 30fps @ 1080P/720P Sub Stream: D1/VGA(640$\times$480)/360P/QVGA @ 30fps
Video Bitrates	32Kbps – 8Mbps, VBR/CBR
Audio Standard	G.711-u/G.711-a
IR LED	20 pcs small SMD IR Leds
IR Distance	99ft-131ft
Focal Length	2.8-12mm varifocal lens
Optional Function	POE, Audio, etc.
Zoom	4x Optical Zoom
Protocol	HTTP/RTSP/FTP/SMTP/DHCP/NTP/NFS, etc.
P2P	Yes
Web Access	IE , Firefox (32bit esr), etc.
Media	CMS, Android, IOS
ONVIF	2.6 compatible
Network Port	1-RJ45, 100Mbps, POE optional
Power Supply	12 VDC $\pm$ 10%
Power Consumption	$<$ 8 W
Operating Temperature	-30°C to +60°C, 10%-90%RH
Resolution	5 Megapixel
Camera Style	Bullet
Color	White

APPENDIX B
LIDAR SPECIFICATIONS

The lidar used in this work was the VLP-16, also known as the ‘Puck.’ Figure B.1 shows a photo of the sensor. It is a standard lidar model widely used for many autonomous vehicle and drone applications. The specifications are summarized in Table B.1 and Table B.2 [26]. Dimensions of the sensor are shown in Figure B.2.

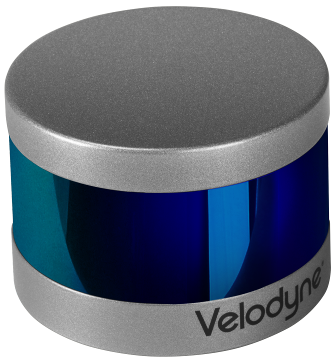


Figure B.1. Photo of the VLP-16 Lidar used for this work [26]

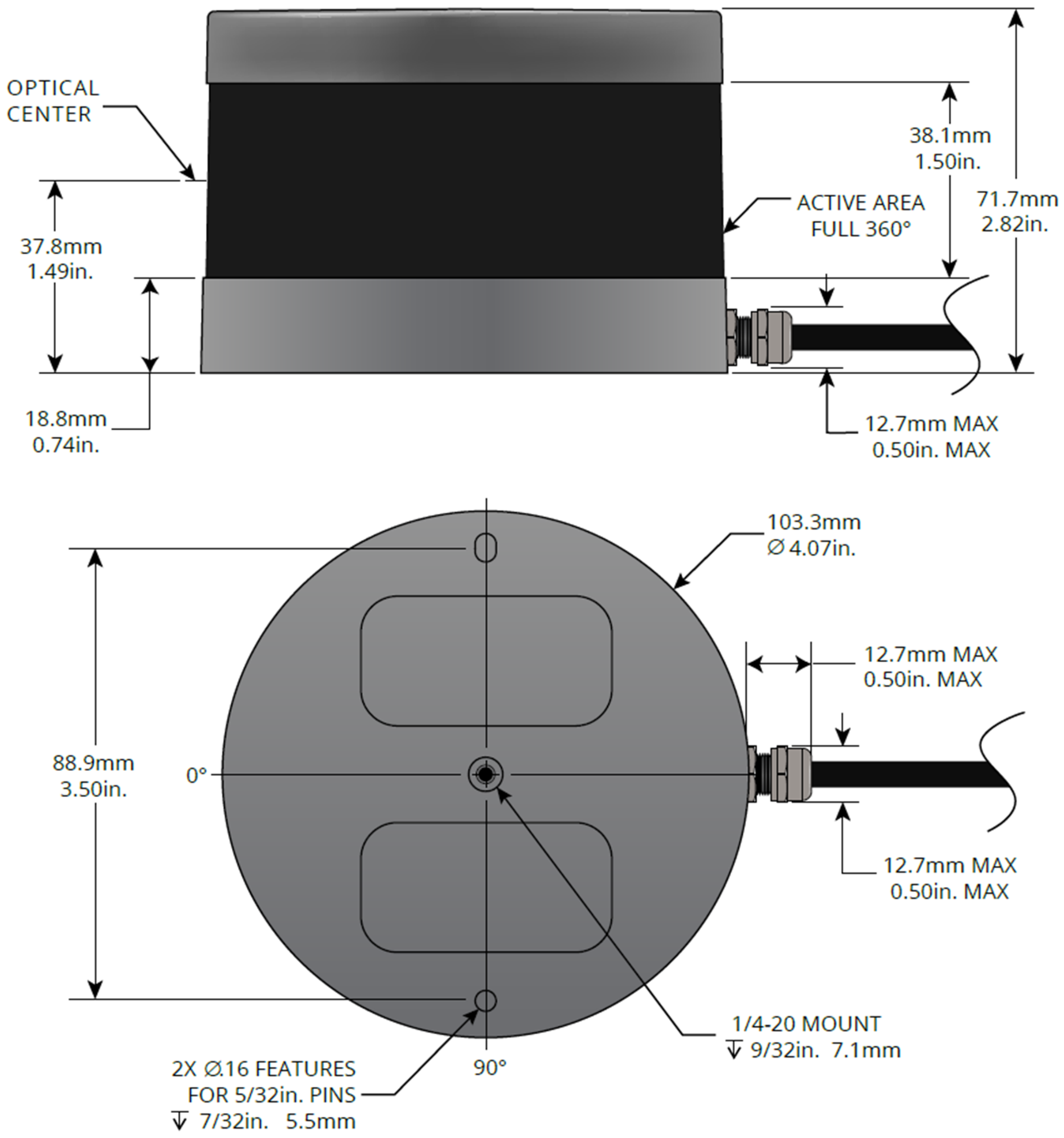


Figure B.2. Dimensions of the VLP-16 Lidar [26]

Table B.1. VLP-16 ‘Puck’ LiDAR Specifications

Specification	Description
Sensor:	
Channels	16
Measurement Range	100 m
Range Accuracy	Up to ±3 cm
Field of View (Vertical)	+15.0° to -15.0° (30°)
Angular Resolution (Vertical)	2.0°
Field of View (Horizontal)	360°
Angular Resolution (Horizontal)	0.1° – 0.4°
Rotation Rate	5 Hz – 20 Hz
Web Server	Integrated for Easy Monitoring and Configuration
Laser:	
Laser Product Classification	Class 1 Eye-safe per IEC 60825-1:2007 & 2014
Wavelength	903 nm
Mechanical/Electrical/Operational:	
Power Consumption	8 W
Operating Voltage	9 V – 18 V (with Interface Box and Regulated Power Supply)
Weight	Approx 830 g (without Cabling and Interface Box)
Dimensions	See Figure B.2
Environmental Protection	IP67
Operating Temperature	-10° C to +60° C
Storage Temperature	-40° C to +105° C

(continued in Table B.2 on next page)

Table B.2. VLP-16 ‘Puck’ LiDAR Specifications (continued)

Specification	Description
Output:	
3D Lidar Data Points Generated	Single Return Mode:
	Approx 300,000 points per second
	Dual Return Mode:
	Approx 600,000 points per second
Connection	100 Mbps Ethernet
UDP Packets Contain	Time of Flight Distance Measurement
	Calibrated Reflectivity Measurement
	Rotation Angles
	Synchronized Time Stamps
	(μs resolution)
GPS	$GPRMC and $GPGGA NMEA Sentences
	from GPS Receiver (GPS not included)

APPENDIX C

CALCULATING THE CAMERA FOV

Cameras have a FOV from which they capture incoming light rays from the environment. This FOV is dependent on the construction of the camera and its lenses. This work had to remove the lens distortion from the captured images, thereby changing the actual FOV of the new cropped images compared to the original images. Thus, the FOV obtained in the camera specifications could not be used directly. The steps involved to determine the FOV after distortion removal are as follows:

1. Mount a checkerboard pattern, with known dimensions of the squares, on a wall or other vertical surface. Either tape it to the wall or make sure it is placed vertically using a level.

2. Point the camera at the pattern, making sure it is pointed normal to the plane of the pattern.

3. Adjust the distance between the camera and the pattern until the pattern covers the entire image generated by the camera.

4. Note down the perpendicular distance from the pattern to the camera and save the image from the camera.

5. Undistort the generated image as explained in Section 3.2.

6. Since the size of the pattern boxes is known, use the visible checkerboard squares to determine the length of the pattern visible both horizontally and vertically in the undistorted image.

7. Use Equation C.1 to calculate the horizontal FOV (HFOV) in degrees, where l_H is the length of the checkerboard pattern visible horizontally in mm and d_C is the perpendicular distance from the pattern to the camera in mm.

$$\text{HFOV} = 2\ \tan^{-1} \frac{l_H/2}{d_C} \tag{C.1}$$

8. Use Equation C.2 to calculate the Vertical FOV (VFOV) in degrees, where l_V is the length of the checkerboard pattern visible vertically in mm and d_C is the perpendicular distance from the pattern to the camera in mm.

$$\mathrm{VFOV} = 2\,\tan^{-1}\frac{l_V/2}{d_C} \tag{C.2}$$

Figure C.1 shows two of the cameras used in this work facing a checkerboard pattern, in the process of determining their FOV. The checkerboard pattern is taped to a sheet of cardboard which is propped up to be vertical using a spirit level, seen on the top of the cardboard. The cameras are placed onto a horizontal table.

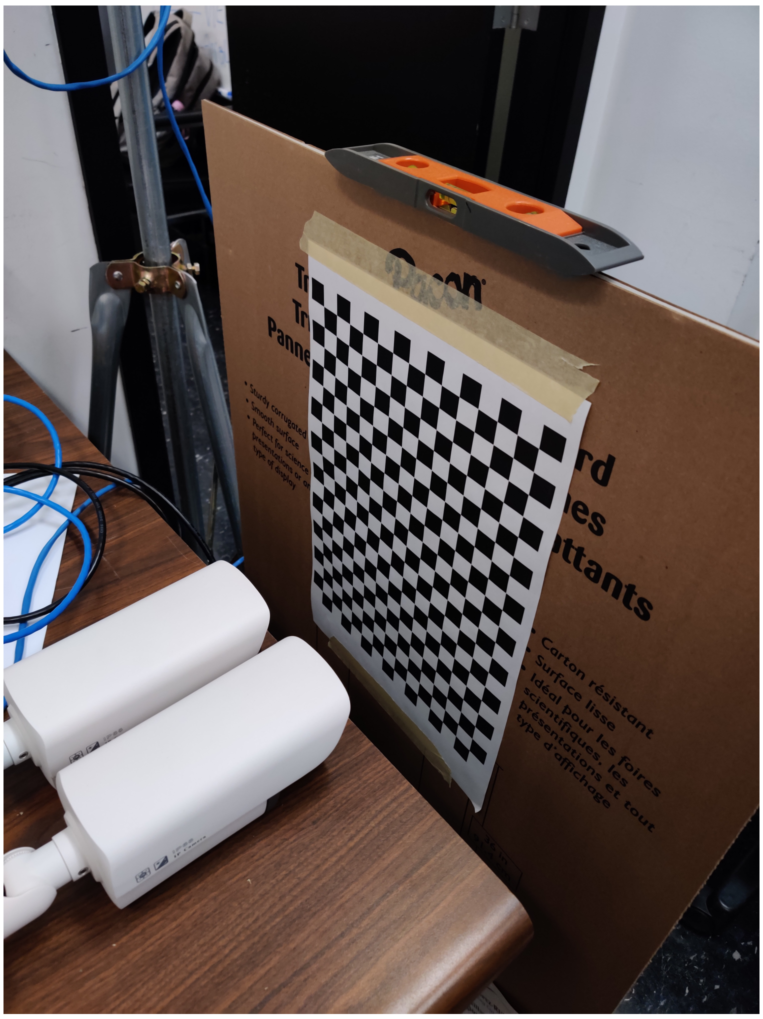

• Carton résistant
• Surface lisse
• Idéal pour les foires scientifiques, les présentations et tout type d'affichage

Figure C.1. Cameras set in front of a checkerboard pattern to calculate FOV

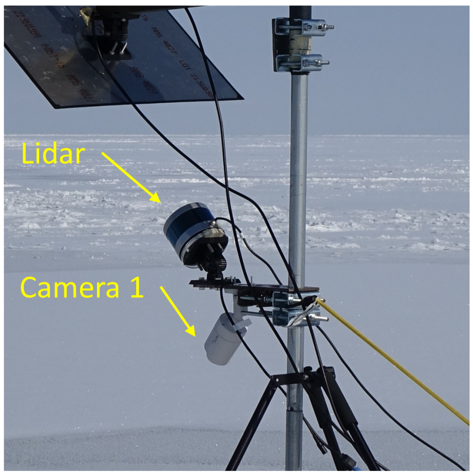
APPENDIX D

SENSOR SETUP

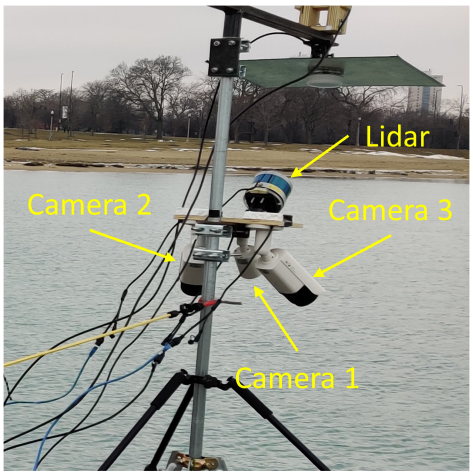
The setup of the camera and lidar sensors changed slightly throughout the project. For the cases shown in this documentation, there were primarily two setups. Setup 1 involved one camera and one lidar, as seen in Figure D.1(a). The arrangement of the sensors in this setup is explained through the angles listed in Table D.1. Setup 2 was designed to increase the camera coverage of the lake and involved three cameras and one lidar, as shown in Figure D.1(b). The data from each camera was independently forward projected with the lidar data to produce its own reconstruction of the environment. The arrangement of the sensors for Setup 2 is listed in Table D.1.

The angles listed here are explained further along with the corresponding coordinate axes in Section 3.3. For the lidar, el_L is the Elevation angle of the lidar $\hat{y}_L$ axis with respect to the Boom $\hat{b}$ axis and lidar az_L is the Azimuth angle of the $\hat{y}_L$ axis with respect to the Boom $\hat{b}$ axis. For the cameras, el_C is the Elevation angle of the camera $\hat{z}_C$ axis with respect to the Boom $\hat{b}$ axis and az_C is the Azimuth angle of the camera $\hat{z}_C$ axis with respect to the Boom $\hat{b}$ axis. All angles are in degrees.

When setting up the hardware the camera and lidar mounts have different methods of setting the angles. The lidar mount has a few possible angled positions which can be fixed by sliding a locking pin in.The angle of the mount is set before attaching the lidar onto it and the angle will not change unless the locking pin is removed again. The camera mounts, on the other hand, have a swivel joint with indicator clicks every 9°. The camera is rotated to the correct orientation and is then left in that position. Since the camera mounts can have their angles altered during the setup process at the test site through vibrations or accidental force, they are all checked again before beginning data collection using a digital angle finder.



(a) Setup 1



(b) Setup 2

Figure D.1. The different hardware setups for Lidar and Camera. Photo credit: David Stuart

Table D.1. Camera-Lidar hardware Setup 1

	Lidar		Camera 1	
Azimuth (deg)	az_L	0	az_C	0
Elevation (deg)	el_L	-45	el_C	-45
Misalignment (deg)			δ	3.2

Table D.2. Camera-Lidar hardware Setup 2

	Lidar		Camera 1		Camera 2		Camera 3	
Azimuth (deg)	az_L	0	az_C	0	az_C	-25	az_C	25
Elevation (deg)	el_L	-45	el_C	-45	el_C	-36	el_C	-36
Misalignment (deg)			δ	3.2	δ	0	δ	0

APPENDIX E
DATA CAMPAIGNS

Since the inception of this project many data campaigns have been performed to collect sensor data. The following table highlights some of the details regarding those data campaigns. The latitude and longitude position of the test locations was extracted from Google Maps after identifying the setup location on the map. The lake surface conditions were determined visually. Water surface conditions had free flowing water with waves. Ice surface conditions had a frozen surface with no movement; in some cases ice was also covered with snow. The mixed ice and water surface conditions had water regions with patches of frozen ice floating on the surface.

For the camera-lidar hardware setup (last column of Table E.1) refer to Appendix D. It should be noted that for Test 1_East and 1_West, although both lidar and camera were present, only the lidar was functional and collected data. Figure E.1 shows the locations of these data campaigns marked in yellow crosses with labels to indicate which data campaign tests took place.

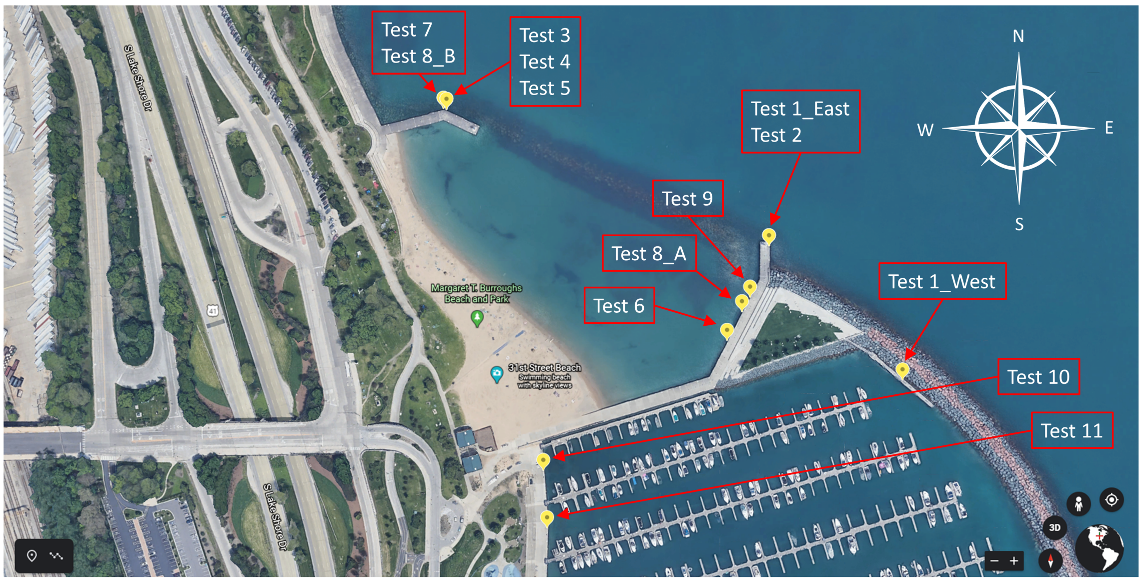


Figure E.1. All Data campaign test locations

The main GNSS-R hardware was constant throughout the many data cam-

Table E.1. Details of the GNSS-R Data campaigns

Data Campaign	Date	Latitude	Longitude	Lake surface	Camera-lidar Setup
1_East	Jan 23,2018	41.83978 N	87.60421 W	Water	1
1_West	Jan 23,2018	41.83893 N	87.60306 W	Ice	1
2	Feb 22, 2018	41.83978 N	87.60421 W	Water	1
3	Mar 2 2018	41.84066 N	87.60698 W	Water	1
4	Mar 9 2018	41.84066 N	87.60698 W	Water	1
5	Jul 5, 2018	41.84066 N	87.60698 W	Water	1
6	Jan 27, 2019	41.83918 N	87.60457 W	Water	1
7	Feb 1, 2019	41.84066 N	87.60701 W	Ice	1
8_A	Feb 19, 2019	41.83936 N	87.60444 W	Ice	1
8_B	Feb 19, 2019	41.84066 N	87.60701 W	Ice	1
9	Jan 31, 2020	41.83946 N	87.60437 W	Water	2
10	Feb 14, 2020	41.83835 N	87.60615 W	Ice & Water	2
11	Feb 21, 2020	41.83798 N	87.60612 W	Ice & Water	2

paigns. There had been changes such as the addition of cameras, changing GPS gains and even changing some of the sensor angles. Figure E.2 shows an example sensor suite setup for the data campaign number 7.

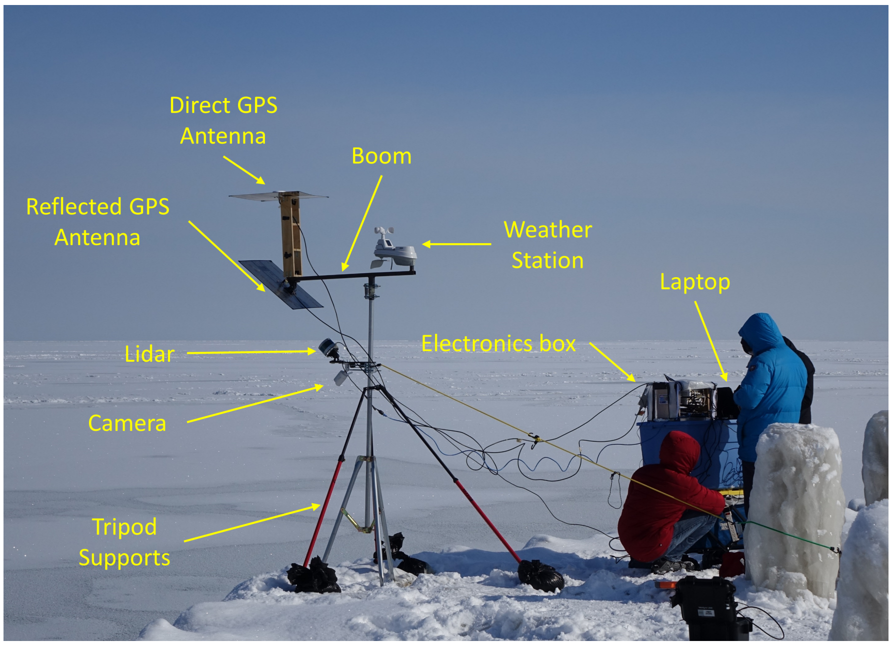


Figure E.2. The full GNSS-R Sensor Suite. Photo credit: David Stuart

BIBLIOGRAPHY

[1] J. L. Garrison, A. Komjathy, V. U. Zavorotny, and S. J. Katzberg, "Wind speed measurement using forward scattered GPS signals," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 1, pp. 50–65, 2002.

[2] Y. S. V. Sreenivash and S. Datta-Barua, "Automated ionospheric scattering layer hypothesis generation for detected and classified auroral global positioning system scintillation events," *Radio Science*, vol. 54, 2019.

[3] R. Parvizi, "A novel remote sensing system using reflected GNSS signals," Ph.D. dissertation, Illinois Institute of Technology, 2020.

[4] S. Datta-Barua, R. Parvizi, E. Donarski, S. Stevanovic, N. Wang, K. Herron, and B. Pervan, "Great lake surface characterization with GNSS reflectometry," in *Proceedings of the 29th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2016)*, 2016, pp. 872–880.

[5] R. Parvizi, B. Pervan, and S. Datta-Barua, "Surface ice reflectometry with a dedicated GNSS receiver I: Hardware and field test," N.D., under review.

[6] R. Parvizi and S. Datta-Barua, "Surface ice reflectometry with a dedicated GNSS receiver II: Signal processing enabling acquisition from reflected signals," N.D., under review.

[7] R. Parvizi and S. Datta-Barua, "De-noising GNSS-Reflectometry measurements from a freshwater surface," in *2019 URSI Asia-Pacific Radio Science Conference (AP-RASC)*. IEEE, 2019, pp. 1–1.

[8] R. Parvizi, J. Henry, N. Honda, E. Donarski, B. S. Pervan, and S. Datta-Barua, "Coordination of GNSS signals with LiDAR for reflectometry."

[9] R. Parvizi, H. S. Zadeh, L. Pan, B. Pervan, and S. Datta-Barua, "Multi-sensor study of lake michigan's surface using GNSS-Reflectometry," in *Proceedings of the 31st International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2018)*, 2018.

[10] P. Wei, L. Cagle, T. Reza, J. Ball, and J. Gafford, "LiDAR and camera detection fusion in a real-time industrial multi-sensor collision avoidance system," *Electronics*, vol. 7, no. 6, p. 84, 2018.

[11] V. D. Silva, J. Roche, and A. Kondoz, "Robust fusion of LiDAR and wide-angle camera data for autonomous mobile robots," *Sensors*, vol. 18, no. 8, p. 2730, 2018.

[12] J. Li, X. He, and J. Li, "2D LiDAR and camera fusion in 3D modeling of indoor environment," *2015 National Aerospace and Electronics Conference (NAECON)*, 2015.

[13] K. Banerjee, D. Notz, J. Windelen, S. Gavarraju, and M. He, "Online camera LiDAR fusion and object detection on hybrid data for autonomous driving," *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[14] K. Korzeniowska, "Modelling of water surface topography on the digital elevation models using LiDAR data," in *AGILE'2012 International Conference on Geographic Information Science*, 2012.

[15] H. Steuer, U. Schäffler, and A. Gross, “Detection of standing water bodies in lidar-data,” in *Earth Observations of Global Changes EOGC 2011*, 2011.

[16] A. Baldridge, S. Hook, C. Grove, and G. Rivera, “The ASTER spectral library version 2.0,” *Remote Sensing of Environment*, vol. 113, no. 4, p. 711–715, 2009.

[17] H. S. Lim, M. Z. M. Jafri, K. Abdullah, and M. N. A. Bakar, “Water quality mapping using digital camera images,” *International Journal of Remote Sensing*, vol. 31, no. 19, p. 5275–5295, Oct 2010.

[18] What is camera calibration? - MATLAB & simulink. [Online]. Available: https://www.mathworks.com/help/vision/ug/camera-calibration.html [Accessed: March 20, 2020].

[19] Z. Tang, R. Grompone von Gioi, P. Monasse, and J. Morel, “A precision analysis of camera distortion models,” *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2694–2704, 2017.

[20] Velodyne Lidar. (2020, March) Deepen delivers annotation excellence for autonomous development. [Online]. Available: https://velodynelidar.com/blog/deepen-annotation-process-autonomous-development/ [Accessed: March 24, 2020].

[21] Single camera calibrator app - MATLAB & simulink. [Online]. Available: https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html [Accessed: March 15, 2020].

[22] Developing an algorithm for undistorting an image - MATLAB & simulink. [Online]. Available: https://www.mathworks.com/help/symbolic/examples/developing-an-algorithm-for-undistorting-an-image.html [Accessed: March 25, 2020].

[23] GW Security USA, private communication, 2019.

[24] GW5050IP 5MP HD-IP PoE 4x optical varifocal lens bullet security camera. [Online]. Available: https://www.gwsecurityusa.com/product/gw5050ip-hd-ip-poe-camera/ [Accessed: January 27, 2020].

[25] GW5037IP 5MP HD-IP PoE fixed lens bullet security camera. [Online]. Available: https://www.gwsecurityusa.com/product/gw5037ip-5mp-hd-ip-fixed-lens-poe-camera/ [Accessed: December 12, 2020].

[26] Downloads, Velodyne Lidar. [Online]. Available: https://velodynelidar.com/downloads/ [Accessed: June 13, 2019].