ENTER YOUR TITLE

by

ENTER YOUR NAME

Enter Your Previous Degrees

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

ENTER YOUR DEGREE NAME

Enter Your Department Name
Enter Your College Name

KANSAS STATE UNIVERSITY
Manhattan, Kansas

Graduation Year

Approved by:

Major Professor
Enter Your Major Professor's Name

Copyright

Abstract

Enter the text of your abstract in the abstract.tex file. Be sure to delete the text below before you submit your ETDR.

This template uses a separate file for each section of your ETDR: title page, abstract, preface, chapters, reference, etc. This makes it easier to organize and work with a lengthy document. The template is configured with page margins required by the Graduate School and will automatically create a table of contents, lists of tables and figures, and PDF bookmarks.

The file etdrtemplate.tex is the "master" file for the ETDR template. This is the file you need to process with PDFLaTeX in order to produce a PDF version of your ETDR. See the comments in the etdrtemplate.tex and other files for details on using the template. You are not required to use the template, but it can save time and effort in making sure your ETDR meets the Graduate School formatting requirements.

Although the template gives you a foundation for creating your ETDR, you will need a working knowledge of LaTeX in order to produce a final document. You should be familiar with LaTeX commands for formatting text, equations, tables, and other elements you will need to include in your ETDR.

Table of Contents

List of Figures

List of Tables

Acknowledgments

Enter the text for your Acknowledgements page in the acknowledge.tex file. The Acknowledgements page is optional. If you wish to remove it, see the comments in the etdrtemplate.tex file.

Dedication

Enter the text for your Dedication page in the dedication.tex file. The Dedication page is optional. If you wish to remove it, see the comments in the etdrtemplate.tex file.

Preface

Enter the text for your Preface page in the preface.tex file. The Preface page is optional. If you wish to remove it, see the comments in the etdrtemplate.tex file.

Chapter 1

Background

address at transplanter? One-time shot if things mess up. Difficult to identify plants. Constant stopping, too much motion and difficult shading make its not ideal for images.

1.1 Similar Research

1.2 Coordinate Systems

A key step in any mapping process is the correct choice of coordinate frames. In the context of mapping a coordinate system defines the location of an item with respect to another item. These coordinate systems, or frames of reference, are typically fixed to items that are significant in the mapping process which include the Earth, field, platform and cameras. The specific coordinate systems used in this research are discussed in the following sections.

1.2.1 Latitude and Longitude

A common frame that is fixed to the Earth is a spherical coordinate frame, with the angles referred to as latitude and longitude. However, since the Earth is not a sphere these angles are projected onto an ellipsoid called a datum. The datum that is used in this research is the World Geodetic System (WGS-84) which is the default datum used by the Global

Positioning System (GPS) ¡TODO cite?¿. The third dimension is altitude and is measured relative to the ellipsoid. Coordinates in the frame are denoted (Φ, λ, A).

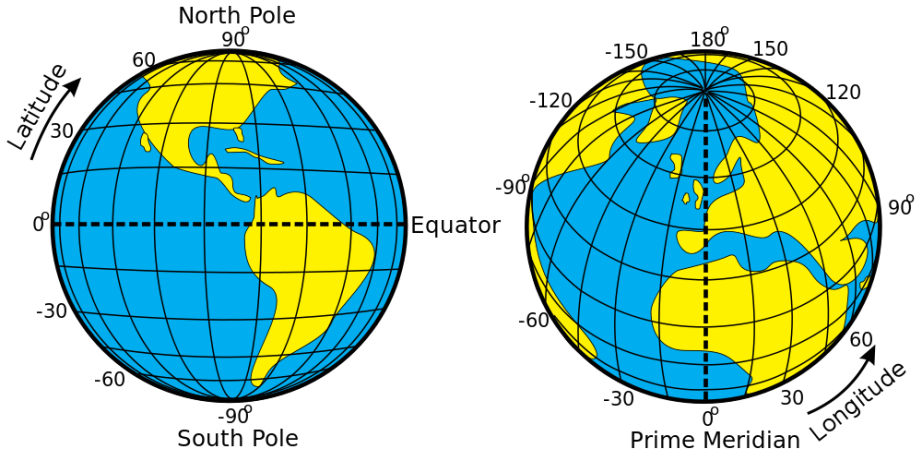

Figure 1.1: *Illustration of Latitude* (Φ) *and Longitude* (λ). *Obtained under the Creative Commons License from the Wikimedia Commons.*

1.2.2 Universal Transverse Mercator (UTM)

In order to describe the points on the Earth's surface in a two-dimension plane, as is desired for mapping, a projection model must be used on the angular latitude and longitude coordinates. The projection model used in this research is known as the Universal Transverse Mercator (UTM) which splits the Earth into 60 lateral bands, examples of which are shown in figure 1.2. In each zone an item is described by its easting, northing and altitude coordinates. However, due to how the platform coordinate system is defined it's beneficial to have the Z axis in the downward direction. This requires the easting and northing be switched to maintain a right-handed coordinate system. Therefore, UTM coordinates are described by (N, E, D). Another important modification is that the Z, or down, component is measured relative to the ground, not the WGS-84 ellipsoid.

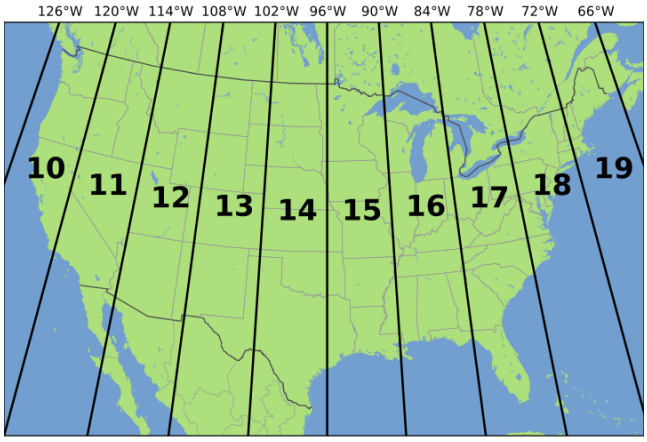

Figure 1.2: *Visualization of UTM zones over the United States. Obtained under the Creative Commons License from the Wikimedia Commons.*

1.2.3 Field Coordinate System

One issue with the easting and northing described by UTM is that rows in a fields are not always planted north and south or east and west. Many of the post-processing steps benefit from removing this relative field orientation. A new field coordinate system is defined where the y-axis runs parallel to the rows and increases in the planting direction of the first row, and the x-axis increases in the direction of increasing row numbers. The origin is selected so that all items in the field have positive x and y coordinates. Similar to UTM the units of this frame are meters. Coordinates in this frame are denoted (x_f, y_f, z_f).

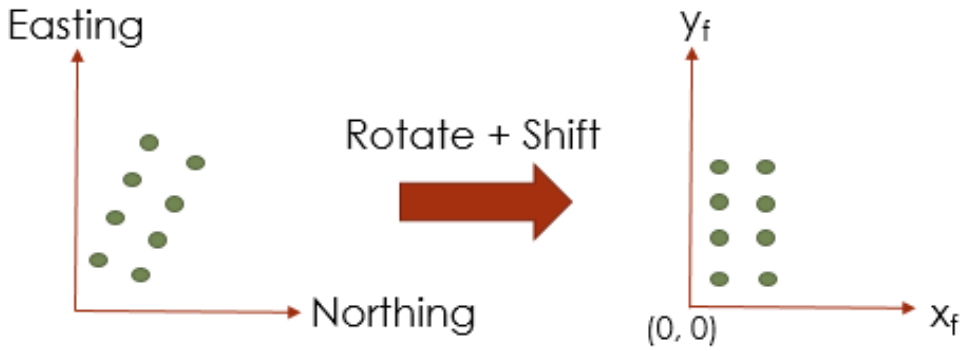

Figure 1.3: *Conversion from modified UTM coordinate frame (left) to field coordinate frame (right)*

1.2.4 Platform Coordinate System

Another useful coordinate system is one fixed to the platform holding the cameras. This coordinate system allows the relative spacing and orientation between the cameras and the GNSS ¡Todo already defined?¿ antenna to be specified. Also this coordinate system defines how the platform's orientation is defined in terms of Euler angles, which is useful for accounting for non-level camera orientation. The axes of this coordinate system are shown in the figure 1.5, which defines the x-axis out of the front of the platform, the y-axis out the right hand side and the z-axis orthogonal in the downward direction.

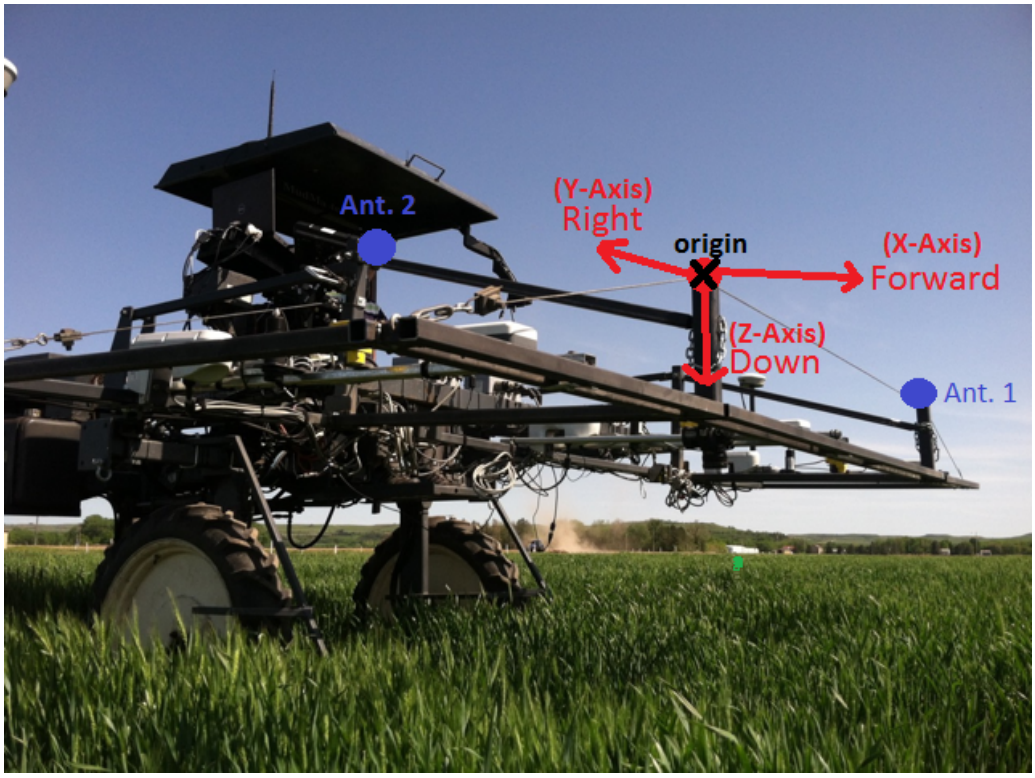

Figure 1.4: *Example of platform coordinate frame on a tractor systems with two GNSS antennas shown in blue. The origin is the average of these antenna locations and is shown as a black X.*

1.2.5 Camera Coordinate System

The camera coordinate system describes the locations of objects in the world relative to the camera. Typically a camera coordinate system is defined by the x-axis out of the right

hand side of the camera, the y-axis out of the bottom and the z-axis along the optical axis. ¡TODO cite¿ However, for this application an alternative camera coordinate system is defined which has the x-axis out of the top of camera, the y-axis out of the right side and the z-axis along the optical axis. This is so when the camera is mounted on the platform facing the ground with the top forward, the camera axes will align with the forward-right-down platform coordinate system. This makes the meaning of the Euler angles consistent, and is also enforced by the data collection program. The origin is defined to be located at a distance f in front of the imaging sensor along the optical axis, where f is the focal length of the lens. This is discussed more in the next section.

Figure 1.5: *Image showing camera coordinate axes.*

1.3 Coordinate Projections

An important step in the mapping process is converting between pixel and world coordinates, which is also the subject of many computer vision algorithms. Since pixels are described in $\mathbb{R}^2$ and world points in $\mathbb{R}^3$, this conversion requires a projection.

However, before this can be defined a model must be chosen which describes how points in the world are projected onto the imaging sensor. This section first presents the model used in this research and then discusses two equivalent methods that can be used for converting coordinates.

1.3.1 Projection Model

A common choice for a thin, convex lens is the central perspective imaging model shown in figure 1.6.

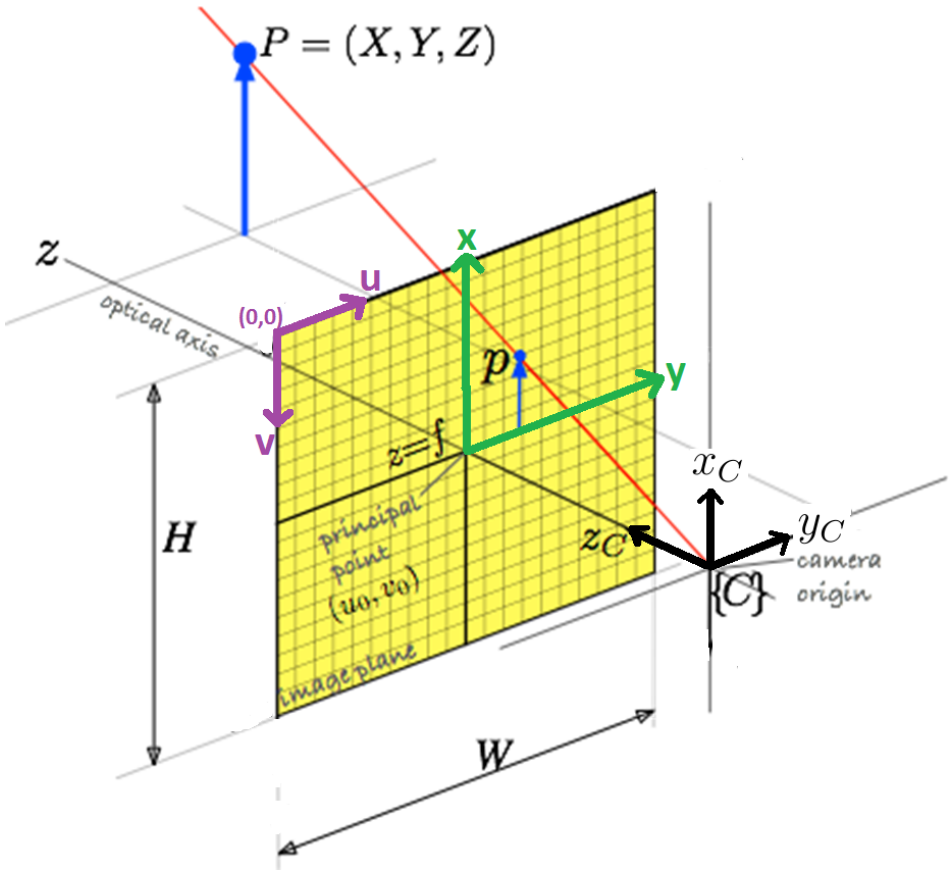

Figure 1.6: *Central perspective imaging model showing the image plane in yellow. Adapted from "Robotics, Vision and Control" 1st edition, by Peter Coorke, 2013, p.252. Modified to show pixel frame in purple, the sensor frame in green and switched the ordering of the camera axes.*

This model is based on the concept of an ideal lens that is treated as an equivalent pinhole, which implies the lens is perfectly focused and has no geometrical distortions. The ideal lens will have two focal points along the optical axis which can be seen in figure 1.7. The central perspective model defines the image plane to be at the focal point in front of the lens, which is the left focal point shown in the figure.

Points in the image plane can be described using two different coordinate frames. The first is the sensor frame and the second is the pixel frame. The origin of the sensor frame

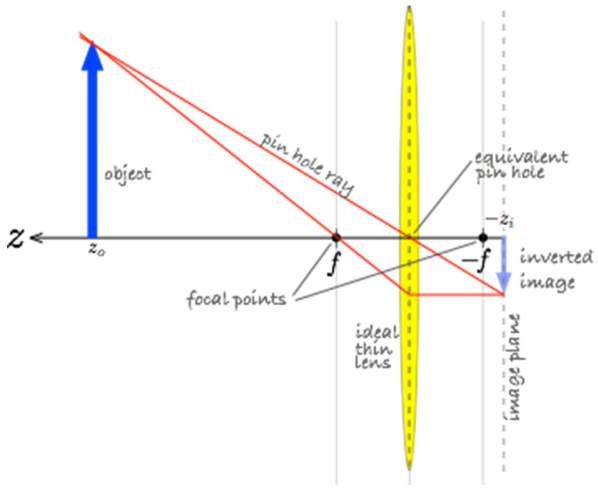

Figure 1.7: *Equivalent pin hole lens showing both focal points. Adapted from "Robotics, Vision and Control" 1st edition, by Peter Coorke, 2013, p.252*

is located at the point where the optical axis intersects the image plane. A point in this frame is described by (x, y), where the axes are in the same direction as camera frame and are shown in blue in figure 1.6.

On the other hand, the origin of the pixel frame is located at the top left of the image plane and the x-axis increases to the right and the y-axis increases downwards. Rather than (x, y) a pixel's location is denoted as (u, v). The center of the image sensor is referred to as the principal point and is denoted (u_0, v_0). Another key difference between these frames is the units. The sensor frame has units of millimeters while the pixel frame has units of pixels.

1.3.2 Projection Methods

The conversion between pixel and world coordinates can go both ways, and both are used in the post-processing pipeline. However, going from pixel to world coordinates is used more often and is slightly more challenging, so that is what is presented in this section. The conversion involves three frame transformations

$$\text{Pixel } (u, v) \rightarrow \text{Sensor } (x, y) \rightarrow \text{Camera } (X, Y, Z) \rightarrow \text{World } (N, E, DF)$$

This is referred to as a backwards projection, since it is projecting two-dimensional points back out into the world. As mentioned above this is more challenging because depth informa-

tion is lost during the forward projection as the image is captured. There are many ways of recovering this depth information, which typically use multiple images or cameras. However, the height of objects being mapped are relatively small and is not useful for this mapping application. For that reason an assumption is made that every point in the world frame has a height of zero. This reduces the world to a plane and results in a $\mathbb{R}^2$ to $\mathbb{R}^2$ mapping which is possible with a single image.

There are two different methods for performing this backwards projection. The first method discussed operates in Euclidean space and is more intuitive to understand. The second method builds on this method by instead working in Projective space which produces a more efficient, but equivalent, conversion. Both of these methods use the central perspective model and both make the same assumptions about the camera and lens. These are:

- The lens does not cause any distortion.
- There is no skew or displacement in how the image sensor is positioned within the camera.
- The focal length exactly matches the manufacturer's specification.

All of these assumptions are false to some extent, and it's possible to correct these assumptions using certain camera calibration procedures ¡TODO cite¿. However, for the system presented in this research these effects are considered negligible. Another assumption is the camera's orientation is relative to the ground. However, typically orientation angles are measured with respect to the gravity vector, which on hills will not be orthogonal to the ground. If the mapping is done on a hill then the simplest solution is to make sure the camera's orientation about the platform's X and Y axes is zero.

Euclidean Space

The first transformation is to convert pixel coordinates to sensor coordinates. Using the frame definitions shown in figure 1.6, this can be represented in matrix form as

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 & -\rho_h \\ \rho_w & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} \rho_h v_0 \\ \rho_w u_0 \end{bmatrix}$$

where ρ_w and ρ_h are the width and height of each pixel in millimeters, and u_0 and v_0 are the coordinates of the principal point in pixels. The next transformation is to convert from sensor coordinates to camera coordinates. It's clear from the projection model that if the sensor coordinates are combined with the focal length f as

$$(x, y, f)$$

then the result is a vector in the camera frame, but with units of millimeters rather than meters. The last frame transformation is to describe this vector in terms of the world frame. This can be accomplished by a rotation matrix R

$$\begin{bmatrix} n \\ e \\ d \end{bmatrix} = R * \begin{bmatrix} x \\ y \\ f \end{bmatrix} \tag{1.1}$$

where this rotation matrix is composed from the active, extrinsic rotations about the worlds north, east and down axes in that order, by angles ϕ (roll), θ (pitch) and ψ (yaw). These are the Tait-Bryan angles of the camera frame with respect to the world frame. Technically this is a passive rotation by negative angles since the vector is changing frames, but using active rotations avoids many double negatives.

$$
\begin{aligned}
R &= R_z(\psi) * R_y(\theta) * R_x(\phi) \qquad (1.2)\\
&= \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & cos(\phi) \end{bmatrix} \\
&= \begin{bmatrix} \cos(\theta)\cos(\psi) & \sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi) & \sin(\phi)\sin(\psi) + \cos(\phi)\sin(\theta)\cos(\psi) \\ \cos(\theta)\sin(\psi) & \cos(\phi)\cos(\psi) + \sin(\phi)\sin(\theta)\sin(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) \\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix}
\end{aligned}
$$

Lowercase coordinates (n, e, d) in equation 1.1 are used to designate that these components are in the north, east and down directions, but the units are still in millimeters and this does not represent the world point yet. The position of this vector in the world frame is given by the camera's world position

$$
T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \begin{bmatrix} N_{cam} \\ E_{cam} \\ D_{cam} \end{bmatrix} \qquad (1.3)
$$

The actual world point can be found by finding the intersection of this vector, (n, e, d), with the flat plane of the Earth given by the equation $D = 0$.

This can be done by parameterizing the vector with S as

$$
\begin{bmatrix} N \\ E \\ D \end{bmatrix} = S \begin{bmatrix} n \\ e \\ d \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \qquad (1.4)
$$

plugging in the third component, D, into the plane equation and solving for

$$
S = -T_z/d,
$$

which can be plugged back into equation 1.4. If d is zero then the position vector is parallel

to the Earth's surface and will never intersect it.

While this method is easy to visualize, it requires multiple steps for each pixel that must be converted to world coordinates. A more efficient method is presented in the next section.

Projective Space

An ideal solution would be to describe the backwards projection from the pixel plane to the world plane as a single matrix that can be applied in one step. In order to do this, coordinates in each frame must be converted into the Projective space which adds an extra coordinate. This coordinate represents scale, and a set of coordinates in this space is referred to as homogeneous coordinates.

There are many benefits to using homogeneous coordinates which is why they are commonly used in computer vision. These benefits include

1. performing translations and rotations in a single matrix.

2. avoiding unnecessary division in intermediate calculations which is typically slower than other operations.

3. representing a coordinate at infinity with real numbers which is true when the extra coordinate is zero.

In order to differentiate between regular Euclidean coordinates, homogeneous coordinates are followed by an apostrophe. For example coordinates in the pixel frame can be described in Projective space as (u', v', w') where

$$u' = u * w' \text{ and } v' = v * w'.$$

In backwards projection the pixel coordinates (u, v) are what is known so w' is always set to 1 so it does not change the scale. In order to convert from pixel to sensor coordinates the same relationship using the pixel sizes and principal point are used, however when homogeneous coordinates are substituted this relationship can be described in a single matrix

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 0 & -\rho_h & \rho_h v_0 \\ \rho_w & 0 & \rho_w u_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}$$

This is typically referred to as the inverse parameter matrix, or K^{-1}. The second transformation is to convert sensor coordinates to camera coordinates. A consequence of defining the origin of the camera frame to be the location of the equivalent pin-hole is that all incoming rays of light converge to the origin of the camera frame. This leads to the simple relationship using the focal length f of

$$x = fX/Z \text{ and } y = fY/Z$$

which can easily be derived by similar triangles. When described as homogeneous coordinates this relationship becomes

$$x' = fX,\ y' = fY \text{ and } z' = Z$$

or in matrix form

$$\begin{bmatrix} X_u \\ Y_u \\ Z_u \end{bmatrix} = \begin{bmatrix} 1/f & 0 & 0 \\ 0 & 1/f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}.$$

The subscript u denotes that these coordinates in the camera frame are unscaled, similar to how (x, y, f) is not correctly scaled in the Euclidean method. Also note that (X_u, Y_u, Z_u) are not homogeneous coordinates. This matrix is referred to as the inverse camera matrix, or C^{-1}. The last frame transformation is to go from the camera to world frame and correctly scale the position vector. This can be done in one step if homogeneous world coordinates (N', E', D', S') are used. In terms of the rotation matrix R and translation vector T defined in equations 1.2 and 1.3

$$\begin{bmatrix} R^{-1} & -T \end{bmatrix} \begin{bmatrix} N' \\ E' \\ D' \\ S' \end{bmatrix} = \begin{bmatrix} X_u \\ Y_u \\ Z_u \end{bmatrix}.$$

If each element in R is denoted by its row, i, and column, j, as R_{ij}, and since the inverse of an orthogonal matrix is equal to its transpose then this is expanded to

$$\begin{bmatrix} r_{11} & r_{21} & r_{31} & -T_x \\ r_{12} & r_{22} & r_{32} & -T_y \\ r_{13} & r_{23} & r_{33} & -T_z \end{bmatrix} \begin{bmatrix} N' \\ E' \\ D' \\ S' \end{bmatrix} = \begin{bmatrix} X_u \\ Y_u \\ Z_u \end{bmatrix}.$$

However, this matrix is clearly not invertible and the assumption that $D = 0$ must be enforced. Since $D' = D/S'$ then D' is also zero, and it can be removed along with the third column of the matrix. After inverting this results in

$$\tilde{p} = \begin{bmatrix} N' \\ E' \\ S' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{21} & -T_x \\ r_{12} & r_{22} & -T_y \\ r_{13} & r_{23} & -T_z \end{bmatrix}^{-1} \begin{bmatrix} X_u \\ Y_u \\ Z_u \end{bmatrix}.$$

Even though an entire column of the rotation matrix is discarded no information is lost as this column can be described as the cross product of the first and second columns. This inverted matrix is denoted ξ^{-1}, so that the entire projection can be represented as a single homography matrix

$$H = \xi^{-1} C^{-1} K^{-1}$$

.

Once (N', E', S') is computed the actual northing and easting is simply the inhomogeneous coordinates

$$N = N'/S' \text{ and } E = E'/S'$$

While it's not obvious, this result is indeed equivalent to the one derived in the first method. This is verified symbolically in Appendix TODO ref.

Chapter 2

System Design

The mapping system consists of the various components needed to capture, organize and locate images of the field, as well as any additional field items needed for identifying plants. The proper design of the mapping system is the most important step in the overall mapping process. Poor design leads to insufficient image quality, missing field coverage or improperly located images which no amount of post-processing can correct.

The first part of the system discussed is the markers used for plant identification, because the required size of these markers impose constraints on the rest of the system. Next the base platform and additional equipment, such as cameras, is presented along with reasoning about nominal parameters such as vehicle speed and the camera height above the ground. This chapter concludes by describing additional field markers which are not strictly necessary, but help improve the robustness of the mapping process.

2.1 Plant Identification

As mentioned in the introduction, the mapping process involves not just determining plant coordinates, but also assigning each plant to a group. Each plant group is referenced by a unique identifier, such as 1035. The method used in this research is to encode this information in a two-dimensional barcode that is placed at the beginning of each group of plants in the

field. Thus each barcode can be referenced only using its unique ID.

If a plant group needs to be planted in different parts of the field then a repetition letter is appended to the group number. For example, three codes containing the text 1035A, 1035B and 1035C all belong to plant group 1035. Since the two-dimensional barcodes are only placed at the beginning of the group it's critical to know the direction each row is planted in order to associate the correct set of plants with the plant ID. Codes could be placed on both sides of each group to remove this added challenge, but this doubles the amount of construction time, field debris and chance of missing a code. Instead the row direction is encoded in row markers which is discussed later in this chapter.

2.1.1 Code Format

For this research project, Quick Response (QR) codes were selected as the barcode format. This is a standardized format that was made publicly available by Denso Corporation over 20 years ago. It was first used for item tracking in the Japanese automotive industry, and has most recently become well known for encoding Uniform Resource Locators (URLs) for websites. ¡TODO ref article¿. Two important characteristics of this format is it can be read from any orientation, and it can still be read if part of the code is damaged. Various other types of formats, such as Aztec or Micro QR, can encode information in slightly less number of squares by restricting the character encoding, but offer less error correction. Also, at the time this research was conducted these alternate formats were not supported by any of the open-source readers that the researcher investigated.

Figure 2.1: *Comparison of 2D barcode formats encoding the text 1035 with high error correction. From left to right: Quick Response (QR), Aztec and Micro QR.*

2.1.2 Size Constraint

For fields with thousands of different plant groups it's not feasible to place these QR codes by hand. Therefore, they must fit through the deposit cylinders of the transplanter shown in figure ¡TODO ref experiment¿. In order to not get caught in the cylinders the codes cannot be larger than 2.5 centimeters in diameter. Since there needs to be a white margin around the actual QR code, the code itself ends up being roughly 2 centimeters. The QR code format is split into different levels which define how many squares make up the code. The first version is a grid of 21 by 21 squares and is shown in figure 2.1. This results in a maximum square size of only 1 millimeter. This small square size requires the use of a high resolution camera and lens.

2.1.3 Code Construction

The codes must be easy to produce due to the potentially large number of codes required for each field. The ¡TODO¿ program can be used to generate all QR codes at once, and a thermal printer can print the codes on pot labels rapidly. However, the pot labels need a solid base to stay upright and grounded during transplanting. The researchers at the Land Institute in Salina, Kansas, developed a biodegradable compost ¡TODO word this better¿ that the pot labels are inserted into. An example of one of these QR codes can be seen in figure 2.2.

Figure 2.2: *QR code printed on pot label and planted in field.*

2.2 Platform Design

There are many types of platforms that could be used for mapping. Aerial vehicles have the benefit of autonomously traversing the field without the possibility of running over plants. However, they are unable to provide external lighting or shading which is important when post-processing the images. In addition, the size constraint on QR codes would require low altitude flights at a constant altitude to keep the codes properly focused which is not easy to achieve. Higher altitude flights would be possible with a telescopic lens, but this increases cost, weight and most critically the effects of error in camera orientation. For these reasons only ground platforms are considered.

Two different types of ground platforms were investigated, a manual push-cart and a four-wheel robotic vehicle. The push-cart excels in it's simplicity, however for this thesis only the robotic platform is discussed. The main benefits of a robot is the ability to drive at a constant speed and the option to operate autonomously. Driving at a constant speed is important to ensure sufficient overlap between successive images, and self-driving vehicles remove much of the tedious work associated with imaging large fields.

2.2.1 Robotic Platform

The selected robotic platform is the Husky A2000 mobile robot made by Clearpath Robotics. The Husky is a four wheeled differential drive robot measuring 30 inches long and 27 inches wide. A custom C-channel structure was added to the top of the robot to enable it to image the field. This structure can be seen in figure 2.3. Attached to the front of the structure are two Canon 7D digital single-lens reflex (DSLR) cameras. Using two cameras allows two rows to be mapped at the same time.

TODO include max robot speed and payload.

On the back of the C-structure are two white, Trimble AG25 antennas which attach to a Trimble BX982 global navigation satellite system (GNSS) receiver. This receiver is mounted to the top of the robot.

Figure 2.3: *Husky A2000 mobile robot equipped with two cameras.*

2.2.2 GNSS Receiver

The BX982 receiver provides centimeter level accuracy when paired with a fixed base receiver broadcasting RTK correction signals. For this application the fixed base is a Trimble Ag542 receiver with a Trimble Zypher Geodetic antenna as shown in figure 2.4. This base station communicates with the SNB900 rover radio mounted to the robot over a 900 MHz radio link. The dual antenna design allows the robot to determine its heading, or yaw, to within approximately 0.1 degree. This accurate yaw is important for geo-locating plants and QR codes within images, as well as allowing the robot to operate autonomously as discussed in section 2.3.3.

2.2.3 Cameras and Lighting

The Canon 7D features an 18 megapixel (MP) sensor and is fitted with a fixed 20 millimeter focal length wide-angle lens. The camera contains an Advanced Photo System type-C (APS-C) sensor rather than a full frame 35mm sensor, which paired with the wide-angle lens gives a horizontal angle of view of 58.3 degrees and a vertical view of 40.9 degrees. As shown in figure 2.3 the cameras are mounted far out in front of the robot so that the wheels or front bumper do not show up in the image and reduce the effective field of view. In addition, each

Figure 2.4: *Ag542 base station mounted on tripod.*

camera is mounted with a 90 degrees yaw with respect to the platform so that the longer side of the image is aligned with the forward movement of the robot as seen in figure 2.5.

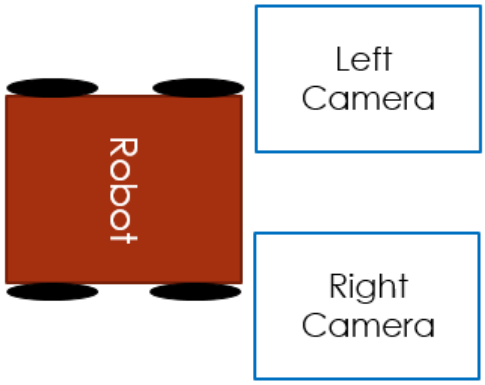

Figure 2.5: *Depiction of camera field of views relative to robotic platform.*

Talk about LED lights.

2.2.4 Determining Parameters

There are many parameters, such as camera height and vehicle speed, which are chosen to ensure quality images and sufficient field coverage. The most important to determine first is the camera height as that determines the resolution of the image. If the image is too low of

Figure 2.6: *Canon 7D and the external lighting LED bars.*

resolution then the QR codes will be unreadable. If each pixel could correspond to exactly one square on the QR code then the minimum resolution would be 1 pixel per millimeter, since each square is roughly 1 millimeter in size. In reality this is almost never the case, and the theoretical minimum is 3 pixels for one square. If only 2 pixels per square are used then the light from the square could be split in half with the adjacent white square and the result would be an unresolved gray square. However, all lenses also introduce some loss in contrast which is a function of many things such as lens diffraction, aperture and pixel position. Also in reality the camera's optical axis is not always perfectly orthogonal to the QR code. To account for these extra effects the actual minimum resolution is set to 5 pixels per millimeter.

The 18 million effective pixels on the sensor are split into a 5184 by 3456 grid. The minimum resolution then requires a maximum image size of 1037 by 690 millimeters which constrains the cameras to be no higher than 930 millimeters above the QR codes. In order to make the imaging process more robust, the cameras are mounted 700 millimeters above the ground which corresponds to an image size of 781 by 520 millimeters, and a resolution of approximately 6.5 pixels per millimeter. Mounting the cameras lower than the maximum requirement also always the external lighting to be more concentrated and requires smaller shading if the imaging is done during the day.

Another requirement that is added to make the mapping process more robust is that each QR code must be in a minimum of two images. This helps solve temporary issues

such as bugs flying in front of the camera as well as offers multiple perspectives if the code is accidentally planted at an angle. In order to ensure this the maximum spacing between successive images is given by the equation

$$\begin{aligned} \text{max spacing} &= (\text{image width} - \text{QR side length} - \text{pad})/2 \\ &= (781 - 25 - 75)/2 \\ &= 340 \text{ millimeters} \end{aligned}$$

where the pad is extra spacing to account for variations in camera latency and reduced lighting at the image border.

An additional constraint on the cameras that must be taken into account is the minimum trigger period. Many cameras, including the Canon 7D, are capable of exposing images very rapidly and then buffering them before they are processed. However, the minimum trigger period considered in this section is the minimum amount of time for an image to be exposed, processed and saved without continued buffering, as buffering can only be sustained for so long. This was experimentally determined for the Canon 7D to be 0.7 seconds.

In order to satisfy the maximum image spacing while not exceeding the minimum trigger period, the robot cannot drive faster than 0.5 meters per second. One downside of driving this fast is cameras begin to noticeably shake when the field is not smooth which can lead to occasionally blurry images. Therefore the nominal robot speed is set to 0.4 meters per second. These platform parameters are summarized in table 2.1.

Table 2.1: *Summary of platform parameters.*

Parameter	Value	Units
Camera Height	700	millimeters
Image Size	781 x 520	millimeters
Image Resolution	6.5	pixels / millimeter
Trigger Period	0.7	seconds / image
Robot Speed	0.4	meters / second

2.2.5 On-board Computers

There are a total of four computers used on the Husky.

Main Husky Computer - custom mini-ITX situated inside the main compartment of the robot running Ubuntu 12.04 along with the Robot Operating System (ROS). This computer contains all of the telemetry and guidance logic and is discussed more in section 2.3.1.

Husky Microcontroller - Atmel ARM-based SAM7XC256 enclosed in the back of the robot chassis. This microcontroller receives linear and angular velocity commands from the main computer and reports feedback from the robot's encoders, battery and motors.

Husky Interface Computer - Lenovo T400 laptop that's also running Ubuntu 12.04 along with ROS. This computer sits on top of the Husky and is used to send commands to the robot using a secure shell (SSH)

Data Collection Computer - Lenevo S431 laptop running Windows 7 that also sits on top of the Husky. This computer runs the data collection program responsible for saving data from the cameras and GNSS receiver.

For simplicity the output of the GNSS receiver is split to both the data collection computer as well as the main Husky computer. It would be ideal to run the data collection program on the Husky interface computer to eliminate the need for an extra computer. However, the data collection program requires a Windows based operating system to interact with the Canon cameras, and at the time of this research ROS is not stable on Windows.

2.3 Guidance and Control

The Husky came with basic driving functionality, however this was not sufficient for the mapping process. This section describes the additional functionality developed for the robot that enables it to operate in autonomous or semi-autonomous modes.

2.3.1 Base Functionality

When the Husky was purchased the main computer came installed with the Robot Operating System, which is popular open-source framework that provides many of the same services as traditional operating system such as inter-process communication and hardware-abstraction. One major benefit is ROS allows different functionality to be split up into separate processes, referred to as nodes. This promotes code re-use and prevents one component from crashing the entire system. The nodes that were pre-installed on the Husky are the

Teleop node - receives driving commands from the Logitech controller. The default functionality is when the X button is held down and the left and right analog sticks command linear and angular velocity, respectively.

Husky node - in charge of sending the velocity commands over a serial port to the microcontroller that controls the motors.

IMU node - driver for the UM6 orientation sensor that came installed on the robot. However, since this node is not used since the platform is stable and the multiple GNSS antennas determine yaw.

2.3.2 Cruise Control

When driving through the field it's important to maintain a constant speed to ensure all QR codes and plants are imaged. With the basic teleop functionality this was difficult to achieve while also keeping the robot centered in the middle of the row. To solve this issue the researcher extended the teleop node to include cruise control functionality that is commonly seen in automobiles.

Cruise control can be enabled by either pressing both the Enable 1 and Enable 2 buttons at the same time or by the Preset button. The Preset button defaults to a certain configuration speed, by default 0.4 meters per second. If the Override button is pressed then the linear speed is temporarily calculate from the left analog stick, as is the case in basic driving

Figure 2.7: *Logitech controller for Husky showing cruise control functionality.*

mode. The Resume button returns to the last speed, if there was one, and the up and down arrows on the D-pad vary the commanded speed in increments of 0.05 meters per second.

2.3.3 Automated Control

While this cruise control feature makes it feasible to manually drive the robot through the field, for large experiments this is a tedious task that required ten or more hours of keep the robot centered between the rows.

ROS contains well-developed navigation functionality that allows the robot to convert odometry and sensor data into velocity commands for the robot. This navigation, however, is based around advanced functionality such as cost maps, detailed path planning and map-based localization. All of which are unnecessary for this application. Therefore, the researcher decided to develop a simple guidance solution that is implemented in the following nodes

GPS node - combines position and yaw data from the GPS receiver and publishes this data to the rest of the ROS system.

Waypoint upload node - allows the Husky interface computer to load a set of waypoints

into to robot.

Guidance node - computes robot velocity needed to follow a set of waypoints.

2.4 Data Collection Software

A critical part of the mapping process is being able to accurately associate each image with the position and orientation of the camera at the time the image was taken. This process was accomplished using the Dynamic Sensing Program (DySense), which is an open-source data collection program that provides the means to obtain, organize and geo-locate sensor data. This program was developed by the researcher in order to standardize data collection across various types of platforms and sensors. A screen shot of DySense can be seen in figure 2.8.

Similar to how ROS splits different functionality into processes, DySense can split sensor drivers into processes which allows them to be written in any programming language. The camera sensor driver is written in C# and uses the EOS Digital Software Development Kit (EDSDK) to interact with the camera. This driver allows the images to be downloaded from the camera in real-time and assigns each one a unique file name and UTC time stamp of when the image was exposed.

DySense also uses GPS and IMU drivers to record the platform's position and orientation. This knowledge of the platform state, as well as the camera offsets in the platform frame, allows the positions and orientations of each camera to be calculated at the same time as every image.

2.5 Additional Markers

In addition to the group QR codes there are two other types of markers used in the mapping process, row end markers and plant markers.

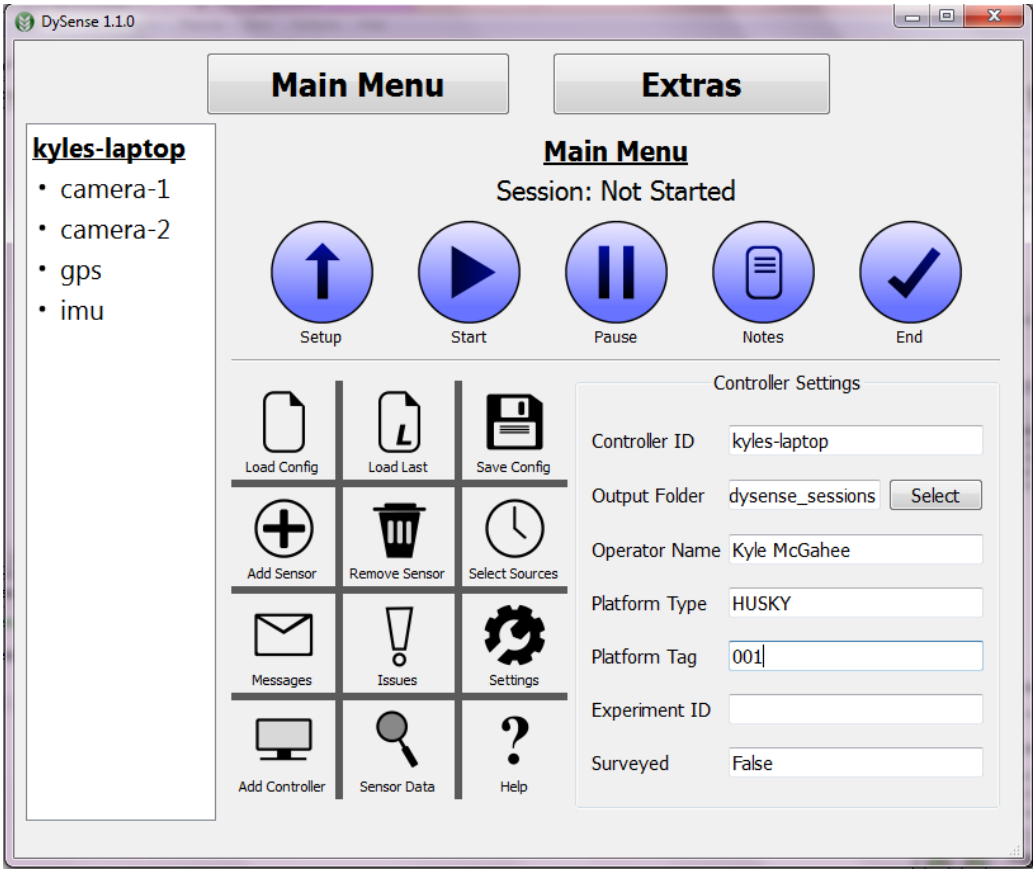

Figure 2.8: *Screenshot of the data collection program.*

2.5.1 Row Markers

Row markers are placed at the beginning and end of each row. Similar to identifying plant groups, these markers are also implemented using QR codes. These row codes store the row number and signify whether the code is the start ("St") or the end ("En") of each row. As discussed in section 2.1, it's critical to know the planting direction of each row because that defines which plants are associated with each group QR code.

2.5.2 Plant Markers

The second type of marker is used to help distinguish regular plants from other plant debris that may be found in the field after tilling ¡TODO¿. This marker is optional, but helps improve the robustness of locating plants. Depending on the size and quality the field these markers can be used on every plant, or for example every 4 plants.

Figure 2.9: *QR codes marking the start and end of row 21.*

The marker used in this experiment is a blue dyed wooden stick approximately 5 inches in length that is placed in the center of each plant. The color blue is selected because it provides the largest difference between other hues likely found in the field, such as yellow/green in plants and red in soil. In addition to marking the plants, this stick also helps prevent the plants from flipping over when exiting the planter.

Another type of plant marker worth mentioning is a colored tag pierced through the top of an un-dyed wooden stick. This type of marker can provide addition information for manual plant inspection and is also much more saturated than the dyed sticks, making it easier to detect. The reason this marker is not used in this experiment is it was not investigated until after the conclusion of this experiment.

Figure 2.10: *Examples of blue stick (left) and colored tag (right)*

Bibliography

Appendix A

Appendix A

Matlab Code

```
% Author: Kyle McGahee
% Description: Show that the two methods for coordinate
    projections are symbollically equivalent.

clear all

%% Method 1 - Euclidean Space

% Pixel coordinates and camera parameters.
syms u v f pw ph u0 v0

% Pixel to sensor frame
sensor = [0 -ph; pw 0]*[u; v] + [ph*v0; pw*u0];

% Rotation matrix
```

```
syms r11 r12 r13 r21 r22 r23 r31 r32 r33
R = [r11 r12 r13;
     r21 r22 r23;
     r31 r32 r33];

% Camera's postition in the world frame.
syms tx ty tz
Tr = [tx; ty; tz];

% Convert to camera frame by appending focal length and then
   rotate to world.
ned = R*[sensor; f];

% Convert this vector to a parameterized position vector.
syms S
pos_vec = S*ned + Tr;

% Solve for S so that the last row is zero (the item is on the
   flat world plane)
S = solve(pos_vec(3) == 0, S);

% Plug back into pos_vec to find actual actual world point.
NED = subs(pos_vec);

% Remove last component because it's zero.
NE_euclidean = NED(1:2);
```

```
%% Method 2 - Projective Space

% Convert from pixel to sensor frame in homogenous coordinates.
% sensor_p == sensor prime == sensor'
sensor_p = [0 -ph ph*v0; pw 0 pw*u0; 0 0 1]*[u; v; 1];

% Convert to camera frame
cam = [1/f 0 0; 0 1/f 0; 0 0 1]*sensor_p;

% 3x4 Transformation matrix
% From world to camera frame.
T = [transpose(R) -Tr];

% Modified transformation assuming D=0 so third column is removed.
Tm = [T(:,1:2) T(:,4)];

% Solve for world point.
world_p = inv(Tm)*cam;

% Convert back to non-homogenous coordinates by dividing by scale
   (S')
NE_projective = world_p(1:2) / world_p(3);

% Third row (R3) is cross product of first two.  Replace
   symbollically
% in 1st method since these terms won't show up in this method.
R3 = cross(R(1,:), R(2,:));
```

```
r31 = R3(1);
r32 = R3(2);
r33 = R3(3);
NE_euclidean = subs(NE_euclidean);

% Display results
%NE_euclidean = simplify(NE_euclidean)
%NE_projective = simplify(NE_projective)

% This will print (1) (true) when the symbollic equations are
    equivalent.
isAlways(NE_euclidean == NE_projective)
```

Matlab Output

```
ans =
    1
    1
```

Appendix B

Title for This Appendix

Enter the content for Appendix B in the appendixB.tex file. If you do not have an Appendix B, see comments in the etdrtemplate.tex file for instructions on how to remove this page.