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

Chapter Title

In this chapter there examples of various features you may want to incorporate into your document. Here's an example of a figure inserted into the text:

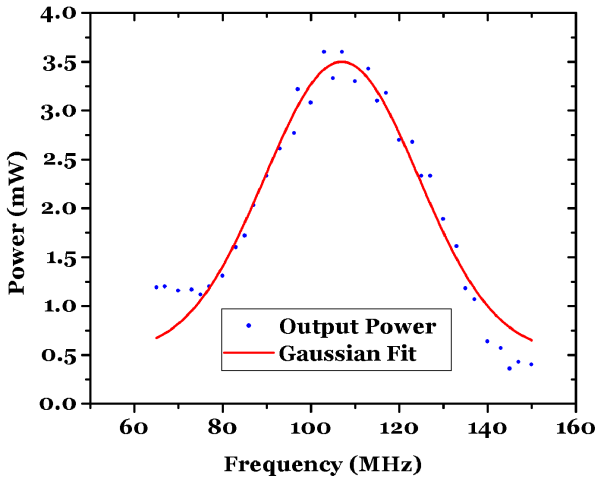

Figure 1.1: *Full caption to appear below the Figure*

See the file chapter1.tex for examples of the commands used to insert a figure or table, add a caption, etc. Here is an example of a table:

Table 1.1: *Caption to appear above the table*

Column 1 Heading	Column 2 Heading	Column 3 Heading
Col 1 Row 1	Col 2 Row 1	Col 3 Row 1
Col 1 Row 2	Col 2 Row 2	Col 3 Row 2
Col 1 Row 3	Col 2 Row 3	Col 3 Row 3

1.1 Making References to Figures or Tables

It is possible to create cross-references and hyperlinks to items or sections within your paper. For example, here is a reference to Fig. 1.1 mentioned at the beginning of this chapter and a reference to the Table 1.1.

1.2 Making a Reference to a Chapter Subsection

In this section, we refer back to text mentioned in Section 1.1 on page 2.

1.3 Making a Citation

Here's an example of a citation to a single work.[1] It's also possible to make multiple citations.[2;3]

This template uses BibTeX to manage and format citations. BibTeX is not the only way to create a bibliography within LaTeX, but it's generally considered to be the best option for long documents like a thesis or dissertation.[4] There are a few more sample citations in this paragraph so you can see examples of how in-text references are made and how the bibliography is formatted.[5] See the file "BibTeX Guide.pdf" for information on how to use BibTeX.

Chapter 2

This is Chapter 2

To refer to Chapter 1, use the slash ref command along with the "makereference" label which was assigned back at the beginning of Chapter 1.

2.1 Page Number References

It is possible to refer to a specific page number, such as page 1. Add a slash label command and a unique name for each page to be referenced later in the text.

2.2 Referring to Sections Within Chapter 1

It is possible to refer to sections within a chapter. Add a slash label command and a unique name with the section number for each section to be referenced later in the text. Her is an example of a figure in section 1.1 and an example of a table in section 1.2. In section 1.3, we looked at examples of bibliographic citations.

Chapter 3

This is Chapter 3

Here are more examples of references to previous sections. In Chapter 1 there were several sections, including section 1.1, section 1.2, and section 1.3.

Likewise, in Chapter 2, there are sections 2.1 and 2.2.

Chapter 4

System Design

TODO

4.1 Plant Identification

A common way to encode information in a machine-readable format is to use a two dimensional barcode. These barcodes

what we're using it for

4.1.1 Code Format

For this project Quick Response (QR) codes were selected as the barcode format. This is a standardized format that is license free and allows for error correction if part of the code is unreadable. What I'm encoding

Figure

4.1.2 Size Constraints

Need to make small... Mention physical size to fit through transplanter. The standard defines 40 different sizes of codes ranging from a 21x21 grid up to a 170x170 grid. The size, or version, is automatically chosen

Various other types of formats, such as ¡¿ or ¡¿, allow information to be encoded in slightly less number of squares, but at the time of the research these formats weren't supported by any of the open-source readers that the researcher investigated.

4.2 Platform Design

The base platform selected is the Husky A2000 mobile robot made by Clearpath Robotics. The Husky is a four wheeled differential drive robot measuring 30 inches long and 25 inches wide. As seen in figure TODO a custom C-channel structure was added onto the top of the robot to enable it to image the field. Attached to the front of the top structure are two Canon 7D digital single-lens reflex (DSLR) cameras. On the back are two antennas ¡TODO what kind¿ which attach to a Trimble BX982 global navigation satellite system (GNSS) receiver which is mounted to the top of the robot.

4.2.1 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 ¡¿ antenna as shown in figure TODO. The robot receiver and base receiver communicate over a ¡¿ 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 ¡TODO¿.

4.2.2 Camera and Lighting

The Canon 7D contains a ¡¿ megapixel sensor and is fitted with a fixed 20mm focal length wide-angle lens. The Canon 7D contains an Advanced Photo System type-C (APS-C) sensor rather than a full frame 35mm sensor, which paired with the 20mm lens gives a horizontal angle of view of ¡¿ degrees and a vertical view of ¡¿ degrees.

Camera placement

Talk about LED lights.

4.2.3 Nominal Speed

Overlap For redundancy and improved accuracy

4.2.4 On-board Computers

There are a total of four computers used on the platform. The first is located within the main compartment of the Husky a custom mini-ITX computer running the Robot Operating System (ROS) which is discussed more in section TODO. Mention wifi/router/laptops.

4.3 Guidance and Control

These packages are found

4.4 Base Functionality

When the Husky was purchased the main computer came installed with the Robot Operating System (ROS), 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 of ROS is it 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 listed below

IMU Node Teleop Node Husky Node

These base packages allowed the robot to be manually driven by the logitech controller shown in figure TODO. The buttons did ¡¿ and the joystick sent ¡¿ commands.

4.5 Cruise Control

However, 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. TODO explain the buttons.

4.6 Automated Control

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

path planning localization motion

ROS contains well-developed navigation functionality that allows the robot to convert odometry and sensor data into velocity commands for the robot. However this navigation 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 implement a simple, custom guidance solution that is implemented in the following nodes

GPS Waypoint Upload Guidance

Required ¡waypoints¿ For simplicity no filtering was used . For faster speeds it may be necessary.

4.7 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 ¡¿.

Similar to how ROS splits up different functionality into processes, DySense can split sensor drivers into processes which allows them to be written in any popular programming language. The camera sensor driver was written in C and used 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, given a unique file name and assigned a time stamp of when the image was exposed.

TODO go over how data is collected

storing height above ground and height above ellipsoid separately.

TODO include figure of DySense

4.8 Additional Markers

In addition to the group QR codes there are two other types of markers used in the mapping process. The first is a row end marker which is also represented as a QR code. These row codes store the row number and signify whether the code is the start or the end of each row. It's important to know the planting direction of each row because that defines which plants are associated with each group QR code.

4.8.1 Row Codes

row code size.

4.8.2 Plant Markers

The second type of marker is used to mark plants to help distinguish them from other plant debris that may be found in the field after TODO. The marker used in this experiment was a

blue dyed wooden stick approximately 5 inches in length that was placed in the center of each plant. The color blue was 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 helped prevent the plants from flipping over when exiting the planter.

TODO include figure of blue stick and row code

Chapter 5

Post-Processing Pipeline

Once the images are collected they must be converted into a field map. This task is accomplished by a set of scripts that are run in a pipeline fashion, where the output of one script is used as the input to the next. This chapter provides a general overview of the pipeline followed by a detailed explanation of each script.

5.1 Pipeline Overview

In this pipeline each script is referred to as a stage, where each stage accomplished one specified task. The main reason the post-processing is split into separate stages is several stages take a significant amount of time to run, so it's beneficial to not re-run the entire pipeline when changes are made. Each stage is summarized in the following list:

Stage 0 Calculate the position and orientation (pose) of each image.

Stage 1 Find and read QR codes in each images.

Stage 2 Create the structure of the field using the QR codes.

Stage 3 Detect leaves and plant markers in each images.

Stage 4 Cluster plant parts from stage 3 into possible plants, and filter out unlikely plants.

Stage 5 Assign individual numbers to plants and save final field map to a comma separated value (CSV) file.

The two stages that take the most time are 1 and 3 as they both deal with opening each image and searching through it. Even though these stages are similar, they are kept separate because having access to the field structure can significantly speed up the clustering step in 4.6

Conceptually the output of each intermediate stage consists of objects which directly relate to the field, for example QR codes, plants or rows. In reality the output is a single file containing the serialized representation of each object. This makes it trivial to pass these objects from one script to another.

The location of the code is listed in Appendix A. TODO include ref.

5.2 Stage 0 - Calculating Camera Pose

5.3 Stage 1 - Extracting QR Codes

The first goal after calculating the position and orientation of each image is to find and read all QR codes in the image set. This process involves four steps which are applied to every image.

The first step is converting the image from the default Blue Green Red (BGR) color space to the Hue Saturation Value (HSV) color space. As can be seen in figure TODO, this is a cylindrical coordinate system which separates image intensity from the color information which makes it more robust to changes in lighting. This color space is also more aligned to how humans think about color TODO insert reference.

TODO insert figure of HSV color space

The second step is to separate the white QR codes from the rest of the image. This is accomplished by applying a range threshold for each of the HSV components as described below

dst(I)=lowerb(I)0src(I)0upperb(I)0lowerb(I)1src(I)1upperb(I)1

This results in a binary image where pixels that are mostly white are 1 (all white) and everything else is 0 (all black).

The third step is finding the set of external contours, or outermost edges, of each object in the binary image. Each set of contours is then assigned a bounding box, which is the smallest rotated rectangle that encompasses all of the object's contours. These bounding box's are filtered to remove one's that are either too small or too large to be QR codes.

The final step is to use these bounding box's to extract parts of the original to run through QR reading program. From the researcher's experience the ZBar open-source program provided the best results.

TODO could talk about adaptive thresholding.

The data returned by the ZBar library is used to determine if the code corresponds to a plant group or to the start/end of a row. If no data, or bad data, is returned by the ZBar library the extracted image is saved for the operator to review after the stage has completed.

TODO

5.4 Stage 2 - Creating Field Structure

The second stage of the pipeline involves assigned row numbers to each QR code and then creating plant groups that can span multiple rows.

As an optional input to this stage, the user can specify a file containing any QR codes from the previous stage that weren't automatically detected.

The first step of this stage is to pair the start and end QR code associated with each row. Since the locations of the codes are known the average row heading can be calculated as shown in figure TODO. This row heading is used to transform the world frame into a what is referred to as the field frame which has the y axis running along the rows. The origin of the field frame is defined such that all the QR codes have a positive position in both the x and y axes. If not specified any future algorithms in the pipeline use these field coordinates rather than world coordinates.

The next step is to assign each group QR code to a specific based only it's location. As some rows can span several hundred meters it's not always possible to assign a QR code to the nearest row defined as a vector between the row start and end codes. Instead a sweeping algorithm is used which is described in the following steps:

The idea is to sweep across the field, from left to right and incrementally add QR codes to each row. Once a code is added to a row it splits that row into smaller segments. Each new code computes the shortest distance to the existing row segments.

The group QR codes are sorted in order of increasing x-field coordinates. For each code find the lateral distance to the nearest row segment, where a segment is defined

It's possible that when the transplanter reaches the end of a row the current plant group isn't finished and continues into the next pass. A pass refers to the transplanter driving once down the field, so a 2-row transplanter would have a pass containing 2 rows. The di

5.5 Stage 3 - Extracting Plant Parts

Similar to the process of extracting QR codes, this stage converts each image to the HSV color space, applies a range threshold, and filters the objects based on size. Since this stage is looking for plant leaves and plant markers it uses different ranges for the HSV components which are shown below.

TODO show range values for filter

Similar to the QR codes these values are set based on experimentation.

Besides the difference in range values this stage also adds an extra step after the threshold and before finding the contours in the image. Then step applies two filters, the first being erosion and the second dilation. The erosion filter removes noise from the image and the then the dilation step connects adjacent contours. These effects can be seen in figure TODO. The reason this step isn't used for the

TODO maybe explain why connecting contours is a good thing.

The reason why this extra step isn't used in stage 1 is that step is already more robust based on the selected HSV range and the square nature of the QR codes doesn't benefit from

connecting contours.

5.6 Stage 4 - Locating Plants

The most challenging part of the pipeline is reliably determining which plant parts found in the previous stage belong to the same plant, and which of those are actual plants that should be mapped. This is challenging because there is often unavoidable plant debris in the field that comes from the tilling right before planting. TODO verify tilling word. Plant markers, such as the blue sticks, help with this issue, but as discussed in TODO ref analysis section the blue sticks could not always be detected. In addition, for large experiments it's not always feasible to have individual markers for every plant.

The task of grouping plant parts into individual plants is done using a hierarchical clustering algorithm. In this application a cluster is represented by the minimum bounding rectangle of one or more plant parts. If two rectangles are clustered together the resulting cluster is represented by the smallest bounding rectangle that fits both the original rectangles. This algorithm combines the nearest two clusters into a single cluster and keeps repeating this process until an end condition is met. The distance between two clusters is defined to be the smallest distance between any of the 4 corners of the rotated rectangles. The end conditions are either (1) there is nothing left to cluster or (2) the nearest cluster is too far apart to be clustered based on a user defined threshold. Also there is a maximum size limit on the clusters which is set to be the maximum expected plant size in the field. Finally any small, unclustered plant parts are removed from the list of possible plants.

After the clustering is completed for an entire plant segment, the list of possible plants is passed to a recursive splitting algorithm to filter out the real plants. The algorithm consists of the following steps

Step 1 Calculate the lateral and projection distance to the segment for each possible plant.

Step 2 Remove any plants that don't fall within the segment based on the projection distance.

Step 3 Find the most likely plant based on various characteristics which is described in more detail below. If there aren't any possible plants then create one in the next location based off the expected transplanter spacing.

Step 4 Repeat the previous step, but starting at the end of the segment and find the next most likely plant by working backwards.

Step 5 Split the original segment into smaller segments using the most likely plants as new points. If the new segments are too short to contain plants then the algorithm is finished, otherwise recursively go back to step 1 for each of the new segments.

TODO include figure of algorithm

In order to determine the most likely plant, each possible plant is assigned a penalty value. This value calculated as

Penalty = (L + P + C) / B

where those variables represent

Lateral Error (L) How far off the plant is from the expected line segment.

Projection Error (P) How far away the plant's projection onto the segment is from where the closest expected plant would be.

Closeness (C) How far away the plant is from the start/end of the segment, with the idea that the lateral and projection errors become less reliable the farther away you are from a known item's location.

Plant-Part Boost (B) Based on what types of plant parts, for example leaves or blue stick parts, are found in the plant.

The idea behind working from the both directions at the same time is the start and end QR codes are known locations in the field, and thus plants near them can be more reliably detected.

5.7 Stage 5 - Saving Field Map

The final stage in the pipeline is generating the final field map. This process first begins by assigning every plant in the field a unique number so it can be easily referenced by another program using the plant's coordinates. This numbering begins with plant 1 at the start of the first row and then follows a serpentine pattern which can be seen in figure TODO. This numbering system is chosen because it represents how researcher's are most likely inspect plants when walking through the field.

Chapter 6

Post-Processing Pipeline

Once the images are collected they must be converted into a field map. This task is accomplished by a set of scripts that are run in a pipeline fashion, where the output of one script is used as the input to the next. This chapter provides a general overview of the pipeline followed by a detailed explanation of each script.

6.1 Pipeline Overview

In this pipeline each script is referred to as a stage, where each stage accomplished one specified task. The main reason the post-processing is split into separate stages is several stages take a significant amount of time to run, so it's beneficial to not re-run the entire pipeline when changes are made. Each stage is summarized in the following list:

Stage 0 Calculate the position and orientation (pose) of each image.

Stage 1 Find and read QR codes in each images.

Stage 2 Create the structure of the field using the QR codes.

Stage 3 Detect leaves and plant markers in each images.

Stage 4 Cluster plant parts from stage 3 into possible plants, and filter out unlikely plants.

Stage 5 Assign individual numbers to plants and save final field map to a comma separated value (CSV) file.

The two stages that take the most time are 1 and 3 as they both deal with opening each image and searching through it. Even though these stages are similar, they are kept separate because having access to the field structure can significantly speed up the clustering step in 4.6

Conceptually the output of each intermediate stage consists of objects which directly relate to the field, for example QR codes, plants or rows. In reality the output is a single file containing the serialized representation of each object. This makes it trivial to pass these objects from one script to another.

The location of the code is listed in Appendix A. TODO include ref.

6.2 Stage 0 - Calculating Camera Pose

6.3 Stage 1 - Extracting QR Codes

The first goal after calculating the position and orientation of each image is to find and read all QR codes in the image set. This process involves four steps which are applied to every image.

The first step is converting the image from the default Blue Green Red (BGR) color space to the Hue Saturation Value (HSV) color space. As can be seen in figure TODO, this is a cylindrical coordinate system which separates image intensity from the color information which makes it more robust to changes in lighting. This color space is also more aligned to how humans think about color TODO insert reference.

TODO insert figure of HSV color space

The second step is to separate the white QR codes from the rest of the image. This is accomplished by applying a range threshold for each of the HSV components as described below

dst(I)=lowerb(I)0src(I)0upperb(I)0lowerb(I)1src(I)1upperb(I)1

This results in a binary image where pixels that are mostly white are 1 (all white) and everything else is 0 (all black).

The third step is finding the set of external contours, or outermost edges, of each object in the binary image. Each set of contours is then assigned a bounding box, which is the smallest rotated rectangle that encompasses all of the object's contours. These bounding box's are filtered to remove one's that are either too small or too large to be QR codes.

The final step is to use these bounding box's to extract parts of the original to run through QR reading program. From the researcher's experience the ZBar open-source program provided the best results.

TODO could talk about adaptive thresholding.

The data returned by the ZBar library is used to determine if the code corresponds to a plant group or to the start/end of a row. If no data, or bad data, is returned by the ZBar library the extracted image is saved for the operator to review after the stage has completed.

TODO

6.4 Stage 2 - Creating Field Structure

The second stage of the pipeline involves assigned row numbers to each QR code and then creating plant groups that can span multiple rows.

As an optional input to this stage, the user can specify a file containing any QR codes from the previous stage that weren't automatically detected.

The first step of this stage is to pair the start and end QR code associated with each row. Since the locations of the codes are known the average row heading can be calculated as shown in figure TODO. This row heading is used to transform the world frame into a what is referred to as the field frame which has the y axis running along the rows. The origin of the field frame is defined such that all the QR codes have a positive position in both the x and y axes. If not specified any future algorithms in the pipeline use these field coordinates rather than world coordinates.

The next step is to assign each group QR code to a specific based only it's location. As some rows can span several hundred meters it's not always possible to assign a QR code to the nearest row defined as a vector between the row start and end codes. Instead a sweeping algorithm is used which is described in the following steps:

The idea is to sweep across the field, from left to right and incrementally add QR codes to each row. Once a code is added to a row it splits that row into smaller segments. Each new code computes the shortest distance to the existing row segments.

The group QR codes are sorted in order of increasing x-field coordinates. For each code find the lateral distance to the nearest row segment, where a segment is defined

It's possible that when the transplanter reaches the end of a row the current plant group isn't finished and continues into the next pass. A pass refers to the transplanter driving once down the field, so a 2-row transplanter would have a pass containing 2 rows. The di

6.5 Stage 3 - Extracting Plant Parts

Similar to the process of extracting QR codes, this stage converts each image to the HSV color space, applies a range threshold, and filters the objects based on size. Since this stage is looking for plant leaves and plant markers it uses different ranges for the HSV components which are shown below.

TODO show range values for filter

Similar to the QR codes these values are set based on experimentation.

Besides the difference in range values this stage also adds an extra step after the threshold and before finding the contours in the image. Then step applies two filters, the first being erosion and the second dilation. The erosion filter removes noise from the image and the then the dilation step connects adjacent contours. These effects can be seen in figure TODO. The reason this step isn't used for the

TODO maybe explain why connecting contours is a good thing.

The reason why this extra step isn't used in stage 1 is that step is already more robust based on the selected HSV range and the square nature of the QR codes doesn't benefit from

connecting contours.

6.6 Stage 4 - Locating Plants

The most challenging part of the pipeline is reliably determining which plant parts found in the previous stage belong to the same plant, and which of those are actual plants that should be mapped. This is challenging because there is often unavoidable plant debris in the field that comes from the tilling right before planting. TODO verify tilling word. Plant markers, such as the blue sticks, help with this issue, but as discussed in TODO ref analysis section the blue sticks could not always be detected. In addition, for large experiments it's not always feasible to have individual markers for every plant.

The task of grouping plant parts into individual plants is done using a hierarchical clustering algorithm. In this application a cluster is represented by the minimum bounding rectangle of one or more plant parts. If two rectangles are clustered together the resulting cluster is represented by the smallest bounding rectangle that fits both the original rectangles. This algorithm combines the nearest two clusters into a single cluster and keeps repeating this process until an end condition is met. The distance between two clusters is defined to be the smallest distance between any of the 4 corners of the rotated rectangles. The end conditions are either (1) there is nothing left to cluster or (2) the nearest cluster is too far apart to be clustered based on a user defined threshold. Also there is a maximum size limit on the clusters which is set to be the maximum expected plant size in the field. Finally any small, unclustered plant parts are removed from the list of possible plants.

After the clustering is completed for an entire plant segment, the list of possible plants is passed to a recursive splitting algorithm to filter out the real plants. The algorithm consists of the following steps

Step 1 Calculate the lateral and projection distance to the segment for each possible plant.

Step 2 Remove any plants that don't fall within the segment based on the projection distance.

Step 3 Find the most likely plant based on various characteristics which is described in more detail below. If there aren't any possible plants then create one in the next location based off the expected transplanter spacing.

Step 4 Repeat the previous step, but starting at the end of the segment and find the next most likely plant by working backwards.

Step 5 Split the original segment into smaller segments using the most likely plants as new points. If the new segments are too short to contain plants then the algorithm is finished, otherwise recursively go back to step 1 for each of the new segments.

TODO include figure of algorithm

In order to determine the most likely plant, each possible plant is assigned a penalty value. This value calculated as

Penalty = (L + P + C) / B

where those variables represent

Lateral Error (L) How far off the plant is from the expected line segment.

Projection Error (P) How far away the plant's projection onto the segment is from where the closest expected plant would be.

Closeness (C) How far away the plant is from the start/end of the segment, with the idea that the lateral and projection errors become less reliable the farther away you are from a known item's location.

Plant-Part Boost (B) Based on what types of plant parts, for example leaves or blue stick parts, are found in the plant.

The idea behind working from the both directions at the same time is the start and end QR codes are known locations in the field, and thus plants near them can be more reliably detected.

6.7 Stage 5 - Saving Field Map

The final stage in the pipeline is generating the final field map. This process first begins by assigning every plant in the field a unique number so it can be easily referenced by another program using the plant's coordinates. This numbering begins with plant 1 at the start of the first row and then follows a serpentine pattern which can be seen in figure TODO. This numbering system is chosen because it represents how researcher's are most likely inspect plants when walking through the field.

Chapter 7

Post-Processing Pipeline

Once the images are collected they must be converted into a field map. This task is accomplished by a set of scripts that are run in a pipeline fashion, where the output of one script is used as the input to the next. This chapter provides a general overview of the pipeline followed by a detailed explanation of each script.

7.1 Pipeline Overview

In this pipeline each script is referred to as a stage, where each stage accomplished one specified task. The main reason the post-processing is split into separate stages is several stages take a significant amount of time to run, so it's beneficial to not re-run the entire pipeline when changes are made. Each stage is summarized in the following list:

Stage 0 Calculate the position and orientation (pose) of each image.

Stage 1 Find and read QR codes in each images.

Stage 2 Create the structure of the field using the QR codes.

Stage 3 Detect leaves and plant markers in each images.

Stage 4 Cluster plant parts from stage 3 into possible plants, and filter out unlikely plants.

Stage 5 Assign individual numbers to plants and save final field map to a comma separated value (CSV) file.

The two stages that take the most time are 1 and 3 as they both deal with opening each image and searching through it. Even though these stages are similar, they are kept separate because having access to the field structure can significantly speed up the clustering step in 4.6

Conceptually the output of each intermediate stage consists of objects which directly relate to the field, for example QR codes, plants or rows. In reality the output is a single file containing the serialized representation of each object. This makes it trivial to pass these objects from one script to another.

The location of the code is listed in Appendix A. TODO include ref.

7.2 Stage 0 - Calculating Camera Pose

7.3 Stage 1 - Extracting QR Codes

The first goal after calculating the position and orientation of each image is to find and read all QR codes in the image set. This process involves four steps which are applied to every image.

The first step is converting the image from the default Blue Green Red (BGR) color space to the Hue Saturation Value (HSV) color space. As can be seen in figure TODO, this is a cylindrical coordinate system which separates image intensity from the color information which makes it more robust to changes in lighting. This color space is also more aligned to how humans think about color TODO insert reference.

TODO insert figure of HSV color space

The second step is to separate the white QR codes from the rest of the image. This is accomplished by applying a range threshold for each of the HSV components as described below

dst(I)=lowerb(I)0src(I)0upperb(I)0lowerb(I)1src(I)1upperb(I)1

This results in a binary image where pixels that are mostly white are 1 (all white) and everything else is 0 (all black).

The third step is finding the set of external contours, or outermost edges, of each object in the binary image. Each set of contours is then assigned a bounding box, which is the smallest rotated rectangle that encompasses all of the object's contours. These bounding box's are filtered to remove one's that are either too small or too large to be QR codes.

The final step is to use these bounding box's to extract parts of the original to run through QR reading program. From the researcher's experience the ZBar open-source program provided the best results.

TODO could talk about adaptive thresholding.

The data returned by the ZBar library is used to determine if the code corresponds to a plant group or to the start/end of a row. If no data, or bad data, is returned by the ZBar library the extracted image is saved for the operator to review after the stage has completed.

TODO

7.4 Stage 2 - Creating Field Structure

The second stage of the pipeline involves assigned row numbers to each QR code and then creating plant groups that can span multiple rows.

As an optional input to this stage, the user can specify a file containing any QR codes from the previous stage that weren't automatically detected.

The first step of this stage is to pair the start and end QR code associated with each row. Since the locations of the codes are known the average row heading can be calculated as shown in figure TODO. This row heading is used to transform the world frame into a what is referred to as the field frame which has the y axis running along the rows. The origin of the field frame is defined such that all the QR codes have a positive position in both the x and y axes. If not specified any future algorithms in the pipeline use these field coordinates rather than world coordinates.

The next step is to assign each group QR code to a specific based only it's location. As some rows can span several hundred meters it's not always possible to assign a QR code to the nearest row defined as a vector between the row start and end codes. Instead a sweeping algorithm is used which is described in the following steps:

The idea is to sweep across the field, from left to right and incrementally add QR codes to each row. Once a code is added to a row it splits that row into smaller segments. Each new code computes the shortest distance to the existing row segments.

The group QR codes are sorted in order of increasing x-field coordinates. For each code find the lateral distance to the nearest row segment, where a segment is defined

It's possible that when the transplanter reaches the end of a row the current plant group isn't finished and continues into the next pass. A pass refers to the transplanter driving once down the field, so a 2-row transplanter would have a pass containing 2 rows. The di

7.5 Stage 3 - Extracting Plant Parts

Similar to the process of extracting QR codes, this stage converts each image to the HSV color space, applies a range threshold, and filters the objects based on size. Since this stage is looking for plant leaves and plant markers it uses different ranges for the HSV components which are shown below.

TODO show range values for filter

Similar to the QR codes these values are set based on experimentation.

Besides the difference in range values this stage also adds an extra step after the threshold and before finding the contours in the image. Then step applies two filters, the first being erosion and the second dilation. The erosion filter removes noise from the image and the then the dilation step connects adjacent contours. These effects can be seen in figure TODO. The reason this step isn't used for the

TODO maybe explain why connecting contours is a good thing.

The reason why this extra step isn't used in stage 1 is that step is already more robust based on the selected HSV range and the square nature of the QR codes doesn't benefit from

connecting contours.

7.6 Stage 4 - Locating Plants

The most challenging part of the pipeline is reliably determining which plant parts found in the previous stage belong to the same plant, and which of those are actual plants that should be mapped. This is challenging because there is often unavoidable plant debris in the field that comes from the tilling right before planting. TODO verify tilling word. Plant markers, such as the blue sticks, help with this issue, but as discussed in TODO ref analysis section the blue sticks could not always be detected. In addition, for large experiments it's not always feasible to have individual markers for every plant.

The task of grouping plant parts into individual plants is done using a hierarchical clustering algorithm. In this application a cluster is represented by the minimum bounding rectangle of one or more plant parts. If two rectangles are clustered together the resulting cluster is represented by the smallest bounding rectangle that fits both the original rectangles. This algorithm combines the nearest two clusters into a single cluster and keeps repeating this process until an end condition is met. The distance between two clusters is defined to be the smallest distance between any of the 4 corners of the rotated rectangles. The end conditions are either (1) there is nothing left to cluster or (2) the nearest cluster is too far apart to be clustered based on a user defined threshold. Also there is a maximum size limit on the clusters which is set to be the maximum expected plant size in the field. Finally any small, unclustered plant parts are removed from the list of possible plants.

After the clustering is completed for an entire plant segment, the list of possible plants is passed to a recursive splitting algorithm to filter out the real plants. The algorithm consists of the following steps

Step 1 Calculate the lateral and projection distance to the segment for each possible plant.

Step 2 Remove any plants that don't fall within the segment based on the projection distance.

Step 3 Find the most likely plant based on various characteristics which is described in more detail below. If there aren't any possible plants then create one in the next location based off the expected transplanter spacing.

Step 4 Repeat the previous step, but starting at the end of the segment and find the next most likely plant by working backwards.

Step 5 Split the original segment into smaller segments using the most likely plants as new points. If the new segments are too short to contain plants then the algorithm is finished, otherwise recursively go back to step 1 for each of the new segments.

TODO include figure of algorithm

In order to determine the most likely plant, each possible plant is assigned a penalty value. This value calculated as

Penalty = (L + P + C) / B

where those variables represent

Lateral Error (L) How far off the plant is from the expected line segment.

Projection Error (P) How far away the plant's projection onto the segment is from where the closest expected plant would be.

Closeness (C) How far away the plant is from the start/end of the segment, with the idea that the lateral and projection errors become less reliable the farther away you are from a known item's location.

Plant-Part Boost (B) Based on what types of plant parts, for example leaves or blue stick parts, are found in the plant.

The idea behind working from the both directions at the same time is the start and end QR codes are known locations in the field, and thus plants near them can be more reliably detected.

7.7 Stage 5 - Saving Field Map

The final stage in the pipeline is generating the final field map. This process first begins by assigning every plant in the field a unique number so it can be easily referenced by another program using the plant's coordinates. This numbering begins with plant 1 at the start of the first row and then follows a serpentine pattern which can be seen in figure TODO. This numbering system is chosen because it represents how researcher's are most likely inspect plants when walking through the field.

Bibliography

[1] J. Weiner, V. S. Bagnato, S. Zilio, and P. S. Julienne. Experiments and theory in cold and ultracold collisions. *Rev. Mod. Phys.*, 71:1–85, 1999. http://test4.com.

[2] W. D. Phillips, J. V. Prodan, and H. J. Metcalf. Laser cooling and electromagnetic trapping of neutral atoms. *J. Opt. Soc. Am. B*, 2:1751–1767, 1985. URL http://test3.com.

[3] M. M. T. Loy. Observation of population inversion by optical adiabatic rapid passage. *Phys. Rev. Lett.*, 32:814–817, 1974. URL http://test.com.

[4] P. L. Gould, P. D. Lett, P. S. Julienne, and W. D. Phillips. Observation of associative ionization of ultracold laser-trapped sodium atoms. *Phys. Rev. Lett.*, 60:788–791, 1988. URL http://test5.com.

[5] J. S. Melinger, A. Hariharan, S. R. Gandhi, and W. S. Warren. Adiabatic population inversion in i_2 vapor with picosecond laser pulses. *J. Chem. Phys.*, 95:2210–2213, 1991. URL http://test2.com.

Appendix A

Title for This Appendix

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

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.