Utilizing ArUco markers to define implement boundaries

by

Riley John Sleichter

B.S., Kansas State University, 2021

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Biological and Agricultural Engineering
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2024

Approved by:

Major Professor
Dr. Ajay Sharda

Copyright

Abstract

John Deere and Blue River Technology's autonomous tillage system combines multidisciplinary efforts and cutting-edge technology to achieve Level 5—Unsupervised Autonomy. To create this engineering marvel, countless parameters need defined to ensure safe operation of the system; some of these parameters are static, while other of these parameters are dynamic. One particular set of parameters define the tillage implement's boundaries for the software stack to utilize, and today that parameter is static. However, to meet Deere and Company's public, lofty ambitions to scale autonomous tillage to farms around the world, the parameters that define the implement boundaries need to become dynamic. This research project provides a potential solution to defining these implement boundary parameters in a dynamic fashion.

By utilizing Augmented Reality University of Cordoba (ArUco) fiducial markers in unison with the existing Blue River Technology software and hardware stack, a set of dynamic implement boundary parameters is defined. ArUco markers are strategically located around the implement for a moment in time, where in a static fashion, the existing camera hardware takes a snapshot of the implement and markers, then utilizes that information to create and store the newly defined implement parameters. After this brief moment in time, the ArUco markers may be removed from the implement, and the system may proceed with the newly defined and stored implement boundary parameters.

This paper provides a background on fiducial markers, describes the process of locating the ArUco markers, and steps through the software developed to implement this boundary parameter definition.

Table of Contents

List of Figures

List of Tables

Acknowledgements

The information described below has been filed for provisional patent with the United States as U.S. Application No. 63/548,777 entitled Utilizing Fiducial Markers to identify Implement Pixels within an Image, filed on February 1, 2024. The provisional patent was filed on behalf of Blue River Technology Inc.

The writer greatly appreciates the continued support of the John Deere and Blue River Technology teams to enable this research project to take place.

Chapter 1 - Introduction

Introduction

Two major agricultural and socioeconomic challenges must be conquered by the time the world population plateaus at 10 billion people near 2050 (United Nations, 2017): feeding the world and meeting consumer demand. As the population grows, one would expect the demand for food to follow linearly to the growth in population, however, with the global middle-class continuing to expand, the growth in demand for food forgoes this hypothesis and is increasing at an increasing rate. As the world becomes wealthier, the demand for higher quality food is added to the stress of producing a higher quantity of food.

This challenge is multiplied as the traditional production agriculture theory of: "use more to produce more" is now irrelevant as agricultural land is depleting at an alarming rate due to urban sprawl (Robinson, 2019), significant water reserves and aquifers have been abused to the point of near extinction (Vaughn-Uding, 2021), modern agricultural monocropping practices are proving to disrupt the natural ecosystem, and the 'brain drain' of rural America continues to empty our once flourishing rural communities. Traditional linear input-output plots no longer suffice United States production agriculture in 2023, and to begin combatting this issue, agriculture must focus on producing more with less.

Blue River Technology and John Deere have committed to producing more with less. To augment the declining labor supply, autonomous vehicles have become a focus, and the organizations created a perception system to begin the journey to autonomy. However, with the wide array of shapes, sizes, and configurations of implements to be pushed in front or pulled behind a tractor, a new set of challenges is coming forth. This perception system must be pliable and portable to easily match various customer configurations and desires of the future farm.

Problem Statement

Today, the perception system from Blue River Technology and John Deere is specifically catered to each configuration of tractor and implement. This rigid system is proving to be challenging to manage with research machines scattered about the United States, and the challenge will be amplified with the shift to mass production. Currently, the perception system supports one tractor model and select implements.

Tractor:

- 2021- 8R 410

Implements:

- 2660 VT
- 2430 CP

However, with the ambitious and public goal of John Deere to create a fully autonomous corn and soybean production system by 2030 (Lambert, 2023), including tillage, seeding, spraying, and harvesting, this rather rigid perception system may not be sustainable for the future goals of the business.

One of the main challenges is catering the Computer Vision Machine Learning (CVML) and robotics software to the implement connected to the machine. Currently, changing between implement configurations is not easily supported, and to be successful in scaling both type and quantity of configurations, this challenge must be solved, and this project is targeted at resolving this issue.

This research project's focus is taking a new approach at the methodology of accounting for the implement behind the tractor by utilizing classical computer vision image processing and five transferable, Aruco Module (ArUco) markers strategically located on the implement to

successfully identify the location of the implement to communicate with the rest of the perception system. The ArUco markers will be intentionally located on the corners of the implement to signify the edges of the implement for the algorithm, as shown in figure 1.1.

Figure 1.1 Example of ArUco markers on a 2430 CP implement

The ArUco markers are correlated together in an image, and the script will identify the markers as the ‘corners’ of the implement. Then, the corners will utilize the stereodepth metadata associated with the pixels to identify the point locations in 3-dimensions. Finally, these points will be utilized in the robotics pipeline to identify the position of the implement in 3-dimensions. The remainder of the robotics stack will utilize the newly identified implement position for downstream tasks.

This project would allow the ability for customers to easily change their autonomous tractor to different implements by only changing the markers on the implement at the time of setup. Additionally, allowing the ability for this program to begin scaling easily to different implements and configurations.

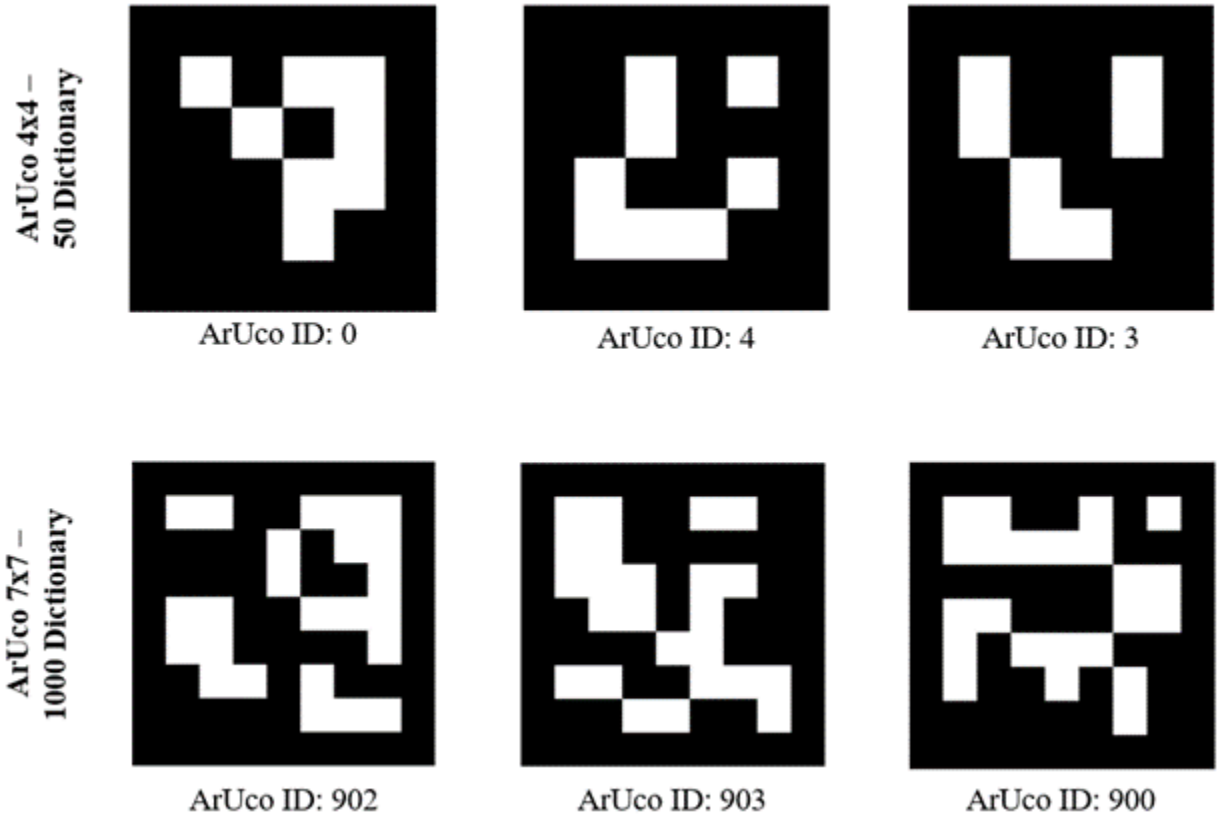
Introduction to Fiducial Markers

Fiducial markers are markers of known pattern and size placed in the physical world to augment robotics systems through computer vision. The markers were originally designed for augmented reality applications and are widely used across the robotics discipline (M. Kalaitzakis, 2020). Most everyday consumers will recognize a fiducial maker as a Quick Response (QR) code, which have become commonplace in modern society for quickly scanning and accessing information.

A QR code works by arranging a series of black and white pixels into a unique pattern that encodes a string of data (Freda, 2022). This QR code is scanned by a QR code scanner, which utilizes existing algorithms to decipher the encoded information, then take the user to the encoded Uniform Resource Locator (URL) link. Fiducial markers work in a similar manner, however, do not have the ability to encode as much information as a QR code.

While a fiducial marker works similarly to a QR code in the way information is encoded, one difference is that fiducial markers belong to a predefined dictionary of unique identities, where each arrangement of encoded black and white pixels correlates with a specific marker identity. Within these dictionaries of fiducial markers, there is a predefined number and size of distinct fiducial marker encodings that correspond to distinct identities within that dictionary. For example, within the ArUco 4x4_50 dictionary, there are 50 distinct fiducial marker encodings that are predefined, and the marker with the distinct identify of 0 always shows up as the encoding as shown in the figure below. Two examples of ArUco marker dictionaries are displayed in figure 1.2 below: ArUco 4x4_50 dictionary and the ArUco 7x7_1000 dictionary:

Figure 1.2 Example of ArUco marker dictionaries and IDs



ArUco Marker Detection

ArUco markers are common in augmented reality and robotics applications because they are easy to implement through the OpenCV library and are reliable for identifying location and object pose. OpenCV contains great documentation to describe the marker detection process, pose estimation, and functions within the ArUco marker contribution library. It is highly suggested to read the documentation provided by OpenCV for a more in depth understanding of ArUco markers.

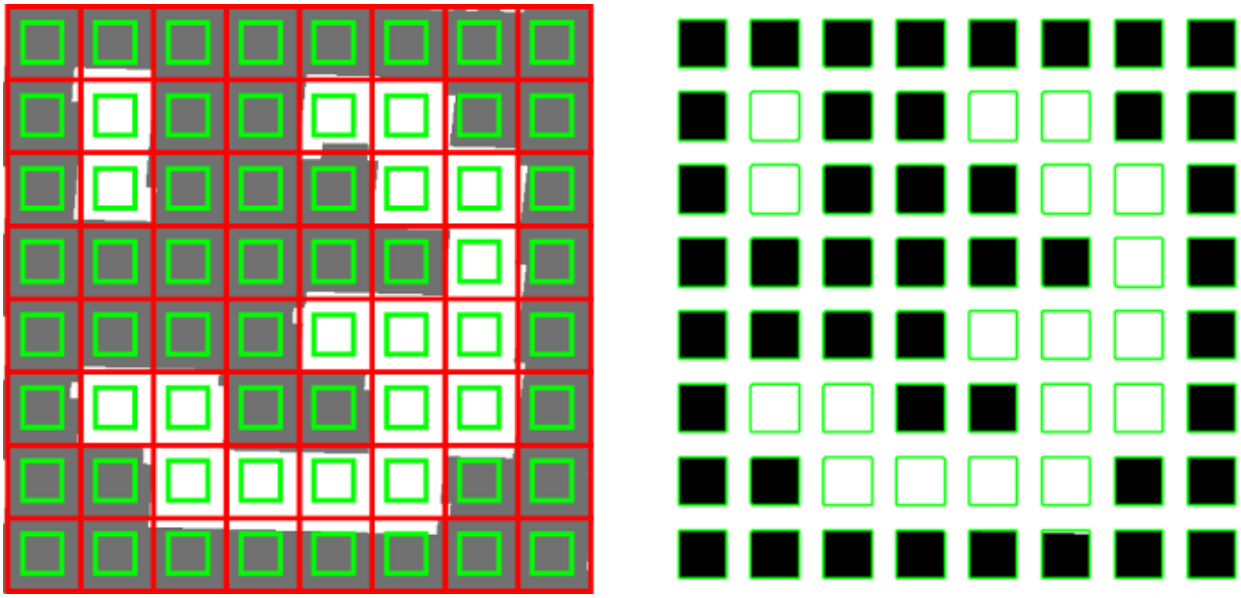
As a brief overview ArUco marker detection can be broken into two steps (OpenCV):

1) Detection of marker candidates

2) Evaluation of candidates by analyzing their encoding

The provided OpenCV detection algorithm executes these two steps: reads the encoded information, then searches the predefined dictionary for encoding matches. If a match is found, the algorithm returns the pixel coordinates defining the four corners of the marker and the identify of the marker. Figure 1.3 visualizes how the algorithm extracts bits of encoded information from the ArUco marker (OpenCV).

Figure 1.3 Visualizing ArUco marker candidate analysis (from OpenCV)



A significant amount of parameters can be finetuned by the developer to impact the sensitivity and effectiveness of the detection algorithm for the user's use case. All of these parameters are well documented within OpenCV.

Fiducial Marker 'Families'

There are multiple "families" of markers within the world of fiducial markers and the two prevalent families are AprilTags and ArUco markers. The tag families operate similarly at a high level, but vary in their detection algorithms, which result in varying detection latency and

accuracy between the families (Ceaser, 2015). For this project, ArUco markers were selected due to their ease of integration within the OpenCV library. Additionally, the OpenCV library imports multiple dictionaries of varying size for ArUco markers, enabling a wide selection of marker dictionaries.

Within the ArUco marker tag family, multiple dictionaries of predefined markers are defined. There are two variables to define within the marker dictionary: the marker bit dimension and the dictionary size. The available ArUco marker dimensions within the OpenCV imported library are: 4x4, 5x5, 6x6, and 7x7. The marker dimension defines the number of bits encoded in each of the two dimensions of the encoding, therefore, a 4x4 marker has 4 black and white encodings in the first dimension and 4 black and white encodings in the second dimension. The larger the encoding size, the more potential unique marker identities are available. Within each of these marker dimensions, different dictionary sizes are available to query. Each marker dimension is available in a dictionary size of 50, 100, 250, or 1000 distinct identities. The predefined marker dimensions and dictionaries imported with OpenCV are seen in table 1.1:

Table 1.1 Predefined dictionaries within OpenCV (from OpenCV)

Marker Encoding Size	Dictionary Sizes
4x4	50, 100, 250, 1000
5x5	50, 100, 250, 1000
6x6	50, 100, 250, 1000
7x7	50, 100, 250, 1000

Marker Dimensions and Dictionary Size

Often in fiducial marker applications, less information needs encoded than in the use case of a QR code. QR codes often encode an entire URL, whereas fiducial markers are only encoding their unique identity. Fiducial marker detection depends on the camera's ability to

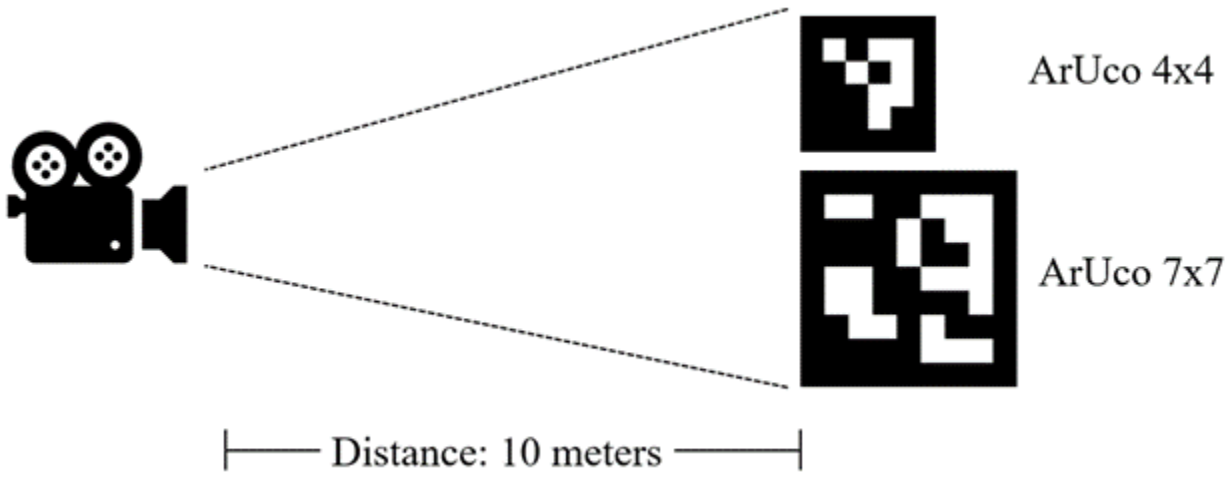
resolve each of the black and white ‘bits’ of information into individual pixels for information extraction. Therefore, to successfully identify the marker a smaller marker encoding dimension allows the marker to be more easily sensed at greater distances.

For marker detection the camera must resolve each ‘bit’ of encoded information into a minimum number of black or white pixels for the OpenCV detection algorithm to extract the encoded piece of information. The camera’s ability to resolve its field of view into pixels at distance directly impacts the performance of the detection algorithm. For example: a camera may be able to resolve a 1-meter x 1-meter square ArUco marker, 10 meters away from the camera into 400 pixels, which results in a successful detection! However, when the same 1-meter x 1-meter marker is moved to 20 meters away, now the same camera can only resolve that square into 100 pixels. There is now not enough pixels for the detection algorithm to extract the encoded information from the marker, thus the detection fails.

The above example is relevant, as the more bits of information encoded into the ArUco marker, the more pixels need successfully resolved by the camera to effectively extract the encoded information. For a small dimension ArUco marker (ie: 4x4) less encoded information needs resolved by the camera for successful detection. For a large dimension marker (ie: 7x7) more encoded information needs resolved by the camera for successful detection.
Given a fixed distance between the ArUco marker and the camera, the physical size of the marker is determined by the camera’s ability to resolve each bit of encoded information in the marker into pixels. Therefore, the more encoded information present in the marker, the physically larger the marker must be for the camera to effectively resolve each bit into pixels. For example: at a fixed distance of 10 meters between the camera and the ArUco marker, a 4x4 marker can be physically smaller than a 7x7 marker, as the 7x7 marker needs the physical size of

each bit of information to be large enough for the camera to resolve accurately into pixels. An illustration of this can be seen in the figure below:

Figure 1.4 Illustration visualizing marker dimension and required physical marker size



Dictionary size (50, 100, 250, or 1000) impacts the likelihood of false positive marker detections. For example, if the algorithm is parsing a dictionary of size 50 compared to 1000, the algorithm is more likely to return a false positive or incorrect marker identity from the dictionary of size 1000 because there are more possible markers to chose from. Use the smallest dictionary possible for your application to minimize the potential for false positive detection.

For selecting the fiducial marker dictionary, the use case of the markers needs to be considered. Markers with a smaller encoding size can be physically smaller and still detected at greater distances, as the detection algorithm requires a minimum number of pixels for each bit of information to be resolved into for detection. At a far distance, a marker with an encoding size of 4x4 can physically be smaller and still meet the resolution minimum for the detection algorithm to detect than a 6x6 encoding. With a smaller dictionary, a lower likelihood of false positive

detections is likely, as there are fewer unique identities for the algorithm to falsely detect (Ceaser, 2015).

Chapter 2 - Utilizing ArUco Markers to Define Implement Boundaries

Why Fiducial Markers for this Experiment?

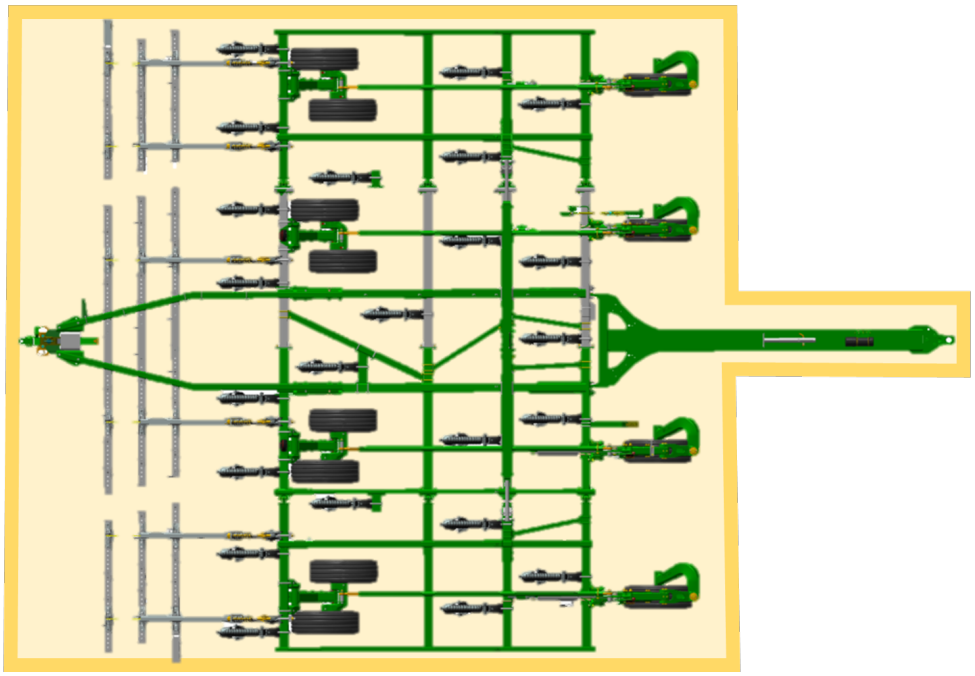
There are many potential solutions utilizing deep learning, computer vision, and fiducial markers to resolve the robotic dependencies. However, the object of this project was to utilize traditional computer vision instead of deep learning to minimize computational requirements for deployment.

Furthermore, fiducial markers were utilized for this experiment due to the infrastructure already being available for detecting these in a computer vision task, thus allowing for quick prototyping in this computer vision and robotics pipeline.

Physical Sizing and Placement of the Fiducial Markers

The objective of this project is to define the boundary of an implement utilizing ArUco markers, as the robotics stack will utilize the defined implement boundary. To better understand the objective at hand, the illustration has been created below in figure 2.1 visualizing the top-down view of an implement. The light yellow polygon in the background of the implement represents the ground footprint of the implement, and the dark yellow border defines an approximate boundary for the implement.

Figure 2.1 Visualizing the implement boundary in dark yellow and implement footprint in light yellow



The first step in defining the implement boundary with ArUco markers was to adequately size and attach the markers to the implement. Based on the geometry of the implement and field of view of the camera being utilized, it was determined that five ArUco markers placed in the locations illustrated below would generate an implement boundary most resembling the ground footprint of the implement, while still allowing the markers to be mounted to the implement frame.

Figure 2.2 Placement of the ArUco markers on the physical implement

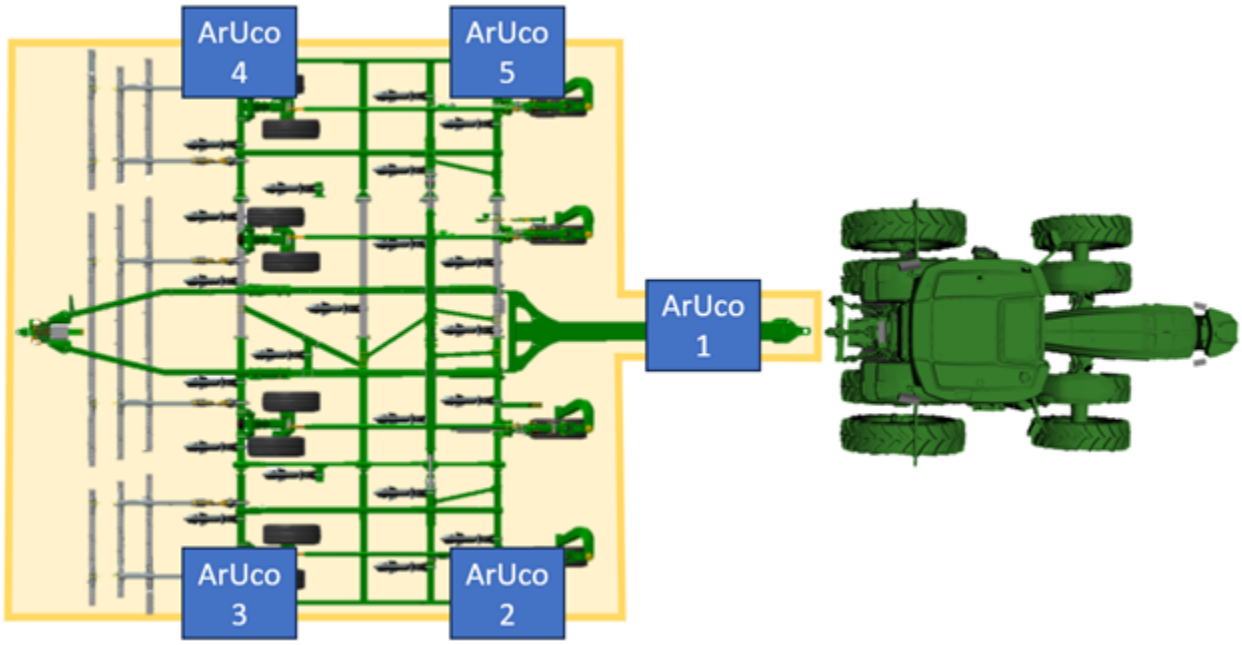
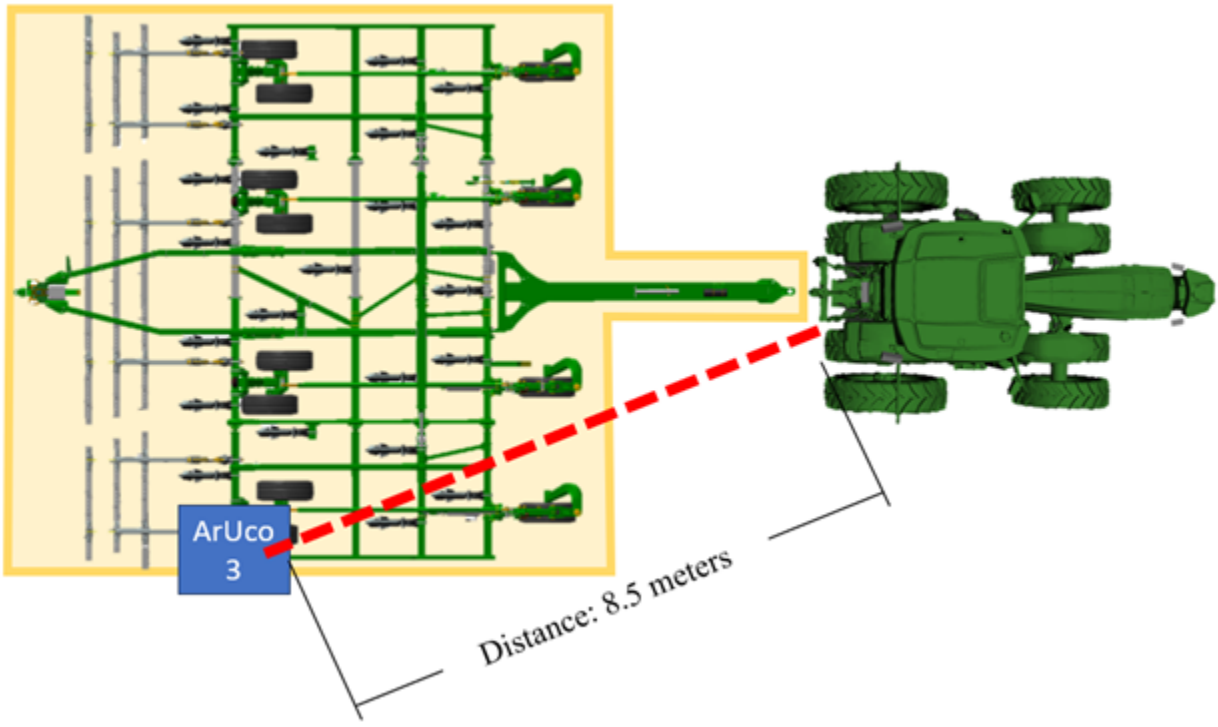


Figure 2.3 Physical distance between the camera utilized and the rear marker



The camera utilized in the project is physically mounted on the rear of the tractor cab as seen in figure 2.3.

As seen in the picture above, the rear two markers are physically a long distance from the camera—mounted on the rear of the tractor cab, making it challenging for the detection algorithm identify the fiducial markers in the image due to the few number of pixels the marker is resolved to in the digital world. The rear two marker's distance (ArUco 3 and 4 in the illustration) from the camera is the limiting factor to the physical size needed for the markers to be detected by the detection algorithm. Therefore, the physical size of the markers is calculated on the distance for the rear two markers.

To calculate the size of the markers many parameters are needed, with the main two being: camera specifications and distance between the marker and the camera.

The camera utilized in this project is a MoTec 30cm stereo camera, which utilizes two individual camera lens and imager pairs spaced 30cm apart to generate the image and calculate disparity for estimating depth of objects. The relevant camera specifications for this project are the number of pixels an object will resolve into at a given distance. For information security, those camera specifications are withheld from this paper, however, the results of the study are described below.

The maximum distance from the camera imager to the ArUco marker is 8.5 meters in the tractor and implement combination presented. Therefore, the number of pixels the ArUco marker is resolved to at various sizes is seen in the table below:

Table 2.1 Number of pixels camera resolves ArUco marker into at 8.5 meter distance

ArUco Marker Size	Resolved Pixels
12" x 12"	233.7841
15" x 15"	375.1969
18" x 18"	526.2436
21" x 21"	730.0804
24" x 24"	964.1025

According to the GeoForAll Lab, the minimum number of pixels needed for the algorithm to detect an ArUco marker is 20 pixels x 20 pixels—400 pixels total. Therefore, according to the calculated pixel resolutions above, the minimum size needed is 18” x 18” for the ArUco marker, which will resolve into 526 pixels.

However, it was determined that the focus of this experiment is not to determine how small the marker can be, but rather utilizing these ArUco markers to generate the boundary of the implement. Therefore, it was decided to oversize the ArUco markers to 24” x 24”, doubling the minimal number of pixels needed for the algorithm to detect the markers. This ensures that the detection algorithm can detect the markers.

Fiducial Marker Dictionary Decision

With the physical size of the marker defined, an ArUco marker dictionary needed to be defined to create these markers. For reasons stated in the ‘Marker Dimensions and Dictionary Size’ section, the ArUco dictionary of markers sized 4x4 and dictionary size of 50.

By selecting a dictionary that is only encoded with 4 bits by 4 bits, there is less information needing to be resolved, resulting in the camera detection algorithm to resolve a physically smaller marker with fewer pixels. Additionally, selecting a dictionary size of 50 allows the detection algorithm fewer opportunities for false positive detections, as there are fewer markers for the dictionary to detect.

With the fiducial markers sized and dictionary established, it was time to design a bracket to mount these markers onto the implement with.

Marker Bracket Design

Utilizing Creo and Deere implement models, an adjustable and universal bracket was designed to easily mount onto the frame of the John Deere 2430 Chisel Plow. The design of this bracket allowed for maximum adjustability through set screws and pins to allow the ArUco markers to be adjusted on the implement to optimize the ArUco marker placement from the mounted camera's field of view. The brackets were manufactured by the writer at Kansas State University within the Agricultural Engineering shop. A rendering of the bracket and the bracket mounted on the implement can be seen in the figures below.

Figure 2.4 Creo rendering of the mounting bracket

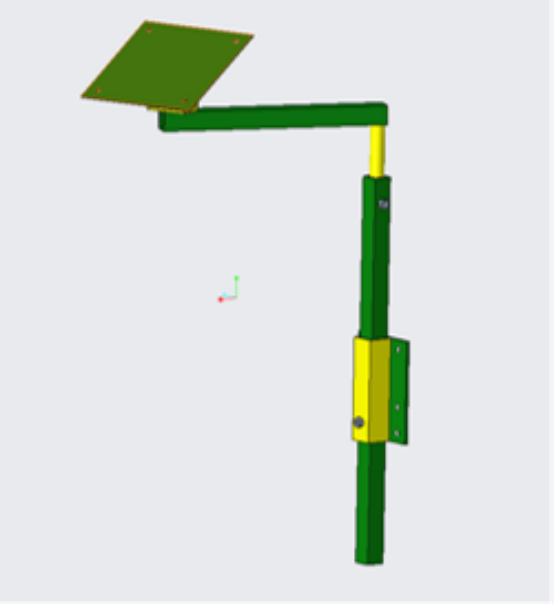


Figure 2.5 The bracket mounted on the implement

Data Collection and Detection Algorithm

With the fiducial markers physically in place on the implement, it is time to utilize the camera mounted on the tractor to detect the location of the markers on the implement. The detection algorithm and data processing are completed offline and not in real time.

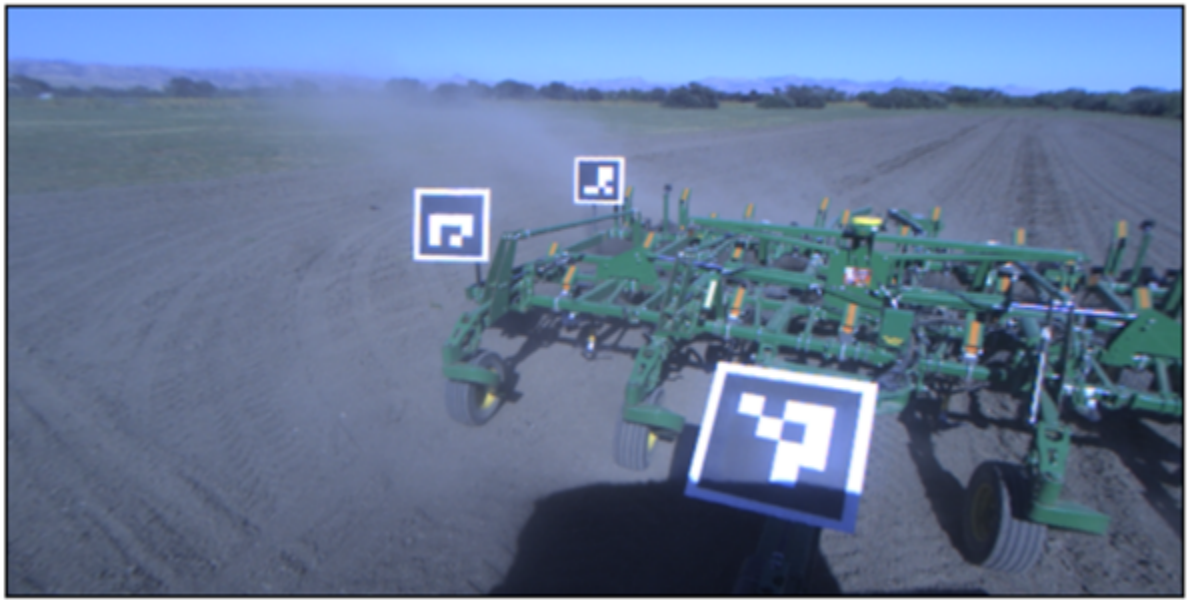
The question then comes to mind: Do the markers stay attached to the implement all the time or are they removed? Both scenarios are tested in this research project.

When the markers were attached to the implement continuously during operation, two notable issues arose:

1) First, while the machine is turning, the field of view from the camera changes due to the pivoting action that occurs during a turn at the tractor's hitch pin. During the turning motion, the field of view of the implement changes from the mounted camera, and the camera loses sight of all five ArUco markers. Thus, leaving this strategy to be susceptible to turning and wider implements—as wider implements would span past the cameras

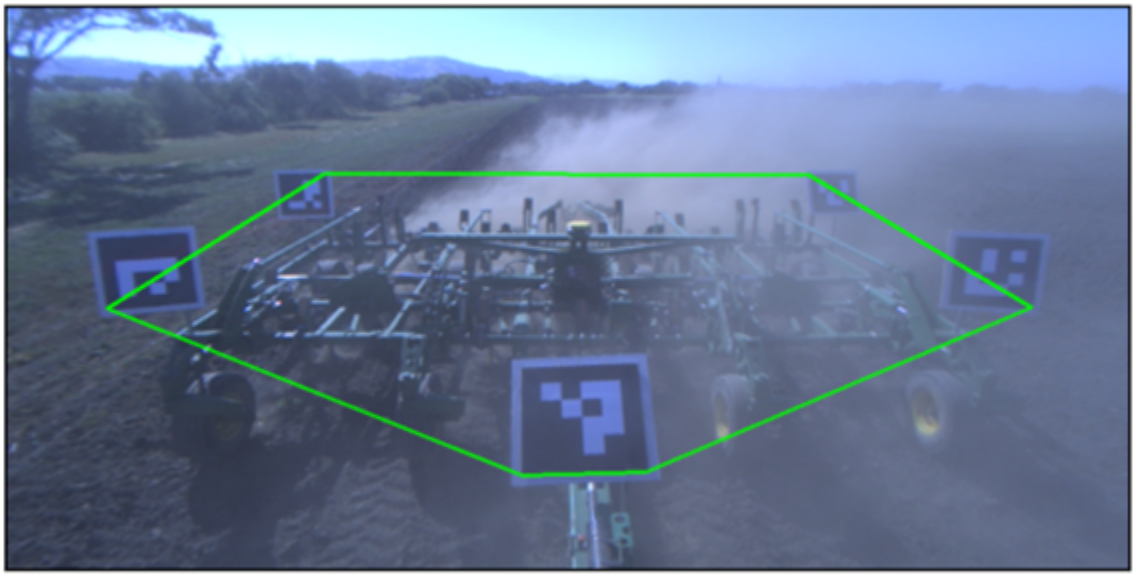
field of view to no longer contain all ArUco markers. An example of this error occurring during a turn can be seen below:

Figure 2.6 Not all markers present in camera field of view during turn



2) The second notable issue from this strategy is that mounting the ArUco markers permanently during operation would mean the detection algorithm would have to execute and detect the markers in every frame to define the implement boundary. An example of a defined implement boundary that is detected in every frame is seen below. The green line defines the implement boundary established by detecting the ArUco markers. However, this poses problems as the algorithm would have to successfully generate this bounding box in every frame to pass the implement boundary on to downstream robotics tasks. As stated in the first problem, not all ArUco markers are present in every frame, resulting in frames with missing markers

Figure 2.7 Every ArUco marker must successfully be detected within every frame



After experimentation it was decided that a more feasible option would be to place the markers strategically around the implement to capture one image frame with the markers present. When this frame is acquired, the implement boundary defined by the ArUco markers would become a saved parameter, thus allowing the ArUco markers to be physically removed from the implement during operation.

Now, data needs to be collected and processed to create these implement boundary parameters.

Data Collection and Detection Algorithm

To convert an image with ArUco markers into viable boundaries for the implement, a few steps must be taken, which will be explained in the following. However, first, a high-level overview of the process needs explained for understanding.

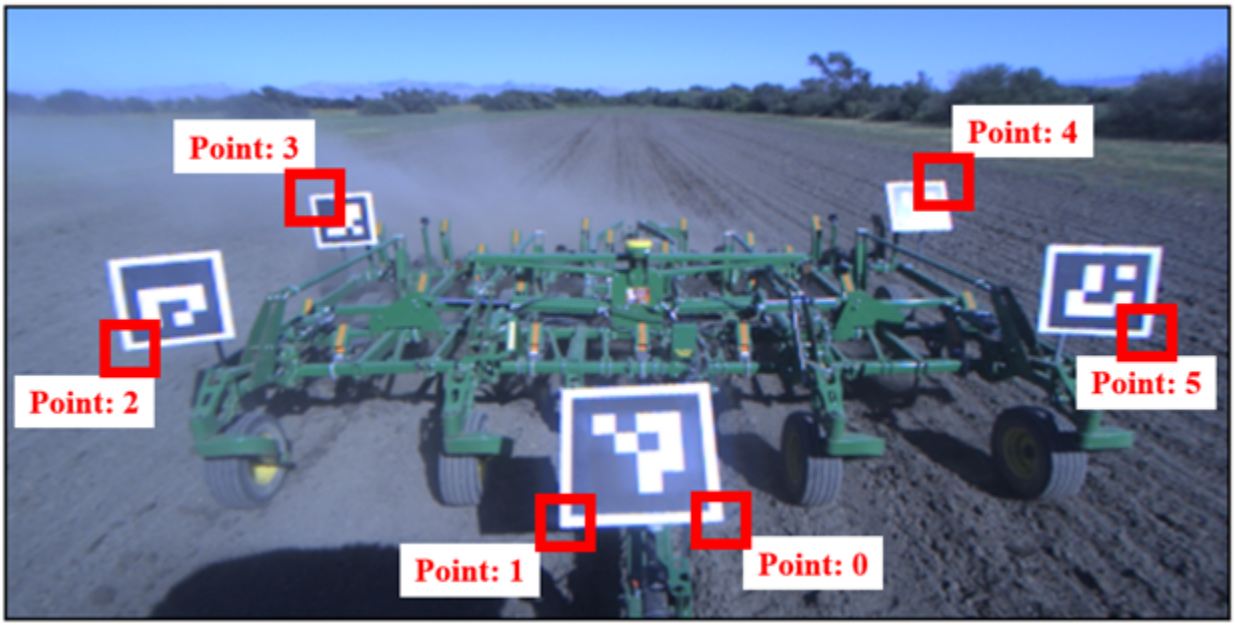
In an image containing the ArUco markers, the distance between the camera and the marker must be known, as this allows the position of the marker to become known in three-

dimensions. A raw image with ArUco markers is only two-dimensional, as seen in the examples above. To add a third dimension—depth—to each pixel in the image, a stereo pair of cameras must be utilized. Stereo pair cameras utilize disparity between two camera imagers located a fixed distance apart to estimate the depth value of the pixels in one image. Once the position of these ArUco markers is known in three-dimensions, then their position with respect to the implement can be stored as a fixed value, thus creating the implement boundary.

To begin, the specific pixels need identified that will be utilized to define the implement boundary. As from these pixels, the estimated pixel depth from the stereo pair will be utilized to estimate the position of the pixel in three-dimensions. Six pixels are identified within the image to define the implement boundary. The reason these six pixels were selected is because they are physically the farthest away from the implement frame in a top-down perspective. This allows the implement boundary to be defined at the farthest physical location from the implement, thus generating the largest implement boundary possible.

The ArUco marker detection algorithm returns a list of pixel coordinates for each ArUco marker identified. The list of coordinates contains the tuples defining the four corner pixels of each ArUco marker. The six pixels selected can be seen in the diagram below:

Figure 2.8 Specific pixels to be identified with detection algorithm



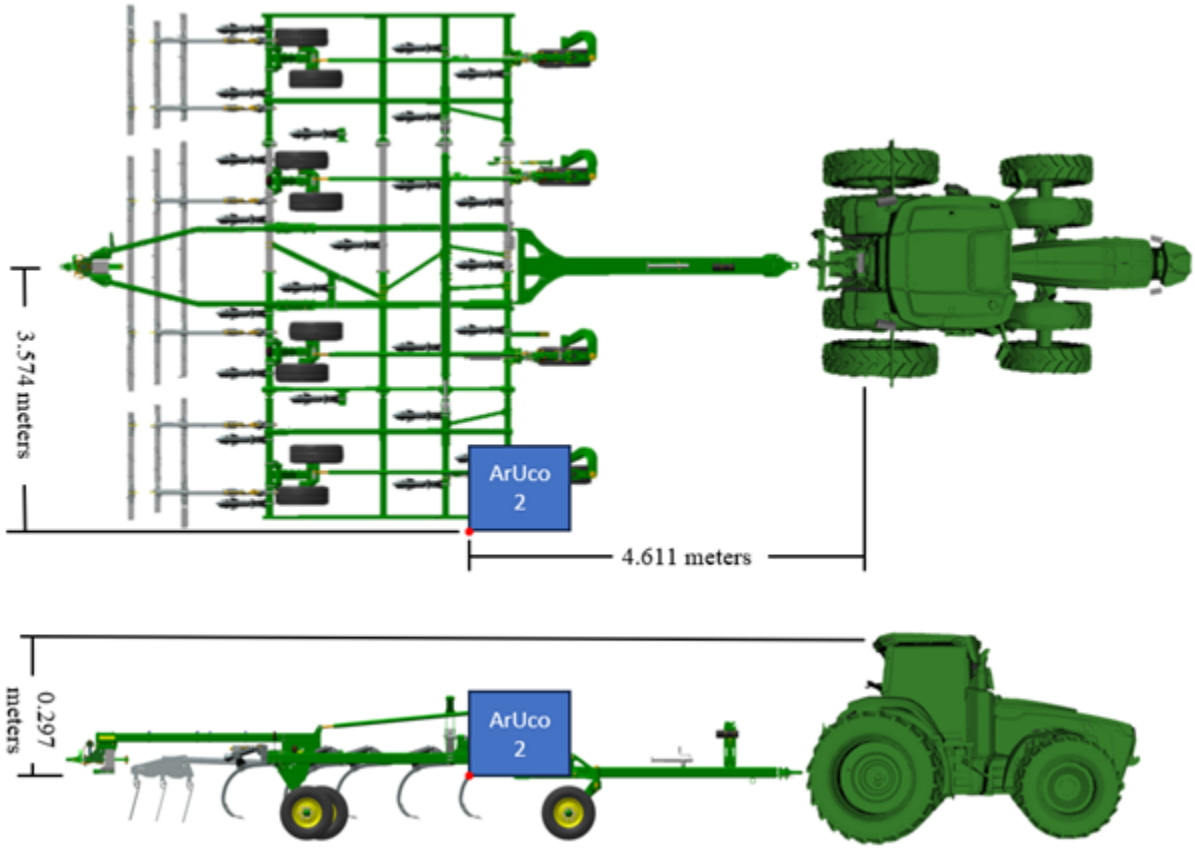
The implementation of this pixel coordinate extraction and depth estimation was done in an offline Jupyter notebook. The input to the notebook is the rectified, normalized image with corresponding depth estimations from the stereo camera pair. The output of the notebook is the corresponding X, Y, Z, depth estimation values with respect to the camera origin for each of the six specific points of interest. The depth model output for each point of interest is found in the table below:

Table 2.2 Predicted stereo-depth for each point of interest

ArUco Marker ID	Point Number	Image Frame Pixel Coordinate	Depth Model Output wrt Camera Origin(meters)
0	1	(471, 429)	[-0.242, 0.936, 2.863]
0	0	(585, 425)	[0.36, 0.902, 2.822]
1	2	(100, 288)	[-3.574, 0.297, 4.611]
2	3	(295, 165)	[-3.331, -1.329, 8.081]
3	4	(734, 160)	[3.372, -1.447, 8.331]
4	5	(936, 278)	[3.645, 0.214, 4.669]

An example of the above table: from the Index 1 ArUco marker displayed in the above figure, the bottom-left pixel associated with the bottom left corner of the fiducial marker was identified to extract the depth model output for this pixel. The pixel coordinates returned by the detection algorithm are (100, 288) from the origin—the top left corner of the image. With this coordinate pair, the depth model estimation for this pixel is parsed, returning values of [-3.574, 0.297, 4.611] with units of meters. This information tells us that the bottom left corner of the Index 1 marker—Point 2—is located 3.574m left, 0.297m below, and 4.611m away from the origin of the left camera imager in three-dimensions. An illustration can be found below:

Figure 2.9 Example of stereo-depth output in three-dimensions



By parsing out the depth estimation for only one specific pixel, the depth estimation utilized is subject to error defined by the error associated with the depth estimation model. An

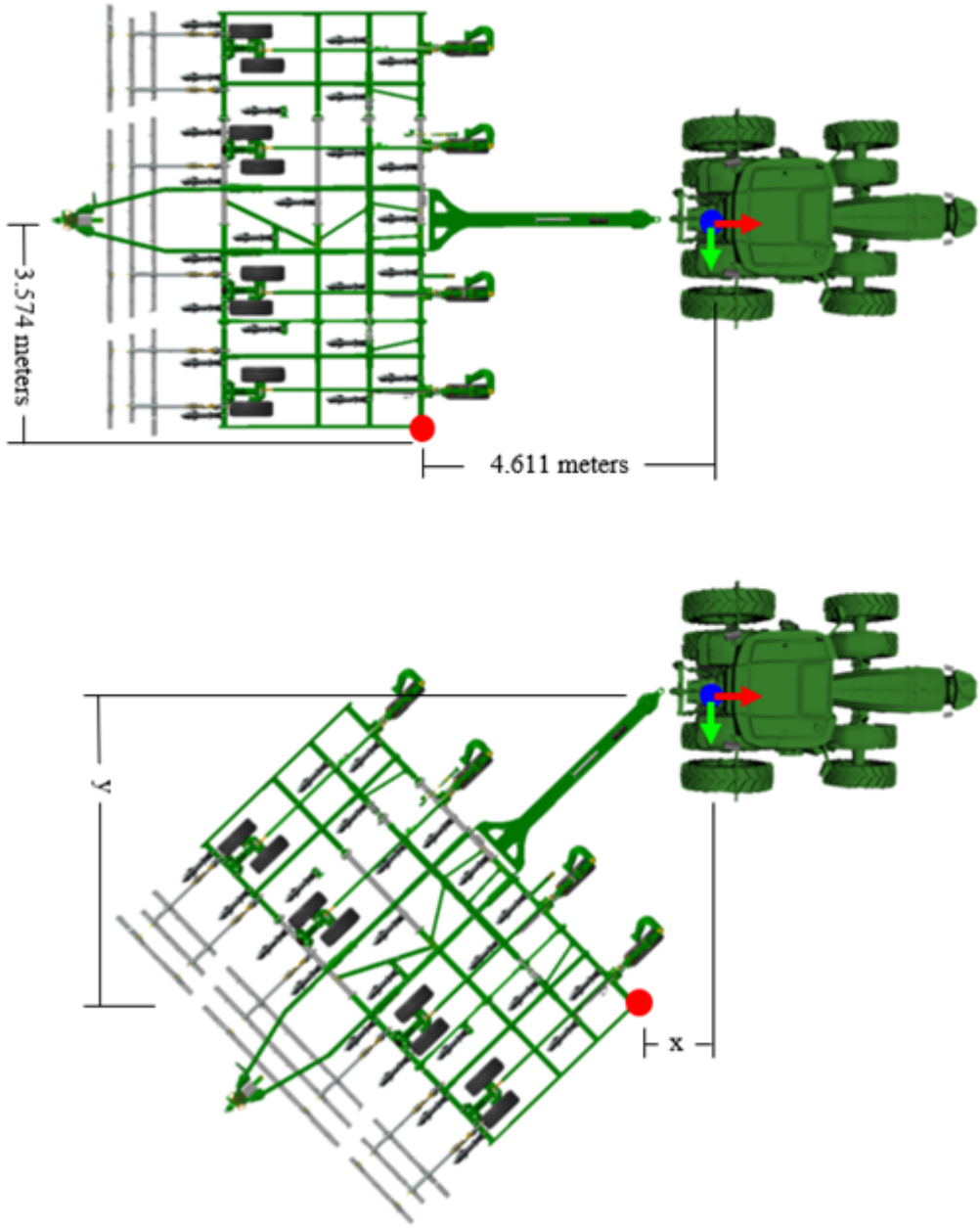
area of further development to alleviate this error induced and make the depth prediction more robust is to average the depth model prediction from a small group of nearby pixels. This work was not completed within the scope of this project.

Now with the location of these physical markers known in three-dimensions with respect to the camera origin, the coordinate positions need to transformed to have respect to the implement origin.

Robotics Operating System Implementation for Transformation

Now, the physical location of these coordinates is known in three-dimensions with respect to the camera origin. These coordinates work to define the implement boundary in three-dimensions while stationary because the entire system is static. However, once the machine starts moving, a pivot point is introduced at the hitch pin between the tractor and the implement. Therefore, when the machine turns the coordinates previously defined with respect to the camera origin are no longer relevant to define the implement boundary because the implement has physically changed location with respect to the camera origin. The diagram below explains the issue created by inducing the pivot point:

Figure 2.10 Point position estimation error induced by implement pivot point



To resolve this issue induced by the pivoting action of the implement, a transformation must be completed to know the three-dimensional position of the markers with respect to the implement. The coordinates must be transformed from having respect to the camera origin to having respect to the implement origin. To perform this transformation, the Robotics Operating System 2 (ROS2) transformation tree is utilized.

To execute this transformation a ROS2 node was created named `point_transform`. This node is a tf_listener that listens to the static transforms published on the /tf_static topic and dynamic transforms published on the /tf topic to compute the matrix multiplication needed to transform the coordinates from one frame to the next. Currently, this node is not real-time and publishes only one point at a time from a hard coded user input.

This node was utilized to transform the original coordinates with respect to the camera origin to coordinates with respect to the implement origin. The output of the node is found in the table below:

Table 2.3 Transformed coordinate location with respect to implement origin

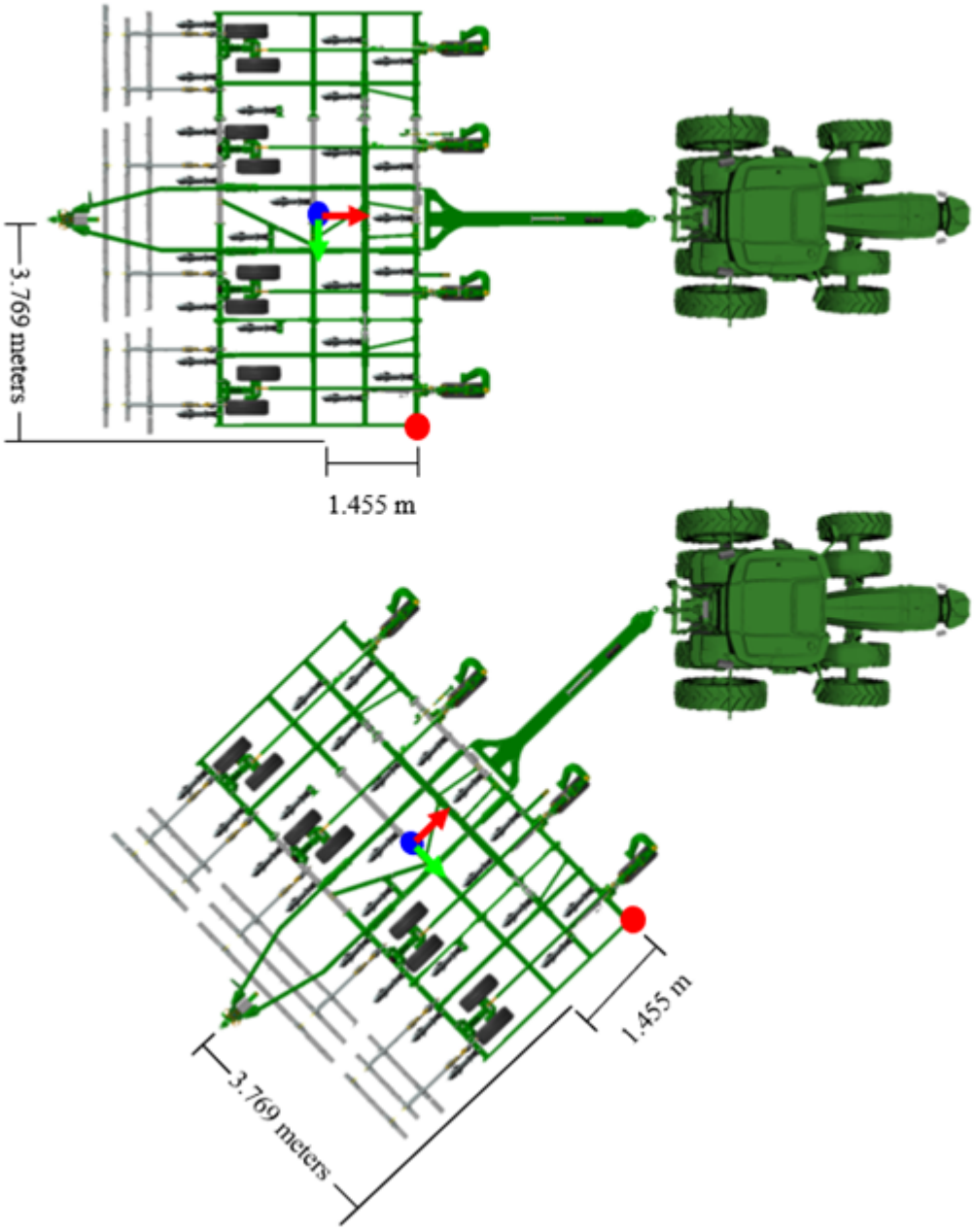
ArUco Marker ID	Point Number	Image Frame Pixel Coordinate	Depth Model Output wrt Camera Origin(meters)	Transformed Coordinates wrt Implement Origin (meters)
0	1	(471, 429)	[-0.242, 0.936, 2.863]	[3.353, -0.413, 1.357]
0	0	(585, 425)	[0.36, 0.902, 2.822]	[3.347, 0.202, 1.384]
1	2	(100, 288)	[-3.574, 0.297, 4.611]	[1.455, -3.769, 1.339]
2	3	(295, 165)	[-3.331, -1.329, 8.081]	[-2.421, -3.359, 1.732]
3	4	(734, 160)	[3.372, -1.447, 8.331]	[-2.643, 3.174, 1.712]
4	5	(936, 278)	[3.645, 0.214, 4.669]	[1.374, 3.484, 1.414]

An example of the coordinate transform from the camera origin to the implement origin: Point Number 2 in the examples shared above is pixel coordinate (100, 288). This information tells us that the bottom left corner of the Index 1 marker—Point 2—is located 3.574m left, 0.297m below, and 4.611m away from the origin of the left camera imager in three-dimensions. Now, the ‘point_transform’ ROS2 node utilizes existing static transformation matrices to transform this point to have respect to the implement origin, resulting in [1.455, -3.769, 1.339] in units of meters. This information tells us that the bottom left corner of the Index 1 marker—Point 2—is located 1.455m in front, 3.796m right, and 1.339m above the origin of the implement in three-dimensions.

This information is valuable because this position is static with respect to the implement origin! Therefore, regardless of the position of the implement with respect to the tractor, this point with always be 1.455m in front, 3.796m right, and 1.339m above the origin of the implement.

Since this position is static and always known, it can be utilized to define the implement boundary for this specific implement! A diagram illustrating the position with respect to the implement origin can be found below:

Figure 2.11 Transformed point location in three-dimensions with respect to implement origin

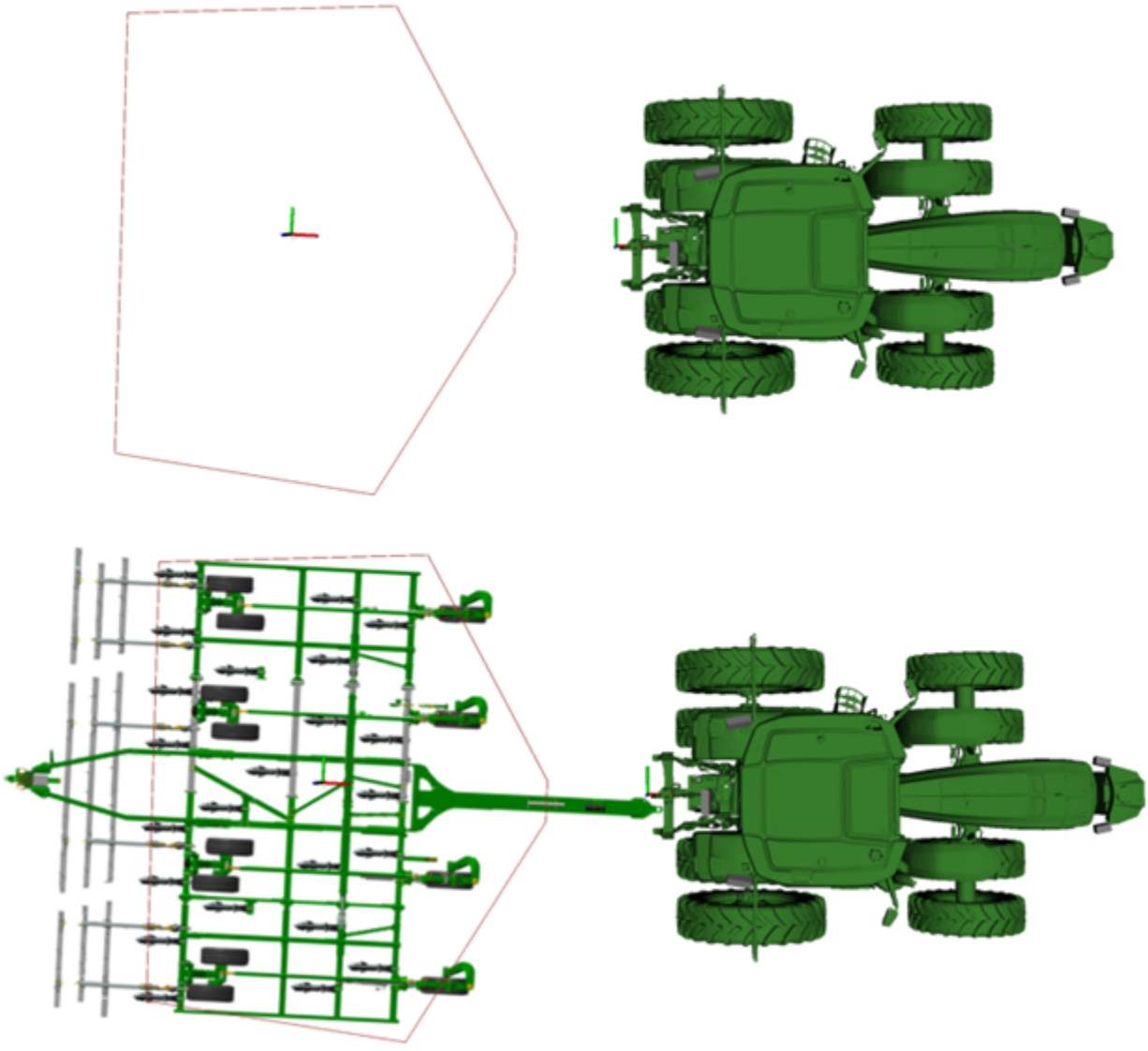


Implement Boundary Visualization

With all six of these points transformed to have respect to the implement origin, the points can be utilized to generate a reliable implement boundary. Much of this work is easier to comprehend when visualized, therefore a visualization was created to display the newly defined implement boundary.

A new ROS node was created called ‘new_mask_viz’. This node creates a PolygonStamped message, drawling a polygon defined by the six points with respect to the implement origin. This PolygonStamped message is published and visualized within the RVIZ2 simulation environment to display the newly generated implement boundary. An image displays the newly defined implement boundary within RVIZ2 and can be seen below:

Figure 2.12 RVIZ2 Visualization of newly defined implement boundary



The newly defined implement boundary is visualized by the red polygon to the left of the tractor as seen in the image.

When the implement is placed on top of the visualized implement boundary, it is noted that the newly defined implement boundary is not as large as the implement. This is because the points defined by the rear ArUco markers in this experiment are mounted on the rear frame of the chisel plow. Therefore, the rear markers are not physically behind the drag harrow that is drug behind the implement. This causes the newly defined implement boundary to be shallower than the implement, as the location of the ArUco markers does not contain the implement drag harrow.

If this ArUco marker data was collected again, the ArUco markers could be placed on stands to physically outline the entire implement footprint from a top-down perspective. In this data collection scenario, the same processing pipeline would be utilized and could result in a larger, more accurate implement boundary.

Results

As seen in the image above, visualizing the newly defined implement boundary, this research project has confirmed the ability to utilize ArUco fiducial markers within the Blue River Technology perception stack to define the implement boundary parameters.

To quantify the boundary parameters created with ArUco markers and compare them with hard-coded boundary parameters, the area of the implement boundary was calculated as seen in the table below.

Table 2.4 Comparing area contained by implement boundary definition processes in square meters

Area contained by ArUco defined implement boundary:	Area contained by current implement boundary:
35.1472 m^2	63.622 m^2

By comparing these areas, one will notice that the ArUco defined boundary is only 55.24% of the area defined by the known Deere implement dimensions. This is because the points defined by the rear ArUco markers in this experiment are mounted on the rear frame of the chisel plow. Therefore, the rear markers are not physically behind the drag harrow that is drug behind the implement. This causes the newly defined implement boundary to be shallower as the green, modeled cuboid containing the implement drag harrow.

References

Cesar, Diego & Gaudig, Christopher & Fritsche, Martin & Reis, Marco & Kirchner, Frank. (2015). An evaluation of artificial fiducial markers in underwater environments. 10.1109/OCEANS-Genova.2015.7271491.

Detection of ArUco Markers (2024). OpenCV. Retrieved from https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html

Freda, Anthony. (2022). Title of Journal Article: What Are QR Codes and How Do You Scan Them? Retrieved from https://www.avast.com/c-what-is-qr-code-how-to-scan#:~:text=A%20QR%20code%20(short%20for,to%20store%20and%20access%20data.

GeoForAll Lab (2023). Find-GCP. GitHub. https://github.com/zsiki/Find-GCP

Lambert, Kiley. (2023). Title of Journal Article: How Deere is preparing for a fully autonomous farm by 2030. https://www.cnbc.com/2023/12/01/how-deere-is-preparing-for-a-fully-autonomous-farm-by-2030.html

M. Kalaitzakis, S. Carroll, A. Ambrosi, C. Whitehead and N. Vitzilaios, "Experimental Comparison of Fiducial Markers for Pose Estimation," 2020 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 2020, pp. 781-789, doi: 10.1109/ICUAS48674.2020.9213977.

Robinson, Rachel. (2019). Losing Ground: Urban Sprawl Brings Challenges and Opportunities for Agriculture. American Angus Association. Retrieved from https://www.angus.org/Media/IAmAngus/losingGround#:~:text=Urban%20sprawl%20is%20a%20complex,Reach%20out

United Nations. (2017). Department of Economic and Social Affairs. Retrieved from https://www.un.org/en/desa/world-population-projected-reach-98-billion-2050-and-112-billion-2100

Vaughn-Uding, Vincent. (2021). Title of Journal Article: Overexploitation of the Ogallala Aquifer. https://storymaps.arcgis.com/stories/d818872aa6df4cfa9683373e1e6b5ae6

Appendix A - Areas of Further Development

- To implement this into the existing software stack at Blue River Technology and Deere, further development is needed to make these algorithms real time, as they are currently functional independently and in post processing only.
- Utilize multiple surrounding pixels from the identified point pixel and average the estimated depth from the surrounding pixels to estimate the depth of the point defining the implement boundary.
- Gather data while optimizing the placement of the ArUco markers to cover the entire footprint of the implement from a top-down perspective
- Gather data on wider implements