

**VOLUME MEASUREMENT OF BIOLOGICAL MATERIALS IN
LIVESTOCK OR VEHICULAR SETTINGS USING COMPUTER VISION**

by

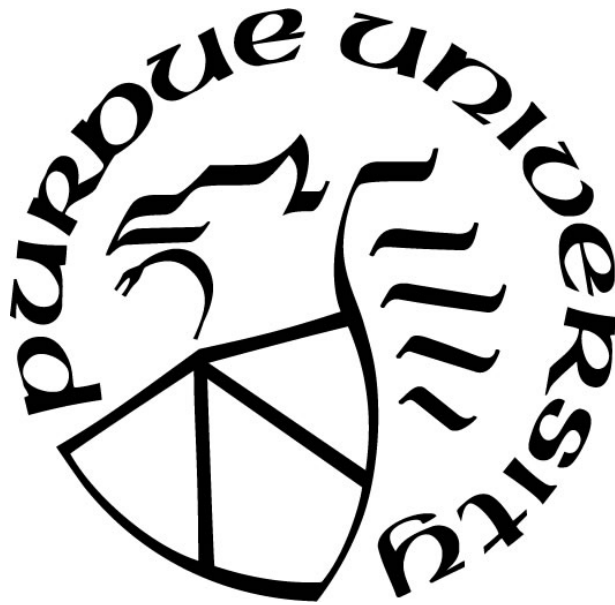
Matthew Rogers

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy



School of Agricultural and Biological Engineering

West Lafayette, Indiana

August 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Dennis Buckmaster, Chair

School of Agricultural and Biological Engineering

Dr. Jian Jin

School of Agricultural and Biological Engineering

Dr. Ankita Raturi

School of Agricultural and Biological Engineering

Dr. Amy Reibman

School of Electrical and Computer Engineering

Approved by:

Dr. Nathan S. Mosier

Dedicated to God and Family for leading me to Purdue and keeping me on the right path. I especially remember having a conversation with my grandfather, Samuel Keel Kelly at his dining table one morning in Lancaster, South Carolina where we were talking about the possibility of me pursuing a PhD. I wasn't sure at the time if a PhD was right for me, but he encouraged me to do it. It is my hope that the skills that I have developed at Purdue University can be used for God's glory, wherever he may lead me.

ACKNOWLEDGMENTS

Thank you to the CNH Industrial New Holland Innovations team, John Posselius and Dr. Christopher Foster, for making this dissertation possible. Thank you to the PI's, Professors Dennis Buckmaster and James Krogmeier, for arranging the cooperative effort between Purdue and CNH Industrial. Thank you to Research and Extension Experimental Learning for Undergraduates (REEU) student participant Kelly Lewis from Oklahoma State University for working with me to construct and run some of the testing systems designed for this work. Thank you to Jeff Fields, manager of the Purdue Animal Sciences Research and Education Center farm, for providing both equipment and the arrangement of corn to be delivered and used for testing. Thank you to Dr. Jackie Boerman for providing TMR and access to the dairy barns. Thank you to Scott Brand, manager of the ADM Agricultural Innovation Center, for providing both outdoor and indoor testing space and repairing a broken PTO shaft.

TABLE OF CONTENTS

LIST OF TABLES.....	9
LIST OF FIGURES	10
ABSTRACT.....	14
1. STATE OF THE ART OF AGRICULTURAL VEHICLE COMPUTER VISION FOR BIOLOGICAL MATERIAL PERCEPTION	15
1.1 Chapter Abstract.....	15
1.2 Background.....	15
1.2.1 Camera Technologies.....	15
1.2.2 Laser-Based Systems.....	19
1.2.3 Other technologies.....	20
1.2.4 Knowledge Gap.....	21
1.3 Goals.....	22
2. ENVIRONMENTAL CHAMBER COMPARISON OF STEREO CAMERA AND LIDAR	
23	
2.1 Abstract.....	23
2.2 Background.....	23
2.2.1 Hardware	23
2.2.2 Dust Problem.....	24
2.2.3 Software	25
2.2.4 Gap	27
2.3 Objectives	27
2.4 Streaming Point Clouds in Python	27
2.4.1 Methods.....	27
2.4.2 Results and Discussion.....	29
2.5 Camera Testing in Grain Transport Conditions	30
2.5.1 Methods.....	30
LiDAR.....	30
Stereo Vision	31
2.5.2 Results and Discussion.....	32

LiDAR.....	32
Stereo Vision	35
2.6 Dust Box Design.....	36
2.7 Camera Testing in Dust Box	39
2.7.1 Methods.....	39
2.7.2 Results	44
No Dust Comparison.....	45
Low Dust Comparison	49
High Dust Comparison.....	53
Return Mode Comparison	56
2.7.3 Discussion	58
No Dust	58
Low Dust.....	58
High Dust	59
Lighting	60
Return Mode.....	60
2.8 Conclusions and Recommendations.....	61
3. VOLUME MEASUREMENT OF PARTICULATE BIOLOGICAL MATERIALS	62
3.1 Abstract.....	62
3.2 Introduction	62
3.2.1 Literature Review	63
Discussion of Feed and Animal Tracking Possibilities.....	64
3.3 Objectives	65
3.4 Methods and Materials	65
3.4.1 Volume Measurement Theory.....	66
3.4.2 Assumptions	68
3.4.3 Calibration.....	69
3.4.4 Biological Material Volume Testing.....	71
3.5 Results and Discussion.....	75
3.5.1 TMR	75
3.5.2 Corn Grain.....	78

3.5.3	Discussion	81
3.6	Conclusion	84
4.	NEAR REAL-TIME GRAIN VOLUME OF LOADING VESSELS	85
4.1	Abstract	85
4.2	Background	85
4.3	Objectives	89
4.4	Methods and Materials	89
4.4.1	Template matching	90
4.4.2	Depth masking	93
4.4.3	HSV masking	95
4.4.4	Color masking	98
4.4.5	Normal Vector masking	98
4.4.6	Combining all masking strategies to the depth stream	101
4.4.7	Volume Calculation	103
4.5	Discussion	105
4.5.1	Limitations	106
4.6	Conclusions	110
5.	Overall conclusions and recommendations	111
5.1	Engineering Contributions	111
5.2	Scientific Contributions	111
5.3	Recommendations for Future Work	112
APPENDIX 1. GRAIN VOLUME MEASUREMENT PYTHON CODE FOR FEED BUNKS		114
APPENDIX 2. ROS POINT CLOUD SCATTERPLOT PYTHON CODE		117
APPENDIX 3. RGB AND HSV IMAGE CHANNEL THRESHOLDING		119
APPENDIX 4. VESSEL/GRAIN CART RELATIVE POSITION TRACKING PYTHON CODE		124
APPENDIX 5. TRUCK STAKE POCKET TEMPLATE MATCHING		126
APPENDIX 6. VESSEL VOLUME CALCULATION		128
APPENDIX 7: ROS USB CAMERA IMAGE STREAMING IN OPENCV		131
APPENDIX 8. CARNEGIE ROBOTICS S21 DATA SHEETS		132

APPENDIX 9. VELODYNE LYDAR DATA SHEETS	139
APPENDIX 10. PANAVISE DRAWING.....	141
APPENDIX 11. VIDEOS LINKS	142
REFERENCES	143

LIST OF TABLES

Table 3.1. Software used for this work.	73
Table 3.2. Hardware and materials used for this work.	74
Table 3.3. Ranges and max values of corn volume measurement scenarios. Scenarios 7 and 8 should have the same volume; 9, 10, and 11 should have the same volume; and 12 and 13 should have the same volume.	78

LIST OF FIGURES

Figure 1.1. Monitor view from the John Deere Active Fill Control (John Deere, 2022)	18
Figure 1.2. Sensor array used for grain depth detection. (Gaard 2012).....	21
Figure 2.1. Data pipeline for data collection used in chapters 2, 3, and 4.	28
Figure 2.2. Data post-processing pipeline for image processing and volume calculations.	29
Figure 2.3. (a) VLP-16 LiDAR sensor mounted on spotlight, (b), (c) LiDAR overlooking truck bed.....	31
Figure 2.4. (a) Close-up view of mounted stereo camera on the grain cart, (b) Stereo camera overlooking truck, (c) PanaVise mounting system (PanaVise, 2022).	32
Figure 2.5. RVIZ image of the LiDAR scan of the empty truck.	33
Figure 2.6. RVIZ image of the LiDAR scan of the truck during filling.	34
Figure 2.7. RVIZ image of the LiDAR scan of the truck during filling, with grain surface progressed near completion.	34
Figure 2.8. RVIZ image of the LiDAR scan of the filled truck.....	35
Figure 2.9. Imagery during truck filling. (a) color image, (b) grayscale depth, and (c) depth image with blue pixels representing "NaN" values caused either by dust or occlusion.....	36
Figure 2.10. (a) Constructed wood dust chamber and (b) lighting with the mounted LiDAR shown at the far end.....	38
Figure 2.11. (a) View inside dust chamber and (b) arm holes for access.	38
Figure 2.12. “Wedding cake” sensor target.	39
Figure 2.13. LiDAR, RGB, and stereo cameras mounted for dust chamber testing.....	40
Figure 2.14. Velodyne LiDAR settings used during testing.	41
Figure 2.15. Stereo camera configuration page brought up using the <i>roslaunch rqt_reconfigure</i> command in terminal.	42
Figure 2.16. (a) Hair dryer and (b) air nozzle used for dust chamber testing.	43
Figure 2.17. Point cloud collection process for dust chamber.....	44
Figure 2.18. LiDAR point clouds and color image in full light conditions with no dust. Point clouds: (a) isometric, (b) side, (c) front, (d) top. (e) Color image.....	46
Figure 2.19. (a) LiDAR point cloud and (b) color image of half-light conditions with no dust. .	47
Figure 2.20. Stereo camera point clouds and color image in full light conditions with no dust. Point clouds: (a) front, (b) top, (c) isometric. (d) Color image.	48

Figure 2.21. (a) Stereo camera point cloud and (b) color image of half-light conditions with no dust.....	49
Figure 2.22. LiDAR point clouds and color image in full light, low dust conditions. Point clouds: (a) isometric, (b) side, (c) front, (d) top. (e) Color image.....	50
Figure 2.23. (a) LiDAR point cloud and (b) color image of half-light conditions with low dust. 51	
Figure 2.24. (a) LiDAR point cloud and (b) color image of no light conditions with low dust. ..	51
Figure 2.25. Stereo camera point clouds and color image in full light conditions with low dust. Point clouds: (a) front, (b) top, (c) isometric. (d) Color image.....	52
Figure 2.26. (a) Stereo camera point cloud and (b) color image of half-light conditions with low dust.....	53
Figure 2.27. (a) LiDAR point cloud and (b) color image of full-light conditions with high dust. 54	
Figure 2.28. (a) LiDAR point cloud and (b) color image of half-light conditions with high dust.	54
Figure 2.29. Stereo camera point clouds and color image in full-light conditions with high dust. Point clouds: (a) front, (b) top, (c) isometric. (d) Color image.....	55
Figure 2.30. (a) Stereo camera point cloud and (b) color image of half-light conditions with high dust.....	56
Figure 2.31. LiDAR strongest return mode. (a) Point cloud and (b) color image of half-light conditions with high pressure dust. One light fell off the wall.....	57
Figure 2.32. LiDAR dual return mode. (a) Unregistered points and (b) registered points (right) of half-light conditions with high pressure dust. One light fell off the wall.	57
Figure 2.33. LiDAR last return mode. Half-light conditions with high pressure dust. One light fell off the wall.	58
Figure 3.1. (a) Illustration showing the relationship between pixel and the depth it represents. (b) Each pixel (below) has a volume associated with it.	67
Figure 3.2. Importance of the parallel plane assumption illustrated. If camera lens plane (top right) is not parallel with the ground plane, extra volume (redlines) is added to the volume estimate of the cubic structure.	69
Figure 3.3. Parallax problem of underestimating volume. Camera lens is represented by top line and unmeasured area is represented by the red area of the square frustum.....	69
Figure 3.4. Calibration images showing the paper on (a) ground and (b) raised levels.	70
Figure 3.5. Flowchart of volume calculation process, from calibration to final calculation.	71
Figure 3.6. Buckets of TMR and spread TMR.	72
Figure 3.7. Step ladder and camera mount used for TMR imaging in the lab.....	73

Figure 3.8. Bucket of TMR, still in tact, and a flat surface (left: color image; right: depth image).	75
Figure 3.9. Two buckets of TMR (in tact) on a flat surface (left: color image; right: depth image).	75
Figure 3.10. Two buckets of TMR on a flat surface with one spread (left: color image; right: depth image).	76
Figure 3.11. Two buckets of TMR on a flat surface with both spread (left: color image; right: depth image).	76
Figure 3.12. Two buckets of TMR on a flat surface both spread and or spread further (left: color image; right: depth image).	76
Figure 3.13. Two buckets of TMR on a flat surface with both spread further (left: color image; right: depth image).	77
Figure 3.14. A spread TMR on a flat surface after a 5 gallon bucket was removed (left: color image; right: depth image).	77
Figure 3.15. Flat surface showing all feed removed (left: color image; right: depth image).	77
Figure 3.16. Flat surface showing approximately one gallon of grain with craters. ((a) color image; (b) depth image).	78
Figure 3.17. Flat surface showing approximately one gallon of grain with craters resembling a “smiley face.” ((a) color image; (b) depth image).	79
Figure 3.18. Flat surface showing approximately two gallons of grain in a pile. ((a) color image; (b) depth image).	79
Figure 3.19. Flat surface showing approximately two gallons of grain with one crater. ((a) color image; (b) depth image).	79
Figure 3.20. Flat surface showing approximately two gallons of grain smoothed out. ((a) color image; (b) depth image).	80
Figure 3.21. Flat surface showing approximately four gallons of grain in a pile. ((a) color image; (b) depth image).	80
Figure 3.22. Flat surface showing approximately four gallons of grain with three craters. ((a) color image; (b) depth image).	81
Figure 3.23. Flat surface showing a handful of corn grain tossed in the middle. ((a) color image; (b) depth image).	81
Figure 3.24. An example of occlusion. The image on the left is a color image of 2 buckets of TMR. Right is the same image from depth feed showing occlusion in blue.	83
Figure 4.1. Data collection pipeline for grain truck loading	90
Figure 4.2. Actual stake pocket strip used for template matching.	91

Figure 4.3. Arrows pointing to the near-edge stake pockets of the passing truck.	92
Figure 4.4. Relative pass percentage being calculated from the vertical stake pocket.	92
Figure 4.5. Original color image of truck with no computer vision techniques applied.	94
Figure 4.6. Color image of truck with surrounding ground pixels masked out.	94
Figure 4.7. Color image of truck with bed rails and crossmembers masked out.	95
Figure 4.8. Trackbars used to adjust maximum and minimum HSV and RGB values of an image.	96
Figure 4.9. HSV color space image converted from the color image.	97
Figure 4.10. Masked color image with the mask carefully chosen from ranges of color spaces within the HSV image.	97
Figure 4.11. Color image masked from carefully chosen RGB values.	98
Figure 4.12. Depth image that has been converted to represent normal vectors coming off of the surface formed by the depth pixels.	100
Figure 4.13. Color image masked by the normal vector mask of the depth image.	101
Figure 4.14. All masked were applied to this color image.	102
Figure 4.15. All masks except the normal vector mask have been applied to the normal vector stream.	102
Figure 4.16. Calculated volume of the grain truck using all masks discussed.	104
Figure 4.17. Calculated volume of grain truck using all masks except the normal vector mask.	105
Figure 4.18. Dust plumes formed when loading grain.	107
Figure 4.19. Dust and occlusion shown in blue.	107
Figure 4.20. The depth mask on the left is lagging behind the color image stream on the right.	109

ABSTRACT

A Velodyne Puck VLP-16 LiDAR and a Carnegie Robotics Multisense S21 stereo camera were placed in an environmental testing chamber to investigate dust and lighting effects on depth returns. The environmental testing chamber was designed and built with varied lighting conditions with corn dust plumes forming the atmosphere. Specific software employing ROS, Python, and OpenCV were written for point cloud streaming and publishing. Dust chamber results showed while dust effects were present in point clouds produced by both instruments, the stereo camera was able to “see” the far wall of the chamber and did not image the dust plume, unlike the LiDAR sensor. The stereo camera was also set up to measure the volume of total mixed ration (TMR) and shelled grain in various volume scenarios with mixed surface terrains. Calculations for finding actual pixel area based on depth were utilized along with a volume formula exploiting the depth capability of the stereo camera for the results. Resulting accuracy was good for a target of 8 liters of shelled corn with final values between 6.8 and 8.3 liters from three varied surface scenarios. Lessons learned from the chamber and volume measurements were applied to loading large grain vessels being filled from a 750-bushel grain cart in the form of calculating the volume of corn grain and tracking the location of the vessel in near real time. Segmentation, masking, and template matching were the primary software tools used within ROS, OpenCV, and Python. The S21 was the center hardware piece. Resulting video and images show some lag between depth and color images, dust blocking depth pixels, and template matching misses. However, results were sufficient to show proof of concept of tracking and volume estimation.

1. STATE OF THE ART OF AGRICULTURAL VEHICLE COMPUTER VISION FOR BIOLOGICAL MATERIAL PERCEPTION

1.1 Chapter Abstract

This chapter includes a review of the use of cameras and laser-based depth perception instruments to image and quantify biomass materials, primarily in agriculture. Efforts from academia (generally published) and industry (unpublished or perhaps patented) were included. This overview identified gaps and opportunities leading to the overall aims of this work related to stereo vision vs LiDAR comparison, volume estimation, and tracking vessel volumes in real time.

1.2 Background

1.2.1 Camera Technologies

Camera technology has been used as a tool to gather volume, depth, or classify information of agricultural and biological materials. As an example, stereo vision has been used to measure the volume of soil contained in excavator buckets (Anwar et al., 2014). Exact information on the camera models used in this work were not mentioned, but a pair of high definition USB cameras were used to form a stereo camera. A point cloud of an empty bucket was formed and compared to the point cloud of the filled bucket. Markers for the container edges were manually entered into the algorithm. Error in estimated volume got progressively better as the bucket was filled, from a 15.4 % overestimation of volume at one-third full to a 0.57 % overestimation of volume for the full bucket.

An Intel RealSense RGB-D R200 infrared stereo camera was used for fruit detection in a grape orchard while mounted on a Niko crawler tractor (Milella et al., 2019). Robot Operating System (ROS) was used to acquire the images. Together with a VGG-19 convolutional neural network, grape clusters were identified with a 91.52% accuracy.

A 120 mm baseline 6 mm focal lens Bumble Bee stereo camera was used to distinguish between cut and uncut corn rows when mounted on a combine harvester (Rovira-Mas et al., 2007). The

developed algorithm was used as the combine was driven through the field along the edges of the harvested corn, and the computer vision results were monitored via an on-board computer.

A stereo camera was used to create a depth map and segment the crop from the ground, and then measure the crop height (Kim et al., 2021). To accomplish the depth map, a Tara USB 3.0 stereo camera was mounted on a tractor and driven through a crop field while the camera collected images. A good relationship to actual crop heights was achieved, with R^2 between 0.78 and 0.84 for Chinese cabbage, potato, sesame, radish, and soybean with mean range error percentages (MRE) ranging from 3.4 to 1.0.

Another study used what appears to be an RGB camera (camera not specified) and a neural network to measure the grain fill of a loading grain cart from a combine harvester (Shkanaev et al., 2017). The camera was mounted on the combine auger tube and captured the images of the grain filling the cart. 907 images were collected from the combine and then manually labelled with the following classes: grain stream, grain already in cart, cart interior surface, cart exterior surface, and background objects. 868 of the images, which were manipulated 24 times to enhance the data set, were trained in the neural net to allow the imaging system to segment the relevant parts of the images. The images were flipped horizontally, rotated, cropped by 20%, and the color channels were slightly changed. Grain already in the cart was identified with 90% accuracy with the remaining 39 image test data set, but it is unknown if the method was field tested. Grain volume was not calculated nor estimated.

Potter (2015) developed an automated grain auger position control system for use with a John Deere 9870 STS combine as part of an Iowa State and John Deere collaborative effort to automate combine unloading (SmartUnload project). The National Robotics Engineering Center of Carnegie Mellon University (CMU) developed the machine vision aspect of the project which consisted of two stereo cameras (make and model not provided), one mounted on the side of the combine to detect the cart and another mounted on the auger of the combine to allow the grain flow to be within the field of view of the camera. These cameras were interfaced with a laptop that in turn communicated with a MicroAutoBox which served as the auger controller. The auger was instrumented with an auger angle sensor, wobble sensor, and pressure sensor to detect auger angle,

grain flow, and swing cylinder pressure, respectively. All of these instruments communicated with the MicroAutoBox CAN bus messages from the combine also interacted with the MicroAutoBox.

In an earlier collaborative effort with the Robotics Center of CMU, the New Holland Machine Company (Now part of CNH Industrial) and CMU developed a fully autonomous New Holland 2550 Speedrower, appropriately named Demeter after the Olympian goddess of harvest and agriculture (Pilarski et al., 2002). The sensing and perception system was accomplished using two color cameras (model not specified) mounted on the cab to detect crop cut lines, crop row ends, and obstacles. The system successfully harvested a few hundred acres of forage during testing.

John Deere Active Fill Control is an automated trailer filling system for forage harvesters utilizing a stereo camera, image-processing module, and additional LED lighting (John Deere 2010). The product was developed in conjunction with the National Robotics Engineering Center of Carnegie Mellon University. Using a graphical user interface (gui), the operator inputs the desired fill level, fill strategy, edge distance, fill offset, and spout rotation offset. Fill offset places more chopped forage on one side of the trailer versus the other to offset for inclination. After the tractor is driven into detection range, the system automatically fills the trailer. The system can side load a trailer from front to back, back to front, front to back to front, and back to front to back. Figure 1.1 shows the John Deere Active Fill graphical user interface.



Figure 1.1. Monitor view from the John Deere Active Fill Control (John Deere, 2022)

Similar to the John Deere Active Fill Control system for forage harvesters, the Claas Auto Fill also makes use of 3D camera technology to control the flap of the Jaguar forage harvester and as well as the direction of the discharge spout (Claas, 2022). It does not appear that the system is fully autonomous, as the forage harvester operator still must use a video screen (known as Cemos) to monitor the filling progress at each point in the trailer. As one portion of the trailer begins to fill, the operator picks the next successive point in the trailer and the Auto Fill system moves the forage impact point of the spout to the next optimum point.

Similar to the John Deere Active Fill Control system, CNH Industrial also developed an automatic trailer filling system to unload forage harvesters (CNH Industrial, 2022). This system is known as the IntelliFill system for New Holland FR9000 series forage harvesters. Instead of a stereo camera, the system employs an infrared 3D camera for imaging the chopped forage. The system reportedly works equally well in bright sunlight and dark nights. The system was developed using model-based design. MathWorks MatLab Simulink controller models were used for algorithm development. In field testing, the Simulink controller model operating via laptop processed the images and relayed commands to the CAN bus which relayed to the machine controllers. The lateral movement of the spout, direction of flow out of the spout, camera angle, and alerts to the

driver are controlled by the controllers. New Holland did not specify if this system could be combined with an auto-steering device to make the process autonomous.

John Deere developed a patent for the automatic unloading of agricultural material from one vehicle to another (Bonefas 2018). In the claims section, the vehicle that is being filled is a transport vehicle. The system uses a stereo imaging device to identify the spout of the unloading vehicle and the perimeter of the holding container of the receiving vehicle. Edge detection is utilized to identify the spout in the images and later orientate the spout with respect to the vehicles. Other strategies, such as pattern or color detection may be used for spout detection. Edge detection is also used to identify the perimeter of the storage container of the receiving vehicle. Vehicle alignment is based on spatial offset between the spout and the container perimeter. The preferred embodiment section mentions using an imaging module to detect volume or fill level inside the loading container. Distance or range from the spout to the material profile within the container can be determined. This patent also mentions communications between the vehicles as a master/slave system, where the master vehicle can control the steering controller of the slave vehicle.

1.2.2 Laser-Based Systems

Laser-based instruments have also been used for sensing or perception of agricultural materials in field conditions similar to stereo cameras. In a work exploring laser scanner applications for crop productions, an ibeo-ALASCA XT laser scanner was mounted to the front area of a tractor and tested in biological material measurement scenarios (Ehlert and Heisig, 2013). It was tested to detect round bales in a field, measure height of crop stands, and estimate crop biomass. It sufficiently detected round bales and generated images of crop stands. Crop biomass was predicted using laser scan height return values within $R^2 = 0.96$ for wheat and $R^2 = 0.95$ for maize. The models were exponential in nature and other fit statistics such as mean square error were not provided.

In a separate study, an ibeo-ALASCA XT laser rangefinder scanner was used to determine winter wheat stand height. (Ehlert et al., 2010) The instrument was mounted on a tractor moving at 6, 12, 18, and 24 km/hr. Winter wheat height measurements coming from the laser scanner were

independent of the ground speed of the vehicle in the range of 6 to 24 km/h., suggesting the instrument is suitable for agricultural applications.

Mean diameters of orange fruit still on the tree were estimated within 1.0 mm using a ScanStation P20 laser scanner (Mendez et al., 2019). Although reliable, some scans were excessively slow (as long as 13 minutes).

A Sick LMS 200-30106 LiDAR was mounted to a combine while driving over four plots of wheat planted at different densities (Saeys et al., 2009). Densities ranged from 50 to 400 ears of wheat per square meter planted in intervals of 50 ears per square meter. Combine speed was tested at 2, 3, 4, and 5 km/hr. Data was collected with the thresher on and off. The standard deviation of the measured penetration depths was found to predict the crop density within .81 and .96 R^2 . Model slope or other details were not provided for the linear model.

A SICK LMS 291 LiDAR was mounted on a tractor to identify and measure straw output of cut straw from a combine (Lenaerts et al., 2012). The detected swath path was easily distinguishable from the ground on the laser height map (presumably for auto guidance during baling).

1.2.3 Other technologies

In his Iowa State University thesis, Gaard (2012) attached a system of four Pepperl and Fuchs UC6000-30GM-IUJR2-V15 ultrasonic sensors to a John Deere 9860 STS combine. The sensors were mounted on a plate that was parallel to the ground and the axis of rotation was perpendicular to the grain. The sensor plate was mounted about 1.2 m from the auger chute. The plate was mounted on a motor shaft which in turn rotated the sensors at four revolutions per minute. The sensors were mounted at 45.5, 32.5, 19.5, and 6.5 degrees with respect to the motor shaft so each sensor could detect a different grain profile. The rotating sensors produced a data profile of four concentric imperfect circles as the sensor array moved over the grain profile. The surface of the grain going into the 640-bushel Brent gravity wagon was predicted at eight points set up as a grid system and compared to the actual depth. An average error of 72.4 mm was found while predicting grain depths at all eight points, with the largest magnitude difference being 251.5 mm (too few

points identified as grain at this position) and the smallest being 4.6 mm. Gaard recommends feature extraction technology to aide in identifying specific grain vessel attributes.



Figure 1.2. Sensor array used for grain depth detection. (Gaard 2012)

Elmer's Welding and Manufacturing was granted a patent describing a system for autonomously controlling a grain cart or directing a grain cart operator. Inputs such as PTO speed, grain height within the loading vessel, and the relative position of the grain cart in relation to the grain cart are used to autonomously guide the grain cart or direct the grain cart operator. Instructions to the operator can be provided via an in-cab screen (Banthia et al., 2021).

1.2.4 Knowledge Gap

Stereo vision cameras can provide a depth measurement for each pixel that is visible to both lenses. This distance measurement, with a bit of context such as the background, can be used to estimate volumes without the need for AI/ML training. This gap in the literature regarding the use of depth imaging devices used specifically for the measurement of volume of biological materials leaves many opportunities to improve operations. With that in mind, specific goals were developed for this dissertation.

1.3 Goals

The work of this dissertation was to:

1. Compare stereo vision to LiDAR with regard to vision quality in simulated agricultural conditions.
2. Develop and evaluate a volume measurement system that can be applied to biological materials.
3. Apply volume measurement techniques from (2.) and enhance with computer vision methods to measure changing volume of biological materials loading into relatively large transport or holding vessels.

2. ENVIRONMENTAL CHAMBER COMPARISON OF STEREO CAMERA AND LIDAR

2.1 Abstract

An environmental testing chamber with varying light and dust conditions was constructed to compare the depth capabilities of a Carnegie Robotics S21 stereo camera and a Velodyne VLP-16 Puck LiDAR in adverse conditions. Both systems were relatively unaffected by varying light conditions, but the LiDAR was able to image dust plumes as opposed to the stereo camera which was unable to see past the dust plumes but also could not image the dust plumes.

2.2 Background

2.2.1 Hardware

The Carnegie Robotics Multisense S21 is a 2 megapixel (other variations available) stereo camera with a 21 cm baseline. The camera can operate at a resolution of 1024 horizontal pixels by 544 verticals pixels with 128 disparities (number of pixels searched) in the depth image. The resolution can be adjusted, but at the expense of decreasing frames per second. At 1024 by 544 by 128 (settings used in this paper) the camera operates at 10 frames per second. The focal length of 4.8 mm (other variations available) gives a field of view of $115^\circ \times 68^\circ$. The camera is IP68 rated, making it well suited to physically withstand agricultural field harvest and similar conditions. The camera has a minimum range of 1.5 m, and a reported error of up to 4.8 mm at a range of 50 m. The camera can stream data via a ROS-based API.

The Velodyne Puck, known as the VLP-16 when originally purchased for the project, is a LiDAR sensor with a range of 100 m. The Puck delivers a 365° horizontal field of view and a 30° vertical field of view, with 15° above and below the LiDAR sensor coordinate origin. 16 lasers scan at a resolution of 2° vertically. Advertised accuracy is up to 3 cm (listed as “typical” with no other specifications). Reflectivity of objects can also be observed in addition to point distance. Calibration files are available from the manufacturer, but the user manual does not state that the units require calibration out of the box. At 600 rpm (settings used in this paper); the azimuth

resolution is 0.2°. The sensor carries an IP67 environmental rating. The LiDAR can also stream data via a ROS-based API. Published independent testing showed the accuracy of the VLP-16 to be accurate within the temperature range of 0 to 40°C and did not require significant warm-up time. Accuracy within 3 cm ranging error as specified by the manufacturer should be expected. (Glennie et al., 2020).

2.2.2 Dust Problem

Depth sensors utilizing stereo or laser technology have been known to be adversely affected by their environments. Cameras can suffer from image saturation due to sunlight (Yoneda et al., 2020). LiDAR performance can be adversely altered by external light (Wevolver, 2020). False positives in LiDAR can be due to rain, fog, and dust (Broggi, 2020) (Wevolver, 2020). A SICK LMS 291 LiDAR was mounted on a tractor to measure straw output of cut straw from a combine. It was found that “Dust, precipitation, or fog can reflect part of the energy hence creating echoes.” The detected swath path was easily distinguishable from the ground on the laser height map, however (Lenaerts et al., 2020).

Kise et al. (2009) conducted an “airborne obscurant test” to determine how sensitive depth perceptions were to dust. An enclosed chamber, 17 meters long, 3 meters wide and 2 meters high was used for testing. Sensors were placed at one end and the target, a 1.5 m tall by 17.5 cm wide white cylinder, was placed at the other end to be imaged. Peat moss, used as the dust, was manually agitated with a leaf blower. Sensor models were not provided but after dust was blown, it took the stereo camera 146 seconds to return to within 10% error depth measurements, 105 seconds for a laser sensor to do the same and another laser sensor did not return to 10% error within the time limits when the sensors were five meters from the target.

Ryde and Hillier (2016) designed a dust, rain, and mist chamber to test a SICK LMS 291-S05 and a Riegl LMSQ120 laser scanner. Simulated dust conditions were created by a fan inducting airflow across a vibrating table where the dust was piled. Dust was 1-5 kg of talcum powder. The chamber was room with a floor area of 30 m by 5 m and a height of 2 m. A retro-reflective target was placed at various distances from the target sensors and the data from the sensors was recorded. Dust was quantified using laser pointers directed at a white screen where the white screen was imaged by

cameras to quantify dust concentration. The laser scanners were found to provide accurate distance measurements up to a transmission coefficient of 92-93% per meter for relatively bright targets closer than 25 m.

Philips et al (2016) found LiDAR returns exhibited predictable behavior when testing in a dust chamber. The work was inspired by LiDAR imaging systems used for mining trucks loaded with earth materials. A dust chamber was created to test a SICK LMS511, which was mounted at one end of the chamber and a target wall was on the other end being imaged. The chamber was 5 m long and Arizona Coarse dust was used as the test dust. 10 centrifugal fans circulated dust in the system. Transmittance of the generated dust was also measured using a SICK T-50 Dusthunter. The unit consists of a sender-receiver at one end of the chamber and measures the light reflected from the target at the other end. Dust was able to exit the chamber during testing. The authors found that images of the target were unimpeded in sparse dust but in dense dust, the surface of the dust cloud was imaged. The transition between no impedance and imaging dust was determined by the density of the dust and reflectivity of the target. Finally, under certain conditions, no range measurement was returned. They found the LiDAR behavior predictable by the elastic LiDAR equation. Similar results were reported for the SICK LD-LRS3100, SICK LD-MRS, and Velodyne HDL-64E.

2.2.3 Software

Barth et al. (2014) explained the concept of Robot Operating System. “The ROS system provides support for interfacing with sensors and actuators, communication between software components, and also high-level modules for navigation and path planning.” Message passing is administered via ethernet and ROS also manages parallel execution between software modules, called nodes. Relocation of nodes to separate computers is possible to divide processing loads of hardware. ROS features event-driven communication, request-reply communication, and action communication. Event-driven communication allows nodes to either publish or subscribe topics. Request-reply is used via services and are used when information needs to be shared before a node can execute. Actions are used for supervision. (Barth et al., 2014). Although not a truly real-time framework, time-critical components can be developed separately and then interfaced with ROS (Barth et al., 2014). “Real-time software guarantees correct computation at the correct time.” (Kay, 2016) Linux

is not considered a real-time operating system, and since ROS operates on the system, the ROS system cannot be guaranteed real-time. However, sub-second accuracy of reactions or measurements are still possible within ROS.

An example of the flexibility of using ROS is the Thorvald II Robot, a modular robot system that allows the flexibility to customize robots for agricultural use (Grimstad and From, 2017). This robot utilized ROS for the basis of the robot software. Nodes are written to be independent of hardware, this way only a few parameters need to be changed when changing hardware on the robots. There is an active community of robot users for agriculture, with an active GitHub containing some ROS projects (Robot Agriculture, 2022).

RVIZ is a ROS visualization software package which is free and open source, working under the BSD 3-Clause "New" or "Revised" license (Huang, 2017). RVIZ allows point clouds, depth images, or other ROS topics to be visualized on a grid in a quick way using a graphical user interface. Some point clouds in this work are generated using RVIZ due to its ease of use. However, RVIZ does not allow the user to post process point cloud data within a coding environment.

OpenCV is an open-source computer vision library that can be used with C++, Java, and Python. The library is released under the BSD 3-clause license. Hundreds of computer vision algorithms are supported (OpenCV, 2022). Moreover, OpenCV can interface with ROS topics and images via the Rospy and CvBridge Python client libraries.

Matplotlib is a popular library that creates “static, animated, and interactive visualizations in Python.” (Matplotlib, 2022). Matplotlib works very well for creating scatter plots, and a good way to represent a 3D point cloud is to use a scatter plot. *matplotlib.animation.FuncAnimation()* works by repeatedly calling a function, whichever function that may be that outputs the data points that need updating. Integrating Rospy and Matplotlib *FuncAnimation()* creates a way to stream point clouds within Python, after the code has been structured.

2.2.4 Gap

Based on the literature available and our own observances with dust using the Multisense S21 and the Velodyne Puck, a series of testing was necessary. There was no previous work with corn dust, hence this work.

2.3 Objectives

1. Develop a method to stream point clouds into the Python environment while streaming the data from ROS.
2. Compare vehicular-mounted sensor data between VLP-16 and Multisense S21 from corn harvest conditions.
3. Construct an environment to simulate corn dust plume environments encountered during corn harvest.
4. Compare data quality of the VLP-16 to the Multisense S21 when operating in the dust chamber.

2.4 Streaming Point Clouds in Python

2.4.1 Methods

During the investigation of the LiDAR system, which does not stream images, a way to stream point clouds for post-processing and visualization was investigated in accordance with objective 1. Because the scatter plot is constantly being updated within ROS, a way of constantly updating the scatter plot was needed. A function called *matplotlib.animation.FuncAnimation()* within Python seemed like the best choice due to the existing Python code up to that point already heavily containing Matplotlib functions.

A major contribution of this work was the development of the ability to stream a point cloud from ROS into Python and animate it using the tools just described within the Python coding environment (Appendix 2). The missing link was the rearrangement of the Python code that had been used to stream ROS topics into an object-oriented structure. ROS topics require the use of callback function to stream into Python from ROS. The outputs of the callback function need to

be within an object-oriented structure before going to the *matplotlib.animation.FuncAnimation()* to be updated continuously. The data pipeline for the point clouds generated are shown in Figure 2.1. Figure 2.2 shows the data post-processing pipeline.

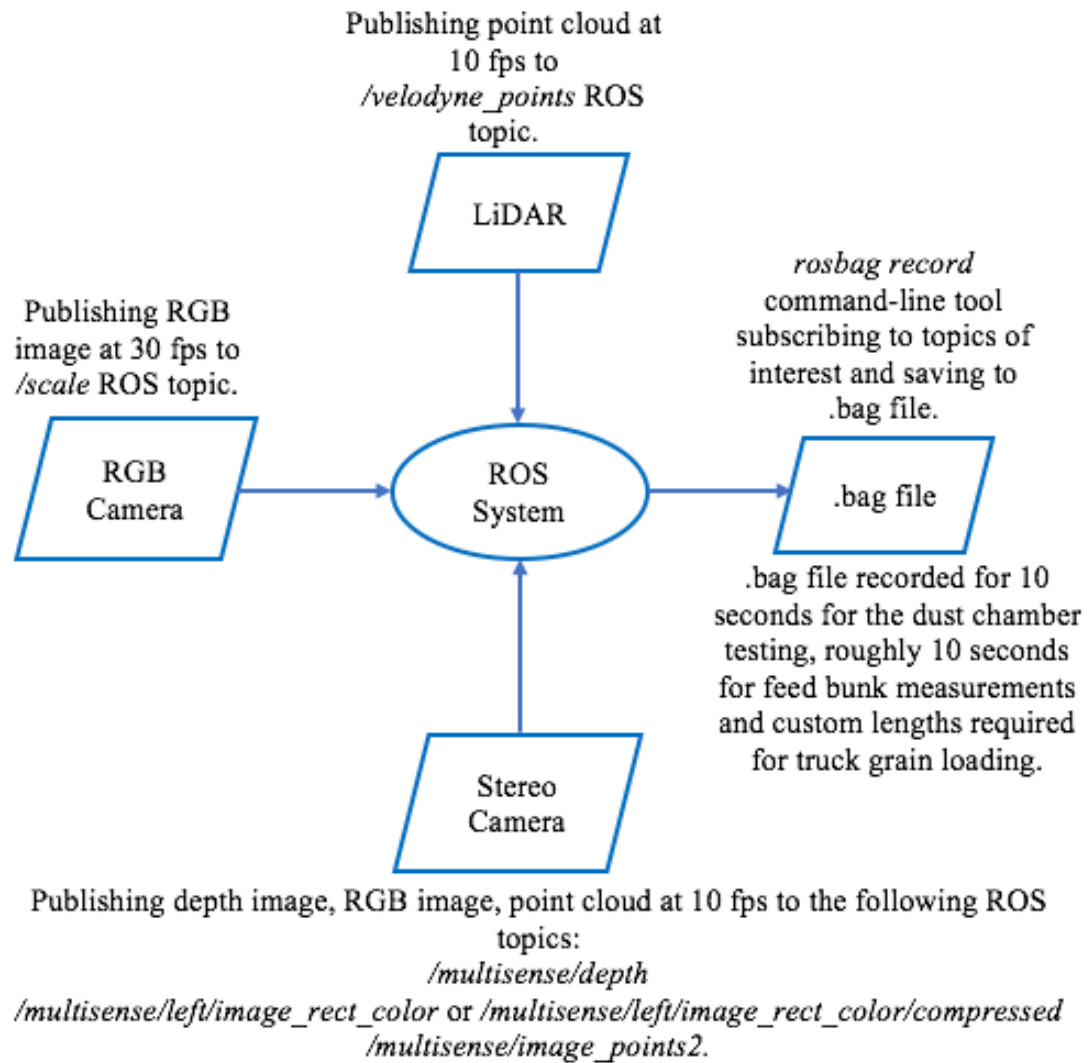


Figure 2.1. Data pipeline for data collection used in chapters 2, 3, and 4.

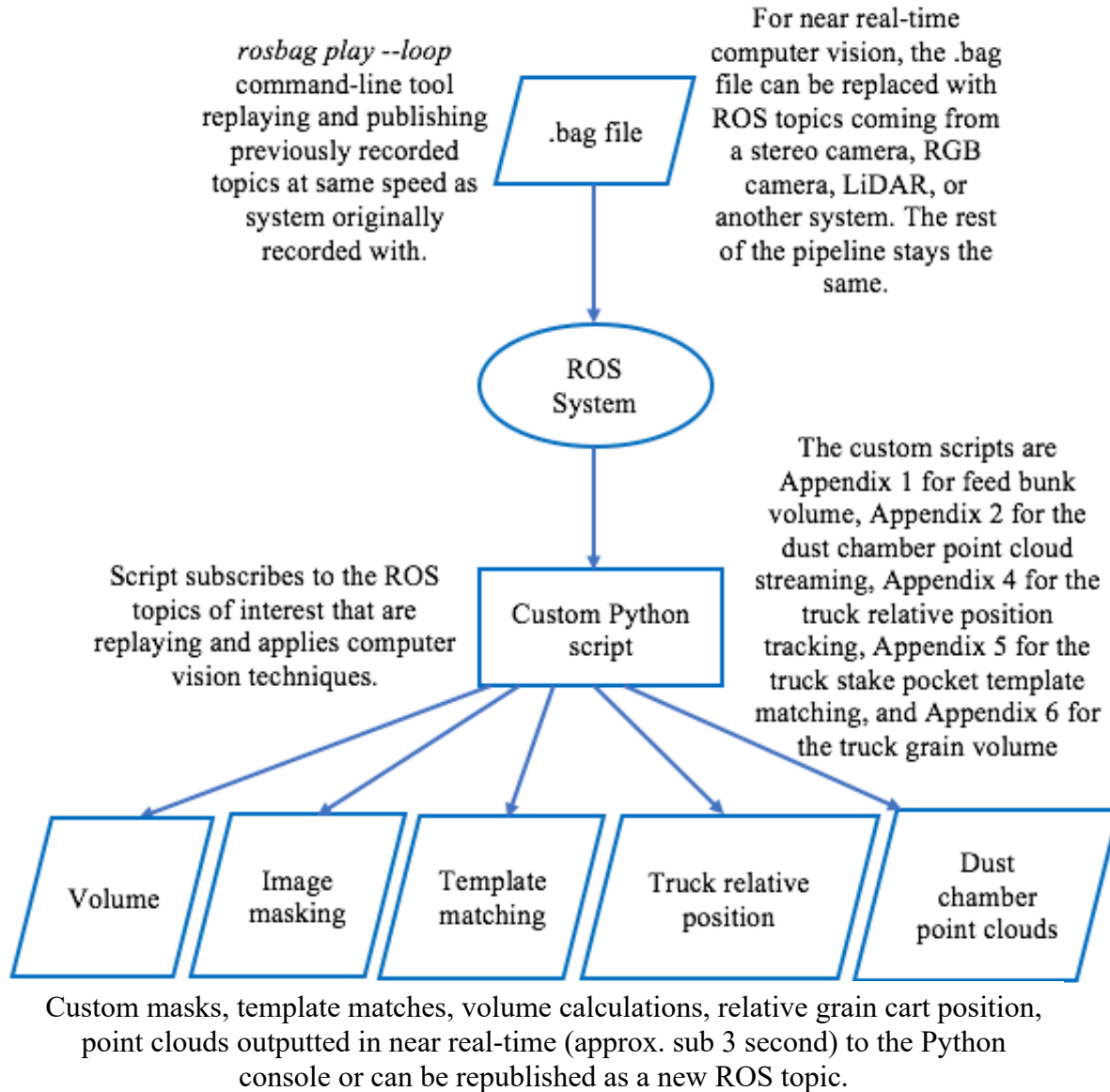


Figure 2.2. Data post-processing pipeline for image processing and volume calculations.

2.4.2 Results and Discussion

Published work nor web forums suggest any previous integration of ROS and Python for streaming a point cloud in near real time. Streaming a point cloud can be done, but software such as RVIZ is required. The ability to operate ROS, Python, and point cloud streaming together opens the door to open-source coding and development for projects requiring point clouds for computer vision. Point clouds were successfully streamed within Python from a ROS Topic. Units for the point

cloud were in meters. Point cloud images will be used extensively in this chapter in section 2.7 from the developed Python method. Now researchers or programmers can apply Python code and libraries to manipulate and extract information from point clouds, such as converting them to depth images, applying them to volume calculations, trimming excess point cloud points, or other post processing or real time data extraction strategies.

2.5 Camera Testing in Grain Transport Conditions

A comparison was conducted between the VLP-16 LiDAR the Multisense S21 stereo camera from Carnegie Robotics in grain transport conditions in accordance with objective 2. The comparison consisted of visually comparing similar grain-flow situations for each instrument.

2.5.1 Methods

The truck test was conducted on separate days at the Purdue University ADM building parking lot located in West Lafayette, Indiana. A J&M 750 grain cart was used for testing; it was attached to a New Holland T8 tractor. Each imaging system was tested in a similar manner: Similar mounting points were used for the depth instruments; the same grain cart driver; similar grain cart unloading flow rate were used. The tractor was operated at 1000 power take-off (PTO) rpm and the auger door within the grain cart was opened all the way to achieve maximum grain flow. Images to compare the LiDAR and stereo camera were collected from the mounted position of the sensors explained below. ROS was used to collect .bag files of point cloud and depth image streams for replaying later using the visualization software discussed earlier in this chapter. Still (paused) images of the point clouds and depth images were chosen during peak grain flow for comparison of sensor capabilities. OpenCV was used to visualize and capture the images used for the stereo camera, and RVIZ was used for the LiDAR imagery.

LiDAR

The Velodyne needed to be mounted in a quick, secure, but temporary fashion that would not be damaging to the grain cart or the LiDAR. Figure 2.3 shows the mounted LiDAR system.

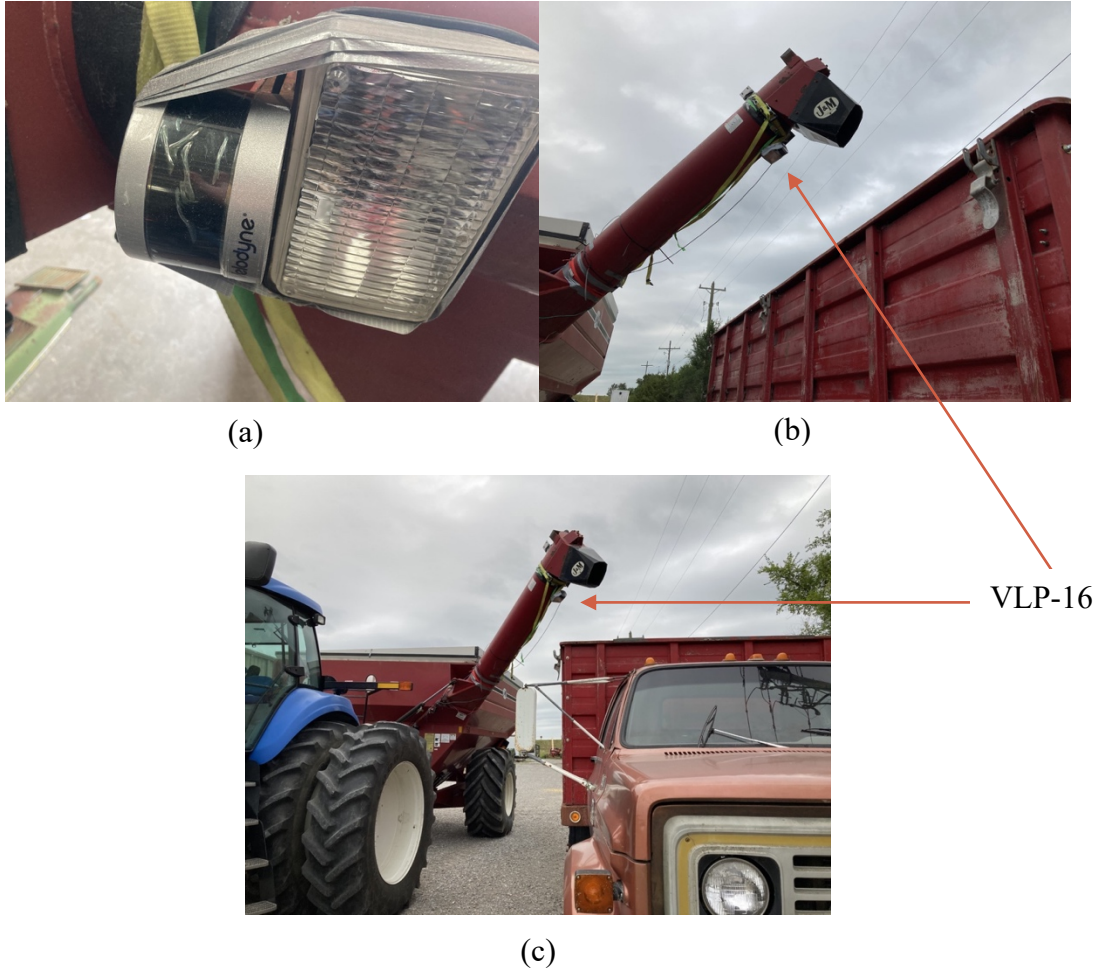


Figure 2.3. (a) VLP-16 LiDAR sensor mounted on spotlight, (b), (c) LiDAR overlooking truck bed.

Stereo Vision

The stereo camera mount was more complicated. A backing plate was designed with a shaft attached to it, and the shaft mounted to a PanaVise rotating clamp. The PanaVise was, in turn, mounted to a flat steel piece with slots cut into it for adjustable c-clamp positions. The c-clamps were slid up and down as needed for adjustment. The flat steel piece with the c-clamps could then be mounted with a ratchet strap around the auger tube of the grain cart. The ratchet strap was wrapped around the circumference of the auger tube and through the c-clamps of the flat steel piece ratcheted tightly around auger tube like a belt. The field of view of the camera was adjusted so that the x direction of the camera frames would be parallel with the length of the truck and also

the direction of travel of the grain cart. Figure 2.4 shows the mounting of the grain cart camera system.

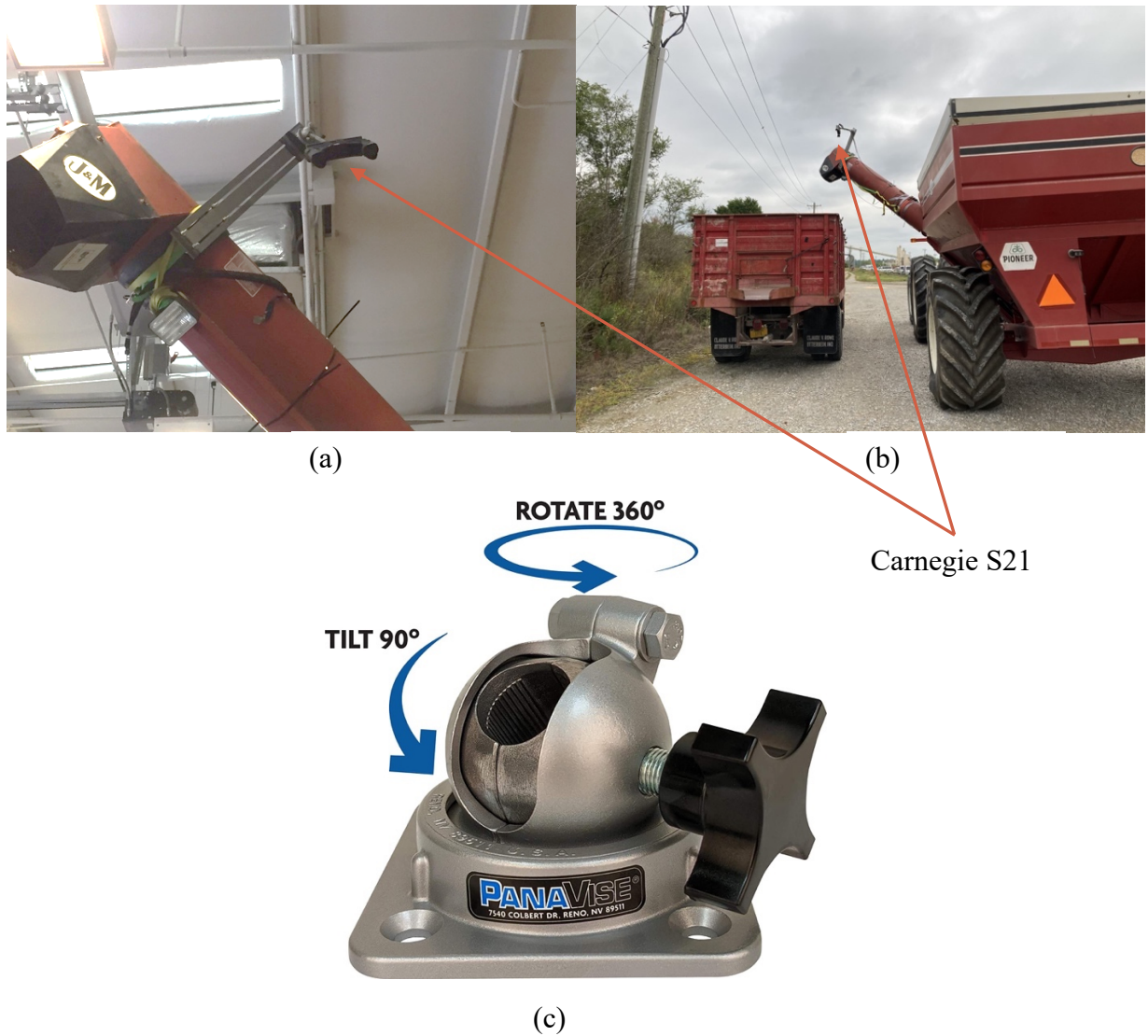


Figure 2.4. (a) Close-up view of mounted stereo camera on the grain cart, (b) Stereo camera overlooking truck, (c) PanaVise mounting system (PanaVise, 2022).

2.5.2 Results and Discussion

LiDAR

Visual results for the LiDAR were not encouraging. Figures 2.5 through 2.8 show the progression of driving to the grain truck, unloading start, unloading midpoint, and near finished unloading with

just a fine trickle of grain still coming from the auger. The first and last images of the sequence show the highest capability of the LiDAR when no dust is interfering with the scan lines. However, the dust clearly interferes with the point clouds during peak unloading phases of the grain cart. The dust effects are so severe that scan lines cannot even be seen in the truck bed during one part of the unloading process, although some scan lines return after the grain pile has risen after a few moments. Better mounting may have afforded more scan lines in the truck during unloading grain, but it is evident that the LiDAR system is heavily affected by the dust that unloading corn generates , and the LiDAR system cannot be a single tool solution for computer guidance when it comes to unloading corn.

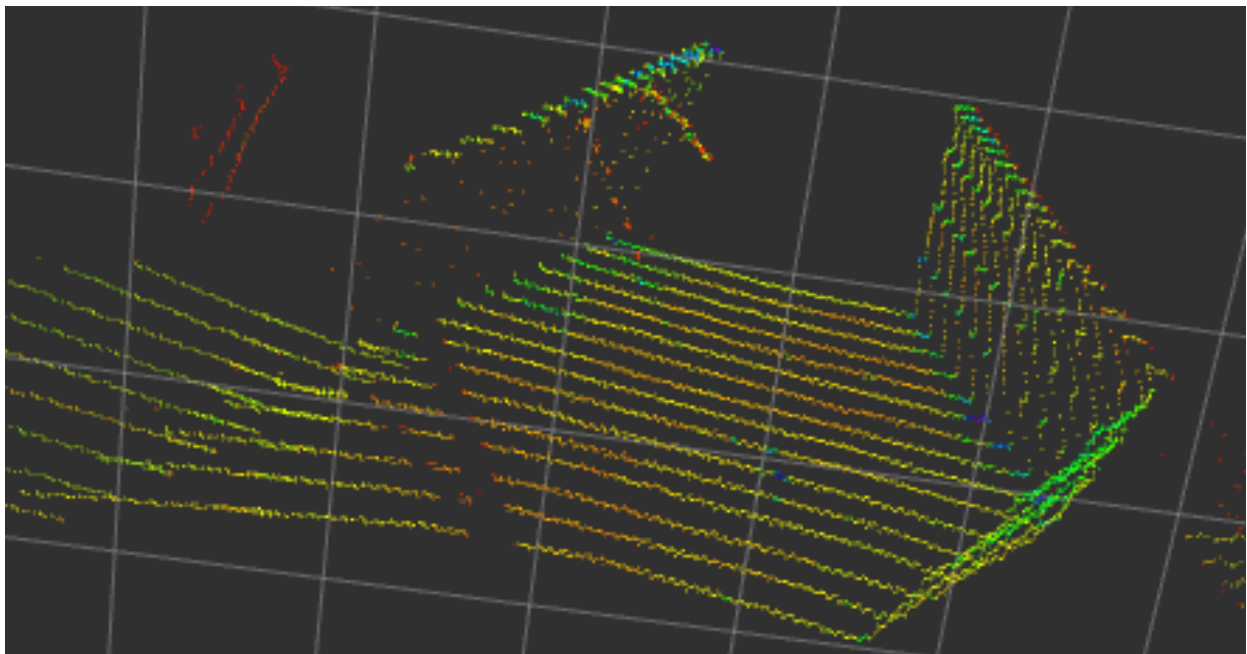


Figure 2.5. RVIZ image of the LiDAR scan of the empty truck.

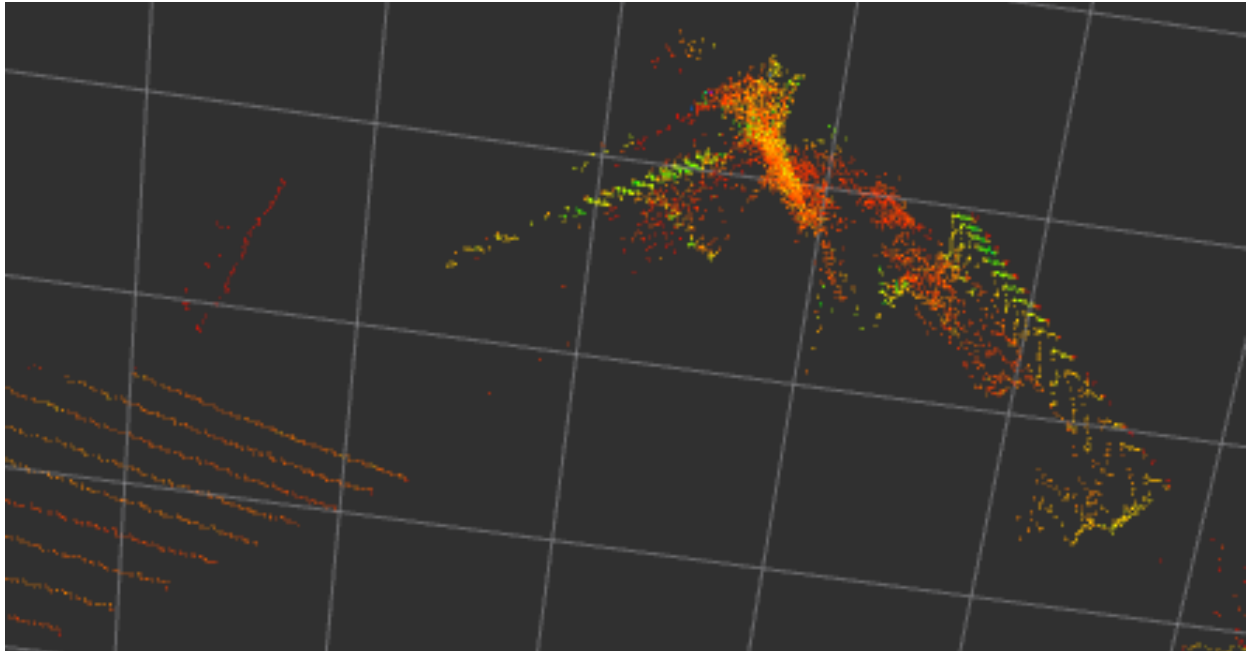


Figure 2.6. RVIZ image of the LiDAR scan of the truck during filling.

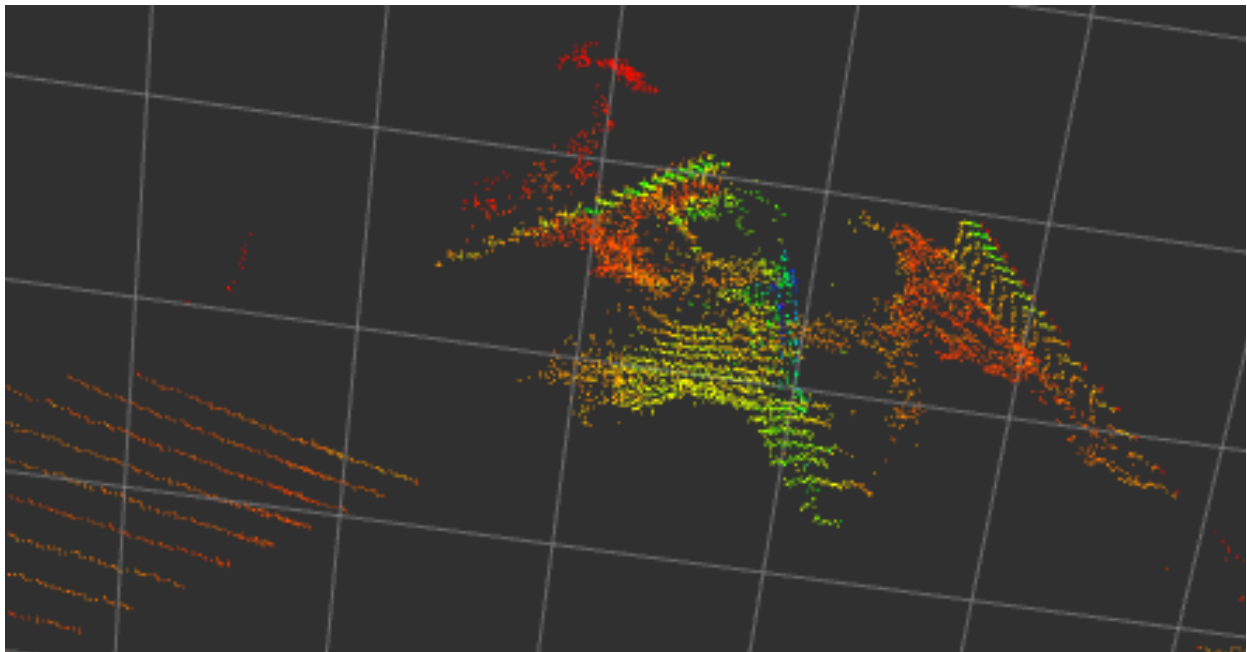


Figure 2.7. RVIZ image of the LiDAR scan of the truck during filling, with grain surface progressed near completion.

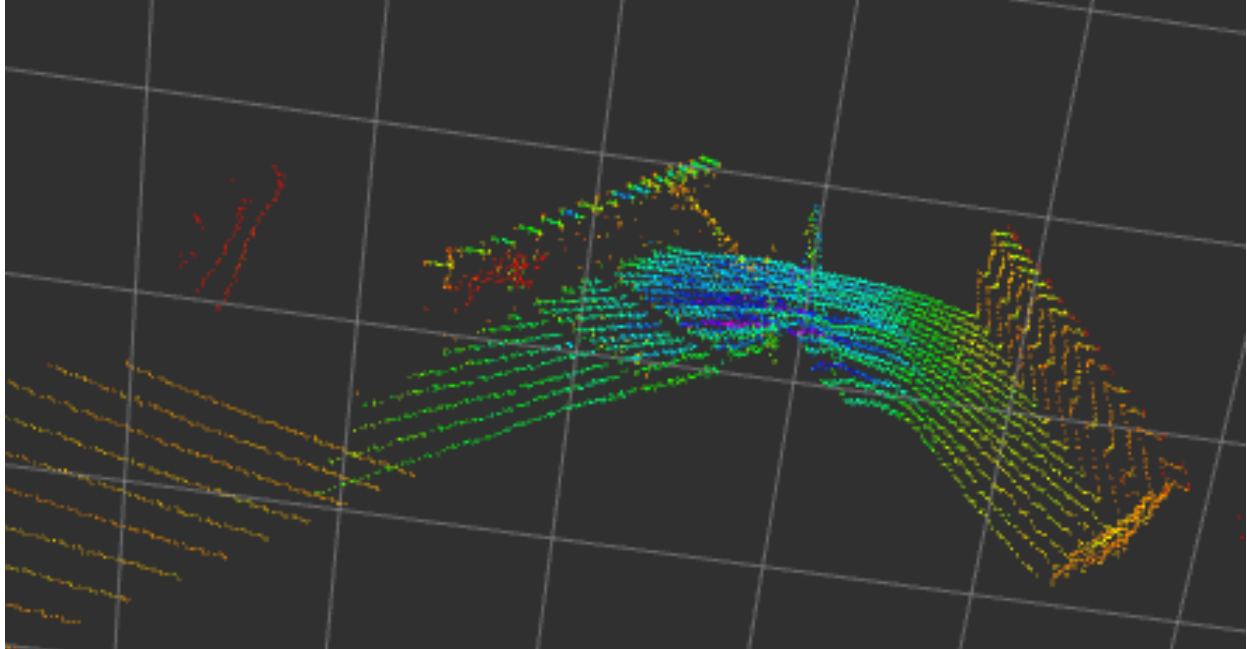


Figure 2.8. RVIZ image of the LiDAR scan of the filled truck.

Stereo Vision

Visual results for the Multisense show adverse effects of dust during unloading phases, but in a different way. Depth images were used instead of point clouds. Dust in the field of view of the stereo camera seems to produce non-calculated values, or “NaN” values, for those pixels. Figure 2.9 shows the color image and depth images from approximately the same unloading moment. One depth image has the non-returnable pixels colored blue, and the other depth image has the pixels gray-scaled according to depth value. Figure 2.9 (c) shows NaN pixels colored in blue. Some NaN pixels are also caused by occlusion. Moreover, some pixels can still be seen in the truck unloading zone of the depth image, though admittedly not many. However, using this instrument, computer guidance systems would have the color image frames to use if the depth image frames have too many NaN values. Some frames were near 100% NaN coverage for approximately three seconds. Those frames were comparatively rare compared to frames with at least some usable pixels. For volume calculations, NaN returns are better than the returns the LiDAR provides, which are actual returns coming off of the dust cloud. NaN returns still allow for estimating of the grain volume using the averages of the pixels that do have usable returns. Calculating volume

using point cloud points returning from the dust cloud will lead to poor volume calculations. In other words, some missing data is better than incorrect data.

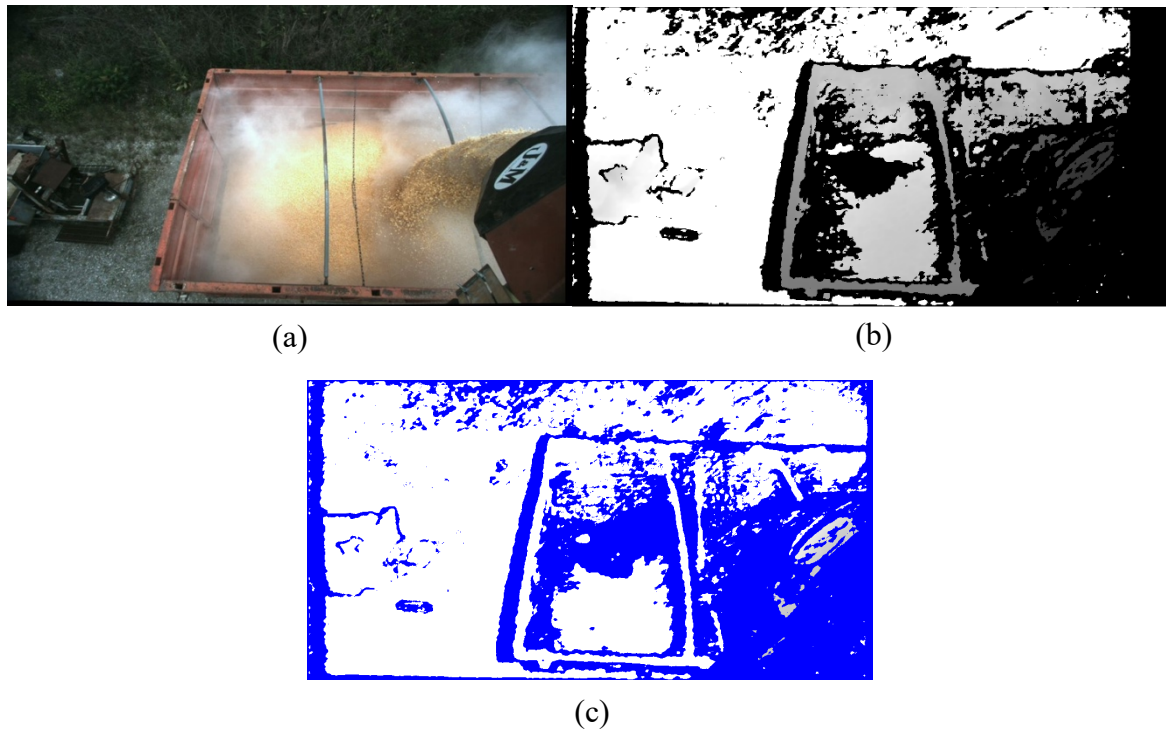


Figure 2.9. Imagery during truck filling. (a) color image, (b) grayscale depth, and (c) depth image with blue pixels representing "NaN" values caused either by dust or occlusion.

2.6 Dust Box Design

A dust box was constructed to test how the LiDAR and stereo camera point cloud data changes in different dust and lighting conditions to complete objective 3. Two levels of light and two levels of dust were tested. Color image streams were used to visually quantify the dust. The dust developed during the high-pressure dust box testing was very similar to the dust observed during grain cart unloading. Reflectivity of the target was also not considered, for similar reasons. The surface material of the target, corn grain, is the same material that is loaded into grain carts. This section of the paper is primarily concerned with which sensor can best image corn, so the reflectivity of the target was deemed irrelevant.

A 4.3 m (14 ft) long square wooden tube was constructed to simulate dusty corn unloading conditions in different lighting environments (Figure 2.10). This distance represents the

approximate length of the distance the stereo camera was mounted at from the floor of the grain cart. The box was 0.6 m (2 ft) wide. The outer shell was constructed of plywood, and the frame was constructed of lumber planks. Two 13 cm (5 in) holes, 45 cm (18 in) apart and 11 cm (4.5 in) from the bottom floor edge to the center of the hole, were drilled into the side of the plywood. Dust clouds could be created using an air moving instrument, such as a hair dryer or an air nozzle attached to an air compressor. Short 13 cm (5 in) diameter PVC joints were placed into the holes so a person could more comfortably use their hands in the box (Figure 2.11). Box ends of the box were sealed with plywood, with the far end away from the depth measurement instruments sealed with a plywood board that was covered with black Gorilla brand tape.

Adjustable lighting was added to the walls of the dust chamber. Barrina Plant Grow Lights were used with three each on each wall. They were mounted lengthwise halfway up the wall (Figure 2.10). These lights contained 1152 LEDs in total using 252 Watts. These lights were advertised as full spectrum lights. The instrument far end of the dust chamber had more lights mounted on the wall to illuminate potential targets even further. Four “Relassy 15000 Lux Sunlike Full Spectrum Grow Lamps” were mounted on each wall, each lamp containing two bulbs for a total of eight bulbs on each side of the dust chamber.

To simulate dusty conditions experienced during corn loading into trucks, corn meal was added to the floor of the dust chamber. It was piled in the middle of the chamber so it could easily be blown using hair dryers and air nozzles. The dust piles were usually no higher than approximately four inches off the floor to avoid entering too much of the field of view of the instruments. 1700 g (600 oz) of Bob's Red Mill Organic Medium Grind cornmeal was used for all testing.

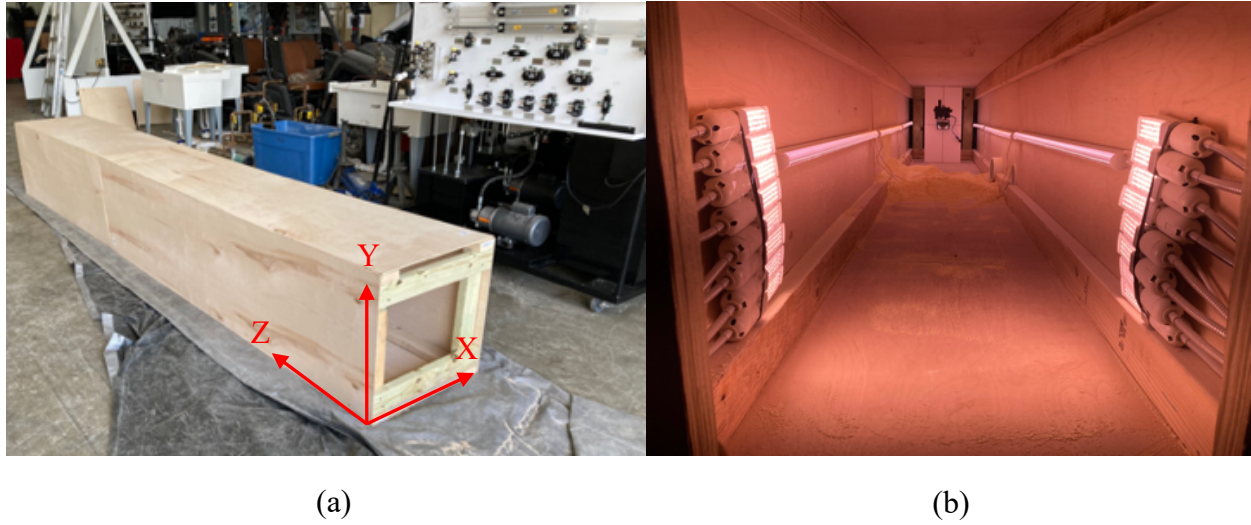


Figure 2.10. (a) Constructed wood dust chamber and (b) lighting with the mounted LiDAR shown at the far end.



Figure 2.11. (a) View inside dust chamber and (b) arm holes for access.

A simulated “wedding cake” was constructed using Styrofoam blocks. An 30 cm (11.9 in) by 30 cm by 10 cm (4.0 in) block was used as the base, and a 25 cm (9.8 in) by 25 cm by 10 cm block was used in the middle, and the top was a 20 cm (7.9 in) by 20 cm by 10 cm block. All blocks were FloraCraft brand. The blocks were aligned by the lengthwise center when stacked on top of the other, but the back faces of the blocks were parallel forming a flat surface of the back of the wedding cake. The cake surface was covered with corn grain to simulate the material imaged by

the devices when measuring corn volume into a loading truck. Figure 2.12 shows the wedding cake target.

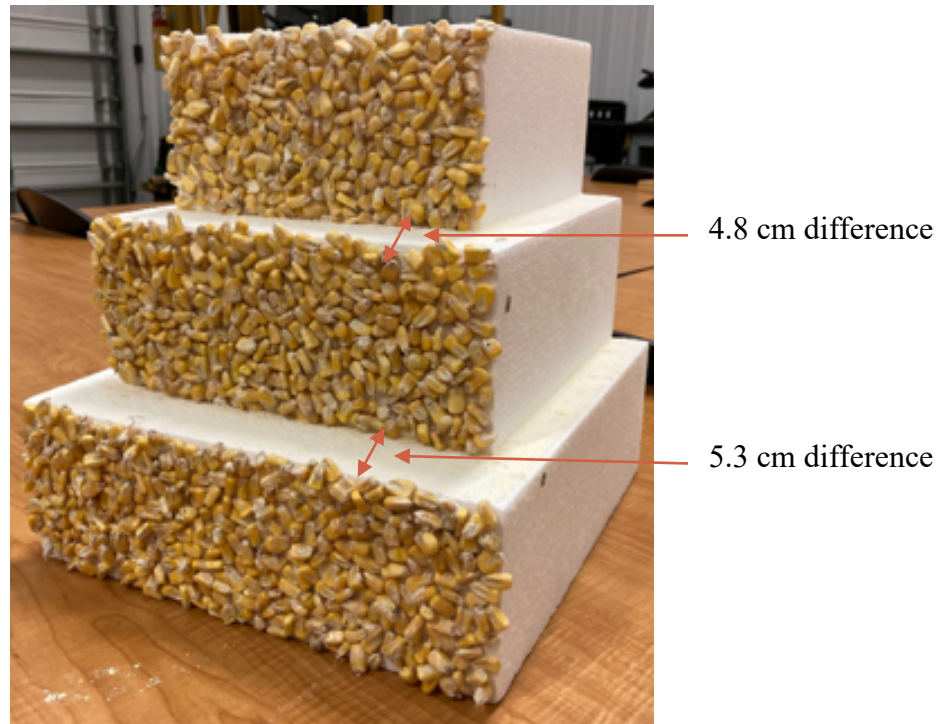


Figure 2.12. “Wedding cake” sensor target.

2.7 Camera Testing in Dust Box

2.7.1 Methods

The sensors were mounted in the centroid of 61 cm (2 ft) by 30.5 cm (1 ft). Styrofoam boards. Holes were cut into the Styrofoam to firmly mount the sensors. The color camera was also mounted into the Styrofoam. A wooden backing plate was added to the stereo camera Styrofoam to add extra support due to the weight of the stereo camera. The extra RGB camera was always used for the LiDAR but not for the stereo camera, because it also yields a color image stream (Figure 2.13).

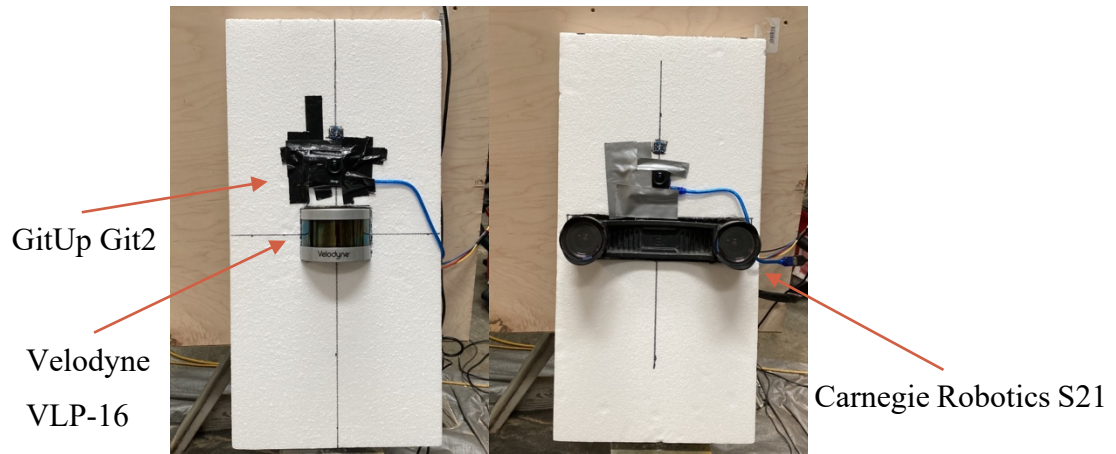


Figure 2.13. LiDAR, RGB, and stereo cameras mounted for dust chamber testing.

The Velodyne settings are shown in Figure 2.14, with the exception of the trials testing the difference between strongest, last, and dual return modes. The LiDAR consists of 16 laser rows that rotate 600 revolutions per minute. The lasers collect 300000 points per second. The rows of lasers create scans 2 degrees apart. The maximum range of the LiDAR is 100 m with no indication of minimum range in the user manual.

Velodyne® LiDAR

Sensor Model: VLP-16 S/N: AH23026785 MAC: 60-76-88-10-68-a1 Factory MAC: 60-76-88-10-68-a1

VLP-16 USER INTERFACE Configuration System Info Diagnostics

Return Type: Strongest

Motor RPM: 600 Set

FOV Start: 0 End: 359 Set

PPS Qualifier

Require GPS Receiver Valid: ☐ On ☒ Off

Require PPS Lock: ☐ On ☒ Off

Delay: 5 Set

GPS Qualifier

Require GPS Receiver Valid: ☐ On ☒ Off

GPS Position: ☐ PPS: Absent

Motor State: ☒ On RPM: 605 Lock: ☐ Off Phase: 239.77

Laser State: ☒ On

Phase Lock ☐ On ☒ Off Offset: 0 Set

Host (Destination)

IP Address: 255.255.255.255 Data Port: 2368 Telemetry Port: 8308 Set

Network (Sensor)

DHCP: ☐ On ☒ Off

IP Address: 192.168.1.201 Mask: 255.255.255.0 Gateway: 192.168.1.1

MAC Address: 60-76-88-10-68-a1 Set

Figure 2.14. Velodyne LiDAR settings used during testing.

The Multisense S21 stereo camera operates at a resolution of 1024 x 544 pixels and 10 frames per second. Figure 2.15 shows the settings of the camera that were used during camera testing. The camera searches 128 pixels to find matches in each lens (128 disparities). The error can be found according to the equation from Carnegie Robotics:

$$e = (D^2 \cdot p) / (b \cdot f + D \cdot p) \quad [2.1]$$

where:

e = error, m

D = distance, m

p = pixel error (.5 per Carnegie Robotics)

b = baseline, 0.21 m

f = focal length, 4.8 mm (is converted to pixels at full horizontal resolution, approximately 931.981 pixels).

Error measuring a target 5.0 meters from the camera would result in a maximum error of 0.063 m (1.26%).

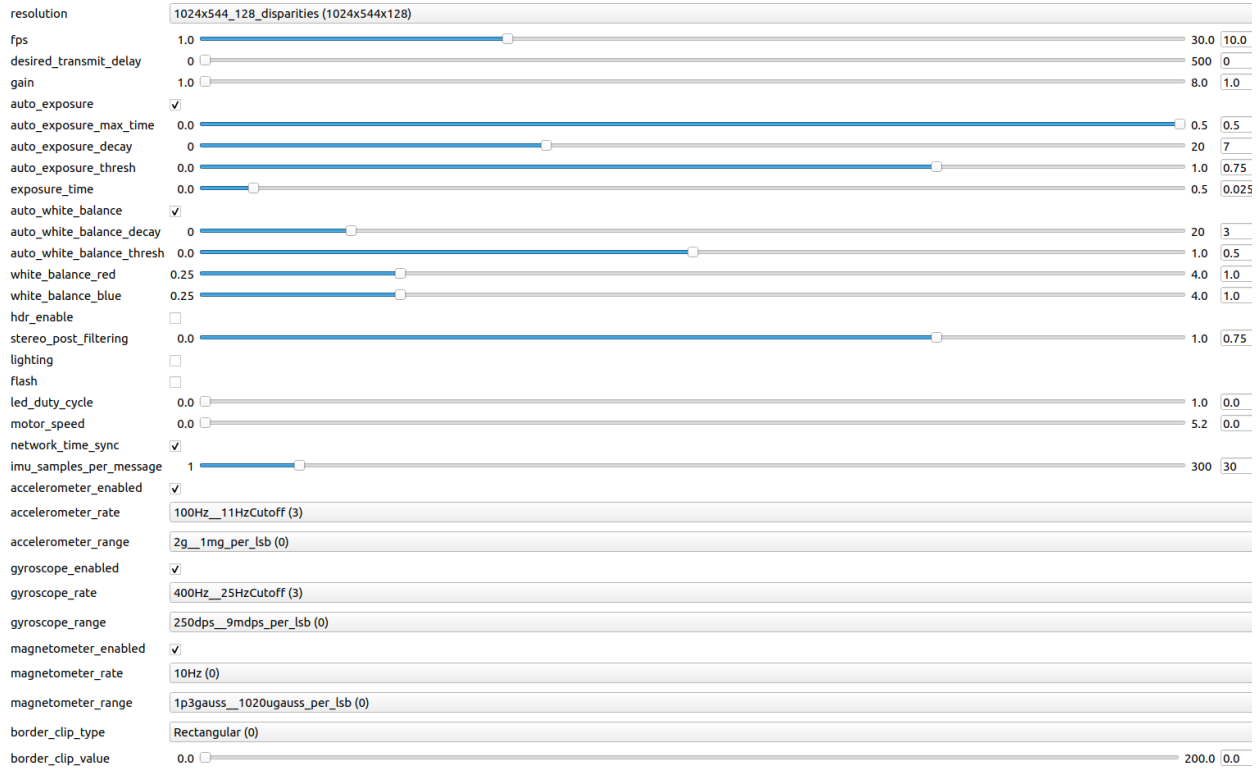


Figure 2.15. Stereo camera configuration page brought up using the *roslaunch rqt_reconfigure* command in terminal.

All data was collected via ROS into .bag files. Images and scatter plots were generated using Python, OpenCV, and Matplotlib (Appendices 2 and 7). Bag files were replayed on a repeating loop and then paused to gather the point clouds and color images.

Nozzles and hair dryers were used to circulate dust in the dust chamber. Hair dryers were used because two were on hand and ready to go, but also allow a controllable flow rate of air that appeared to generate a sizeable dust cloud to the human eye when blowing the corn meal. Revlon RV473CP1 1875-Watt hair dryers (Figure 16) were used for the low dust conditions. The hair dryers were operated with the “high” and “cool” settings selected. The “cold” and “turbo” buttons were taped down to flow rate as high and cool as possible. To prevent the hair dryers from clogging, each hair dryer used was fitted with Fram 11048 automotive air filters on the air inlet of the hair dryers. A generic looking (no brand was listed on the item) air nozzle was used for the high dust conditions and also the return mode comparisons. The air nozzle had a 1.2 cm (0.5 in) round outlet

and the air compressor was set at 830 kPa (120 psi), which was maintained while the nozzle was operated. The air nozzle is pictured in Figure 2.14.

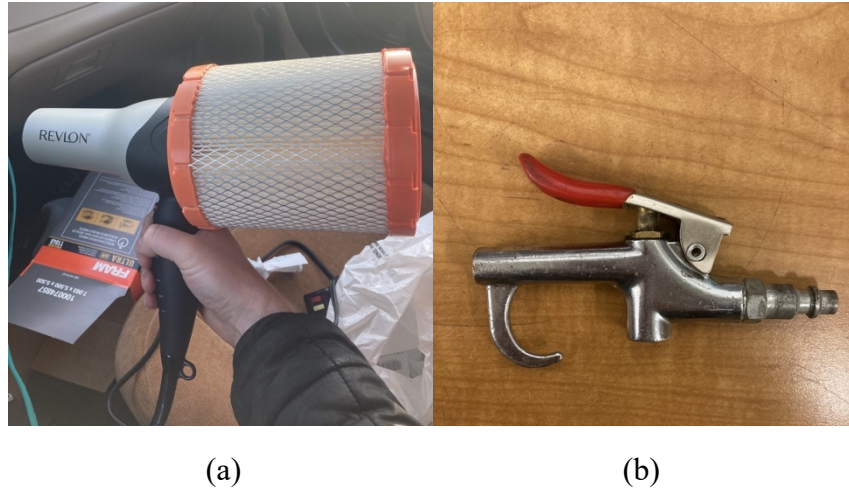


Figure 2.16. (a) Hair dryer and (b) air nozzle used for dust chamber testing.

The Velodyne VLP-16 includes three laser return modes to choose from: strongest, dual, and last. With the exception of the return mode specific testing, all testing of the LiDAR was conducted in the strongest return mode. To determine if there is a noticeable difference between the modes, the dust chamber was used to form dust clouds while the LiDAR recorded in all three modes in separate sessions. No target was placed in the back of the chamber; it was just a dull black Gorilla tape wall. One wall of lights (right side wall) was on. The same compressed air nozzle (not the hair dryers) was used to form dust plumes. Figure 2.17 shows the physical data collection procedure used for the dust chamber.

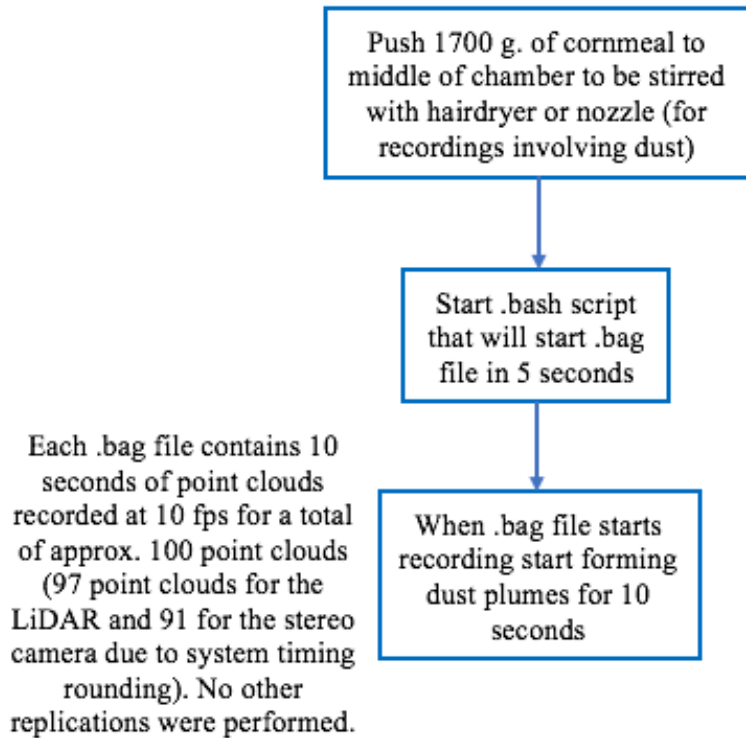


Figure 2.17. Point cloud collection process for dust chamber.

2.7.2 Results

Comparisons shown below show the instruments operating in full light, half-light, no dust, minimum dust, and maximum dust conditions. The view of the scatterplots is situated in an attempt to give a broad view of the contents of the dust chamber. The axes for the LiDAR scatterplots are the y axis showing the height of the dust chamber, the x axis showing the width of the box, and the z axis on the right of the image showing the length of the dust chamber (see Figure 2.10a). The RGB images were generated using a separate camera for the LiDAR and stereo images with the exception of the high-pressure testing, where the built-in color image stream for the stereo camera was used. The separate RGB camera was a GitUp Git2 1080p action camera. Color images and point cloud images are able to be streamed together but grabbing images of the same moment in time is more difficult. Point clouds and color images shown in this chapter are within approximately a second or less but cannot be guaranteed to be at the exact moment in time. However, they do represent the general trend of the environmental behavior of the dust chamber during the testing session.

No Dust Comparison

Figures 2.18 and 2.19 show the LiDAR images collected in no dust conditions in full and half-light. Figures 2.20 and 2.21 show the same images for the stereo camera testing. Point clouds and color images were collected successfully. These images can serve as a baseline for comparisons.

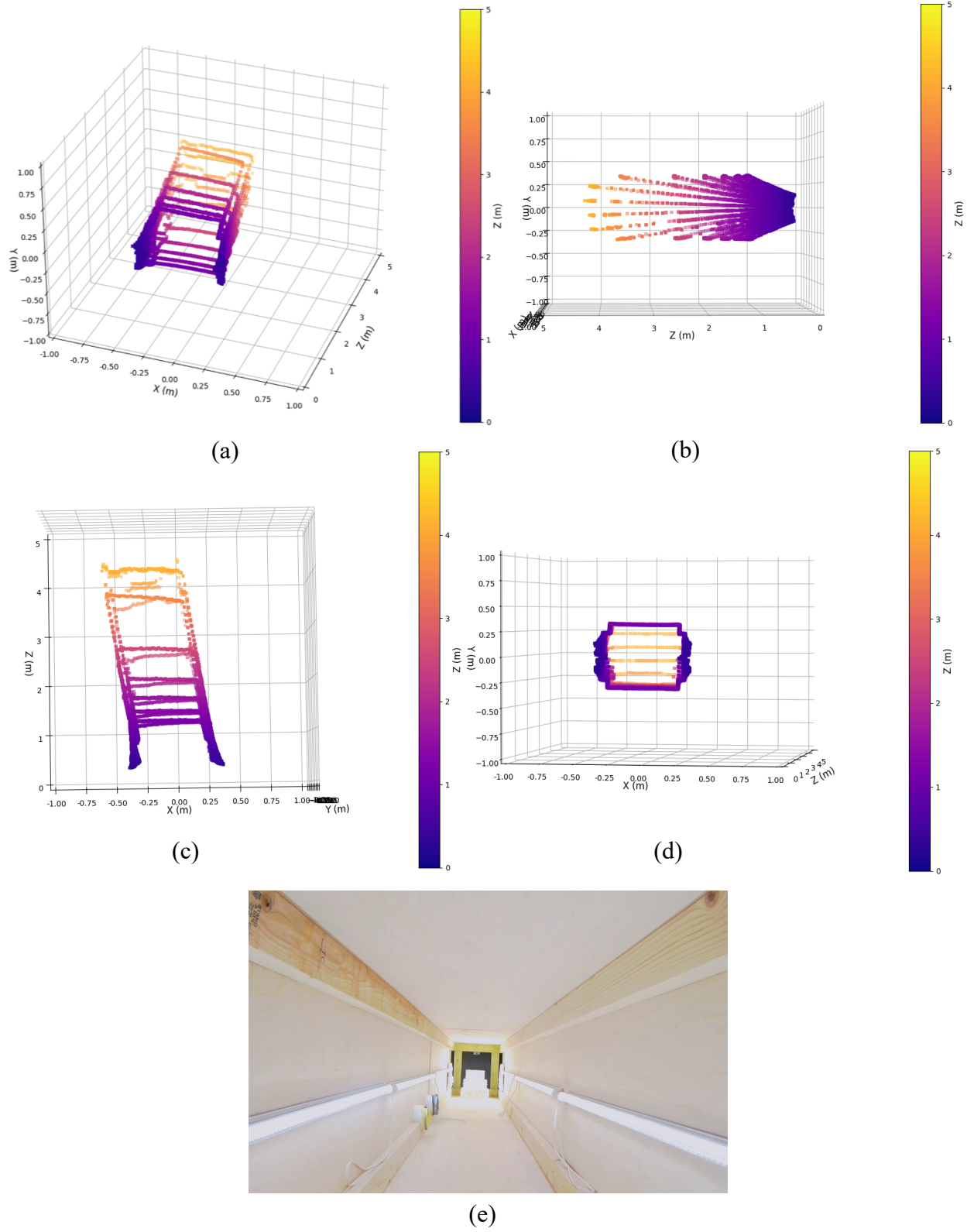
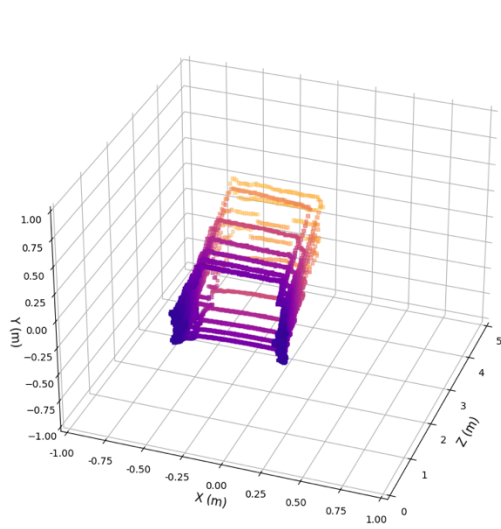
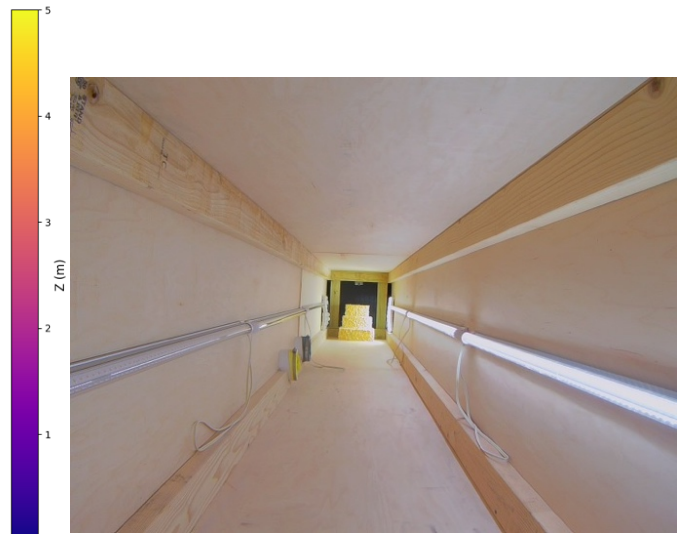


Figure 2.18. LiDAR point clouds and color image in full light conditions with no dust. Point clouds: (a) isometric, (b) side, (c) front, (d) top. (e) Color image.



(a)



(b)

Figure 2.19. (a) LiDAR point cloud and (b) color image of half-light conditions with no dust.

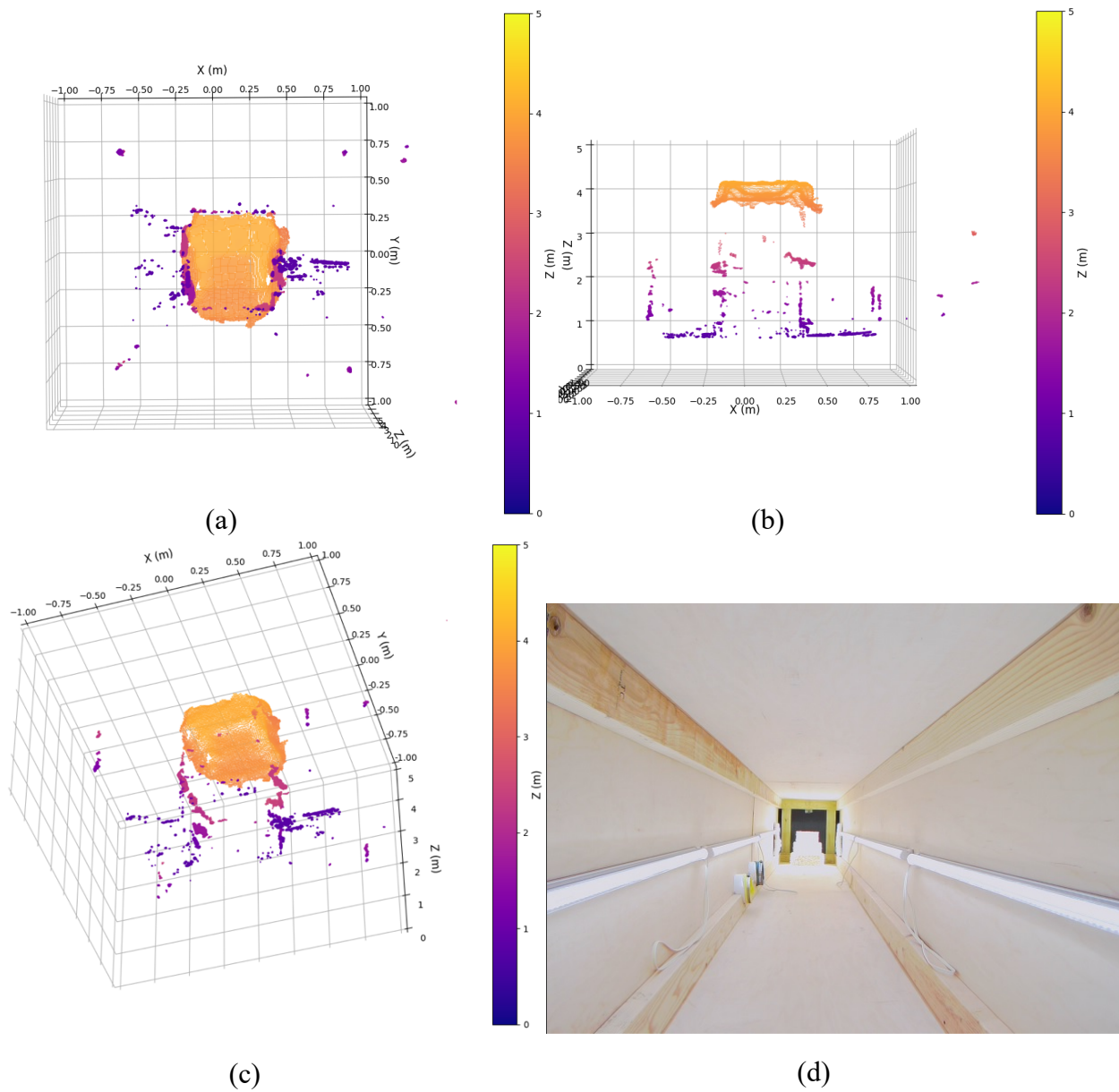


Figure 2.20. Stereo camera point clouds and color image in full light conditions with no dust. Point clouds: (a) front, (b) top, (c) isometric. (d) Color image.

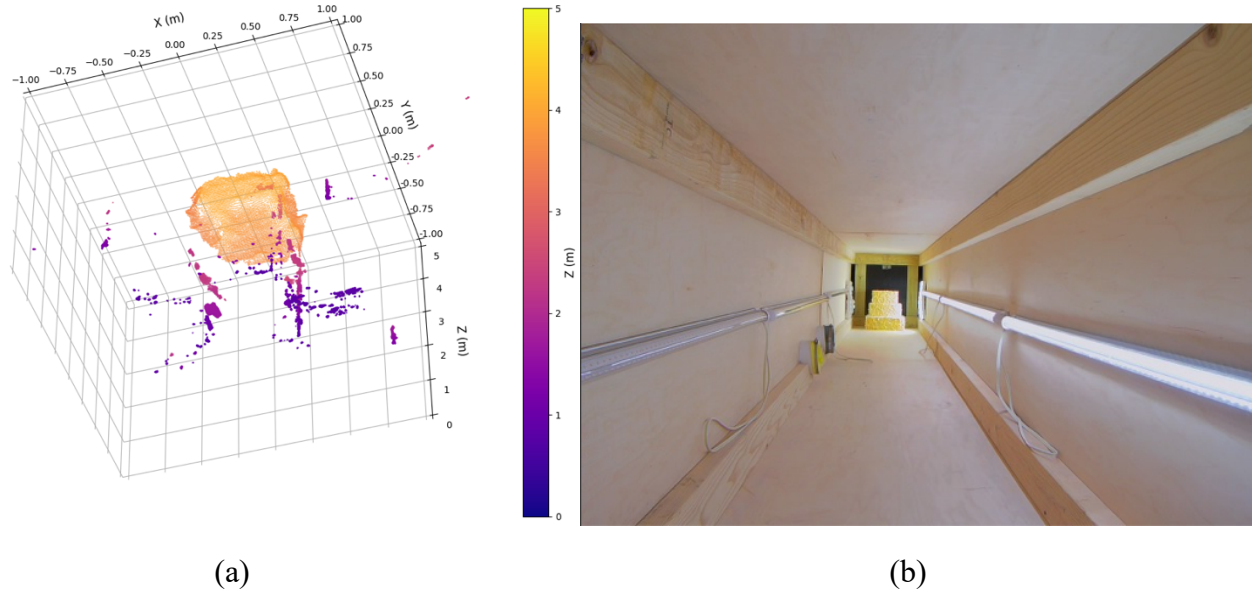
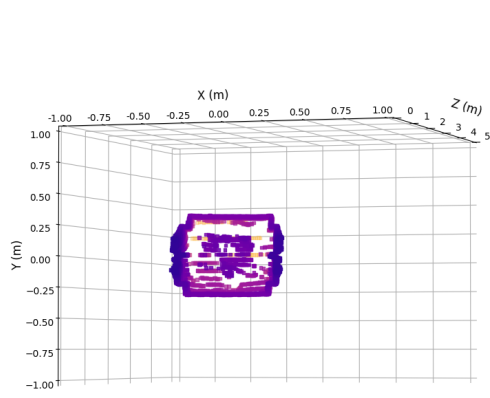


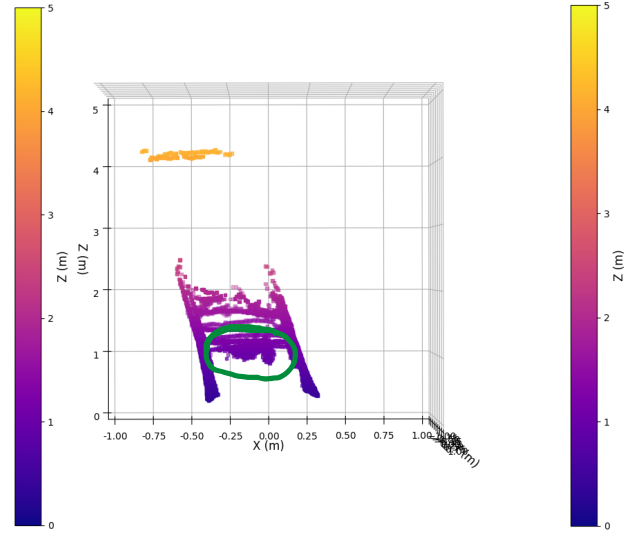
Figure 2.21. (a) Stereo camera point cloud and (b) color image of half-light conditions with no dust.

Low Dust Comparison

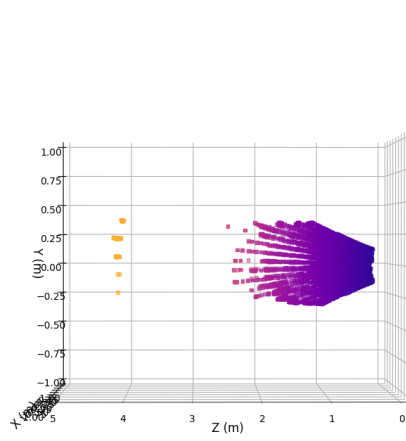
For the low pressure (hair dryer produced dust) environmental testing, a no light condition was tested for the LiDAR only and not for the stereo camera because the stereo camera would not have any point returns in no light conditions. Full-light and half-light conditions were evaluated for both systems. Figure 2.22, 2.23, and 2.24 show the results for the LiDAR and 2.25 and 2.26 show the results for the stereo camera.



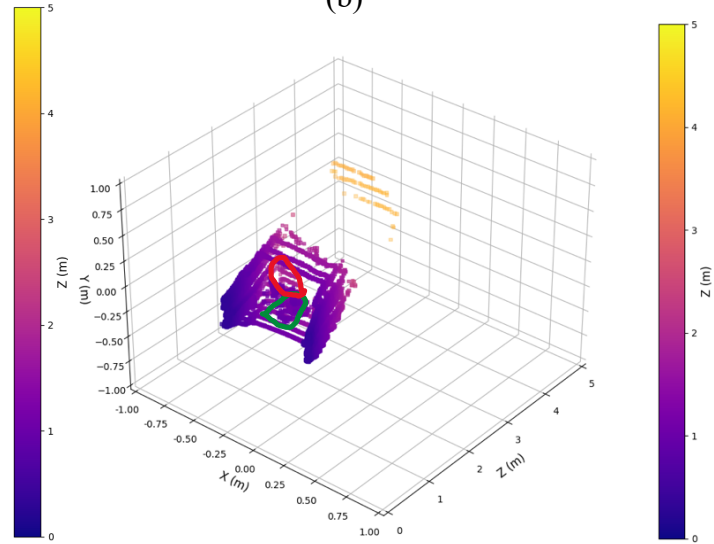
(a)



(b)



(c)

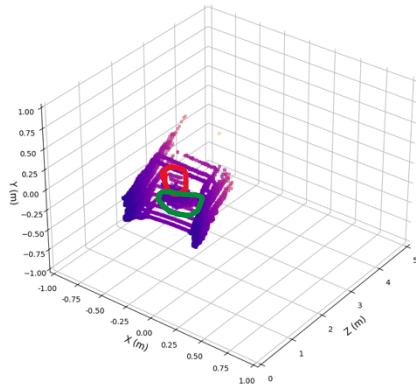


(d)



(e)

Figure 2.22. LiDAR point clouds and color image in full light, low dust conditions. Point clouds: (a) isometric, (b) side, (c) front, (d) top. (e) Color image.

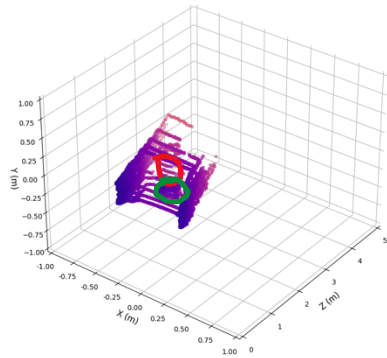


(a)

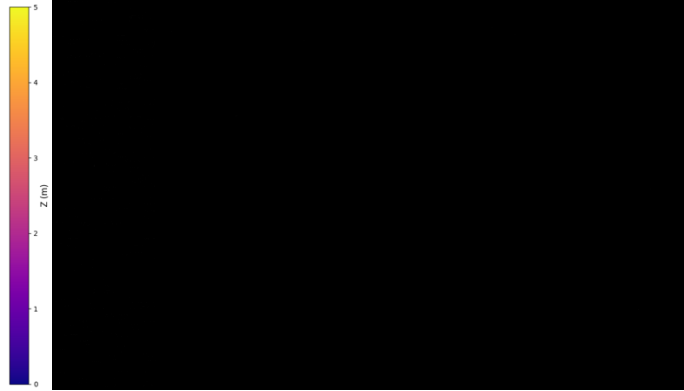


(b)

Figure 2.23. (a) LiDAR point cloud and (b) color image of half-light conditions with low dust.



(a)



(b)

Figure 2.24. (a) LiDAR point cloud and (b) color image of no light conditions with low dust.

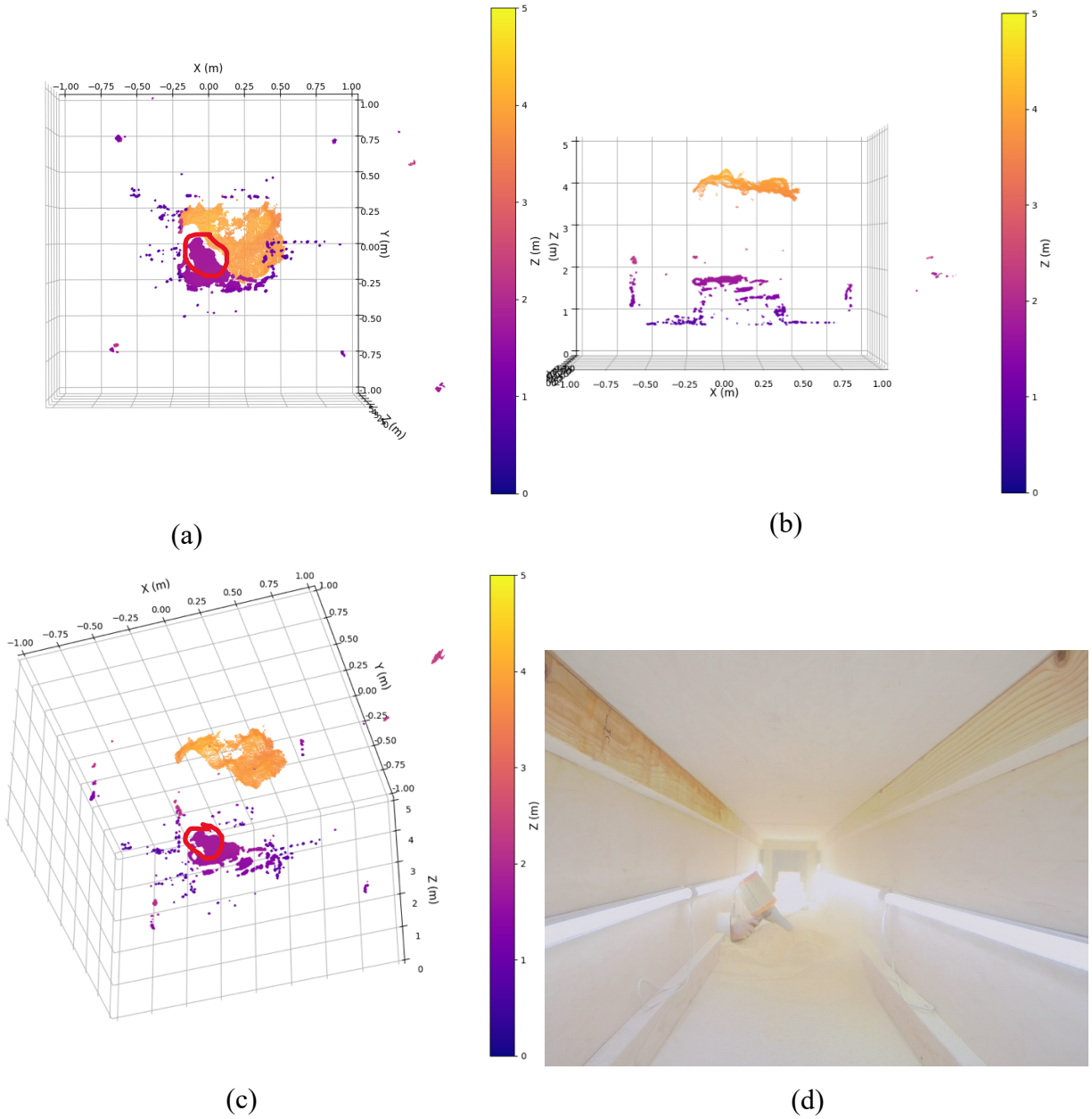


Figure 2.25. Stereo camera point clouds and color image in full light conditions with low dust.
Point clouds: (a) front, (b) top, (c) isometric. (d) Color image.

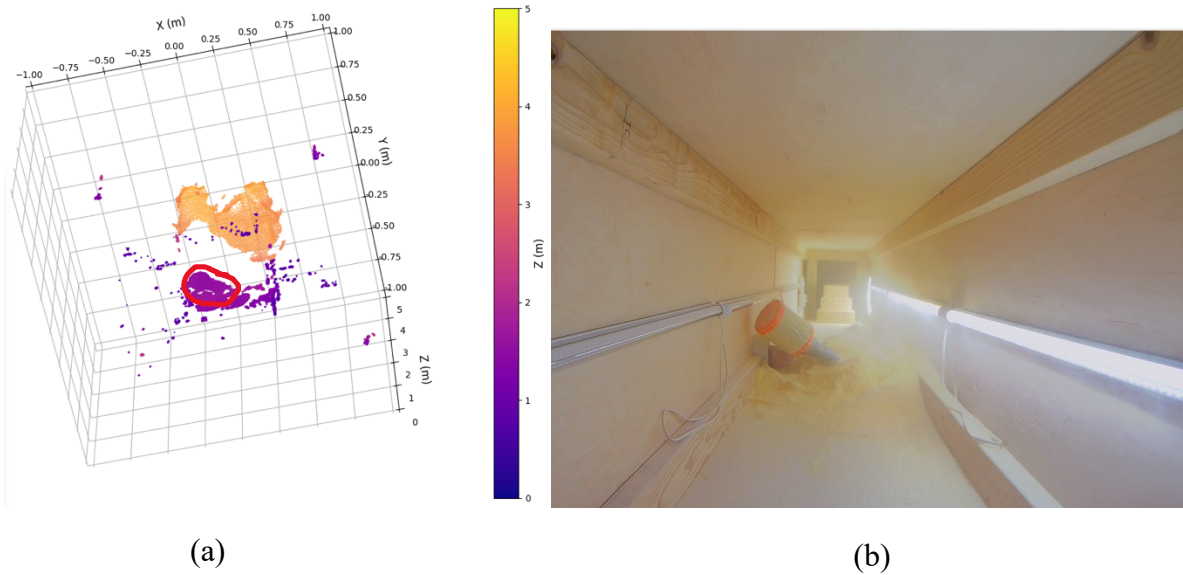


Figure 2.26. (a) Stereo camera point cloud and (b) color image of half-light conditions with low dust.

High Dust Comparison

Figures 2.27 and 2.28 show the LiDAR results for high-pressure (air nozzle produced dust) testing and 2.29 and 2.30 show the results for the stereo camera. In this section only for this chapter, the built-in stereo camera color image stream was used for the stereo camera RGB image due to the GitUp Git2 camera connection breaking. Figures 2.18 through 2.26 color image streams and 2.27, 2.28 and 2.31 will still use the GitUp. In the chapters that follow, RGB image streams used for computer vision work are exclusively from the stereo camera image stream.

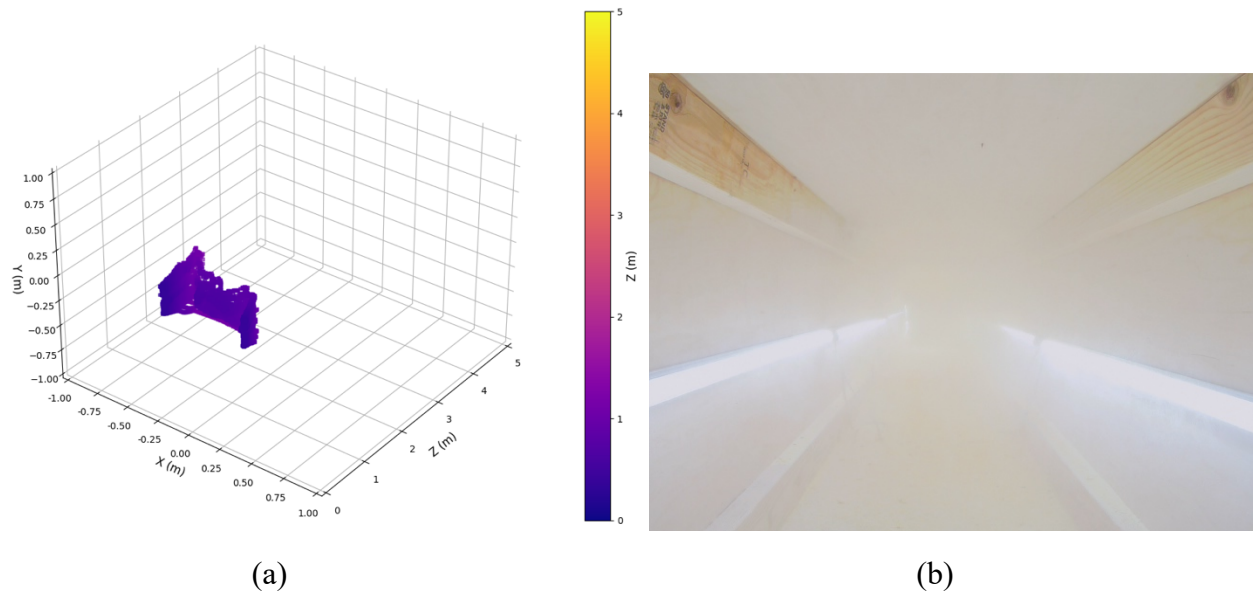


Figure 2.27. (a) LiDAR point cloud and (b) color image of full-light conditions with high dust.

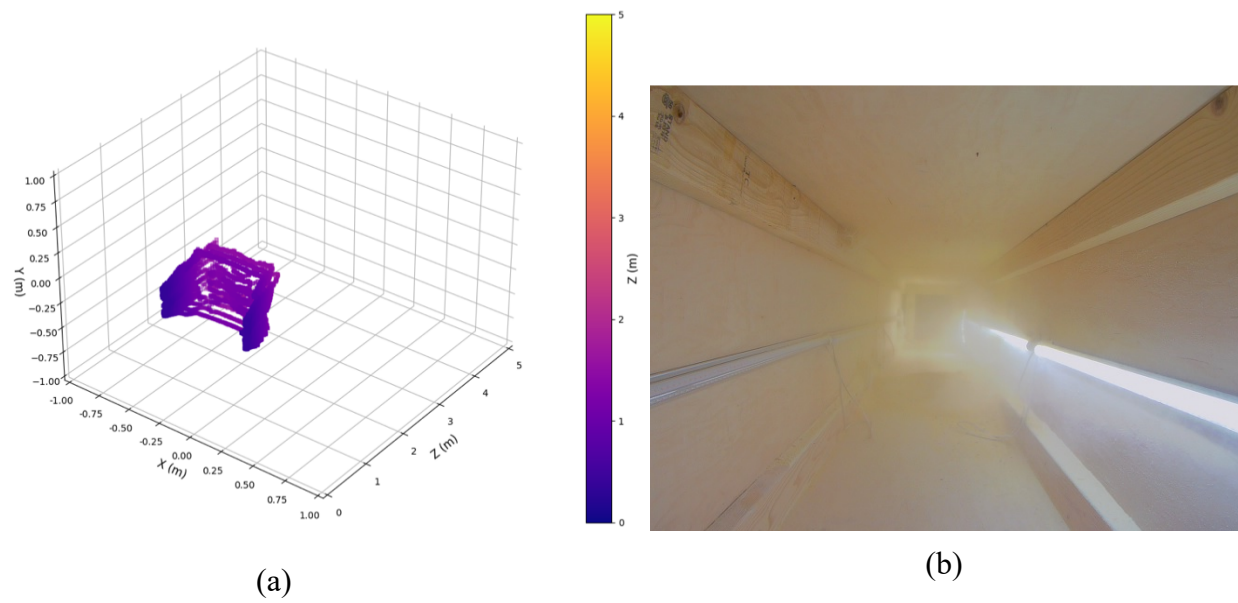
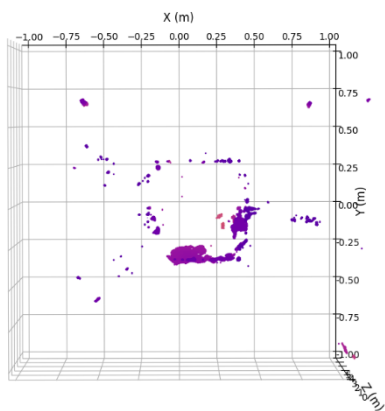
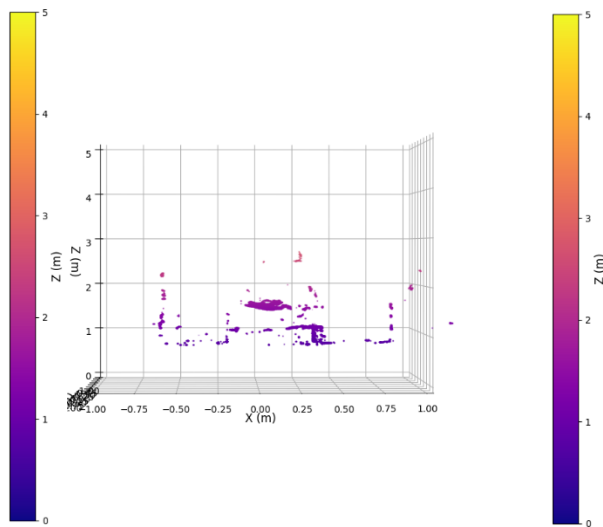


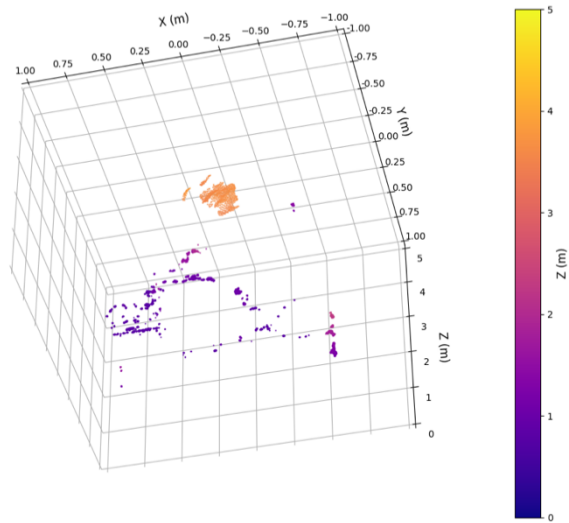
Figure 2.28. (a) LiDAR point cloud and (b) color image of half-light conditions with high dust.



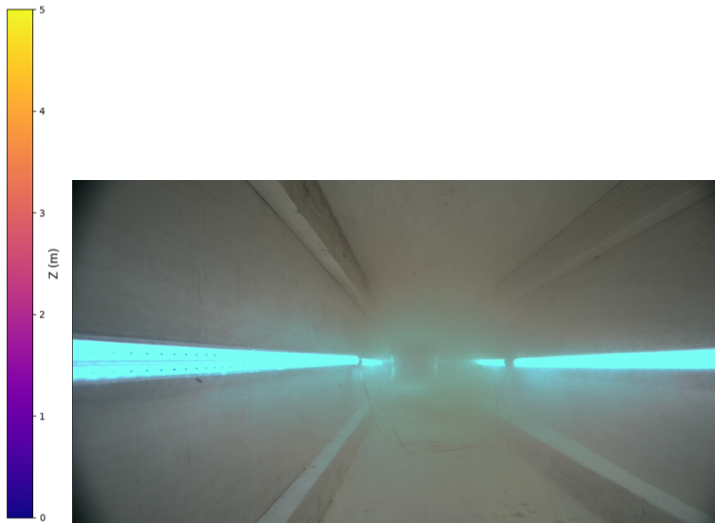
(a)



(b)



(c)



(d)

Figure 2.29. Stereo camera point clouds and color image in full-light conditions with high dust. Point clouds: (a) front, (b) top, (c) isometric. (d) Color image.

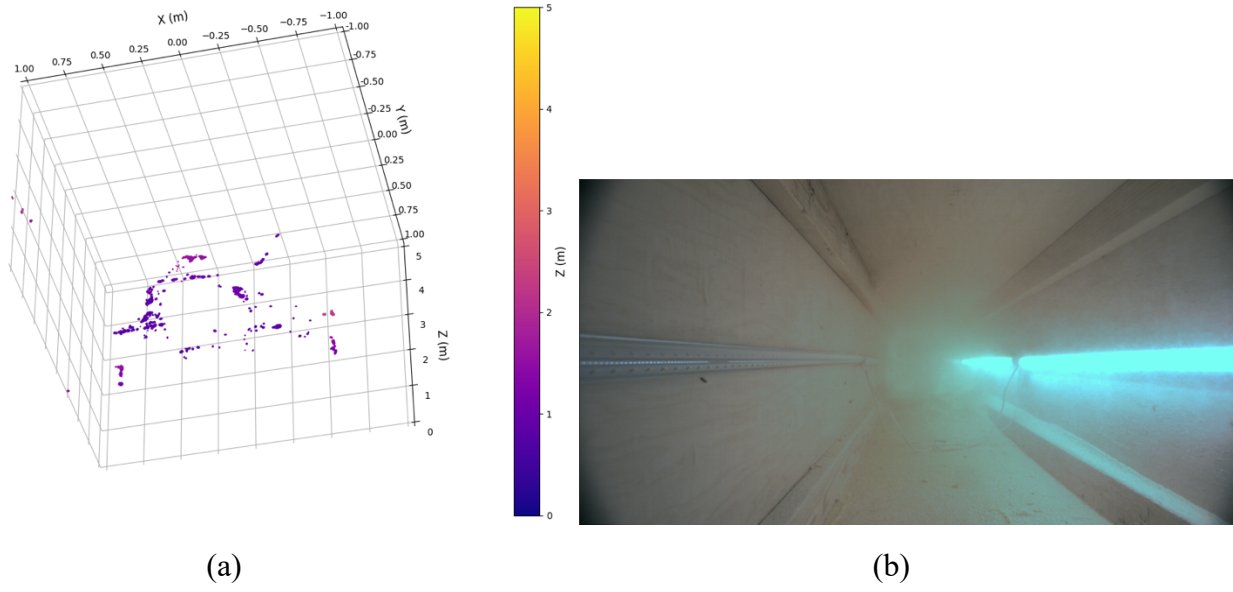


Figure 2.30. (a) Stereo camera point cloud and (b) color image of half-light conditions with high dust.

Different return modes for the LiDAR sensor (last or dual) were not tested up to this point. Using a different return mode for the grain cart would be unlikely to change results, because the dust level of the grain cart is similar to the dust level used for the return mode comparison in the next section. The low-dust comparison for dual or last return mode is unknown. However data was not collected retroactively for the dual or last modes for the low pressure comparisons because the high pressure return mode comparison showed no return modes capable of measuring past the high level of dust, which lead to the decision to move forward with the Multisense S21 for vehicle grain volume measurement. If one is interested in lower levels of dust, multi-mode testing will be required of the Puck to see which mode is the best. However, corn may not be the desired dust material in that instance, as one may be more interested in dust encountered outside of loading conditions in the agriculture fields.

Return Mode Comparison

Figures 2.31, 2.32, and 2.33 show the point clouds for the strongest, dual, and last return modes.

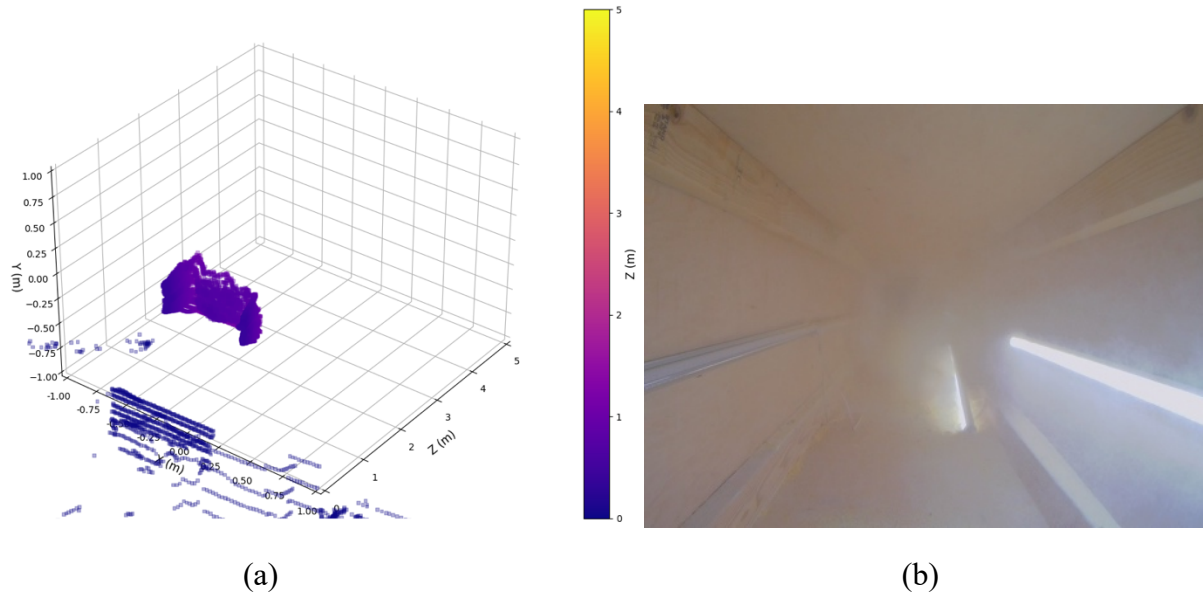


Figure 2.31. LiDAR strongest return mode. (a) Point cloud and (b) color image of half-light conditions with high pressure dust. One light fell off the wall.

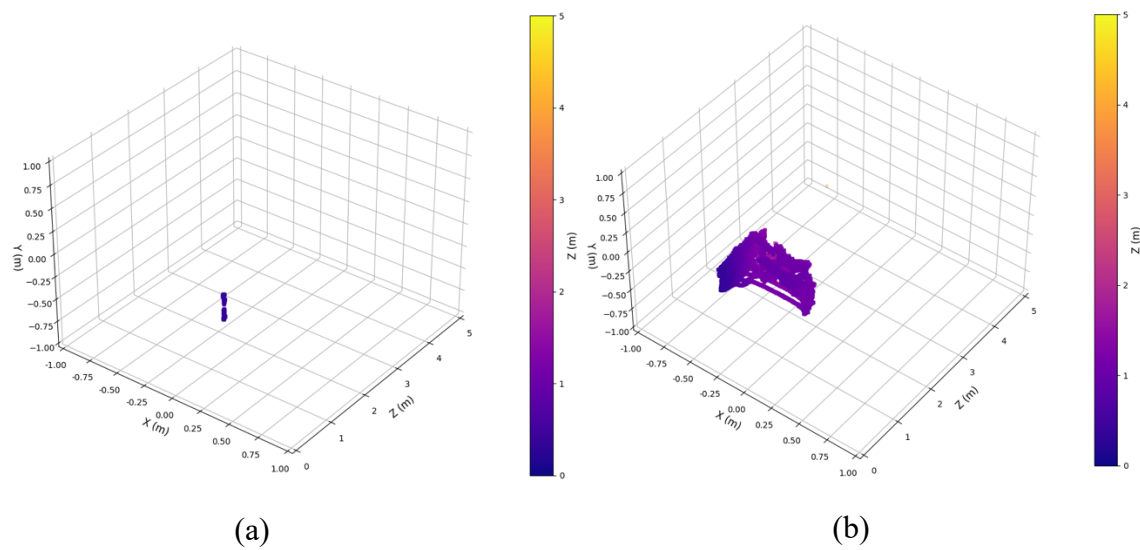


Figure 2.32. LiDAR dual return mode. (a) Unregistered points and (b) registered points (right) of half-light conditions with high pressure dust. One light fell off the wall.

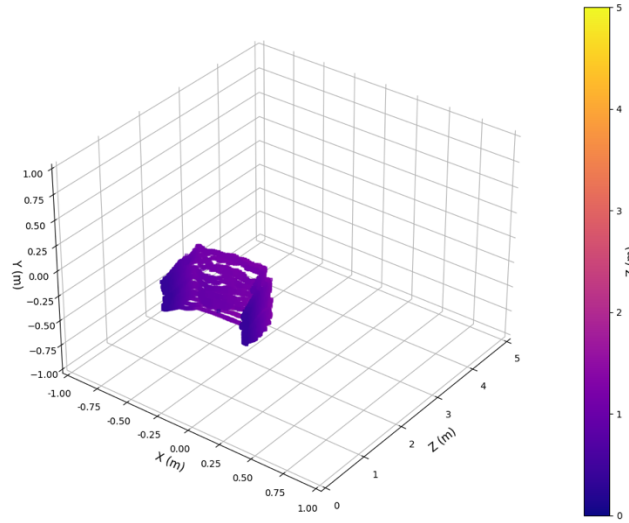


Figure 2.33. LiDAR last return mode. Half-light conditions with high pressure dust. One light fell off the wall.

2.7.3 Discussion

No Dust

Point clouds for the Velodyne for no dust in both full light and half-light look exactly the same. The outline of the dust chamber is very apparent to a view from the point clouds.

Similar to the Velodyne, the Multisense S21 no dust condition results both look exactly the same. These point clouds can also be used as a baseline to compare the dusty conditions point clouds against. Some faint occlusion from the edges of the wedding cake layers can be seen in the point cloud. Because the stereo does not do a “scanning” action when collecting point clouds, the walls of the dust chamber are not as prevalent in the point clouds compared to the LiDAR system.

Low Dust

A prevalent dust cloud can be seen in the images to the right and in front of the hair dryer in the LiDAR point clouds. The hair dryer can also be seen in the images, which was to be expected but must be noted. The hair dryer has been circled in red (Figures 2.22d, 2.23a, 2.24a) in most of the point clouds where the hair dryer was relatively easy to distinguish. The dust plumes were circled

in green (Figures 2.22b, 2.22d, 2.23a, 2.24a). The same effect can be seen across the low-pressure tests, regardless of lighting conditions.

The changing lights did not seem to result in any changes to the results for the stereo camera point clouds (Figure 2.25c vs 2.26a). Dust was not obvious in the point clouds (Figures 2.25 and 2.26). The hair dryer is apparent. The hair dryer is circled in red in the point clouds (Figures 2.25a, 2.25c, and 2.26a). The back wall and cake area were apparent in the point clouds as well when viewed in a larger full screen (not pictured but would be apparent in Figure 2.25a). In this low-dust comparison, the Multisense S21 is more robust to dust interference compared to the Velodyne VLP-16, with a lower percentage of pixels appearing to be affected by the dust.

Overall the lighter dust results were intriguing. When just looking at the Multisense images, the back wall is still visible where it is not blocked by the hair dryer or occlusion from the hair dryer (Figure 2.25b). These results indicate that the Multisense is robust to at least some dust particles, evidently finer dust particles in relation to the bulk of average corn meal grain size. On the contrast, a plume of dust can be seen in front of the hair dryer when looking at the LiDAR results (Figures 2.22b and 2.22d). The LiDAR is affected by dust that does not affect the stereo camera. Returns off of the dust plume could cause false computer vision assumptions, depending on the application, such as a false assumption of an obstacle or the false assumption of grain being present.

High Dust

Once again, light conditions in the dust chamber did not change anything regarding laser scan returns for the LiDAR system. Changes to the laser scan returns for the high dust conditions seem more dramatic compared to the low dust conditions, with no scan line even reaching the back of the box (Figure 2.27a and 2.28a). Every scan line is returning from a return off of the dust clouds.

As with the previous tests, changing the LED lights did not have an impact on point cloud results for the stereo system either. Unlike the low dust conditions, most of the back wall is now not present in the point cloud (Figure 2.29a, 2.29b, 2.29c, 2.30a). Unlike the LiDAR, the returns are just not being calculated and NAN values are put in place for those points as was also present in the low-pressure tests. This way, values for the dust cloud are not present in the point cloud. Dust

cloud measuring or visualization may have value in certain applications, but for the goal of measuring corn dust, it does not.

Heavier dust conditions caused issues for both systems. Neither system could calculate returns from the back of the chamber. As seen earlier but now much more exaggerated, the LiDAR shows returns from the dust clouds as opposed to the stereo camera, which just returns NAN values which the user of the instrument can choose to represent in the depth images however they choose. This discovery of how well the LiDAR system images dust in agricultural environments may lead to further studies in drift of corn dust, soil drift, or fertilizer drift in corn harvest, forage harvesting, dry fertilizer applications, or tillage. In regard to which system seemed more impressive from the dust chamber testing, the stereo camera seemed slightly more impressive in regard to the goal of measuring corn depth in loading vehicles. Out of the box it offers more point returns, and a lesser percentage of its points seem affected by the dust plumes.

Lighting

The lighting conditions of this test did not appear to have an effect on either system, either in dusty conditions or in non-dusty conditions. This does not mean that light can never have an effect on the imaging, it just means that the light sources used for the testing in this environment did not have any effects on the measurement systems. Laser lights of the same wavelength of the laser used by the VLP-16 could in theory cause false return values. Moreover, certain lighting conditions could cause some over saturation issues of the surfaces for the stereo camera. However, none of these phenomena were experienced during testing in this section. Moreover, light needs to return to the stereo camera lens for the camera to sense anything. Therefore, non-light conditions without an outside light source will deem photon sensing abilities of the system useless.

Return Mode

Both systems exhibited similar laser blocking effects from the dust clouds (Figures 2.31, 2.32, 2.33). The dual mode exhibited some sort of flickering or frame skipping as seen in Figure 2.32 which was captured during testing. Strongest (Figure 2.31) and last (Figure 2.33) both seemed to exhibit similar results with no differences noticeable between the return modes.

2.8 Conclusions and Recommendations

Point clouds were successfully streamed within Python to visualization the 3D data flow of the LiDAR in completion of objective 1. Videos of streaming are listed in Appendix 11 for YouTube links and housed in the Purdue University Research Repository (Rogers and Buckmaster, 2022). Raw data ROS .bag files are also stored in the repository. Images from Python were successfully generated and are displayed throughout this chapter using the same post-processing software as the LiDAR. 2., Both depth-measurement devices were compared when filling the same grain-carrying vessel while mounted on a grain cart. An observable difference included dust being measured by the LiDAR compared to creating “NAN” pixels with the stereo camera. Another difference was the absence of observable surface returns of the grain surface level within the loading vessel coming from the LiDAR due to dust coverage. While imaging in similar conditions, the stereo camera was able to image some pixels of the grain surface within the loading vessel. 3, A dust box was successfully constructed for the creation of varying light and dust plume conditions. Both sensors reacted to the dust, but in different ways as was observed in the outdoor grain vessel loading situation. The LiDAR imaged the dust while the stereo camera was unable to perform pixel matching in the affected areas, creating “NAN” pixels which can be ignored. The stereo camera was able to image some pixels at the rear of the dust chamber, while the LiDAR was not. Lighting did not change sensed return values for either sensor. During high-pressure conditions, neither sensor was able to return values from the back of the test chamber, indicative of full dust blockage. Return modes of LiDAR did not change results within the high-pressure dust chamber. The data quality has been compared to complete objective 4. Considering the effect of the dust conditions, the stereo camera seems more robust than LiDAR for the automatic volume measurement of grain in vessels.

3. VOLUME MEASUREMENT OF PARTICULATE BIOLOGICAL MATERIALS

3.1 Abstract

TMR and corn grain volume was measured using a Carnegie Robotics S21 stereo camera in near real time (sub-second). Volume was measured regardless of the pile characteristics from a distance of 3 m. Resulting accuracy was good for a target of 8 liters of shelled corn with final values between 6.8 and 8.3 liters from three varied surface scenarios. Development of the volume calculation is discussed, and the required camera calibration techniques needed for accurate results are outlined.

3.2 Introduction

Precision livestock farming, like precision field crop farming, holds potential to improve efficiencies and in doing so, improve profits and sustainability. But in the case of many livestock situations, only the output, such as weight gain, wool production, or milk production, is measured on a per animal (or in case of dairy cattle, per udder quarter) basis. Inputs are aggregated across groups because the feeding is most commonly in groups. Even in the case of computer-controlled feeders, part of the intake for each animal must be inferred from group data. We need methods, both in research and in production settings, to measure, monitor, and control at an individual animal basis.

In many other situations such as with grain, fertilizers, or feedstuffs, volume estimation is important for inventory tracking and even logistics of operations. Weighing material coming in and going out can work in some situations but not all. A method that is not disruptive and camera-based would improve flow of many operations and enable up-to-date information. Stereo vision camera technologies enable distance measurement for each pixel. This offers tremendous opportunity for volume measurement.

3.2.1 Literature Review

Computer vision and animal husbandry have been a popular topic for decades. A 1991 Belgian literature review (Van der Stuyft, 1991) examined the feasibility of applying computer vision including 2D images and structured light to livestock systems in order to improve the “health, welfare and efficiency in livestock production.” Schofield (1990) developed a method for measuring the area of a pig using images taken directly above the pig. Using the top area of the pig without the head and ears was found to be most effective approach to estimating pig weight and they were able to get within 95% of the correct weight with image analysis (Schofield, 1990).

Stereo cameras have proven useful for measuring the amount, either in height or volume, of biological plant materials. A Belgian study found stereo vision to be a “cheap, compact, and flexible way” to look at wheat canopy architecture and measure canopy height in field conditions. Spike top height of winter wheat was measured to an accuracy of 97.1% using stereo vision (Dandifosse, 2020). A Swedish study used stereo cameras to measure the volume of timber on logging trucks as they drove through a station containing 3 pairs of stereo cameras. The resulting mean relative percentage error (MRPE) was as low as 6.7% depending on environmental conditions with the cloudy dataset of images actually producing the best results compared to sunny, night, or glared (Rundgren, 2017).

Some studies have begun the work on applying cameras to animal feed tracking. An Israeli study utilized stereo vision cameras to estimate the amount of total mixed ration (TMR) eaten for each trip to a feed slot used by individual cows in an open dairy cow shed (Bezen, 2020). A convolutional neural network (CNN) was trained with stereo images of TMR heaps to learn the weight of TMR. Also, a Faster R-CNN was trained with images of digits located on collars worn by cows to identify each individual cow. The RGB-D cameras were mounted above the heads of the feeding cows in the feeding slots. 93.65% of the cows were able to be identified and the amount of feed consumed for a single meal in the range of 0 – 8 kg was measured within a mean absolute error (MAE) of 0.127 kg and a mean square error (MSE) of 0.034 kg. While this system yielded good results, the required training of neural networks to measure feed amounts is a hindrance. The required training to read cattle collars may be not be a hurdle, though, as the trained network for collars should work in most dairy environments. A Chinese study attempted to combine scores of

body condition and individual cow recognition with a camera system measuring depth, gray, and phase congruency. Images of cow backs were taken and together with manually collected data were used to train a CNN. 3430 images comprised the data set, where 2400 were used for training and 1030 were used for testing. Accuracy for online testing was 93.7% for individual cow identification and 40.9% for body condition scores (Yukun, 2019). Once again, this study requires the use of CNNs in specific environments.

Discussion of Feed and Animal Tracking Possibilities

As a typical scenario, a dairy cow may consume up to 22 kg of dry matter feed per day, spread between 1 and 6 feeding bouts per day. Dividing by the dry matter percentage (assuming 60%), 37 kg of a total mixed ration (TMR) is fed per day. TMR has a nominal density of .3 kg/L but this can vary greatly. This 37 kg TMR at .3 kg/L would be a total volume of 123 L. A reasonable accuracy goal for intake measurement at on individual animal basis would be 1%, therefore a system that can measure 120 L of feed within 1.2L. There will be some volume balance calculations (pseudo-calibration) also possible since amount delivered is generally known and in research settings, the refusals (feed left behind daily) are also measured.

A variety of computer vision techniques can be used to identify unique markers on animals. To reduce complexity, these markers will need to be artificial, such as ear tags, paint, distinguishing colors, or number collars. Template matching has the potential to match the template of unique characters on collars, ear tags, or painted numbers on the animals. Masking based on unique factors known in the images can reduce the complexity of finding the unique features. With the depth camera, we can eliminate areas of images that would not match the depth of where the unique identifying features are expected to be found. As an example, ear tags would not be expected on the ground level or near the ceiling in cattle sheds. If lighting is somewhat consistent, it may help identify the unique features using color matching either in the RGB or HSV space. A yellow tag should never appear red unless unique lighting conditions permit it. Knowledge of the cattle shed working atmosphere will need to be known to create a robust system. Moreover, as shown by Bezen, CNNs may be viable to train to learn to identify the numbers on cattle collars (Bezen, 2020). The cattle collar numbers should appear very similar in a variety of environments, so a standard cattle collar (or ear tag) training set could be established to identify the individual animals. All of

these solutions have the potential to work with a single depth camera doing both depth calculation and unique animal identification in both depth and color. Finally, the depth camera or the color camera feed of the stereo camera combined with the calibration techniques discussed in this paper can be used to identify certain unique body features of the animals, in a similar fashion discussed by Shofield (1990) and Yukun (2020). Width of the animal, height, distance from ground to belly, or other features could be measured, enhancing the value to animal husbandry.

3.3 Objectives

Toward the larger goal of estimating individual animal feed consumption, the specific objectives of work reported in this chapter were to:

1. Develop an algorithm to automatically compute actual pixel area from a stereo vision camera.
2. Demonstrate the measurement of the volume of a pile of biological particulate materials using livestock feed and corn grain.

3.4 Methods and Materials

Regular RGB cameras provide length and width of images (hence pixels) as metadata. Stereo cameras also enable the measurement of depth. The depth channel enables, with appropriate mathematical manipulations, accurate computation of the volume of objects within the image after proper calibration and techniques are utilized. Analogous to digital elevation models in topography, stereo vision can yield a quantitative 3-dimensional representation of a surface. With knowledge of the surface below a pile of materials, the difference in the surfaces can be used to compute volume. In the instance of a flat background surface such as a feed bunk, it is conceptually simple, although algorithmically complex, to compute volume of feed on offer. This approach of measuring volume allows for the near real-time measurement of feed volume laying within a known area of a feed bunk. With continual monitoring, cumulative feed consumption of the animals can be estimated. Both TMR and silage will be referred to as feed in this section and corn grain will be referred to as grain.

3.4.1 Volume Measurement Theory

Each pixel in a depth image is a visual representation of a section of a surface in space that has been captured by the imaging device. If we know “the backstop” or under surface, then each pixel also has an associated volume with it. We must know the length, width, and depth of each pixel. Similarly, a color image with no depth associated with it still scales the length and width of the objects in space to pixels in the image. The knowledge of the number of pixels along the length and width of an object in an image being directly proportional to the actual length and width of the object in space that was captured in the image is used to determine the length and width of actual objects using only the image. This knowledge is also assuming that the image has been rectified (as is indeed the case for the camera used in this chapter) which means that the length and width values represented by pixels near the edge are the same as in the center of the image.

Calibration is required to determine how many pixels in the x and y direction of the pixel grid within images represent standard units of measurement. Moreover, a calibration equation needs to be established relating the change in depth to the change in horizontal and vertical pixels representing a unit of length. This is due to the number of horizontal and vertical pixels representing a unit of length at a certain distance from the camera will change as the distance from the camera (depth) changes.

Consider a situation where a 1 m by 1 m square grid is laid on the ground and then imaged using a depth camera 1 m directly above the grid. Let us assume the x and y (length and width) measurement scale are the same and the square grid surface plane and the camera lens plane are parallel. We placed the grid so that the length and width of the grid are parallel to the x and y directions of the image grid. It is observed that at the 1 m depth, 1000 pixels in the x direction and 1000 pixels in the y direction directly align with the grid on the ground. After this image, another image is taken by raising the camera another 1 meter from the square grid but the grid alignment with the x and y directions stays the same. For this image, 500 pixels in the x direction and 500 pixels in the y direction are observed to align with the square grid. Using this knowledge, a linear equation can be developed to determine the length of objects observed in images when the number of pixels in the x or y direction are known and the distance that the image was taken is known. (Please note that the depth does not need to be known if the depth is constant for the objects being

observed. If an imaging system is being used to measure the length of individual corn stalks where each stalk is laying on a conveyor belt, the depth can be assumed to be constant and a yard stick for calibration can simply be placed on the conveyor belt. One would count the number of pixels in the x and y directions representing the known length of the yard stick shown in the image. It is easier to calibrate if the length of the yardstick is parallel to the image borders.) Once the change in area that each pixel accounts for in relation to the depth is figured out, volume can be calculated assuming each pixel is a small cube. Each pixel represents a discrete volume because it represents the length and width of an object, and has a number associated with it which is the actual depth (in meters for the camera used in this work) of the object represented by the pixel from the camera (Figure 3.1).

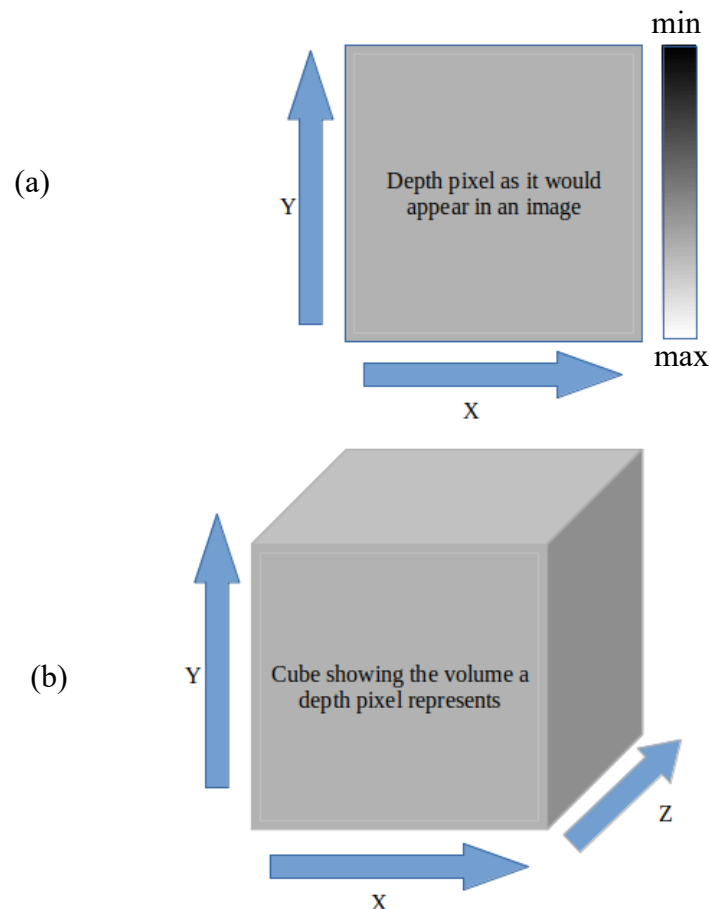


Figure 3.1. (a) Illustration showing the relationship between pixel and the depth it represents. (b) Each pixel (below) has a volume associated with it.

As another example, let's say the camera is "zoomed in" to an image, and we are using a depth camera. The next section discusses calibrating depth and area, but let's also assume that the calibration process is already done. Next, assume the pixel represents an area of 0.0001 m² (pixels of 1 cm x 1 cm), and the pixel's third dimension (depth) is indicated to be 0.80 m. If the backstop (floor or container) had a depth dimension of 1.0m for the same pixel, the volume of the space represented by that pixel would be the area associated with that pixel (0.0001 m²) multiplied by the product/feed/grain depth associated with it (1.0-0.8 = 0.2m) for a volume of 0.00002 m³ or 20 cm³. This principle of the pixels representing area and volume is used to calculate the volume of the material. A pile of material would be assumed "solid" with the same material underneath the top surface. If the camera plane is parallel to the ground plane, the volume is the sum of the depth-adjusted or corrected area of each pixel times the material depth of each pixel:

$$V_{total} = \sum_{p=1}^n A_{c,p} * (D_p - D_{p,r}) \quad [3.1]$$

Where: V is the volume,

p is the pixel,

n is to total pixels,

$A_{c,p}$ is the area corrected for image distance,

D is the depth,

And $D_{p,r}$ is the depth of reference of the same pixel.

3.4.2 Assumptions

Figure 3.2 illustrates the parallel plane assumption of the mounted camera system. Overestimation of volume is possible due to the false assumption of biological material being present along the camera line of sight all the way to the base of the holding container. Figure 3.3 illustrates the parallax assumption that may lead to miscalculations of volume, again due to the nature of the camera line of sight being incapable of seeing enlargements of objects in certain situations where the rate of base enlargement is not large enough for the camera angle of sight to register the changes. Parallax is worst when neighboring pixels have large depth differences. While boxes might lead to large parallax error, piles of feed should have relatively low parallax errors.

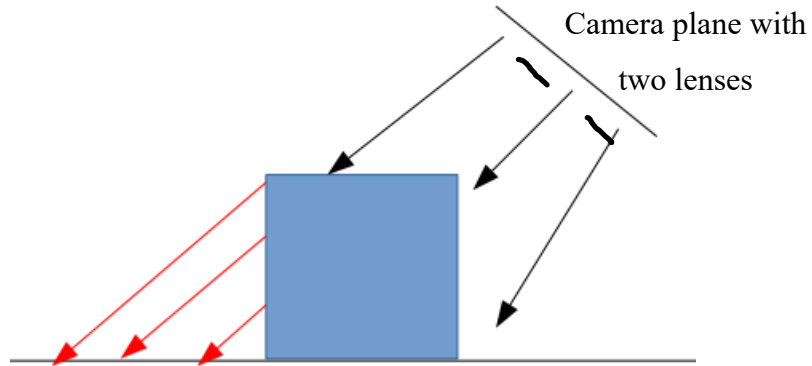


Figure 3.2. Importance of the parallel plane assumption illustrated. If camera lens plane (top right) is not parallel with the ground plane, extra volume (redlines) is added to the volume estimate of the cubic structure.

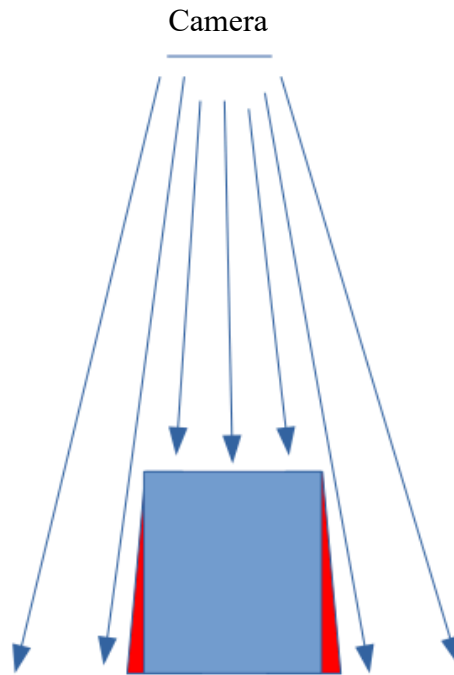


Figure 3.3. Parallax problem of underestimating volume. Camera lens is represented by top line and unmeasured area is represented by the red area of the square frustum.

3.4.3 Calibration

The camera lens planes were kept parallel with the ground where the calibration materials were placed. A standard piece of paper (8.5" by 11") was used to gather images at 2 different depths. Figure 3.4 shows the calibration images.

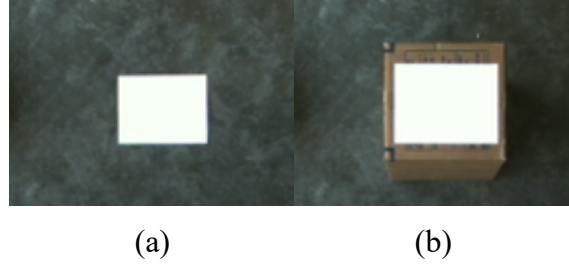


Figure 3.4. Calibration images showing the paper on (a) ground and (b) raised levels.

The known area of the paper is 0.06032 m^2 , which is constant. The first depth calibration images were ground level, where the paper was at 0 m at depth (ground plane is used as the depth reference in this chapter). Next, the paper was placed closer to the camera lenses (0.448 m high) on a box. When the paper was on the ground and further away from the camera, the number of pixels representing the paper in the image were less (1496) compared to the number of pixels when the paper is near the camera (2040). The paper was placed so that the edges of the paper were parallel with the x and y planes of the image matrix. The ground level area to pixel ratio was found to be $4.03 (10)^{-5} \text{ m}^2/\text{pixel}$ the same ratio was found to be $2.96 (10)^{-5} \text{ m}^2/\text{pixel}$ for the raised level. The total image frame for the entire project was cropped to 56,700 pixels. The area to pixel ratios were multiplied by the total number of frame pixels to gather the total represented area for ground level pixels (0 m depth) and pixels at the raised level (.448 m depth). The ground level pixels represented an area of 2.28 m^2 and the raised depth pixels represented an area of 1.68 m^2 . The area represented changed from 2.29 m^2 to 1.68 m^2 as the height increased by 0.448 m. These numbers lead to the slope for the linear equation to be -1.36 m with an intercept of 2.29 m^2 . The calibrated area will be known as corrected area and will later be used for the TMR volume equation. Figure 3.5 shows a flow chart for some of the area calibration as well as the rest of the volume calculation process. Finally, the equation for the corrected area is:

$$\text{corrected area (m}^2\text{)} = -1.36(m) \cdot \text{height}(m) + 2.29(\text{m}^2) \quad [3.2]$$

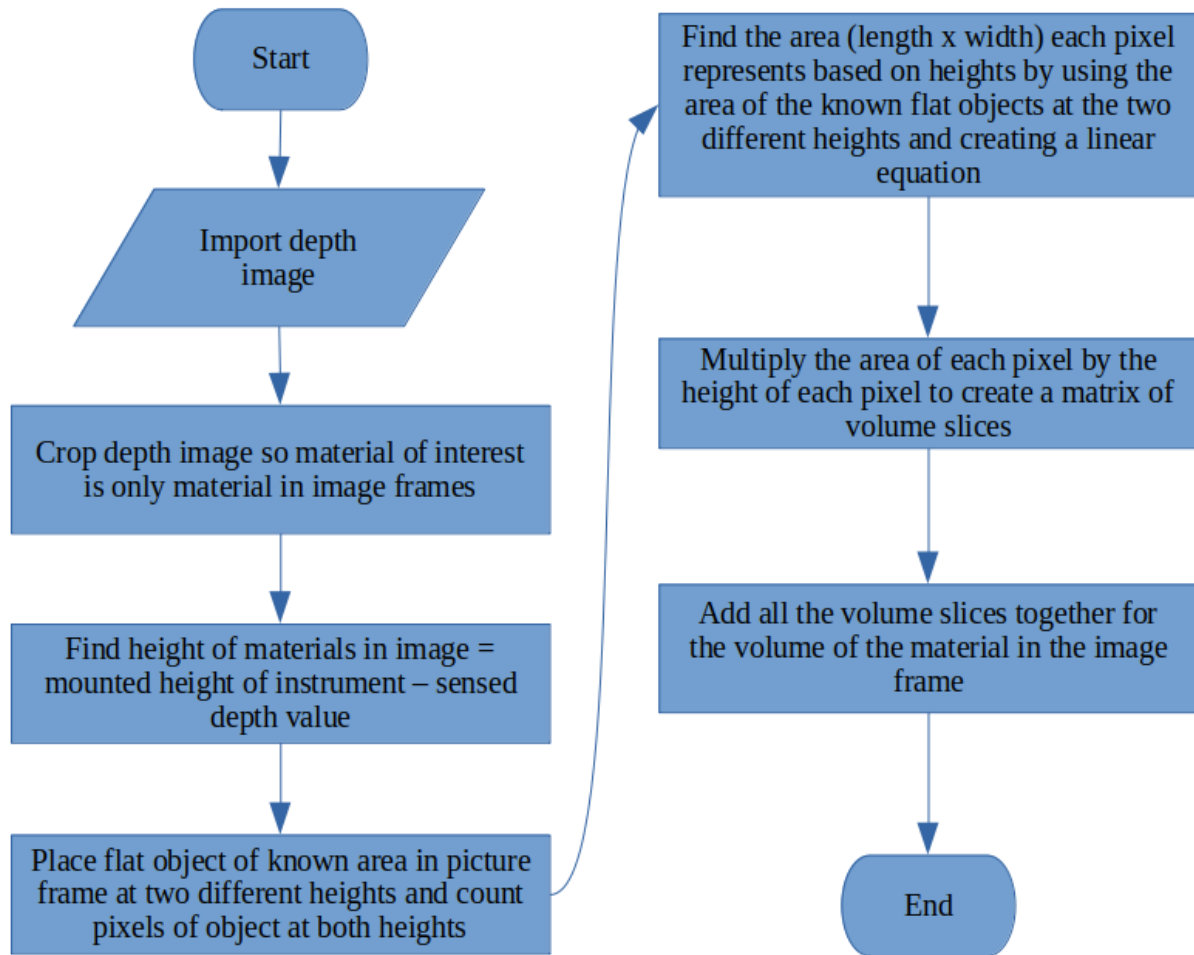


Figure 3.5. Flowchart of volume calculation process, from calibration to final calculation.

3.4.4 Biological Material Volume Testing

As a demonstration, some TMR was collected and spread out on a concrete floor and imaged using a stereo camera. Figure 3.6 shows the TMR still in the bucket and dumped out. The stereo camera was mounted 3.0 m above the ground using a step ladder (figure 3.7) and calibrated in accordance with the calibration section. The camera used was a Carnegie Robotics Multisense S21 (figure 3.8) that operates using Robot Operating System (ROS) Noetic and Python was used to interpret the data coming from the images. OpenCV, rospy, NumPy (NumPy, 2022), and cv_bridge were the Python libraries needed to import the images from ROS into Python and convert the images into usable volume information in the manner discussed in the Volume Theory (3.4.1) section of this chapter. The lens planes of the stereo camera and the ground plane were kept approximately parallel at all times by visually confirming the images on a computer screen. The ground floor was

the zero-depth reference such that if no TMR was below the camera, depth readings of 0.0 would be registered. The feed was collected and placed inside two (nominal) five-gallon buckets. Together, the buckets hold a volume of 10.5 gallons, so the TMR should be roughly 47.7 liters (10.5 gallons). The volume measurement was developed using the volume techniques discussed in the previous sections. In the same fashion as the TMR, corn grain volumes were also measured. Shelled corn grain was used since its density would be more consistent than the forage-based TMR. Corn grain was also spread in piles of different sizes and shapes. Calibration and measurement formula development were the same as the TMR although the calibration coefficient and intercept were different due to slightly different camera mounting heights (sub cm difference). Table 3.1 lists the software used for the data collection.



Figure 3.6. Buckets of TMR and spread TMR.



Figure 3.7. Step ladder and camera mount used for TMR imaging in the lab.

Table 3.1. Software used for this work.

Software Item	Description
Computer operating system	Ubuntu 20.04.2 LTS
ROS	Noetic 1.15.11
Python	3.8.10
OpenCV	4.5.4-dev
rospy	1.15.11
numpy	1.21.0
cv_bridge	1.15.0
multisense ros noetic package	4.0.5

A few scenarios of posing the TMR were conducted to test the feasibility of the volume measuring system in the order:

1. One bucket of packed TMR.
2. Two buckets of packed TMR (one added).
3. One bucket packed; one bucket spread.
4. Two buckets of TMR, both spread.

5. Two buckets of TMR, right side spread further.
6. Two buckets of TMR, left side spread further to approximately same as right side.
7. .023 m³ of the sample removed from the spread samples.
8. All TMR removed.

The following scenarios were evaluated for the corn grain volume measuring system:

9. Approximately 4.5 L (1 gal) of material spread.
10. Approximately 4.5 L (1 gal) of material with craters shaped as a “smiley face.”
11. Approximately 9.1 L (2 gal) of material piled.
12. Approximately 9.1 L (2 gal) of material with a single crater.
13. Approximately 9.1 L (2 gal) of material spread flat.
14. Approximately 18.2 L (4 gal) of material piled.
15. Approximately 18.2 L (4 gal) of material with three craters.
16. A small handful of material.

Table 3.2 lists the hardware used for the data collection. Figures 3.8 to 3.25 illustrate the series of TMR samples and figures 3.16 through 3.23 illustrate the series of shelled corn samples.

Table 3.2. Hardware and materials used for this work.

Material	Description
Laptop	Dell Precision 3520
Stereo Camera	Carnegie Robotics S21
Camera platform	Step ladder with crossbar used for mounting
Flat level for TMR	Concrete ground floor
Flat level for calibration	Cardboard box
TMR feed sample	Provided from Purdue ASREC Dairy Unit

3.5 Results and Discussion

3.5.1 TMR

The TMR sample used had a varied density depending on whether the sample was packed or spread. Even so, figures 3.8 through 3.15 demonstrate that volume estimation of feed laying on a flat surface using stereo imagery is feasible. As the sample was increased (figures 3.9) or density decreased (3.10, 3.11, 3.12, 3.13), the volume estimate increased. When sample was removed (3.14, 3.15), the volume estimate decreased. A heat map was applied to the depth images, with lighter yellows representing taller piles. Blue and red pixels are mixed together due to some pixels having slightly negative height at the blue areas (sub mm) due to slight rounding errors in some pixels.

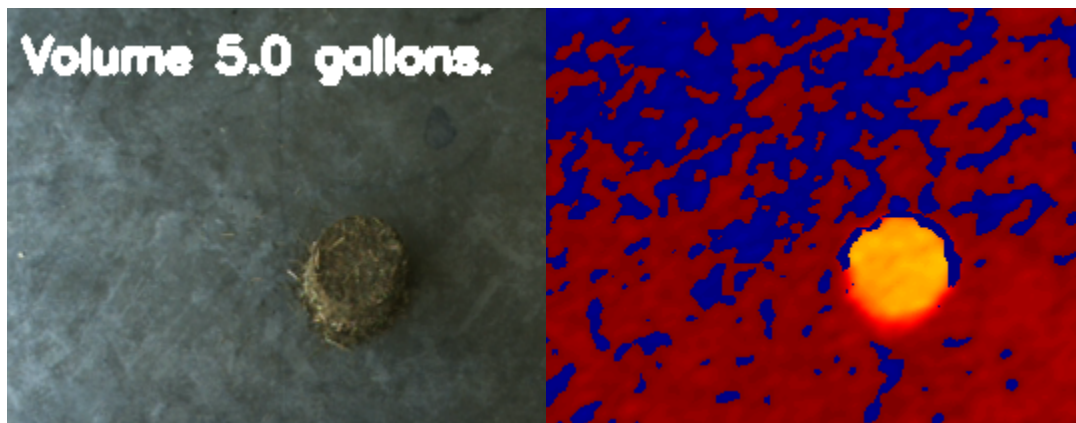


Figure 3.8. Bucket of TMR, still in tact, and a flat surface (left: color image; right: depth image).

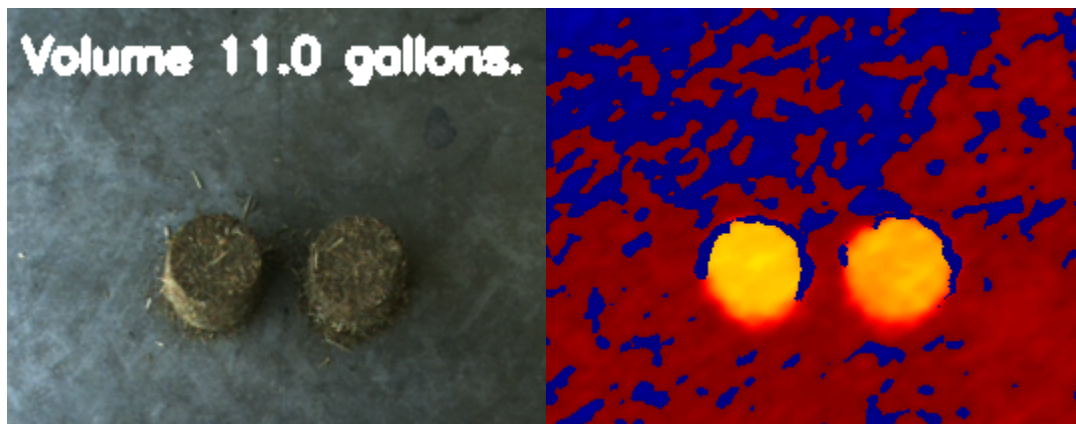


Figure 3.9. Two buckets of TMR (in tact) on a flat surface (left: color image; right: depth image).



Figure 3.10. Two buckets of TMR on a flat surface with one spread (left: color image; right: depth image).

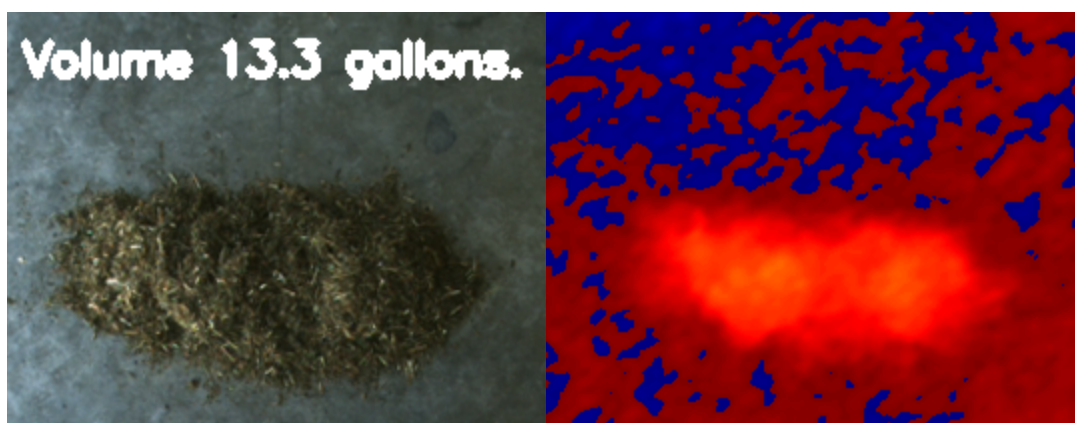


Figure 3.11. Two buckets of TMR on a flat surface with both spread (left: color image; right: depth image).

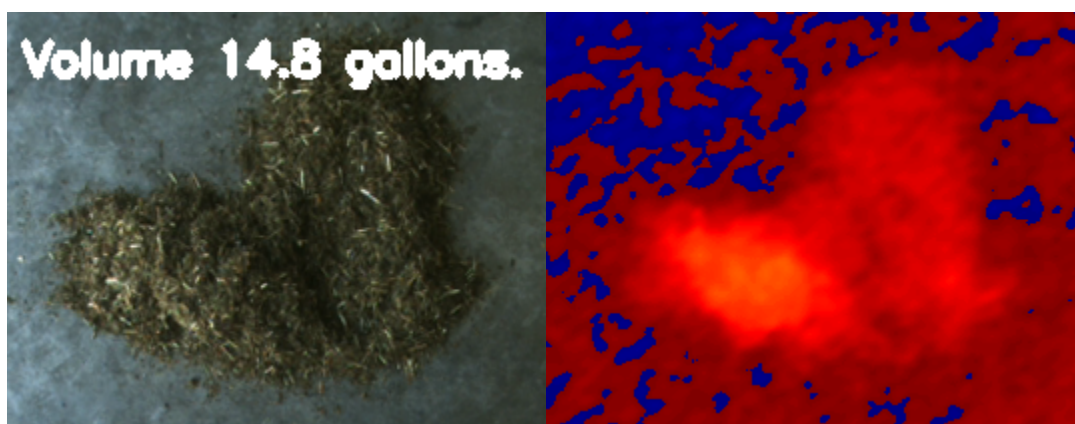


Figure 3.12. Two buckets of TMR on a flat surface both spread and or spread further (left: color image; right: depth image).



Figure 3.13. Two buckets of TMR on a flat surface with both spread further (left: color image; right: depth image).



Figure 3.14. A spread TMR on a flat surface after a 5 gallon bucket was removed (left: color image; right: depth image).

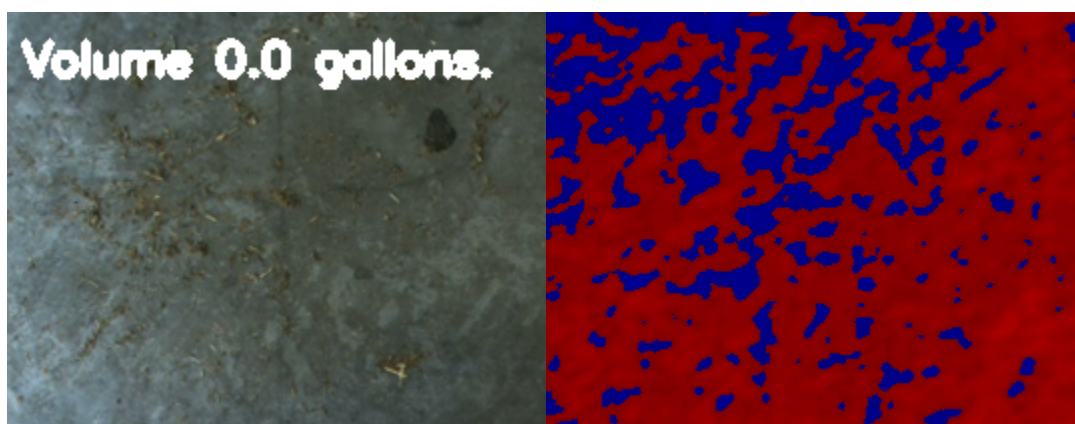


Figure 3.15. Flat surface showing all feed removed (left: color image; right: depth image).

3.5.2 Corn Grain

Figures 3.16 through 3.23 show the corn volume calculations, with the color images on the left and the depth images on the right. Table 3.3 showcases the maximum volumes observed for the corn pile scenarios and the range of values that were observed. The range occurs over the course of about 10 seconds for each scenario. The corn volume measurement system is replaying the .bag file that was recording at 10 frames per second and applying the new volume measurements to each frame. Each frame is a new calculation because the disparities calculated by the stereo camera may be slightly different from frame to frame for each individual pixel, resulting in a slightly different depth measurement in each pixel from one frame to the next.

Table 3.3. Ranges and max values of corn volume measurement scenarios. Scenarios 7 and 8 should have the same volume; 9, 10, and 11 should have the same volume; and 12 and 13 should have the same volume.

Figure	3.17	3.18	3.19	3.20	3.21	3.22	3.23	3.24
Scenario	7	8	9	10	11	12	13	14
Range (liters)	.8	.5	.8	0	.8	.5	0	.5
Max (liters)	3.0	4.2	8.0	6.8	8.3	15.1	20.0	0.0

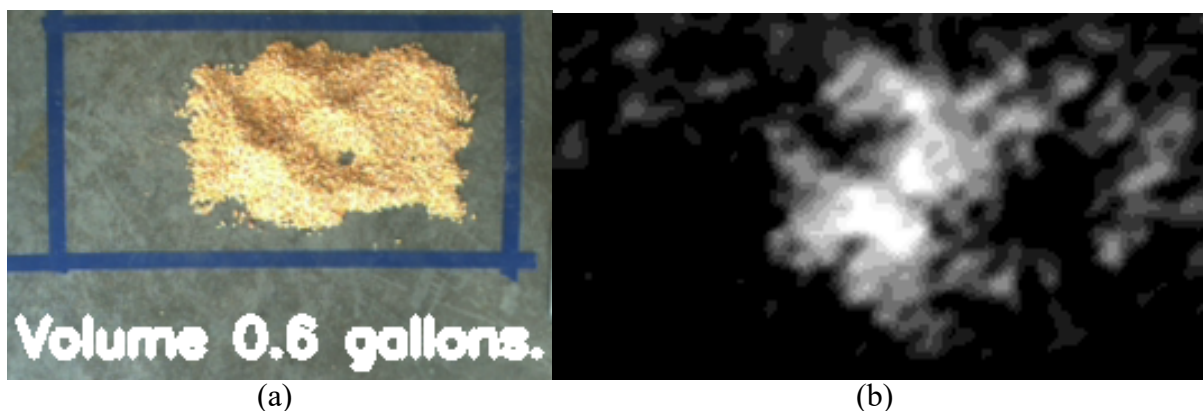


Figure 3.16. Flat surface showing approximately one gallon of grain with craters. ((a) color image; (b) depth image).

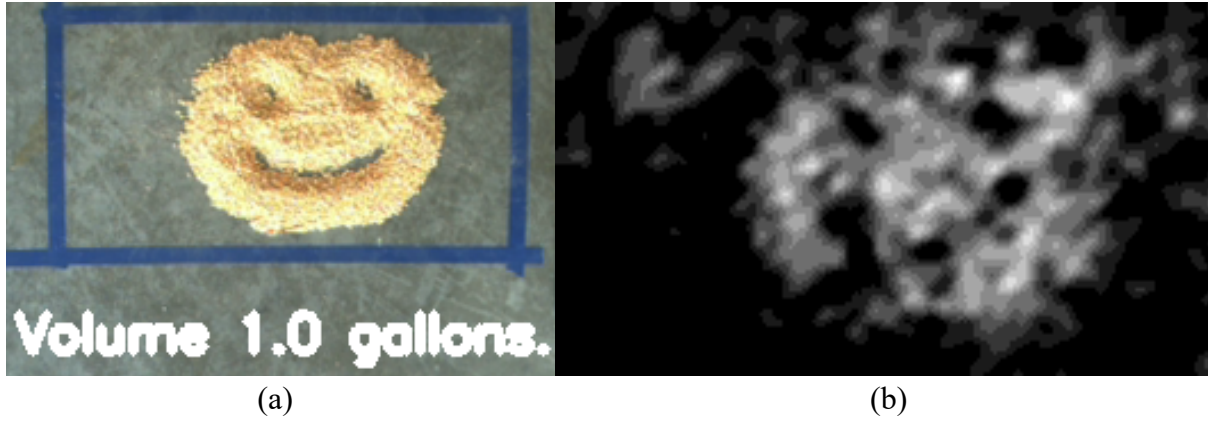


Figure 3.17. Flat surface showing approximately one gallon of grain with craters resembling a “smiley face.” ((a) color image; (b) depth image).

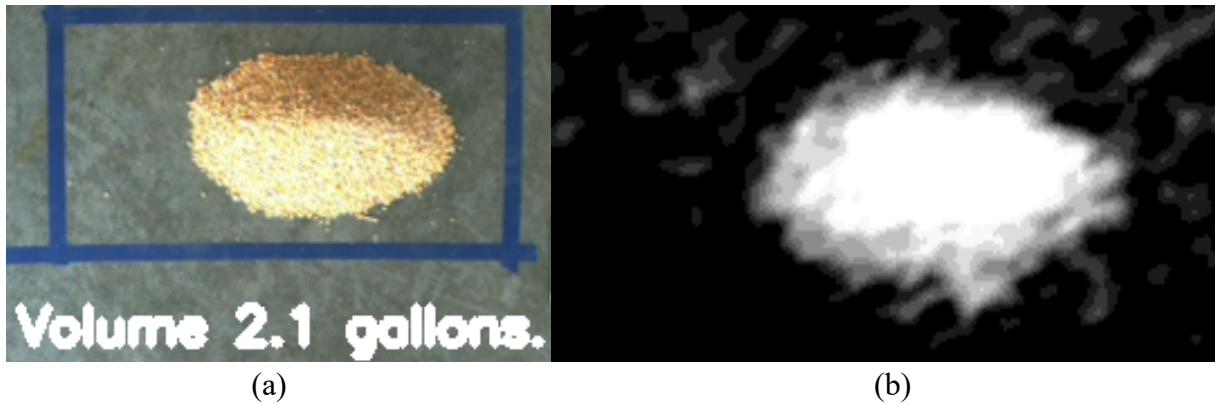


Figure 3.18. Flat surface showing approximately two gallons of grain in a pile. ((a) color image; (b) depth image).

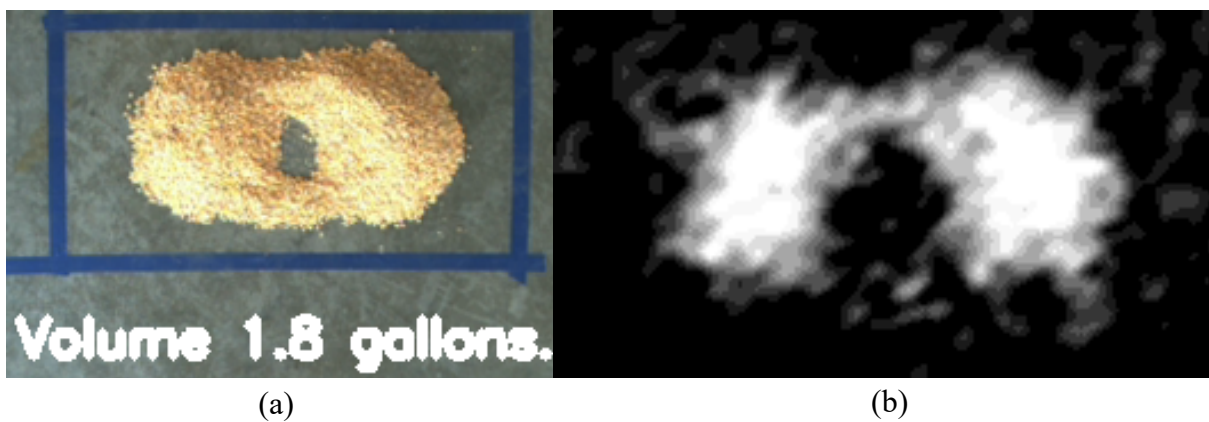


Figure 3.19. Flat surface showing approximately two gallons of grain with one crater. ((a) color image; (b) depth image).

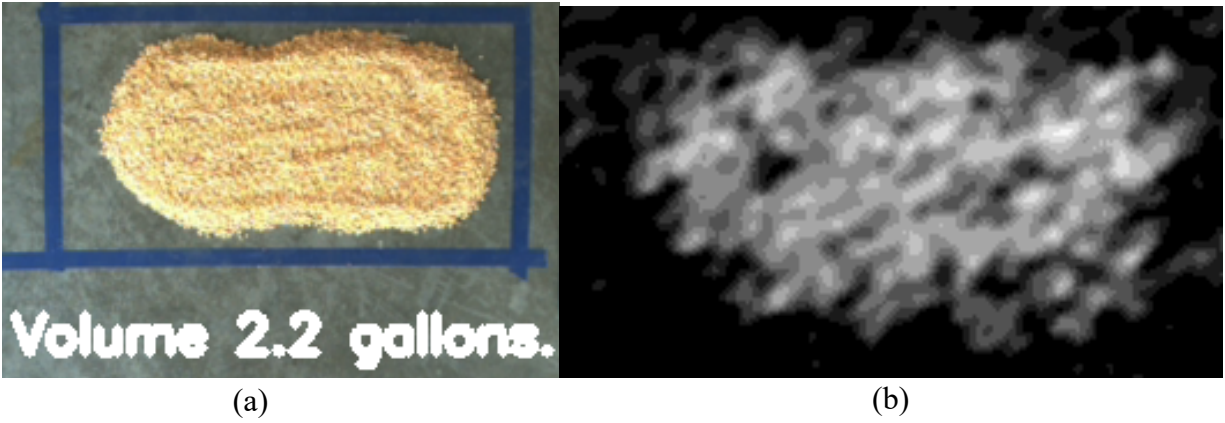


Figure 3.20. Flat surface showing approximately two gallons of grain smoothed out. ((a) color image; (b) depth image).

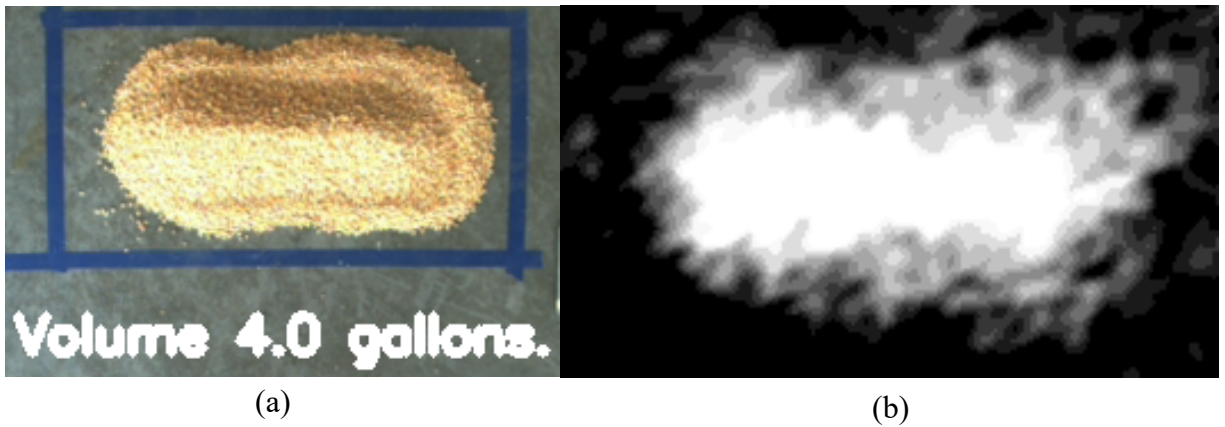


Figure 3.21. Flat surface showing approximately four gallons of grain in a pile. ((a) color image; (b) depth image).

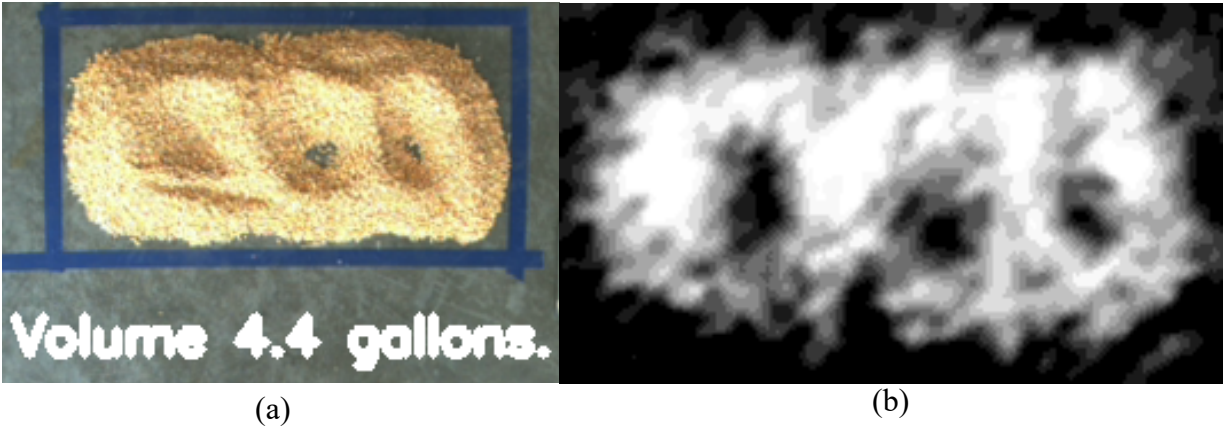


Figure 3.22. Flat surface showing approximately four gallons of grain with three craters. ((a) color image; (b) depth image).

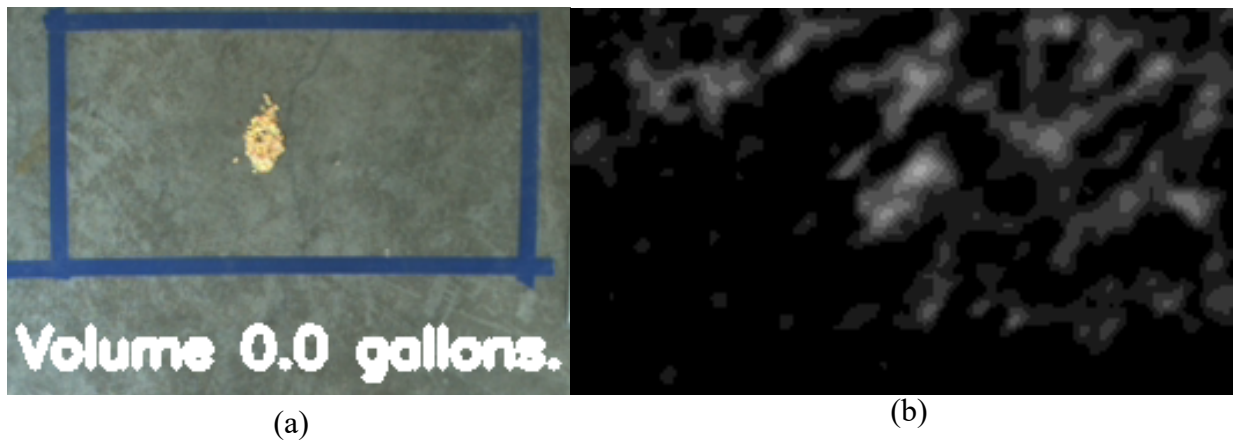


Figure 3.23. Flat surface showing a handful of corn grain tossed in the middle. ((a) color image; (b) depth image).

3.5.3 Discussion

Inconsistencies with TMR density can cause issues since the camera estimates accurate volumes which might yield inconsistent weight. A few options may help mitigate this phenomenon: if a TMR feed dispenser does not already have weight scales built in, a few feed bunk sections can be first filled with feed and then the depth cameras are used to determine the volume of the feed in those feed sections. Next, the feed from those bunk sections should be weighed to calculate the TMR density. If the TMR was mixed consistently, the density should be consistent across all feed bunks (or feed bunk sections) where the same TMR batch was used. Another option is to use the scales in the TMR feed dispenser to determine the weight of the TMR being fed to the animals

during the feeding session. The weight of all the TMR dispensed can be used with the change in volume found using the depth cameras by noting the volumes before and after the TMR is dispensed. The change in volume and the weight from the dispensing vehicle will determine the density of the TMR, which, again, should be consistent through the bunks.

Corn grain, which has relatively consistent density, better highlights the limitations of the volume measurement system. Table 3.3 sums up the limitations by showing the ranges of values observed for each scenario and the range of maximum values for the scenarios. The range can be subtracted from the maximum values to find the collective range among the same nominally sized scenarios. Scenarios 7 and 8 should have the same volume measurements, as should 9, 10, and 11; and 12 and 13. In other words, the nominally one-gallon scenarios had a range of values between 2.2 and 4.2 L (range subtracted from maximum values to find the minimum values). The nominally two-gallon scenarios had a range of values between 6.8 and 8.3 liters. The nominally four-gallon scenarios had a range of values between 14.6 and 20.0 liters. Evidently, even without changing the characteristics of the pile, there is a range of volume measurements. The range of values may be due to very slight movements of the camera or the camera stereo matching software itself. Some very slight creep in the mounting of the camera was observed throughout the day. Overall the camera crept about 9 mm (0.34 in). Even while the camera was not measuring volume and instead only measuring the raw distance values from the camera to the taped area (same as within figures 3.17 to 3.24) the camera measured 2.962414, 2.961831, 2.9638484, 2.9642437, and 2.9639204 m in successive frames. Camera raw values varied at the 0.001 m digit. The consecutive depth measurements account for 2.4 mm (2.9642437 m – 2.961831 m) of drift in depth values with no adjustments being made to the system with the exception of ambient lighting not being controlled. That .0024 m variation over an area of nominally 0.7 sq. m would explain 1.7 L difference. These measurements show a different measurement per frame when measuring distances, which would yield slightly inconsistent volume measurements because distance is a factor within the volume formula.

Another potential issue is stereo occlusion due to lumps in the TMR piles or other “obstacles.” Occlusions (Figure 3.24) occur when one lens has a view of a point of a surface in space, the but the other lens cannot view that same point due to the viewing angle being slightly different. This

occlusion issue may not be completely solvable, but it can be limited. First, the stereo camera should have as direct of a view of the feeds as possible, in other words, to have a direct “view from above.” This way, all sides of lumps, or miniature “mountains” in the TMR, can be seen from both lenses and assumptions about how much material laying behind the shadows of a “mountain” will not have to be made. Second, the TMR should be kept as flat and smooth-surfaced as possible to prevent the peaks and valleys from happening on the surface in the first place. Occlusions may also differ depending on the feed bunk style and that will determine how the surface of the TMR will lay in the bunk and perhaps determine the nature of how the cattle feed and form new peaks and valleys in the TMR surface.



Figure 3.24. An example of occlusion. The image on the left is a color image of 2 buckets of TMR. Right is the same image from depth feed showing occlusion in blue.

While the animal is eating, the pixels that cannot be identified as possible TMR pixels should be identified as foreign object pixels, such as feeding animals, hands, tools, or instruments. The foreign object pixels would not be used to calculate TMR volume and the TMR depth for those pixels could be estimated using interpolation using nearest-neighbor pixel values or those pixels could just be ignored and the average depth value substituted for those pixels until the foreign object is not within the field of view anymore. A flag should be raised for that region of the feed bunk within the graphical user interface (GUI) showing the volumes of the feed bunk regions to indicate that the volume at that moment in time may not be as accurate as when no foreign objects are within the field of view of the stereo camera. Known depth constraints can be used, such as feed should only be between certain depths. Pixels outside those depths are non-feed. This knowledge could help identify when a cow begins and finishes a feeding bout.

Finally, the camera lenses may need to be cleaned due to the likely accumulation of dirt and grime on the lenses. Many stereo cameras are of automotive grade and can withstand typical animal shed environments. From general observation, the Carnegie Robotics stereo camera works well with a small layer of dust on the lens, so the camera lenses may only need to be cleaned weekly. Clever camera mounting designs can be utilized to make cleaning the lenses less tedious, such as a system that easily raises and lowers the camera for cleaning or mounting the camera behind a glass case and then spraying the glass case with a cleaning solution.

3.6 Conclusion

The stereo camera has the potential to be very useful for researchers and producers in animal husbandry. As shown in the proof of concept, the device can determine the volume of TMR that is added or removed from a certain area of the image. The same concept can be used with individual animals to measure unique physical features of animals desirable for better agricultural production. The capabilities of the camera may also be useful for individual animal identification via techniques such as template matching, color masking, depth masking, or CNNs. Objective 1: calculate actual pixel area using a relationship with depth was completed. Objective 2: TMR and corn grain volume measurement was accomplished. Ranging errors of the camera were disappointing, but improvements can be made. Mounting of the camera can be more rigid to prevent measurement creeping. Some cows, especially lactating cows, may consume as much as 36 kg of dry matter feed per day, so with this amount of feed being consumed throughout a day the errors may be insignificant. A camera with more disparities (more pixel matches) may increase precision. Using a single instrument system to accomplish both tasks of volume measurement and animal identification with open source software can reduce the cost of precision livestock husbandry for researchers or food producers.

4. NEAR REAL-TIME GRAIN VOLUME OF LOADING VESSELS

4.1 Abstract

A variety of computer vision techniques were applied to depth and color video of a grain cart loading grain into a grain truck with the goal of measuring volume in near real time. ROS interacting with Python were the main software interfaces used. OpenCV (OpenCV, 2022), NumPy (NumPy, 2022), and Matplotlib (Matplotlib, 2022) were the main Python libraries utilized. An estimate of near real-time volume in the truck was obtained (approx. sub 3 second). Template matching was successful in finding matches for vessel-grain cart relative position as indicated in the resulting images and videos. Vessel volume was successfully estimated within 1.4 m³ for the grain transport vessel. Limitations observed include lag between the depth and color image streams, and varying template matching and grain segmentation success in dusty and changing light conditions.

4.2 Background

Masking is a computer vision strategy used to eliminate pixels of images based on certain thresholds. Thresholds can be certain color band limits in the case of RGB and HSV images, or depth limits in the case of depth images where each pixel represents a depth based on the distance from the camera lenses.

The red, green, and blue bands of the colors which combine to create the color combinations that digital screens use are scaled between 0 and 255 in OpenCV. To convert between the RGB color space to the HSV color space, each color band is scaled down to fit between 0 and 1. Value (V) is assigned the max value from the rescaled R, G, and B values. S (saturation) is calculated using the following relationships:

$$S = \left\{ \begin{array}{ll} \frac{V - \min(R, G, B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{array} \right\} \quad [4.1]$$

H (hue) is calculated using the following relationships:

$$H = \left\{ \begin{array}{ll} 60(G - B)/(V - \min(R, G, B)) & \text{if } V = R \\ 120 + 60(B - R)/(V - \min(R, G, B)) & \text{if } V = G \\ 240 + 60(R - G)/(V - \min(R, G, B)) & \text{if } V = B \\ 0 & \text{if } R = G = B \end{array} \right\} \quad [4.2]$$

if $H < 0$,

$$H = H + 360 \quad [4.3]$$

Finally, the HSV values are scaled using the following relationships:

$$\begin{aligned} V &= 255V \\ S &= 255S \\ H &= \frac{H}{2} \end{aligned} \quad [4.4]$$

Once the color space conversions are made, H is mapped to the B channel, S is mapped to G, and V is mapped to R when displayed digitally. The mapping from one color space to another causes the image to look “unnatural.” If software native to HSV color spaced were used, the images would look “natural.”

The normal vector runs in the direction perpendicular to the surface plot. The depth image can be assumed to represent a surface plot, $z=f(x, y)$. The normal vector at each pixel can be found according using the following sequence of equations:

First, the normal vector direction is calculated:

$$N = \begin{bmatrix} f_x(x_o, y_o) \\ f_y(x_o, y_o) \\ -1 \end{bmatrix} \quad [4.5]$$

Where:

$f_x(x_o, y_o)$ and $f_y(x_o, y_o)$ are the partial derivatives,

N is the normal vector direction.

Next, the normal vector magnitude is calculated:

$$\text{Magnitude } (|N|) = \sqrt{(f_x(x_o, y_o))^2 + f_y(x_o, y_o)^2 + 1^2} \quad [4.6]$$

Where:

$|N|$ is the normal vector magnitude.

Finally, the unit normal vector is calculated:

$$\hat{N} = \frac{N}{|N|} = \left\langle \frac{f_x(x_o, y_o)}{|N|}, \frac{f_y(x_o, y_o)}{|N|}, \frac{1}{|N|} \right\rangle \quad [4.7]$$

Where:

\hat{N} is the unit normal vector.

Template matching operates by taking a cut from a larger image to form a template. The template is moved over the original image, and measures how closely related the pixels in the template are to the pixels in the main image. The template is moved from left to right, top to bottom. For each pixel that the template is centered on, the corresponding resultant matrix is filled with the matching metric result. The higher the value in the case of the work in this project (sometimes for different resultant matrices the lowest value is the best match), the higher likelihood of a good match of the template. A value of “1” indicates a perfect match. The method used for this work is known as Normalized Correlation Coefficient Matching. A perfect mismatch will result in a value of “-1.” The resultant matrix can be found using the following sequence of equations:

The correlation coefficient resulting matrix formula is:

$$R_{coeff} = \sum_{x', y'} T'(x', y') \cdot I'(x + x', y + y') \quad [4.8]$$

To calculate R_{coeff} , $T'(x', y')$ and $I'(x', y')$ need to be found using the following sequence:

The template mean is subtracted from the template for each pixel:

$$T'(x', y') = T(x', y') - \frac{\sum_{x'', y''} T(x'', y'')}{(w \cdot h)} \quad [4.9]$$

The area of interest mean is subtracted from the area of interest for each pixel:

$$I'(x', y') = I(x', y') - \frac{\sum_{x'', y''} I(x'', y'')}{(w \cdot h)} \quad [4.10]$$

[4.8] is the formula for the non-normalized correlation coefficient. To become normalized, [4.8] is divided by the root of the sum of the squared pixel for each relative-mean matrix:

$$R_{coeff_normed} = \frac{\sum_{x', y'} T'(x', y') \cdot I'(x + x', y + y')}{\sqrt{\sum_{x', y'} T'(x', y')^2 \cdot \sum_{x', y'} I'(x + x', y + y')^2}} \quad [4.11]$$

As an example of the template matching process, an image matrix I is composed of the following

pixels: $\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$. The “2” is to be matched in the image. The template matrix, T, is:

$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$. The resultant matrix is a size of (W-w+1, H-h+1), with W, w, H, and h being with

width and height of I and T, respectively. Therefore, the precalculated resultant R matrix has the

shape of: $\begin{bmatrix} R & R & R & R \\ R & R & R & R \\ R & R & R & R \end{bmatrix}$. The area of the image containing the “2” will be the current area of

interest. $I_{(x+x',y+y')}$ therefore is: $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$. If the top left of the I matrix was the area of interest,

$I_{(x+x',y+y')}$ would have been $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, but for simplicity the pixels surrounding the “2” is the

area of interest. To obtain the full resultant matrix, all areas of the image would be searched. To

calculate $I'_{(x+x',y+y')}$, subtract the average of all elements of $I_{(x+x',y+y')}$ from itself: $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} -$

$\frac{10}{9} = \begin{bmatrix} -0.111 & -0.111 & -0.111 \\ -0.111 & .889 & -0.111 \\ -0.111 & -0.111 & -0.111 \end{bmatrix}$. A similar procedure is followed to find $T'_{(x',y')}$. The average

of all elements in $T_{(x',y')}$ are subtracted from itself to find $T'_{(x',y')}$: $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} - \frac{10}{9} =$

$\begin{bmatrix} -0.111 & -0.111 & -0.111 \\ -0.111 & .889 & -0.111 \\ -0.111 & -0.111 & -0.111 \end{bmatrix}$. Now the $T'_{(x',y')}$ and $I'_{(x+x',y+y')}$ matrices are plugged into the

correlation coefficient formula. To follow the formula, multiply each pixel in $T'_{(x',y')}$ with the

corresponding pixel in $I'_{(x+x',y+y')}$:

$\begin{bmatrix} -0.111 * -0.111 & -0.111 * -0.111 & -0.111 * -0.111 \\ -0.111 * -0.111 & .889 * .889 & -0.111 * -0.111 \\ -0.111 * -0.111 & -0.111 * -0.111 & -0.111 * -0.111 \end{bmatrix} = \begin{bmatrix} .0123 & .0123 & .0123 \\ .0123 & .7901 & .0123 \\ .0123 & .0123 & .0123 \end{bmatrix}$. Next,

all these matrix elements are summed up to .888. This value is the correlation coefficient for pixel

“2” in I. If the other values for R in R were calculated, no other R values would be as high as .888.

Now, to normalize R, element-wise square each pixel of I' and T,' sum all the squared elements

in each matrix, multiply the sums together, and take the square root of the product. This number is the denominator of the R_coeff_normed function within OpenCV:

$$\sqrt{\sum \begin{bmatrix} -0.111^2 & -0.111^2 & -0.111^2 \\ -0.111^2 & .889^2 & -0.111^2 \\ -0.111^2 & -0.111^2 & -0.111^2 \end{bmatrix} * \sum \begin{bmatrix} -0.111^2 & -0.111^2 & -0.111^2 \\ -0.111^2 & .889^2 & -0.111^2 \\ -0.111^2 & -0.111^2 & -0.111^2 \end{bmatrix}} = \sqrt{.7901} = .888.$$

The normalizer is .888 so .888 is divided by .888 and results in “1,” indicating a perfect template match for that area of I. Before working out the equations, it was known that the resultant should be “1” because the pixels in that area of the image and the pixels of the template are exactly the same.

Combining the known knowledge of masking, normal vectors, and template matching, a set of objectives were developed for this chapter.

4.3 Objectives

1. Isolate grain pixels in stereo vision images.
2. Measure near real-time instantaneous volume in a grain hauling vessel.
3. Develop grain vessel position tracking techniques with the same instrument used in (2).

4.4 Methods and Materials

The strategy used for volume calculation and was to first eliminate all pixels not associated with the grain, and then perform volume calculations on the remaining grain pixels. The routine established to find the grain pixels involved some intermediary steps including:

1. Masking based on depth.
2. Masking based on HSV.
3. Masking based on color.
4. Masking based on normal vectors of the depth image.

Vessel tracking was accomplished using masking based on depth and then using template matching to find the desired stake pockets of the grain vessel.

A New Holland T8 tractor was used to unload grain into a grain truck in the same manner as described in Chapter 3. Unloading data was recorded in ROS .bag files and replayed for the

computer vision development. Two unloading passes and one pass of a full truck were successfully recorded. Figure 4.1 shows the data recording pipeline for the outdoor testing.

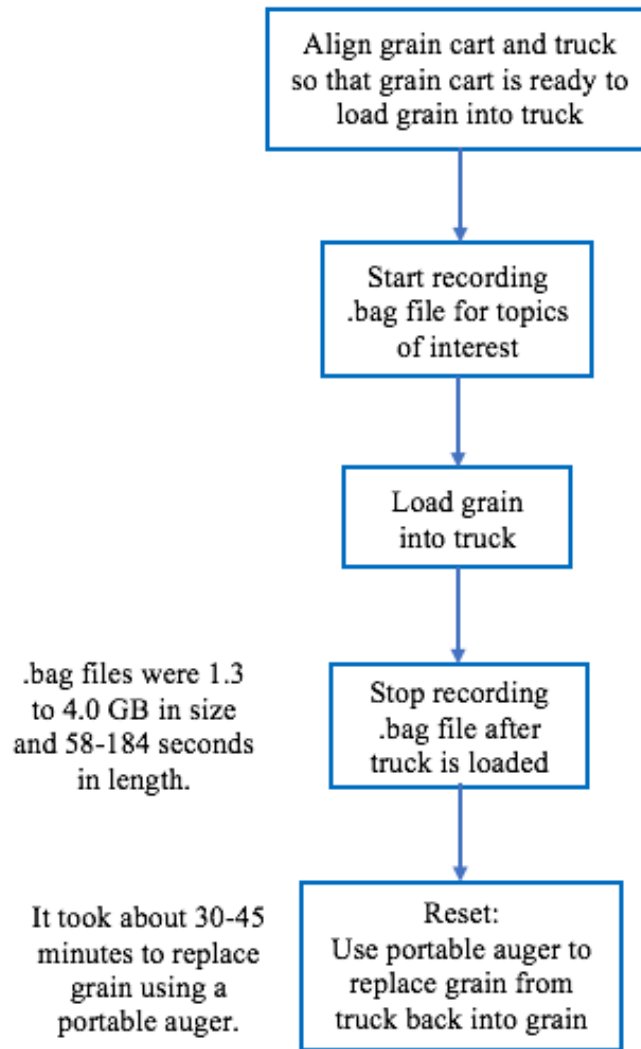


Figure 4.1. Data collection pipeline for grain truck loading.

4.4.1 Template matching

To accomplish template matching, the color images were first separated into red, green, and blue channels. Stake pockets were chosen as templates because they exhibit little change from frame to frame as the truck moves. Templates of stake pockets were taken from the red channel, and the matches for the stake pockets were found in the red image channel. The depth image stream was used to mask out the pixels from the red color band channel. The depth pixels that were not

between 1.95 and 2.10 m from the camera were those masked; the remaining were mostly pixels containing the top of the grain holding container (the vessel rim) which contains the stake pockets. Next, single-row wide strips were trimmed from the red channel image stream to form the templates. The template strip was cut from a stake pocket from one edge of the pocket, to the hollow middle, to the other edge of the pocket (Figure 4.2).

Figure 4.2. Actual stake pocket strip used for template matching.

The vertical stake pocket was used to track the relative position between the auger chute and the horizontal position of the vessel. The horizontal stake pockets (along the bottom horizontal edge of the top of the vessel wall) were template matched for vessel alignment controls (not implemented here). The horizontal stake pockets can aid in insurance that the grain cart and grain vessel are moving in parallel fashion in relation to each other. Multiple template strips are needed to find an entire stake pocket. Some strips may be false matches, but the areas where the most template strips are able to be matched are most likely the stake pockets of interest. A snippet of Sean Penn's Stack Overflow post code was used for a portion of the large stake pocket template blob detection work (Penn, 2017). The stake pockets are matched and marked in green pixels, and the depth mask is overlaid. Finally, a threshold of finding only the largest green areas is applied for the horizontal stake pockets in which multiple matches is desired (Figure 4.3). For the vertical stake pockets, the single largest stake pocket is found. The x pixel coordinate of the vertical pixel is used to calculate the relative position of the chute to the truck during an unloading pass (Figure 4.4). The x pixel coordinate is rescaled to be between 1 and 100 with 1 meaning the beginning of the pass the 100 being the end of the pass.

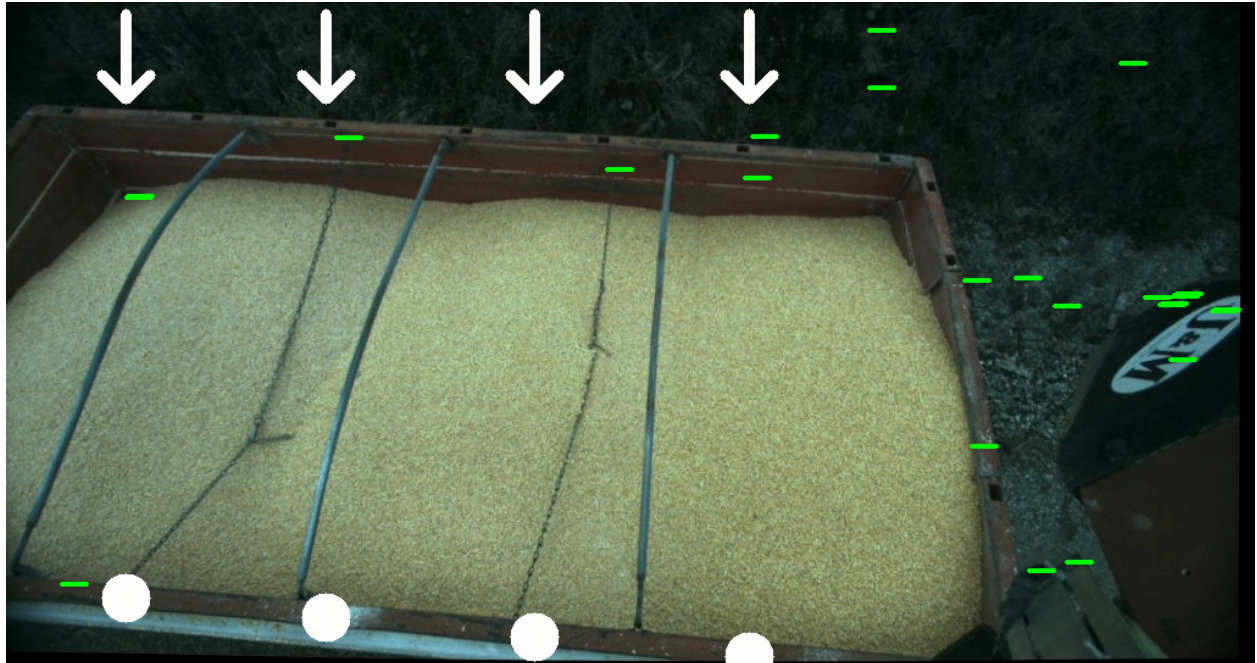


Figure 4.3. Arrows pointing to the near-edge stake pockets of the passing truck.

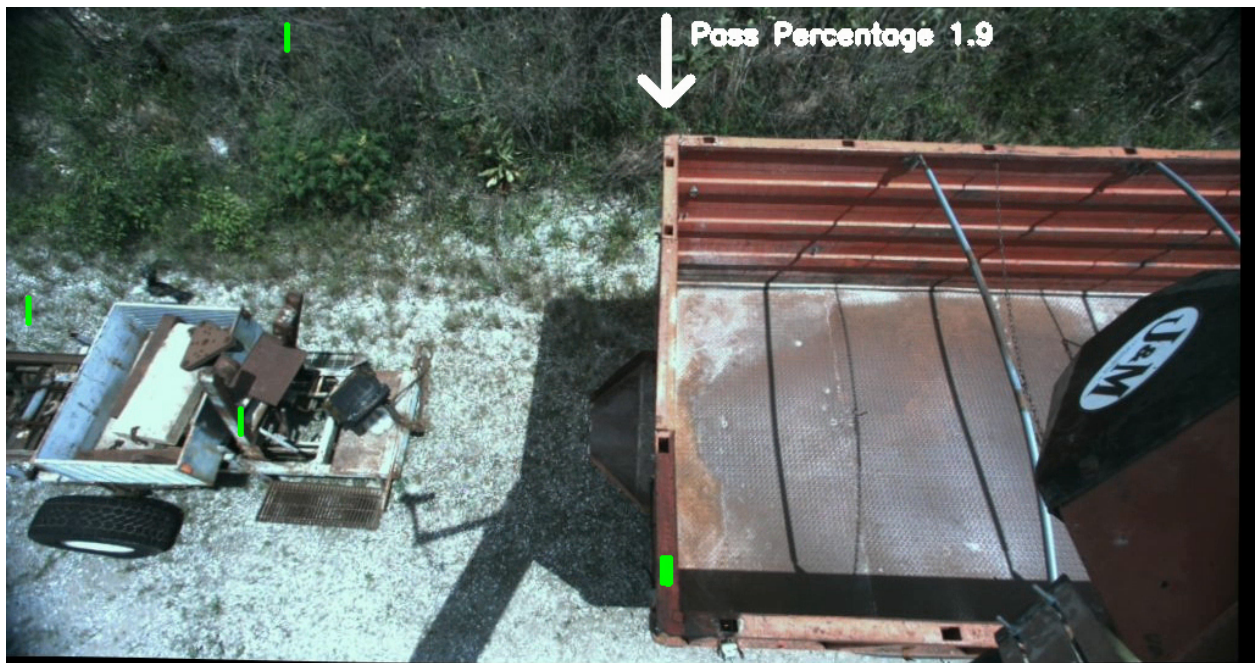


Figure 4.4. Relative pass percentage being calculated from the vertical stake pocket.

4.4.2 Depth masking

Masking based on depth is a prime advantage to using a relatively high-density stereo camera for computer vision work. The depth values can be used to mask away areas where it is known to not have pixels of interest due to the range of depth values they may contain. Depth was used in a couple of phases for masking. In one phase, it was used to eliminate all areas of the images that were below the floor height of the truck grain bed. The next phase was to eliminate all depths of the bed rails and higher. This method eliminated the vast majority of pixels not associated with the grain, but some wall pixels still remained. Some pixels from the woods in the background also still remained, and when driving the grain cart some equipment that was passed would still register pixels within the grain pixel range. Figures 4.5, 4.6, and 4.7 show the original color image before masking, the ground masking applied to the color image, and rail masking applied to the color image. The images were taken later in the grain loading process as the dust cleared. The same period of timestamps will also be used for the rest of the mask images discussed later in this section. The masking is also applied to the depth image which is where the volume calculations come from which are explained later in this section. For illustration purposed the masks are applied to the color image and is also used as a “visual” truth to verify that once all masks are applied that only the grain pixels remain. Because the depth image is the original source of the masks, the non-returnable pixels due to occlusion or dust also act as a mask, as seen in the color ground mask image. The pixels on the ground surrounding the truck are masked but so are the pixels surrounding the cross members of the truck bed due to occlusion.



Figure 4.5. Original color image of truck with no computer vision techniques applied.

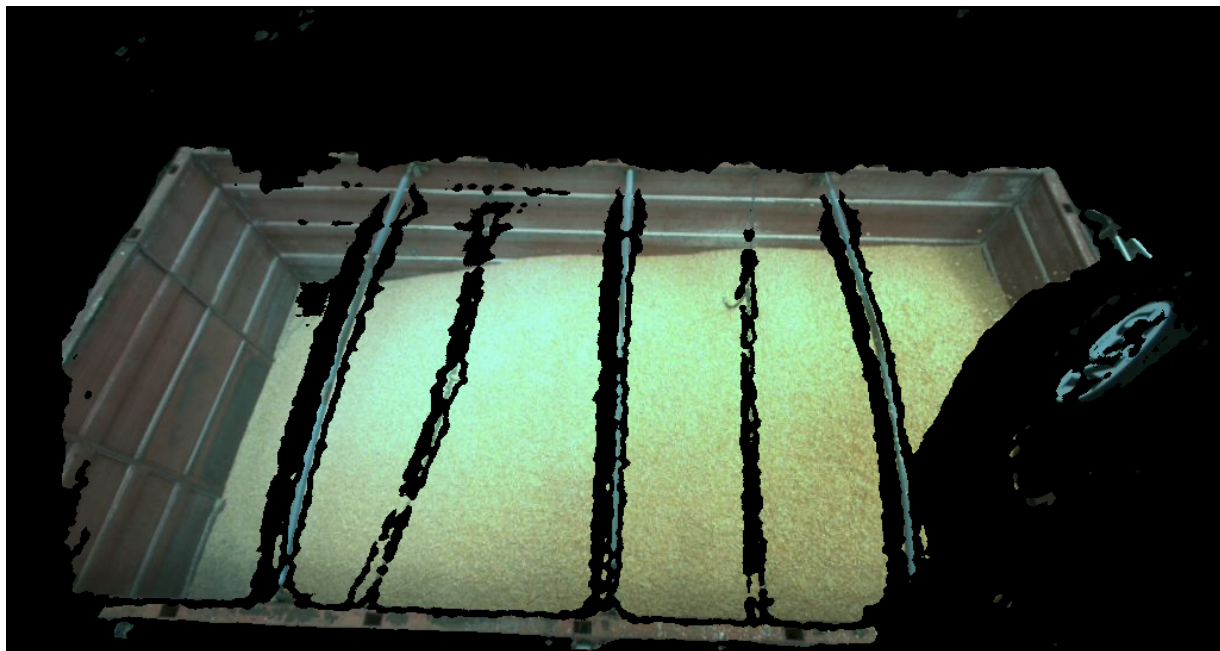


Figure 4.6. Color image of truck with surrounding ground pixels masked out.



Figure 4.7. Color image of truck with bed rails and crossmembers masked out.

4.4.3 HSV masking

The truck used for testing was a mostly red truck parked on a white gravel surface with light-density wooded trees in the background. Those contrasting in color items were exploited for masking, both in the regular RGB color space and in the HSV color space. HSV stands for hue, saturation, and value. The RGB images were converted into HSV using `cv.bgrtoHSV()` command within OpenCV.

To explore the masking options, the video frame was downloaded from the .bag files from ROS. The main reason for downloading the image was to implement adjustable track bars (Figure 4.8). Adjustable track bars may be able to implement with real-time images streaming from ROS, but time constraints prohibited further investigation. The track bars allowed for the adjustment of maximum and minimum hue, saturation, and value ranges to show in the images, with pixels not meeting the requirements automatically masked out according to the track bar. The track bar code was originally developed by Nathan Lam for HSV images (Lam, 2020). The code was modified to include RGB thresholding as well.

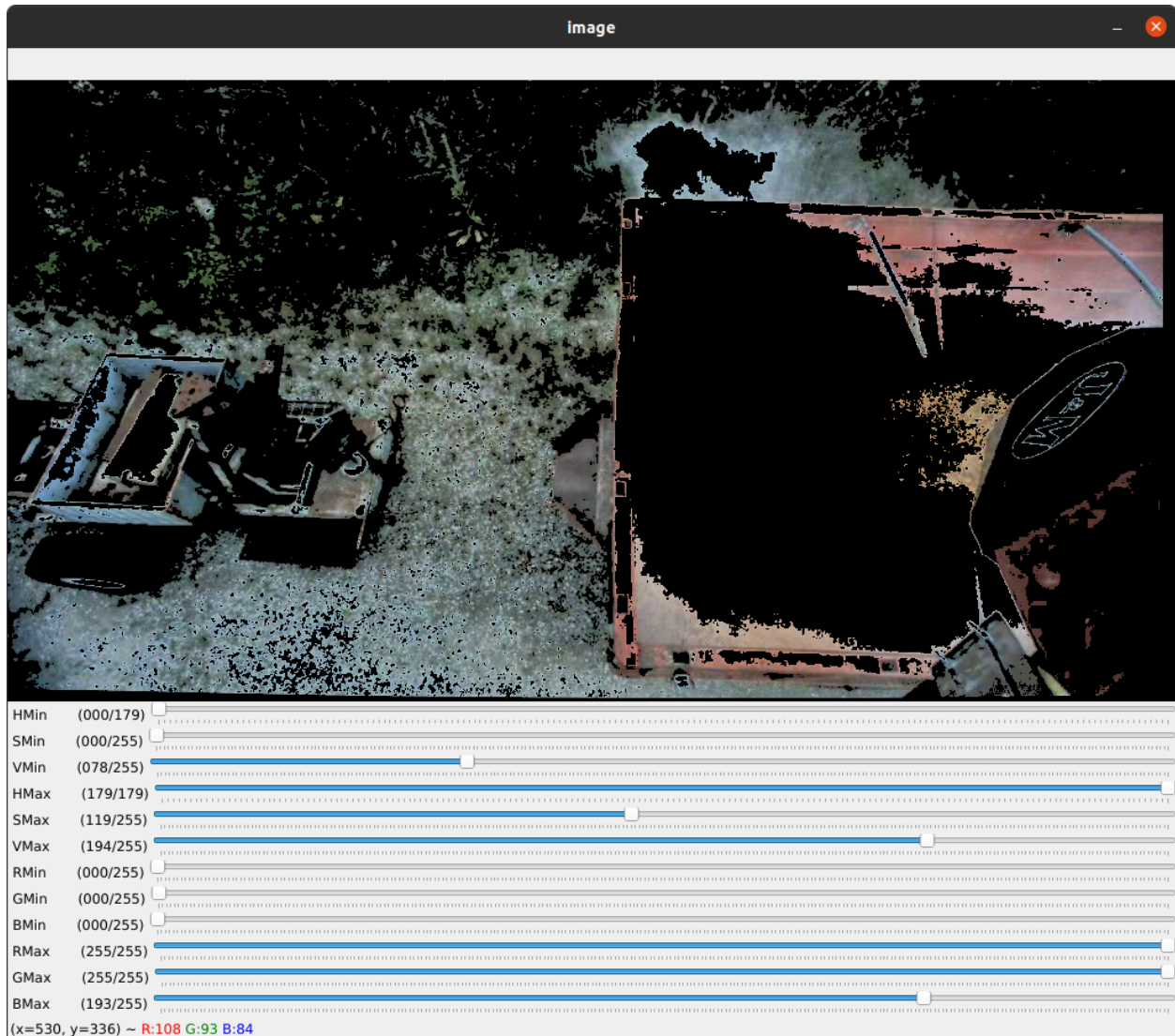


Figure 4.8. Trackbars used to adjust maximum and minimum HSV and RGB values of an image.

Suitable values were chosen to eliminate as much background as possible and leave the grain behind. Leaving only grain was not possible using HSV masking method, but this method did come close in leaving mostly only grain. Figure 4.9 shows the converted image that has been converted to an HSV image. Figure 4.10 shows masking that has been applied to the HSV color spaces. This is another method that worked relatively well but did not eliminate all background pixels. This method also appears to have removed some grain pixels from the bottom edge of the truck bed. There is a tradeoff between masking unwanted pixels with the risk of masking wanted pixels. The image appears to have left most of the grain pixels, so the tradeoff was accepted.

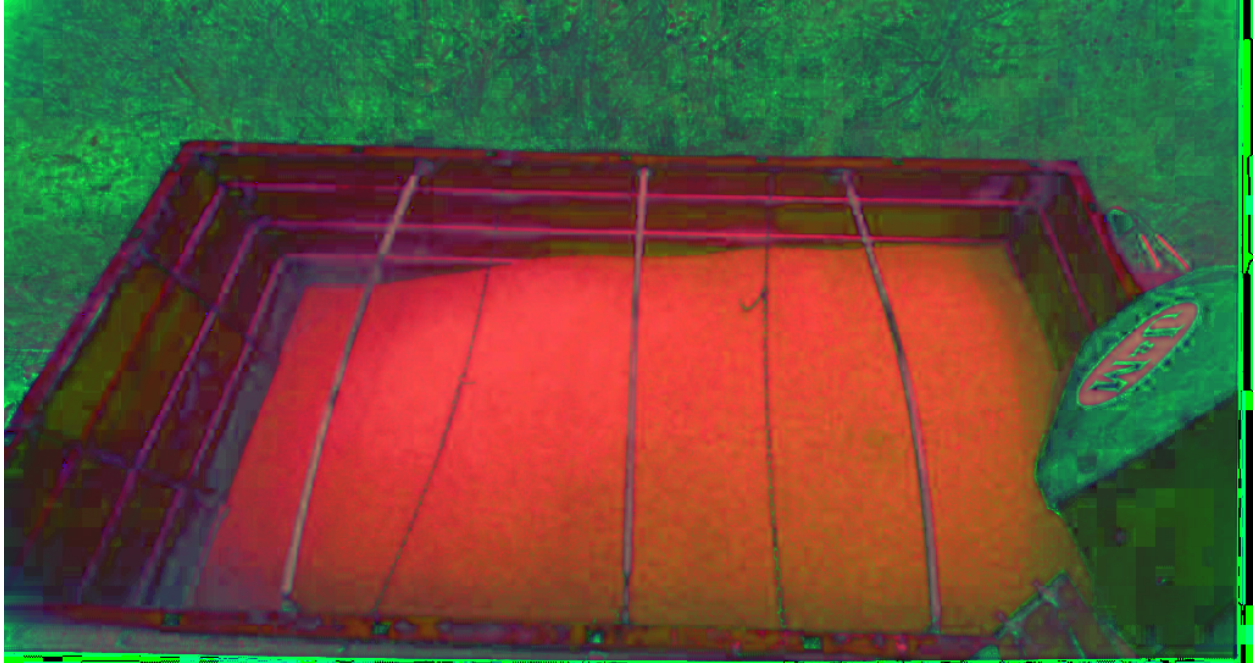


Figure 4.9. HSV color space image converted from the color image.



Figure 4.10. Masked color image with the mask carefully chosen from ranges of color spaces within the HSV image.

4.4.4 Color masking

In as similar procedure to HSV masking, color (RGB) masking was implemented. The color images were separated into individual R, G, and B image streams. A track bar system was implemented, and the allowable R, G, and B reflection ranges were adjusted to allow as little background noise as possible. The truck's paint condition was very poor, so the red paint was not as useful of a masking source as previously hoped. However, it was able to further reduce background noise and therefore was implemented. The mask appears to have similar performance of the HSV mask, but does mask different areas as the light and image objects change during grain cart movement. Figure 4.11 shows the color image masked from carefully chosen RGB values.

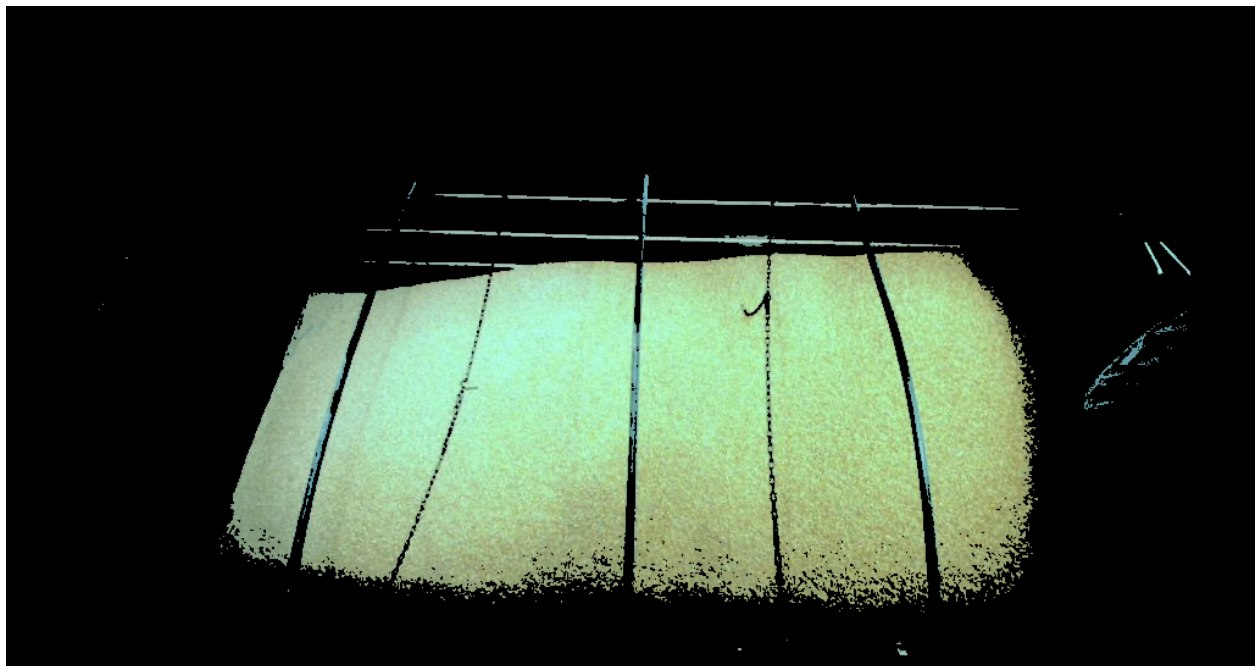


Figure 4.11. Color image masked from carefully chosen RGB values.

4.4.5 Normal Vector masking

Normal vector masking was another strategy to remove background from the grain pixels. In general, the previous four methods just discussed did a good overall job of removing most noise with the exception the grain truck walls not yet covered by grain. However, each wall can be thought of as a relatively flat plane surface, and each plane lays in a different orientation compared to the grain that is being loaded. With this knowledge in mind, the normal vectors coming off of the depth image can be calculated for each pixel that has a depth value associated with it. The

normal vectors for the walls should be drastically different compared to the normal vectors for the grain.

The normal vector in the context of this work is the vector that is perpendicular to the surface being conveyed by the depth image. The depth image, also sometimes referred to as a range image, can represent a three-dimensional surface because it represents points measured in three-dimensional space. The depth images are essentially surface plots, with each pixel representing length (x), width (y), and depth (z). The z values are represented by the value of the pixel in meters. The x and y value are determined by their location within the Cartesian coordinates represented by the depth image matrix. The normal vectors can be used to make assumptions about objects in the depth images. It was hypothesized that similar planes within a depth image should have similar normal vectors. To test this claim, the depth images were used to calculate the normal unit vectors. A similar process for a different application was posted by Tim Day in the Stack Overflow web forum (Day, 2016). The NumPy *gradient()* function was used to calculate the differentials $f_x(x_o, y_o)$ and $f_y(x_o, y_o)$. Using this function on the single dimension depth matrix (single dimension matrix whose values represent depth in space, so it is a single dimension matrix representing three-dimensional space) produces two gradient matrices of the same dimension and size of the depth image, a differential in the y direction and a differential in the x direction for each pixel. Next, a matrix of “ones”, which produces a matrix the same dimension as the other image matrices, but each value is only one. *Numpy.linalg.norm()* is then used to normalize the vectors represented by the matrices by taking the square root of the sum of squares of the differentials and the ones. This function yields a matrix with each pixel representing the magnitude of the vectors represented by combining the differential matrices and the “ones” matrix. Next, the differential matrices and “ones” matrix were divided by the magnitude matrix that was just created in a pixelwise fashion, creating a new three-dimensional unit normal vector matrix \hat{N} . Each channel of this matrix represents a different component of the unit normal vector. When this matrix is shown in image form in OpenCV, the red channel represents the first component $\frac{f_x(x_o, y_o)}{|N|}$, the green channel represents the second component $\frac{f_y(x_o, y_o)}{|N|}$, and the blue channel is represented by $\frac{1}{|N|}$. When the code was implanted on the truck images, the walls of the trucks did appear in different colors. See Figure 4.12 for the normal vector image and Figure 4.13 for the color image masked using normal vector

values. The normal vector image has been blurred using *cv2.GaussianBlur()*. Code implemented is shown in Appendix 6 and also uploaded to the Purdue University Research Repository (Rogers and Buckmaster, 2022).

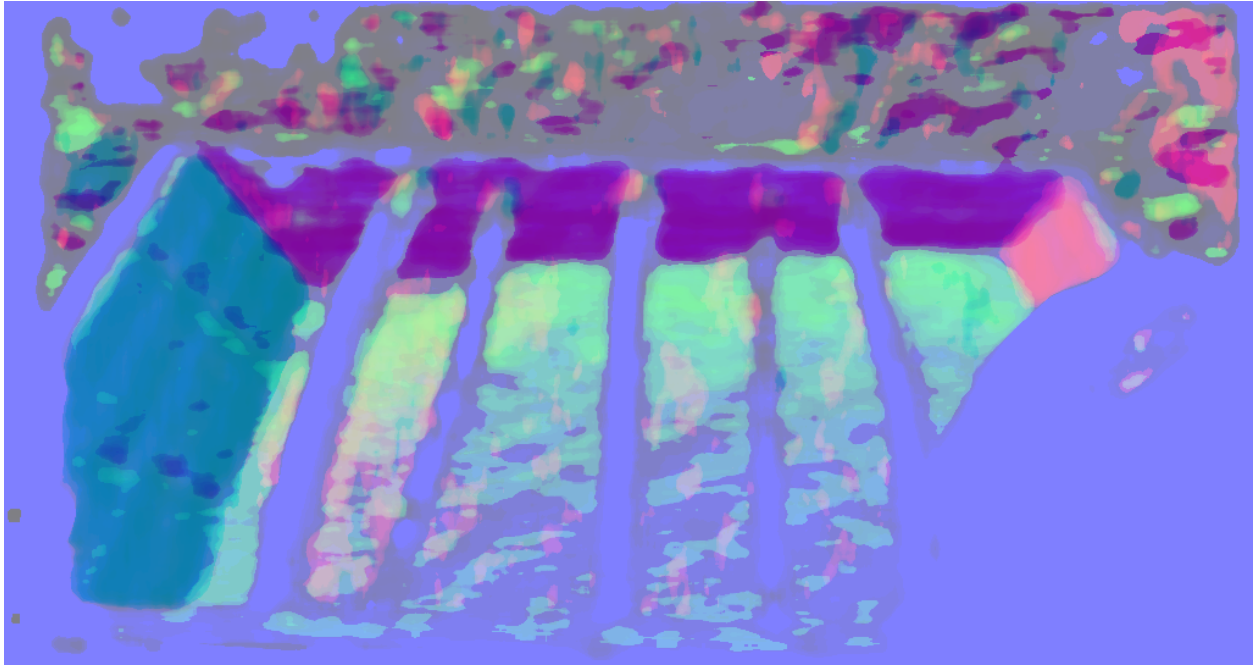


Figure 4.12. Depth image that has been converted to represent normal vectors coming off of the surface formed by the depth pixels.



Figure 4.13. Color image masked by the normal vector mask of the depth image.

4.4.6 Combining all masking strategies to the depth stream

The previous masking strategies were all combined into a single video stream of images. This stream of images should only show grain pixels, from which volume can be calculated.

Figure 4.14 and 4.15 show all masks and all but the normal vector masks being applied to the color image stream. The all mask image combines the RGB, HSV, ground, rail, and normal masks into a single mask to apply to the color image and the depth image for volume calculations. The combined masks of all masks except the normal mask is also pictured as demonstration of how the quality of mask seems to change when the normal vector mask is added. Without the normal vector mask, a very good view of the grain is present but at the expense of wall pixels remaining. The effect of the wall pixels will be demonstrated in the volume calculation section, where the volume is calculated for both the all mask situation and the all mask except normal vector situation.



Figure 4.14. All masked were applied to this color image.

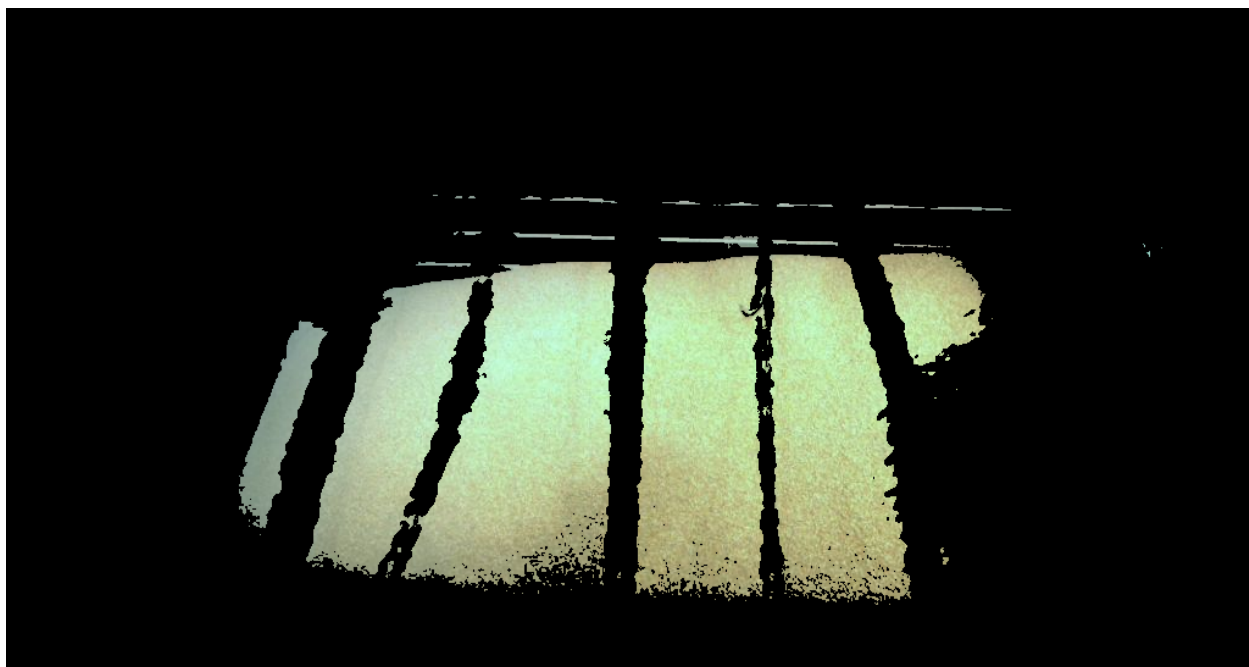


Figure 4.15. All masks except the normal vector mask have been applied to the normal vector stream.

4.4.7 Volume Calculation

With all masks combined, only the grain pixels remained. The remaining pixels needed to be corrected for the tilt of the camera. Tilt was corrected by reading pixel values of the rail pixels in the top and bottom of the left vertical rail of the truck bed. The top corner had a depth of 3.043 m and the bottom corner had a value of 2.171. These two points were spread over a y-pixel length of 392 pixels. Slope of the tilt is the difference of the pixel value over the number of pixels in the direction of change, which in this case of -.002806 m/pixel. The depth pixels were corrected for slope in relation to the bottom left bed rail corner, which had a y-pixel coordinate of 509. Therefore, the equation for corrected for tilt is:

$$\text{new depth} = \text{old depth} + \text{slope} * (\text{delta from pixel 509}) \quad [4.12]$$

Where

new depth is the depth corrected for tilt,

old depth is the non-adjusted depth that still has tilt, and

slope is the change of pixel values over the number of pixels in the direction of change.

After the equation was applied, the pixels showing grain at a slope due to camera slope no longer had sloped values due to the tilted camera. Now, the equation for volume which was developed in Chapter 3 was applied. In that chapter, volume was calculated for a pile of biological material not held in place by a container with a known volume. For the truck application, however, the truck has a constant filling area. In other words, the perimeter of the walls are the same from the bed floor to the top of the bed rails. Therefore, the pixel area calibration procedure was not needed for the truck volume application and was not used. A key assumption of the following volume calculation is that each pixel represents a volume, as discussed in chapter 4. Also it was assumed that the average pixel value observed in the truck bed was a very close approximation to the actual average depth of grain in the truck bed. This assumption is necessary because not all grain pixels in the bed could be imaged due to the nature of the masking techniques as explained in the previous parts of this chapter. Another assumption that was made is that only pixels remaining with depth values are indeed grain pixels in the bed of the truck. Visually judging the masked images, this assumption mostly holds true with the exception of some background pixels when the truck is being moved into loading position. After assumptions were made, the average value of the pixels needed to be found. To find the average pixel value, it should first be noted that the masking

process changed the “NaN” and masking pixels into pixels holding a value of “0.0,” which would drastically change the average depth value. This issue was fixed using the NumPy and Python commands `adjusted[adjusted == 0.0] = numpy.nan` to convert the pixels in the depth image adjusted for tilt called “*adjusted*” back from “0.0” back into “NaN” values. “NaN” pixels no longer have numerical data in them. The average value of the remaining pixels was calculated using the command `adjusted = numpy.nanmean(adjusted)` which finds the average of all pixels while ignoring the NaN pixels. Once the average value of the depth pixels was known, they were multiplied by the floor area of the truck to find the real-time volume of the grain in the truck. Volume was found using the equation:

$$\text{Grain volume} = \text{Floor area} * \text{average pixel height} \quad [4.13]$$

Figure 4.16 shows the volume detection system calculating the volume of the grain while simultaneously streaming image data from ROS. Figure 4.16 is the volume measured using all masks. Figure 17 shows the volume of measured using all masks with the exception of the normal vector mask. Approximately 1 cubic meter is the difference between the two methods.



Figure 4.16. Calculated volume of the grain truck using all masks discussed.



Figure 4.17. Calculated volume of grain truck using all masks except the normal vector mask.

A collection of videos has been curated and is available for viewing on YouTube (links in Appendix 11) and available for download from the Purdue Research Repository (Rogers and Buckmaster, 2022). Videos include pixels showing occlusion and dust in blue, a lag comparison showing lag between the depth and color image streams, near real-time volume measurement (sub 3 second), truck-grain cart relative position tracking, and multiple template matches of the truck stake pockets.

4.5 Discussion

The perimeter of the truck bed forms a rectangular area of 10.96 m^2 which can be rounded up to 11 m^2 for practical purposes. Manual measuring tape measurements showed the depth of the bed from the floor to the top rails to be 51 inches, or 1.3 meters. Therefore, a maximum volume of roughly 14 m^3 is to be expected, but from the color pictures of the image it's obvious that the truck was not nearly completely filled and both volume measurements appearing close to what is in the truck. The truck wall has the floor, the ridges, and then the top rail. As seen in the photos, the grain has not reached over the middle ridge anywhere in the photo, so an estimate less than 7 m^3 should be expected. It can also be deduced that the front half of the truck mostly made it to halfway grain

height and back half sloped from about 0 meters depth to about half depth capacity. Approximately half the truck is at half capacity and half the truck is at a quarter capacity. 25% of 7 m³ and 50% of 7 m³ add up to about 5.25 m³. It can reasonably be concluded that the stereo camera provided reasonable estimates of the volume in the grain bed (6.6 m³).

Results in the form of videos recorded from the .bag files replaying, and the computer vision techniques being applied are available as a component of this dissertation. The videos are housed in the Purdue University Purr Library (Rogers and Buckmaster, 2022). Videos are provided to aid in the evaluation of the system developed in this dissertation.

ROS (*.bag) files collected tended to be in the 1-3 GB size in range, which creates transfer and storage difficulties. Data from additional unloading passes were collected, but to save space, the depth values were recorded in the /multisense/openni_depth/compressed image topic; unfortunately and later determined, these could not be decompressed and rescaled to present depth in meters. Collection of image video in .bag file form and running agricultural equipment simultaneously did prove challenging, and perhaps that added stress contributed to running the cart PTO at an excessive speed in one instance resulting in a broken PTO shaft.

4.5.1 Limitations

Inconsistency in the image streams is the enemy of computer vision projects. Inconsistencies arise from the nature of working with outdoor power equipment conveying biological material. The most immediate apparent issue is dust. Dust blocks the field of view of the depth camera rather easily. Dust blocks both the corn grain piling in the truck, but it also blocks the background elements, such as the truck bed features. Figure 4.18 shows a color image of loading grain and dust plumes forming. Figure 4.19 shows a corresponding depth image with “NAN” pixels caused by either dust or occlusion colored blue for illustrative purposes.



Figure 4.18. Dust plumes formed when loading grain.

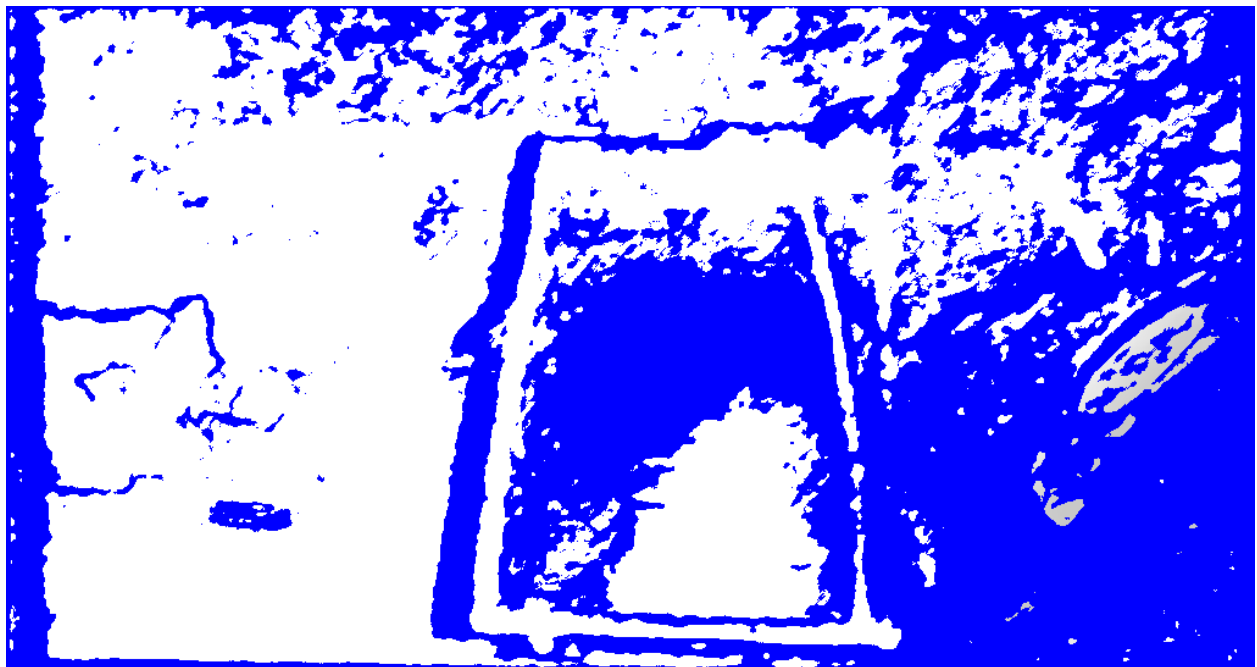


Figure 4.19. Dust and occlusion shown in blue.

Unfortunately, the normal vector method removed a fair amount of grain pixels at the expense of removing the truck wall pixels as intended (figure 4.13). A few pixels other than grain pixels also remain, but when all masks are applied to the image those pixels will disappear behind the masks. A good overall profile of the grain is still present, but it appears that the left side of the grain where the slope of the grain is relatively lower is missing more pixels than the right side of the bed (figure 4.14). When averaging of the grain pixels is used as a component of the grain volume calculation, the skew of missing pixels coming from the lower depth pixels will in turn skew the volume estimate to be below the actual volume. However, if a better job of filling the truck is done where the slope of grain is reduced and the truck filled closer to capacity, the volume estimation should be better due to the remaining grain pixels representing closer to the actual value of all grain pixels. Also of note, if the main goal of depth imaging of grain pixels is to avoid grain spillage, there are still plenty of pixels available to alert a user that the height of grain is approaching the maximum height that the truck can handle, if such a system is implemented. Moreover, the normal vector masking technique may not be needed in all situations. Its primary intention is to remove pixels that are observed on the wall left over when the other masking methods are used. Those pixels can be ignored, which will alter the volume estimate somewhat, but as the grain pixels cover the walls, the wall pixels will disappear anyway, solving the original wall pixel problem.

The lack of image stabilization is also apparent in the videos (Appendix 11) (Rogers and Buckmaster, 2022). A vibrating image, given that the camera is still able to process clearly, should not affect the computer vision quality. However, a human being using a graphical interface may experience annoyances from looking at an unstable image stream. Some image stabilization techniques were looked into for this project, but they require saving points of interest of previous frames and current frames simultaneously to correct for differences in frame orientation. Saving data from previous frames as the next frame is streaming within ROS is not a straightforward project and were not implemented for this project. Image stabilization was experimented with using a video stream downloaded from ROS due to the inability to program the stabilization within ROS. However, the results were not encouraging. Future implementation may be possible by using a gyroscope or accelerometer, which are both built into the Multisense S21 to approximate real-time movement simultaneously with the image streams and form a correction algorithm to smooth the image frames.

Image lagging, especially after computer vision techniques were applied, was apparent between the two image streams, depth and color. Figure 4.20 showcases the phenomenon where the depth masking system on the left is still masking out the crossbars of the truck bed but is applying the masking to the gravel, because the depth data being used for the masking is lagging behind the color image. Appendix 11 and the data repository (Rogers and Buckmaster, 2022) have videos the research videos that show the lag.

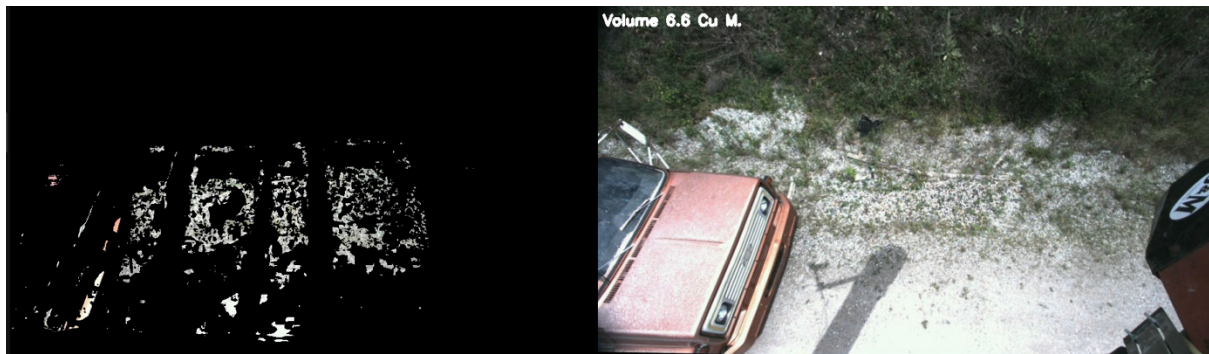


Figure 4.20. The depth mask on the left is lagging behind the color image stream on the right.

Another area of concern is the inconsistency of the colors used for masking. If the truck were a solid color, masking based on the truck's color would be much easier. Consistent grain truck colors may be less of an issue for most grain producers, as they tend to use semi-trailers or unibody trucks that are generally newer and therefore have better paint than what was used for this project.

Perhaps a very simple way of avoiding the automatic computer vision masking issue altogether would be for a user to manually draw a bounding box template of the area within the image frame where the grain is expected to be filled. The camera would then only use the pixels for volume calculation that are specified by the user. This option is also very practical, as it could be used for rapid testing procedures to verify that a camera system is capable of measuring grain in a variety of field conditions without having to worry about having the more complicated computer vision aspects working.

4.6 Conclusions

Grain pixels of a loading grain transportation vessel were successfully isolated in accordance with (1). Near real-time (approx. sub 3 second) volume of the loading vessel was calculated (2). Finally, feature tracking was accomplished (3). While limitations, primarily in the form of dust, changing light conditions, and video lagging were encountered, the proof of concept of using a stereo camera for the purpose of volume detection and vehicle guidance in agricultural working conditions was accomplished.

5. OVERALL CONCLUSIONS AND RECOMMENDATIONS

As noted in early sections, this work focused on comparing technologies (LiDAR and stereo vision) to measure volumes of biological materials in different realistic and fabricated conditions. To accomplish this work required several engineering (including software) contributions prior to the “science” aspects. Scientific contributions include the specific and repeatable methods of evaluating the depth instruments and the mathematical formulas developed or used for the volume measurement of biological materials and template matching of passing vehicle features. Ideas for future work are discussed for chapters 2-4 and include ideas for dust chamber test standards, enhanced feed and animal tracking, and mitigation of the template matching shortcomings for the grain truck.

5.1 Engineering Contributions

Engineering and system design contributions include the specific use of tools to accomplish the data collection and evaluation methods of the computer vision systems used for the biological material volume measurement. Specifically, to our knowledge, this work has accomplished the following for the first time:

- A data collection pipeline has been developed for agricultural engineering applications that fuses ROS, Python, and image and perception systems.
- A point-cloud and depth-image post processing pipeline was developed with Python and ROS.
- A physical method for testing depth perception systems in adverse agricultural conditions was developed.
- A method for measuring volume in near real-time of biological materials in feed bunk or vehicular settings was developed.

5.2 Scientific Contributions

Scientific contributions include the specific and repeatable methods of evaluating the depth instruments and the mathematical formulas developed or used for the volume measurement of

biological materials and template matching of passing vehicle features. Specific new information added to the literature includes

- Light Detection And Ranging (LiDAR) and stereo vision instruments were evaluated in a repeatable manner consistent with the scientific method. Stereo vision was found to be better in the repeated dust plume conditions for the context of volume measurement.
- The limits of template-matching computer vision algorithms were evaluated for moving outdoor agricultural vehicles.
- Mathematical relationships were scientifically evaluated to measure volume of biological materials in feed bunk and vehicular settings.
- An open-source repository has been made available in the Purdue Research Repository (Rogers and Buckmaster, 2022) for further research and evaluation by anyone interested in the dust chamber, feed bunk, or vehicle grain unloading datasets.

5.3 Recommendations for Future Work

As autonomous and driver assistance systems become more prevalent in agricultural applications, the ability to sense and perceive in adverse conditions is paramount. Further development and standardization of testing procedures of sensing and perceiving instruments for specific agricultural conditions is needed. A standard testing procedure and rating system for the different classes of perception systems (LiDAR, SONAR, stereo vision, etc.) should be developed for agricultural conditions. Those conditions include vibration, varied lighting, dust, and precipitation.

There are many applications of image processing that includes depth measurement in livestock systems. Autonomous TMR feeding could sense and perceive paths using the stereo camera, but also measure the feed needed and measure the feed already fed to cows. Cattle tracking, both in livestock barns and in pastures can be accomplished using unique identifying markers placed on each animal. Depth and color image streams can be combined to calculate body indices which may indicate health or milk production.

Beyond livestock, portion control throughout the food value chain can be accomplished through real-time volume estimation, from the manufacturing phase to food dispensing for fast-food, food lines, or vending systems. Outside edible food systems, cropping system applications include the

measurement of fertilizer materials, seeding materials, or harvested materials other than corn grain. Yield monitors on combines or other harvest equipment could be constantly calibrated by measuring the volume of material entering the hopper.

The images and videos provided by this work show promise in using a stereo camera for guidance and near real-time volume measurement. Stronger hardware to handle simultaneous depth and color image streams and complex computer vision algorithms will improve the lag symptoms experienced by the system developed in this work. Trucks with clearly defined, uniform colors would be easier to segment away from the corn grain compared to the truck used for this work. Moreover, standard template matching can be applied to multiple different trucks. Standard templates, such as standardized numbers and letters, can be placed in strategic areas on the truck surface that can be template-matched and used for guidance. In addition, radio frequency identification (RFID) tags or similar technology can be applied to the grain vessels to further enhance the perception system for automatic grain unloading systems. Dust will be nearly non-existent in the grain cart grain compartment where the grain is draining and moving into the truck bed. A sensing and perception system focused on the grain cart grain compartment instead of or as a complement to the grain truck would help with the grain volume estimates of the grain moving into the grain truck. A stereo camera could measure the volume leaving the grain cart in real-time and the amount of grain leaving the grain cart should roughly be the same as the volume entering the grain truck. Finally, knowledge of the grain unloading system such as the flow rate of the grain travel may help with depth image frames in the grain truck where dust covers most of the field of view. The last reliable depth image frame not affected by dust can be used with the flow rate to estimate the current grain volume and height in relation to the wall height.

APPENDIX 1. GRAIN VOLUME MEASUREMENT PYTHON CODE FOR FEED BUNKS

```
#!/usr/bin/env python3

#import modules//
import rospy
import cv2
import numpy
from cv_bridge import CvBridge

#import the message types from ROS//
#rostopic info shows message like Type: sensor_msgs/Image,//
#and we enter it like shown below//
from sensor_msgs.msg import Image
bridge = CvBridge()

def depth_callback(depth_msg):    #for left color image//
    global cv_image
    global cropped_image
    global adm_image
    global box_image
    global pile_image
    cv_image = bridge.imgmsg_to_cv2(depth_msg, desired_encoding="bgr8")
    adm_image = cv_image[83:180, 437:633]
    box_image = cv_image[61:116, 482:598]
    pile_image = cv_image[99:145, 498:537]

def depth_callback2(depth_msg2):    #for the depth image//

    cv_image2 = bridge.imgmsg_to_cv2(depth_msg2, desired_encoding="32FC1")
    #passthrough also works besides 32fc1
    # display the depth

    cv2.imshow("DEPTH", cv_image2)

    cropped_depth = cv_image2[30:240, 390:660]
    pile_depth = cv_image2[99:145, 498:537]
    adm_depth = cv_image2[83:180, 437:633]
    box_depth = cv_image2[61:116, 482:598]
    adm_rawdepth = numpy.nanmean(adm_depth)
    box_rawdepth = numpy.nanmean(box_depth)

    adm_corn_depth = 2.97213164 - adm_depth
    cropped_height = 3.00487065 - cropped_depth
    cropped_height2 = cropped_height / 1 * 1000
    cropped_height2 = numpy.uint8(cropped_height2)
    mean_height = numpy.nanmean(cropped_height)
    ##print (mean_height)
    #base area of the cropped depth area is 2.28628575 squared meters
```



```

#base area of the cropped depth area .448 m from the ground is
1.67660955 squared meters
#formula to correct area based on height is -
1.360884375(height)+2.28628575
corrected_area = (-1.360884375 * mean_height) + 2.28628575
#print (corrected_area)
ind_corrected_area = (-.00002398025625 * cropped_height) + .0000403225
ind_adm_corn_area = (-.00002649273259 * adm_corn_depth) +
.00003723608642
#print (numpy.sum(ind_corrected_area))
##print (ind_corrected_area)
volume = corrected_area * mean_height
ind_volume = ind_corrected_area * cropped_height
adm_corn_volume = ind_adm_corn_area * adm_corn_depth
#print (ind_volume)
ind_volume = numpy.nansum(ind_volume)
ind_volume_gallons = ind_volume * 264.172 #264.172
adm_corn_volume = numpy.nansum(adm_corn_volume)
adm_corn_volume_gallons = adm_corn_volume * 61023.7438368
#adm_corn_volume_gallons = adm_corn_volume
#print (ind_volume_gallons) #this is the latest method...
volume_gallons = volume * 264.172
vg_string = str(round(volume_gallons, 1))
ind_vg_string = str(round(ind_volume_gallons, 1))
cornvolstring = str(round(adm_corn_volume_gallons, 4))
print (cornvolstring)

#rainbow the depth image for sanity check...
colormap_jet = cv_image2 / 1 * 255
colormap_jet = numpy.uint8(colormap_jet)
colormap_jet = cv2.applyColorMap(colormap_jet, cv2.COLORMAP_JET)
colormap_jet_of_cropped = cropped_depth / 1 * 255
colormap_jet_of_cropped = numpy.uint8(colormap_jet_of_cropped)
colormap_jet_of_cropped = cv2.applyColorMap(colormap_jet_of_cropped,
cv2.COLORMAP_JET)
cropped_colormap_jet= colormap_jet[30:240, 390:660]
#colorize the nan (bad) pixels of the depth image...
depth_bgr = cv2.cvtColor(cv_image2, cv2.COLOR_GRAY2BGR) #was
cropped_depth
blue, green, red = cv2.split(depth_bgr)
blue = cv2.patchNaNs(blue, 255)
green = cv2.patchNaNs(green, 0)
red = cv2.patchNaNs(red, 0)
depth_truth = cv2.merge((blue, green, red))
cv2.imshow("truth", depth_truth)
cropped_depth_truth = depth_truth[270:310, 500:548]

pixel1=cropped_height[137,177]
pixel2=cropped_height[136,94]
pixel3=cv_image2[213,503]
pixel4=cv_image2[213,551]

```

```

#let's use some text...

cv2.putText(cv_image, "Volume %s cu in." % cornvolstring, (390,220),
cv2.FONT_ITALIC, 0.75, (255, 255, 255), 4) #this is the latest one...(8,
31)
cv2.imshow("COLOR RECTIFIED", cv_image)

#cropped_height_string = str(cropped_height)
numpy.savetxt("empty_raw2.txt", pile_depth, delimiter=",")
#with open('pile_raw.txt', 'w') as f:
#    f.write(cropped_height_string)
cv2.imshow("cropped_height2", cropped_height2)
##cv2.imshow("left color", cv_image)
#lets crop the color image...

cv2.imshow("cropped", cropped_image)
cv2.imshow("tape box", adm_image)
cv2.imshow("pile image", pile_image)


cv2.waitKey(60)


# initialize the node, execute main loop//
def start():

    rospy.init_node('COLOR_NODE')
    rospy.Subscriber('multisense/left/image_rect_color', Image,
depth_callback)
    #rospy.Subscriber('multisense/left/image_color', Image,
depth_callback2)
    rospy.Subscriber('multisense/depth', Image, depth_callback2)

    rospy.spin()

if __name__ == "__main__":
    start()

cv2.destroyAllWindows()

```

APPENDIX 2. ROS POINT CLOUD SCATTERPLOT PYTHON CODE

```
#!/usr/bin/env python3

#this one is working as of july 25 2021
import rospy
from roslib import message
import ros_numpy
from sensor_msgs.msg import PointCloud2
from sensor_msgs import point_cloud2
import numpy as np
import sensor_msgs.point_cloud2 as pc2
import struct
import ctypes
import pandas
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from mpl_toolkits import mplot3d

class NumberCounter:

    def __init__(self):

        self.fig, self.ax = plt.subplots(subplot_kw=dict(projection='3d'))
        self.ln = plt.scatter([],[], s=.5, marker="s") #s=11
        self.lm = plt.colorbar().set_label(label='Z (m)', font = 'times
new roman', size=12)

    def plot_init(self):
        self.ax.set_xlim(-1, 1)

        self.ax.set_xlabel("X (m)", font = "times new roman", fontsize
=12)

        self.ax.set_ylim(-1, 1)
        self.ax.set_ylabel("Y (m)", font="times new roman", fontsize=12)
        self.ax.set_zlim(0, 5) #0 to 4 for multisense...
        self.ax.set_zlabel("Z (m)", font="times new roman", fontsize = 12)

        self.ax.set_yticklabels(['1.00', '0.75', '0.50', '0.25', '0.00',
'-0.25', '-0.50', '-0.75', '-1.00'], font="times new roman")

        self.ax.w_xaxis.set_panel_color((1, 1, 1, 1.0))
        self.ax.w_yaxis.set_panel_color((1, 1, 1, 1.0))
        self.ax.w_zaxis.set_panel_color((1, 1, 1, 1.0))

        return self.ln, self.lm

    def callback(self, ros):
        self.gen = pandas.DataFrame(list(pc2.read_points(ros,
skip_nans=True))) #makes generated object read points...
```

```

        #breaking down the top line into individual lines for coding
analysis...
        self.gen2 = pc2.read_points(ros, skip_nans=True)
        #print (self.gen2)
        self.listed = list(self.gen2)
        self.listed = np.array(self.listed) #added 7/30/2021...
        self.a=self.listed[:,0]
        self.b=self.listed[:,1]
        self.c=self.listed[:,2]
        self.d=self.listed[:,3]
        print (self.d)

    def update_plot(self, frame):

        self.ln._offsets3d=(self.listed[:,0], self.listed[:,1],
self.listed[:,2])
        self.ln.set_array(self.listed[:,2])
        self.ln.set_clim(0, 5)

        self.ln.set_cmap("plasma")

        return self.ln, self.lm

rospy.init_node('number_counter', anonymous=True)
num = NumberCounter()

cloud_sub = rospy.Subscriber("/multisense/image_points2", PointCloud2,
num.callback)
ani = FuncAnimation(num.fig, num.update_plot, init_func=num.plot_init)
plt.show(block=True) plt.show(block=True)

```

APPENDIX 3. RGB AND HSV IMAGE CHANNEL THRESHOLDING

```
#heavily modified from Nathan Lam's work:
# https://gist.github.com/nathancy
# https://gist.github.com/nathancy/85002f65ad2b8d62cf4e43edebe78bf3

# I added in the RGB thesholding...

import cv2

import sys

import numpy as np


def nothing(x):
    pass


# Create a window
cv2.namedWindow('image')


# create trackbars for HSV change

cv2.createTrackbar('HMin','image',0,179,nothing) # Hue is from 0-179 for
Opencv
cv2.createTrackbar('SMin','image',0,255,nothing)
cv2.createTrackbar('VMin','image',0,255,nothing)
cv2.createTrackbar('HMax','image',0,179,nothing)
cv2.createTrackbar('SMax','image',0,255,nothing)
cv2.createTrackbar('VMax','image',0,255,nothing)


# create trackbars for RGB change

cv2.createTrackbar('RMin','image',0,255,nothing)
cv2.createTrackbar('GMin','image',0,255,nothing)
cv2.createTrackbar('BMin','image',0,255,nothing)
```

```

cv2.createTrackbar('RMax','image',0,255,nothing)
cv2.createTrackbar('GMax','image',0,255,nothing)
cv2.createTrackbar('BMax','image',0,255,nothing)

# Set default value for MAX HSV trackbars.
cv2.setTrackbarPos('HMax', 'image', 179)
cv2.setTrackbarPos('SMax', 'image', 255)
cv2.setTrackbarPos('VMax', 'image', 255)

# Set default value for MAX RGB trackbars.
cv2.setTrackbarPos('RMax', 'image', 255)
cv2.setTrackbarPos('GMax', 'image', 255)
cv2.setTrackbarPos('BMax', 'image', 255)

# Initialize to check if HSV min/max value changes
hMin = sMin = vMin = hMax = sMax = vMax = 0
phMin = psMin = pvMin = phMax = psMax = pvMax = 0

# Initialize to check if RGB min/max value changes
rMin = gMin = bMin = rMax = gMax = bMax = 0
prMin = pgMin = pbMin = prMax = pgMax = pbMax = 0

img = cv2.imread('1036533.png')
output = img
waitTime = 33

while(1):

```

```

# get current positions of all trackbars
hMin = cv2.getTrackbarPos('HMin','image')
sMin = cv2.getTrackbarPos('SMin','image')
vMin = cv2.getTrackbarPos('VMin','image')

hMax = cv2.getTrackbarPos('HMax','image')
sMax = cv2.getTrackbarPos('SMax','image')
vMax = cv2.getTrackbarPos('VMax','image')

rMin = cv2.getTrackbarPos('RMin','image')
gMin = cv2.getTrackbarPos('GMin','image')
bMin = cv2.getTrackbarPos('BMin','image')

rMax = cv2.getTrackbarPos('RMax','image')
gMax = cv2.getTrackbarPos('GMax','image')
bMax = cv2.getTrackbarPos('BMax','image')

# Set minimum and max HSV values to display
lower = np.array([hMin, sMin, vMin])
upper = np.array([hMax, sMax, vMax])

# Set minimum and max RGB values to display
lowerrgb = np.array([rMin, gMin, bMin])
upperrgb = np.array([rMax, gMax, bMax])

# Create HSV Image and threshold into a range.

```

```

hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

mask = cv2.inRange(hsv, lower, upper)

outputhsv = cv2.bitwise_and(img,img, mask= mask)

# Create rgb threshold into a range.

maskrgb = cv2.inRange(outputhsv, lowerrgb, upperrgb)

output = cv2.bitwise_and(outputhsv,outputhsv, mask= maskrgb)


# Print if there is a change in HSV value

if( (phMin != hMin) | (psMin != sMin) | (pvMin != vMin) | (phMax !=
hMax) | (psMax != sMax) | (pvMax != vMax) ):

    print("(hMin = %d , sMin = %d, vMin = %d), (hMax = %d , sMax = %d,
vMax = %d)" % (hMin , sMin , vMin, hMax, sMax , vMax))

    phMin = hMin

    psMin = sMin

    pvMin = vMin

    phMax = hMax

    psMax = sMax

    pvMax = vMax


# Print if there is a change in RGB value

if( (prMin != rMin) | (pgMin != gMin) | (pbMin != bMin) | (prMax !=
rMax) | (pgMax != gMax) | (pbMax != bMax) ):

    print("(rMin = %d , gMin = %d, bMin = %d), (rMax = %d , gMax = %d,
bMax = %d)" % (rMin , gMin , bMin, rMax, gMax , bMax))

    prMin = rMin

    pgMin = gMin

    pbMin = bMin

    prMax = rMax

```



```
pgMax = gMax
pbMax = bMax

cv2.imshow('image',output)

if cv2.waitKey(waitTime) & 0xFF == ord('q'):
    break

cv2.destroyAllWindows()
()
```

APPENDIX 4. VESSEL/GRAIN CART RELATIVE POSITION TRACKING PYTHON CODE

```
#!/usr/bin/env python3

#import modules//
import rospy
import cv2
import numpy
from cv_bridge import CvBridge

lower = (0, 245, 0)
upper = (0, 255, 0)

#import the message types from ROS//
#rostopic info shows message like Type: sensor_msgs/Image,//
#and we enter it like shown below//
from sensor_msgs.msg import Image
from sensor_msgs.msg import CompressedImage
bridge = CvBridge()
side = cv2.imread('/home/matt/stakepocket_strip.png')
side = cv2.cvtColor(side, cv2.COLOR_BGR2GRAY)
w, h = side.shape[::-1]
threshold = 0.97

def depth_callback(depth_msg):    #for left color image//
    global cv_image
    cv_image = bridge.compressed_imgmsg_to_cv2(depth_msg,
desired_encoding="bgr8")
    global r
    b, g, r = cv2.split(cv_image)

def depth_callback2(depth_msg2):    #for the depth image//
    cv_image2 = bridge.imgmsg_to_cv2(depth_msg2, desired_encoding="32FC1")
#passthrough also works besides 32fc1
    #make a mask for the depth image between 1.9 and 2.4 m...
    rail = cv2.inRange(cv_image2, 1.95, 2.10)
    global cv_image
    matches = cv2.matchTemplate(r,side,cv2.TM_CCOEFF_NORMED)
    locations = numpy.where( matches >= threshold)
    for pt in zip(*locations[::-1]):
        cv2.rectangle(cv_image, pt, (pt[0] + w, pt[1] + h), (0,255,0), 2)
    #print (pt)
    rail = cv2.bitwise_and(cv_image, cv_image, mask=rail)
    mask = cv2.inRange(rail, lower, upper)
    try:

        #the next three lines taken from stack overflow user Simon Penn:
        # https://stackoverflow.com/users/7544511/simon-penn
        # https://stackoverflow.com/questions/42924059/detect-different-
color-blob-opencv
```

```

        contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)
        blob = max(contours, key=lambda el: cv2.contourArea(el))
        #print (blob)
        M = cv2.moments(blob)
        x_coord = int(M["m10"] / M["m00"])
        y_coord = int(M["m01"] / M["m00"])
        x_coordpc = x_coord + 20
        percentage = -0.22*(x_coord)+120.888
        #print (percentage)
        percentage_string = str(round(percentage, 1))
        print (percentage_string)
        cv2.arrowsLine(cv_image, (x_coord, 10), (x_coord, 80),
(255,255,255), 8, tipLength=0.4)
        cv2.putText(cv_image, "Pass Percentage %s " % percentage_string,
(x_coordpc,31), cv2.FONT_ITALIC, 0.75, (255, 255, 255), 4)
    except (ValueError, ZeroDivisionError):
        pass

    cv2.imshow("color", cv_image)
    cv2.waitKey(60)
# initialize the node, execute main loop//
def start():
    rospy.init_node('COLOR_NODE')
    rospy.Subscriber('multisense/left/image_rect_color/compressed',
CompressedImage, depth_callback)
    rospy.Subscriber('multisense/depth', Image, depth_callback2)
    rospy.spin()

if __name__ == "__main__":
    start()

cv2.destroyAllWindows()

```

APPENDIX 5. TRUCK STAKE POCKET TEMPLATE MATCHING

```
#!/usr/bin/env python3

#import modules//
import rospy
import cv2
import numpy
from cv_bridge import CvBridge

lower = (0, 245, 0)
upper = (0, 255, 0)
threshold_area = 20

#import the message types from ROS//
#rostopic info shows message like Type: sensor_msgs/Image,//
#and we enter it like shown below//
from sensor_msgs.msg import Image
from sensor_msgs.msg import CompressedImage
bridge = CvBridge()
side = cv2.imread('/home/matt/sidepocket_strip.png')
side = cv2.cvtColor(side, cv2.COLOR_BGR2GRAY)
w, h = side.shape[::-1]
threshold = 0.97

def depth_callback(depth_msg):    #for left color image//
    global cv_image
    cv_image = bridge.compressed_imgmsg_to_cv2(depth_msg,
desired_encoding="bgr8")
    global r
    b, g, r = cv2.split(cv_image)

def depth_callback2(depth_msg2):    #for the depth image//
    cv_image2 = bridge.imgmsg_to_cv2(depth_msg2, desired_encoding="32FC1")
#passthrough also works besides 32fcl
    #make a mask for the depth image between 1.9 and 2.4 m...
    rail = cv2.inRange(cv_image2, 1.90, 2.00)
    global cv_image
    global r
    print (r)
    matches = cv2.matchTemplate(r,side,cv2.TM_CCOEFF_NORMED)
    locations = numpy.where( matches >= threshold)
    #print (locations)
    for pt in zip(*locations[::-1]):
        cv2.rectangle(cv_image, pt, (pt[0] + w, pt[1] + h), (0,255,0), 2)
        #print (pt)
    rail = cv2.bitwise_and(cv_image, cv_image, mask=rail)
    mask = cv2.inRange(rail, lower, upper)
    try:
        contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)
        for cnt in contours:
```

```

        area = cv2.contourArea(cnt)
        if area > threshold_area:
            M = cv2.moments(cnt)
            center = (int(M["m10"] / M["m00"]), int(M["m01"] /
M["m00"]))
            x_coord = int(M["m10"] / M["m00"])
            y_coord = int(M["m01"] / M["m00"])
            # cv2.arrowedLine(mask, (x_coord, 10), (x_coord, 80),
(255,0,0), 8, tipLength=0.4)
            cv2.arrowedLine(cv_image, (x_coord, 10), (x_coord, 80),
(255,255,255), 8, tipLength=0.4)
            cv2.circle(cv_image, center, 20, (255,255,255), -1)

    except (ValueError, ZeroDivisionError):
        pass

    cv2.imshow("color", cv_image)

    cv2.waitKey(60)
# initialize the node, execute main loop//
def start():
    rospy.init_node('COLOR_NODE')
    rospy.Subscriber('multisense/left/image_rect_color/compressed',
CompressedImage, depth_callback)
    rospy.Subscriber('multisense/depth', Image, depth_callback2)
    rospy.spin()

if __name__ == "__main__":
    start()

cv2.destroyAllWindows()

```

APPENDIX 6. VESSEL VOLUME CALCULATION

```
#!/usr/bin/env python3

import modules//
import rospy
import cv2
import numpy
from cv_bridge import CvBridge

import the message types from ROS//
#rostopic info shows message like Type: sensor_msgs/Image,//
#and we enter it like shown below//
from sensor_msgs.msg import Image
from sensor_msgs.msg import CompressedImage
bridge = CvBridge()

#create a matrices that correct for the sloped depth images...
mat1 = numpy.arange(-508, 36, 1).reshape((544,1))
mat1 = numpy.tile(mat1, (1, 1024))
mat1 = mat1 * .002811

#lower and upper rgb values for the normals image stream...
lowerrgb = numpy.array([181, 133, 127])
upperrgb = numpy.array([217, 230, 127])

#lower and upper rgb and hsv values for the straight color image stream...
# Set minimum and max HSV values to display
lowerhsv_rgbimage = numpy.array([0, 0, 120])
upperhsv_rgbimage = numpy.array([83, 255, 255])

# Set minimum and max RGB values to display
lowerrgb_rgbimage = numpy.array([58, 147, 0])
upperrgb_rgbimage = numpy.array([255, 255, 255])

#lower and upper gray values for the depth image stream...
##lower_gray = numpy.array([3.0])
##upper_gray = numpy.array([4.0])
lower_gray = numpy.array([2.3]) #the larger this number the more pixels
taken off top of truck...
upper_gray = numpy.array([6.0])

#lower and upper ground values for the corrected depth image stream...
lower_ground = numpy.array([0.0])
upper_ground = numpy.array([3.45]) #3.1

def depth_callback(depth_msg):    #for left color image//
    global cv_image
    cv_image = bridge.compressed_imgmsg_to_cv2(depth_msg,
desired_encoding="bgr8")
```

```

##      cv2.imshow("mask_all", cv_image)
##      cv2.waitKey(60)
      mask_rgbimage = cv2.inRange(cv_image, lowerrgb_rgbimage,
upperrgb_rgbimage)
      hsv = cv2.cvtColor(cv_image, cv2.COLOR_BGR2HSV)
      mask_hsvimage = cv2.inRange(hsv, lowerhsv_rgbimage, upperhsv_rgbimage)
      global masks1
      masks1 = mask_rgbimage & mask_hsvimage
      masked_color = cv2.bitwise_and(cv_image, cv_image, mask=mask_rgbimage)
      pub = rospy.Publisher('imager', Image,
queue_size=10)
      pub.publish(bridge.cv2_to_imgmsg(masks1))

def depth_callback2(depth_msg2):    #for the depth image//
    cv_image2 = bridge.imgmsg_to_cv2(depth_msg2, desired_encoding="32FC1")
    #passthrough also works besides 32FC1
    adjusted = cv_image2 + mat1

    ground_mask = cv2.inRange(adjusted, lower_ground, upper_ground)
    #ground_mask
    colormap_jet = cv_image2 / 1 * 255

    colormap_jet = numpy.uint8(colormap_jet)
    #following 9 lines of code taken from stack overflow user sowlosc
    # https://stackoverflow.com/users/10636162/sowlosc
    # https://stackoverflow.com/questions/53350391/surface-normal-
calculation-from-depth-map-in-python/53351384#53351384
    zy, zx = numpy.gradient(colormap_jet)
    normal = numpy.dstack((-zx, -zy, numpy.ones_like(colormap_jet)))
    n = numpy.linalg.norm(normal, axis=2)
    normal[:, :, 0] /= n
    normal[:, :, 1] /= n
    normal[:, :, 2] /= n
    normal += 1 #normal = normal + 1
    normal /= 2
    normal *= 255
    normal = normal[:, :, :-1]
    normal_blur = normal.astype(numpy.uint8)
    normal_blur = cv2.medianBlur(normal_blur, 21)
    mask_normal = cv2.inRange(normal_blur, lowerrgb, upperrgb) #normal
mask
    mask_rail = cv2.inRange(adjusted, lower_gray, upper_gray) #rail mask
    # cv2.imshow("maskrail", mask_rail)
    masks2 = ground_mask & mask_rail & mask_normal
    masks2 = masks2 & masks1
    ##### adjusted = cv2.bitwise_and(adjusted, adjusted, mask=masks2)
    masked_depth = cv2.bitwise_and(cv_image, cv_image, mask=masks2)
    # cv2.imshow("grain pixels", masked_depth)
    pub_mask_depth = rospy.Publisher('mask_depth', Image,
queue_size=10)
    pub_mask_depth.publish(bridge.cv2_to_imgmsg(masked_depth))
    adjusted = cv2.bitwise_and(adjusted, adjusted, mask=masks2)

```

```

adjusted_image = adjusted
adjusted[adjusted == 0.0] = numpy.nan

# cv2.imshow("mask_all_adjusted", adjusted_image)
adjusted = numpy.nanmean(adjusted)
print (adjusted)
adjusted = 3.268 - adjusted
volume = adjusted * 10.96
vg_string = str(round(volume, 1))
cv2.putText(cv_image, "Volume %s Cu M." % vg_string, (8,31),
cv2.FONT_ITALIC, 0.75, (255, 255, 255), 4)
#cv2.imshow("volume", cv_image)
vertical = numpy.concatenate((masked_depth, cv_image), axis=1)
cv2.imshow("comparison", vertical) #####
pub_mask_all = rospy.Publisher('mask_all', Image,
queue_size=10)

cv2.waitKey(60)

# initialize the node, execute main loop//
def start():
    rospy.init_node('COLOR_NODE')
    rospy.Subscriber('multisense/left/image_rect_color/compressed',
CompressedImage, depth_callback)
    rospy.Subscriber('multisense/depth', Image, depth_callback2)
    rospy.spin()

if __name__ == "__main__":
    start()

cv2.destroyAllWindows()

```


APPENDIX 7: ROS USB CAMERA IMAGE STREAMING IN OPENCV

```
#!/usr/bin/env python3

#import modules//
import rospy
import cv2
import numpy
from cv_bridge import CvBridge

#import the message types from ROS//
#rostopic info shows message like Type: sensor_msgs/Image,//
#and we enter it like shown below//
from sensor_msgs.msg import Image
bridge = CvBridge()

def depth_callback(depth_msg):    #for left color image//
    global cv_image
    global cropped_image
    cv_image = bridge.imgmsg_to_cv2(depth_msg,
desired_encoding="passthrough")

    cv2.imshow("COLOR IMAGE", cv_image)
    cv2.waitKey(60)

# initialize the node, execute main loop//
def start():

    rospy.init_node('COLOR_NODE')

    rospy.Subscriber('scale', Image, depth_callback)

    rospy.spin()

if __name__ == "__main__":
    start()

cv2.destroyAllWindows()
```

APPENDIX 8. CARNEGIE ROBOTICS S21 DATA SHEETS



MULTISENSE S21

PHYSICAL



Height:	8.2 cm
Width:	28.5 cm
Depth:	10 cm
Weight:	1.5 kg

ENVIRONMENTAL



Operating Temperature:	-10 to 50 C
Environmental Rating:	IP 68

RUGGEDIZATION



Corning®
Gorilla®
Glass Lens
Shields:

The unique composition of Gorilla® Glass allows for a deep layer of high compressive stress, that acts as "armour", making the lens shields exceptionally tough and resistant to chips and scratches.

ELECTRICAL



Voltage (nominal):	24 V
Voltage (range):	18–36 V
Power (nominal):	7 W
External Connector:	+ Power: M12-A5/Male + Ethernet: M12-XB/Female

IMAGE SENSORS



Model*:	CMV2000
*Monochrome and color options available.	
Resolution:	2048 x 1088
Active Area:	11.2 x 6 mm
Frame Rate:	30 FPS max
Sensitivity*:	5.56 V/lux-s
*Value for monochrome imagers. Bayer filter on color imagers reduces sensitivity.	
Color Filter Array:	Bayer

LENSES



Focal Length*:	4.8–8.0 mm
*4.8 mm standard.	
Field of View:	Varies with lens
Aperture:	Fixed to f4.0 at factory
Focus:	Fixed at factory

ILLUMINATION



Number of LED Illuminators*:	2
*Supports up to two additional external LEDs.	
Color Temperature:	4100K
Brightness:	690 lm each
Power*:	6 W per LED
*Light power is at 100% duty, no strobing. Strobing is user-adjustable.	
Field of View:	+ 1 @ 18° + 1 @ 44°
Synchronization:	Continuous illumination or synchronized to camera exposure.



INTERFACE



Network Interface*: 1 Gigabit ethernet port
(1000BASE-T)

**Full-duplex only. Can auto-negotiate down to 10/100 speeds at significant impact to sustained camera framerate.*

Throughput*: Up to 120 MB/s

**Achievable throughput depends on quality of host side ethernet adapter/drivers.*

Jumbo Frames*: Up to 9000 bytes

**Full frame rates may not be achievable without use of jumbo-frames.*

Low-level Protocol: UDP/IP; IPv4 only

IP Address Assignment: Static

Device Discovery: Direct connect to known IP

Application Interface (C++): High-performance C++ API with support for blocking, polled and asynchronous (callback based) methods

Application Interface (ROS): ROS-based API and tool set

**View live image and 3D range data, adjust camera and stereo parameters, log and playback data, check calibration, and change IP address.*

Image Formats*: Grayscale, YCbCr; packed, planar; various bit depths

**Formats may be selected to optimize use of available network bandwidth. API can provide efficient automatic conversion to standard byte-aligned formats on host side.*

Image Streams*: Unrectified (left/right), rectified (left/right), and disparity (depth)

**ROS API streams point clouds, depth images, and RGB images.*

STEREO VISION



Algorithm: SGM
(Semi-global stereo matching)

Maximum Disparities: 256

Sub-pixel Resolution: 1/16th pixel

Peak Throughput: 2 GPxD/s
(Giga-pixel disparities/second)

Performance @ 2048 x 1088: 7.5 FPS with up to 128 disparities

Performance @ 2048 x 544: 15 FPS with up to 128 disparities

Performance @ 1024 x 544: 30 FPS with up to 128 disparities

Minimum Range: 0.4 m

TRIGGERING/SYNCRONIZATION



External Opto-isolated Input: 2x

External Opto-isolated Output: 2x

Time-base*: Internal timebase with sub-microsecond resolution

**Used to timestamp all outgoing data (including disparity maps and captured images).*

Time Synchronization*: External pulse input (e.g. pulse-per-second) time system with host

**PPS mutually exclusive with external trigger (due to limit of one external input). PPS signal sets sub-second time, while network message sets absolute time.*

Camera Trigger Sources: Internal free-running; network message; external trigger input

Opto-isolated Output Sources*: Synchronized to camera exposure; pulse-per-second

**Allows external cameras and illumination devices to be synchronized with internal camera exposure. Alternatively, external devices may be synchronized such that their exposures never overlap with internal camera exposure (for example, in order to support a structured illumination device that is only visible to some of the cameras).*



MULTISENSE STEREO

COMPACT & ACCURATE 3D DATA COLLECTION



MultiSense 3D Cameras

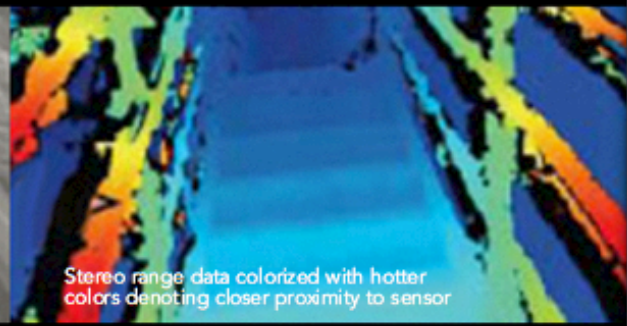
Using less than 10 Watts, the MultiSense 3D cameras produce full-frame range images at up to 30 Hz with approximately 1 frame of latency.



Carnegie Robotics.

MULTISENSE STEREO

COMPACT & ACCURATE
3D DATA COLLECTION



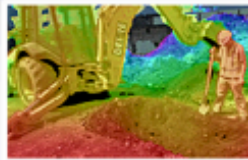
Stereo range data colorized with hotter colors denoting closer proximity to sensor

KEY FEATURES



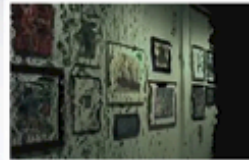
RUGGED DESIGN

Our cameras are designed and tested for harsh environments. Carnegie Robotics subjects each sensor to a battery of vibration and thermal tests as part of its quality process.



DENSE, LOW LATENCY STEREO IMAGERY

MultiSense can find over 11 million feature matches every second. If desired, the stereo point cloud can be augmented by overlaying color image data onto the point cloud—resulting in compelling, very low latency, life-like 3D data sets.



EASY DATA OPTIONS

3D point clouds from the stereo camera can be colorized onboard the sensor. The sensor provides extremely dense "full frame" range data at user-configurable frame-rates and also outputs standard color video.



MORE POINTS, LESS HASSLES

Our easy to use ROS-based API and tools allow you to view live image and 3D range data; adjust camera and stereo parameters; log data; playback logs; check the unit's calibration; and change the sensor's IP address. An open-source C++ library and Gigabit Ethernet interface make it easy to integrate live data into your robot, vehicle, mobile equipment, lab environment, or other application.

Visible light LEDs
neutral color and energy efficient IR LEDs are optional

Gorilla Glass Windows
resistant to scratching and impacts low optical distortion

Rugged Connectors
waterproof and EMI shielded

Mounting
steel key-locking inserts



MultiSense S7

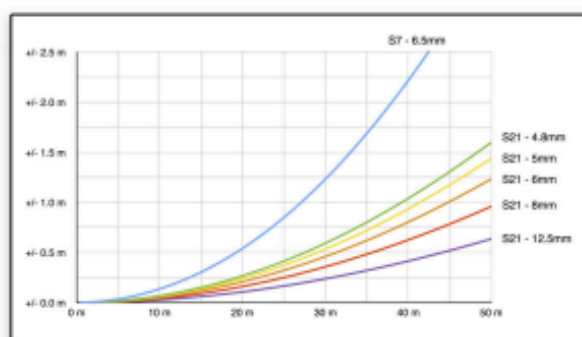


MultiSense S21

carnegierobotics.com


Carnegie Robotics.

Mechanical/Environmental	S7	S7S	S21	Note
Operating Temperature	-10 to 50 C			
Environmental Rating	IP68			
Height	6.5 cm	6.5 cm	6.3 cm	
Width	13 cm	13 cm	27.4 cm	
Depth	13 cm	11 cm	13.0 cm	
Weight	1.2 kg	1.1 kg	1.5 kg	
Electrical				
Voltage (nominal)	24-48 V		12-48 V	
Voltage (maximum)	24-48 V		10-48 V	S21 requires 10 V to power on, voltage can then drop to 6 V.
Power (nominal)	7 W			
Power (full lighting)	7 W, 19 W	n/a	n/a	7 W with lights strobing, 19 W with no strobing.
External Connector	Glensair Mighty Mouse, 801-009-07MT9-19PA			S7 also includes 2x 801-009-07MT6-4SA connector for external lighting power and control.
Image Sensors				
Model	CMOSIS CMV2000 or CMV4000			Monochrome or IR sensor options also available.
Resolution	2048 x 1088 or 2048 x 2048			
Active Area	11.2 x 6 mm or 11.2 x 11.2 mm			Larger image area corresponds to CMV4000 imager.
Frame Rate	30 FPS max			
Sensitivity	5.56 V/lux-s			For monochrome imagers, Bayer filter on color imagers reduces sensitivity.
Color Filter Array	Bayer			
Lenses				
Focal Length	6.5 mm		4.8-12.5 mm	
Field of View	80° x 49° (2MP sensor); 80° x 80° (4MP sensor)		Varies with lens	
Aperture	Fixed at factory			Possible values: f1.4 to f16.
Focus	Fixed at factory			
Illumination				
Number LED Illuminators	2	0	0	S7 has support for additional 2x external LEDs.
Color Temperature	4100K	n/a	n/a	
Brightness	690 lm each	n/a	n/a	
Power	6 W per LED	n/a	n/a	S7 light power is at 100% duty, no strobing. Strobing is user-adjustable.
Field of View	1 @ 18°, 1 @ 44°	n/a	n/a	
Synchronization	Cont or Sync	n/a	n/a	Continuous illumination or synchronized to camera exposure.
Stereo Vision				
Algorithm	SGM			
Maximum Disparities	256			
Sub-pixel Resolution	1/16th pixel			
Peak Throughput	2 GPxD/s (Giga-Pixel-Disparities/second)			
Performance @ 2048 x 1088	7.5 FPS with up to 128 disparities			
Performance @ 2048 x 544	15 FPS with up to 128 disparities			
Performance @ 1024 x 544	30 FPS with up to 128 disparities			
Minimum Range	0.4 m	0.4 m	1.5 m	With 2048 pixel horizontal resolution, 256 disparities.
Triggering/Synchronization				
External Opto- isolated Input	1x	1x	2x	
External Opto-isolated Output	1x	1x	2x	
Time-base	Internal timebase with sub-microsecond resolution			Used to timestamp all outgoing data (including disparity maps and captured images).
Time Synchronization	External pulse input (e.g. Pulse-Per-Second) Time system with host			PPS mutually exclusive with external trigger (due to limit of 1 external input). PPS signal sets sub-second time, while network message sets absolute time.
Camera Trigger Sources	Internal free-running; Network message; External trigger input			
Opto-isolated Output Sources	Synchronized to camera exposure; Pulse-per-second			Allows external cameras and illumination devices to be synchronized with internal camera exposure. Alternatively, external devices may be synchronized such that their exposures never overlap with internal camera exposure (for example, in order to support a structured illumination device that is only visible to some of the cameras).
Interface				
Network Interface	1 Gigabit Ethernet port (1000BASE-T)			Full-duplex only. Can auto-negotiate down to 10/100 speeds at significant impact to sustained camera framerate.
Throughput	Up to 120 MB/s			Achievable throughput depends on quality of host side Ethernet adapter/drivers.
Jumbo Frames	Up to 9000 bytes			Full frame rates may not be achievable without use of jumbo-frames.
Low-level Protocol	UDP/IP; IPv4 only			
IP Address Assignment	Static			
Device Discovery	Direct connect to known IP			
Application Interface (C++)	High-performance C++ API with support for blocking, polled and asynchronous (callback based) methods.			
Application Interface (ROS)	ROS-based API and tool set			View live image and 3D range data, adjust camera and stereo parameters, log and playback data, check calibration, and change IP address.
Image Formats	Grayscale, RGB, YCbCr; Packed, Planar; Various bit depths			Formats may be selected to optimize use of available network bandwidth. API can provide efficient automatic conversion to standard byte-aligned formats on host side.
Image Streams	Unrectified (left/right), Rectified (left/right), and Depth			



Estimated stereo accuracy as a function of lens type and range

Grow with us.



Carnegie Robotics LLC
4501 Hatfield Street
Pittsburgh, PA 15201


Phone: (412) 251-0321
Fax: (412) 251-0319
info@carnegierobotics.com

APPENDIX 9. VELODYNE LYDAR DATA SHEETS

Velodyne Lidar[®]

Puck[™]


VERSATILE REAL-TIME LIDAR SENSOR



Puck

Velodyne Lidar's Puck is a small and compact lidar that is performance and power optimized for usage across a variety of applications ranging from automotive, mapping, robotics, security, smart cities and more. The Puck is attractively priced and built on the foundations of Velodyne's leadership in lidar, enabling real-time, surround view, 3D distance and calibrated reflectivity measurements.


The Puck has a range of 100 m and generates up to ~600,000 points/second, across a 360° horizontal field of view and a 30° vertical field of view. It uses proven, Class 1 eye-safe 905 nm technology with substantial autonomous fleet validation, making the Puck a sensor of choice for lower speed autonomous vehicle (AV) applications. The Puck has best-in-class power, which enables operation over a wide temperature range. Its use of off-the-shelf components enables enhanced scalability and attractive volume pricing. Like other Velodyne sensors, the Puck has world-class technical support available across North America, Europe & Asia from the world's leading lidar company.



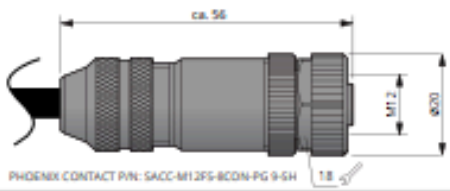
Puck at a glance

- 100 m range with compact form factor
- Proven 905 nm tech, with largest install base
- Top-of-the-line field-of-view
- Best-in-class accuracy and calibrated intensity
- Best-in-class power & temperature range
- Sensor-to-sensor interference mitigation feature
- Optional, enhanced short range detection
- Versatile, with attractive volume pricing

DIMENSIONS (Subject to change)




M12 CONNECTOR OPTION



For other connector options contact
Velodyne Sales (sales@velodyne.com)

velodynelidar.com



Real-Time Lidar Sensor

The Puck provides high definition 3-dimensional information about the surrounding environment.

Specifications:	
Sensor:	<ul style="list-style-type: none"> • 16 Channels • Measurement Range: 100 m • Range Accuracy: Up to ± 3 cm (Typical)¹ • Field of View (Vertical): $+15.0^\circ$ to -15.0° (30°) • Angular Resolution (Vertical): 2.0° • Field of View (Horizontal): 360° • Angular Resolution (Horizontal/Azimuth): 0.1° – 0.4° • Rotation Rate: 5 Hz – 20 Hz • Integrated Web Server for Easy Monitoring and Configuration
Laser:	<ul style="list-style-type: none"> • Laser Product Classification: Class 1 Eye-safe per IEC 60825-1:2007 & 2014 • Wavelength: 903 nm
Mechanical/ Electrical/ Operational	<ul style="list-style-type: none"> • Power Consumption: 8 W (Typical)² • Operating Voltage: 9 V – 18 V (with Interface Box and Regulated Power Supply) • Weight: ~830 g (without Cabling and Interface Box) • Dimensions: See diagram on previous page • Environmental Protection: IP67 • Operating Temperature: -10°C to $+60^\circ\text{C}$³ • Storage Temperature: -40°C to $+105^\circ\text{C}$
Output:	<ul style="list-style-type: none"> • 3D Lidar Data Points Generated: <ul style="list-style-type: none"> - Single Return Mode: ~300,000 points per second - Dual Return Mode: ~600,000 points per second • 100 Mbps Ethernet Connection • UDP Packets Contain: <ul style="list-style-type: none"> - Time of Flight Distance Measurement - Calibrated Reflectivity Measurement - Rotation Angles - Synchronized Time Stamps (μs resolution) • GPS: \$GPRMC and \$GPGGA NMEA Sentences from GPS Receiver (GPS not included)

63-9229 Rev-K VLP-16

For more details and ordering information, contact Velodyne Sales (sales@velodyne.com)

1. Typical accuracy refers to ambient wall test performance across most channels and may vary based on factors including but not limited to range, temperature and target reflectivity.

2. Operating power may be affected by factors including but not limited to range, reflectivity and environmental conditions.

3. Operating temperature may be affected by factors including but not limited to air flow and sun load.



CLASS 1 LASER PRODUCT

Copyright ©2019 Velodyne Lidar, Inc. Specifications are subject to change. Other trademarks or registered trademarks are property of their respective owners.

Velodyne Lidar, Inc. 5521 Hellyer Ave, San Jose, CA 95138 / lidar@velodyne.com / 408.465.2800

velodynelidar.com

APPENDIX 10. PANAVISE DRAWING

ALL DIMENSIONS ARE
FOR REFERENCE ONLY

DIMENSIONS ARE IN INCHES [MILLIMETERS]
UNLESS OTHERWISE NOTED

SPECIFICATIONS

OVERALL HEIGHT 3 3/16 [80.78]

SHIPPING WEIGHT 1.7 POUNDS [.77 KG]

COLOR SILVER PAINT

MOVEMENT:

TILT 180°

ROTATION 360°

TURNS 360°

MATERIAL:

1. HALF BALL HOUSING 380 ALUMINUM DIE CASTING
PAINTED SILVER
2. KNOB 380 ALUMINUM DIE CASTING
PAINTED BLACK
STEEL SCREW ZINC PLATED
3. BODY 380 ALUMINUM DIE CASTING
PAINTED SILVER
4. HALF BALL (2 EA.) 380 ALUMINUM DIE CASTING
BURNISHED

NOT SHOWN:

RETAINER RING 380 ALUMINUM
NATURAL FINISH

SCREW (2 EA) 8-32 X 3/8 FLAT HEAD
ZINC PLATED

BOLT 5/16-18 X 1
STEEL HEX HEAD
ZINC PLATED

ALL DESIGN, OPERATIVE, AND PROCESS DATA PERTAINING TO THE ARTICLE ON THIS SHEET IS THE PROPERTY OF PANAVISE PRODUCTS, INC., RENO, NV. THIS INFORMATION IS DISCLOSED IN CONFIDENCE AND IS NOT TO BE COPIED, REPRODUCED, OR REVEALED TO OR APPROPRIATED BY OTHERS, IN PART OR IN WHOLE, WITHOUT THE EXPRESSED CONSENT OF PANAVISE. THIS PRINT IS LOANED, AND MUST NOT BE USED IN ANY MANNER DETRIMENTAL TO THE INTEREST OF PANAVISE, AND MUST BE RETURNED ON DEMAND.

PANAVISE
7540 Lobert Drive, Reno, Nevada 89511-1225
Phone (702) 850-2900, Fax (702) 850-2925

TITLE

HEAVY DUTY BASE
MODEL 400

BASIC DIM. ENGLISH [ALTERNATE DIM. METRIC]

DRAWN	DATE	SCALE	SIZE	DWG. NO.	REV
I BENNETT	DEC 95	NONE	SHT 1 OF 1	400 SP	A

SIZE

DWG. NO.

400 SP

REV

APPENDIX 11. VIDEOS LINKS

- Dust videos, with dust and occlusion pixels highlighted in blue:
 - <https://youtu.be/T05JJdJ2WGk>
 - <https://youtu.be/c6su6ujlJas>
- Near real-time volume measurement videos (lag mostly due to hardware limitations, approx. sub 3 second lag):
 - <https://youtu.be/8AKulfh-Nv8>
 - <https://youtu.be/iyAok8QTtRk>
 - <https://youtu.be/65FfeNRWyBU>
- Relative grain-truck grain-cart position tracking:
 - https://youtu.be/5_joka4SDlQ
 - <https://youtu.be/TEiCl-tCV9U>
 - <https://youtu.be/JvTVf1NUATU>
- Multiple stake pocket template matching:
 - https://youtu.be/F_fX7HjcWTY
 - <https://youtu.be/A9SvHpQXODo>
 - <https://youtu.be/cwwiYwcy-qs>
- Frame lag between depth and color images:
 - <https://youtu.be/rvSzDVpXA54>
 - <https://youtu.be/CseIfRUiS3Q>
 - <https://youtu.be/zyfcfr-7PrY>
- Point cloud streaming from dust chambers using Python:
 - Low dust, LiDAR: <https://youtu.be/T1y2qCPrGI0>
 - Low dust, stereo camera: <https://youtu.be/yIpnjqxIJ6s>

REFERENCES

- Anwar, H., S.M. Abbas, A. Muhammad, and K. Berns. 2014. Volumetric estimation of contained soil using 3d sensors. In Intl. Commercial Vehicle Technology Symposium (pp. 11-13).
- Banthia, V., & Friesen, M. I. J. (2021). *U.S. Patent No. 11,008,177*. Washington, DC: U.S. Patent and Trademark Office.
- Barth, R., J. Baur, T. Buschmann, Y. Edan, T. Hellström, T. Nguyen, and R. Vitzrabin. 2014. Using ROS for agricultural robotics-design considerations and experiences. In Proceedings of the Second International Conference on Robotics and associated High-technologies and equipment for agriculture and forestry (pp. 509-518).
- Bezen, R., Y. Edan, and I. Halachmi. 2020. Computer vision system for measuring individual cow feed intake using RGB-D camera and deep learning algorithms. *Computers and Electronics in Agriculture*, 172, 105345.
- Bonefas, Z. T., H. Herman, and J. Campoy. 2018. Fill level indicator for an automated unloading system. U.S. Patent No. 10,019,790. Washington, DC: U.S. Patent and Trademark Office.
- Broggi, Alberto. 2020. A closer look at lidar and stereo vision.
<https://www.ambarella.com/blog/a-closer-look-at-lidar-and-stereovision/> Accessed 27 July 2022.
- Claas. 2022. <https://www.claasofamerica.com/product/forageharvesters/jaguar900-series/cropflow> Accessed 27 July 2022.
- CNH Industrial. 2022. <https://agriculture.newholland.com/middleeast/en/precision-land-management/products/harvest-solutions/intellifill-system> Accessed 27 July 2022.
- Dandrifosse, S., A. Bouvry, V. Leemans, B. Dumont, and B. Mercatoris. 2020. Imaging wheat canopy through stereo vision: Overcoming the challenges of the laboratory to field transition for morphological features extraction. *Frontiers in Plant Science*, 11, 96.
- Day, Tim. 2016. Response to “Calculate surface normals from depth image using neighboring pixels cross product.” Stack overflow web forum.
<https://stackoverflow.com/questions/34644101/calculate-surface-normals-from-depth-image-using-neighboring-pixels-cross-product> Accessed 27 July 2022.
- Ehlert, D., M. Heisig, and R. Adamek. 2010. Suitability of a laser rangefinder to characterize winter wheat. *Precision agriculture*, 11(6), 650-663.
- Ehlert, D. and M. Heisig. 2013. Applications of a laser scanner for crop production. *Mechanization in agriculture & Conserving of the resources*, Vol. 59 (2013), Issue 3, pg(s) 7-10.

- Gaard, J. D. 2012. Grain wagon fill detection using ultrasonic sensors (Doctoral dissertation, Iowa State University).
- Glennie, C. L., A. Kusari, and A. Facchin. 2016. Calibration and stability analysis of the VLP-16 laser scanner. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 40.
- Grimstad, L., and P.J. From. 2017. The Thorvald II agricultural robotic system. *Robotics*, 6(4), 24.
- Huang, Justin. ros-rviz/license GitHub page. <https://github.com/jstnhuang/ros-rviz/blob/master/LICENSE> Accessed 27 July 2022.
- John Deere. 2022. <https://www.deere.com/en/technology-products/precision-ag-technology/guidance/active-fill-control/> Accessed 27 July 2022.
- Kashiha, M., C. Bahr, S.A. Haredasht, S. Ott, C.P. Moons, T.A. Niewold, and D. Berckmans. 2013. The automatic monitoring of pigs water use by cameras. *Computers and Electronics in Agriculture*, 90, 164-169.
- Kay, Jackie. 2016. Introduction to real-time systems. https://design.ros2.org/articles/realtime_background.html#:~:text=Real%2Dtime%20software%20guarantees%20correct,systems%2C%20spacecrafts%20and%20planetary%20rovers. Accessed 27 July 2022.
- Kim, W. S., D.H. Lee, Y.J. Kim, T. Kim, W.S. Lee, and C.H. Choi. 2021. Stereo-vision-based crop height estimation for agricultural robots. *Computers and Electronics in Agriculture*, 181, 105937.
- Kise, M., Z.T. Bonefas, S.J. Moorehead, and J.F. Reid. 2010. Performance evaluation on perception sensors for agricultural vehicle automation. In 2nd International Conference on Machine Control & Guidance (pp. 265-273).
- Lam, Nathan. 2020. Response to topic “Choosing the correct upper and lower HSV boundaries for color detection with `cv::inRange` (OpenCV).” Stack Overflow web forum. <https://stackoverflow.com/questions/10948589/choosing-the-correct-upper-and-lower-hsv-boundaries-for-color-detection-withcv/59906154#59906154> Accessed 27 July 2022. GitHub link: <https://gist.github.com/nathancy/85002f65ad2b8d62cf4e43edebe78bf3>
- Lenaerts, B., B., Missotten, J. De Baerdemaeker, and W. Saeys. 2012. LiDaR sensing to monitor straw output quality of a combine harvester. *Computers and electronics in agriculture*, 85, 40-44.
- Matplotlib. 2022. <https://matplotlib.org/> Accessed 27 July 2022.

- Méndez, V., A. Pérez-Romero, R. Sola-Guirado, A. Miranda-Fuentes, F. Manzano-Agugliaro, A. Zapata-Sierra, and A. Rodríguez-Lizana. 2019. In-field estimation of orange number and size by 3D laser scanning. *Agronomy*, 9(12), 885.
- Milella, A., R. Marani, A. Petitti, and G. Reina. 2019. In-field high throughput grapevine phenotyping with a consumer-grade depth camera. *Computers and electronics in agriculture*, 156, 293-306.
- NumPy. 2022. <https://numpy.org/> Accessed 27 July 2022.
- OpenCV. 2022. <https://opencv.org/> Accessed 27 July 2022.
- PanaVise. 2022. Model 400 Heavy Duty Base. <https://www.panavise.com/index.html?pageID=1&page=full&--eqskudatarq=30> Accessed 27 July 2022.
- Penn, Simon. 2017. Code posted on topic “Detect different color blob opencv.” Stack overflow web forum. <https://stackoverflow.com/questions/42924059/detect-different-color-blob-opencv> Accessed 27 July 2022.
User link: <https://stackoverflow.com/users/7544511/simon-penn>
- Phillips, T. G., N. Guenther, and P.R. McAree, P. R. 2017. When the dust settles: The four behaviors of lidar in the presence of fine airborne particulates. *Journal of field robotics*, 34(5), 985-1009.
- Pilarski, T., M. Happold, H. Pangels, M. Ollis, K. Fitzpatrick, and A. Stentz. 2002. The demeter system for automated harvesting. *Autonomous Robots*, 13(1), 9-20.
- Potter, B. C. 2012. Design and assessment of an automated grain auger position control system (Doctoral dissertation, Iowa State University).
- Robot Agriculture. 2022. GitHub page. <https://github.com/ros-agriculture> Accessed 27 July 2022.
- Rogers, M. B. and D. Buckmaster .2022. Volume measurement of biological materials in livestock or vehicular settings using computer vision. Purdue University Research Repository. [doi:10.4231/NE9P-HC76](https://doi.org/10.4231/NE9P-HC76)
- Rovira-Más, F., S. Han, J. Wei, and J.F. Reid (2007). Autonomous guidance of a corn harvester using stereo vision. *Agricultural Engineering International: CIGR Journal*.
- Rundgren, E. 2017. Automatic Volume Estimation of Timber from Multi-View Stereo 3D Reconstruction.
- Ryde, J., and N. Hillier. 2009. Performance of laser and radar ranging devices in adverse environmental conditions. *Journal of Field Robotics*, 26(9), 712-727.

- Saeys, W., B. Lenaerts, G. Craessaerts, and J. De Baerdemaeker, J. 2009. Estimation of the crop density of small grains using LiDAR sensors. *Biosystems Engineering*, 102(1), 22-30.
- Schofield, C. P. 1990. Evaluation of image analysis as a means of estimating the weight of pigs. *Journal of Agricultural Engineering Research*, 47, 287-296.
- Shkanaev, A., A.V. Panchenko, and N.I. Kodosov. 2017. Grain wagon fill detection using camera and deep convolution network. *International Journal of Applied Engineering Research*, 12(21), 11077-11081.
- Van der Stuyft, E., C.P. Schofield, J.M. Randall, P. Wambacq, and V. Goedseels. 1991. Development and application of computer vision systems for use in livestock production. *Computers and Electronics in Agriculture*, 6(3), 243-265.
- Wevolver., 2020. 2020 Autonomous Vehicle Technology Report.
<https://www.wevolver.com/article/2020.autonomous.vehicle.technology.report#references>
 _s Accessed 27 July 2022.
- Yoneda, K., N. Suganuma, R. Yanase, and M. Aldibaja. 2019. Automated driving recognition technologies for adverse weather conditions. *IATSS research*, 43(4), 253-262.
- Yukun, S., H. Pengju, W. Yujie, C. Ziqi, L. Yang, D. Baisheng, and Z. Yonggen, Z. 2019. Automatic monitoring system for individual dairy cows based on a deep learning framework that provides identification via body parts and estimation of body condition score. *Journal of dairy science*, 102(11), 10140-10151.