CSCI 370 Final Report

Team Stratom

Aaron Boyd
Adrian Mendez
Ethan Hickman
Konrad Quam

Revised December 12th, 2025

CSCI 370 Fall 2025

Professor Donna Bodeau

Table 1: Revision history

Revision	Date	Comments
Rev - 1	Dec 12, 2025	

Figure 4: Prediction with Obstruction

Table of Contents

Figure 4: Prediction with Obstruction

I. Introduction

The transportation of military cargo necessitates both effective and robust autonomous navigation systems. The dunnage detection system is a crucial development for Stratom's Autonomous Pallet Loader, denoted as APL. The Autonomous Pallet Loader is Stratom's proprietary forklift system capable of moving cargo over a variety of terrain types [1]. Goals outlined at the beginning of development include the creation of a fully-functional, RGB based detection pipeline running on the NVIDIA Jetson hardware; a real-time produced annotated image of the dunnage formation in visualization software; and, a published message containing the dunnage's pose information, as well as any other pertinent information [2]. This system is intended to be an eventual replacement of the current LiDAR-based perception pipeline currently used on the APL. Stratom is the specified client for this project. They are a robotics engineering company specializing in autonomous ground vehicles and ground vehicle solutions for the United States military. [1] Their specified needs include a more reliable, camera based method for identifying dunnage to support pallet movements, as their current LiDAR system is unreliable in scenarios involving obstructions and challenging elements [2]. There are no previous software revisions of this camera-based dunnage detection system, as the project is essentially starting from scratch. All training, validation, and testing data was collected by the project team using the supplied equipment from the client. The dataset used for training the dunnage detection system includes original and annotated images of various dunnage orientations in several locations to simulate both acceptable and unacceptable conditions. No external datasets exist for this specific task. Stratom is the primary stakeholder of the dunnage detection system, who will evaluate and potentially integrate it into their APL platform. Future student-led teams may extend this perception pipeline beyond the scope of this project.

II. Functional Requirements

The proposed system must detect dunnage within a range of 10 meters from the camera, which is mounted at a height between 0.5 and 1.5 meters off the ground. It should account for varying viewing angles and distances typically from the APL's operational environment. The system must also reliably identify the position and orientation, also known as pose, of each piece of dunnage to support further autonomous decision-making. The dunnage detection system must also reliably identify multiple pieces of dunnage within a single image frame, as multiple pieces of dunnage encompass one singular load of cargo. Each instance of dunnage should be recognized as a distinct object for the purposes of the system's output data.

Upon successful detection, the dunnage detection system should produce an annotated image that visually highlights each dunnage piece, annotations including the estimated pose of each dunnage beam overlaid on the camera's live feed. This image should also be viewable in RViz, so that real-time visualization of the model's verification is possible. In addition, the dunnage detection system should include a custom ROS2 message that includes a list of messages corresponding to any detected dunnage. These messages should include the timestamped position, orientation data, and any additional information, such as detection confidence scores, the model's inference time in milliseconds, and any other relevant metrics.

The dunnage detection system should use ROS2 parameters to maintain better development. These parameters include topic names, confidence thresholds for the detection algorithm, and any other relevant information found during the machine learning model's training. Each of these parameters should be stored with YAML files during the system's launch to ensure flexibility during testing and deployment. In addition, the system should implement SSO authentication for both secure user access and integration with Stratom's existing infrastructure.

Lastly, the dunnage detection system should report the image's inference time for every frame processed, measured in milliseconds, to facilitate further optimization down the line. This metric should be present in both the system's log output and published ROS2 message as stated previously, providing a real-time performance metric on the provided hardware.

Figure 4: Prediction with Obstruction

III. Non-Functional Requirements

The dunnage detection system should be developed in C++ or Python. In addition, the system should be developed with proper software development practices to maintain clarity. The software itself should exist within the ROS2 Humble framework so that the final product can easily integrate with existing robotics infrastructure at Stratom. All source code must be version-controlled and made available on GitHub under a specific project repository. This repository should include complete documentation, instructions for building the software, and any dependency files necessary to facilitate future work. Version control should include descriptive commits, branches, and pull requests, so that development history can easily be audited.

The system should use the RealSense D435 from Intel as its primary perception sensor, accessing its data from the package featured in ROS2. This package includes synchronized RGB and depth information. The RGB data should be the primary focus for perception, but the depth information will enhance pose estimation and model validation.

Deployment of the dunnage detection system should happen on an NVIDIA Jetson computer provided by Stratom. The software itself should function within a Docker container environment consistent with others at Stratom, ensuring that builds are reproducible. Machine learning models used in the dunnage detection system should be trained using frameworks licensed for commercial use. Stratom specified using the MIT license for their needs, so that produced work can easily integrate with other commercial products without causing licensing problems.

IV. Risks

Since this project integrates both hardware and software dependent systems, there are a number of technical and skills-related risks that should be identified and planned for. The following section identifies these key risks, evaluates their potential impact and likelihood, and outlines a proposed mitigation plan to minimize any potential consequences.

The first major risk outlined is hardware damage. Stratom provided the development team with both the Jetson and the RealSense camera, as well as several pieces of dunnage. All of these are essential to project development and would cause significant disruption to workflow if damaged. Mitigation includes following safe practices like avoiding testing during inclement weather and minimizing strain on cables used during testing.

Failing software is another major risk, specifically involving the Jetson architecture. Given that the development team only has the single device in their possession, most development takes place independently, meaning that there are some environmental differences, even within the Docker container. Therefore, successful runs on an x86 development machine may not work as intended on the Jetson. Mitigation for this risk includes an incremental deployment of various components to test for compatibility early on in development. In addition, limiting library usage to libraries with official Jetson support should limit dependency troubles and when dependency problems do arise, documentation should reflect that. Software failure is not limited to the Jetson architecture, however. Pose estimation failure is another likely and serious risk during development, especially given its complexity. Mitigation for pose estimation failure includes validating images directly with extrinsics before pose estimation happens. Manual validation, across the entire dataset, is time consuming and is not within the scope of this project.

Insufficient or poor training data is another significant risk, though not as likely as others already discussed. The model's performance does rely on diversity and volume, so limited or inconsistent annotations could result in a poor model. Mitigation for this risk includes scheduled, intentional, and frequent data collection sessions throughout development and following a standard for annotation in Label Studio during the annotation process. Lastly, time management and coordination during development is a significant risk. Lapses

Figure 4: Prediction with Obstruction

in communication and scheduling conflicts could lead to an uneven workload distribution and could disrupt the project's general workflow. Mitigation for this risk includes regular standups and meetings with both the development team and the client.

V. Definition of Done

The final delivery to the client must be able to output a live feed of annotated images, which show where the dunnage is in the image as well as its position in relation to the camera. In addition, pose-stamped messages must be published in ROS2, which should include the information on the position of the dunnage, specifically the distance from the camera to the dunnage and the angle of the dunnage compared with the ideal, head-on, position. The dunnage must be able to be detected up to a depth of 10 meters in a variety of different terrains, including concrete, gravel, sand, and tall grass. A demonstration of the functioning product must be presented to the client, and the accuracy of the dunnage recognition system must be reported to them.

VI. System Architecture

The dunnage detection system will run inside a docker container on an NVIDIA Jetson. The system architecture is organized into two major components or ROS2 nodes: the detection node and prediction node. Both nodes will receive and publish messages on multiple topics. Additionally, there is the ROS2 RealSense camera driver which will act as the input to the system providing messages on certain predefined topics. These components will work in series to form a dunnage prediction pipeline.

The first part of the system will be the RealSense camera driver. This driver will publish data to several topics. The topics actively used by the system are raw color, aligned depth to color, and camera info. The raw color and aligned depth to color topics will contain the color image and depth image data from the RealSense camera. The camera info topic will contain intrinsic camera information used by pose estimation in YAML format.

These topics will be subscribed to by the next component in our system: the detection node. The detection node will leverage a pretrained YOLOv4 Darknet 2D Bounding-Box detection model. The model is trained on an augmented dataset of dunnage and utilizes the raw color image from the RealSense driver. When a valid dunnage configuration is recognized by the model, it will pass along the original raw color, aligned depth to color, camera intrinsic data, and the YOLOv4 prediction output as a composite message called DetectionOutput. The camera intrinsics and prediction output will be serialized as JSON strings.

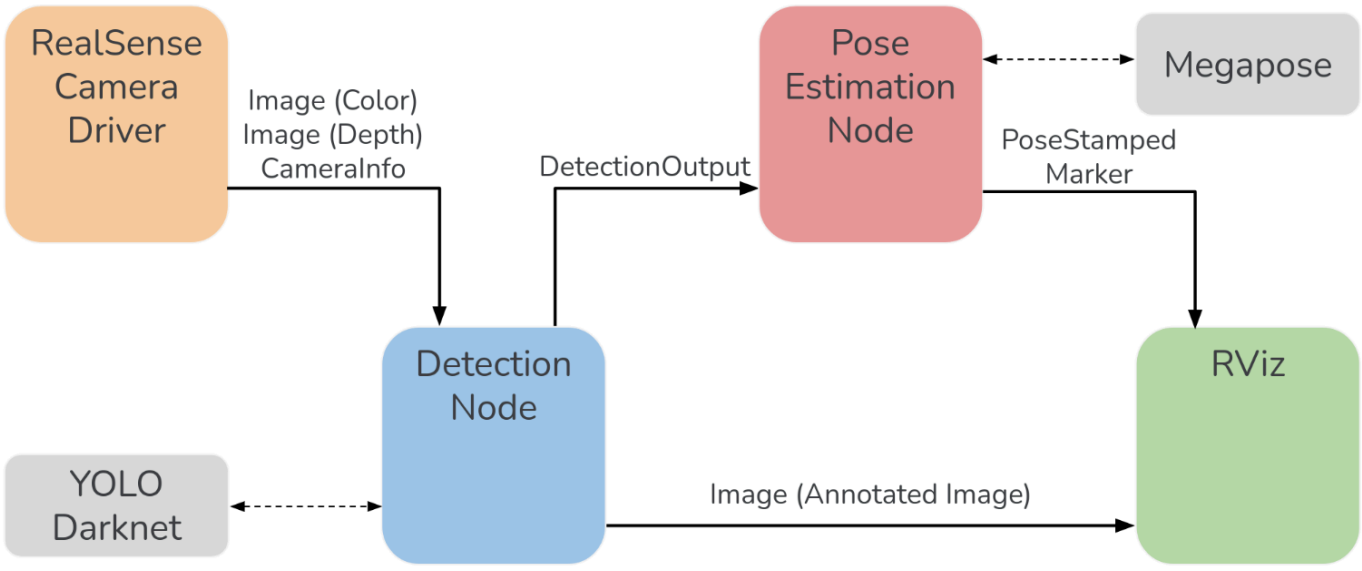

Figure 1: System Architecture Diagram

Figure 4: Prediction with Obstruction

The final node in the system is the pose estimation node which will subscribe to the output from the detection node. Additionally, it will read a 3D ply mesh of the dunnage to feed into Megapose alongside the detection node output. Megapose will then produce an output which will be published as a PoseStamped message that will be viewable in RViz. Additionally, the mesh will be published as a marker message so that RViz can render the predicted dunnage pose in 3D space.

VII. Software Test and Quality

For the Python functions in our project, unit tests will be used to ensure they function properly. The Python package Unittest will be used for this as it already has many built-in functions for unit testing, including functions to assert equality and truth. Then, to ensure the different components of our product work together smoothly, integration tests will be written and performed. These integration tests will encompass the combination of the dunnage recognition software, the position estimation software, and the ROS2 publishing nodes.

To ensure high-quality code, each team member will write and push code to a branch other than main, and the new code will have to go through a code review by one other team member before it can be merged to the main branch. Not only will this help prevent bugs in the code, but it will also lead to fewer headaches when integrating code written by different team members.

The team will also use code metrics to ensure that our code is maintainable and understandable, so that future applications of our software are as seamless as possible. Some metrics include tracking a maintainability index that factors complexity and the volume of code in a program, and test coverage. These metrics will not only ensure future maintainability of our code but also improve communication during the development process.

Then the team will use dynamic program analysis to examine how our software behaves during execution, focusing on code coverage, memory leak detection, and runtime error detection. Python includes the settrace mechanism that tracks various components of a program. This mechanism allows us to rapidly implement various testing techniques such as fuzz testing to identify potential bugs or memory leaks in our software.

The most important part of our quality control efforts relates to the dunnage detection system and the computer vision models that it encompasses. The quality of the dunnage detection computer vision model is primarily evaluated on the mean average precision and speed of inference. The YOLOv4 Darknet detection model and Megapose 6D pose estimation model have each been analyzed with certain metrics and performance on testing subsets of the data.

The YOLOv4 darknet utilizes a standard dataset of pre-labeled images containing dunnage in a specific format. The online labeling platform Roboflow enables an automatic splitting of the data into training, validation, and testing sets. During the training, the YOLO model adjusts its parameters based on the images of the dunnage so that it can detect the dunnage instead of just the generic dataset it was initially trained on. Also, the YOLO software outputs metrics for evaluation on the training and testing sets, which include the accuracy, precision, the F1-score, and helpful charts like the confusion matrix. All of these metrics should be used to evaluate the model after each training loop, and these same metrics will be calculated with the test split to ensure the model will continue to perform well with the new image feed it will see in live deployment. The model will be trained and potentially retrained until high scores are achieved across these metrics.

VIII. Project Ethical Considerations

There are a few key ethical concerns that the team has considered with respect to the dunnage detection system. The first and most pertinent ethical concern relates to principle 1.6 of the ACM Code of Ethics which states to respect privacy [3]. The data collection process of training a 2D or 3D image model involves collecting

Figure 4: Prediction with Obstruction

a significant number of images. It is important that no images of sensitive information or non-participating individuals are gathered. In order to adhere to this principle, the team focused on performing data collection in public areas with minimal human interference. Additionally, the images taken will be manually reviewed to ensure that no passerbys end up having their likeness used in training.

Another potential ethical dilemma is that of software licensing. Principle 1.5 of the ACM Code of Ethics refers to respecting the license agreements of creators [3]. Many existing machine learning models contain copyleft licenses which contain certain restrictions in regards to the redistribution of software. As such the team has decided to focus on using models with more permissive licenses and will communicate with the client as licensing issues arise.

The final main ethical concern relates to potential safety concerns as highlighted in Section 1.1 of the IEEE Code of ethics [4]. With any autonomous system, there are inherent safety issues. While the dunnage detection system is not the actual autonomous system itself, it is crucial that it provides accurate data and predictions in order to not endanger any nearby individuals or equipment.

Looking at the project and evaluating several aspects like common practice, professionalism, and legality helps ensure its ethical viability. For common practice the team ensures that standard practices are followed especially with respect to privacy and consent in data collection [5]. With respect to professionalism, the ACM and IEEE codes of ethics are followed with special attention to the principles previously mentioned [3][4]. Finally, the team uses models in a manner that is legally compliant with the existing license.

IX. Results

Data for the dunnage detection system was collected using the RealSense D435 camera connected to a Linux machine. The camera provided both RGB and depth information to help construct the dataset. Images were taken from a variety of angles and distances surrounding the dunnage configuration in several different locations, with different surfaces for each location. As development progresses, data collection will continue, so that the dataset grows larger and more diverse to achieve stronger yields. Future data collection sessions will prioritize capturing different lighting scenarios, camera angles and distances, and surface types. After images were collected for the dataset, each was annotated using Label Studio and collectively uploaded to Roboflow. The following augmentation techniques were then applied to the dataset within Roboflow: Flipping, Cropping, Brightness, and Exposure.

Figure 2: Dunnage Prediction

The initial stage of model training uses YOLOv4 Darknet, chosen for its fast performance and robust design. The training is performed on the Jetson utilizing the CUDA to accelerate the process. Even so, the process took over twelve hours with the relatively small dataset using transfer learning. Current options under consideration include the team's own personal machines and Google Colab's cloud based system utilizing GPU acceleration. Overall, the performance was good on the dunnage class but had some problems with invalid dunnage configurations. This is expected considering the scope of potential invalid cases along with the smaller amount of data containing invalid setups as opposed to valid configurations.

Figure 3: Invalid Case Prediction

Once the Darknet model was trained, it was then implemented within ROS2 and used for realtime detection. This process was relatively easy to implement.

Figure 4: Prediction with Obstruction

The next stage involved setting up a 6D-pose estimation software to predict the physical location of dunnage. This process was both time consuming and difficult. Several updated and well-documented pose estimation models exist but under various licenses that prove problematic for commercial applications. Additionally, many of the models with permissive licenses are unmaintained and have problems with modern dependency versions.

After significant trial and error, the team was able to utilize the Megapose 6D pose estimation model with some caveats. The accuracy was relatively good but requires further evaluation. The first issue is relatively minor in the long run and involves the pose output being consistently off in rotation. When adjusting RViz viewing angle, the prediction was relatively consistent with the orientation of the original image. This could be resolved by transforming the initial output of the model or potentially through a modification of the ply mesh itself.

Another concern was the inference time. The inference time on the NVIDIA Jetson was approximately ten seconds using megapose's rgb multihypothesis model. This is suboptimal for real-time robotics usage. Potential improvements could be made here by decreasing the number of points in the mesh given the relatively simple shape of the dunnage. Megapose by default requires at least two thousand points but this could theoretically be modified.

Finally, there were some problems with precision as inferences on certain frames would yield abnormal results. This was infrequent compared to the valid outputs but still occasionally occurred. A potential solution on this front would be to add a texture to the mesh.

X. Future Work

Future work on the dunnage detection system consists of several opportunities for extending and improving perception. One opportunity for further improvement consists of training the dataset with Isaac ROS, which is NVIDIA's own robotics framework. Isaac ROS uses TensorRT instead of Pytorch for inference acceleration, which would likely mean performance gains on the Jetson computer. This transition from FoundationPose to TensorRT may substantially decrease inference times. Another opportunity comes from continued data collection, even beyond the scope of the project's requirements. While the current dataset is adequate for initial training, it would benefit greatly from broader diversity across lighting, weather, and terrain conditions. Additionally, more images of invalid configurations of dunnage being added to the dataset would improve the model. The mean average precision for the invalid dunnage class was not particularly high and considering that there are many more invalid configurations than valid configurations, more images of invalid dunnage should be in the dataset than valid configurations. Like in prior data collection sessions, gathering images from various angles and distances would likely improve the model's detection confidence and accuracy.

Another option for future work for the dunnage detection system would be to evaluate alternative pose estimation architectures. Several options discussed at this time include FreeZe and CenterPose. These models use different strategies for pose regression, which could yield greater stability and increased accuracy. Comparing benchmarks from these differing architectures would help identify the most optimal balance of efficiency and accuracy for deployment on the Jetson.

Additionally, future work in automating the training pipeline, like using field data to continuously refine the model would allow the dunnage detection system to improve itself in real time. Combined with improved visualization tools within RViz, an automated pipeline could yield even stronger results, allowing future developers to toggle between detection and confidence maps at will.

Finally, the 6D pose estimation performance could be optimized to a significant degree. Manual modification of the megapose code could potentially enable it to utilize less points potentially improving inference time. Additionally, the precision of megapose could potentially be improved by adding textures

Figure 4: Prediction with Obstruction

alongside the mesh. While there are various different colors and materials used as dunnage, there are potential solutions. Assuming the material is uniform among the set of dunnage, the 2D detection node could be modified to detect the material of the dunnage as well. Then it could indicate to the pose estimation node which texture file to load for inference.

XI. Lessons Learned

Throughout the course of the project, the team learned several key lessons. First, the team spent a considerable amount of time experimenting with extrinsic camera measurements from the initial images taken of the dunnage configuration. Ultimately, these measurements have little relevance in the established workflow. In the future, the team plans to prioritize efforts that directly contribute to the model's performance rather than calibration, unless pose alignment requires them.

Second, the team underestimated the sheer volume of data that is necessary for a strong detection model. While the current dataset is functional and appropriate for current development, it is not varied enough for ideal detection confidence, especially in cases of invalid dunnage. In hindsight, collecting more data earlier and more frequently is a major lesson learned.

Third, while the results of the MegaPose 6D pose estimation generally gave consistent predictions of the dunnage configuration, there was a consistent, significant rotational offset between the pose estimation and the virtual representation of the ground. Reasons for this offset likely originate from an issue with the CAD model supplied to MegaPose, or, a disparity in how coordinates are handled within MegaPose's own architecture. Fixing this disparity proved too challenging given the time constraints of the project.

Finally, the team has learned the value of early and continuous benchmarking during the development process. By establishing a baseline earlier, with performance metrics like inference time, accuracy, and confidence, the team could have identified bottlenecks like the CPU-based training earlier in the development process. In future projects, the team will continue to understand the importance of iterative testing and validation throughout every stage of the workflow.

XII. Acknowledgments

The team would like to thank Stratom for providing the project and supplies for it. From Stratom, we would like to particularly thank Kristin, who was our point of contact at Stratom. She regularly met with us and assisted us when we were stuck, and she also gave us a tour of their facility. The team would also like to thank the Colorado School of Mines for bringing in this project for us and for connecting us with the client. In particular, Donna was a very helpful advisor who helped us get past roadblocks with her suggestions. Additionally, thank you to the rest of the adviser team who make these projects possible.

XIII. Team Profile

Konrad Quam

Aaron Boyd

Adrian Mendez

Ethan Hickman

Figure 4: Prediction with Obstruction

Konrad Quam is a junior computer science major in the data science track. Aaron Boyd is a senior computer science major in the computer engineering track. Adrian Mendez is a senior computer science major in the Business track. Ethan Hickman is a senior computer science major in the data science track.

References

[1] Stratom, Inc., "Autonomous Pallet Loader (APL™) Brochure," Louisville, CO, USA: Stratom, Inc., 2017.

[2] Stratom, Inc. and Colorado School of Mines, "Dunnage Detection – Kickoff and Requirements," Internal Project Document, Fall 2025.

[3] Association for Computing Machinery, "ACM Code of Ethics and Professional Conduct," Association for Computing Machinery, Jun. 22, 2018. https://www.acm.org/code-of-ethics

[4] IEEE, "IEEE Code of Ethics | IEEE," Ieee.org, 2020. https://www.ieee.org/about/corporate/governance/p7-8

[5] "Ethical and Privacy Considerations in Image Processing – csbranch.com," Csbranch.com, 2025. https://csbranch.com/index.php/2024/11/23/ethical-and-privacy-considerations-in-image-processing/

[6] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," Science Robotics vol. 7, May 2022.

Appendix A – Key Terms

Term	Definition
Autonomous Pallet Loader (APL)	*The Autonomous Pallet Loader (APL) is a proprietary product created by Stratom designed to move cargo over varied terrain utilizing Stratom's Summit Software. [1]*
Isaac ROS	*A collection of NVIDIA computing packages designed to develop advanced AI robotics applications.*
Label Studio	*Label Studio is an open source data labeling tool that was used to prepare images for model training and testing.*
Roboflow	*Online data labeling platform that enables automatic augmentation, splitting, and exporting to various formats*
ROS2	*The Robot Operating System (ROS) is a collection of software libraries and tools for building robot applications.ROS2 is the modernized iteration of the original. [6]*
Node	*A self-contained unit running in the ROS2 system*
Topic	*The mechanism by which different nodes communicate with each other. Nodes can either publish or subscribe to various*

Figure 4: Prediction with Obstruction

	topics. Can be thought of as analogous to radio or television channels
Message	*The data being sent or received on a given topic.*
RViz	*RViz is a 3D visualizer for the ROS framework. (Citation Needed)*
Single Sign On (SSO)	*Single Sign On (SSO) allows users to authenticate with a single user ID when trying to access some form of secure software.*
Yet Another Markup Language (YAML)	*YAML is an easily-readable programming language used for writing configuration files.*
JavaScript Object Notation (JSON)	*JSON is a standard format used to store and transfer data*
You Only Look Once (YOLO)	*An image detection method that creates bounding boxes around various classes in a model.*
YOLOv4 Darknet	*The particular version and implementation of YOLO used in this system*
Megapose	*The 6D pose estimation system leveraged by the system for pose estimation*

Figure 4: Prediction with Obstruction