Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

REAL-TIME IMAGE PROCESSING AND EMULATION FOR INTELLIGENT WAREHOUSE AUTOMATION

Sumant Joshi[1], Saurabh Lulekar[1], Tarciana Almeida[1], Abhineet Mittal[1], and Ganesh Nanaware[1]

[1]Worldwide Design and Engineering, Amazon, Seattle, USA

ABSTRACT

Material flow simulation and emulation are essential tools used in warehouse automation design and commissioning, to create a digital twin and validate equipment control logic. The current emulation platforms lack an internal computer vision (CV) toolkit which poses a challenge for emulating vision-based control system behavior which requires real-time image processing capability. This paper addresses this gap by proposing an innovative framework that utilizes a bridge between Emulate3D and MATLAB to establish real-time bidirectional communication to emulate vision-based control systems. The integration enables transfer of visual data from Emulate3D to MATLAB, which provides CV toolkit to analyze vision data and communicate controls decisions back to Emulate3D. We evaluated this approach to develop a small-footprint package singulator (SFPS) and the results show that SFPS achieved target throughput with 45% improvement in singulation accuracy over conventional singulators with 64% less footprint and eliminating the need for gapper equipment required with conventional singulators.

1 INTRODUCTION

The material flow in modern warehouse progresses from inbound receiving through staging, sortation, storage, and finally to outbound processing and loading. While robotics and automated workcells can handle many of these operations, their high capital cost demands thorough system validation before implementation. Simulation and emulation serve as cost-effective tools to evaluate system performance, test volume surges, identify bottlenecks, and assess operational impacts. However, the reliability of these evaluations heavily depends on creating high-fidelity models, particularly for robotic systems.

A robotic system comprises three core components (Figure 1): robot hardware, sensors, and a control system. The hardware encompasses the physical elements - frame, linkages, actuators, and end effectors - that execute movement. Sensors capture crucial data including kinematic information, object detection, and visual attributes. Acting as the systems brain, the control system processes sensor data through AI/ML algorithms and communicates commands to PLCs/IPCs for actuator control.

The current suite of material handling equipment (MHE) emulation software excels at modeling detailed robot hardware using 3D CAD data and accurate kinematics. Basic sensor data, such as photoeye based object detection, encoders for speed, and position measurement can be directly emulated. Similarly, actual PLCs can be connected for low-level control implementation. However, complex data processing - including vision camera images (Figure 2) and depth point clouds (Figure 3) requires ML based feature extraction. This capability is typically unavailable in MHE emulation software, making real-time integration with specialized image analysis tools essential for true emulation.

Figure 2 shows example of robot vision camera image where the robot has to determine pick sequence from bulk based on various parameters such as package type, stacking over other packages, package bounding box, shape, etc. The MHE emulation can generate similar package presentation as shown in side comparison. The emulation generated images (Figure 2) serve as input to ML algorithms that influence robot movements.

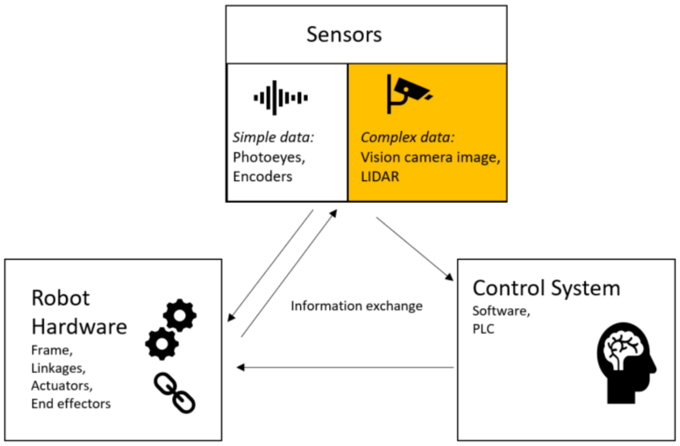

Sensors
Simple data: Photoeyes, Encoders
Complex data: Vision camera image, LIDAR
Robot Hardware
Frame, Linkages, Actuators, End effectors
Information exchange
Control System
Software, PLC

Figure 1: High-level building blocks of robotic systems.

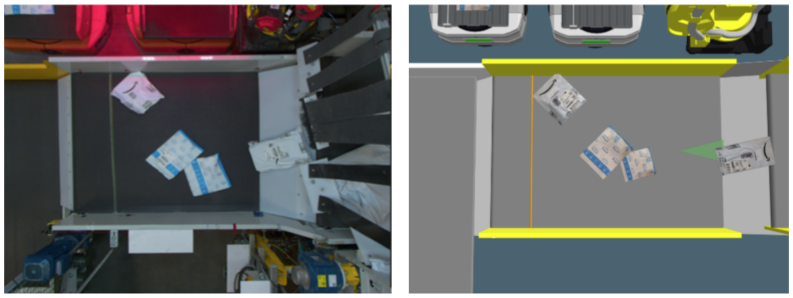


Figure 2: Vision camera - actual vs emulation image.

Figure 3 (Banner Engineering 2025) depicts an example of an automated package sortation process to a cart and determining accurate cart fill level is a key metric to decide when to close the container. A vision camera captures point cloud based on package stacking which is analyzed to determine accurate fill level. Recreating this in emulation/simulation requires real-time image processing capability for testing various container fill scenarios. Without integrated computer vision toolkit, we have to rely on simulation sensor component as abstraction for vision system to get attributes such as package coordinates, orientation, stacking of packages on other packages in bulk flow, etc. This gap in vision parsed data based on CV algorithm (deployed in actual system) vs sensor data from simulation impacts model fidelity and can be addressed using the proposed architecture in this paper.

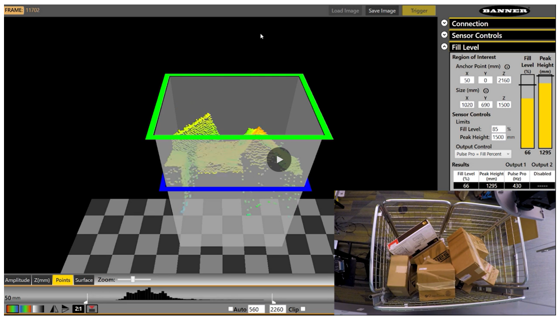


Figure 3: Determining accurate container fullness based on point cloud.

2 RELEVANT RESEARCH

Digital twin technology has been used as a powerful tool for validating MHE in a warehouse (Pandey et al. 2023). These virtual replicas of physical systems offer numerous benefits, including calculating throughput, evaluating control logic, testing mechanics and design layouts and analyzing product flow. Research indicates that project life-cycle can significantly be improved by integrating digital twins, leading to system efficiency while reducing commissioning risks for live systems (Omeragic and Sokic 2020).

Simulation refers to a mathematical model that abstracts system behavior based on certain rules and test multiple what-if scenarios. Emulation refers to recreating system behavior as close as possible to a real deployment. Simulation is helpful for large scale model building in short time and test hypothetical scenarios, while emulation is critical where accurate system behavior is expected and generally takes more time in model development than simulation (Zhang et al. 2012).

Real-time image processing for warehouse automation is an area that has been explored in several studies. For instance, (Wang and Li 2023) developed a clustering technique using the YOLO algorithm to identify similarities in product features. (Weichert et al. 2014) presented an actuator coupling system that uses image processing to gather and group packaging identification data in automated pick and place systems. (Suemitsu et al. 2022) created a training model to identify optimal sequences of activities for automated vehicle-picking systems. These ML-driven approaches also helped improve efficiency in material handling processes, leading to more intelligent and adaptive warehouse systems.

The literature review indicates that there are multiple advancements in digital twin methodology as well as use of image processing in domain of warehouse applications. However, there is opportunity to integrate the benefits of both these technologies within digital twin models that will enable emulation of vision systems for warehouse automation.

3 SIMULATION VS EMULATION VS VISION-INTEGRATED EMULATION

Warehouse design engineers rely on Factory Acceptance Tests (FAT) along with digital modeling tools such as simulation and emulation for evaluating automation technologies and material flow analysis. Figure 4 shows the phases involved in warehouse design life cycle and compares the impact of simulation approach vs emulation approach vs proposed vision-integrated emulation approach on each phase.

The first phase is the design phase where conceptual planning is done about system flow and different process paths. Once the requirements are known, Request of Proposal (RFP) is submitted to vendor. During FAT phase, the subsystem FAT is conducted to evaluate if it can meet all specifications. Once the FAT results data are available, simulation can be correlated for ensuring confidence in model results. Emulation can be time consuming and one-time effort for a frequently deployed robotic module, however once the emulation is setup, it can eliminate need for onsite testing. Vision-integrated emulation provides high fidelity behavior of vision system and eliminates need of simplified logic for extracting attributes based on vision data. For control logic validation phase, simulation can be used for combining subsystems and run a large scale model in short time. Emulation can be used for connecting to PLC and Warehouse Management System. The vision-integrated emulation can be used to mimic production system behavior where external services that use AI/ML can parse vision data in real-time and send controls feedback to emulation model. The commissioning phase involves actual system build based on results from digital warehouse model. From continuous improvement perspective, simulation takes least cost to run analysis, while emulation and vision-integrated emulation require high upfront resources for one-time development but lower cost for troubleshooting, operator training and significant savings on travel and controls software development timeline. The choice of modeling approach depends on level of input details required for each subsystem and expected accuracy of results.

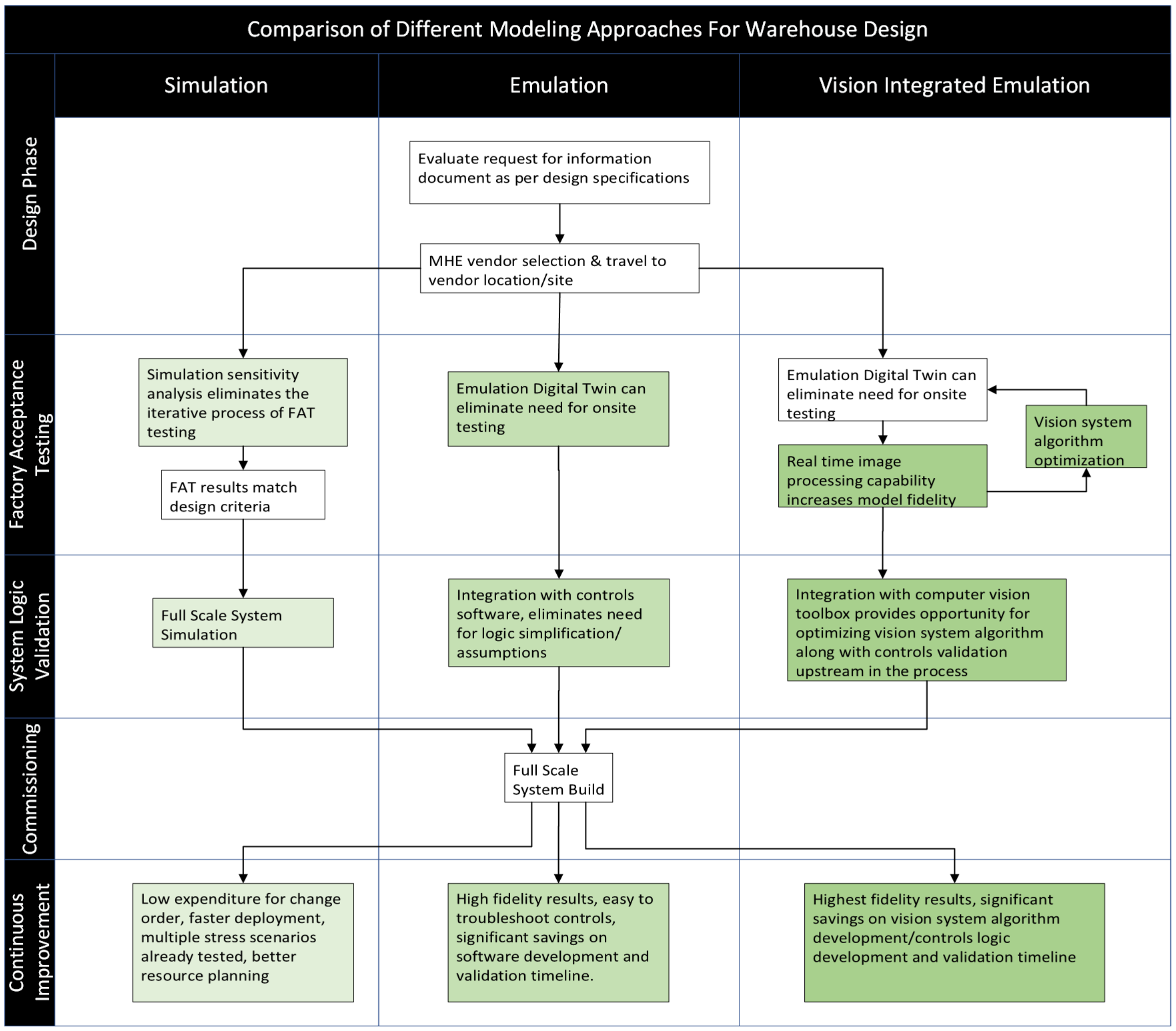

Comparison of Different Modeling Approaches For Warehouse Design
Simulation
Emulation
Vision Integrated Emulation
Design Phase
Factory Acceptance Testing
System Logic Validation
Commissioning
Continuous Improvement
Evaluate request for information document as per design specifications
MHE vendor selection & travel to vendor location/site
Simulation sensitivity analysis eliminates the iterative process of FAT testing
FAT results match design criteria
Emulation Digital Twin can eliminate need for onsite testing
Emulation Digital Twin can eliminate need for onsite testing
Vision system algorithm optimization
Real time image processing capability increases model fidelity
Full Scale System Simulation
Integration with controls software, eliminates need for logic simplification/ assumptions
Integration with computer vision toolbox provides opportunity for optimizing vision system algorithm along with controls validation upstream in the process
Full Scale System Build
Low expenditure for change order, faster deployment, multiple stress scenarios already tested, better resource planning
High fidelity results, easy to troubleshoot controls, significant savings on software development and validation timeline.
Highest fidelity results, significant savings on vision system algorithm development/controls logic development and validation timeline

Figure 4: Simulation vs emulation vs vision-integrated emulation comparison for warehouse design.

4 VISION-INTEGRATED EMULATION ARCHITECTURE

The architecture consists of three main components: 1) Emulate3D for virtual MHE simulation/emulation, 2) MATLAB Interface Layer with Image Processing Toolbox, and 3) Continuous Feedback Loop as shown in Figure 5. The Emulate3D component handles 3D modeling, physics simulation, and control system integration. The MATLAB interface layer serves as a bridge for data conversion and offers multiple libraries for computer vision tasks. The feedback loop creates real-time data transfer between Emulate3D and MATLAB which enables vision based emulation.

4.1 Emulate3D for Virtual MHE Modeling

The process begins with capturing image from the Emulate3D after every fixed time interval, which are then processed through MATLAB's image processing pipeline (Step 1 in Figure 6). The image is captured using camera catalog in Emulate3D. The image is then stored on local hard drive. The camera is adjusted in such a way that it captures the entire line of sight similar to production system. The frequency at which images are captured can be tuned to production camera frame per second specification (as shown in Figure 5 continuous feedback loop block). The next generated image can be overwritten onto the previous image

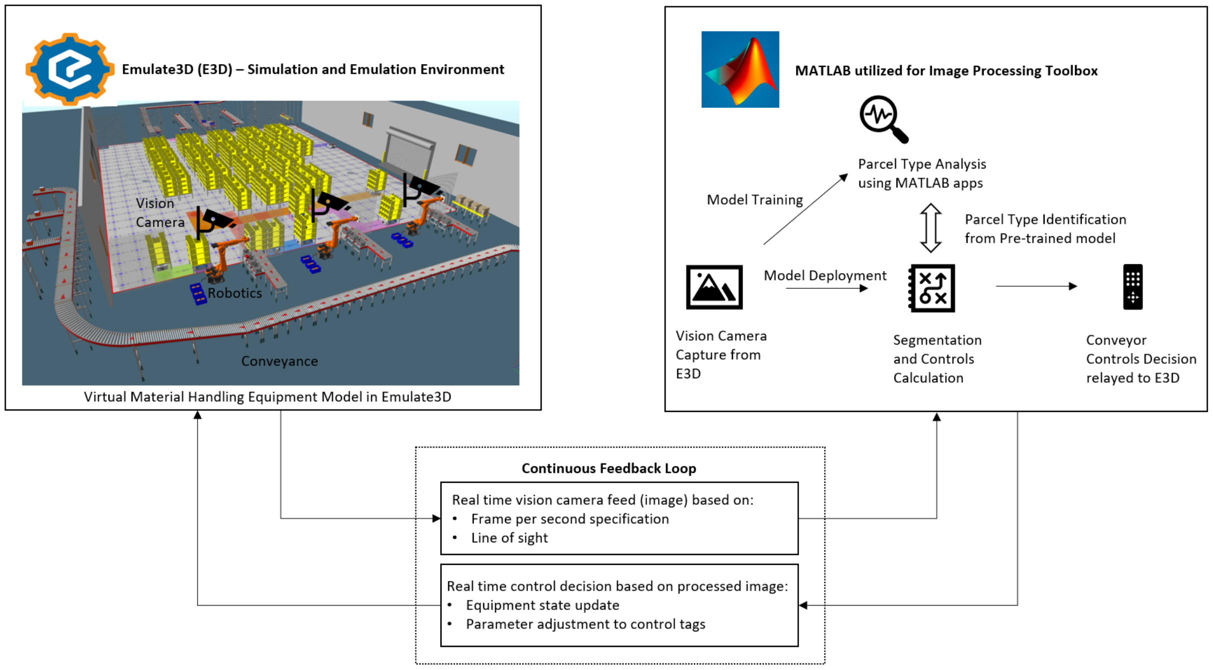

Emulate3D (E3D) – Simulation and Emulation Environment
Vision Camera
Robotics
Conveyance
Virtual Material Handling Equipment Model in Emulate3D
MATLAB utilized for Image Processing Toolbox
Model Training
Parcel Type Analysis using MATLAB apps
Parcel Type Identification from Pre-trained model
Model Deployment
Vision Camera Capture from E3D
Segmentation and Controls Calculation
Conveyor Controls Decision relayed to E3D
Continuous Feedback Loop
Real time vision camera feed (image) based on:
• Frame per second specification
• Line of sight
Real time control decision based on processed image:
• Equipment state update
• Parameter adjustment to control tags

Figure 5: Vision-integrated emulation architecture.

to optimize hard drive storage space. The same image file storage location is used in MATLAB for further image acquisition and processing.

4.2 MATLAB for Image Processing

The MATLAB Image Processing Toolbox is used for image pre-processing and segmentation. We use MATLAB Color Thresholder app to create binary segmentation masks based on different color spaces (MathWorks 2025). We collected simulation images showing various package combinations (based on package different material type, shape, etc.) and used this dataset to train and develop image processing algorithm. Information on the model training and validation is controlled due to proprietary reasons.

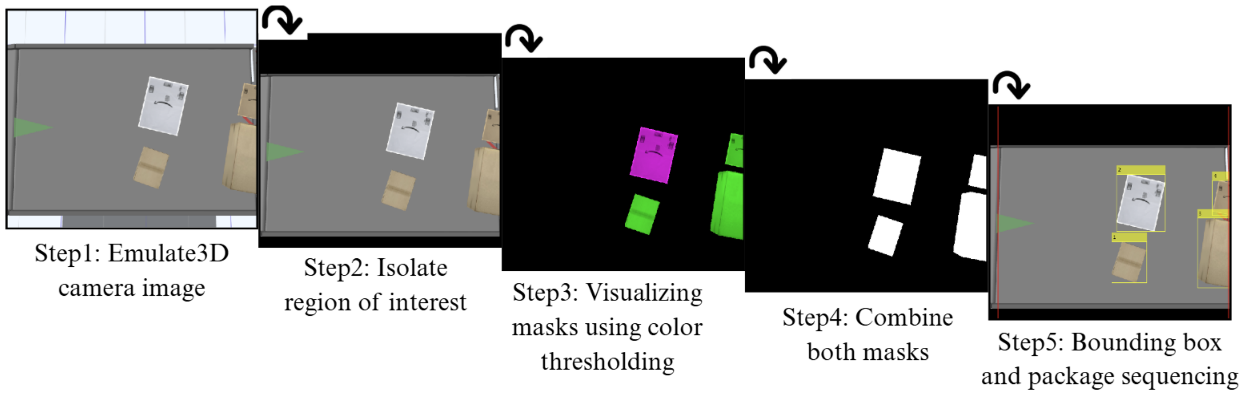

Step1: Emulate3D camera image
Step2: Isolate region of interest
Step3: Visualizing masks using color thresholding
Step4: Combine both masks
Step5: Bounding box and package sequencing

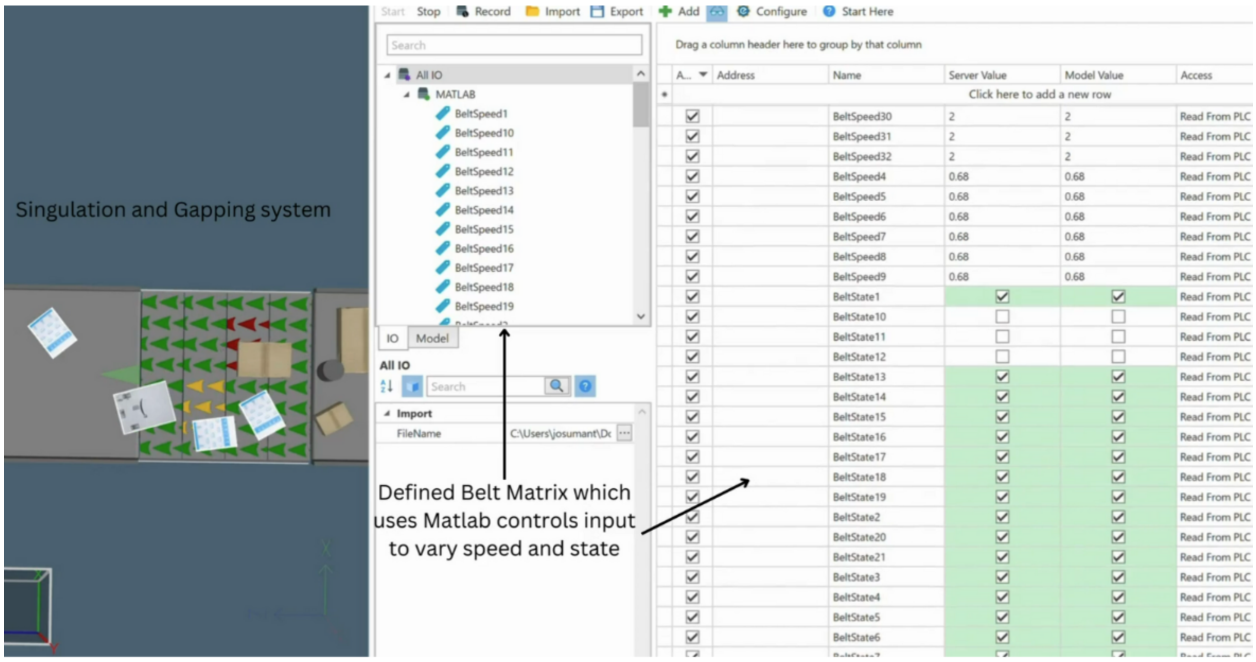
Figure 6: Image processing using color thresholding technique.

Following image acquisition from Emulate3D (Step 1 in Figure 6), the pre-processing stage encompasses two main areas: image enhancement and geometric operations. Image enhancement includes contrast adjustment, brightness correction, sharpening/blurring operations, and color space conversions between formats like RGB and HSV. Color space conversion between RGB and HSV transforms colors between two different representation systems. RGB uses three values (red, green, blue) ranging from 0-255 to create colors by mixing, making it ideal for digital displays. HSV represents colors using Hue (color type, 0-360°), Saturation (color intensity, 0-100%), and Value (brightness, 0-100%), making it more intuitive for human interpretation and image processing tasks. While RGB is better for display and storage purposes, HSV is preferred for color selection, image segmentation, and object detection (Step 3 in Figure 6). We have used HSV color space in Color Thresholder app for our case study. The geometric operations handle tasks such as scaling, resizing, rotation, translation and cropping functions. These two initial stages form the foundation for all subsequent image processing tasks by ensuring the input images are properly acquired and optimized for further analysis.

Image segmentation is a mid-level vision task, where the input is a preprocessed image as shown in step 4 of Figure 6. The output could be image attributes such as a set of points representing the edges of object. The boundaries of objects can be defined based on discontinuity (e.g. abrupt changes in intensity) or similarity (e.g. color, texture) as shown in step 5 of Figure 6.

4.3 Emulate3D-MATLAB Continuous Feedback Loop

The continuous feedback loop establishes bi-directional communication of Emulate3D with MATLAB. For control logic development, we used MATLAB code editor that uses output from processed image. Emulate3D provides connectivity to MATLAB using Tag Browser feature that enables mapping of virtual equipment attributes to control tags in MATLAB as shown in Figure 7. The information that flows from Emulate3D to MATLAB is image data, and from MATLAB to Emulate3D, it is the control decision in form of on/off state and parameter values.



FileName C:\Users\josumant\Dc

A...	Address	Name	Server Value	Model Value	Access
✓		BeltSpeed30	2	2	Read From PLC
✓		BeltSpeed31	2	2	Read From PLC
✓		BeltSpeed32	2	2	Read From PLC
✓		BeltSpeed4	0.68	0.68	Read From PLC
✓		BeltSpeed5	0.68	0.68	Read From PLC
✓		BeltSpeed6	0.68	0.68	Read From PLC
✓		BeltSpeed7	0.68	0.68	Read From PLC
✓		BeltSpeed8	0.68	0.68	Read From PLC
✓		BeltSpeed9	0.68	0.68	Read From PLC
✓		BeltState1	✓	✓	Read From PLC
✓		BeltState10	☐	☐	Read From PLC
✓		BeltState11	☐	☐	Read From PLC
✓		BeltState12	☐	☐	Read From PLC
✓		BeltState13	✓	✓	Read From PLC
✓		BeltState14	✓	✓	Read From PLC
✓		BeltState15	✓	✓	Read From PLC
✓		BeltState16	✓	✓	Read From PLC
✓		BeltState17	✓	✓	Read From PLC
✓		BeltState18	✓	✓	Read From PLC
✓		BeltState19	✓	✓	Read From PLC
✓		BeltState2	✓	✓	Read From PLC
✓		BeltState20	✓	✓	Read From PLC
✓		BeltState21	✓	✓	Read From PLC
✓		BeltState3	✓	✓	Read From PLC
✓		BeltState4	✓	✓	Read From PLC
✓		BeltState5	✓	✓	Read From PLC
✓		BeltState6	✓	✓	Read From PLC

Figure 7: MATLAB controls input to E3D tag browser.

5 CASE STUDY

5.1 Problem Statement

The challenge was to develop a compact yet efficient package singulation system, as existing conventional solutions either required too much space or couldn't achieve desired throughput rate. To address this, we designed a conveyor belt matrix system where each belt could be controlled independently and a vision camera system tracks incoming package positions as shown in Figure 8. The controls code was implemented in MATLAB, which processed the visual data and applied appropriate control actions based on the identified package characteristics.

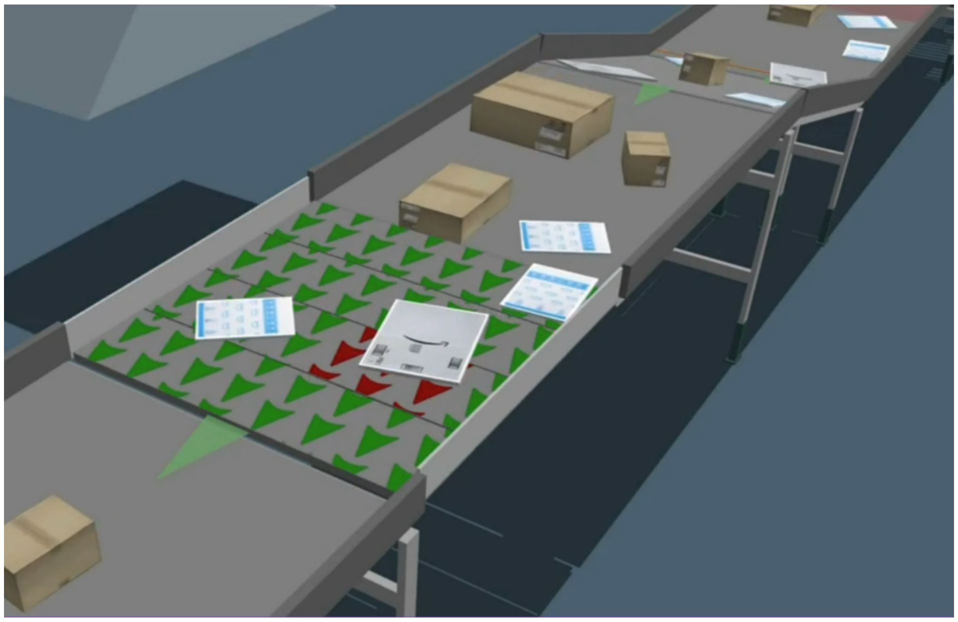


Figure 8: Small-footprint package singulator.

5.2 Methodology

The singulator emulation model using Emulate3D and MATLAB was developed to identify system throughput and refine singulation/gapping logic. Emulate3D generates 2D images at a frequency of 50 ms. These images are imported into MATLAB environment for image processing. Once the individual packages are identified, the centroid and leading/trailing edge of these packages are captured with reference to belt. Figure 9 shows the dimensions A, B, L, W, X, Y and Z captured for every image generated for all packages within line of sight. Using these measurements, control logic built in MATLAB adjusts individual belt speeds to create spacing between packages.

The vision system captures the entire multi-belt section and part of the infeed belt in each image. As packages enter the multi-belt region, they are labeled sequentially, starting from the infeed side. The control logic works backwards, beginning with the package closest to outfeed, measuring gaps between consecutive packages. For instance, it calculates the space between packages by measuring the distance between one package's trailing edge and the next package's leading edge [like (Z4 – (Z3 +W3))]. The system follows a simple decision-making process: if the gap between packages meets or exceeds the required threshold, the belt speeds remain unchanged. However, when the system detects packages that are too close together (gap below threshold), it identifies their location using the row number corresponding to the package's center point. For example, when packages 3 and 4 are side by side, the system identifies Row4 and specifically targets belt (4,4) for speed adjustment as shown in step 1 of Figure 10.

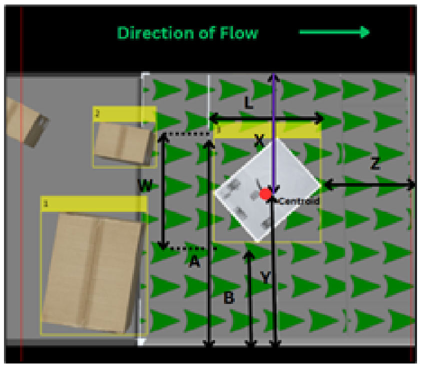


Figure 9: Example of package dimensions captured from image.

The system analyzes package dimensions to determine which conveyor belts need speed adjustments. First, it checks package height against the standard belt width of 0.5ft. When a package is taller, like in our example, the system includes adjacent belts on either side of the center belt. For instance, with the package centered on belt (4,4), it expands control to belts (4,3:5) as shown in step 2 of Figure 10.

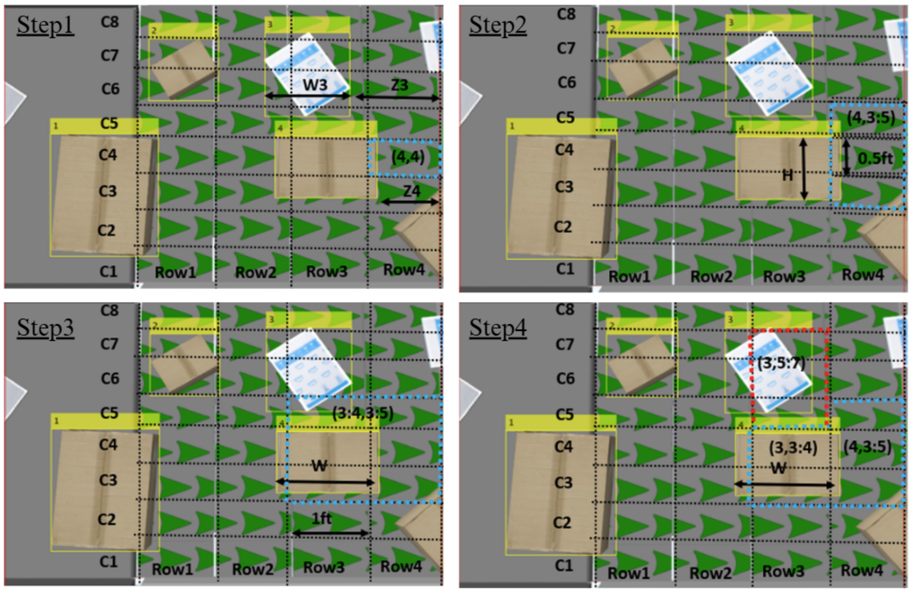


Figure 10: Step-by-step visualization of package singulation logic.

Next, the system examines package width relative to the standard belt length of 1ft. For wider packages, it includes belts in the row behind the center point. In our example, this expands control to belts (3:4, 3:5), creating a larger control zone as shown in step 3 of Figure 10.

The system then checks for multiple packages in the same row. In this case, package 3 and package 4 both fall in row 3. For package 3, if it's taller than 0.5ft, the system controls belts (3,5:7). Similarly, for package 4, it controls belts (3,3:4) and (4,3:5) as shown in step 4 of Figure 10. Based on the measured gap between packages, these belt groups either operate at different speeds or some may temporarily stop to maintain proper spacing. This analysis and adjustment process repeats continuously for all packages in view, ensuring proper spacing through dynamic speed control.

5.3 Mathematical Modeling

Model Variables: A planar image can be represented by a two-dimensional function f (X, Y). The value of f at the spatial coordinates (X, Y) is positive and it is determined by the source of the image. If the image is generated from a physical process, its intensity values are proportional to the energy radiated by a physical source. Therefore, f(X, Y) must be nonzero and finite: $0 \leq f(X,Y) \leq \infty$. A digital image is represented as a M × N matrix

$$\begin{bmatrix} f_{0,0} & f_{0,1} & \cdots & f_{0,N-1} \\ f_{1,0} & f_{1,1} & \cdots & f_{1,N-1} \\ \cdots & \cdots & \cdots & \cdots \\ f_{M-1,0} & f_{M-1,1} & \cdots & f_{M-1,N-1} \end{bmatrix}.$$

The mathematical relation that can be established between gray-level resolution and bits per pixel can be given as

$$L = 2^k (k > 0).$$

In this equation, L refers to number of gray levels. It can also be defined as the shades of gray. k refers to bpp or bits per pixel. The 2 raised to the power of bits per pixel is equal to the gray level resolution. Many times, digital images take values 0, 1, 2, ..., 255, thus 256 distinct gray levels (Young et al. 1998).

From the above construction, we have $L_{min} \leq L(X,Y) \leq L_{max}$, where L(X, Y) is the gray-level at coordinates (X, Y) within interval $[L_{min}, L_{max}]$. The intermediate values are shades varying from black to white. With image denoising, the linear degradation model becomes

$$g(X,Y) = L(X,Y) + n(X,Y).$$

For the case study, the conveyance system can be represented as a matrix of belts C(r,c) with r rows and c columns. The individual conveyor speeds range from 0 to T_{max} FPM (feet per min). The conveyor velocity matrix can be represented as

$$V_{r,c} \in [0, T_{max}].$$

Package Identification and Feature Extraction: MATLAB Color Thresholder app converts the grayscale image L(X, Y) to binary image BW(L, level), by replacing all pixels in the input image with luminance greater than threshold level with the value 1 (white) and replacing all other pixels with the value 0 (black). The binary image BW is a black and white image in order to prepare for boundary tracing. This binary image is preprocessed using morphology functions to remove pixels which do not belong to the objects of interest, fill any gaps or holes to estimate the area enclosed by each of the packages. By looping over the detected boundaries of packages and centroid locations, let x(t) represent the position of a package on a conveyor belt at time t.

Pseudo Logic for Package Singulation: We need to consider the dynamics of package movement and separation to formulate a differential equation (refer Figure 11). We can describe the system using second-order differential equation

$$F(t) = m(dx/dt)^2 + \mu.N(dx/dt)$$

where, *F(t)* = forces acting on the package;
m = mass of package;
μ = coefficient of friction;
N = normal force.
Let *g(t)* = dynamic gap between consecutive packages.

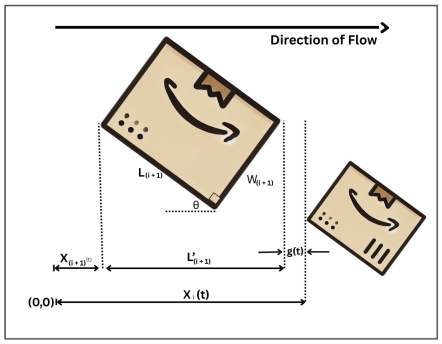


Figure 11: Representation of consecutive packages in direction of flow.

In order to achieve a desired gap G, the complete system could be described as

$$F(t) = m(dx/dt)^2 + \mu.N(dx/dt) + K(g(t) - G).$$

The gap between consecutive packages at time t is

$$g(t) = x_i(t) - x_{i+1}(t) - L'_{i+1}$$

where, K = proportional gain;
L' = apparent length between leading and trailing edge of the package;

$$L'_{i+1} = L_{i+1} \mid cos\theta \mid + W_{i+1} \mid cos(90 - \theta) \mid$$

L = actual length of the package;
W = actual width of the package;
θ= orientation angle of the package relative to the direction of travel.

Hence,

$$F(t) = m(dx/dt)^2 + \mu.N(dx/dt) + K(x_i(t) - x_{i+1}(t) - L_{i+1}cos\theta + W_{i+1} \mid cos(90 - \theta) \mid - G).$$

The belt speed is changed only if the consecutive gap between packages is less than the desired gap G.

Control Belt Matrix Identification: Let C(r,c) represent matrix of independent conveyors with r rows and c columns. Based on the consecutive package location $x(t)$ and $x_{i+1}(t)$ in the direction of flow, two groups of conveyors

$$C[n_i X_{m_i}],$$

and

$$C_{i+1}[n_{i+1} X_{m_{i+1}}]$$

are identified by solving the control component from described differential equation

$$K(g(t) - G) \geq 0.$$

The identified conveyor belt speeds are dynamically changed until the gap between consecutive packages is greater than or equal to desired gap G.

5.4 Outcome

The conventional singulators show throughput in range of 4500 to 5500 packages per hour (PPH) depending on package mix with ~90% singulation accuracy. For our application, our target rate was 7000 PPH but actual tests showed the singulation accuracy dropped significantly to ~50%. Using vision-integrated emulation (Table 1), we were able to precisely track each package and control individual matrix belt which increased our singulation accuracy to 95%. This approach eliminated need for a gapper equipment that is required for operation with conventional singulators and reduced singulator footprint by 64%. We were able to develop high fidelity digital twin, determine optimal belt matrix configuration (8 x 4 belts) to meet 7K PPH target throughput and develop controls software without need for physical machine, thus significantly optimizing commissioning timeline.

Table 1: Comparison of conventional singulators with small-footprint singulation and gapping system.

Attribute	Conventional singulators		Small-footprint singulation and gapping system
System Throughput (PPH)	4500-5500	7000	7000
Singulation Accuracy	~90%	~50%	95%
Footprint (Length in ft)	27-34		11
Gapper Requirement	Yes (9ft length)		No

6 CHALLENGES

Current simulation tools struggle to fully replicate the complexities of real-world package handling. At the physical level, they cannot accurately model package deformation and soft body physics, mechanical vibrations, or critical environmental factors such as temperature, humidity, and static electricity. Vision system simulation poses another set of challenges. Current tools inadequately render realistic lighting conditions, shadows, and reflections, while failing to accurately represent camera characteristics like sensor noise, lens distortion, and motion blur. Environmental influences such as dust, vibration, and changing ambient light conditions are often oversimplified or ignored entirely. The limitations extend to processing and performance aspects as well. Real-time processing delays, edge detection accuracy, and pattern matching challenges are not realistically simulated. System integration aspects, particularly communication latency and hardware timing, are often oversimplified. Perhaps the most challenging are the real-world scenarios that resist accurate modeling - variable package orientations, damaged labels, and partial occlusion cases. These challenges are beyond emulation scope and cannot be accurately modeled.

7 CONCLUSION

Warehouse design is a complex process that integrates various automation technologies. Simulation and emulation tools play a crucial role in speeding up the planning phase, ensuring a high-quality launch with minimal defects. The accuracy of these digital models is essential for making informed decisions about capital investments and operations planning. While simulation models typically rely on simplified representations of software and control logic, emulation models take a step further by integrating actual controls and software to assess system performance more realistically. In this paper, we explored the components of robotic and automation systems that can be modeled with high fidelity using existing standalone software packages. However, we identified a significant gap in vision system modeling within current emulation tools, necessitating integration with external software for comprehensive system representation. To address this, we presented an architecture and case study demonstrating how emulation fidelity can be enhanced for more effective controls software development and validation. The integration of image processing capabilities with simulation and emulation software opens up numerous opportunities for improving warehouse automation deployment.

REFERENCES

Banner Engineering 2025. "ZMX Bin Fill Demo Video". https://www.bannerengineering.com/us/en/support/videos/3d-time-of-flight-time-lapse.html, accessed 08.19.2025.

MathWorks 2025. "Color Thresholder App". https://www.mathworks.com/help/images/ref/colorthresholder-app.html, accessed 08.19.2025.

Omeragic, E., and E. Sokic. 2020. "Counting Rectangular Objects on Conveyors Using Machine Vision". In *2020 28th Telecommunications Forum*. November 24th-25th, Belgrade, Serbia.

Pandey, A., R. Flam, R. Mohammed, and A. Kalidindi. 2023. "Emulation and Digital Twin Framework for The Validation of Material Handling Equipment in Warehouse Environments". In *2023 Winter Simulation Conference (WSC)*, 3250–3261 https://doi.org/10.1109/wsc60868.2023.10408646.

Suemitsu, I., H. K. Bhamgara, K. Utsugi, J. Hashizume, and K. Ito. 2022. "Fast Simulation-Based Order Sequence Optimization Assisted by Pre-Trained Bayesian Recurrent Neural Network". *IEEE Robotics and Automation Letters* 7(3):7818–7825.

Wang, F., and X. Li. 2023. "Research on Warehouse Object Detection for Mobile Robot Based on Yolov5". In *2023 International Conference on Image Processing, Computer Vision and Machine Learning*. November 3rd-5th, Chengdu, China.

Weichert, F., A. Böckenkamp, C. Prasse, C. Timm, B. Rudak, K. Hölscher *et al.* 2014. "Towards Sensor-Actuator Coupling in an Automated Order Picking System by Detecting Sealed Seams on Pouch Packed Goods". *Journal of Sensor and Actuator Networks* 3(4):245–273.

Young, I. T., J. J. Gerbrands, and L. J. van Vliet. 1998. *Fundamentals of Image Processing*. 2.3 ed. Delft: Delft University of Technology.

Zhang, J., V. Le, M. Johnston, S. Nahavandi, and D. Creighton. 2012. "Discrete Event Simulation Enabled High Level Emulation of a Distribution Centre". In *2012 UKSim 14th International Conference on Computer Modelling and Simulation*. March 28th-30th, Cambridge, UK.

AUTHOR BIOGRAPHIES

SUMANT JOSHI is a Senior Simulation Research Scientist with Worldwide Design and Engineering team at Amazon. He holds MS in Industrial and Operations Engineering from the University of Michigan, Ann Arbor and BE in Mechanical Engineering from University of Mumbai. He brings over 9 years of industry expertise in material handling design, new product introduction and simulation science role. His research focuses on discrete event and physics simulations for next generation fulfillment systems, robotics, emulation, AI in simulation and predictive analytics, contributing to Amazon's advanced engineering initiatives. He holds three patents reflecting his innovative contributions to the field. His email address is josumant@amazon.com.

SAURABH LULEKAR is a Senior Simulation Research Scientist with Worldwide Design and Engineering team at Amazon. He received his Master's in Industrial Engineering from Purdue University and Bachelor's in Mechanical Engineering from Indian Institute of Technology, Roorkee. He has over 8+ years of industry experience working in simulation science role at Amazon. His research interests include discrete event and physics simulations for warehouse design applications, controls emulation, robotics and operations planning. His email address is lulekars@amazon.com.

TARCIANA ALMEIDA is a Simulation Research Scientist at Amazon, specializing in process validation and optimization. She holds a degree in Electrical Engineering from Federal University of Sergipe in Brazil and a Master's in Engineering of Technology Robotics from Wayne State University. Her research interests include discrete event simulation and data analytics for logistics optimization, process improvement methodologies, and performance modeling of next-generation fulfillment systems. Her email address is tarcialm@amazon.com.

ABHINEET MITTAL is a Manager of Research Science for Core Fulfillment Simulation within Amazon's Worldwide design and engineering team. He obtained his Master's in Industrial Engineering from Arizona State University in 2014. He has over 11 years of industry experience with 2 years in Fiat Chrysler Automobiles and 9 years at Amazon Simulation Science. His research interests are simulation optimization, robotics, and machine learning. His email address is abhineem@amazon.com

GANESH NANAWARE is a Senior Manager of Research Science with Amazon's Worldwide Design Engineering. He received his Master's in Mechanical Engineering, a Master's in Business Administration, MIT Machine Learning and PMP certification. He has 20+ years of professional experience in leading the research and simulation teams in various industry sectors. His research interest includes science driven process and product optimization, AI based simulation, machine learning, distribution process simulation to drive innovation, and research-based strategy development. His email address is nanawg@amazon.com.