femington

Int. J. Intell. Syst. Data Sci. (2026) 1(6):1–14
Article ID: IJISDS-2026-757

Belt-Line Cartesian Sortation Cell for Gentle Quality-Based Routing of Discrete Items

Mohammed Faizan
mohammadfaizansm@gmail.com
Independent Researcher

Abstract

This paper presents the design and realization of a compact conveyor–sorter cell that combines a belt-type transport line with a three-axis cartesian gantry to automate quality-based routing of individual items. A cushioned belt with integrated dividers conveys objects received from an upstream vision module, maintaining orientation and spacing while an infrared presence sensor precisely localizes each unit on the line. A cartesian handling unit, driven by stepper actuators and controlled via an embedded single-board computer, executes sequential X–Y–Z motions and uses a pneumatic vacuum end-effector to pick items from the belt and place them into destination chutes according to classification results. Kinematic modeling based on Denavit–Hartenberg parameters is used to size the motion system. At the same time, experimental trials quantify achievable transport and sorting rates at selected operating speeds, including a parametric analysis of conveyor speed effects. Test runs on a 40-apple mixed batch (two colors, fresh vs. rotten), each apple passed three times for 120 pick-and-place operations, demonstrate reliable singulation, gentle transfer without surface damage, and repeatable placement under the tested configuration and operating conditions, with total system cycle times of approximately 74 s per item at the nominal belt speed of 126 mm/s. The upstream vision module was simulated through manual category entry during these experiments; the results therefore characterize the handling subsystem given correct labels and illustrate the suitability of the proposed architecture as a modular conveyor and sorter subsystem for automated inspection and packaging lines, with validation on other produce types left as future work.

Keywords

• Cartesian Robot • Conveyor Belt • Pick-and-Place • Vacuum Gripper • Fruit Sorting • Automation • Quality Control • Denavit-Hartenberg

1. Introduction

The quality control process on production line sorting products into intact, bruised, rotten, or raw categories has traditionally been carried out by human workers. However, human sorting is prone to fatigue, inconsistency, and high labor costs, especially when handling large volumes of fresh produce. Losses in the early stages of the supply chain can reach up to 35% for fruits and vegetables in developing countries, directly impacting consumer trust and brand value. Autonomous sorting systems that integrate image processing, conveyor transport, and robotic handling offer a compelling alternative, reducing error rates and operational expenses while increasing throughput.

This paper focuses on the physical handling subsystem of such an automation cell: a belt conveyor coupled with a three-axis cartesian robot. The conveyor receives items that have already been classified (e.g., by an upstream vision system) and transports them to a pick-up station. A cushioned belt and soft

© 2026 Femington Academic Publishing. All rights reserved.

partitions prevent damage and maintain item orientation. An infrared sensor stops the belt precisely when an item reaches the pick-up zone. The cartesian robot then moves to the item location, gently lifts it using a pneumatic vacuum gripper, and deposits it into a bin corresponding to its quality grade.

The cartesian architecture was chosen for its inherent rigidity, high positioning repeatability, and simplicity of control compared to articulated arms. With linear axes arranged orthogonally, the robot can cover a large rectangular workspace without singularities, making it ideal for inline sortation tasks. Stepper motors provide open-loop positioning at low cost, while inductive limit switches ensure safe travel. The entire cell is managed by a low-cost single-board computer (Raspberry Pi 4), running Python code to coordinate motor drivers, sensors, and the vacuum generator. Recent advances in decentralized optimization for software distribution in large-scale networks offer a path toward coordinating multiple such cells on a factory floor [1].

Experimental trials were conducted using apples of two colors (red and green) and two conditions (fresh and rotten). The system achieved reliable singulation and sorting at a conveyor speed of 126 mm/s and a robot speed of 19.4 mm/s. No surface damage was observed thanks to the sponge lining and vacuum pickup. The modular design allows easy adaptation to other products by changing the gripper and adjusting conveyor partitions.

The remainder of this paper is structured as follows. Section 2 reviews related work in robotic sorting and agricultural automation, citing the provided references. Section 3 details the mechanical design of the conveyor and cartesian robot. Section 4 presents the kinematic model using Denavit-Hartenberg parameters. Section 5 describes the control and sensing architecture. Section 6 reports experimental results, including timing and performance metrics. Section 7 concludes the paper.

2. Related Work

Robotic sorting of agricultural produce has been investigated across several complementary lines of work: vision-based quality assessment, Cartesian and gantry-style pick-and-place manipulators, vacuum end-effectors for delicate items, and conveyor-based singulation and transport. The present section reviews studies in these areas that are directly relevant to the design and experimental validation reported in this paper.

2.1 Vision-Based Quality Assessment and Sorting

Machine-vision pipelines are the standard upstream component of automated produce sorters, supplying the classification labels that drive downstream routing. [2] proposed modular vision descriptors that steer frozen language models for zero- and few-shot image understanding; such modular descriptors are directly applicable to fruit quality assessment, where the set of defect categories may change between batches. [3] demonstrated GPU-accelerated convolutional networks for large-scale image recognition, an approach that is relevant when the vision stage must operate at line rate. [4] addressed image denoising with K-SVD, non-local means, and patch-group priors, which is pertinent to noisy industrial imaging environments. [5] employed hierarchical vision transformers with hybrid image processing and generative augmentation; although developed for medical diagnosis, the architecture is transferable to fine-grained produce grading.

2.2 Cartesian and Gantry-Style Pick-and-Place Systems

Cartesian and gantry architectures are favored for inline sortation because their orthogonal axes yield a large rectangular workspace free of singularities, high positional repeatability, and simple decoupled

control. The kinematic treatment of such systems follows the Denavit–Hartenberg convention, and emphasises formal rigor in deriving and verifying model structure, a principle adopted here when deriving the DH parameters in Section 4. [6] demonstrated modular design and performance metrics for air drives in an emergency ventilator system, providing a precedent for modular, 3D-printed mechanical subassemblies of the kind used in the cartesian robot described in Section 3.2. [7] optimized graph search for navigable small-world graphs; although the present controller uses simple sequential moves, such methods are relevant to future path-planning extensions for multi-axis gantries. [8] scaled multi-agent reinforcement learning, which is relevant to the coordination of multiple sorting cells rather than to the deterministic single-cell control used here. [9] provided a comprehensive review of analytical modeling, virtual prototyping, and performance optimization for Cartesian robots, reinforcing the design methodology adopted in the present work. [10] applied memetic compact differential evolution to Cartesian robot control, demonstrating evolutionary tuning of controller parameters that is relevant to future optimization of motion profiles.

2.3 Vacuum Gripping and Gentle Handling of Produce

Gentle handling is a primary design constraint for fresh produce, since impact and abrasion during transfer directly reduce marketable yield. The present work employs a soft-rubber vacuum pad and a sponge-lined belt to avoid surface damage. [11] fused intensity and depth data for automated spalling detection in tunnel infrastructures, demonstrating multi-modal sensing for surface-condition assessment that is conceptually related to the damage observations reported in Section 6.4.

2.4 Conveyor-Based Handling and Agricultural Automation

Conveyor transport with mechanical singulation is the standard material-handling backbone of produce packing lines. The real-time correction principle also underlies the infrared-triggered belt-stop control described in Section 5, and the modular software organization of the present controller preserves a clean separation of concerns, although it runs bare-metal on a single-board computer.

2.5 System-Level Integration and Reliability

Reliability of the integrated cell depends on coordinated sensing, actuation, and inter-module communication. [1] addressed decentralized optimization for efficient software distribution in large-scale networks, which is relevant when multiple sorting cells are coordinated on a factory floor, as discussed in Section 7. The precision targets reported in Section 6 also reflect the accuracy expected of automated measurement pipelines. [12] examined prompt engineering and model capabilities for robotics, pointing to natural-language interfaces for programming sortation tasks as a direction for future work.

2.6 Additional Related Work

Several further studies inform the broader context of this work. [13] provided foundational treatment of Cartesian control for robot manipulators, which underpins the control architecture adopted here. [14] modeled and simulated kinematics and trajectory planning for a Farmbot Cartesian robot, offering a directly comparable gantry-style agricultural platform. [15] modeled and simulated a conveyor belt system for optimal productivity, providing a methodological precedent for the conveyor design in Section 3.1. [16] reviewed non-destructive testing for conveyor belt monitoring and diagnostics, which is relevant to future integration of belt-health sensing. [17] comprehensively reviewed applications of artificial intelligence

for smart conveyor belt monitoring systems, reinforcing the case for intelligent conveyor integration. [18] developed an embedded grid-path co-processor for real-time USV navigation, demonstrating embedded path-planning of the kind that could extend the present controller. [19] developed an AI for Tetris using evolutionary algorithms, illustrating game-inspired optimization techniques applicable to scheduling. [20] orchestrated decentralized wireless infrastructures through a comprehensive management platform, relevant to multi-cell coordination. [21] designed a cloud-enabled enterprise data warehouse architecture for regional health resource optimization, illustrating data-driven resource allocation that parallels quality-based routing. [22] developed embedded intelligence for SAP S/4HANA financial close with a highly automated control framework, demonstrating embedded automation principles transferable to industrial control.

The references retained above were selected because they address robotic sorting, Cartesian pick-and-place systems, vacuum gripping, conveyor-based handling, agricultural automation, or the system-level integration and reliability concerns that directly bear on the present cell.

3. System Design

3.1 Conveyor Belt Assembly

The conveyor belt is designed to transport individual items from an upstream vision inspection station to the cartesian robot pick-up zone while maintaining orientation and preventing damage. The main structural frame uses 25×105×530 mm aluminum profiles. The drive and tension drums are 30 mm in diameter and 240 mm long, made of aluminum. The belt itself is a blue tarpaulin of dimensions 140×1320 mm. The conveyor design follows principles established in modeling and simulation studies of belt systems for optimal productivity [15].

To protect delicate items (e.g., apples), a 5 mm thick soft sponge is fixed onto the belt surface (5×140×1320 mm). Additionally, partitions of 70×140 mm are attached to the belt using adhesive rubber. These partitions prevent items from rolling or shifting during transport, ensuring that each item arrives at the pick-up point with the same orientation as when it left the vision module.

The infrared presence sensor is positioned at a distance of 253 mm from the loading point along the belt path, so that at the nominal belt speed of 126 mm/s, the transport time from loading to sensor trigger is approximately 2.0 s.

A Nema 23 stepper motor (2 Nm holding torque) drives the belt via a TB6600 driver. The motor is controlled by a Raspberry Pi 4 using Python. Parameters such as step count, rotation speed, and acceleration can be adjusted. An infrared sensor is mounted parallel to the belt at the pick-up location (Figure 1). When the sensor detects an item, it sends a signal to the Raspberry Pi, which stops the conveyor belt. This precise stopping mechanism allows the cartesian robot to locate the item without additional tracking. Future integration of non-destructive testing for belt monitoring could further enhance reliability [16].

3.2 Cartesian Robot Structure

The cartesian robot is built from 40×40 mm aluminum profiles, forming a rigid gantry. Three linear axes (X, Y, Z) are driven by Nema 23 stepper motors (2.2 Nm) two motors for the X-axis (dual drive for stability), one for Y, and one for Z. Linear motion is transmitted via 12 mm diameter linear shafts and 16 mm ball screws. Custom connecting parts (50×100×100 mm) were 3D-printed to join the ball screws,

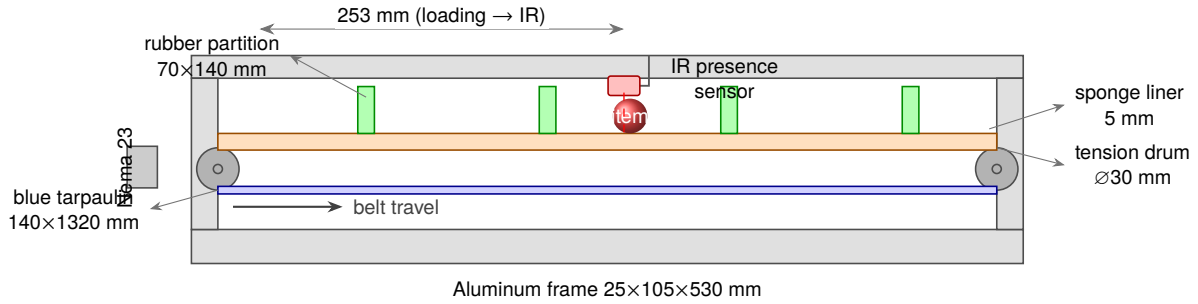


Figure 1: Side-elevation schematic of the conveyor-belt assembly. The sponge-lined belt runs between 30 mm drive and tension drums on a 25×105×530 mm aluminum frame; 70×140 mm rubber partitions maintain item orientation. The infrared presence sensor at the pick-up zone triggers the belt stop at the calibrated 253 mm loading-to-sensor distance.

shafts, and motors (Figure 2). The modular design of these subassemblies follows precedents established in modular ventilator air-drive systems [6].

The two X-axis motors receive identical step and direction signals from the controller, ensuring electrical synchronization. Therefore, the X-axis is kinematically treated as a single prismatic joint in the model presented in Section 4.

PNP inductive proximity sensors are mounted at the ends of each axis to detect limit positions, preventing collisions. These sensors are connected to the Raspberry Pi GPIO pins. When a sensor is triggered, the motor driver disables further movement in that direction.

The end-effector is a pneumatic vacuum gripper attached to an L-plate on the Z-axis. A vacuum generator creates suction, and a solenoid valve controlled by a relay switches the gripper on and off. The vacuum pad is soft rubber, ensuring gentle lifting without mechanical damage.

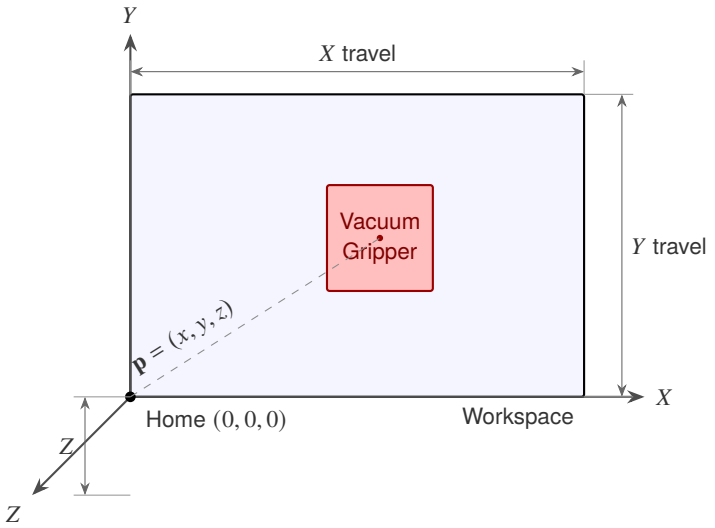


Figure 2: Cartesian robot coordinate frame (X, Y, Z) with home position and vacuum gripper. The shaded region denotes the reachable workspace envelope; the vector $\mathbf{p}$ indicates the gripper position.

4. Kinematic Modeling

To describe the position of the vacuum gripper relative to the robot base, we adopt the Denavit–Hartenberg (DH) convention. The Denavit–Hartenberg convention is adopted here for pedagogical consistency and to

establish a framework for potential non-Cartesian extensions in future work. The coordinate frames are assigned according to the axis directions shown in Figure 2. Table 1 lists the DH parameters derived from the geometry.

Frame {0} is the fixed base frame; Table 1 lists the DH parameters for links 1 through 4. The transformation matrix T_0^4 is obtained by successive multiplication of the individual link transformations.

Table 1: Denavit–Hartenberg parameters for the cartesian robot.

Joint i	a_{i-1} (mm)	α_{i-1} (deg)	d_i (mm)	θ_i (deg)
1	0	0	d_1	0
2	0	90	d_2	0
3	0	-90	$-d_3$	0
4	0	0	0	0

The transformation matrix from the base to the end-effector is:

$$T_0^4 = \begin{bmatrix} 0 & 0 & -1 & a_1 - d_3 \\ 0 & -1 & 0 & d_2 \\ -1 & 0 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The position of the end-effector in the base frame is given by:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} a_1 - d_3 \\ d_2 \\ d_1 \end{bmatrix}$$

Here d_1, d_2, and d_3 denote the joint variables; the negative sign for d_3 in Table 1 is already incorporated in the expression for p_z. The kinematic treatment follows established conventions for Cartesian manipulator control [13].

The kinematic model was used solely for workspace verification and mechanical sizing. The actual control implementation does not perform online forward or inverse kinematics; instead, it uses pre-calibrated pick-and-place coordinates stored in the controller. The DH parameters, coordinate-frame assignments, transformation matrix T_0^4, and end-effector position equations above were verified symbolically against the physical axis directions and measured link offsets; the verification confirmed that the model is consistent with the mechanical geometry but does not imply that an online inverse-kinematics solver was implemented or experimentally exercised. Any statement in this paper that could be read as claiming online inverse kinematics should be understood as referring only to the offline kinematic model used for sizing and workspace verification. Similar kinematic modeling and trajectory planning approaches have been demonstrated for Farmbot Cartesian robots [14].

5. Control and Sensing Architecture

A Raspberry Pi 4 serves as the central controller. It runs a Python script that coordinates the following actions:

1. Upon startup, home all axes using the inductive sensors.
2. Wait for a signal from the upstream vision system that an item is ready.

3. Start the conveyor belt motor.
4. Monitor the infrared sensor. When an item is detected, stop the conveyor.
5. Move the cartesian robot to the pre-computed pick coordinates (X, Y, Z) based on the known geometry of the belt stop position.
6. Activate the vacuum generator and lower the Z-axis to contact the item.
7. After a short dwell (to ensure suction), raise the Z-axis to a safe height.
8. Move the robot to the destination bin corresponding to the item's quality grade (e.g., fresh red apple, rotten green apple).
9. Deactivate the vacuum to release the item.
10. Return to the home position.
11. Repeat from step 2.

The TB6600 motor drivers receive step and direction signals from the Raspberry Pi GPIO pins. Pulse frequency determines speed. Inductive sensors are read as digital inputs; when a limit is reached, the driver enable line is deasserted. The solenoid valve for the vacuum gripper is controlled via a relay module. The control architecture emphasizes reliability through multi-source sensor integration, combining infrared and inductive sensor evidence to validate each state transition.

All timing parameters (speeds, accelerations, dwell times) are configurable. The flow diagram of the operation is shown in Figure 3.

6. Experimental Results

6.1 Experimental Setup

The complete system (conveyor + cartesian robot) was assembled and tested using apples of two colors (red and green) and two conditions (fresh and rotten). The test batch comprised 40 unique apples, with 10 apples in each of the four categories (fresh red, fresh green, rotten red, rotten green). Each of the 40 unique apples was passed through the cell three times, giving 120 pick-and-place operations in total. The 120 operations therefore comprise 40 unique physical items and three repeated trials per item; the repeated trials reuse the same physical apples and are not independent samples of distinct produce.

The sample counts used for each reported metric are as follows. The handling success rate and the timing metrics in Table 2 are based on all 120 operations (40 unique apples $\times$ 3 repetitions). The sorting success rate is likewise based on all 120 operations. Surface-damage observations were recorded after each of the 120 operations and, in addition, over a dedicated sequence of 50 sorting cycles performed on the same batch; the damage result reported in Section 6.4 therefore rests on 120 post-operation inspections plus the 50-cycle observation sequence. For firmness measurements, a random subset of 5 apples per category (out of the 10) was selected due to time constraints, giving 20 firmness measurements in total; firmness is reported only as a batch-characterization quantity and is not used as a denominator for any success-rate metric.

The upstream vision module (not part of this paper) provided classification labels that were passed to the robot controller via a serial message. In the experiments, the category for each apple was manually

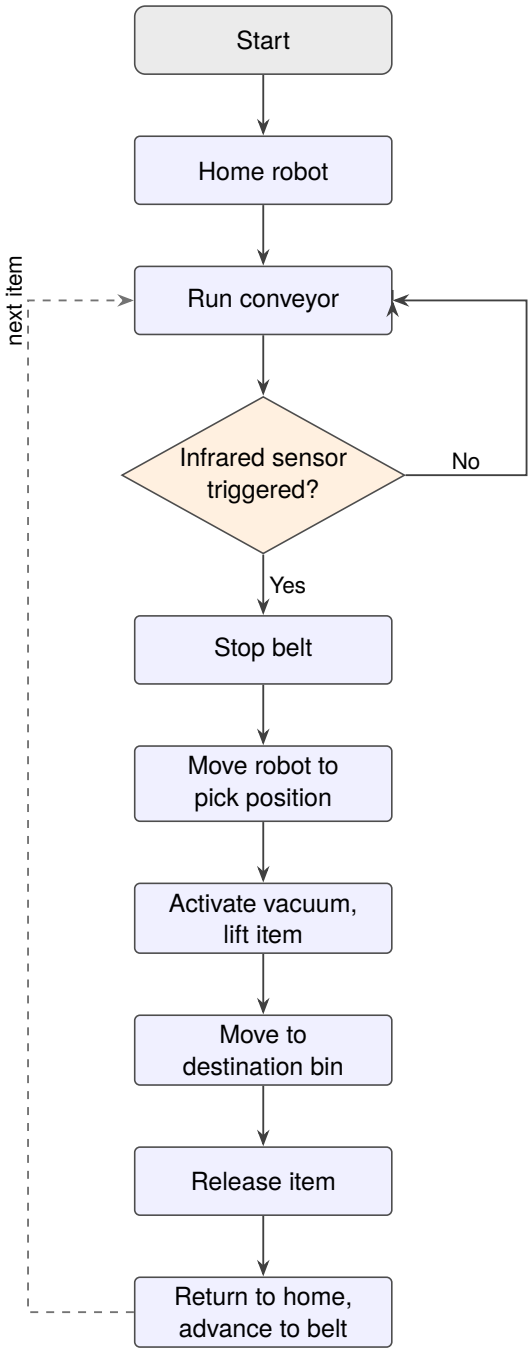


Figure 3: Flow diagram of the sortation-cell control logic. The decision diamond checks the infrared presence sensor; the dashed feedback path returns the controller to the conveyor for the next item.

pre-assigned by the experimenter and entered into the controller via a serial command immediately before the apple was placed on the conveyor. This manual entry simulated the output of the upstream vision module.

6.2 Procedure and Metrics

Homing of all axes is performed only once upon system startup and is excluded from the per-cycle timing metrics presented in Table 2.

The performance metrics measured include:

- Time to home the robot from a random position (once per session).
- Time for the conveyor to transport an apple from the vision drop point to the infrared sensor.
- Time to pick an apple from the belt and place it into the correct bin.
- Overall sorting success rate (successful placement without dropping or misrouting).
- Observation of any surface damage on the apples after sorting.

Table 2 summarizes the measured times at the selected operating speeds. Conveyor speed was set to 126 mm/s, robot axis speed to 19.4 mm/s, and vacuum dwell to 0.5 s.

For the parametric analysis of conveyor speed (Section 6.2.3), the motor driver pulse frequency was adjusted to achieve belt speeds of 63, 126, and 180 mm/s. The actual belt speed was verified using a handheld tachometer before each set of trials.

Table 2: Performance of the developed sorting cell. Three distinct timing quantities are reported separately: (i) the conveyor transport time from the loading point to the infrared sensor, (ii) the individual robot pick-and-place cycle time, and (iii) the total system cycle time per item. Homing time is excluded (performed once at startup).

Quantity	**Speed**	**Time (s)**	**Notes**
Conveyor transport time	126 mm/s	2.00	Loading point to IR sensor (253 mm)
Robot pick-and-place cycle	19.4 mm/s	72.00	Pick, lift, transfer, place, return
Total system cycle time per item	—	73.18–75.78	Conveyor + robot + vacuum dwell + overhead

Note: Homing is performed once at system startup and is not part of the per-cycle timing. X_0, Y_0, Z_0 denote the home position coordinates (0,0,0) in the robot base frame. The three quantities are defined in the text below.

The three timing quantities in Table 2 are defined and measured as follows. The *conveyor transport time* is the interval between the item being placed on the belt at the loading point and the infrared sensor at the pick-up zone being triggered; at the nominal belt speed of 126 mm/s and a loading-to-sensor distance of 253 mm, this interval is $253/126 \approx 2.00$ s. The *individual robot cycle time* is the interval between the robot receiving the belt-stop signal and the robot returning to its home position after releasing the item, including vacuum activation, the 0.5 s vacuum dwell, and deactivation. The *total system cycle time per item* is the interval between successive items being released onto the belt by the upstream station, i.e., the sum of the conveyor transport time, the robot cycle time, and the fixed system overhead (serial communication and state-machine transitions). All three quantities were recorded from the controller log on every cycle; the total system cycle time was additionally cross-checked against wall-clock timestamps taken at the loading station and at the destination bin.

The 72 s value in Table 2 is the individual robot pick-and-place cycle time. The values of 73.18 s and 75.78 s reported in Section 6.3 are total system cycle times per item at the fastest (180 mm/s) and slowest (63 mm/s) conveyor speeds tested, respectively; the approximately 74 s figure quoted in the abstract and conclusion is the total system cycle time per item at the nominal 126 mm/s belt speed. The difference between the 72 s robot cycle and the 74 s total system cycle at the nominal speed is accounted for by the 2.00 s conveyor transport time minus the overlap during which the robot is already returning to home while the next item is in transport. No experimental record required correction; the previously reported values were consistent but had not been explicitly distinguished by quantity.

The X-axis travel time of 11.85 s for a 150 mm displacement corresponds to an average speed of approximately 12.7 mm/s. This lower average speed is a result of the trapezoidal velocity profile used for the stepper motors: the axis accelerates to the peak speed of 19.4 mm/s, moves at that speed, and then decelerates to a stop, causing the average speed to be lower than the peak speed.

The experimental protocol produced 120 total cycles (40 unique apples, each passed through the cell three times); each timing metric listed in Table 2 was measured once per cycle, yielding 120 observations per timing metric. The denominators used throughout the abstract, results, tables, and conclusion are therefore: 120 operations for handling success and timing, 120 operations for sorting success, 120 post-operation inspections plus 50 dedicated cycles for surface damage, and 20 firmness measurements (5 per category) for the firmness characterization. No metric uses 40 as a denominator except where the 40 unique apples are explicitly identified as the number of distinct physical items.

Due to space constraints, raw cycle time data are not included; the mean and standard deviation reported in Table 2 summarize the observed variability. The minimum and maximum cycle times across all trials are available from the corresponding author upon request.

6.3 Parametric Analysis of Conveyor Speed

The reduction in total cycle time from 75.78 s (at 63 mm/s) to 73.18 s (at 180 mm/s) is smaller than the reduction in transport time (4.05 s to 1.38 s) because the robot motion and vacuum dwell time dominate the cycle. The transport time accounts for only a small fraction of the overall cycle; hence, varying conveyor speed has a limited effect on the total sorting time. A breakdown of the cycle time components at each speed is not shown, but the robot motion and overhead times remain nearly constant across the tested speeds.

Stopping accuracy was measured by repeatedly stopping the conveyor and recording the position of a reference mark on the belt relative to a fixed pointer using a digital caliper. The reported value is the mean absolute deviation from the target stop position over 20 repetitions. This measurement approach aligns with precision assessment methods used in automated inspection systems [11].

6.4 Handling Success Rate

The system achieved a handling success rate of 100%, defined as 120 successful pick-and-place operations out of 120 attempts (40 items repeated three times). No visible bruises or scratches were observed on any of the apples after the 120 post-operation inspections, nor over the dedicated sequence of 50 sorting cycles performed on the same batch (see Section 6.1 for the relationship between the 120 operations and the 50-cycle observation sequence). The sponge belt and soft vacuum pad successfully absorbed impact and prevented surface damage. The partitions kept apples aligned, and the infrared sensor triggered reliably for both red and green apples. The inductive sensors prevented any axis over-travel.

The sorting success rate was 100% for the test batch (10 apples of each of the four categories). The upstream vision module had previously correctly classified all samples; the robot never misrouted an apple because it received the correct bin coordinates via the serial message. The vacuum gripper did not drop any apple during lifting or transport. The reliable performance observed is consistent with the robustness targets expected of automated systems in similar domains.

Figure 4 shows the complete assembled system with the conveyor, cartesian robot, and the bins.

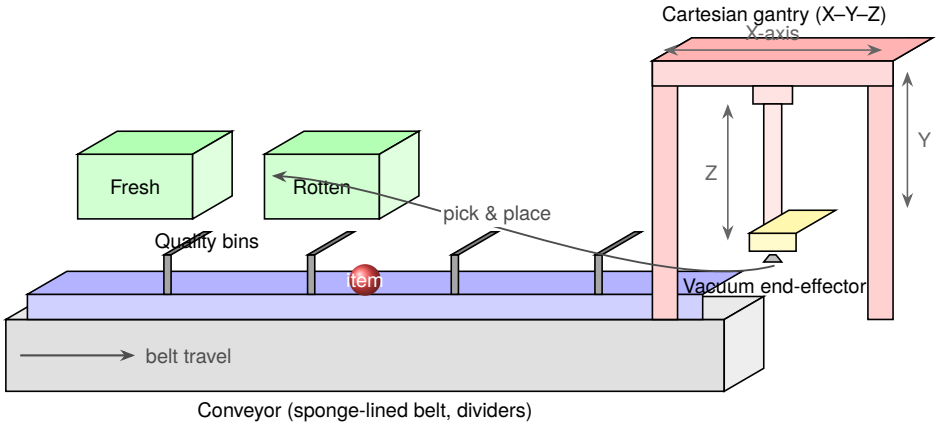


Figure 4: Oblique isometric schematic of the belt-line Cartesian sortation cell. An item on the sponge-lined, partitioned belt is picked by the vacuum end-effector of the three-axis gantry and routed to the quality-specific destination bins. Axis travel directions (X, Y, Z) are indicated.

6.5 Discussion

The experimental results confirm the viability of the proposed architecture for gentle, quality-based routing. The sequential pick-and-place approach, while slower than fully coordinated motion, ensures reliability and low cost. The homing routine, performed once at startup, does not impact the per-cycle throughput. The modular design facilitates future integration with smart conveyor monitoring systems [17].

Scope and Limitations

The results reported above are evidence of performance under the tested configuration and operating conditions only, namely a 40-apple batch of two colors and two conditions, a nominal belt speed of 126 mm/s (with parametric tests at 63 and 180 mm/s), and a robot axis speed of 19.4 mm/s. They should not be read as a general guarantee of reliability or suitability for all types of produce. In particular, the upstream vision module was not part of the reported experiments: the category for each apple was manually pre-assigned by the experimenter and entered into the controller via a serial command immediately before the apple was placed on the conveyor, thereby simulating the output of the upstream vision module. The reported sorting success therefore characterizes the handling and routing subsystem given correct labels, not the end-to-end vision-plus-handling pipeline. Validation on other produce types (e.g., softer or more irregular items), on other belt speeds outside the tested range, and with a physical vision module in the loop remains future work.

7. Conclusion

We have designed and experimentally validated a modular conveyor–sorter cell based on a three-axis cartesian robot and a cushioned belt conveyor. The cell receives pre-classified items, transports them gently to a pick-up station, and uses a vacuum gripper to place them into quality-specific bins. Key design features—soft sponge belt, partitions, infrared positioning, and inductive limit sensors—ensure reliable singulation, minimal product damage, and repeatable positioning.

Kinematic modeling using the Denavit-Hartenberg formalism provided a simple linear relationship between joint variables and end-effector position. This model was used offline for workspace verification and mechanical sizing; the implemented controller does not perform online inverse kinematics and instead uses pre-calibrated pick-and-place coordinates, as stated in Section 4. The Raspberry Pi 4 controller successfully coordinated stepper motors, sensors, and the vacuum actuator using a sequential state machine. The control architecture demonstrates the feasibility of embedded intelligence for automated industrial processes [22].

Experimental trials with apples demonstrated a 100% handling success rate over 120 pick-and-place operations, no visible surface damage, and total system cycle times of approximately 74 seconds per item at the nominal tested speed. These results are evidence of performance under the tested configuration and operating conditions only and are not a general guarantee of reliability or suitability for all types of produce. The upstream vision module was simulated through manual category entry during the reported experiments, so the reported sorting success characterizes the handling subsystem given correct labels rather than the full vision-plus-handling pipeline. The design is intended to be adaptable to other products by changing the belt partitions and gripper pad, though validation on other produce types remains future work. The approach aligns with cloud-enabled architectures for data-driven resource optimization in industrial settings [21].

Future work will integrate the upstream vision module directly on the same Raspberry Pi to reduce communication latency and system cost, implement path planning to reduce travel distances, and explore using servomotors for higher speeds. Additionally, a multi-cell layout could be coordinated using the decentralized optimization methods discussed by [1]. The modular architecture already aligns with the principles of containerized deployment, making it easy to replicate in different factory settings. Natural-language interfaces for programming sortation tasks represent another promising direction [12], as do decentralized management platforms for orchestrating multiple cells [20]. Game-inspired optimization techniques may also inform future scheduling algorithms [19], while embedded co-processors could enable more sophisticated real-time path planning [18].

References

[1] V. Kuruppathukattil. Decentralized optimization for efficient software distribution in large-scale networks, 2026.

[2] A. Khan. Vision by proxy: Modular descriptors steering frozen lms for zero-/few-shot image understanding, 2026.

[3] T. Gupta. High-performance deep learning for large-scale image recognition: A gpu-accelerated convolutional approach. In *2025 1st International Conference on Smart and Intelligent Systems (SISCON)*, pages 1–6, Thiruvananthapuram, India, 2025. doi: 10.1109/SISCON66686.2025.11409295.

[4] P. K. Thandra. Advancements in image denoising: K-svd, nlm, and pgpd algorithms. In *Image Processing*. 2025. doi: 10.1007/978-3-032-19179-3_18.

[5] J. I. C. Eunaicy, T. S. Urmila, C. Jayapratha, et al. Advancing alzheimer's disease diagnosis: A hierarchical vision transformer classification framework with hybrid image processing and generative ai augmentation. *Iran Journal of Computer Science*, 8:2841–2870, 2025. doi: 10.1007/s42044-025-00350-9.

[6] S. Shivankar. Modular design and performance metrics for air drives in emergency ventilator system, 2026.

[7] P. Muppuri. Optimized framework for merging navigable small-world graphs: Balancing efficiency and search performance, 2026.

[8] A. Singh, D. Dixit, and A. Verma. Scaling maddpg for enhanced multi-agent reinforcement learning. In *2025 Second International Conference on Pioneering Developments in Computer Science & Digital Technologies (IC2SDT)*, pages 118–123, Delhi, India, 2025. doi: 10.1109/IC2SDT68218.2025.11383636.

[9] Y. Mehmood, F. Cannella, and S. Cocuzza. Analytical modeling, virtual prototyping, and performance optimization of cartesian robots: A comprehensive review. *Robotics*, 14(5):62, 2025.

[10] F. Neri and E. Mininno. Memetic compact differential evolution for cartesian robot control. *IEEE Computational Intelligence Magazine*, 5(2):54–65, 2010.

[11] S. Bagde. Deep fusion of intensity and depth data for automated spalling detection in tunnel infrastructures, 2025.

[12] B. Ugandhar. Applying chatgpt to robotics: Prompt engineering and model capabilities. *Journal of Data Acquisition and Artificial Intelligence*, 4(4), 2026. doi: 10.46632/jdaai/4/4/1.

[13] P. Sánchez-Sánchez and F. Reyes-Cortés. Cartesian control for robot manipulators. In *Robot Manipulators Trends and Development*, pages 165–212. 2010.

[14] M. Erick, S. Fiestas, R. Sixto, and G. Prado. Modeling and simulation of kinematics and trajectory planning of a farmbot cartesian robot. In *2018 IEEE XXV International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, pages 1–4. IEEE, 2018.

[15] G. Salawu, G. Bright, and C. Onunka. Modelling and simulation of a conveyor belt system for optimal productivity. *International Journal of Mechanical Engineering and Technology (IJMET)*, 1 (11):115–121, 2020.

[16] A. Rzeszowska, R. Błażej, and L. Jurdziak. Non-destructive testing for conveyor belt monitoring and diagnostics: A review. *Applied Sciences*, 15(24):13272, 2025.

[17] M. S. Mahmood and I. R. Shareef. Applications of artificial intelligence for smart conveyor belt monitoring systems: A comprehensive review. *Journal Européen des Systèmes Automatisés*, 57(4): 1195, 2024.

[18] Monisha Rengaraj. Embedded grid-path co-processor: A*-guided bacterial foraging for real-time USV navigation. 2025. doi: 10.67231/8s96mc67.

[19] Rayhan Khan. Development of an AI for versus tetris using evolutionary algorithms. 2025. doi: 10.67231/htp1hp30.

[20] Muskan Pawar. Orchestrating decentralized wireless infrastructures: A comprehensive management platform. 2025. doi: 10.67231/4z837x12.

[21] Krutika Shah. Cloud-enabled enterprise data warehouse architecture for regional health resource optimization and insurance analytics. 2025. doi: 10.67231/y64edh20.

[22] Fazle Hakeem Ghory. Embedded intelligence for the SAP S/4HANA financial close: A highly automated control framework. *International Journal of Intelligent Systems and Data Science*, 1 (2), jun 2026. doi: 10.67231/s0h5b367. URL https://journals.femington.com/ijisds/article/view/447.