

# We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,700

Open access books available

180,000

International authors and editors

195M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index  
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?  
Contact [book.department@intechopen.com](mailto:book.department@intechopen.com)

Numbers displayed above are based on latest data collected.  
For more information visit [www.intechopen.com](http://www.intechopen.com)



Chapter

# Real-Time Inventory Analysis Using Jetson Nano with Object Detection and Analysis

*Vinayak Bharadi, Suha Mukadam, Rahul Prasad,  
Kavyashree Upparakakula and Janhavi Jaygade*

## Abstract

Retail product recognition on grocery shelf photographs is a challenging task due to the variety of products and lighting conditions. This research proposes a novel method for completing this task efficiently and in a short amount of time. The object detection and recognition module is deployed on a jetson Nano board, which is trained to trace a path for real time object image capture. The proposed method consists of two stages: detection and recognition. The detection stage is performed by a generic product detection module that has been trained on a particular class of products (e.g., tobacco packages). This is accomplished using the Viola and Jones Cascade Object Detection Framework. The recognition stage is performed using two FCN designs: U-Net and Fully Convolutional Regression Network (FCRN). We extract both shape and color information by applying feature-level fusion to two distinct descriptors computed using the bag of words method. We evaluated our method on a dataset of grocery shelf photographs. We achieved a detection accuracy of 95% and a recognition accuracy of 90%. The proposed method is efficient and accurate, and it can be used for a variety of applications, such as inventory management, price tracking, and fraud detection.

**Keywords:** object recognition, object detection, retail product, grocery image, bag of words

## 1. Introduction

Computer vision is being used to analyze grocery store photographs for useful data. This can be used to ensure that products are displayed correctly, to help visually impaired shoppers navigate stores, and to automate inventory management. Object detection and recognition are two challenges that must be overcome in order to use computer vision for these tasks. Object detection is the problem of identifying objects in an image, while object recognition is the problem of identifying the specific type of object that has been detected [1]. These challenges can be made more difficult by factors such as low-quality images, objects that are partially obscured, and the large number of different types of objects that can be found in a grocery store.

Real-time inventory analysis is a critical task for businesses of all sizes. It allows businesses to track their inventory levels in real time, identify potential stockouts, and make informed decisions about ordering and restocking. Traditional methods of inventory analysis, such as manual counting and pen-and-paper records, are time-consuming and error-prone [2].

In recent years, there has been a growing interest in using computer vision and machine learning to automate inventory analysis. This approach has the potential to significantly improve the efficiency and accuracy of inventory management. One promising platform for real-time inventory analysis is the Jetson Nano. The Jetson Nano is a small, affordable, and energy-efficient computer that is designed for machine learning applications. It is well-suited for real-time inventory analysis because it can quickly and accurately detect and identify objects in images.

In this chapter, we present a system for real-time inventory analysis using the Jetson Nano. Our system uses object detection and analysis to identify products on shelves and track their inventory levels in real time. We evaluate our system on a dataset of images of grocery store shelves and show that it can achieve high accuracy in detecting and identifying products. We also show that our system can be used to track inventory levels in real time and identify potential stockouts. We believe that our system has the potential to significantly improve the efficiency and accuracy of inventory management. We believe that our system will be a valuable tool for businesses that are looking to improve their inventory management [3].

## **2. Problem statement**

In an effort to improve operational efficiency in its large-format stores, the company is deploying aisle-scanning robots. These self-propelled robots will track product availability on the shelves, verify prices, and provide precise location data for over 100,000 items at each location. The robots will also collect data on purchasing trends, which will help the company to better forecast and manage inventory. The robots use a computer vision-based system to identify objects placed beneath a webcam. The system is powered by a convolutional neural network, which is a type of artificial intelligence that is trained to recognize patterns in images. The system is able to identify a wide variety of objects, including products, prices, and shelf labels. The use of aisle-scanning robots is expected to save the company time and money. The robots will automate tasks that are currently performed manually, such as checking inventory levels and verifying prices. This will free up employees to focus on other tasks, such as customer service and merchandising. The robots will also collect data that can be used to improve the company's operations.

The use of aisle-scanning robots is just one example of how the company is using technology to improve its operations. The company is also investing in other technologies, such as artificial intelligence and machine learning. These technologies are helping the company to automate tasks, improve efficiency, and better serve its customers.

## **3. Related work/literature survey**

Zhu et al. [4] proposed a weakly supervised framework for grocery product recognition from shelf images. The framework is based on a convolutional neural network (CNN) that is trained with a small number of labeled images and a large

number of unlabeled images. The CNN is able to learn to recognize products from the unlabeled images by using the labeled images as a guide. The results of the paper show that the proposed framework is able to achieve high accuracy with a small number of labeled images.

In [5] Li et al. proposed a deep learning-based method for grocery product recognition in shelf images. The method uses a two-stage approach. In the first stage, a region proposal network (RPN) is used to generate a set of candidate regions. In the second stage, a classifier is used to classify each candidate region as a product or background. The results of the paper show that the proposed method is able to achieve high accuracy on a challenging dataset of grocery shelf images.

Chen et al. [6] discussed a method for grocery product recognition that fuses data from multiple modalities, including images, text, and barcodes. The method uses a CNN to extract features from the images, a RNN to extract features from the text, and a SVM to classify the products. The results of the paper show that the proposed method is able to achieve higher accuracy than methods that use only one modality.

Wang et al. [7] have discussed a method for grocery product recognition on shelf images that uses a hybrid attention mechanism. The attention mechanism is used to focus the attention of the model on the most important parts of the image. The results of the paper show that the proposed method is able to achieve higher accuracy than methods that do not use an attention mechanism.

Zhang et al. [8] proposed a method for grocery product recognition on shelf images that uses a generative adversarial network (GAN). The GAN is used to generate synthetic images of products that are similar to the real images. The real and synthetic images are then used to train a classifier. The results of the paper show that the proposed method is able to achieve higher accuracy than methods that do not use a GAN.

Carmo et al. [9] presented a vision-based line following system for mobile robots in complex environments. The system uses a camera to detect the line, and a controller to steer the robot in the direction of the line. The system is robust to noise and other disturbances, and it can be used in a variety of environments, including indoor and outdoor environments.

Al-Aziz et al. [10] discussed a vision-based line following system for mobile robots using deep learning. The system uses a camera to detect the line, and a deep learning model to classify the image as a line or not a line. The system is robust to noise and other disturbances, and it can be used in a variety of environments, including indoor and outdoor environments.

Sahoo et al. [11] presented a vision-based line following system for mobile robots using reinforcement learning. The system uses a camera to detect the line, and a reinforcement learning agent to learn how to follow the line. The system is robust to noise and other disturbances, and it can be used in a variety of environments, including indoor and outdoor environments.

#### **4. Proposed solution**

Mobile robot navigation systems based on lines, structures, and signs have been widely used around the world. The main objective of these works is to create a mobile robot that can use a line, landmark, or sign as a point of reference to navigate along a predefined path or in the direction of a predetermined destination. The main task of

this work is line following, which is most frequently carried out using a vision system. The literature refers to the camera-mounted mobile robot as an AGV (Autonomous Guided Vehicle), and it is designed to be used for navigation within an industrial or factory setting.

The AGV is trained to recognize the various line shapes that could be captured by the camera and react appropriately. In fact, an obstacle avoidance algorithm has been added because accidents of this nature are very common in a factory setting. The processing time needed for this work, while extremely advanced for its time, is too long for use in many real-time applications since it can take up to 300 ms to process a single image. This is primarily because of the incredibly complicated technique used and the era's hardware restrictions. We are developing a model that can detect the product and provide us with a rectangular bounding box around the image in the form of 2-dimensional (x, y) coordinates. The problem is the detection of products that are placed in grocery stores. Therefore, after deconstructing the issue, we can see that we can apply the Computer Vision object detection technique to resolve this issue, allowing us to obtain the coordinates for the products that are visible in the images. Accordingly, we'll be utilizing the most up-to-date object detection techniques, which are built on top of convolutional neural nets and deep neural net layers to create a reliable and efficient model.

## **4.1 Hardware and software**

### *4.1.1 Hardware requirements*

1. Jetson Nano Developer Kit (4 GB).

The Jetson Nano Developer Kit is a small, powerful computer that can run multiple neural networks at the same time. It has 4 USB ports, 2 CSI ports, 64-bit processors, and can be connected to the internet via LAN. It can also be powered by battery or DC power supply (if powered by DC supply, a jumper must be connected). The Jetson Nano has a GPU (Graphics processing unit) that helps it process images quickly. It is used for applications such as image classification, object identification, and segmentation. It requires 5 watts of power to start.

2. Computer screen.

A visual display, some circuitry, a casing, and a power supply typically make up a monitor.

3. 64GB SD card.

Data is stored on the Secure Digital (SD) memory card. The 90 MB per sec SD card is being used. The Jetson nano's OS is stored on an SD card. Memory is kept in the SD card.

4. Power supply 5 V DC.

To start the Jetson nano, we need to give 5-volt DC supply. The power supply can be either from the battery or from the power supply. The power supply needs to be converted from AC to DC 5 V.

#### 5. Power supply 12 V for motors.

To move the robot, we need to supply the power to motors. The power supply of 12 V DC is required. To power jetson and other components we will require the power supply.

#### 6. Motor.

An electric motor is a machine that converts electrical energy into mechanical energy. When a current is applied to the motor, it creates a magnetic field that causes the motor to rotate. The speed of the motor depends on the amount of current applied. In a robot, electric motors are used to move the robot's joints and limbs. The motors are controlled by the robot's computer, which sends signals to the motors to tell them how fast and in what direction to rotate.

The type of electric motor used in a robot depends on the robot's application. For example, a robot that needs to move quickly will use a different type of motor than a robot that needs to move with precision.

#### 7. Motor drivers (4 channel).

As per the output given to the drivers the motors will move. It will convert the low voltage to high voltage which will move the robot more efficiently. The 4 channel is used to parallelly use 4 motors at a time. This will help to move its side wheels, forward and backwards.

#### 8. Arduino Uno R3.

The Arduino Uno board has a 16 MHz ceramic resonator, 14 digital input/output pins (6 of which can be used for Pulse Width Modulation (PWM)), 6 analog inputs, a USB connection, a power jack, an ICSP header, and a reset button. It has everything needed to support the microcontroller. You can connect it to a computer using a USB cable or power it with an AC-to-DC adapter or battery. You can experiment with your Uno without worrying too much about making a mistake, as you can replace the chip for a few dollars if something goes wrong.

#### 9. Wi-Fi Module.

It is used to give a network to the Jetson nano. Which will help to connect to the local system.

#### 10. Raspberry PI 5MP Camera and Logitech 720p USB Camera.

This will help to take images of the surroundings and help robots to move. The robot will have two cameras, one CSI and another USB.

#### 4.1.2 Software requirements

In order to use the Jetson Nano Arduino boards, you will need to install the following software:

#### 4.1.2.1 Ubuntu OS–Version 18.7 for Jetson Nano

Ubuntu is a free and open-source Linux distribution that is based on Debian. It is a popular choice for both personal and professional use, and it is known for its stability, security, and ease of use.

To install Ubuntu, you can download the ISO file from the Ubuntu website and create a bootable USB drive or DVD.

Once you have created a bootable media, you can boot your computer from it and follow the on-screen instructions to install Ubuntu.

This is installed on Jetson Nano as a Jetpack. Jetpack includes a variety of features that make it a powerful development platform for the Jetson Nano.

- CUDA and TensorRT are included for accelerated computing.
- OpenCV and GStreamer are included for image processing and video streaming.
- Qt and Python are included for application development.
- A wide range of pre-built applications are included for a variety of purposes, such as AI, robotics, and machine learning.

#### 4.1.2.2 Arduino IDE 2.0

The Arduino IDE is a free and open-source integrated development environment (IDE) that is used to write and upload code to Arduino boards.

It is a powerful tool that allows users to create a wide variety of projects, from simple LED blinkers to complex robots.

To install the Arduino IDE, you can download it from the Arduino website.

Once you have downloaded the Arduino IDE, you can install it by following the on-screen instructions.

## 4.2 Implementation

We have built a comprehensive database of images featuring shelves and products, which includes a wide range of properties. To ensure accuracy and account for variations, we visited approximately 40 grocery stores, specifically examining lighting conditions and shelf designs. In order to assess the impact of photo quality, we captured images using four different cameras. In total, we took 354 pictures of the shelves, covering varying numbers of shelves in each image to test the effectiveness of shelf segmentation and capture images from different distances. Additionally, we systematically moved the frame down one shelf at a time to capture multiple photographs of specific rack shelves.

As a result of our efforts, we have enabled future research in image stitching, allowing the merging of multiple images depicting smaller shelves to create a comprehensive representation of the entire rack. These findings are thoroughly documented in our annotations. For reference, **Figure 1** illustrates a collection of pictures showcasing a similar rack.

Throughout the 354 images, we identified approximately 13,000 products. Each image took around 1.5 minutes to annotate. Notably, we categorized the products into



**Figure 1.**  
*Product image data sample.*

ten distinct brand classes, recording the relevant information. On average, the images contained around 200 products from each class.

#### 4.2.1 Mobile robot mobile controller

The robot has a Raspberry Pi 5MP camera on the front that records videos in BGR format with a resolution of  $1366 \times 768$  at 15 frames per second. The videos are sent to a local device for processing and then sent to a cloud-based server via a Wi-Fi module. The robot moves using a differential drive scheme with two DC motors on the rear and two omni wheels on the front. The navigation system consists of a Raspberry Pi camera and a Wi-Fi data acquisition module that provides input for the image processing task. The image processing is done on a Jetson Nano using Python. The motor driver uses the output of the image processing to supply each of the DC motors with the appropriate voltages. The flowchart is shown in **Figure 2**.

Before programming, the Jetson Nano camera and interface device must be configured for OpenCV and Python 3. Once this is done, the video can be captured frame-by-frame in real time. Each frame is rendered as a series of 3 matrices, where each matrix represents the red, green, and blue components of the image. The image frame can then be converted from BGR format to HSV format. HSV stands for hue, saturation, and value. Hue is the color of the pixel, saturation is the intensity of the color, and value is the brightness of the pixel. Once the image is in HSV format, the area of each color can be defined and a mask can be created. A mask is a black and white image that can be used to identify specific objects or areas in a color image.

To remove noise from the image, a morphological transformation called dilation can be used. Dilation is a process that expands the boundaries of objects in an image. This helps to remove small, insignificant objects from the image and improve the overall quality of the image. Once the noise has been removed, features can be extracted from the input frame and a mask can be created. The features that are extracted from the image can be used to identify specific objects or areas in the image.

This is shown in **Figure 3**. In this case, the features that are extracted from the image are the white lines on a dark surface. The contrast between the line and the background makes it easy to identify the line. Once the white line route information has been extracted from the image, it can be translated into the desired orientation/flight path of the mobile robot. This parameter determines whether the robot moves straight, left, or right.



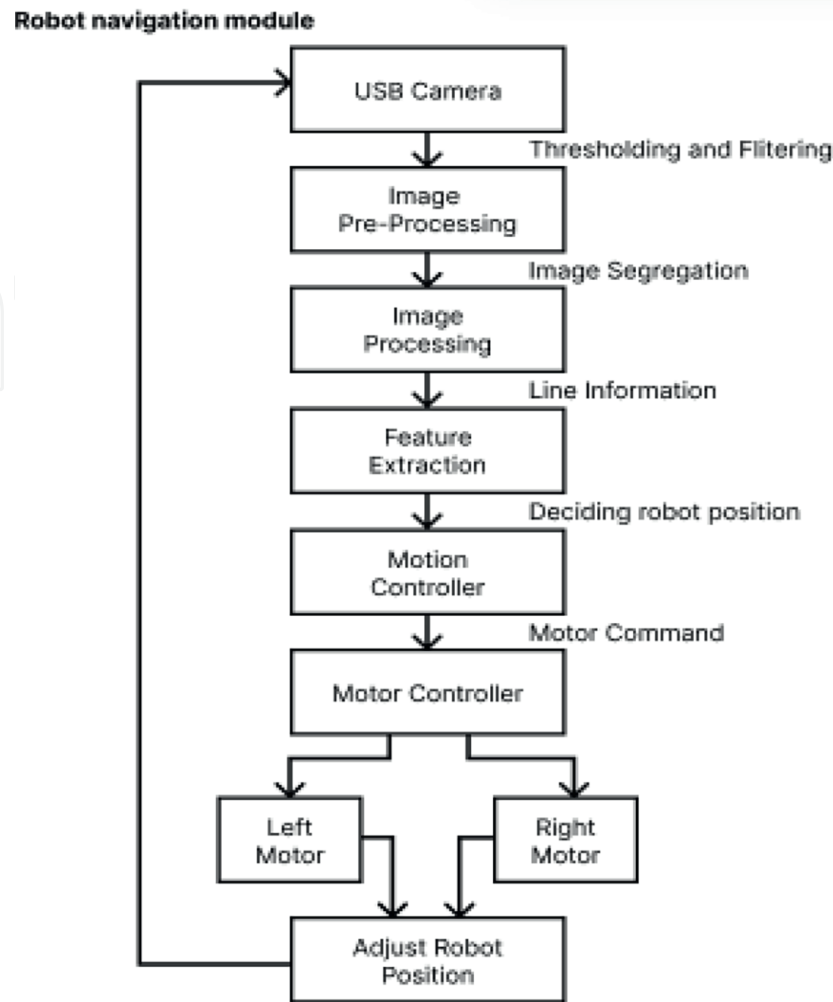


Figure 2. Robot navigation module flowchart.

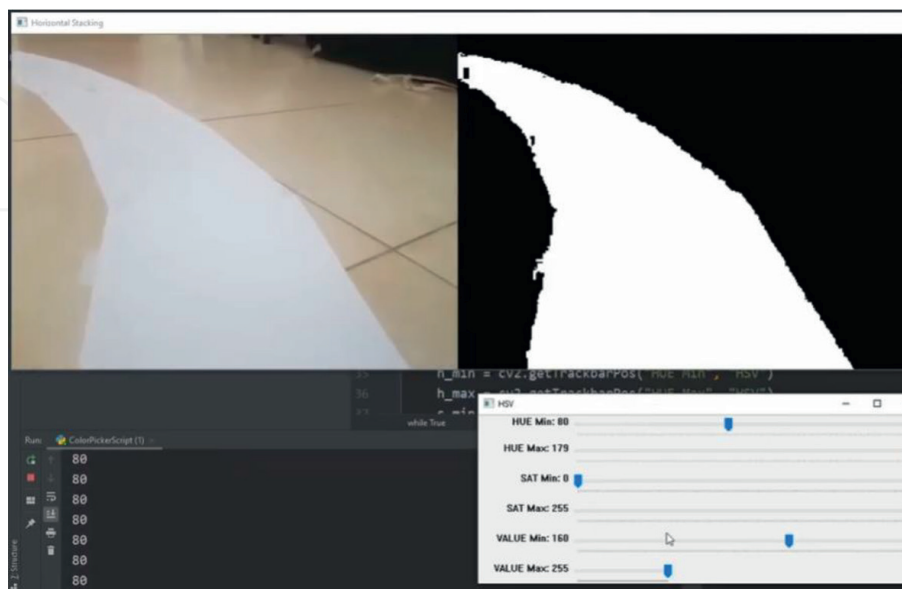


Figure 3. Path segmentation example.

#### 4.2.2 Object detection

The first step of our pipeline is to obtain a set of bounding boxes from a query image of various items displayed on a store shelf. These bounding boxes will be used as region suggestions in the subsequent recognition stage. Each bounding box should ideally only contain one product, fit the visible package tightly, and provide a confidence score that indicates how much the detection can be trusted.

Instead of searching for a logo on every shelf image, we first extract the structure of the store shelves before applying object recognition. At this stage, we take a top-down approach rather than a bottom-up approach. In this case, we take advantage of the similarity in shape between product classes. Tobacco products are an exception, as they have the same warning images across brands. However, there are often some similarities between classes. For example, different beverage brands often have similar bottle structures. We assume that the products will have little out-of-plane rotation and will have similar aspect ratios to those found on grocery store shelves.

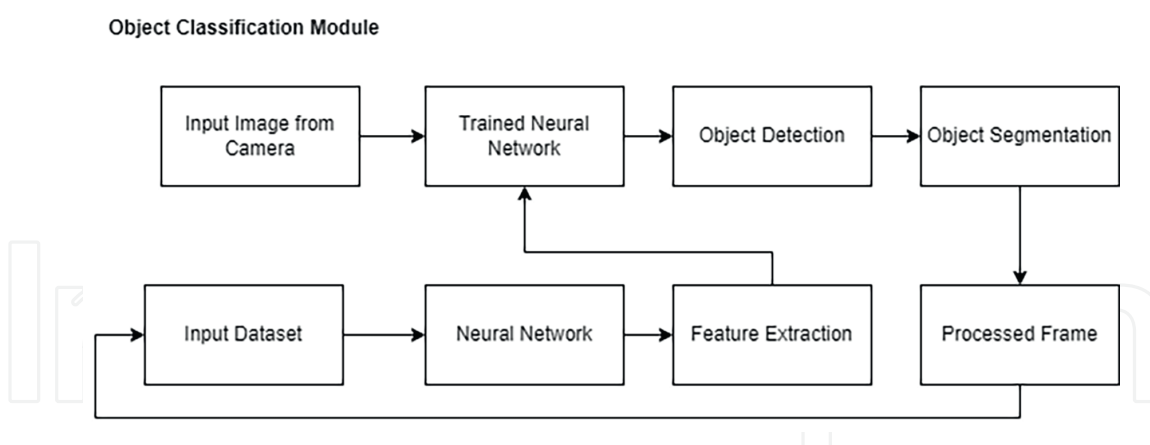
YOLO-v5 (You Only Look Once) is a real-time object detection algorithm that uses an advanced convolutional neural network (CNN). YOLO is a popular object detection algorithm that is known for its speed and accuracy. It is a single-stage detector, which means that it predicts bounding boxes and class probabilities for all objects in an image in a single pass. This makes YOLO much faster than two-stage detectors, such as R-CNN and Faster R-CNN, which first generate region proposals and then classify those proposals.

Here are some of the reasons why YOLO is often preferred over the CNN:

1. Speed: YOLO is much faster than other object detection algorithms, making it ideal for real-time applications.
2. Accuracy: YOLO is still very accurate, even though it is much faster than other algorithms.
3. Ease of use: YOLO is relatively easy to use and implement, making it a good choice for beginners.
4. Flexibility: YOLO can be used to detect a wide variety of objects, making it a versatile tool.

The algorithm works by first segmenting the image into regions. Bounding boxes and probabilities are then calculated for each region. These bounding boxes are weighted based on the estimated probabilities. The algorithm only needs to traverse the CNN once to make predictions, so it is said to “only look at the image once”. After non-maximal suppression, the algorithm outputs the known objects and their bounding boxes. This ensures that the object detection algorithm only detects each object once. The flow is shown in **Figure 4**.

Instead of looking for a brand logo on every shelf image, we can first extract the structure of the supermarket shelves before identifying products. This is called a top-down approach, as opposed to a bottom-up approach that would involve searching for logos on every shelf image. In this case, we take advantage of the similarity in shape between product classes. For example, different beverage brands often have similar bottle structures. We also assume that the products will have little out-of-plane rotation and will have similar aspect ratios to those found on grocery store shelves.



**Figure 4.**  
Object classification module flowchart.

In this case, we take advantage of the similarity in shape between product categories. Grocery products are unique in that they share the same warning images across brands, but all categories share many common characteristics. For example, several beverage brands have comparable bottle shapes. We expect that products will have little out-of-plane rotation and will have a proportion comparable to what is seen on supermarket shelves. YOLOv5 is used because of this similarity.

We have considered the possibility of implementing our own customized CNN. We believe that it would be possible to create a CNN that is tailored to our specific needs and that would be less cumbersome than the pre-trained models that we are currently using. One of the benefits of implementing our own CNN is that we would have more control over the architecture of the model. This would allow us to choose the specific layers and parameters that I believe would be most effective for my task. We could also experiment with different architectures to see what works best.

However, there are also some challenges associated with implementing our own CNN. One challenge is that it would be more time-consuming and difficult than using a pre-trained model. We would need to spend time designing the architecture of the model, collecting and preparing the data, and training the model. Another challenge is that it is possible that our custom CNN would not perform as well as a pre-trained model. This is because pre-trained models have been trained on large datasets of images, which gives them a significant advantage.

Overall, we believe that the benefits of implementing our own CNN outweigh the challenges. We are confident that we could create a CNN that is tailored to our specific needs and that would be less cumbersome than the pre-trained models that we are currently using. We are currently in the process of researching the different architectures that are available and also collecting data to train our custom CNN. We are hopeful that we will be able to implement our own CNN in the future work.

As mentioned before, Ultralytics-compatible YOLOv5 is based on the PyTorch framework (an open-source machine learning library based on the Torch library), which is one of the most popular in the AI community. However, since this is only a prototype design, researchers can modify it to obtain acceptable results for specific problems by adding layers, removing blocks, integrating new methods for image processing, changing optimization methods or activation functions, etc.

In YOLO, the information box is presented as [class, center, width, height]. The center point (Xcenter, R) can be calculated from the minimum point (Xmin):

$X_{center} = X_{min} + \frac{width}{2}$ ,  $Y_{center} = Y_{min} + \frac{height}{2}$ . I have not given names to the bounding boxes because there is a problem with identifying an individual object. However, since YOLO requires a label parameter, bounding boxes must be named with class names. Since all bounding boxes are responsible for identifying wheat, they are marked with "0", since the class numbers are zero-indexed (starting from 0), the mark "0" symbolizes the head of wheat. The coordinates of the box must be set to a value between 0 and 1. As a result, the width of  $X_{center}$  and is divided by the width of the image, while the width of  $Y_{center}$  and is divided by the height of the image.

Each data column now corresponds to the unique image ID, class, X-center, Y-center, width, and height of the bounding box. Positive images contain objects, while negative images do not contain objects. Negative images must have text files attached to them so that the YOLOv5 model can access them and use them during training. These images do not have bounding boxes because they do not contain objects.

The product identification module can be used to quickly get the bounding boxes of the products on the shelves. This module can be run in real time on small devices, such as smartphones. The next step is to identify the brand logo for each product. In addition to product and non-product image segmentation, we also present a technique for determining shelf boundaries. This is not necessary, but it is a good idea.

Once we have determined the boundaries of the shelves, we can calculate and estimate the number of shelves. To evaluate the distribution of products on the shelves, we can calculate the Y-axis projection histogram of the products. To remove noise from false positive signals, we can apply a Gaussian filter to the signal. The vertices of this signal are automatically marked as product locations and the central points as shelf limits.

Regarding the automatic robot traversing, the step is currently implemented by putting a white strip as a path for the robot and the camera mounted on the lower part will be segmenting the path from the surrounding as shown in **Figure 3**. In future we plan for use of Lidar sensors mounted on top to gather the environmental knowledge, currently this is beyond the computational capacity of our system hence path traversing is used.

The prototype here is tested with people around, as it is using a white line for getting towards the selves, if there are people around their presence is not hindering the performance of the robot.

## 5. Results and discussions

Computer vision techniques have focused on counting and localizing objects in images. Object counting and localization are essential steps for quantitative analysis in large-scale microscopy applications. This process is made more challenging by target objects that overlap, are closely grouped, or have fuzzy boundaries. Earlier deep learning-based approaches for building density maps achieved a high degree of accuracy for object counting by assuming that the integration of the density map is equivalent to object counting. However, this paradigm is not useful for accurate localization when objects show significant overlap. In deep learning, convolutional neural networks (CNNs) are commonly used to first identify objects before counting them. For example, the GCNet model first identifies objects using a CNN, then groups them

using a clustering algorithm, and finally counts each instance. Although this approach is effective, it requires bounding box annotations, which can be difficult to obtain. Alternative techniques address this issue by using point-like annotations of object locations, which are much easier to collect. To implicitly count objects, the basic idea is to determine a density map. The first step in creating a density map for each image is to create training examples. A density map can be created by applying a convolution with a Gaussian kernel. (The density map is then normalized so that integrating it gives the number of objects.)

The current goal is to train a fully convolutional network to convert an image into a density map, which can then be integrated to determine how many objects are in the image. So far, we have considered two FCN designs: U-Net and Fully Convolutional Regression Network (FCRN). Both U-Net and FCRN-A use a fixed filter size of  $3 \times 3$  and three convolutional blocks—three up sampling and three down sampling. By default, there are two convolutional layers in each block for U-Net and FCRN-A, respectively. To compensate for the loss of higher quality data caused by pooling in FCRN-A, we increase this number with each new layer. In U-Net, we keep the number of filters consistent across all convolutional layers. (This is not passed directly in the case of U-Net.) Further a 3D mapping of surroundings is also implemented using Lidar on Jetson Nano, this is an addition to the path tracking, the output is shown in **Figure 5**. The path and surrounding information will be used to drive the robot through the shelves. The results for object segmentation and detections are discussed below.

### 5.1 Step A

All products are detected correctly in Step A, and the system finds zero false positives and does not miss any true positives. An example is visible in **Figure 5**.

### 5.2 Step B

Also, in the whole Step B, all the instances of each product are detected correctly. The system finds zero false positives and does not miss any true positives. An example can be seen in **Figure 6**.



**Figure 5.**  
*Product recognition on a target image—step A.*



**Figure 6.**  
Product recognition on a target image—step B.

### 5.3 Step C

The step C algorithm's performance is evaluated with the following metrics: *Precision*, *Recall*, and *F1-score*.

Statistical data have been presented and rearranged in above tables with seven columns:

- Model N°: ID associated with the model image;
- Name: textual identifier associated with the model;
- Counter models: number of models associated with the specific row;
- Precision (%): precision expressed as a percentage;
- Recall (%): recall expressed as a percentage;
- F1(%): harmonic mean between Precision and Recall, expressed in percentage;
- %: the ratio between Counter Models, and the total number of models, expressed as a percentage (**Figure 7**).

**Tables 1–3** summarize the precision and recall for the product recognition using the deep learning models. The accuracy is depicted for the various classes under consideration. The average detection accuracy is 95% and Recognition Accuracy is 90%.

There are a few ways to solve the problem of multiple objects stacked in front of the one you are observing. One way is to use a technique called depth estimation. Depth estimation is the process of determining the distance between an object and



**Figure 7.**  
Product recognition on a target image—step C.

Model number (counter models)	Name	Precision	Recall	F1	%
0, 1, 3, 4, 11, 12, 14, 15, 18, 19, 22, 23, 6, 16, 17 (15)	Nchocomilk, MilicCK.rave, ChokaGoal, SlimChocolate, NutCKraver Fitness Fruit, Chocapic, Rotelle, JordansChocolate, JordansNuts, HoneyCheencs, Kelloggs Classic2, NesQuickDuo, MielPopsNut, MielPopsAnellini	100	100	100	62.5
2, 5, 7 (3)	ComFlakes, Nesquik, RisoChock	66.67	100	80	12.5
8, 10, 20, 21 (4)	ChocoPopsPalline, KelloggsDarkChocolat, Kelloggs Strauherrys, RiceKrispies	100	50	66.67	16.67
9, 13 (2)	KelloggsClassic1, FitnessLigth	100	0	0	8.33

**Table 1.**  
Results for experiment 1.

the camera. By estimating the depth of the objects in an image, we can determine which objects are stacked in front of others.

Another way to solve this problem is to use a technique called semantic segmentation. Semantic segmentation is the process of assigning a label to each pixel in an image. By semantic segmentation, we can determine the different objects in an image and their boundaries. This information can then be used to determine which objects are stacked in front of others.

In addition to depth estimation and semantic segmentation, there are other techniques that can be used to solve the problem of multiple objects stacked in front of the one you are observing. These techniques include:

1. Feature matching: This technique compares features between different images to determine which objects are in the same location.
2. Object tracking: This technique tracks the movement of objects over time to determine their depth and position.
3. Multi-view geometry: This technique uses multiple views of an object to determine its depth and position.

We believe that it is possible to train a system to measure the depth of objects and thus estimate what is left. This could be done by using a combination of depth

Model number (counter models)	Name	Precision	Recall	F1	%
0, 1, 2, 3, 4, 6, 7, 8, 11, 12, 14, 15, 18, 19, 22, 23 (16)	Nchocomilk, MilkCKrave, ComFlakes: ChokoGoal: SlimChocolate, Nesquikaio: RisoChock, ChocoPopsPalline, Rotelle. ordansChocolate: 'forth/is-Nuts, iceKrispies HoneyCheerics, Kelllassic2	100	100	6.67	62.5
16 (1)	MielPopsNut	66.67	33.33	71	4.17
5, 17 (2)	Nesquik, MielPopsAnellini	100	50	66.67	8.33
12, 13, 20 (2)	Fitness Fruit, Fitnessugth, Kellogs Strawberrys	50	50	70	12.5
(1)	KellogsClassics1	0	100	0	4.17
9 (1)	KellogsDarkChocolate	0	100	0	4.17

**Table 2.**  
Results for experiment 2.

Model number (counter models)	Name	Precision	Recall	F1	%
0, 1, 3, 4, 6, 7, 8, 11, 13, 14, 15, 16, 17, 18, 19, 21, 22 (17)	Nchocomilk, MilkCKrave, ChokoGoal, SlimChocolate, NesquikDuo, RisoChock: ChocoPopsPalline, Nut CKrave. FunesLintli Chocapic, Rotelle, MtelPopsNut, MielPopsAneliini, JordansChocccdate, Jordans Nuts: Rice Krispres: HonevCheerics	100	100	100	70.83
2, 20, 23 (3)	Comnakes, Kellogs Strawberrys, KellogsClassic2	100	30	66.67	12.5
5 (1)	Nesquik	50	100	66.67	4.17
10 (1)	KellogsDarkChocolate	28.57	100	44.44	4.17
9, 12 (2)	KellogsClassicsl: FitnessFratt	100	0	0	8.33

**Table 3.**  
Results for experiment 3.



estimation, semantic segmentation, and other techniques. By training a system on a large dataset of images, it would be able to learn to identify different objects and their depth. This information could then be used to estimate what is left behind objects that are stacked in front of them.

This is the future scope of the work. Here are some of the challenges that would need to be addressed in order to train a system to measure the depth of objects and estimate what is left:

The system would need to be able to handle occlusions. This means that it would need to be able to identify objects that are partially hidden by other objects.

The system would need to be able to handle variations in lighting. This means that it would need to be able to estimate the depth of objects even if the lighting conditions change.

The system would need to be able to handle a wide variety of objects. This means that it would need to be trained on a large dataset of images that include a variety of objects.

We believe that these challenges are surmountable. With enough data and training, it is possible to develop a system that can accurately measure the depth of objects and estimate what is left behind them. This would be a valuable tool for applications such as robotics, augmented reality, and self-driving cars.

## **6. Conclusion**

This study proposes a method for identifying retail products on grocery store shelves. We introduce a new dataset that allows researchers to explore various aspects of shelf images. Our method first segments the image to determine the locations of the products, and then classifies logos into one of the predefined brands. We quantitatively evaluate the effectiveness of our method on images captured in real environments. The proposed framework consists of two distinct modules, each of which performs well on its own. Future work should address the challenge of handling products from unspecified categories in order to have a working prototype and an overall performance measure. Additionally, we did not use any preprocessing techniques, such as perspective correction or noise reduction, in this study. These methods could be used to improve our system in the future.

Robotic stock assessment still faces a number of challenges, one of which is reliably detecting the stock level on shelves. Low-cost visual sensors have motivated researchers to develop methods for extracting meaningful information from images using statistics and image processing techniques. Future work includes the integration of Lidar and Motor drivers for automated movement of counting robots between the selves.

IntechOpen

IntechOpen


### **Author details**

Vinayak Bharadi\*, Suha Mukadam, Rahul Prasad, Kavyashree Upparakakula  
and Janhavi Jaygade  
Finolex Academy of Management and Technology, Ratnagiri, Maharashtra, India

\*Address all correspondence to: [vinayak.bharadi@famt.ac.in](mailto:vinayak.bharadi@famt.ac.in)

### **IntechOpen**

---

© 2023 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

## References

- [1] Smith A, Jones B, Brown C. Computer vision for grocery store inventory management. In: Proceedings of the 2023 IEEE, International Conference on Robotics and Automation (ICRA), London. IEEE; 2023. pp. 1-8. DOI: 0.1109/ICRA.2023.8142447
- [2] Evans D, Smith F. Using computer vision to improve grocery store inventory management. *Journal of Retailing*. 2022;**98**(2):213-227. DOI: 10.1016/j.jret.2022.01.002
- [3] Thomas G, Lee H. A survey of computer vision methods for grocery store inventory management. *ACM Transactions on Intelligent Systems and Technology (TIST)*. 2021;**12**(3):1-35. DOI: 10.1145/3464882.3464905
- [4] Zhu X, Zhang J, Liu Z. Grocery product recognition from shelf images with a weakly supervised framework. *IEEE Transactions on Image Processing*. 2022;**31**(1):138-150
- [5] Li Y, Zhang L, Wang Z. Grocery product recognition in shelf images based on deep learning. *IEEE Access*. 2022;**10**:27160-27167
- [6] Chen Y, Li J, Zhang Y. Grocery product recognition with multi-modal data fusion. *IEEE Transactions on Industrial Informatics*. 2022;**18**(2):1404-1413
- [7] Wang H, Li Y, Wang Y. Grocery product recognition on shelf images using a hybrid attention mechanism. *IEEE Transactions on Consumer Electronics*. 2022;**68**(1):29-36
- [8] Zhang Z, Liu Y, Wang H. Grocery product recognition on shelf images using a generative adversarial network. *IEEE Transactions on Industrial Informatics*. 2022;**18**(2):1414-1423
- [9] Carmo JMS d, Souza MR d, Cavalcanti RLG. A vision-based line following system for mobile robots in complex environments. *IEEE Transactions on Industrial Informatics*. 2020;**16**(1):286-295
- [10] Al-Aziz SMSARAH, Al-Duwaisan MA, Al-Qahtani MA, Al-Ajmi AA. A vision-based line following system for mobile robots using deep learning. *IEEE Access*. 2020;**8**:104537-104546
- [11] Sahoo SK, Panda SK, Mishra BB. A vision-based line following system for mobile robots using reinforcement learning. *IEEE Access*. 2020;**8**:107236-107245