Instituto Tecnológico
y de Estudios Superiores de Occidente
Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática
Especialidad en Sistemas Embebidos

Computer vision application proposal for smart inventory systems in convenience store reach-in refrigerators

TRABAJO RECEPCIONAL para obtener el **GRADO** de **ESPECIALISTA EN SISTEMAS EMBEBIDOS**

Presentan:
Contreras López Rodrigo
Ortega Tapetillo Juan Carlos

Asesor **González Jiménez Luis Enrique**

Tlaquepaque, Jalisco. Agosto de 2022.

Computer vision application proposal for smart inventory systems in convenience store reach-in refrigerators

Juan Carlos Ortega Tapetillo
Especialidad en Sistemas Embebidos
ITESO
Guadalajara, México
juan.ortega@iteso.mx

Rodrigo Contreras López
Especialidad en Sistemas Embebidos
ITESO
Guadalajara, México
rodrigo.contreras@iteso.mx

Luis Enrique González Jiménez
Especialidad en sistemas embebidos
ITESO
Guadalajara, México
luisgonzalez@iteso.mx

***Abstract*—Inventory systems in reach-in refrigerators employ manual or smart inventory outdated methods, although efficient, new methods like computer vision could render better results in less time, with less human intervention. The objective of this work proposes a computer vision system to acquire an inventory of products placed in reach-in convenience store refrigerators. A comparative of different computer vision object recognition models was performed to select the most appropriate model for the application. Then, based on the model characteristics and the application requirements, a YOLOv4 object recognition model was selected. Along with a 2-dimension camera positioning rig to capture a live video feed of the products to count for the inventory. Future works could include a real size prototype and further development into a commercial product.**

Keywords—Computer vision, smart inventory, object recognition, YOLOv4, reach-in refrigerator.

I. Introduction

Smart inventory management has a been a great solution for business workflow and operations, nonetheless, the benefits deeply rely on the speed and accuracy of the data acquisition scheme. These systems acquire data through different technological means such as barcodes and readers, QR codes and cameras, RFID tags, and other systems operated by employees. This data acquisition scheme becomes increasingly difficult when handling individual items, and even more when the work environment is not ideal. For instance, convenience store refrigerators have many single products in a small space where product manipulation and positioning are required.

Computer vision and deep learning have proved to be great solutions for simple object detection and data acquisition since the first object detection convolutional neural network (CNN) model was launched in 2012 [1]. The region-based convolutional neural network (R-CNN) is an object detection model based on CNN, a part of the deep learning family of algorithms [2] [3]. These algorithms have been used in artificial intelligence (AI), driving assistants, face detection and remote sensing applications.

After more than 10 years since Viola-Davis object detection algorithm [4] was released, the object detection algorithms based on features, progressed at a slow pace. This set in motion data scientists across the globe to find a better way to detect objects. In 2012 A. Krizhevsky *et al.* applied a convolutional neural network to train a large dataset and improve the object classification. This led R. Girshick *et al.* to propose R-CNN in 2014. After R-CNN, deep learning object detection has been growing exponentially with the development of faster and lighter deep learning models.

The objective of this work is to propose a smart inventory system with computer vision to count products placed in reach-in convenience store refrigerators. By diminishing human interaction on the data acquisition, the speed and accuracy of inventories will increase as well as the interval between data collection will be shortened.

II. Model and hardware selection

A. Selecting an object recognition model

The deep learning object detection algorithms have evolved into several application-oriented models. The latest state-of-the-art models range in accuracy, inference speed, and computational costs. To select an appropriate model the trade-offs must be weighed against each other.

The top candidates for our application include:

- Faster R-CNN [5]
- YOLOv4 [6]

- Single Shot Detection (SSD) [7]

Extensive research into these 3 models led to the next conclusions about each system [8]:

- SSD: Not very effective when it comes to smaller objects as higher resolution images reduce the overall performance.
- Faster R-CNN: Significantly more accurate than SSD and YOLO but significantly slower, as it requires multiple passes over a single image, not ideal for real-time applications.
- YOLOv4: Ideal for live-video feed and good accuracy.

With the previous points taken into account the best two candidates for our use-case were the Faster R-CNN and YOLOv4, as the SSD would not work well with small objects like the ones the proposed system is bound to count. To select the appropriate object recognition model for the proposed system the following conditions were considered: proximity between objects to detect, lighting conditions, and object size. Besides the environmental conditions, these system requirements should be considered for the selection of the object detection model: close object detection, object counting accuracy, computational cost, and inference speed.

A Faster R-CNN model would require taking still-frame photos of the products displayed in the refrigerator. This proposes several challenges. First, a wide-angle view is needed to capture all the products displayed, but this would only work for products on the top tier or the first layer. If the camera is moved from tier to tier, different capture angles would be needed to avoid occlusions. This approach would imply a higher system complexity, therefore, the recognized objects between frames would need to be identified and subtracted from frame series in the same tier illustrated in the following diagram (*Fig. 1)*.

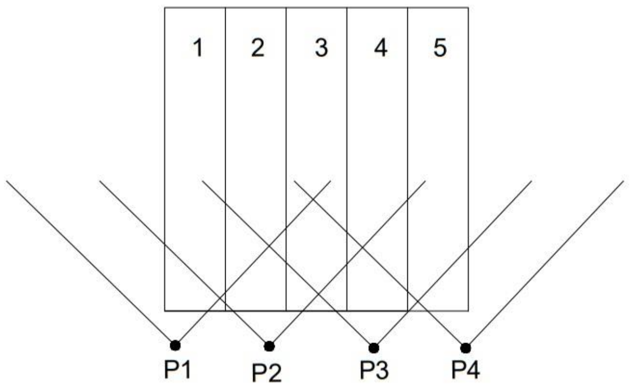

Fig. 1 Camera positions for a 90° field of view lens to avoid occlusions.

A YOLOv4 model would require capturing a video live-feed and a moving camera. This proposes the same level of complexity in hardware as the Faster R-CNN approach but less logic complexity. A region of interest (ROI) line could be applied to the model to count detected objects that move from one region to another (*Fig. 2)* as proposed by A. Parico *et al.* in their real time fruit counter [9].

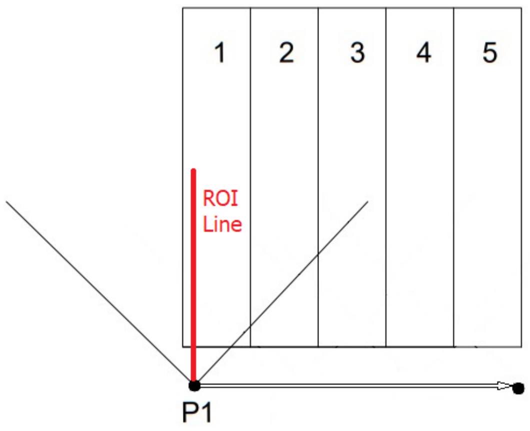

Fig. 2 Camera with the video feed and ROI line.

To determine the object recognition model to be used, the available hardware options have to be considered.

B. *Selecting the hardware*

The system needs to be portable enough to reside in the back part of a reach-in refrigerator, it should support low temperatures, and have a low profile to avoid blocking the normal operation of the reach-in shelves.

Considering the previously mentioned requirements, the basic hardware components for a system with these characteristics are:

- System-on-board computer with enough processing power to run computer vision tasks
- Computer-vision oriented processor to run an object recognition model
- Small camera

The candidate object recognition models for the system also require a camera with a wide-angle field of view (FOV) (>90°) or a moving camera with a narrower field of view (≤90°).

Wide angle lenses distort the images, so the reproduction fidelity is lowered. This implies that a calibration step must be setup [10] before feeding the images through the object recognition models. A calibration step would require extra processing power even before using the object recognition model. Additionally, the camera needs to move through predefined positions over the length of the shelf tiers, which in turn, requires position sensing and software tied for each position.

A narrow FOV requires the camera to move in a smooth motion through the length of the shelf tiers but does not require image calibration since the distortion is very low in these types of lenses. This maintains a high reproduction fidelity. In addition to this, position sensing becomes much simpler as only tier-number and start/end sensors are needed.

Simplicity is essential when designing a system intended to be deployed in a commercial environment. Thus, a narrow FOV camera and a live video-feed with the YOLOv4 object recognition model were selected as the main components of the system. The YOLOv4 model as any deep learning algorithm performs a large number of parallel operations which demand high computational power. This traditionally is achieved with graphic processing units (GPU) [6] in a computer environment. Since the application of the system is intended to be portable, an embedded hardware approach is needed.

Embedded computer vision requires low power consumption while having high computational capabilities. Repurposing a GPU to train and deploy object recognition models is very effective in research environments, but it is not an adequate solution for portable applications. Nonetheless, deployment in a commercial product can be set-up with pre-trained models; this allows the hardware to be downscaled to any processor that can perform a high number of parallel operations in a short period of time.

A great solution for embedding computer vision is found in a specific system architecture that can take any Linux-capable system and a dedicated vision processing unit (VPU) to outsource the real-time model processing [11]. A Raspberry Pi system-on-board computer is more than capable of running dedicated Linux operating systems and instructing to allocate external processing though USB interfaces. A Raspberry Pi 4b miniature computer was then selected as the base for the proposed system.

Dedicated VPUs for embedded applications were developed by Intel Corporation in recent years under the name Movidius [12]. The Movidius VPUs have a software driven multi-core memory subsystem with multiple ports and caches that can be configured to allow a wide range of workloads, and wide and deep register files in conjunction with a variable length long instruction word (VLLIW) that controls multiple functional units. The Movidius Myriad X has 16 highly parallelizable vector processors named streaming hybrid architecture vector engines (SHAVE) that provide highly sustainable performance efficiency consuming ~1 Watt. The SHAVE processor is a hybrid stream processor architecture that combines the best features of GPUs, digital signal processors (DSP) and reduced instruction set computers (RISC).

The Movidius Myriad X has a native 4K image processor pipeline which supports image sensors connected directly to the VPU to be used on-device for computer-vision oriented cameras.

The OpenCV AI Kit (OAK)[13] employs this VPU with a 4K sensor and a 81° FOV lens pipelined directly to the processor. This data path avoids a bottleneck in the proposed system by transmitting the model results to the Raspberry Pi (24Kbs data stream) whereas, a configuration of an external 4K camera video-feed (2.1Gps data stream) has to go through the Raspberry Pi to the VPU and back to the computer. An OAK-1 camera was selected for the system.

C. *Moving the camera*

With the YOLOv4 model running on the OAK-1 hosted on the Raspberry Pi 4b with pre-trained models, the system is capable of detecting and counting the objects that move across the ROI line. In a reach-in refrigerator, the objects are static. In order to "move" the products across the ROI line, the camera moves horizontally along the shelve tiers. To count each product the camera also moves vertically from tier to tier. This represents a challenge as a certain level of control and repetition must be achieved to position the camera every time the counting process is performed, a system similar to that of a modern 3D printer is proposed.

Three 17HS8401 NEMA-17 stepper motors with 1.8° per step and a load capacity of 5.5kg/cm are used to provide an x-axis and y-axis coordinate movement. These motors are mounted on rolling carts along a 20x20mm v-slot extruded aluminum frame with static toothed belts to provide traction (*Fig.3)*.

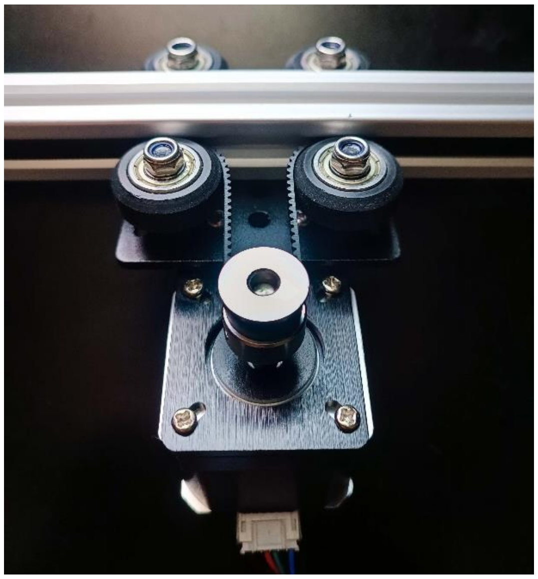

Fig. 3 Toothed belt mounted on a rail with a stepper motor.

Two motors run in parallel for the y-axis motion, while one motor runs along the x-axis. The initial and final position for the x-axis is sensed by reflective infrared optical pair sensors positioned at the start and end of the x-axis rail, while the y-axis position is sensed by a series of reflective infrared optical pair sensors positioned parallel to each level (*Fig.4)*.

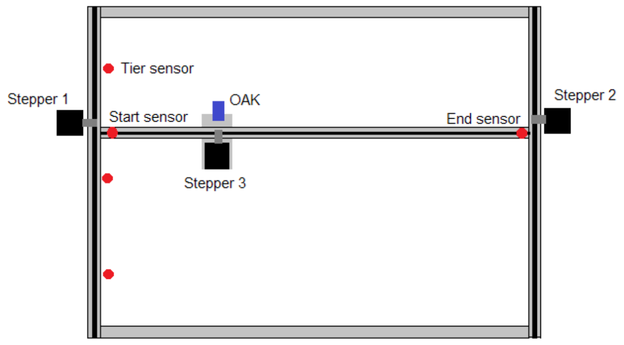

Fig. 4 Camera movement mechanism diagram.

The stepper motors are driven by A4988 stepper drivers controlled by the Raspberry Pi which receives feedback from the optical sensors. This scheme allows the camera to be moved in a smooth fashion along all the products in each tier, and to repeat this for each tier in a precise manner.

III. Model training

The YOLOv4 model requires training for this specific application to achieve the best inference confidence. The training dataset must be composed of at least 50 images of the detection scene for each product. The detection scene for this application is the camera view of the products from behind the reach-in shelf.

Training an object recognition model with modest hardware could take several hours with a medium sized dataset. For this system, training is performed through a cloud service. Cloud model training is available through different providers, however, Roboflow[14] offers this service with the most complete toolset.

To pre-train a minimal capable model, each displayed product is assigned as a unique class in the model. For a full-sized reach-in refrigerator an average of 10 classes over a 7 tier shelve is needed.

Model training should be performed with different parameters to fine tune the detection capabilities of the system, and to add any new products or new product presentations.

IV. Conclusions

Although the proposed system may be a very powerful tool for smart inventory purposes, and it is intended to improve productivity of commercial ventures, some important aspects must be considered for a full commercial product deployment.

A. *Convenience*

The proposed system if further developed into a commercial product will be able to perform accurate inventories in a small-time interval over several stores. This may be upscaled into a big-data system able to predict market tendencies and re-stocking logistics in near real-time.

The usage of the proposed system also allows the user to re-assign human power to other tasks, thus reducing the labor needed to maintain an accurate inventory. This is very convenient when the system is implemented in a large enough scale.

The portability, modularity, and low cost of the system provide a cost-efficient solution for the actual and future inventory needs.

B. *Future improvements and developments*

The system can be upgraded to obtain a better count certainty by using the object detection model in combination with an object tracking algorithm through a unique object ID system.

Acknowledgments

All our acknowledgements go to Luis Rizo, Luis Enrique González, Maria Fernanda Ortega, and Lorena Michele Brennan who pushed us to find new alternatives and new horizons.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems*, 2012, vol. 25. Accessed: Jun. 20, 2022. [Online]. Available: https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[2] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks.” arXiv, Feb. 23, 2014. Accessed: Jun. 20, 2022. [Online]. Available: http://arxiv.org/abs/1312.6229

[3] “arXiv Fulltext PDF.” Accessed: Jun. 20, 2022. [Online]. Available: https://arxiv.org/pdf/1312.6229.pdf

[4] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, Kauai, HI, USA, 2001, vol. 1, p. I-511-I–518. doi: 10.1109/CVPR.2001.990517.

[5] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” arXiv, Jan. 06, 2016. Accessed: Jul. 11, 2022. [Online]. Available: http://arxiv.org/abs/1506.01497

[6] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection.” arXiv, Apr. 22, 2020.

[7] W. Liu *et al.*, “SSD: Single Shot MultiBox Detector,” vol. 9905, 2016, pp. 21–37. doi: 10.1007/978-3-319-46448-0_2.

[8] S. Srivastava, A. V. Divekar, C. Anilkumar, I. Naik, V. Kulkarni, and V. Pattabiraman, “Comparative analysis of deep learning image detection algorithms,” *J. Big Data*, vol. 8, no. 1, p. 66, Dec. 2021, doi: 10.1186/s40537-021-00434-w.

[9] A. I. B. Parico and T. Ahamed, “Real Time Pear Fruit Detection and Counting Using YOLOv4 Models and Deep SORT,” *Sensors*, vol. 21, no. 14, Art. no. 14, Jan. 2021, doi: 10.3390/s21144803.

[10] J. M. I. Zannatha, “Calibración y Corrección de la Distorsión Radial en Sistemas de Visión para Robots,” p. 7.

[11] A. Pester and M. Schrittesser, “Object detection with Raspberry Pi3 and Movidius Neural Network Stick,” in *2019 5th Experiment International Conference (exp.at’19)*, Jun. 2019, pp. 326–330. doi: 10.1109/EXPAT.2019.8876583.

[12] “Intel® Movidius^{TM} Vision Processing Units (VPUs),” *Intel*. https://www.intel.com/content/www/us/en/products/details/processors/movidius-vpu.html (accessed Jul. 25, 2022).

[13] “OAK-1,” *Luxonis*. https://shop.luxonis.com/products/oak-1 (accessed Jun. 21, 2022).

[14] “Roboflow: Give your software the power to see objects in images and video.” https://roboflow.com/ (accessed Jul. 26, 2022).