Ingeniería y
Competitividad
Vol 26 (1) 2024; doi: 10.25100/iyc.v26i1.13230

Vol 26 (1)

ISSN
0123-3033
e- 2027-8284

Original Research

Human action detection for inventory control using computer vision

Detección de acción humana para el control de inventario utilizando visión por computadora

Francisco Bernal-Baquero[1] Darwin E. Martínez[1]

[1]Engineering Department, Sergio Arboleda University, Bogotá, Colombia

Abstract

Computer vision (CV) can be a process that facilitates some tasks in inventory management, through this process a permanent analysis of an inventory can be performed and thus keep record of all movements made, delivering an instant report when required. This means an improvement in security, since by keeping a strict control of the existing elements in the inventory it is possible to know if an element belongs or not to an inventory or when an element is removed or added, after this need for inventory control, the need arises to design an intelligent system that can facilitate inventory control. Through the combination of 2 frameworks, the creation of an algorithm capable of performing the identification and counting of objects, as well as the identification of the hand to determine when a human manipulation is performed to the inventory. To achieve this objective, two algorithms were used: MediaPipe and YOLOv5 combined with the COCO dataset, the first one was used for hand detection and the second one identifies and counts the objects. After testing the algorithm, it was determined that the hand recognition of MediaPipe had an accuracy of 96% and the detection and classification of objects using YOLO was 43.7%. Challenges for the algorithm were overlapping, occlusion/self-occlusion of objects, or loss of focus of items due to the sensor.

Resumen

La visión por computadora (VC) puede ser un proceso que facilite algunas tareas en la gestión de inventarios, por medio de este proceso se puede realizar un análisis permanente de un inventario y así mantener registro de todos los movimientos realizados, entregando un reporte instantáneo cuando sea requerido. Esto supone una mejora en la seguridad, ya que al mantener un control estricto de los elementos existentes en el inventario se puede saber si un elemento pertenece o no a un inventario o cuando se retira o agrega un elemento, tras esta necesidad de control de inventario, surge la necesidad de diseñar un sistema inteligente que pueda facilitar el control de inventarios. Mediante la combinación de 2 frameworks, se realiza la creación de un algoritmo capaz de realizar la identificación y conteo de objetos, así como la identificación de la mano para determinar cuándo se realiza una manipulación humana al inventario. Para lograr este objetivo, se utilizaron dos algoritmos: MediaPipe y YOLOv5 combinado con el dataset de COCO, el primero se usó para la detección de manos y el segundo identifica y cuenta los objetos. Después de las pruebas realizadas al algoritmo se determina que el reconocimiento de manos de MediaPipe tuvo una precisión del 96% y la detección y clasificación de objetos usando YOLO fue de 43.7%. Teniendo como retos el algoritmo la superposición, la oclusión/auto oclusión de los objetos, o la pérdida de foco de los elementos debido al sensor.

Keywords: Hand detection, Object counting, Object detection, Mediapipe, YOLO

Palabras clave: Conteo de objetos, Detección de manos, Detección de objetos, MediaPipe, YOLO..

How to cite?

Bernal-Baquero, F., Martínez, D.E. Human action detection for inventory control using computer vision, Ingeniería y Competitividad, 2024, 26(1): e-21813230.

https://doi.org/10.25100/iyc.v26i1.13230

Recibido: 14-09-23
Aceptado: 26-02-24

Correspondencia:
francisco.bernal02@usa.edu.co

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike4.0 International License.

Conflict of interest: none declared

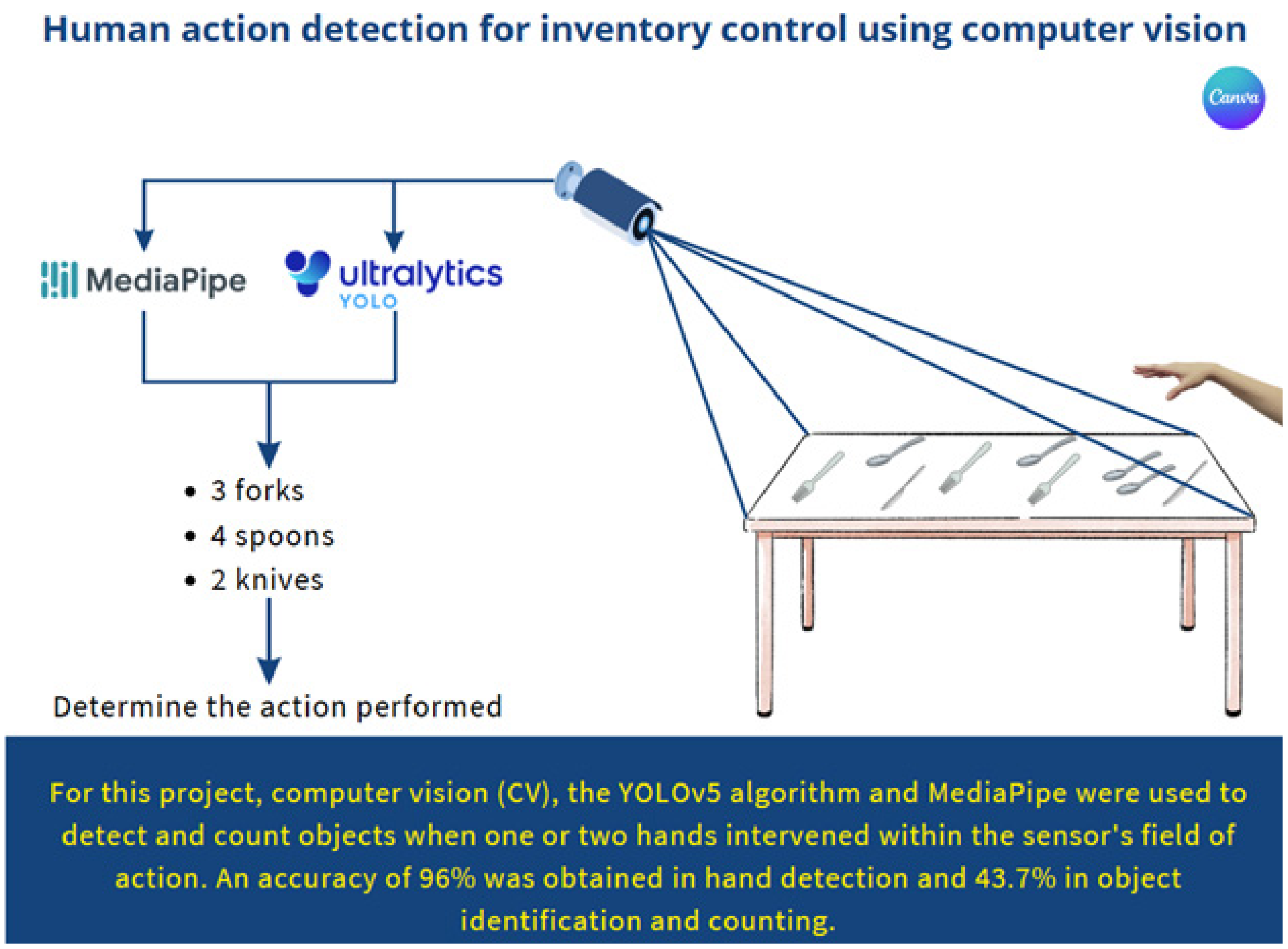
OPEN ACCESS

Why was it carried out?
We tried to answer a question: How by means of computer vision can we count the elements of an inventory when by a human interaction this is modified? The answer to this question gave us a technological alternative to inventory control, increasing security and keeping a permanent control over the elements that compose it, thus helping to keep track of the items that enter or leave the inventory.

What were the most relevant results?
The combination of MediaPipe for hand recognition and the use of a YOLOv5 trained model with a custom dataset for object recognition, resulted in a very good tool that met the requirements of the algorithm, achieving a hand recognition with an accuracy of 96% and 43.7% for objects, it is noteworthy that the YOLOv5 model was trained with very few images (about 320) so it is considered that the object recognition model performed better than expected.

What do these results provide?
The algorithm is presented as an aid in inventory control because with it you can have greater control of inventory management and know quickly how many items you have. The algorithm presented, apart from counting the objects, is capable of detecting the hands that modify the inventory and also detecting whether objects have been added or removed from the inventory, adding an extra to the control of objects in the inventory.

Graphical Abstract


Human action detection for inventory control using computer vision
Canva
MediaPipe
ultralytics
YOLO
• 3 forks
• 4 spoons
• 2 knives
Determine the action performed
For this project, computer vision (CV), the YOLOv5 algorithm and MediaPipe were used to detect and count objects when one or two hands intervened within the sensor's field of action. An accuracy of 96% was obtained in hand detection and 43.7% in object identification and counting.

doi: 10.25100/iyc.v26i1.13230

Introduction

Computer vision (CV) is a field of computer science that focuses on the development of algorithms and techniques that allow computers to “see” and understand the world around us. Computer vision includes image processing, machine learning and pattern recognition to extract information of interest from image data; it can also be used to perform tasks such as: object detection, image classification, edge detection, object tracking, among others (1).

Computer vision is increasingly used in the world of inventory (2) and security (3). According to Arias, computer vision is very useful for inventory control, as well as for detecting unwanted objects, allowing to improve security by recognizing potentially dangerous objects such as weapons or bombs and alerting users (3). On the other hand, Kalahiki proposes the use of CV to achieve a more reliable inventory control by means of several existing techniques such as: data collection, data preprocessing and transfer learning (2).

This paper seeks to solve the following question: how can we use CV to count the elements of an inventory when it is modified by human interaction? The answer to this question provides a technological alternative to inventory control, increasing security and keeping a permanent control over the elements, thus helping companies to be more effective in keeping track of items entering or leaving their inventories or identifying objects that should not be in their inventory.

Methodology

MediaPipe

MediaPipe is a framework for building applied multimodal machine learning pipelines (such as video, audio, time series data, etc.), and for different platforms (e.g., Android, iOS, Web, and perimeter devices). MediaPipe provides multiple features, including face detection, hand tracking, gesture detection, and object detection (4).

Hand tracking

MediaPipe hand tracking uses a machine learning (ML) process with two models working together:

1. A palm detector that operates on a complete input image and locates palms by means of an oriented hand bounding box.
2. A hand landmark model that operates on the cropped hand bounding box provided by the palm detector and returns high fidelity 2.5D landmarks.

By identifying the clipped palm image with the hand-held landmark model, see Figure 3, the need for data augmentation (e.g., by rotations, translations, and scaling) is drastically reduced and allows the network to focus more on the accuracy of landmark localization. In a real-time tracking scenario, a bounding box is obtained from the previous frame’s landmark prediction as input for the current frame, thus avoiding applying the detector on every frame. Instead, the detector is only applied at the first frame or when the hand prediction indicates that the hand has been lost (5).

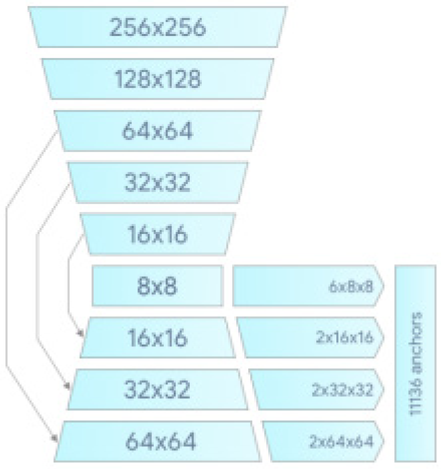


Figure 1 Architecture of the palm detector model. (5)

Palm detection model

According to Zhang (5) detecting hands is a complex task, as it requires several challenges, the model must be able to recognize hands in a variety of sizes and be able to detect them even when they are occluded or self-occluded. Unlike faces that have contrasting patterns, for example, around the eyes and mouth, the lack of such features in hands makes it comparatively more difficult to detect them reliably from their visual features alone. To solve these challenges, Zhang (5) proposes to address them with different strategies, the first is to perform palm training, rather than a hand detector, since, according to them estimating bounding boxes of rigid objects such as palms and fists is much simpler than detecting hands with articulated fingers. Since palms are smaller, the “non-maximal suppression” algorithm works even in complex cases such as a handshake.

Palms can be modeled using only bounding boxes, ignoring other aspect ratios, which reduces the number of anchors. After this process, they use an encoder-decoder feature extractor such as FPN to have greater awareness of the scene context, even for small objects, and finally, they minimize the focal loss to handle many anchors resulting from high scale variability. The high-level palm detector architecture is shown in Figure 1.

Hand reference model

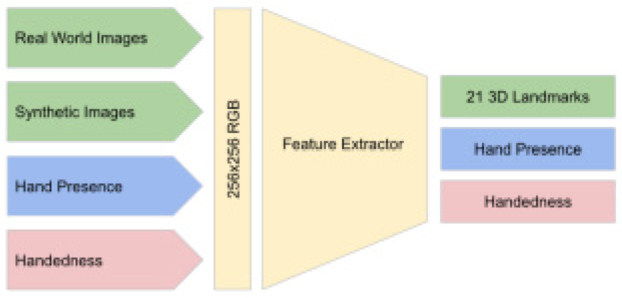


Figure 2 Architecture of the MediaPipe hand detection model. The model has three outputs that share a feature extractor. Each head is trained by corresponding data sets marked in the same color. (5)

After running palm detection on the entire image, the subsequent hand landmark model performs an accurate localization of 21 2.5D coordinates within the hand regions detected through regression. The model learns a consistent internal hand pose representation and is robust even to partially visible hands and self-hiding. The model has three outputs, see Figure 2:

1. 21 hand reference points consisting of "x, y" coordinates and relative depth, see Figure 3.
2. An indication of the probability of hand presence in the input image.
3. A binary classification of laterality of the hand, e.g., left, or right hand.

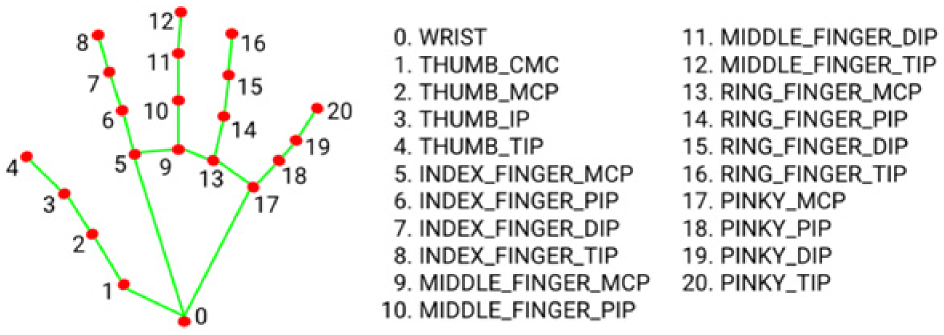


Figure 3 Hand points taken by MediaPipe for hand recognition and tracking. Image taken from MediaPipe's Hand Landmarks detection guide documentation.

You Only Look Once (YOLO)

YOLO was presented to the public in 2015 (6), due to its speed, accuracy, and ability to run in real time, its use for object detection became widespread. It is based on the Googlenet architecture (7), the operation of YOLO consists in the division of the image into a grid, where in each cell is verified if there is an object of interest inside. With this process, several bounding boxes of different sizes are generated, it must be defined with what confidence level a prediction of the existence of an object is made. A high confidence level implies that the algorithm will be stricter in detecting and classifying an object and a low confidence level will make the algorithm more permissive in detecting and classifying.

Versions

YOLOv4 and YOLOv5 introduce changes that generate an improvement in speed and performance tested in different benchmarks. YOLOv5 delivers similar results to YOLOv4, but its training time is even shorter. In addition, the development of this architecture is implemented natively in Pytorch, unlike previous versions based on Densenet.

Additionally, YOLOv5 consists of different models, which have different configurations see Figure 4, so the right model should be chosen according to the specific needs of each project, so it should be determined whether higher accuracy or speed is required. Models such as YOLOv5x and YOLOv5x6 will produce better results in terms of accuracy and detection, but have more parameters, require more CUDA memory to train and are slower to run. Ultralytics recommends using the smaller models for mobile development, and the larger models for development with cloud-based architectures.

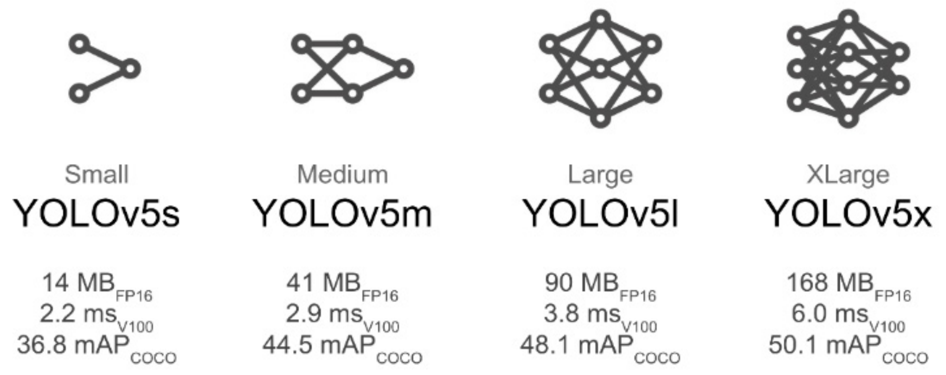


Figure 4 Analysis speed, mAP and weight of the different YOLOv5 models. Image taken from the ultralitycs github repository (8).

The diagram below shows the general methodology, see Figure 5.

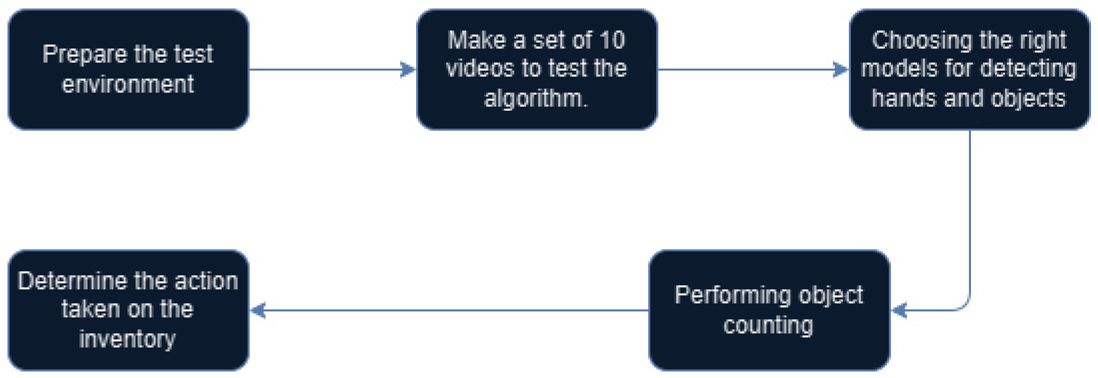

Prepare the test environment
Make a set of 10 videos to test the algorithm.
Choosing the right models for detecting hands and objects
Performing object counting
Determine the action taken on the inventory

Figure 5 Methodology performed to detect hands and identify objects in a scene to enable object control in the inventory.

Prepare the test environment

The preparation of the test environment consisted of preparing the surface for the experiment. A red and blue tablecloth was used on a table to contrast with the kitchen utensils. Additionally, a light control was carried out with two halogen lamps. To reduce the shadows that would cause an erroneous counting and detection by the algorithm.

Make a set of 10 videos to test the algorithm

A set of videos was built in order to perform different tests and with them identify the different states of the elements of interest. Each video presents a different challenge for the models, such as: obstruction or overlapping of objects, objects away from the sensor's field of view, adding or removing two or more objects at the same time, or recognizing when 1 or 2 hands enter and leave.

Choosing the right models for detecting hands and objects

Tests were performed on different models for hand and object detection, these tests used the video set to determine which algorithms performed best in detecting the objects of interest. The measure used to select the most suitable model was accuracy.

Performing object counting

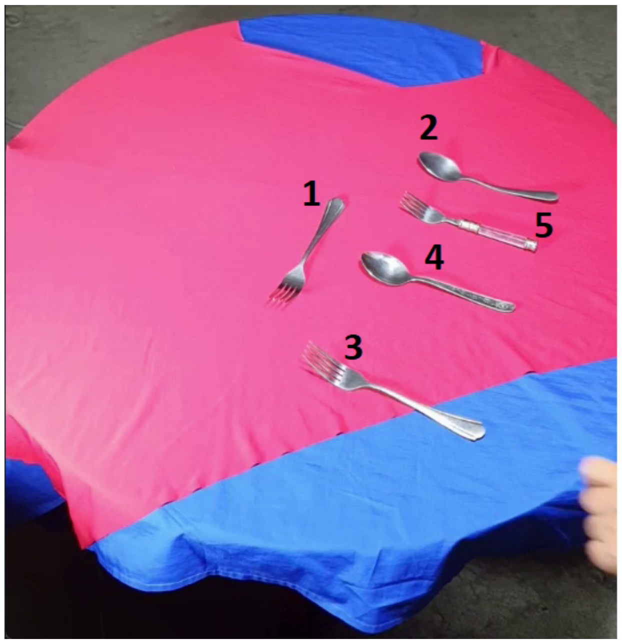
The counting of objects was performed by defining a list of objects of interest, this list serves as a control element that allows the algorithm to filter the objects of interest from the other elements, once the individualization of the objects is done, a count is performed, so the algorithm had the ability to count the objects that are of interest to the algorithm.

Determine the action taken on the inventory

The algorithm can determine two types of actions: the first consists of adding items to the inventory and the second action removes the items from the sensor's field of view.

To determine these actions, the objects are counted at two points, when the hand is detected and when the hand is no longer detected. When the hand is not detected, the algorithm compares the counts before and after the human intervention; if the count of objects after the human intervention is higher, the action of adding objects to the inventory is determined, otherwise it is determined that objects were taken.

The expected response should be as follows, see Figure 6:



Classification:

2 spoons.
3 forks.

Action performed:
Take objects

Figure 6 On the left, organization of the inventory elements and their respective count when the hand is no longer detected by the sensor; on the right, the message of the algorithm indicating the classification and count of the objects and the action performed in the inventory. In this case: Take objects.

Results

In the development of this work, different models and algorithms were used to determine the ideal model in terms of detection, robustness, and reliability, obtaining the following results:

Hand detection metrics

For hand detection and tracking we used the EgoHands dataset, a dataset specialized in hands, labeled, and created by Sven B, Stefan L, David C and Chen Y of Indiana University (9), it was decided to test the hand recognition algorithm. Modifications were made to the dataset to change the features and categories of the images since from these it is possible to discard whether it is a left hand or a right hand or other features that this model does not need for its operation. These data give us information regarding the performance of the model in terms of its ability to correctly identify positive and negative samples.

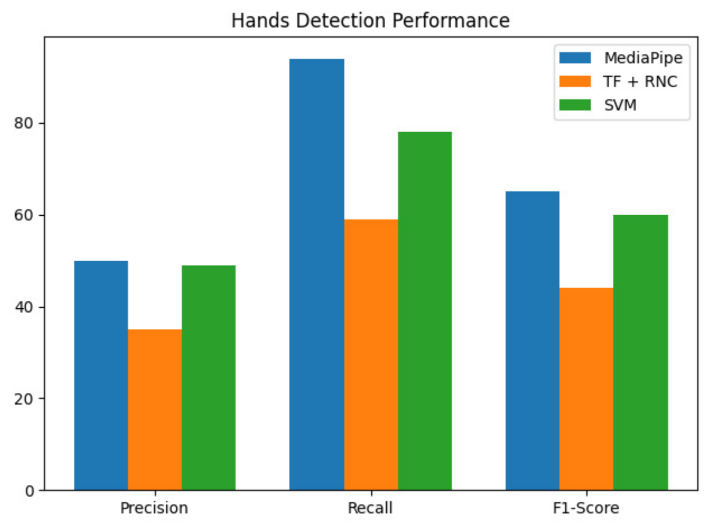


Figure 7 Hand detection performance evaluation graph, using different algorithms: MediaPiPe, Tensorflow combined with Convolutional Neural Networks and Support Vector Machines.

In general, higher accuracy and recall indicate better model performance, while a higher F1-score indicates a trade-off between accuracy and recall. Therefore, it is important to consider both accuracy and recall obtaining a complete assessment of model performance.

Object detection metrics

Since we could not find a specific dataset to test the performance in the detection of the objects of interest, we decided to generate one that would fit the needs of our use case.

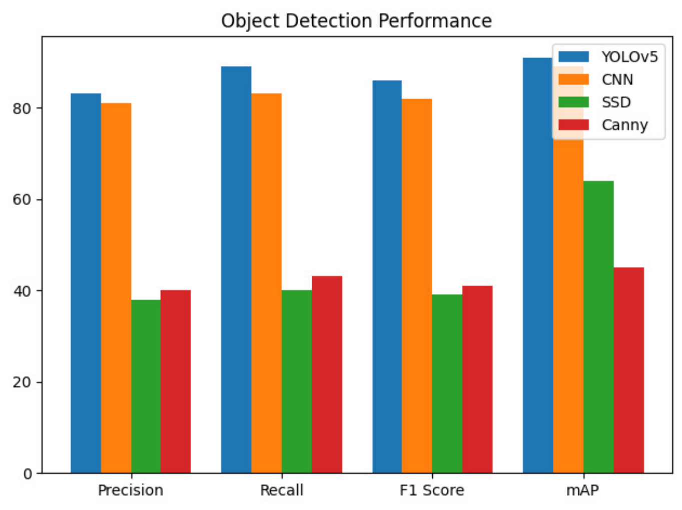


Figure 8 Object detection performance evaluation graph, using different algorithms: YOLOv5, CNN, SSD and canny.

Results analysis

Three models were used for hand detection and tracking: MediaPipe, TensorFlow and SVM. Table 1 shows the performance of the models with the EgoHand dataset. These metrics reveal that the best model for hand detection is MediaPipe, although the accuracy was very low in all models, and in this case the one with the lowest performance was TensorFlow combined with Convolutional Neural Networks.

Model	Precision	Recall	F1 Score
	50%	94%	65%
TF + RNC	35%	59%	44%
SVM	49%	78%	60%

Table 1 Performance comparison between MediaPipe, TensorFlow, SVM for hand detection with the EgoHands dataset.

Based on these results, it has been determined that MediaPipe has the best performance in hand detection and tracking.

1. Accuracy: The accuracy of 0.509 means that approximately half of the model's positive detections are correct, while the other half are false alarms.
2. Recall: The recall of 0.945 means that the model is very good at identifying the hands in the images, as it detects 94.5% of the hands present in the images.
3. F1-Score: The F1-score of 0.65 shows a balance between accuracy and recall. It may indicate that the model is doing a good job of maintaining a high recall while maintaining an acceptable level of accuracy.

On the other hand, for object count detection, tests were performed with 4 models: YOLOv5, convolutional neural networks and SSD MobileNet V2. The following results were obtained:

Model	Precision	Recall	F1 Score	mAP
YOLOv5	83%	89%	86%	91%
CNN	81%	83%	82%	89%
SSD	38%	40%	39%	64%
Canny	40%	43%	41%	45%

Table 2 Performance comparison between YOLOv5, CNN, SSD and Canny for object detection and counting trained with a proprietary dataset.

For object detection it was decided to opt for YOLOv5 even though the metrics were very similar with CNN, and it had a higher F1 Score, in general terms YOLOv5 had higher metrics and the F1 Score of this model indicated a very good performance.

Model comparison

Hand detection vs. object detection is a very high change from each other in terms of accuracy and F1 Score (approximately 30% difference), this is because EgoHands being a more specialized dataset, with a longer trajectory, presented a challenge for the models when looking for hands in unusual places or difficult to locate or hand size, This dataset consists of 3841 training images, from different places and situations; on the other hand, the training for object detection is presented as a dataset built according to the needs, presenting a base of 322 images, in easier situations that did not represent a challenge for the model.

Experimental results

In the context of this project, we sought to implement an algorithm that could recognize objects of interest in a live video sequence that would count the objects it detected. In order to test the algorithm, a set of 10 videos was created and metrics were taken with the pre-recorded videos, obtaining the following results, see Table 3:

Note: In Table 3, a False Positive (FP) or False Negative (FN) is taken as 0, A FP is determined as a successful count amount but labeled as different classes, and a FN is determined as a case where neither the count nor the class match; and cases marked as 1 are counted as True Positive (TP), a TP is counted as a successful count and classification of items; finally a dash (-) indicates that no more test were performed to the model in that video. "Tests" are defined as each hand-in and hand-out detection, in which the model performed the counting of items in the inventory, all videos had different "tests" with video 8 having the most with 11 inventory interactions and video 10 having the least with one interaction.

Table 3 Results of the 10 experimental test videos.

Video	**Test 1**										
Video 1	1	1	1	1	-	-	-	-	-	-	-
Video 2	0	0	1	0	-	-	-	-	-	-	-
Video 3	0	1	1	0	1	-	-	-	-	-	-
Video 4	1	0	0	1	0	0	-	-	-	-	-
Video 5	1	1	1	0	1	-	-	-	-	-	-
Video 6	0	0	0	1	0	0	0	1	-	-	-
Video 7	1	0	0	0	1	0	-	-	-	-	-
Video 8	1	1	1	1	1	1	1	1	1	0	0
Video 9	0	0	0	-	-	-	-	-	-	-	-
Video 10	0	-	-	-	-	-	-	-	-	-	-

Algorithm states

According to the tests performed, it was determined that the algorithm has three possible states as response, these states are successful case in which the classification, count and action performed are correct; unsuccessful case with misclassification, in which the count and action performed are correct but the classification is wrong; and finally, unsuccessful case with misclassification and count, in which the classification, count and action performed are not correct.

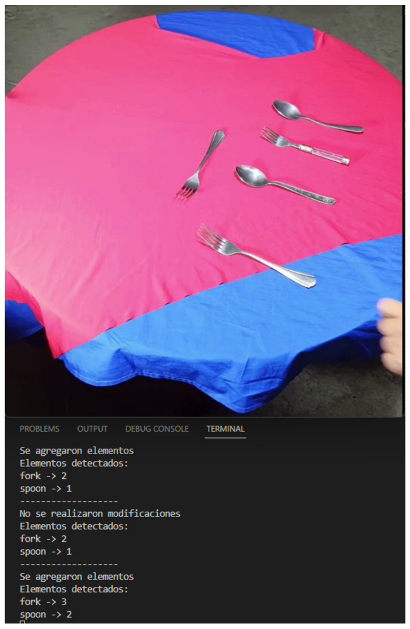
Successful case

A successful case is defined when a count and classification of objects is performed correctly, as shown in Figure 9a taken from video 1, we notice that the algorithm after not detecting the hand performs a correct classification, count and action since in the previous state of the inventory there were 2 spoons and a fork, after the human intervention 3 forks and two spoons were detected according to the program output, signaling the action as adding items.

Unsuccessful case, misclassification, and miscount

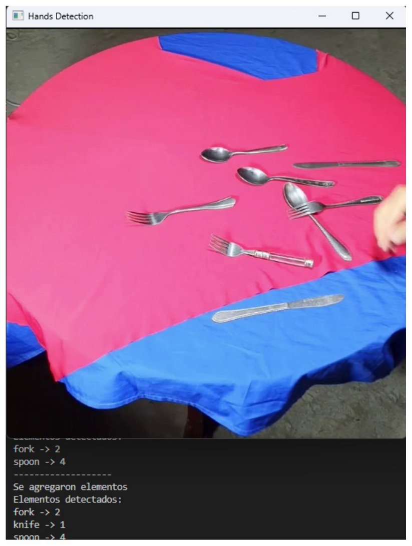
The second erroneous state can be seen in Figure 9b taken from video 9, in which neither the count nor the classification match what is observed in the image. In this video no classification was successful because it was decided to test the algorithm in usual conditions such as overlapping objects or not very common locations of the objects, this inventory configuration negatively impacted the accuracy of the algorithm in the detection and classification of objects.

a)



```
Se agregaron elementos
Elementos detectados:
fork -> 2
spoon -> 1
-------------------
No se realizaron modificaciones
Elementos detectados:
fork -> 2
spoon -> 1
-------------------
Se agregaron elementos
Elementos detectados:
fork -> 3
spoon -> 2
```

b)



```
fork -> 2
spoon -> 4
-------------------
Se agregaron elementos
Elementos detectados:
fork -> 2
knife -> 1
spoon -> 4
```

Figure 9 possible outcomes determined by the algorithm, a) correct outcome in which both object sorting and counting is correct; b) incorrect outcome in which neither object counting nor sorting is correct.

Discussion

Object detection using VC has been a well-studied field and different models and algorithms have been proposed to achieve this goal, some examples are the model proposed by Narendra and Sinisa (10) which is based on detecting elements through the segmentation of primitive objects; or the algorithm proposed by Fidler which focuses its operation on the use of the Gabor filter (14). The work done by Pedro Felzenszwalb (13) describes an object detection system based on the combination of models of deformable parts at different scales. This system can represent highly variable object classes and achieves very good results in PASCAL object detection with approximately 91% accuracy.

Loïc develops a new object detection algorithm called DF-SSD (12), which results in an improvement of the Single Shot MultiBox Detector (SSD). DF-SSD uses Dense Convolutional Network (DenseNet) and feature fusion to address the lack of complementarity between SSD feature layers and its weak detection capability for small objects.

Hand detection and recognition has been mainly focused on the detection of distress signals (16) or as an aid in sign language (11), although according to Nathasia, in the field of machine learning, hand recognition is a difficult challenge to solve. There are different approaches to solve the problem of making VC recognize sign language gestures. In 2021, Aripita Halder, made a methodology that simplified sign language in real time using MediaPipe and SVM, obtaining an average accuracy of 99%, the proposed model is accurate, efficient and can help the communication process (15).

The referenced works have been used as a starting point and aid for the development of this research. In this work, a hand recognition and object classification system is designed and developed as a permanent inventory control measure, which is developed using MediaPipe and YOLOv5.

The algorithm is presented as an aid in inventory control and security because with it you can have greater control of inventory management and know when an item was added or removed at the time of human interaction. The presented algorithm apart from counting the objects can detect the hands that modify the inventory and detecting if I add or remove objects from the inventory, adding an extra to the control of objects in the inventory. Having a dataset built which was used for training and testing the YOLOv5 model, and a dataset specialized in hands for testing the Mediapipe model, the algorithm in a real environment obtained an accuracy of 43.7% in the recognition of objects and 96% in the recognition of hands, these results show that it can effectively perform a control of objects and detect when a human actor interferes in the inventory, and thus achieve to define the action taken.

To improve the accuracy of the algorithm in object recognition by means of the dataset, Kalahiki proposes a constant training approach and thus increase the accuracy of the algorithm in object recognition by preparing it to recognize objects in different positions, light settings, or places where the original dataset would not have been contemplated. Making these changes implies an extra work in the training, but, at the same time, it implies an increasingly more efficient model in terms of recognition according to the results presented by Kalahiki (see Table 4). The accuracy increases each week as new images taken from the same videos that have difficulty to be recognized are trained. In Kalahiki's work, it starts with an accuracy of 61.92% for the linear activation function and in week 3 it increases by 3.68% leaving the accuracy at 65.6%, this model turns out to give a good alternative because as it is trained with more images its algorithm will become more robust.

Table 4 Average accuracy in the generalization data set by activation function. Data taken from (2)

Activation function	Initial average	Average in week 1	Average in week 2	Average in week 3
Lineal	61.92%	62.4%	63.83%	65.6%
ReLU	62.49%	62.89%	64.24%	66.39%
Leaky ReLU	63.16%	62.35%	63.58%	66.59%
SELU	62.62%	62.7%	63.75%	65.77%

On the other hand, this work proposes to focus on the classification and counting of objects being activated when a hand passes through the field of action of the sensor, and determine the action performed, this algorithm had an accuracy of 43.7% in object recognition and 96% in hand recognition, with which the same treatment proposed by Kalahiki could be performed and make the algorithm more robust and prepared to different states not contemplated in the original dataset.

Arias proposes the use of CV using support vector machines (SVM) for a more agile recognition of potentially dangerous objects in an airport context, identifying handguns. Arias obtained a performance increase of 93.36% with respect to a previous implementation using parallel programming, as well as a satisfactory classification of 93.02% in images containing firearms. The algorithm proposed in this work executes the hand and object detection algorithms in parallel, but only makes use of YOLOv5 when the hand is detected, thus avoiding a constant use of object detection and reducing the computational cost of analyzing the frames of each video, this means an improvement in the performance of the algorithm, presenting the following results (see Figure 10).

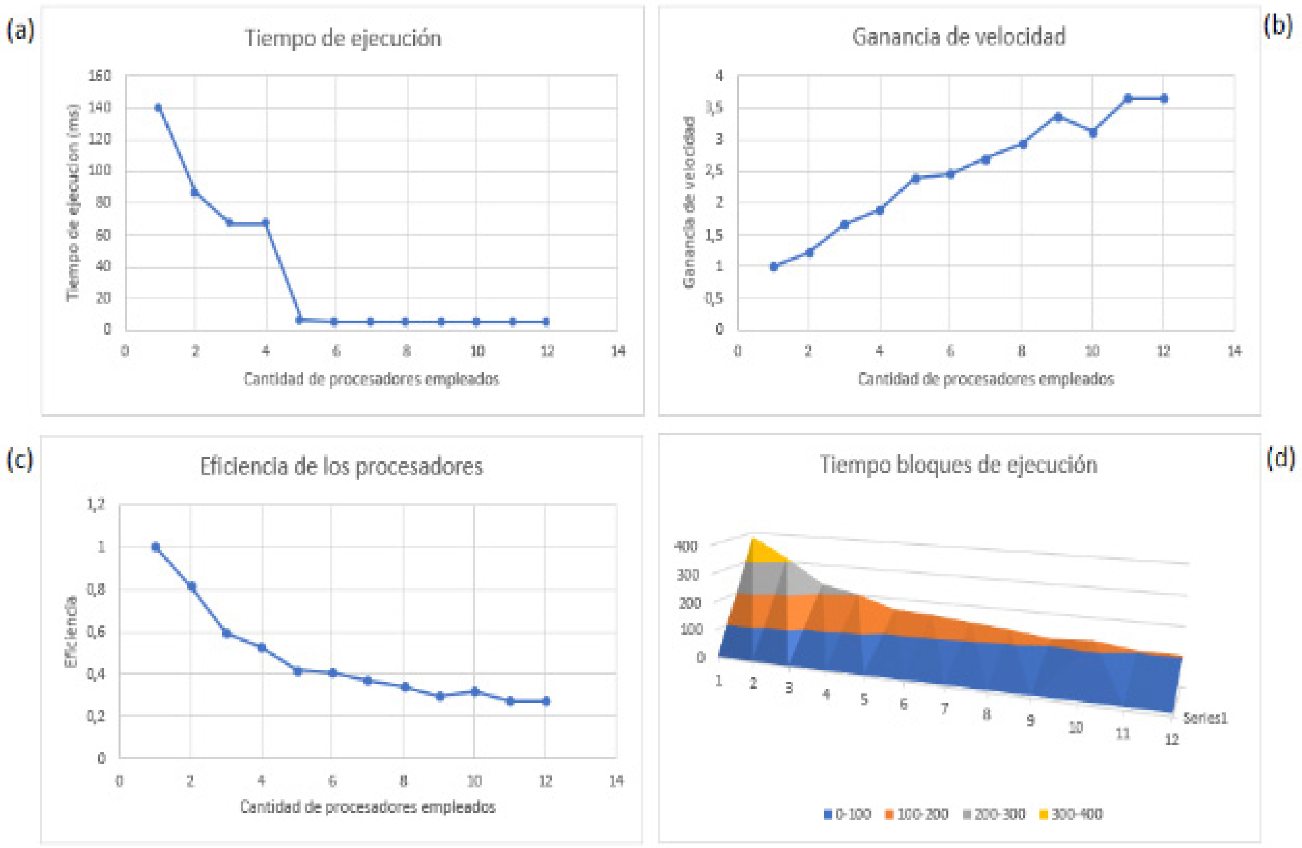


Figure 10. Computational performance graph of the detection model with MediaPipe and YOLOv5; a) code execution time in ms, with different processors in use; b) speed gain based on the processors used; c) algorithm efficiency with different processors in use; d) code block execution time in ms.

With a data set and focused training, this algorithm could serve as a model for the recognition and counting of objects other than those proposed in this article. It presents itself as a good opportunity for inventory control in stores, warehouses, or other places where inventory control is required.

The lack of images to perform the training and the proper integration with the YOLOv5 tool is one of the main causes for which the counting and recognition of cutlery was low, with an adequate data set the accuracy metrics would improve, making the algorithm more useful, being able to overcome the challenges that in its current version, is not able to solve, such as self-occlusion, overlapping objects, or objects partially detected by the field of action of the sensor, for that reason it is expected to have a better prepared dataset and thus improve the recognition of objects.

Conclusions

For hand recognition it is necessary to properly configure MediaPipe, this means that the thresholds must be leveled, a very high one will mean a more thorough detection by the algorithm, which leads to hand recognition under certain very specific conditions, the opposite case means that the algorithm will be more permissive in recognizing hands and any object in the field of view of the sensor could be recognized as such.

As with MediaPipe, YOLOv5 must be configured correctly in terms of threshold classification and detection confidence, this allows for greater control when discarding or recognizing an object within the inventory and recognizing it correctly.

The counting of the objects falls fundamentally on the classification model, a bad classification means a bad counting of objects which in turn indicates an erroneous action by the user, a more exhaustive training must be carried out so that the algorithm can recognize and classify the objects, and this will automatically mean a better counting of the inventory elements.

At the time of obtaining the MediaPipe metrics, the set of test images showed a low score in terms of accuracy, recall and F1, but, at the time of experimentation, the metrics showed a high percentage in all aspects, the opposite case occurred with the MediaPipe metrics, this is due to the quality of the test data versus the experimentation data.

The experimentation also showed a weak point in terms of recognition on surfaces with a color like the objects of interest, the silver-colored cutlery, in the experimentation it was found that with a white surface YOLO did not recognize the objects of interest, therefore, the decision was made to have a surface that contrasted the objects of interest.

References

1. Athanasios V, Nikolas D, Anastasios D, Eftychios P. Deep Learning for Computer Vision: A Brief Review. Computational Intelligence and Neuroscience [Internet]. 2018 feb [quoted 7 aug 2023]. Available on: https://doi.org/10.1155/2018/7068349.
2. Christopher Bradley K. Computer Vision for Inventory Management [master's thesis on the Internet]. Lousiana Tech University; 2020 [quoted 7 aug 2023]. Available on: https://digitalcommons.latech.edu/theses/40/.
3. Felipe R, Leaned Q, Frank S, David C, Enrique M. Software component for weapons recognition in X-ray images [Internet]. 2017 apr [quoted 7 aug 2023]. Available on: https://www.researchgate.net/publication/316628078_Software_component_for_weapons_recognition_in_X-ray_images.
4. MediaPipe team. MediaPipe Framework [Internet]. [Unknown location] [Quoted 8 aug 2023]. Available on: https://developers.google.com/mediapipe/framework.
5. Fan Z, Valentin B, Andrey V, Andrei T, George S, Chuo-Ling C, Matthias G. MediaPipe Hands: On-device Real-time Hand Tracking. 2020 jun [quoted 8 aug 2023]. Available on: https://arxiv.org/abs/2006.10214.
6. Joseph R, Santosh D, Ross G, Ali F. You Only Look Once: Unified, Real-Time Object Detection. 2015 jun [quoted 8 aug]. Available on: https://arxiv.org/abs/1506.02640.
7. Christian S, Wei L, Yangqing J, Pierre S, Scott R, Dragomir A, Dumitru E, Vincent V, Andrew R. Going Deeper with Convolutions. 2014 sep [quoted 8 aug 2023]. Available on: https://arxiv.org/abs/1409.4842.

8. Glenn J. why do I need to train from the pt model you have trained? · Issue #2990 · ultralytics/yolov5 · GitHub [Internet]. [Unknown location] [Quoted 8 aug 2023]. Available on: https://github.com/ultralytics/yolov5/issues/2990.
9. Bambach, S., Lee, S., Crandall, D., and Yu, C. EgoHands Object Detection Dataset [Internet]. [Unknown location] [Quoted 8 aug 2023]. Available on: https://public.roboflow.com/object-detection/hands.
10. Narendra A., Sinisa T. Learning the Taxonomy and Models of Categories Present in Arbitrary Images. 2007 dec [Citado 8 aug 2023]. Available on: https://ieeexplore.ieee.org/document/4409039.
11. Ming Jin C., Zaid O., Mohamed H. A review of hand gesture and sign language recognition technique. 2017 aug [Quoted 8 aug 2023]. Available on: https://link.springer.com/article/10.1007/s13042-017-0705-5.
12. Loïc C., Benoît M., Philippe T. Object Detection with Spiking Neural Networks on Automotive Event Data. 2022 may [Quoted 8 aug 2023]. Available on: https://arxiv.org/abs/2205.04339.
13. Pedro F., Ross B., David M., Deva R. Object Detection with Discriminatively Trained Part-Based Models. 2010 sep [Quoted 8 aug 2023]. Available on: https://ieeexplore.ieee.org/document/5255236.
14. Sanja F., Ales L. Towards Scalable Representations of Object Categories: Learning a Hierarchy of Parts. 2007 jul [Quoted 8 aug 2023]. Available on: https://ieeexplore.ieee.org/document/4270294.
15. Arpita H., Akshit T. Real-time Vernacular Sign Language Recognition using MediaPipe and Machine Learning. 2021 may [Quoted 8 aug 2023]. Available on: https://www.researchgate.net/publication/369945035_Real-time_Vernacular_Sign_Language_Recognition_using_MediaPipe_and_Machine_Learning.
16. Nathasia F., Michael V., Seto B., Abdul H. Hand Gesture Recognition as Signal for Help using Deep Neural Network. 2022 feb [Quoted 8 aug 2022]. Available on: https://www.researchgate.net/publication/358377113_Hand_Gesture_Recognition_as_Signal_for_Help_using_Deep_Neural_Network.