








## Article

# Vision acuity index. A novel metric to select the best fit computer vision algorithm for smart cities and Industry 4.0

Roberto Contreras-Masse <sup>1,2,\*†‡</sup> , Alberto Ochoa-Zezzatti <sup>1‡</sup> , Vianey Torres-Argüelles <sup>1‡</sup> , Vicente García <sup>1‡</sup> , Mayra Elizondo-Cortés <sup>3‡</sup> , Luis Pérez-Dominguez <sup>1‡</sup> , and José Mejía <sup>1‡</sup> 

<sup>1</sup> Universidad Autonoma de Ciudad Juárez (UACJ), Ciudad Juárez, México 32310; alberto.ochoa@uacj.mx (A.O.-Z.); vianey.torres@uacj.mx (V.T.-A.); vicente.jimenez@uacj.mx (V.G.); (luis.dominguez@uacj.mx (L.P.-D.); jose.mejia@uacj.mx (J.M.)

<sup>2</sup> Instituto Tecnológico de Ciudad Juárez (ITCJ), Ciudad Juárez, México 32500

<sup>3</sup> Departamento de Ingeniería en Sistemas DIMEI-FI, Universidad Nacional Autonoma de Mexico (UNAM), Ciudad de México, México 04510; mayra.elizondo@comunidad.unam.mx

\* Correspondence: rcontreras@itcj.edu.mx

† Current address: Av. Tecnológico No. 1340 Fracc. El Crucero C.P. 32500 Ciudad Juárez, Chih, México.

‡ These authors contributed equally to this work.

**Abstract:** Computer vision is considered as an ally to solve business problems that require human intervention, intelligence and criteria. This topic of research has evolved in XXI century at faster pace, delivering various alternatives from open source until commercial platforms. With so many options and market growing, it result difficult to make a decision on which one to use, or even worse, realize it was not suited for different scenarios. In this paper we analyze five options selected arbitrarily and tested on a dataset of 755 images to detect persons in an image, using object detectors. We analyze elapsed time to process an image, error with observations by humans, number of persons detected, correlation of time and person density, object detected size and F1 Score, considering precision and recall. As we found there are score ties and similar behaviors among options available, we introduce a novel index that takes in consideration the number of persons and their pixel size, to propose the Vision Acuity Index of Computer Vision. The results demonstrate this is a good option to serve as indicator to make decisions. Also, this index proposed have a potential to be expanded for different business use cases, and to measure new proposed algorithms in the future along with the traditional metrics used previously.

**Keywords:** Computer vision; performance metrics; Yolo3, AWS Rekognition, Azure Computer Vision

## 1. Introduction

Computer vision is a great tool that performs identification of objects. Those objects can be telephones, chairs, tables, food, and even persons. Other uses are for face recognition, obtain face landmarks to identify if the person is wearing a beard or glasses, or if the person is smiling or frowning. The algorithms have evolved a such point where authorities of national security have relied on this technology to identify people at immigration gates at airports, or allow passengers to board a plane just by showing their face. Among other uses is the object counting, useful for manufacturing, package tracking and measurement for parcels, find defects on production lines, or find a parking lot space in a busy mall. As all this examples show, the business use of computer vision is very broad. Researchers and enterprises are offering options to solve the most challenging use cases, but they often tend to claim

their solution, algorithm or Application Programming Interface (API) is the best option. Companies are facing now this new challenge, how to select or rank the considered options for their business problem, as there are algorithms which performance is very close to each other, and the current metrics such as precision, recall, accuracy and F1 Score are the same for some options. This makes difficult for enterprises and academics to choose a specific option, or what could be the best option for their use case. In this paper we propose a new metric named Vision Acuity Index (VAI) for computer vision. This index takes in consideration the size and density of objects along with the current performance metric used broadly and well accepted like F1 Score. To test our proposed index, we use person detection, therefore, we created a dataset of 755 pictures randomly acquired from Flickr.com containing persons (from none to crowds) to be tested against five existing alternatives: i) Yolo3, ii) Yolo3Tiny, iii) AWS Rekognition (API), iv) Azure Computer Vision (API), and v) HOGCV. We ran each image through each of the five alternatives or options, and record elapsed time, levels of confidence, returned objects, and size of each object. Each picture processed is converted to grayscale and draws green rectangles on the picture for surrounding the person detected and adds a red dot to make easier for humans to see the persons found. Also, manual observation is recorded to track true positives, true negatives, false positives and false negatives to calculate F1 Score.

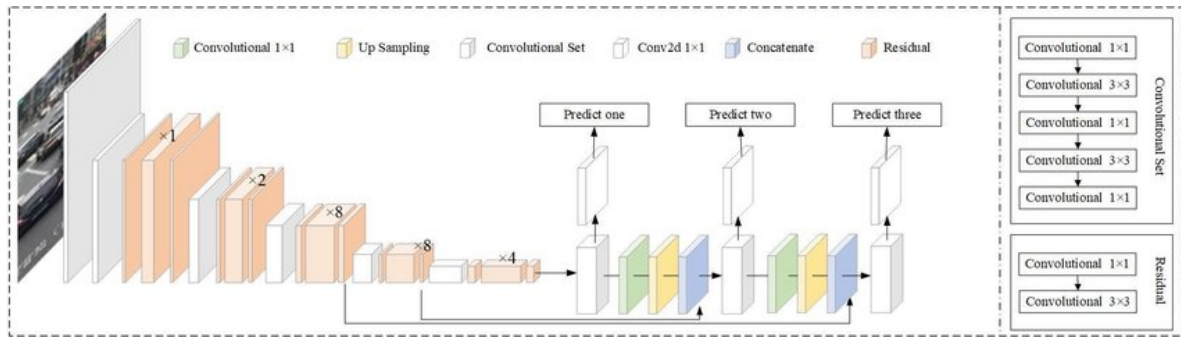
The paper is organized by sections. Section 2 do a review of related work. Section 3 describes the methodology and introduces VAI calculation. Section 4 shows the results of deep analysis performed on the metrics recorded and calculated along with VAI calculation and discuss why this index helps to differentiate and select among computer vision alternatives. Finally, section 5 presents conclusions and the future research.

## 2. Related work

There are different alternatives evolved during time and used broadly in the industry and research. As a brief history summary of milestones, it can be listed Viola-Jones Detector (2001), followed of Histogram of Oriented Gradients (HOG) (2006), and Deformable Part-Based Model (2008). Then, AlexNet is the milestone of convolutional neural network (CNN) usage for object detection (2012). From that point two branches emerge Two-stage detectors (from 2014 to 2017), which have outstanding examples like Recursive-CNN, Spatial Pyramid Pooling Network, Fast Region CNN, Faster Region CNN, Mask Region CNN and Feature Pyramid Networks. Almost in parallel, in 2016 one-stage detectors arrived with You Only Look Once (YOLO), Single Shot Multi-box detector (SSD), RetinaNet, SqueezeDet and CornerNet [1]. from this list, we have an interest on YOLO and HOG to be part of our experiment as they have implementations available in python that are popular and have been benchmarked with Microsoft Common Objects in Context (MS COCO) trained and test datasets.

You Only Look Once (YOLO) algorithm was developed by Redmon, J. et al. in 2016. The processing of images is simple, inspired by human eye mechanism, where the eye glance and image and know what objects are seen, characteristics and interpret context. YOLO resizes the image, execute a single layer convolutional network and use thresholds of confidence to return if an object is detected. The proposed algorithm divide image into a square grid, and each grid cell is responsible to detect objects, predicting a bounding box and confidence score [2]. YOLO next version named YOLOv2 introduced a 19-layer network (Darknet-19) improving in speed but losing accuracy [3], and YOLO3 improved accuracy by sacrificing speed by adding a 106 layer of convolutional networks called Darknet-53, shown in Figure 1. The main metric for accuracy on these models have been mean average precision (mAP) [4]. Tiny YOLOv3 was developed as a simplified version of YOLOv3, it uses seven layers (instead of Darknet-53) and the features are identified by small layers of 1X1 and 3X3, but at the end the loss function is the same as YOLOv3 [5].

Histogram of oriented gradients (HOG) has been used since 2006 to detect objects and persons and it is based on magnitude and orientation [7] with horizontal and vertical gradients computed to later calculate magnitude and angular orientation, dividing the image into cells and grouping the cells in blocks. Blocks are overlapped to find orientations for the same cell to form histogram bins sorted



**Figure 1.** YOLOv3 Architecture of Darknet-53. Source image: [6]

and combined into a final total histogram. Developed by Dalal and Triggs [8] summarized in equation 1, this algorithm has been applied to images and later on in films and videos. In Python, there is an implementation in OpenCV library that uses HOG (HOGCV) as a built-in library, originally developed in C++ and now available for python<sup>1</sup>.

$$T_h = B_i * B_\delta * N_B \quad (1)$$

where  $T_h$  is the total number of features found by HOG descriptor,  $B_i$  are the blocks of the image,  $B_\delta$  are block sizes and  $N_B$  are the number of bins.

For API available there are commercial alternatives from leaders of cloud service providers (CSP) reported on Gartner's magic quadrants (2020). Those are Amazon Web Services, Microsoft Azure and Google Cloud Platform (GCP). The APIs available are AWS Rekognition<sup>2</sup>, Microsoft Azure computer Vision<sup>3</sup>, and GCP Image analysis<sup>4</sup>. There are others alternatives available like IBM Watson Vision<sup>5</sup> and Cloud Sight<sup>6</sup>. Each API has a different return information, but similar for object detection. Usually, the APIs return object probability, type, and coordinates based on the image input. The variables and values are different. For instance, AWS Rekognition returns the proportional value of coordinates  $x, y$  and width and height  $w, h$  as a ratio within the range  $[0,1]$ , and confidence in percentage in range  $[0,100]$ . Azure returns coordinates as absolute position and as well as width and height, however the confidence is a percentage in the range  $[0,1]$ .

In the reviewed literature for measuring accuracy or performance it has been a constant to use Average Precision (AP) [9] and mean Average Precision (mAP) [3–5,10]. AP finds the area under the precision recall curve from 0 to 1 expressed as:

$$AP = \int_0^1 p(r)dr \quad (2)$$

Another performance evaluation is Receiver Operating Characteristic (ROC) and the Area Under the Average ROC (AUC). The input indicators are recall and false positives, and mean accuracy (mA) was also used for object detection algorithms [11]. Equation 3 takes in consideration Positives  $P_i$  and Negatives  $N_i$  and  $\lambda$  as the number of attributes.

$$mA = \frac{1}{2N} \sum_{i=1}^{\lambda} \left( \frac{TP_i}{P_i} + \frac{TN_i}{N_i} \right) \quad (3)$$

<sup>1</sup> [https://docs.opencv.org/3.4/d5/d33/structcv\\_1\\_1HOGDescriptor.html](https://docs.opencv.org/3.4/d5/d33/structcv_1_1HOGDescriptor.html)

<sup>2</sup> <https://aws.amazon.com/rekognition>

<sup>3</sup> <https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/>

<sup>4</sup> <https://cloud.google.com/vision>

<sup>5</sup> <https://www.ibm.com/es-es/cloud/watson-visual-recognition>

<sup>6</sup> <https://cloudsight.ai/>

An experimental approach proposed by [12] measured the performance using accuracy using as inputs number of misses and number of misdetections. Another relation measured is the Accuracy/Speed trade-off [9] when studying focal loss for imbalance between foreground and background when doing object detection.

The approach proposed by Xin Gao [13] includes concepts of Precision, Recall, and F-Score, and Percentage of Wrong Classification (PWC), in addition to frame-oriented metrics such as multiple object detection accuracy (MODA) and multiple object count (MOC). F1-Score is widely reported in literature and it is a well accepted in statistics.

In the literature review there is no mention to the concept of acuity applied to object detection. This opens the opportunity to propose VAI supported on previous measurements.

### 3. Methodology

As literature and commercial vendors report, there are a good variety of options of computer vision algorithms to detect objects, in this case, persons within an image. Some of these options can be found available as open source or as a library, with trained and optimized models ready to be imported and used directly on the images or video. Others are available as application programming interface (API) receiving upload image and return metadata in Java Simple Object Notation (JSON) format. No matter what path is chosen, each option has specific performance related to number of correct objects labeled, confidence percentage and speed of processing.

As it is proposed at the beginning of this paper, in a public place there will be persons walking by at different distances, and therefore, there will be smaller objects that computer vision may or may not have difficulty to detect. In a human analogy, it could be expected a smaller person would have more difficulty to be identified opposed to a larger person, which could be easily identified. This is a human medical and optical concept known as vision acuity. As computer vision or algorithms usually are measured by accuracy, precision, recall or F1-Score or F2-Score, but not by how small the object can be detected, it is important to define the vision acuity index (VAI) of computer vision (Y) algorithms (or commercial APIs) by:

$$Y_j = \log\left(\sum_{i=1}^n \omega_i S_i^{-1}\right) F1_j \quad (4)$$

Where  $\omega$  is the confidence (probability) of object detected in the range [0,1],  $S$  represents the proportional surface of object detected in relation to the whole image surface, as defined in 5, and  $F1_j$  is the F1 Score of the whole image.

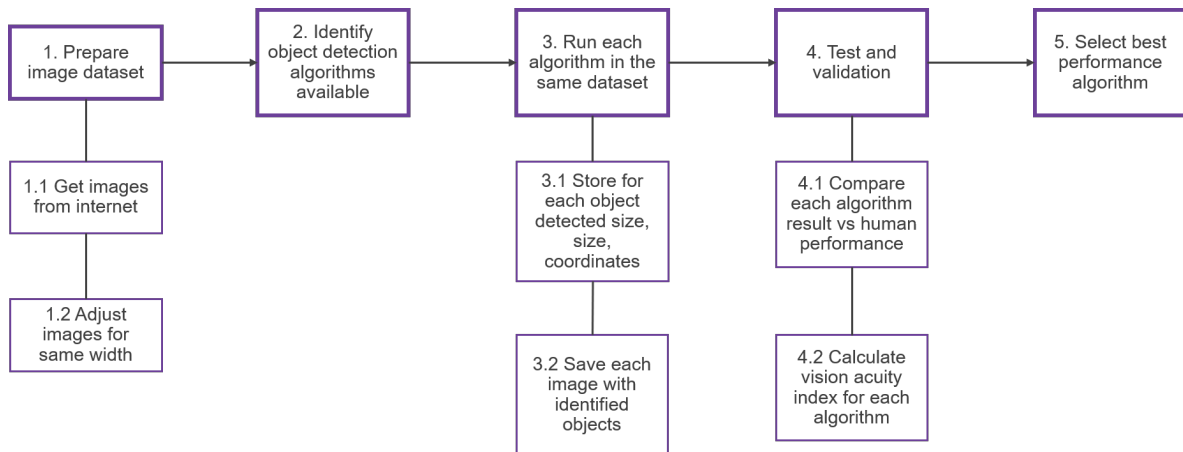
$$S = \frac{wh}{WH} \quad (5)$$

Where  $S$  is the proportional surface of object,  $w, h$  are object's width and height in pixels returned by algorithm, and  $W, H$  are the image's width and height in pixels.

The methodology to find this index is described in Figure 2 with five steps detailed in the following sub sections.

#### 3.1. Prepare image dataset

Image data can be troublesome to build. It involves searching and downloading until complete the appropriate number of samples. One way to achieve this is leverage automation and create a script to search and download the pictures from Flickr.com. Once an account is created, an API key was created to access the search engine directly from python scripts. Our python script consist of two steps:



**Figure 2.** Methodology to select the best vision acuity index of considered computer vision algorithms

1. Get URL list using from Flickr.com using the tag to search the site. In this case it can be used "person", "tourist", "pedestrian". It is recommended to do a quick search directly in Flickr to test the search tag (script 1).
2. Take each URL to download and save the image for future processing. This is done with script 2

With this dataset complete, the last thing to do is to standardize the width of each picture so the algorithms can be tested in similar conditions. In our case it was set to 600 pixels width. This is required also as Flickr is used to upload people pictures coming in different sizes and resolutions, resulting in pictures above 5,000 pixels wide. Those large sizes are not efficient for computer vision applications, and could lead to very slow performances. In addition, applications using video coming from security cameras or Industrial Internet of Things (IIoT) cameras usually consume standard definition (640 X 480 pixels, or 480p) or high definition (1280 X 720 pixels, or 720p). Script 3 specify 600 pixels resize to every picture downloaded and stored locally.

### 3.2. Identify object detection algorithms available

From literature review and related work found, there are around 15 methods for object detection developed in this century. Three methods with person detection capacity available implemented in python or with pre-trained models were selected: Yolo3, YoloTiny and HOGCV. There are also commercial offers from the three leaders of cloud computing providers providing APIs for computer vision with object detection capacity to identify a person. From those, we selected Azure Computer Vision API and AWS Rekognition API.

### 3.3. Run algorithms in the same dataset

Each algorithm will scan each image from resized dataset. To track the outcome and be able to compare and correlate data, images have a naming convention of `img_nnnn.jpg`, where `nnnn` is the consecutive number starting in 0000. For each algorithm a script to process the dataset is required, as each method returns different metadata of found objects. To standardize the outcomes a simple data structure and guiding principles are defined as follows:

- Results of scanned pictures have: a) picture id, that corresponds to image filename; b) image width and height (in pixels); c) number of persons detected; d) elapsed time to process the image (seconds).
- Another report contains the metadata of each object detected stored as: a) picture id, that corresponds to image filename; b) object detected sequential number; c) object dimension in pixels, width and height; d) confidence of being a person, recorded in range [0,1].
- Resized images (600px width x proportional scale height) are stored in color.

---

```

# -----
# Script: Get images URLs by tag from Flickr.com
# By: Roberto Contreras-Masse
# -----

from flickrapi import FlickrAPI
import pandas as pd
import time
import sys

key = #YOUR KEY HERE
secret = #YOUR SECRET HERE

PATH = #YOUR PATH TO SAVE FILES HERE

def get_urls(image_tag, MAX_COUNT):
    flickr = FlickrAPI(key, secret)
    photos = flickr.walk(text=image_tag,
                        tag_mode='all',
                        tags=image_tag,
                        extras='url_o',
                        per_page=50,
                        sort='relevance')

    count=0
    urls=[]
    for photo in photos:
        if count< MAX_COUNT:
            count=count+1
            print("Fetching url for image number {}".format(count))
            try:
                url=photo.get('url_o')
                urls.append(url)
                time.sleep(1)
            except:
                print("Url for image number {} could not be fetched".format(count))
        else:
            print("Done fetching urls, fetched {} urls out of {}".format(len(urls),MAX_COUNT))
            break

    urls=pd.Series(urls)
    print("Writing out the urls in the current directory")
    urls.to_csv(PATH + image_tag+"_urls.csv")
    print("Done!!!")

tag = #YOUR TAG HERE
MAX_COUNT=50
get_urls(tag,MAX_COUNT)

```

---

**Listing 1.** Script to get image matching URLs from Flickr.com

---

```

# -----
# Script: Download and save images
# By: Roberto Contreras-Masse
# -----

import csv
import requests
import os
import sys
import time

def put_images(FILE_NAME):
    urls = []
    with open(FILE_NAME, newline="") as csvfile:
        doc = csv.reader(csvfile, delimiter=",")
        for row in doc:
            if row[1].startswith("https"):
                urls.append(row[1])
    if not os.path.isdir(os.path.join(PATH, FILE_NAME.split("_")[0])):
        os.mkdir(FILE_NAME.split("_")[0])
    t0 = time.time()
    for url in enumerate(urls):
        print("Starting download {} of {}".format(url[0] + 1, len(urls)))
        try:
            resp = requests.get(url[1], stream=True)
            path_to_write = os.path.join(PATH, FILE_NAME.split("_")[0], url[1].split("/")[-1])
            print (path_to_write)
            outfile = open(path_to_write, 'wb')
            outfile.write(resp.content)
            outfile.close()
            print("Done downloading {} of {}".format(url[0] + 1, len(urls)))
        except:
            print("Failed to download url number {}".format(url[0]))
    t1 = time.time()
    print("Done with download, job took {} seconds".format(t1 - t0))

PATH = #YOUR PATH HERE
FILE_NAME = # FILE NAME WITH URLS
put_images(FILE_NAME)

```

---

Listing 2. Script to download and save images

---

```

# -----
# Script: Resize images to same width
# By: Roberto Contreras-Masse
# -----

from PIL import Image
import os, sys
import time

TAG = "manual"
PATH = "c:/Temp/datasets/" + TAG + "/"
PATH_RESIZED = "c:/Temp/datasets/" + TAG + "_resized/"
dirs = os.listdir(PATH)
RENAME_START = 861

WIDTH = 600

def rename(path, ext='.jpg', start=0, zeros=3):
    os.chdir(path)
    for i, filename in enumerate(os.listdir(path)):
        if filename.endswith(ext):
            print('[INFO] renaming ', filename)
            os.rename(filename, path + 'img_' + str(i + start).zfill(zeros) + ext)

t0 = time.time()

if not os.path.isdir(PATH_RESIZED):
    os.mkdir(PATH_RESIZED)

for item in dirs:
    print(PATH + item)
    if os.path.isfile(PATH + item):
        file, extension = os.path.splitext(PATH + item)
        im = Image.open(PATH + item)
        # Obtener las dimensiones de la imagen
        width, height = im.size
        # calcular la razón del aspecto y calcular la nueva altura
        aspectRatio = (width / height)
        newHeight = WIDTH / aspectRatio
        # Escalar la imagen
        imResize = im.resize((WIDTH, int(newHeight)), Image.ANTIALIAS)
        imResize.save(PATH_RESIZED + item, 'JPEG', quality=90)
        print('[INFO] saved: {} ({}x{})'.format(PATH_RESIZED + item, WIDTH, int(newHeight)))

rename(PATH_RESIZED, '.jpg', RENAME_START, 3)

t1 = time.time()
print("Done with download, job took {} seconds".format(t1 - t0))

```

---

Listing 3. Script to resize all pictures to 600 pixels width



(a) Original image resized

(b) Computer Vision result

**Figure 3.** Example of image processed with persons detected by algorithm and draw boxes on persons with red dot to identify each green rectangle center.

- Processed images are stored at independent location per algorithm, with same filename as the original images, but transformed in grayscale. Also, pictures will have a bright green rectangle and a red dot at object center to identify fast the person detected (see Figure 3).

### 3.4. Test and validation

In order to find the performance of methods of person detection and compare among them, the following steps are executed:

- Analyze elapsed times. By analyzing the elapsed time distribution, along with minimum, maximum, mean and standard deviation, algorithms' elapsed time or speed can be compared equally. Also, analysis of number of persons detected versus elapsed time can help to find the behavior of methods in crowded environments. Finally, it is important to find if algorithm describe a trend on elapsed time and persons detected.
- Set a ground truth by human inspection. This is achieved by recording the human inspection of each picture regarding how many persons they detect. In our experiment eight volunteer students from Instituto Tecnológico de Ciudad Juárez participated and recorded their findings. This information is good to calculate the mean person detected by humans (mPDH).
- Compare persons detected by each method against the mPDH to find in quick fashion the error in each image. Three values can be obtained when using equation 6: a) error is negative, meaning the computer algorithm under-detected persons; b) error is zero, meaning the computer algorithm was accurate; and c) error is positive, meaning the algorithm detected more persons than human eye.

$$\varepsilon_i = (CV_i - mPDH_i) \quad (6)$$

where equation 6  $\varepsilon$  is the measured error,  $i$  is the  $i$ -th picture, and  $CV$  is the number of persons detected by computer vision algorithm. Analyze the errors to identify anomalies and possible patterns on each algorithm/method.

- Finally, calculation of F1 Score for each image is required in order to apply equation 4 to obtain the VAI for that picture. F1 Score is done by equation 9.

$$P = \frac{TP}{TP + FP} \quad (7)$$

$$R = \frac{TP}{TP + FN} \quad (8)$$

$$F1Score = 2 \left( \frac{PR}{P + R} \right) \quad (9)$$

**Table 1.** Elapsed time (seconds) metric insights for all options

Option	Mean	Std Dev	Max	Min
Yolo3	1.763	0.0549	1.9957	1.6159
Yolo3Tiny	0.2086	0.0285	0.4023	0.1907
AWS Rekognition	0.563	0.2617	2.508	0.312
Azure	0.3846	0.5133	10.4542	0.2402
HOGCV	2.2289	0.9864	6.7746	0.4259

**Table 2.** Number of objects detected metric insights for all options

Option	Mean	Std Dev	Max	Min
Yolo3	4.6361	4.3742	28	0
Yolo3Tiny	1.9226	1.9646	12	0
AWS Rekognition	6.0684	5.6611	30	0
Azure	2.4981	2.1816	16	0
HOGCV	3.8077	3.0267	15	0

- Knowing the VAI for each image, the average can be calculated to determine the best suited algorithm for our application.

#### 4. Results and discussion

for this paper, it was required to identify the object "person" in different images. Using scripts 1, 2 and 3 the dataset consist of 775 images, of different heights but same width (600 pixels). All pictures were stored in the cloud for easy access later. The images have the naming convention of "img\_nnnn.jpg". The algorithms/models included in this study were Yolo3, Yolo3Tiny, HOGCV and two commercial APIs, AWS Rekognition and Azure Computer Vision. All object detection was executed with python scripts using Google Colaboration environment with GPU enabled notebook, reading the images from the same location in Google Drive, acting as a local drive for the notebook. Results of objects are stored in individual files with naming convention "confidences\_[METHOD NAME].csv" and image performance results are stored in individual files following the naming convention "resultados\_[METHOD NAME].csv".

##### 4.1. Time elapsed analysis

The first analysis was option's speed performance. From data insight shown in Table 1 it can be observed the fastest option is Yolo3Tiny and the slowest is HOGCV; Azure presented the maximum elapsed time with more than 10 seconds, and Yolo3Tiny reported the minimum elapsed time. Figure 4 shows the resulting distribution of each option, where HOGCV presents a double lump distribution, suggesting two main groups of elapsed times, while the rest of options shows a more regular distribution.

Another data coming from the observations is the number of objects detected per image. Table 2 shows the insights of this metric, having the highest mean of detected objects from AWS Rekognition, with 6.06, followed by Yolo3 recording 4.63. Maximum number of persons found is 30 (AWS Rekognition) followed, again, by Yolo3. It is interesting to observe the other options in maximum persons have near 50% of success.

It is important to find if there is observable correlation between the number of objects detected and the speed of the algorithm. Figure 5 shows Azure as the most uniform performance with less variations, although the rest of the options behave in similar fashion. Only HOGCV shows that double pattern.

Figure 6 shows a correlation between Yolo3Tiny speed and maximum number of objects detected. It can be observed it is the lowest elapsed time but also the lowest number of objects found; Azure follows similar pattern as well as Yolo3. It is also interesting to observe HOGCV with second worst

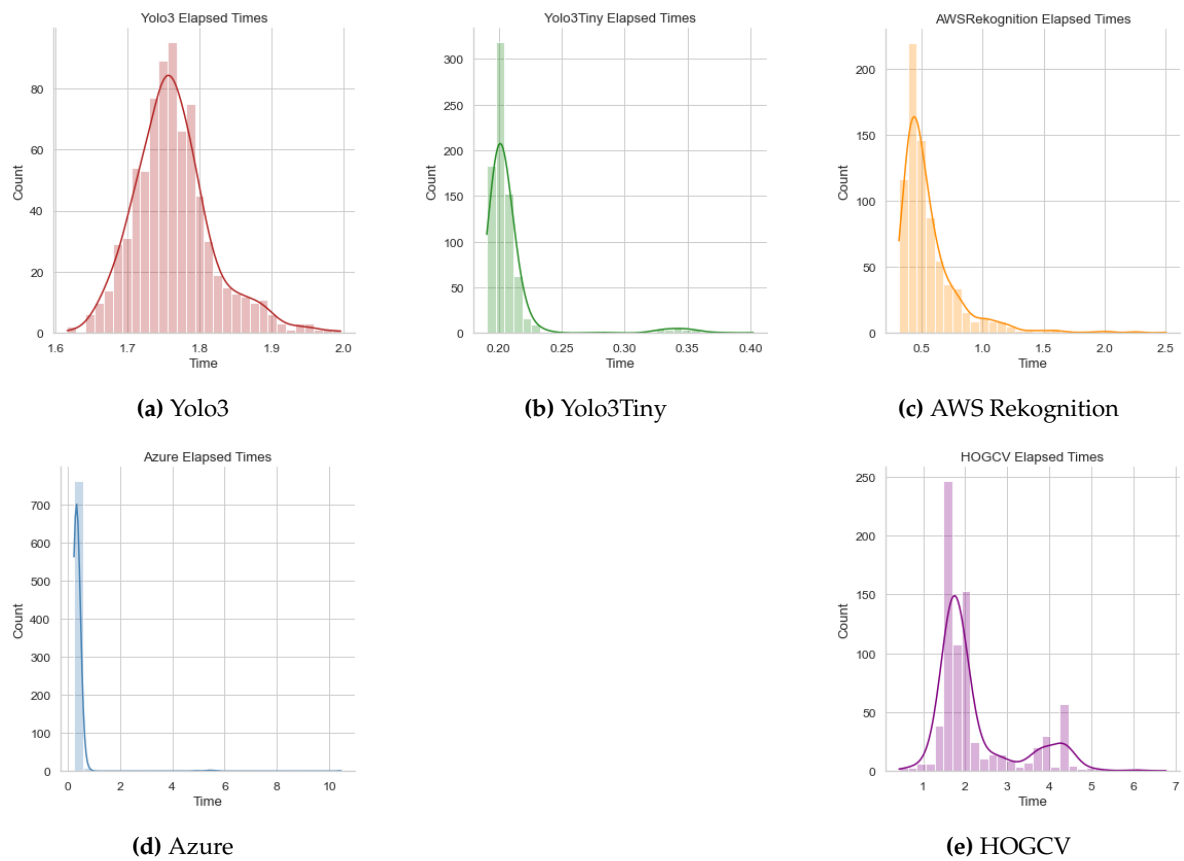


Figure 4. Elapsed time distributions for each option

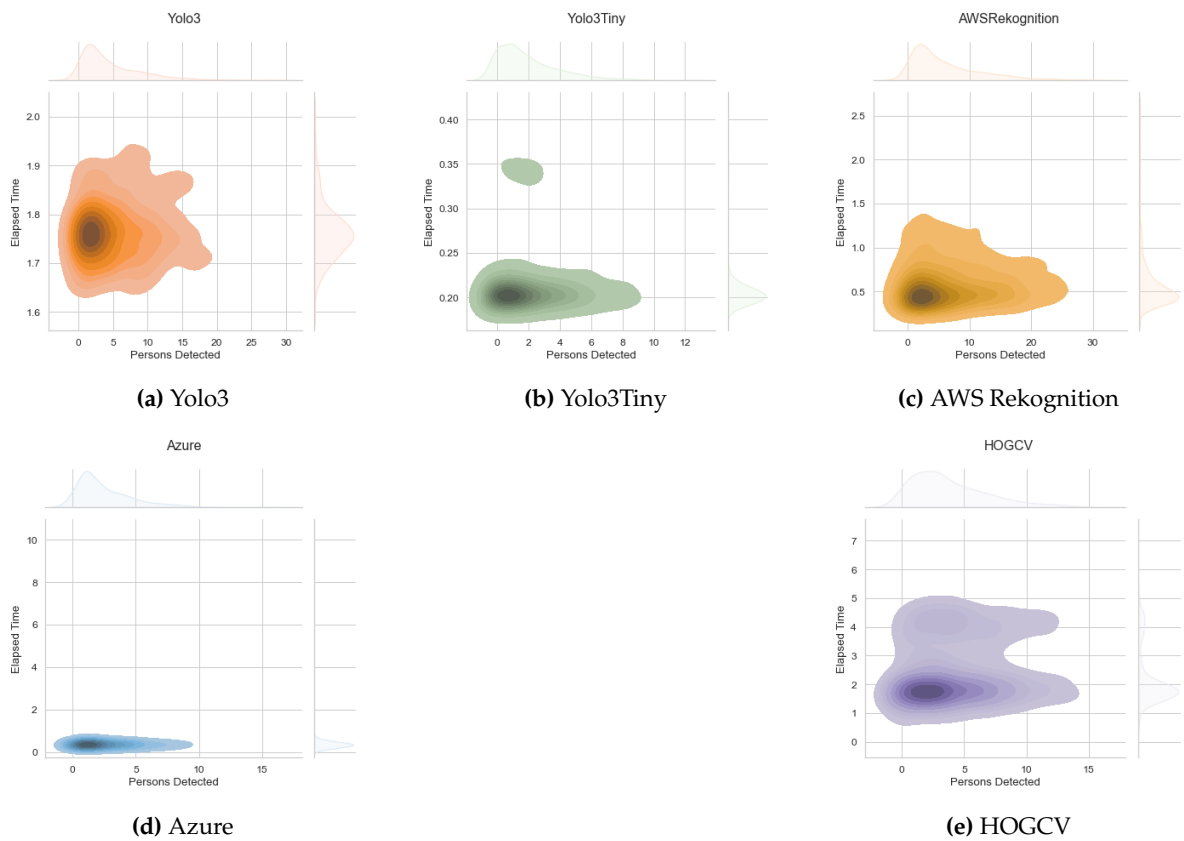


Figure 5. Objects found vs Elapsed time for each option

number of objects detected having the highest elapsed time. The other exception is AWS Rekognition, which has the third best elapsed time average but the best number of objects detected.

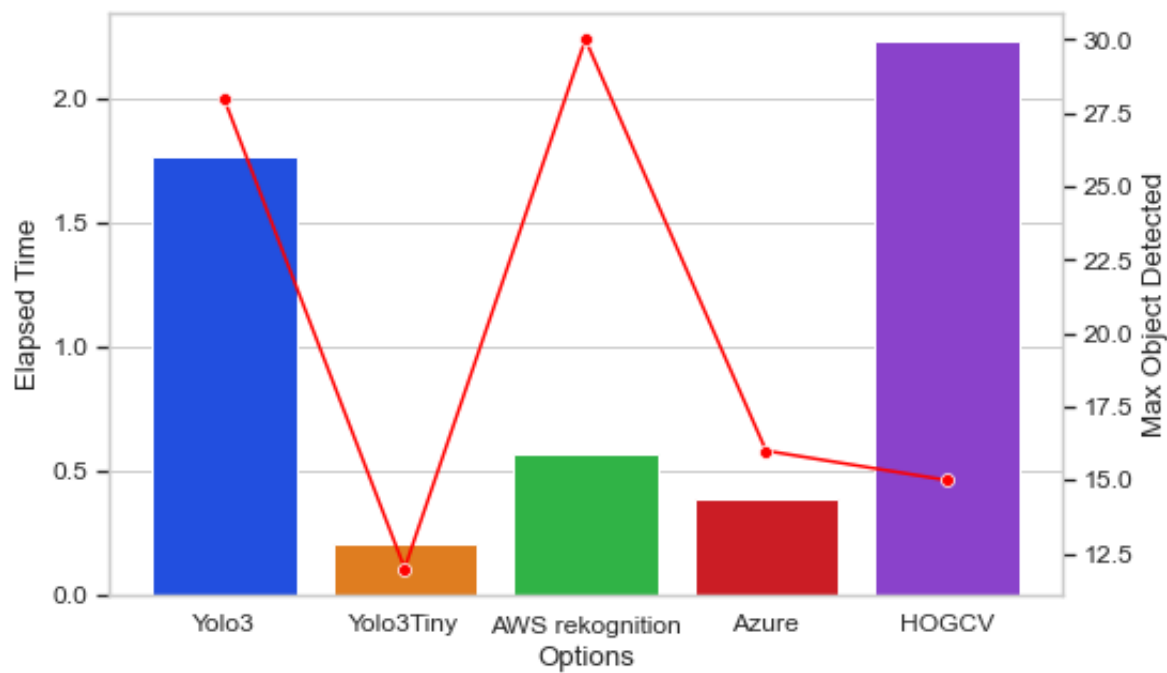


Figure 6. Elapsed time compared versus maximum numbers of objects detected

Finally, the last analysis on the topic is regarding trends, looking for an increase of time for a busier image. In this experiment there is no trend showing higher the number of persons in a picture, the higher the elapsed time. Figure 7 shows all trendlines compared in a single chart.

4.2. Error human versus computer vision

Computer vision methods may detect exactly all persons in an image as the human eye, or a different amount. The result cannot be measured as a label and the process is manual to achieve a "true" value of persons in an image. In this experiment, eight students volunteer to review each of the 775 pictures in the dataset and record how many persons they see. In figure 8 it can be seen there are differences between what humans often see and what a computer vision algorithm can "see". There are cases where human and computer vision match. Based on that analysis it can be conclude the average of persons detected by humans in each image is acceptable to be the "ground truth", and it can be compared against each algorithm option available to measure the error based on equation 6, taking mPDH as the true label to match. Table 3 shows the results of error  $\epsilon$  where Yolo3Tiny is the

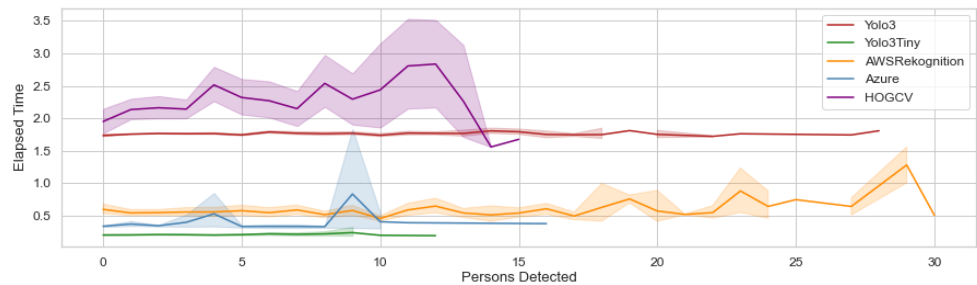
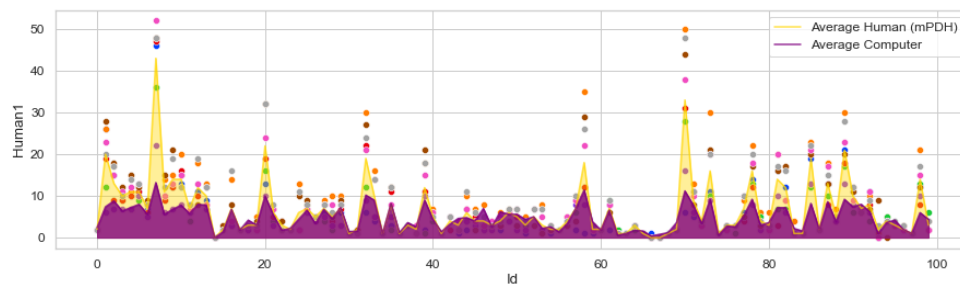
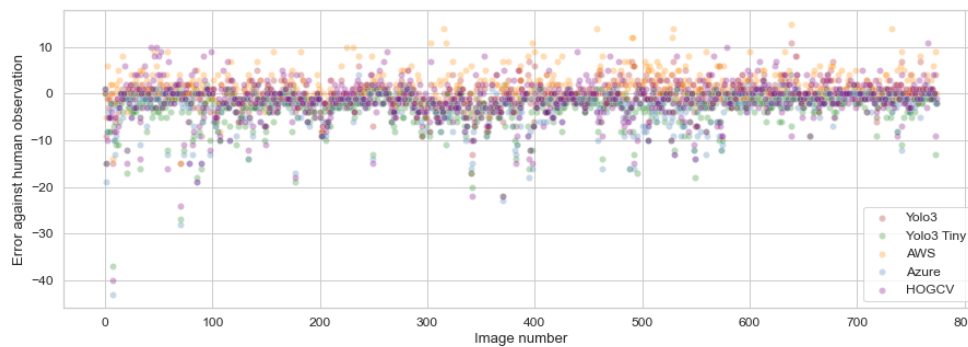


Figure 7. Elapsed time compared versus maximum numbers of objects trendlines



**Figure 8.** Comparison of averages of persons detected by humans and persons detected by computer options, including each volunteer observation (dots) images 0 through 100.



**Figure 9.** Computer vision error against mPDH in all options available.

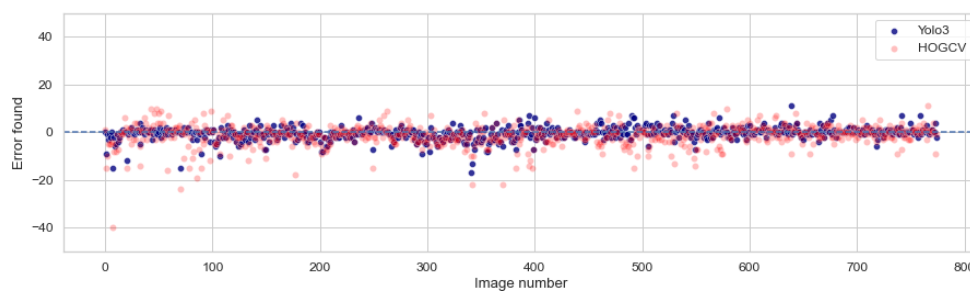
worst option detecting less objects than human just followed by Azure, and the best option is AWS Rekognition. Exact number of objects detected, meaning  $\varepsilon = 0$ , the highest option become Yolo3, commercial APIs are very close in performance, and the worst option is Yolo3Tiny. When algorithms surpass naked eye, AWS obtained the highest value, almost six times better than Azure. Figure 9 shows the dispersion and calculating the standard error of mean (SEM) by dividing standard deviation over the square root of number of observations, as stated in Equation 10.

$$SEM = \frac{\sigma}{\sqrt{n}} \quad (10)$$

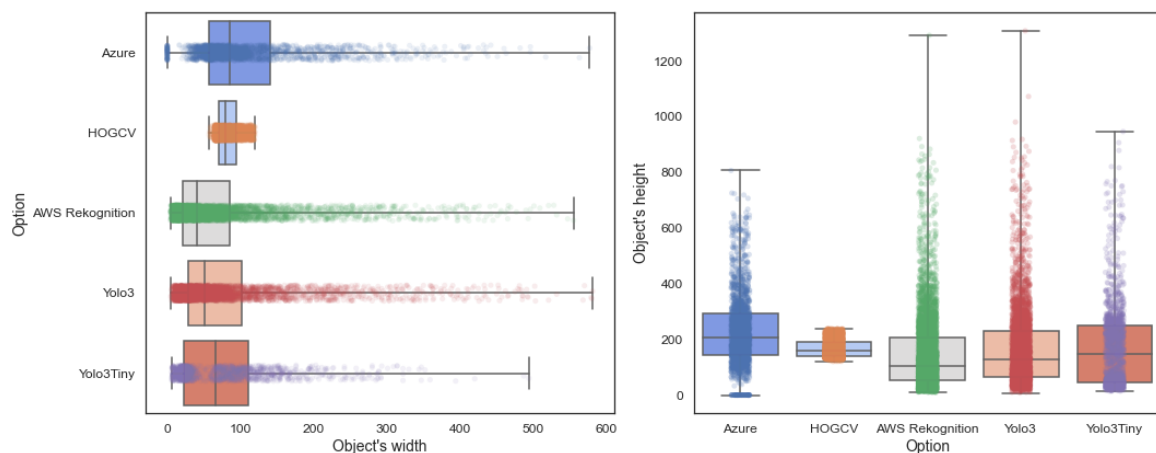
The SEMs are:  $SEM_{Yolo3} = 0.0949$ ,  $SEM_{Yolo3Tiny} = 0.1315$ ,  $SEM_{AWS} = 0.1177$ ,  $SEM_{Azure} = 0.1405$ , and  $SEM_{HOGCV} = 0.1617$ . The least SEM is Yolo3 option, and the second best is AWS, while worst option is HOGCV. This metric is very useful, because just plotting the errors could mislead the interpretation. Figure 10 shows the comparison of algorithms errors from best and worst SEM, Yolo3 and HOGCV respectively. It can be seen easily on the figure plotting their errors doesn't show a significant difference to conclude HOGCV is worst than Yolo3.

**Table 3.** Error count per option divided by less objects found ( $\varepsilon < 0$ ), equal match ( $\varepsilon = 0$ ), or more objects found ( $\varepsilon > 0$ )

Option	$\varepsilon < 0$	$\varepsilon = 0$	$\varepsilon > 0$	Total
Yolo3	341	225	209	775
Yolo3Tiny	659	91	25	775
AWS	234	160	381	775
Azure	553	158	64	775
HOGCV	426	112	237	775



**Figure 10.** Yolo3 (best SEM=0.0949) errors compared against HOGCV errors (worst SEM=0.1617) in all observations.

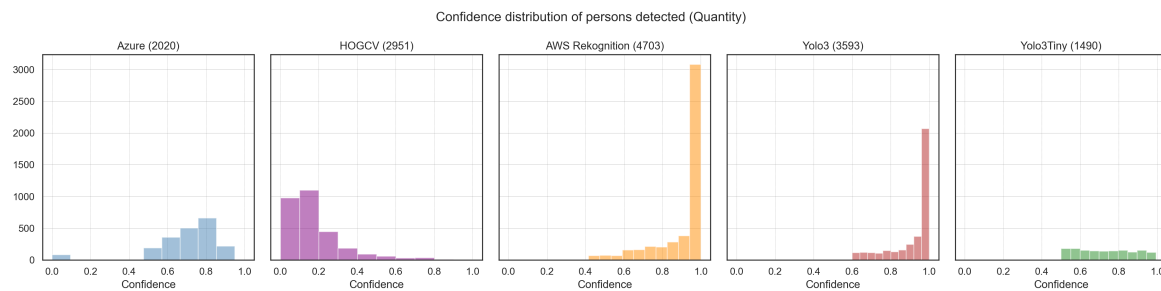


**Figure 11.** Analysis of object found size in pixels for all options

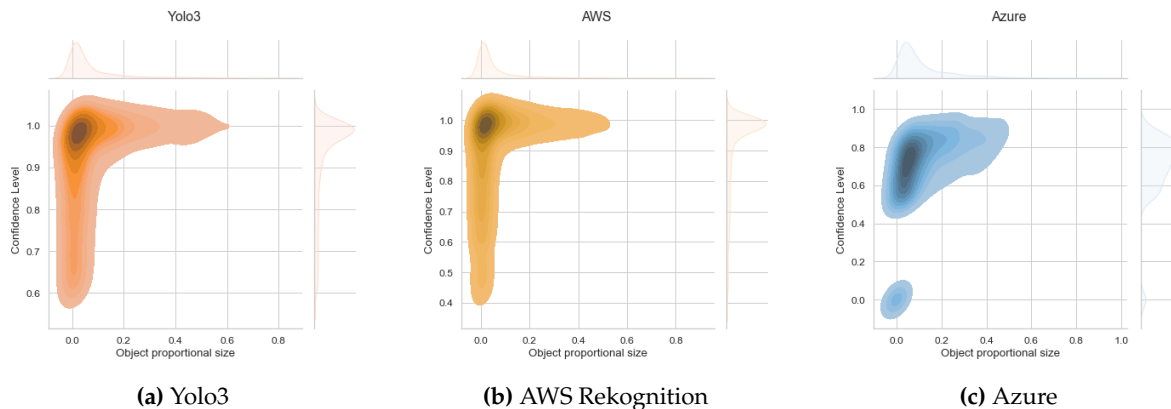
For the 775 images in the dataset all algorithms registered 14757 objects found. Figure 11 describe a very regular object width and size, with two outstanding findings. First, Azure seems to be more relaxed in the location of object within the image, it presented a slight higher value in both dimensions. The second finding is HOGCV is the tightest are among these five options. That could explain why its performance was not very good compared with the other four. Each object was detected with a specific confidence, and Figure 12 shows the distribution of those confidences by option available. From this analysis, AWS seems the best confidence with the majority of their findings above 0.9 confidence. Yolo3 follows the same distribution pattern. Among these two options, AWS detected 1110 more objects than Yolo3, and have mean confidence of 0.9085, while Yolo3's mean confidence is 0.9184. Azure seems to be the next best option with 0.7 confidence mean. HOGCV confidence is kind of low showing 0.1777 confidence mean. Yolo3Tiny confidence mean is 0.7309, however it has the lowest quantity of objects detected. Those findings could suggest Yolo3 and AWS Rekognition are behaving similar. When we compared confidence level against proportional object size in relation to image size, AWS Rekognition and Yolo3 describe a very similar pattern, and the most likely to this behavior is Azure (Figure 13).

#### 4.3. Vision acuity index

As the previous results are not conclusive to determine which method is the best suited for our case, additional analysis is required. The next step is to obtain the precision and recall of all pictures produced (analyzed) by each algorithm. This require manual effort as it requires to record true positives, true negatives, false positives and false negatives. Again, a group of volunteer students of computer engineering program recorded these four metrics on each picture. Because of the error dispersion and discrepancies found during error analysis phase, those results suggest different number of persons detected, different dimensions or boxes where the person is located, and possibly, persons



**Figure 12.** Analysis of confidences of each object detected by option available



**Figure 13.** Bivariate graphic of confidence level and object's proportional size

missed or misclassified. Three examples are shown in this section, one with single person (Figure 14), one with multiple persons of similar proportions (Figure 15) and one example with multiple size persons.

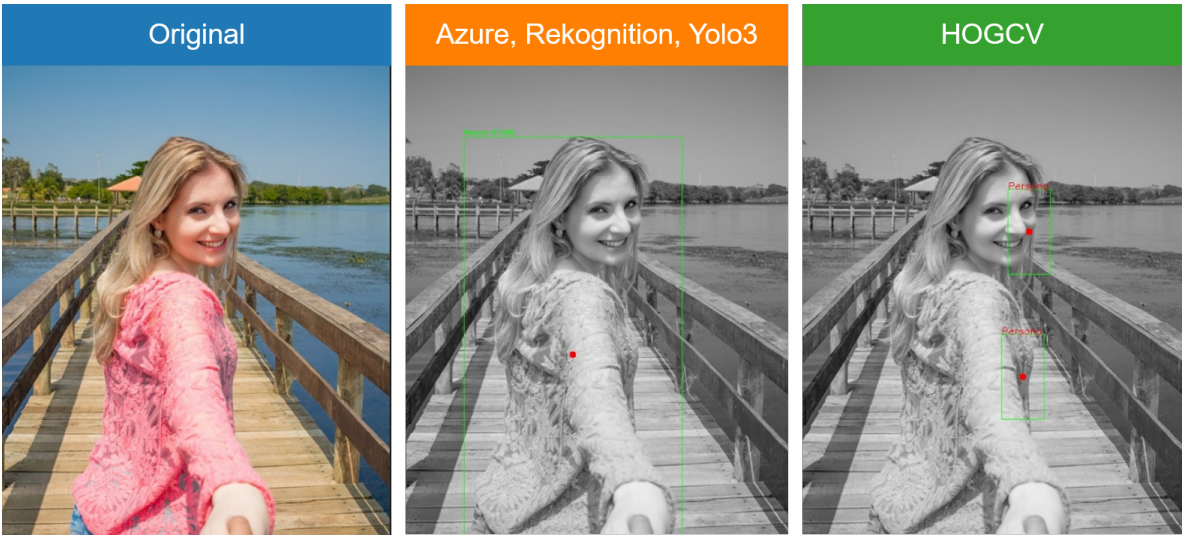
Figure 14 is 600 x 899 pixels. Yolo3 found correctly the person of size 319x705px with a confidence of 0.9933. Yolo3Tiny did not detect the person, so it's a false negative. AWS detected correctly the person of 339x675px with confidence of 0.9764. Azure found the person of 327x668 px with a confidence of 0.816. Last, HOGCV detected two persons (false positives) of sizes 66x132 with a confidence of 0.0979.

Figure 15 has six persons on the image, corroborated by human eye, in a picture of 600x399px. Yolo3 recorded six persons with confidence average of 0.9871 and person sizes of 100x226 in average. Yolo3Tiny has succeeded only to detect five persons with average confidence of 0.7029 and object size average of 113x203px. AWS recorded six persons with average confidence of 0.9746 and object size average of 106x207. Azure also detected six persons but the average confidence was 0.6758 and average object size of 106x212px. Last, HOGCV recorded twelve persons (5 true positive, 6 false positive, 1 false negative) with an average confidence of 0.1763 and average object sizes of 76x153px.

It is clear the performance of algorithms, areas of detected persons and confidence levels are different. In the last example, the size of each person were similar, the picture was centered and they had the similar pose. Even so, the HOGCV couldn't detect correctly the persons, Yolo3Tiny missed one, and the rest presented good results. The F1 Score of each algorithm for this picture was Yolo3 = 1, Yolo3Tiny = 0.90, AWS and Azure = 1, HOGCV = 0.56.

Last example from the dataset is Figure 16 showing a person in first plane and then five more bystanders in different sizes. Again, Yolo3 detected five persons with F1 Score of 0.9091, Yolo3Tiny and Azure found one person and reached F1 Score of 0.2857 each, AWS found all six persons, therefore its F1 Score is 1, and HOGCV detected one person and provided a false positive for a F1 Score of 0.2500.

These results are interesting and also can mislead the decision to select the best option for a specific application. It was found equal F1 Score for two algorithms but with different person detected. Figure 15 and 16 are excellent examples of this problem. In Figure 15, same F1 score 1 is shared by



**Figure 14.** Example of single person detected in image with one person (true positive) and how a method can miss the person detection (HOGCV). Foto source: Flickr.com

Yolo3, AWS and Azure, but can’t decide the best option; Figure 16 Azure and Yolo3Tiny obtained F1 Score of 0.2857. Fq Score by itself can’t determine the option to choose. Also, Azure detected the larger person while Yolo3Tiny detected one of the smallest, showing the size of object plays a part when detecting objects. Similarly, even Yolo3 and AWS have high F1 Scores, AWS could detect the smallest object (the one behind the largest person) with  $228px^2$ . Here is where the concept of vision acuity index (VAI) is relevant.

To apply Equation 4 on Figure 16 we obtained the confidence levels and dimensions of each object, and the F1 Score for the image, summarized in Table 4. By using VAI, it can be observed that even Yolo3Tiny and Azure share the same F1 Score, their VAI is different, showing Yolo3Tiny’s VAI of 0.8712 while Azure’s VAI is 0.3427. This can be interpreted as Azure looks for larger objects, or Yolo3Tiny can detect detail better. VAI for AWS and Yolo3 shows a significant difference meaning AWS is more detailed oriented or it can have a better acuity.

To discuss in depth the results obtained applying VAI, Table 5 shows the VAI obtained for each Figure and option available. Figure 14 obtained F1 Score of 1 with Yolo3, AWS and Azure, meaning the three of them are excellent options. By calculating VAI, it shows Azure has a lower index compared to Yolo3 and AWS, and Yolo3 has a slight edge over AWS. That is explained as Yolo3 reported a higher confidence level on each object detected. In Figure 15, where all persons are of similar dimensions, Yolo3, AWS and Azure again obtained a perfect F1 Score. Once again, Azure has the lowest VAI of those three. This is explained as the confidence reported by Azure in each of the persons detected has a mean of 0.6758, and that could lead to false negatives in real applications or multi-object detection. Yolo3 and AWS differ slightly again on VAI. Yolo3 had a mean confidence of 0.9871 and AWS had 0.9746, however, AWS reported tighter boxes, meaning a preciser object location, therefore higher VAI.

5. Conclusions and future research

The analysis performed with 755 images and five options of algorithms or APIs for detecting persons resulted in 3,775 images analyzed with precision, recall, F1 Score, performance speed and errors in person detection. We observed that choosing an option for speed may not deliver the effectiveness desired, and therefore, the application or business problem to solve would not be as effective as initial thought. Also, make a decision for the solely number of persons detected also could mislead the choice. In this methodology we observed the need to validate with supervised analysis the performance of computer vision options to know ahead of time, what should be the results required. Although, relay on this solely variable is cumbersome and requires a vast amount of time investment



Figure 15. Example of multiple person detection with similar size. Foto source: Flickr.com

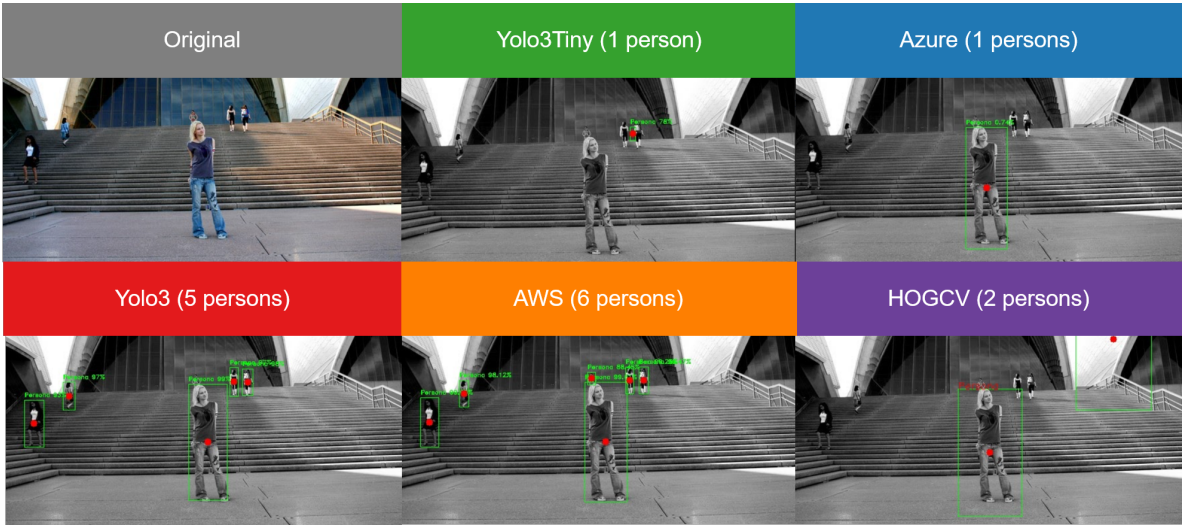


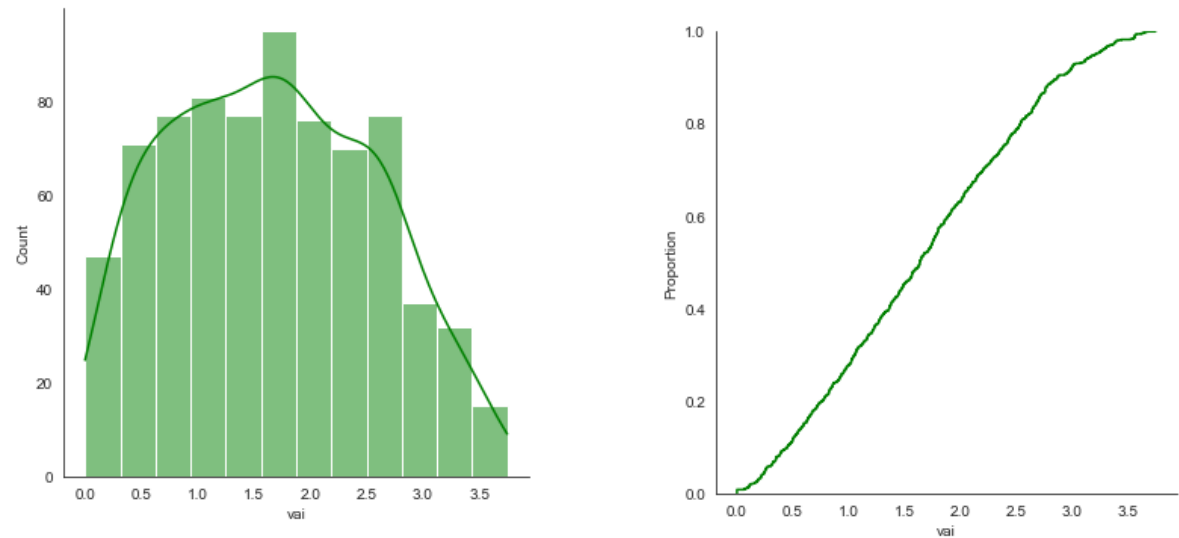
Figure 16. Example of multiple person detection of different sizes. Foto source: Flickr.com

Table 4. VAI calculation for image in Figure 16

Option Available	Conf. ( $\omega_i$ )	Dimensions (i)	$px^2$	Prop Surface ( $S_i$ )	F1Score	VAI
Yolo3	$Obj_1$ : 0.9995	58X175	10150	0.0425	0.9090	2.8081
	$Obj_2$ : 0.987	16X40	640	0.0027		
	$Obj_3$ : 0.9785	13X43	559	0.0023		
	$Obj_4$ : 0.9775	18X42	756	0.0032		
	$Obj_5$ : 0.938	29X71	2059	0.0086		
Yolo3Tiny	$Obj_1$ : 0.7884	8X21	168	0.0007	0.2857	0.8712
AWS Rekognition	$Obj_1$ : 0.9969	27X74	1998	0.0084	1	3.3807
	$Obj_2$ : 0.9962	63X179	11277	0.0472		
	$Obj_3$ : 0.9947	14X40	560	0.0023		
	$Obj_4$ : 0.9925	12X40	480	0.002		
	$Obj_5$ : 0.9813	14X40	560	0.0023		
	$Obj_6$ : 0.8845	12X19	228	0.001		
Azure	$Obj_1$ : 0.744	62X181	11222	0.0469	0.2857	0.3427
HOGCV	$Obj_0$ : 0.306	99X197	19503	0.0817	0.2500	0.2050
	$Obj_1$ : 0.306	117X218	25506	0.1068		

**Table 5.** VAI comparison on each option available for example Figures

Figure	Option Available	F1 Score	VAI
Fig. 14	Yolo3	1	0.3770
	Yolo3Tiny	0	0
	AWS Rekognition	1	0.3620
	Azure	1	0.3042
	HOGCV	0	0
Fig. 15	Yolo3	1	1.8025
	Yolo3Tiny	0.9091	1.4253
	AWS Rekognition	1	1.8127
	Azure	1	1.6422
	HOGCV	0.5556	0.9406
Fig. 16	Yolo3	0.9091	2.8082
	Yolo3Tiny	0.2857	0.8713
	AWS Rekognition	1	3.3808
	Azure	0.2857	0.3427
	HOGCV	0.25	0.2050



(a) Histogram and distribution of VAI index for Yolo3 in dataset

(b) Empirical cumulative distribution

**Figure 17.** VAI index calculated for all 755 images on dataset for Yolo3 algorithm

to have a good evaluation. This will eventually lead to evaluate the results with F1 Score, however, it was also observed the F1 Score is not enough to decide the option to use in an application.

The novel Vision Acuity Index provides a better comparison measurement to evaluate two or more options that may have the same F1 Score or similar metrics. This is very important when you are trying to select the option for a business application, and the only measurement they have is commercial specifications, or data scientist training their own solutions, or leveraging an existing one. The business projects can get a lot of benefit by using this VAI as a method to select object detection algorithms, as now they can evaluate what happens if the environment change or the variables are not the same. As an example, counting people waiting on line for cashier for a retail store chain can have different vantage points, layouts and distance among the stores, making interesting to evaluate which object detector to include and invest on. In the same idea, counting people where the variables are fixed and under control, may or may not get the same performance of the same algorithm used for other scenario.

While analyzing different angles to find the best set of parameters to allow us select the best suited algorithm or option, we developed the methodology that can be used as a starting point for other researchers to build on top of ours. We noticed a side finding as Yolo3 and AWS Rekognition are similar. Similar speed, similar object detected boxes and sizes, close confidences in detection, and very similar pattern when comparing confidence level and object proportion. Therefore, this methodology proof that can be well suited to analyze open source and commercial options.

For future research, we need to apply VAI methodology for additional options such as RetinaNet, CornerNet (as options for one-stage detectors), SPPNet, Fast-RCNN, FPN (as options of two-stage detectors) and additional commercial options such as Google Cloud Platform. For further application, we will apply this methodology for a real world problem, using live video feed, extending the VAI concept with frame processing.

**Author Contributions:** The contributions for this research were conceptualization by R.C-M, A. O-Z; methodology, V.T-A.; software, R.C-M, V.G.; validation, L.P-D., J.M. and M.E-C.; supervision, A.O-Z.

**Funding:** This research received no external funding

**Acknowledgments:** We want to give special thanks to the students of Tecnológico Nacional de México campus Instituto Tecnológico de Ciudad Juárez for their support to manual identify number of persons, true positives, true negatives, false positives and false negatives in the whole dataset.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

AP	Average Precision
API	Application Programming Interface
AUC	Area Under Average ROC
AWS	Amazon Web Services
CNN	Convolutional Neural Network
HOG	Histogram of Oriented Gradients
mAP	Mean Average Precision
mPDH	Mean Person Detected by Humans
ROC	Receiver Operating Characteristic
SEM	Standard Error of Mean
URL	Uniform Resource Locator
VAI	vision Acuity Index
YOLO	You Only Look Once

## References

1. Murthy, C.B.; Hashmi, M.F.; Bokde, N.D.; Geem, Z.W. Investigations of Object Detection in Images/Videos Using Various Deep Learning Techniques and Embedded Platforms—A Comprehensive Review. *Applied Sciences* **2020**, *10*, 3280.
2. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.
3. Redmon, J.; Farhadi, A. YOLO9000: better, faster, stronger. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 7263–7271.
4. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* **2018**.
5. Xiao, D.; Shan, F.; Li, Z.; Le, B.T.; Liu, X.; Li, X. A target detection model based on improved tiny-yolov3 under the environment of mining truck. *IEEE Access* **2019**, *7*, 123757–123764.
6. Mao, Q.C.; Sun, H.M.; Liu, Y.B.; Jia, R.S. Mini-YOLOv3: real-time object detector for embedded applications. *IEEE Access* **2019**, *7*, 133529–133538.
7. Nazir, M.; Jan, Z.; Sajjad, M. Facial expression recognition using histogram of oriented gradients based transformed features. *Cluster Computing* **2018**, *21*, 539–548.
8. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). IEEE, 2005, Vol. 1, pp. 886–893.
9. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. Proceedings of the IEEE international conference on computer vision, 2017, pp. 2980–2988.
10. Padilla, R.; Netto, S.L.; da Silva, E.A. A survey on performance metrics for object-detection algorithms. 2020 International Conference on Systems, Signals and Image Processing (IWSSIP). IEEE, 2020, pp. 237–242.
11. Wang, X.; Zheng, S.; Yang, R.; Luo, B.; Tang, J. Pedestrian attribute recognition: A survey. *arXiv preprint arXiv:1901.07474* **2019**.
12. Kabakus, A.T. An Experimental Performance Comparison of Widely Used Face Detection Tools. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal* **2019**, *8*, 5–12.
13. Gao, X. Performance evaluation of automatic object detection with post-processing schemes under enhanced measures in wide-area aerial imagery. *Multimedia Tools and Applications* **2020**, *79*, 30357–30386.