Stop the Bus!: Computer vision for automatic recognition of urban bus lines.

Hernán J. Maina[2,3] and Jorge A. Sánchez[1,2,3]

[1] CONICET, Córdoba, Argentina
[2] Universidad Nacional de Córdoba, Córdoba, Argentina
jorge.sanchez@unc.edu.ar
[3] Facultad de Matemática, Astronomía, Física y Computación**
hernan.maina@mi.unc.edu.ar

Abstract. In the present work, the problem of the detection and the recognition of bus line numbers of public transport in the city of Córdoba is addressed, using images obtained by standard mobile devices. The goal of this project is the exploitation of computer vision techniques and the analysis of images for the generation of a tool that potentially allows people with visual impairments to be assisted. To achieve this, a modular architecture based on object detectors and optical character recognition is presented and evaluated, mainly constituted by two stages: one for the detection buses, based on the SSD-MobileNet object detection model; and another, responsible for line number recognition, where the EAST and OCR-Tesseract text detection and recognition models are tested, respectively. With a maximum probability of recognition of 62% in a simple image; over an image sequence, the final system was able to correctly recognize the bus line in 72% of the cases tested.

Keywords: computer vision · object detection · text detection · text recognition · buses · visual impairment.

1 Introduction

Currently, millions of people living in the world have difficulties understanding the environment around them due to some type of visual impairment. According to recent data from the World Health Organization, it is estimated that around 1.3 billion people in the world live with some type of visual disability, among which: 188.5 million people categorized as having mild impairment, 217 million a moderate to severe impairment and 36 million people have a total loss of vision.

As with any other disability, living with restrictions represents a constant challenge; in particular, the lack of visual accuracy, makes everyday life situations much more difficult to perform. While alternative approaches to dealing with such routines may be developed, one of the immediate consequences of this impairment, is reflected in the feeling of insecurity experienced when moving or traveling independently, mainly in outdoor and unknown environments [11].

** https://www.famaf.unc.edu.ar

In recent years, due to the major advance in the computer vision discipline, system has been developed to create a solution to such problems. From applications such as TapTapSee[4], that describes an image captured in a few seconds, and developments made by companies like Facebook and Twitter, which through image captioning technology try to expand and adapt the world of social media and visual content; to more extreme attempts at sensory substitution systems such as vOICe[5], intended to facilitate understanding of the surrounding environment [1,4,5].

In this work, a modular algorithm is proposed, adjusted and tested, with the aim of automating the daily task of *bus line number recognition*, thus facilitating access to urban transport in the city of Córdoba, for people with visual impairments.

Pursuing accessibility, and due to the high performance that on average, modern mobile phones and their integrated cameras have; initially, the project was thought to be a future adaptation in a mobile application. However, given the multitude of high-performance portable development platforms such as NVIDIA Jetson Nano, Coral Development Board and some recent Raspberry Pi models, it is possible that any one of these supports could be used for implementation.

2 Architecture

Taking a standard bus stop as the initial scenario, the user will start by indicating the expected bus line number to the system. Simultaneously as the bus approaches and/or stops, the images captured will be processed, generating a list of 5 possible candidate lines. In the case, where the line number is part of such a list, the arrival of the desired bus will be given. Although out of our reach, such a warning could easily be implemented through vibration feedback, freeing the ear and avoiding the auditory overload that makes it difficult to understand the nearby environment.

First, our algorithm, built with an "object detector" module, will receive the input images (frame), and look for regions that could potentially contain buses. Subsequently, each of these areas (ROIs of buses or *Regions Of Interest of buses*) will be received by an intermediate stage of processing, which, through a series of filters and "color space transformations", and with the aim of exposing regions of text, will generate new representations, through changes in contrasts, thresholds, saturations and equalizations; which will feed the "text detection" stage. This, as its name implies, will locate areas of alphanumeric characters in the images, in the hope that some of them contain the line number expected. Next, each one of such detections (ROIs of text), together with the representations obtained as a result of passing through a new intermediate stage of "segmentation", will be decoded into plain text using the "text recognition" module. Finally, after filtering out non-numeric characters, the list of possible numbers of candidate

[4] https://taptapseeapp.com/
[5] https://www.seeingwithsound.com/

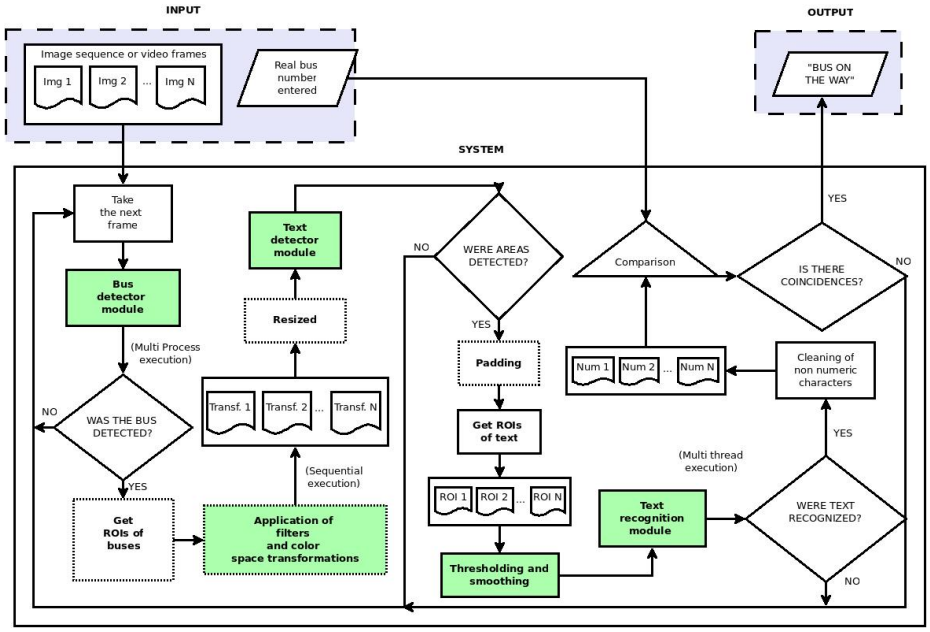

Fig. 1. End-to-end system flow chart and details.

lines will be counted and prepared. The specific stages of the entire process can be seen in Figure (1).

2.1 Bus detection

For bus detection, and in order to find the most suitable model for our needs, Section (4); three different variations of object detectors were implemented and tested: YOLO v2 [13], YOLO v3 Tiny (very small model of YOLO v3 [10]) and SSD-MobileNet 300 [3,7]. The same, selected for their excellent features for real-time detection.

Both YOLO v2 and YOLO v3 Tiny were encoded using a Python wrapper obtained from the Darknet[6] Neural Networks framework compilation; making use of the set of parameters pre-trained by the same creators, on the COCO dataset[7].

For the SSD-MobileNet 300 model, a Caffe version based on the original TensorFlow implementation of Howard et al. was used, which was trained by chuanqi305[8], first over COCO dataset, and then fited over PASCAL VOC[9]. The

[6] https://github.com/pjreddie/darknet
[7] http://cocodataset.org/
[8] https://github.com/chuanqi305/MobileNet-SSD
[9] http://host.robots.ox.ac.uk/pascal/VOC/

reading, was done through the Deep Neural Networks (dnn) module, included in the OpenCV 4.0[10] library.

Note that the values of each set of pre-trained weights already includes the bus category, within such training.

2.2 Lines numbers detection

The main challenge with this task was related to how most of the bus line numbers in the city of Córdoba are represented. These, unlike many other places in the country, where they are painted or stamped using homogeneous vinyls, are made up of a set of LED diodes. Despite being very dense, the movements during the captures, the lighting reflections produced on the windshield, and the very brightness they emit, made the task of identification a challenge. The Figure (3 right) reveals the difficulties mentioned.

Considering the nature of such difficulties, we opted to use EAST (Efficient and Accurate Scene Text detector) [14] as a text detector; first, because of its balance between detection accuracy and speed , and second, because it has a deep learning model that makes it possible to avoid computationally expensive sub-algorithms (such as candidate aggregation and word partitioning), than other text detectors, usually need [8].

EAST was also coded in Python using the Deep Neural Networks module (dnn), included in OpenCV 4.0, which also provided us with the pre-trained serialized model in '.protobuf' format.

2.3 Lines numbers recognition

Although OCR systems are no longer a novelty, they continue to be an active part of multiple investigations, especially those related to images of unrestricted real-world environments [8,9]. Since many of the best algorithms used for such a task are for commercial use, this project chose to use open source Tesseract v4.1 [11]. This particular version, unlike its predecessors, includes a new recognition engine of the LTSM type, which offers relatively greater precision, at the cost of an increase in the required processing power.

2.4 Color spaces transformations, and segmentation

As already mentioned at the beginning of this section, the choice of a good set of color spaces was essential to expose, and later facilitate, the recognition of the line numbers sought. Both the analyzed spaces and the details regarding their selection will be described in Section (3) [2,6].

While Tesseract can work very well under controlled conditions, it could perform less than accurate decoding when faced with significant amount of image noise. To cope with this weakness, it was empirically found that the application

[10] https://opencv.org/opencv-4-0/

[11] https://github.com/tesseract-ocr/tesseract

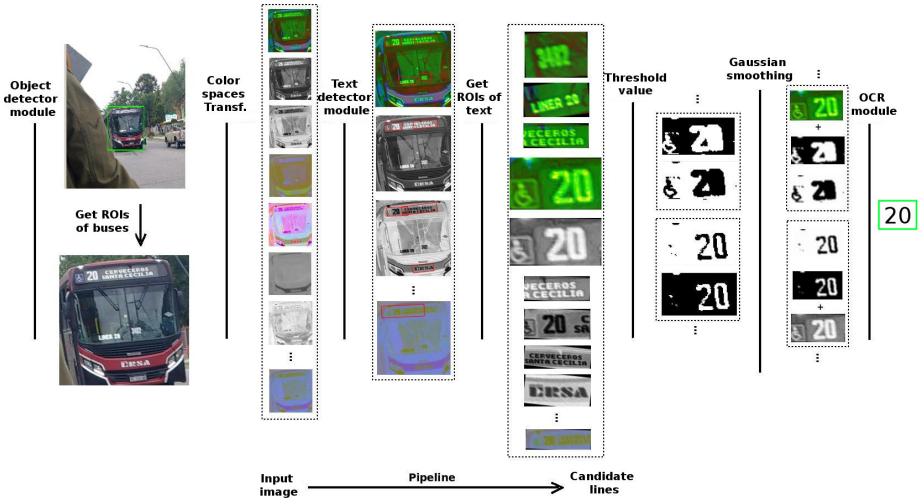

Fig. 2. Sequence of transformations undergone by an image, from its entry into the system, to the prediction of the line number, of the captured bus.

of the Single Threshold value method could, under circumstances, give greater readability to the ROIs of text delivered by EAST. A problem caused by such thresholding, and possibly related to the flashes of light emitted by the same LED diodes that make up the line numbers, was that, in some circumstances, the results were perceived as if they were "pixelated numbers". In order to eliminate such a perception, and avoid potential recognition failures, a Gaussian smoothing was performed; so that in this way, and by blurring its limits, such appearance could be abolished. The partial results and transformations carried out by each of the aforementioned stages were represented in detail in Figure (2).

3 Detection and line recognition stages: Parameters fit

In this section, different parameter configurations related to text detection and recognition modules were analyzed and tested. These experiments were oriented to find the best combination of values that lead to greater precision and percentage of "positive detections"[12] in the last part of the system. The different parameters analyzed correspond to the stages delimited with dotted lines, shown in Figure (1).

These tests were performed over a representative and heterogeneous population sample of 100 bus images, meticulously selected from a total of 983. For this, effort was put into finding the group that best represented ambient lighting factors, technical issues derived from the sources of capture, and levels of stability at the time of capture (static or moving). This set consisted of images

[12] Any correct prediction thrown by a certain combination of parameters.

ranging from 640x480 and 960x1280 to 3000x4000 pixels, coming from 10 mobile phone sources, including: HUAWEI VNS-L23, Motorola MotoG3, Sony F5121 and others.

In order to isolate and make the results independent of possible errors introduced, due to inaccuracies coming from the bus detection module, a "theoretical and accurate" model was assumed. Making manual annotations of the bounding boxes of the buses for each image in the sample, Figure (3 left); they were used and returned, as if it were the detection module itself, each time the system was run with any of these, to locate such a bus.

The different aspects and parameter values were taken into account for each stage and modules mentioned are detailed below.

Obtaining ROIs of buses. This block, after the "buses detector" module, will deal with important tasks in relation to the limits of the ROIs of detected buses. On the one hand, it will filter based on the aspect ratio of the same; although it seems that possibilities are reduced, it is won by discarding lateral detections, which for the most part do not have printed information of the line number sought. For this reason, and detailed in Subsection (3.4), only relationships aspects that are identified with buses coming from the front or slightly lateralized will be accounted for.

On the other hand, this module will re-fit the detections, "moving" the limits of each ROI, so that their respective width and height obtain a square aspect ratio. This small modification, in addition to not interfering with the final results, will allow unifying dimensions and was designed to facilitate the analysis carried out.

Transformations and image filters. Fifty-two (52) transformations were analyzed, made up of seven (7) color spaces: rgb, hls, hsv, lab, yuv, YCrCb and luv; their corresponding individual derived channels, their inverses (mapping from light to dark areas and vice versa) and two edge detection filters, one horizontal and one vertical.

Fig. 3. *To the left:* shown the annotations process using the open source OpenLabeling. *To the right:* shown how are the numbers representations in the buses of Córdoba.

Resized. Because EAST requires images with dimensions multiples of 32 as input to its network, sizes of 64x64, 96x96, 128x128, 160x160 and 320x320 pixels were evaluated. The use of such square dimensions is consistent with the decision made regarding the characteristics that the ROIs of the detected buses should have, and thus avoid possible detection problems, due to deformations in the image, and therefore in the text, that could be caused.

Padding. Once the text areas are detected, it is possible that their limits are so close to the characters that, for example, it causes Tesseract to predict 'o' instead of '9' or '6'. To solve this problem, the "Padding stage" was implemented after the EAST module. In it, four (4) different percentages of 'pad' were tested: 0%, 5%, 10% and 15%, expanding the detection margins, in order to select the one that, on average, allows best results at the time of recognition.

3.1 Representation of test data

Initially, and in order to obtain the best combination of values that deliver the highest possible performance, it became necessary to find a way to represent and structure both the parameters to be tested and the results delivered by the system, as a consequence of such configurations. For this, 100 datasets were made, product of 100 executions, one for each image of the population sample, registering: 1) original number of the bus, 2) final prediction of the system, and 3) values of parameters involved in such prediction. In this way, it was ensured that every possible combination for each image in the sample was tested.

In each run, the internal configurations of the EAST and Tesseract models were: (*confidence*: 0.001, *Intersection over Union 'IoU'*: 0.1) and (*psm*: Mode 7, *oem*: Mode 1, *language*: English) respectively. For the particular case of EAST, the small values used, outside of the normal ranges [0.5, 0.8] for confidence and [0.4, 0.6] for IoU, were to counteract the perceived lack of cohesion in the pixels representing the line numbers of the buses (consequence of what was explained in Subsection (2.2), which caused multiple partial detections cut them in different regions. Figure (4), shows such an inconvenience, making an analogy between a bus from the city of Córdoba and another from the city of Buenos Aires.

3.2 Search for minimum set of best transformations

The first approach in the search for the best set of transformations was to perform, using the datasets made, a Heatmap of "positive detections vs. image transformations". In this way, a clearer vision of the behavior of such detections was obtained, in relation to the parameters 'resized' and 'padding'.

Being $V_{(i,j)}=n$ the value of the (i,j)-th "cell" of the Heatmap, n adds one unit for each of the values of *resized* and *padding* that participated in the *positive detection*, that the i-th transformation maked in the j-th clipping. The Figure (5) shows the Heatmap obtained, with the transformations in decreasing order, based upon the number of sample images (clipping) that each correctly predicted.

Fig. 4. Detection by EAST using: *confidence*: **0.5** and *IoU*: **0.6**. (*To the left*: Buenos Aires's bus - **'rgb'** transformation - vinyl number. *To the right:* Córdoba's bus - **'yuv'** transformation - number formed by LED diodes and partial detections).

Assuming that the best transformation is the one with the highest number of hits, the search for the minimum set could be reduced, initially, to the trivial task of selecting the first n transformations of our Heatmap, and in each step, compute until finding the set that had the highest percentage of predictions delivered over the total images tested. Although this could be considered a valid approximation and work correctly, upon observing in detail, many of the hits were made by multiple transformations simultaneously and with similar intensities, deriving to an unnecessary computational overhead dedicated to the generation of such transformations, avoidable, since such transformations do not contribute more than some than others.

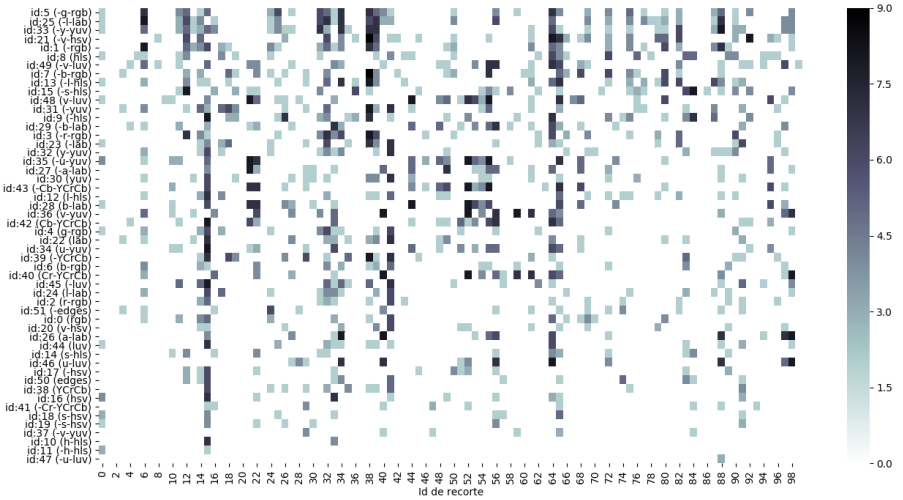

Fig. 5. Heatmap of positive detections vs. transformations. (s-$hls \equiv sss_{hls}$, that is, the result of transforming $rgb \rightarrow hls$, taking channel s, and copying it to the remaining two. A prefix '-' (ex. -hls) indicates the inverse of such a transformation).

To solve the problem, an algorithm capable of reordering such transformations was coded in relation to the level of detection robustness and the order of complementarity between them. In other words, the algorithm takes as input an ordered heatmap (like the one presented above) and maintains the position that the best transformation is the one that has carried out the greatest number of detections; rearranges the rest of the transformations according to their ability to adapt to this last one, prioritizing: 1) extra percentage of detections provided by the new transformation, 2) greater number of intersections of detections, and 3) greater weighted value (given by *resized* and *padding*) in each intersection.

Below, the algorithm output for two specific situations shows the maximum recognition percentages that could be achieved, if it were decided to use the minimum set of 5 and 15 (Top 5 and Top 15) best transformations found:

- **out1:** *65% of lines could be recognized by transformations → ['-g-rgb', 'v-luv', '-Cb-YCrCb', '-u-yuv', '-s-hls'].*
- **out2:** *90% of lines could be recognized by transformations → ['-g-rgb', 'v-luv', '-Cb-YCrCb', '-u-yuv', '-s-hls', 'hls', 'b-lab', '-v-luv', '-b-rgb', '-v-hsv', '-YCrCb', '-hls', 'yuv', 'Cr-YCrCb', 'y-yuv'].*

3.3 Selection of best tuples (resized, padding)

With a clearer idea of what the group of transformations that form the final configuration of the system could be, we began with the selection of the rest of the parameters. Although one could simply find the tuple of parameters (resized, padding) that obtained the highest percentage in the previous Top 15 of transformations, which gave close to 90% hits; since such an algorithm was applied to the total datasets, it is necessary to ensure that the results are not biased and carry out more general tests before confirming them.

For this, it was decided to re-compute the same search, from the minimum set of best transformations, but using as a platform a modification of the "K-fold cross-validation" technique. This was done through the implementation of 5 repetitions of *10-fold cross-validation*, where in each cross validation, $\frac{9}{10}$ random groups of datasets were taken and calculated the best sets of 3, 5, 7, 10, 13, and 15 transformations. Once obtained, each one was used to find the best 2 tuples (resized, padding) that the highest percentages of positive detections would grant to the other $\frac{1}{10}$ set of datasets. In each of the K=10 iterations, both the mean probabilities of each tuple and the frequency of appearance of the transformations that made up each of the best sets were accumulated. Tables (1) and (2) summarize such analyzes.

3.4 Bus aspect ratio

Based on the front dimensions of the buses, and given that 100% of them have their line number on the windshield; the selection criterion was taken to discard any detection made with an aspect ratio that is not identified with 'front' or 'slightly lateralized' approximations. In search of such relationships and using

our population sample as a reference, the mean, median, mode and standard deviation estimators were evaluated, over the short, medium and long distance, in relation to the location of the bus at the time of capture.

Based on the results obtained, Table (3), it was concluded that: let $w \times h$ be the respective width and height of a given detection, a detected bus would be a good candidate, if its ratio of aspect falls in the value range $(ratio - delta) \leq \frac{w}{h} \leq (ratio + delta)$, where $ratio = 0.92$ and $delta = 0.24$; product of the *mode* at medium distance and double its standard deviation to make such filtering more flexible.

4 Bus detection: Model selection

So far, various parameter settings have been analyzed and tested with the intention of enhancing the last part of our pipeline. Now, through precision checks and speed tests, the "theoretical and precise" object detection model used so far will be replaced, and the entire system will be tested.

The following analyzes were performed using a Laptop Toshiba Satelite "Intel(R) Core(TM) i7-3632QM, CPU 2.20GHz; GPU AMD Thames XT-M2 1 Gb DDR3". Each one of them was performed on three object detection models at different input dimensions of their network; The following settings will be taken from now on more than nine (9) models distinguishable from each other: YOLO v2 (608x608, 416x416, 352x352 and 320x320), YOLO v3 Tiny (608x608, 416x416, 352x352 and 320x320), and SSD-MobileNet (300x300).

Table 1. Better sets of transformations and frequencies of apparition. Ref: ($*^1$ = -Cb-YCrCb, $*^2$ = Cr-YCrC).

	Top 3		**Top 5**		**Top 7**		**Top 10**		**Top 13**		**Top 15**	
Rank	Trans.	Freq.	Trans.	Freq.	Trans.	Freq.	Trans.	Freq.	Trans.	Freq.	Trans.	Freq.
1	v-luv	48	v-luv	50	v-luv	50	-s-hls	50	hls	50	hls	50
2	$*^1$	33	-u-yuv	50	$*^1$	49	v-luv	50	-s-hls	50	-s-hls	50
3	-g-rgb	27	$*^1$	40	-u-yuv	42	$*^1$	49	v-luv	50	v-luv	50
4			-s-hls	30	-s-hls	38	hls	47	-v-luv	50	-v-luv	50
5			-g-rgb	24	hls	33	-b-rgb	44	-b-rgb	49	-b-rgb	49
6					-g-rgb	32	-v-luv	44	$*^1$	48	$*^1$	49
7					-v-luv	24	-u-yuv	43	-u-yuv	42	-hls	48
8							b-lab	38	-hls	41	-u-yuv	42
9							-yuv	29	b-lab	37	$*^2$	38
10							-g-rgb	28	-yuv	34	b-lab	37
11									$*^2$	30	-g-rgb	26
12									-g-rgb	25	-yuv	26
13									-v-hsv	23	y-yuv	25
14											-v-hsv	22
15											yuv	19

Table 2. Better tuple combinations (resized, padding), for each set of better transformations.

	Top 3				**Top 5**			
Rank	Resized	Padding	Freq.	Detections	Resized	Padding	Freq.	Detections
1	96	0	23	20%	96	0	21	23,33%
2	128	10	22	28,18%	128	15	18	29,44%
	Top 7				**Top 10**			
Rank	Resized	Padding	Freq.	Detections	Resized	Padding	Freq.	Detections
1	128	0	23	30,43%	128	0	25	42,8%
2	96	0	22	29,55%	160	0	20	43,0%
	Top 13				**Top 15**			
Rank	Resized	Padding	Freq.	Detections	Resized	Padding	Freq.	Detections
1	128	0	27	42,96%	**128**	**0**	**23**	**61,9%**
2	96	5	19	42,63%	**160**	**5**	**21**	**45,78%**

4.1 Detection accuracy

The accuracies in the detections provided by the models were examined, in relation to the buses of our population sample, by calculating IoU indices between: the bounding boxes delivered by them and the ROIs annotated manually (explained in Section (3)). All of them were tested in two categories: short and medium distance, based on the location of the bus at the time of the capture. Regarding the long distance, and without losing sight of our objective, it was concluded that it would be of little use to analyze far buses, since their line number is not visible from such location. In all cases, the parameters configured for each model were: (*confidence*: 0.5, *IoU*: 0.45, *aspect ratio allowed*: all).

Although this test does not seem very useful, it allowed us to know, to a certain extent, the maximum and minimum useful working distances of each model, very important information when selecting the best one. Table (4) shows such results, in addition to the percentage of total detections (*Total Det*) made by each variant.

4.2 Early detections vs. fast detections

At this stage, the trade-off between models with "early detection" and "fast detection" capabilities was analyzed. That is, based on the final percentage of

Table 3. Aspect ratio ($\frac{width}{height}$) of buses vs. distance capture.

	Estimators			
Distance	Mean	Median	Mode	Std
Corta	0,98	0,97	0,9	0,11
Media	1,03	1,05	0,92	0,12
Larga	1,04	1,05	1,06	0,12

performance, the convenience of using fast and not so precise models, of those slower but with greater sensitivity, was studied.

Unlike previous tests, and to get closer to a more real situation, they were tested on 22 new test sets; each one, made up of a series of frames (from 4 to 32), corresponding to the same objective 'bus' arriving at a stop. For each of the models tested, the following were observed: mean of detection rates (*Vel.*), mean of positive detections (*Det+*), mean of frames processed up to 1^{er} Det+ (*F.Proc.*), and mean of the certainty value of such Det+ (*Certainty*). All this, using the best two tuples (128, 0, Top 15) and (160, 5, Top 15) Section (3.3), filtering the aspect ratios of the buses to 0,92 ± 0,24 Section (3.4) and keeping the internal adjustments of such models in: (*confidence*: 0.5, IoU: 0.45). Table (5) and Figure (6), details and shows the summary of this executions.

Fig. 6. Sample of execution of the complete system over one of the test sets.

5 Analysis of results and final simulation

Regarding the accuracies obtained, Table(4), it can be seen that in the models derived from YOLO v2 and YOLO v3 Tiny, the decrease in the input dimension to their networks, is directly proportional to the quantity of detections made. For its part, SSD-MobileNet, although it might seem that it did not compete with its adversaries, both in short and medium distance, obtained almost the

Table 4. Model accuracy vs. detection distance.

	Short distance								
	Yolo v2				**Yolo v3 Tiny**				**SSD M.Net**
≥IoU	608	416	352	320	608	416	352	320	300
.65	95,65%	95,65%	91,3%	95,65%	73,91%	82,61%	86,96%	78,26%	78,26%
.85	78,26%	73,91%	73,91%	69,57%	43,48%	30,43%	39,13%	26,09%	56,52%
Total Det	100,0%	95,65%	91,3%	95,65%	73,91%	82,61%	86,96%	78,26%	78,26%
	Medium distance								
	Yolo v2				**Yolo v3 Tiny**				**SSD M.Net**
≥IoU	608	416	352	320	608	416	352	320	300
.65	98,08%	100,0%	98,08%	96,15%	92,31%	76,92%	63,46%	40,38%	73,08%
.85	73,08%	55,77%	36,54%	34,62%	48,08%	23,08%	21,15%	13,46%	46,15%
Total Det	100,0%	100,0%	98,08%	100,0%	92,31%	80,77%	75,0%	55,77%	73,08%

Table 5. Summary of execution of the system for each object detection model, using the better two settings *(resized, padding, transformations)*, Subsection (3.3).

	(128, 0, Top 15)				**(160, 5, Top 15)**			
Model	Det+	Vel.	F.Proc.	Certainty	Det+	Vel.	F.Proc.	Certainty
YOLOv2 608	**81%**	**33,4s**	**4,7**	**70,8%**	63%	30,2s	4,3	57,4%
YOLOv2 416	68%	25,3s	6,5	66,4%	**81%**	**21,5s**	**5,5**	**67,5%**
YOLOv2 352	**72%**	**18,8s**	**6,6**	**68,9%**	63%	19s	5,8	59,7%
YOLOv2 320	**72%**	**15,3s**	**9,3**	**71,6%**	68%	16,5s	5,9	66,3%
YOLOv3 tiny 608	**68%**	**14,2s**	**5,6**	**70,9%**	68%	13,9s	4,9	66,9%
YOLOv3 tiny 416	63%	12,3s	8,3	47,6%	54%	11,2s	7,3	68,6%
YOLOv3 tiny 352	40%	10,3s	9,9	60,4%	36%	11,8s	10,1	64,6%
YOLOv3 tiny 320	50%	8s	9	63,1%	50%	9,5s	9,2	61%
SSD-Mobilenet 300	**72%**	**5s**	**6,2**	**71,1%**	59%	4,8s	5,5	61,1%

same percentages of detections as the YOLO v3 Tiny variants, granting 78% and 73%, in comparison with 79% and 75%, which on average obtained the variants of the latter.

Regarding the results obtained in Table (5), and taking into account that a bus, from its approach, until its return to its route, takes between 5 and 20 seconds. Although the variants of YOLO v2 (608 and 416) have delivered the best percentages of positive detections, given their average recognition speeds, they would have a great chance of missing the bus. Consequently, the model to be selected must have a good balance between speed and precision in the medium distance.

For all the above, it was decided to SSD-MobileNet 300 as the definitive model, using the configuration (128, 0, Top 15). This choice was based on the 'very fast' (around 5 seconds) execution times and an acceptable percentage of recorded hits, which, by allowing a higher frequency of executions, would greatly facilitate the possibilities of recognition. With a maximum probability of 62% in a single image; over an image sequence, the final system, was able to correctly recognize the bus line in the 72% of the cases tested, correctly recognizing the number of the line, on average, before the 7^{mo} frame analyzed, and giving confidence in their predictions that range from 71%.

Finally, making use of the parameters and models previously determined, it was decided to emulate a situation closer to reality by adapting the algorithm, so that, through a webcam, it could retrieve images in real time instead of being loaded from disk. At 5 fps and a resolution of 640x480 pixels, the camera was positioned in front of a television, in which, trying to recreate a 'virtual bus stop', a video of a bus arriving to a bus stop was played. From a basal approach, where the line recognition module was triggered the instant a bus is detected, to a bounded buffer producer-consumer, it was the considered implementations during this test. In it, a particular line, with 60% certainty, in 3.18 seconds, was recognized using the producer-consumer strategy with k=3.

6 Conclusion

In this investigation, an attempt was made to exploit the advances in the field of artificial vision, to provide a tool to assist people with visual impairments. Using object detection models and OCR techniques, a system was built capable of automating the task of recognizing bus line numbers. Judging by the results obtained and despite the great limitation to perform detections in dimly lit scenes, the final system was above the expectations was able to correctly recognize the bus line in the 72% of the cases tested, with an confidence range from 71%.[13]

In order to improve the system, the application of new strategies to reduce the set of image transformations used, the adaptation of the algorithm to GPU, and the incorporation of recent and more fast object detection models, could provide significant speed increases and recognition.

References

1. Brown, D., et al.: Seeing with sound? exploring different characteristics of a visual-to-auditory sensory substitution device. Perception **40**, 1120–35 (2011). https://doi.org/10.1068/p6952
2. Garcia-Lamont, F., et al.: Segmentation of images by color features: A survey. Neurocomputing **292**, 1 – 27 (2018). https://doi.org/10.1016/j.neucom.2018.01.091
3. Howard, A., et al.: Mobilenets: Efficient convolutional neural networks for mobile vision applications (2017)
4. Jabnoun, H., et al.: Object recognition for blind people based on features extraction. In: International Image Processing, Applications and Systems Conference. pp. 1–6 (2014)
5. Jiang, R., et al.: Let blind people see: Real-time visual recognition with results converted to 3d audio (03 2016)
6. LearnOpenCV: Color spaces in opencv. `https://www.learnopencv.com/color-spaces-in-opencv-cpp-python/`
7. Liu, W., et al.: Ssd: Single shot multibox detector. vol. 9905, pp. 21–37 (2016)
8. Long, S., et al.: Scene text detection and recognition: The deep learning era. ArXiv **abs/1811.04256** (2018)
9. Modi, H., Parikh, M.: A review on optical character recognition techniques. International Journal of Computer Applications **160**, 20–24 (2017)
10. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement (2018)
11. Riazi, A., et al.: Outdoor difficulties experienced by a group of visually impaired iranian people. Journal of Current Ophthalmology **28**(2), 85–90 (2016)
12. Russell, S.J.: Artificial intelligence: a modern approach, vol. 1. Prentice Hall, 4 edn. (2020)
13. Shaifee, M., et al.: Fast yolo: A fast you only look once system for real-time embedded object detection in video. Journal of Computational Vision and Imaging Systems **3**(1) (2017). https://doi.org/10.15353/vsnl.v3i1.171
14. Zhou, X., et al.: East: An efficient and accurate scene text detector. IEEE Conference on Computer Vision and Pattern Recognition (2017). https://doi.org/10.1109/cvpr.2017.283

[13] The final result can be observed in YouTube. See, `https://youtu.be/DeLpJ9ud7p4`