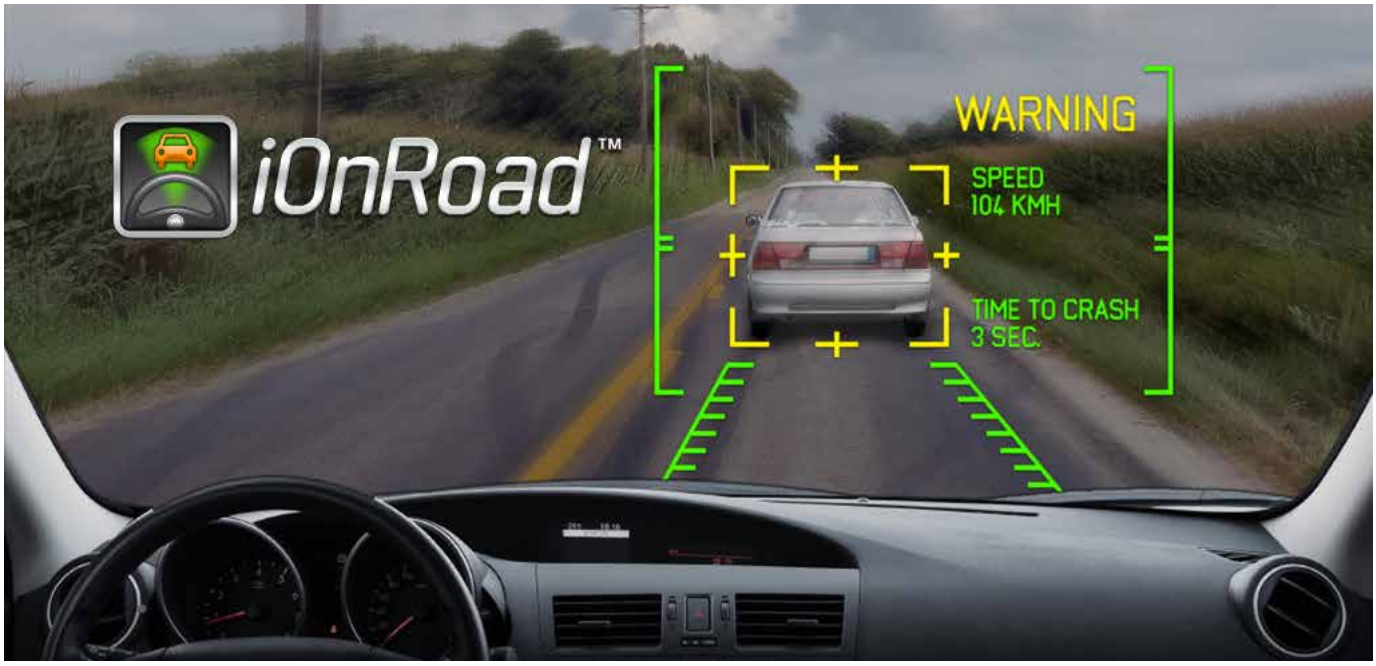
Clasificación de Obstáculos a Través del Algoritmo SDD

Francisco Jacob Ávila Camacho[1]
Leonardo Moreno Villalba[2]
Adolfo Meléndez Ramírez[3]
Juan Manuel Stein Carrillo[4]

RESUMEN

El presente trabajo muestra los resultados de la implementación del algoritmo SSD (Single Shot Detector) en la clasificación de obstáculos, no sólo la detección de objetos, para ser implementado en sistemas autónomos de navegación o en los lentes para invidentes que permitan identificar objetos que se encuentran fijos o en movimiento y representan un riesgo para la navegación. Se llevaron a cabo varios experimentos y se logró la clasificación de 89.99% de los objetos en tiempo real, con lo que se demuestra que el algoritmo puede ser implementado en aplicaciones de navegación con una precisión bastante aceptable, para evitar la colisión con obstáculos en sistemas de navegación autónoma.



PALABRAS CLAVE: Inteligencia artificial, clasificación de objetos, personas invidentes, sensores de movimiento, herramientas de apoyo.

Acerca de los autores...

[1,2,3,4] Docente de la División de Ingeniería en Sistemas Computacionales, Tecnológico de Estudios Superiores de Ecatepec.

1fjacobavila@tese.edu.mx, 2leonardo.moreno@tese.edu.mx, 3adolfo.melendez@tese.edu.mx, 4jmsteinc@tese.edu.mx

INTRODUCCIÓN

La clasificación de obstáculos como apoyo a la navegación autónoma o a la implementación de lentes para invidentes permite identificar objetos en movimiento o fijos que pudieran representar un peligro para la persona o el dispositivo o equipo en navegación. Dicha clasificación permite conocer el tipo de obstáculo que se presenta, con el fin de tomar las acciones adecuadas para evitar una posible colisión. La clasificación puede ser implementada a través de la detección de objetos utilizando visión artificial, sin embargo, en la mayoría de los casos, los algoritmos aplicados para la detección de objetos se basan en el esquema de cuadros delimitadores, lo que requiere de un procesamiento más intenso a la hora del entrenamiento [1].

En este trabajo se propone el uso del algoritmo SSD (Single Shot Detector) que ha demostrado ser eficiente, rápido y con una alta precisión, a diferencia de otros algoritmos de visión por computadora basados en el esquema de cajas delimitadoras, que ejecutan un muestreo consecutivo para cada caja, generando un proceso lento para aplicaciones en tiempo real a pesar de utilizar redes neuronales convolucionales recurrentes y rápidas Fast-RCNN. El algoritmo SSD se basa en el esquema de redes profundas que no realiza un muestreo doble de pixeles para las cajas delimitadoras, lo que resulta en una mejora significativa tanto en velocidad como en precisión. La mejora en velocidad proviene de la eliminación del esquema de cajas delimitadoras y del subsecuente estado de muestreo [2].

La propuesta del algoritmo SSD incluye el uso de un filtro convolucional pequeño para predecir categorías de objetos y compensaciones en la ubicación de las cajas delimitadoras; posteriormente, se aplican predictores separados para la detección de diferentes detalles relacionados y estos filtros se aplican a múltiples características en los estados sucesivos de la red para realizar una detección de múltiples escalas.

En este sentido, se puede lograr una alta precisión utilizando entradas de baja resolución, con lo que se incrementa la velocidad de detección.

1. ANTECEDENTES

En [3] se presenta un detector de rostros en tiempo real, adicionalmente Chencheng Ning y sus colaboradores en [4] proponen unos cambios al algoritmo SSD para aumentar la precisión en la clasificación sin afectar la velocidad, a través de la incorporación de una caja de inicio que reemplaza a las capas adicionales que utiliza originalmente el algoritmo; de esta forma, la red puede captar más información sin aumentar la complejidad.

En el campo de la detección de texto en imágenes, el algoritmo SSD también ha sido utilizado exitosamente como el algoritmo Single Shot Text Detector en [5], donde los autores crean cajas delimitadoras a nivel de palabras, utilizando un mecanismo de atención que identifica regiones de texto a través de un mapa de aprendizaje automático, eliminando la interferencia que pudiera generar el fondo de la imagen en las características convolucionales, lo que representa la clave para la inferencia precisa en las palabras detectadas.

En cuanto al video en tiempo real, los autores de [6] implementaron el algoritmo SSD como un Temporal Action Detector, que permite identificar objetos en videos largos de aplicaciones reales que contienen múltiples acciones; el reto no sólo es reconocer las categorías de acciones, sino

también detectar el tiempo en que inician y en que terminan, por lo que su propuesta se basa en una capa convolucional de una dimensión, para detectar las instancias de acciones en videos en tiempo real.

Por otro lado, Song, Qi, Qian y Feng [7] realizaron una demostración para imágenes de alta resolución en tiempo real, con una especificación de hasta 1Qm de resolución a una velocidad de captura der 1 linea/50ns.

El Lindsay Russel Pathsound es un dispositivo detector de obstáculos, que informa al usuario de la presencia de éstos mediante sonidos acústicos y sistema táctil. El dispositivo tiene tres tipos de sonido que representan tres distancias diferentes en dos zonas: la zona dentro del peligro y la zona fuera del peligro. La distancia máxima detectada es de seis pies. A pesar de sus objetivos, el dispositivo aún no ha llegado a ser comercial, ya que su diseño es muy voluminoso. El Mowat Sonar Sensor tiene una salida sonora y una táctil. El dispositivo emite un ultrasonido de forma cónica y elíptica de 15º de ancho y 30º de alto, cubriendo un área equivalente a la forma humana [13]. Su rango en distancia es de uno a cuatro metros. El dispositivo de ultrasonido más destacado es Sonicguide, también conocido como Binaural Sonar Electronic Travel Aid. El dispositivo está implementado en unas gafas y tiene como salida un sistema estereofónico. El Sonicguide emite una señal sónica de forma cónica que abarca desde 45º a la izquierda hasta 45º a la derecha, con un rango de hasta cuatro metros de distancia respecto a la línea central de visión del usuario. [14]. Según la variación del tono, el usuario es capaz de percibir y reconocer las distancias[15].

Entre los dispositivos de sensores o láser, los más destacados son los *Laser Cane*, *Talking Light*, *Pilot Light*, *SONA*; *Easy Walker*, etcétera.

Laser Cane o Light Amplification by Simulated Emission of Radiation representa la integración del bastón con la tecnología láser[16].

El dispositivo utiliza el láser de galio-arsénico así como pequeñas baterías. El rango de detección es de hasta cuatro metros e informa al usuario de la presencia de los obstáculos mediante señales acústicas[17].

Este trabajo está organizado en dos secciones, en la primera se describen los elementos necesarios para la construcción del prototipo, y en la segunda, el desarrollo del modelo de clasificación y comunicación para los obstáculos fijos y en movimiento, junto con las pruebas realizadas; finalmente, se presentan los resultados, las conclusiones y los trabajos futuros.

2. DETECCIÓN DE OBSTÁCULOS Y PRE-PROCESAMIENTO DE LA SEÑAL

Para el desarrollo del diseño conceptual del prototipo, se utilizó la metodología QFD (por sus siglas en inglés *Quality Function Deployment)*, con la cual se puede llegar a un diseño funcional y competitivo, señalando que éste puede ser mejorado en la etapa de diseño a detalle.

2.1 Descripción de las funcionalidades de la solución

En la Figura 1 se puede apreciar el árbol de funciones del sistema que se va a diseñar, en donde la función global es la detección de señales analógicas y consta de cuatro sub-funciones, las cuales deben de realizarse para poder cumplir con la función global y los requerimientos establecidos por el usuario.

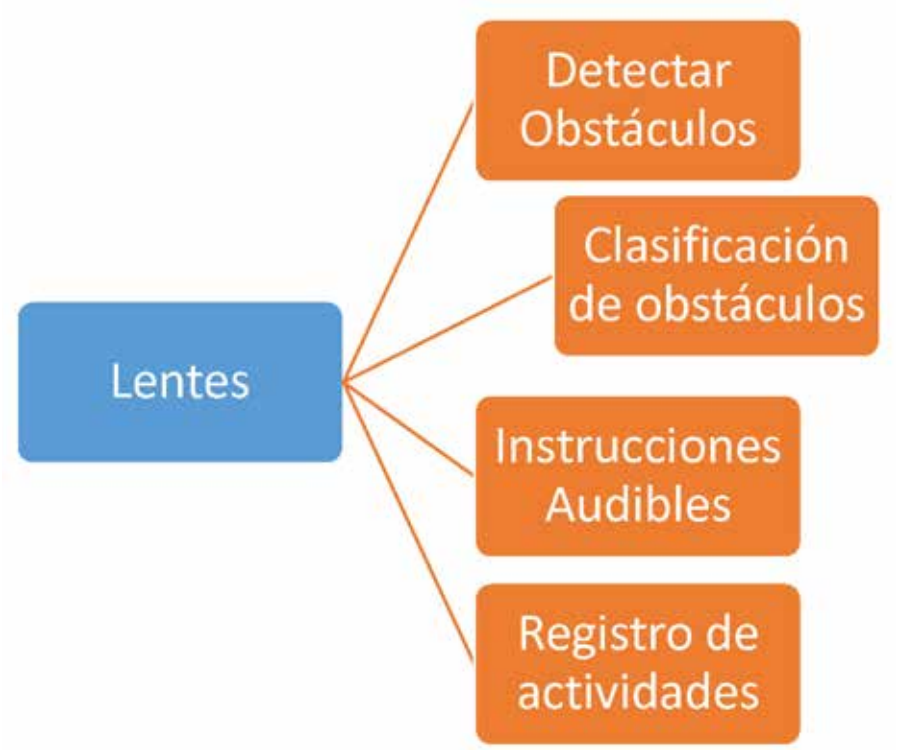


Figura 1

Funcionamiento del prototipo de lentas para la clasificación de objetos

2.2 Sensor de proximidad

Los sensores ultrasónicos tienen como objetivo primordial la detección de objetos a través de la emisión y reflexión de ondas de audio (acústicas). Para ello, emiten un pulso ultrasónico contra el objeto a medir o divisar, y al detectar el sensor al de pulso reflejado, se para un contador de tiempo, el cual inició su conteo al emitir el pulso. Este tiempo es referido a distancia y de acuerdo con los parámetros elegidos de respuesta, se envía una señal eléctrica digital o analógica.

Material necesario: Tarjeta Raspberry Pi, Cable USB para Arduino, Placa soldada, Software IDE de Arduino, Sensor HC – SR04, LED común de cualquier color, Resistencia de 220 ó 330 ohms, zumbador o speaker para 5v Cables. (Fig. 2)

Descripción

I. Se conecta el sensor HC-SR04 a la placa.
II. El sensor se conecta de la siguiente forma.
   1. El pin GND se conecta a tierra.
   2. El pin Vcc se conecta a la corriente de alimentación.
   3. El pin Trig se conecta al pin 4 digital de Arduino.
   4. El pin Echo se conecta al pin 5 digital de Arduino.

Ocupando este sensor en los lentes, al posicionar un objeto en el rango del sensor, el speaker emite un sonido y el LED comienza a parpadear; entre más cerca se éste del sensor, el ruido y el parpadeo serán más rápidos.

Material necesario: Control infrarrojo Raspberry Pi, Cable ploc (tiene que ser estéreo), bocina con entrada de 3.5 milímetros.

Raspberri Pi 2

Poder

El Raspberry Pi 2 puede alimentarse mediante la conexión USB Mini-B, a una fuente de alimentación conmutada de 5V y 2500mA.

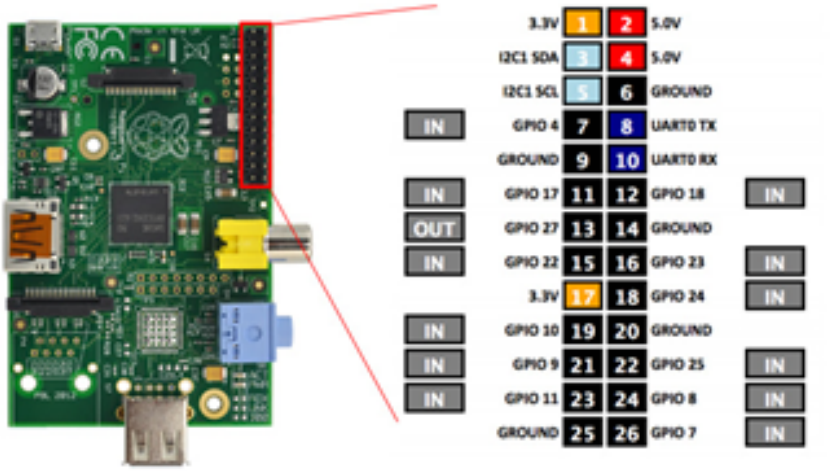
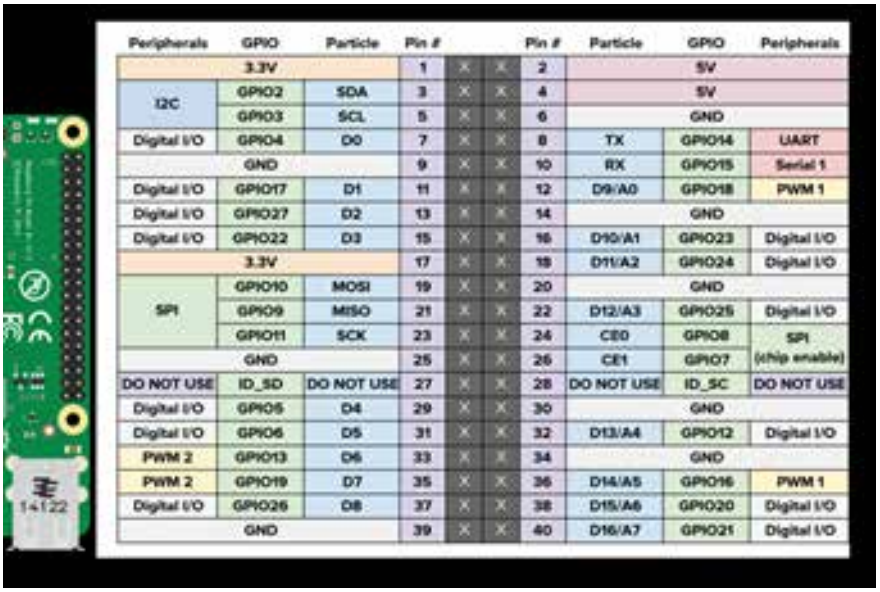


Figura 2

Diagrama de puertos de Raspberry Pi 2.



Peripherals	GPIO	Particle	Pin #			Pin #	Particle	GPIO	Peripherals
	3.3V		1	X	X	2		5V	
I2C	GPIO2	SDA	3	X	X	4		5V	
	GPIO3	SCL	5	X	X	6		GND	
Digital I/O	GPIO4	D0	7	X	X	8	TX	GPIO14	UART
	GND		9	X	X	10	RX	GPIO15	Serial 1
Digital I/O	GPIO17	D1	11	X	X	12	D9/A0	GPIO18	PWM 1
Digital I/O	GPIO27	D2	13	X	X	14		GND	
Digital I/O	GPIO22	D3	15	X	X	16	D10/A1	GPIO23	Digital I/O
	3.3V		17	X	X	18	D11/A2	GPIO24	Digital I/O
SPI	GPIO10	MOSI	19	X	X	20		GND	
	GPIO9	MISO	21	X	X	22	D12/A3	GPIO25	Digital I/O
	GPIO11	SCK	23	X	X	24	CE0	GPIO8	SPI (chip enable)
	GND		25	X	X	26	CE1	GPIO7	
DO NOT USE	ID_SD	DO NOT USE	27	X	X	28	DO NOT USE	ID_SC	DO NOT USE
Digital I/O	GPIO5	D4	29	X	X	30		GND	
Digital I/O	GPIO6	D5	31	X	X	32	D13/A4	GPIO12	Digital I/O
PWM 2	GPIO13	D6	33	X	X	34		GND	
PWM 2	GPIO19	D7	35	X	X	36	D14/A5	GPIO16	PWM 1
Digital I/O	GPIO26	D8	37	X	X	38	D15/A6	GPIO20	Digital I/O
	GND		39	X	X	40	D16/A7	GPIO21	Digital I/O

Figura 2

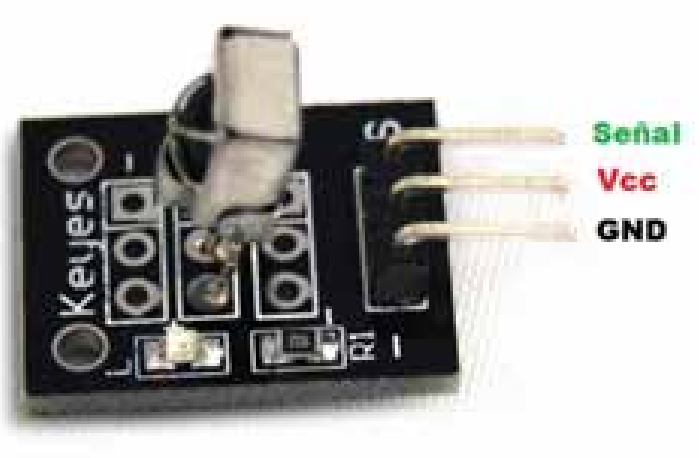
Descripción de puertos y salidas del Arduino Nano.

Control remoto IR

Un módulo AX-1838HS que incluye el receptor IR, resistencias pullup y LED que parpadea cuando un dato IR es recibido por un remoto. El módulo es individual y puede trabajar con controles IR o emisores IR.

Para utilizar el control remoto, es necesario saber que para cada botón existe un código, por lo tanto al momento que presionamos un botón, éste manda una serie de números que será recibida por el sensor IR.

El módulo receptor solo cuenta con tres pines de conexión: dos de alimentación y el otro es el que manda la señal que recibe. Para más información del sensor, revisar la hoja de datos.



Descripción

El AX-1838HS es un receptor infrarrojo miniaturizado para el control remoto y otras aplicaciones que requieren Mejorado el rechazo de la luz ambiente. El diodo PIN y el IC del preamplificador, están montados en un solo leadframe. El paquete epoxi contiene un filtro IR especial. Este módulo tiene un excelente Rendimiento, incluso en luz ambiente T y proporciona protección contra Impulsos de salida no controlados.

Se utilizó el módulo de cámara de 5 megapixeles, el cual está diseñado específicamente para Raspberry Pi, y cuenta con un lente de foco fijo, el cual es capaz de tomar imágenes estáticas de 2592 x 1944, y también es compatible con el formato de video 1080p30, 720p60 y 640x480p60/90.

Figura 4

Conexión del módulo de cámara de 5 mega pixeles a la tarjeta Raspberry Pi

Se eligió esta cámara debido a las características con las que cuenta: resolución de 1,4 Qm X 1,4 Qm píxeles, con tecnología OmniBSI de alto rendimiento (alta sensibilidad, baja diafonía, ruido bajo), un tamaño óptico de 1/4 y funciones de control de imagen automáticas.

3. CLASIFICACIÓN DE OBSTÁCULOS

El enfoque SSD funciona por medio una red convolucional feed-forward que produce una colección de tamaño fijo de cuadros delimitadores y puntajes para la presencia de instancias de objetos y sus deferentes clases en esos cuadros, seguidos por un paso de supresión no máxima para producir las detecciones finales[1].

En resumen, el framework SSD utiliza múltiples capas como clasificadores, donde cada mapa de características se evalúa mediante un conjunto de cajas predeterminadas de diferentes tamaños (relación de aspecto) y cada una en ubicación de forma convolucional, donde cada clasificador predice puntajes de clase y compensación de forma relativa a los cuadros[8].

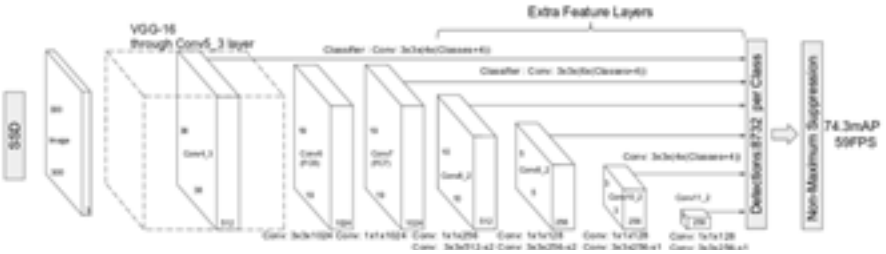


Figura 5

Algoritmo SSD, funcionamiento mediante Redes Convolucionales, Fuente[1]

Para la implementación del algoritmo de la detección de objetos se utilizó en lenguaje de programación Python, debido a su ágil y rápido manejo para aplicaciones de IA, en este caso se utilizó Pytorch como librería para la implementación de la red neuronal con el algoritmo SSD, como se observa a continuación; Pytorch es un framework de machine learning de Lua para Python, el cual, debido a que es orientado a objetos, es muy fácil de utilizar, teniendo soporte de procesamiento Cuda[9].

Pytorch está disponible como un paquete de Python, el cual puede ser instalado mediante Pip o Conda respectivamente[10], para nuestra aplicación se utilizó Conda, con la versión 3 de Python.

```
importtorch
fromtorch.autogradimport Variable
import cv2
from data importBaseTransform, VOC_CLASSES as labelmap
fromssdimportbuild_ssd
importimageio

defdetect(frame, net, transform):
height, width = frame.shape[:2]
frame_t = transform(frame)[0]
  x = torch.from_numpy(frame_t).permute(2, 0, 1)
  x = Variable(x.unsqueeze(0))
  y = net(x)
detections = y.data
scale = torch.Tensor([width, height, width, height])

for i in range(detections.size(1)):
    j = 0
whiledetections[0, i, j, 0] >= 0.6:
      pt = (detections[0, i, j, 1:] * scale).numpy()
      cv2.rectangle(frame, (int(pt[0]), int(pt[1])), (int(pt[2]), int(pt[3])), (255, 0,
0), 2)
      cv2.putText(frame, labelmap[i - 1], (int(pt[0]), int(pt[1])), cv2.FONT_
HERSHEY_SIMPLEX, 2, (255, 255, 255), 2, cv2.LINE_AA)
      j += 1
returnframe

# Creacion de la red neuronal con SSD
net = build_ssd('test')
net.load_state_dict(torch.load('ssd300_mAP_77.43_v2.pth', map_location =
lambda storage, loc: storage))

# Creacion de las transformaciones
transform = BaseTransform(net.size, (104/256.0, 117/256.0, 123/256.0))
```

Código 1

Aplicación del SSD para la detección de objetos usando la librería Pytorch en el lenguaje Python.

El código mostrado permite la implementación del algoritmo SSD para la clasificación de objetos para todo tipo de fuente de video, ya sea un video previamente grabado, una foto o por medio de una llamada a la cámara de video, como se muestra en el siguiente segmento de código.

```
video_capture = cv2.VideoCapture(0)
while True:
  _, frame = video_capture.read()
  gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
canvas = detect(gray, frame)
  cv2.imshow('Video', canvas)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
video_capture.release()
cv2.destroyAllWindows()
```

Código 2

Segmento de código para poder mandar a llamar a la cámara del prototipo en Raspberri

4. Resultados

Después de realizar diferentes pruebas, se obtuvieron resultados muy satisfactorios, pues el prototipo pudo detectar correctamente la mayor

parte de los objetos que estaban frente a él, lo cual puede ser observado en la Figura 6.

Figura 6

Detección de objetos con el prototipo.

Como se puede ver claramente en la imagen, no se reconocieron todos los objetos que eran capturados por la cámara de video del prototipo, esto debido a diferentes factores, el tamaño, la distancia, o la cercanía a otro objeto que llegara a confundir al algoritmo.

De aquí parte la necesidad de implementar algunos otros métodos y algoritmos para la detección de objetos, que mejoren su rendimiento y asertividad. Actualmente el equipo de investigación está trabajando con los algoritmos YOLO3, y con Redes Generativas Adversarias (GANs). El algoritmo YOLO3 permite reconocer los objetos mediante un puntaje de asertividad para cada cuadro delimitador, usando regresiones logísticas[11], esto mejora bastante el rendimiento para el reconocimiento de objetos, lo cual puede ser observado en la Figura 7, que a continuación se muestra.

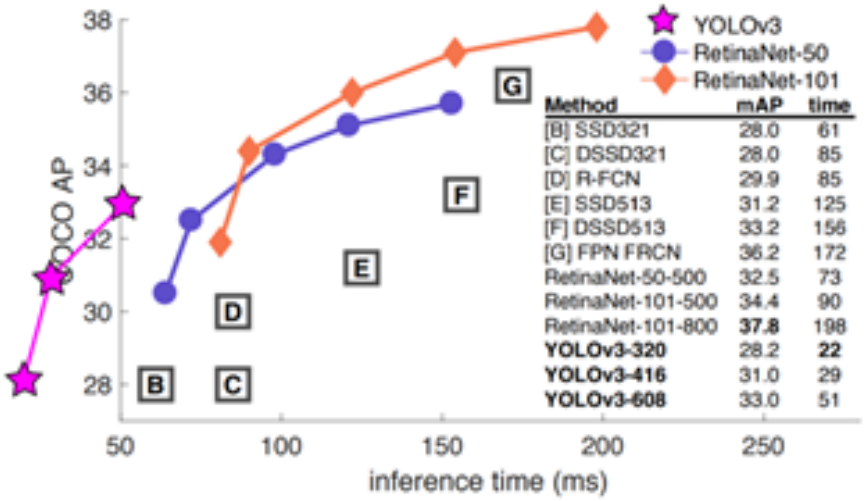


Figura 6

Gráfica de rendimiento del algoritmo YOLO3 comparado con otros métodos[11]

Por otro lado, mediante el uso de GANs podemos enfocarnos en la detección de objetos más pequeños y con cámaras de menor calidad, esto debido a que el método se enfoca en la representación de pequeños objetos y sus similitudes con objetos más grandes, esto a través de correlaciones que son realizadas durante el proceso de entrenamiento[12], lo que nos da una gran área de oportunidad para mejorar el modelo propuesto.

BIOGRAFÍAS

Dr. Francisco Jacob Ávila Camacho. Nació en Puebla en 1967. Obtuvo el título de ingeniero en Electrónica por la Universidad Autónoma Metropolitana, Unidad Azcapotzalco. D.F., México, en el año de 1990; obtuvo el grado de Maestría en Ingeniería en Sistemas Computacionales en el Tecnológico de Estudios Superiores de Ecatepec, Estado de México, en el año 2010, y el Doctorado en Sistemas Computacionales por parte de la Universidad Da Vinci, en México, D.F., en el año 2015. Del 2003 a la fecha se ha desempeñado como profesor investigador en la División de Ingeniería en Sistemas Computacionales en el Tecnológico de Estudios Superiores de Ecatepec; ha impartido conferencias y publicado en memorias de congresos nacionales e internacionales, así como en revistas indexadas y arbitradas. Actualmente está dirigiendo proyectos de investigación financiados por el Tecnológico Nacional de México; sus líneas de investigación incluyen inteligencia artificial, reconocimiento de patrones y algoritmos difusos, minería de datos, internet de las cosas, sistemas distribuidos, sistemas embebidos, braincomputer interface. Es socio fundador de la empresa Ihualia Software, S.A. de C.V., mantiene dos registros en el Instituto Nacional del Derecho de Autor por el software desarrollado para la empresa miembro activo de la Sociedad Mexicana de Tecnologías de la Información, Mecatrónica y Telemática, A.C.

M. en I.S.C. Leonardo M. Moreno Villalba. Es Profesor investigador del Tecnológico de Estudios Superiores de Ecatepec, adscrito a la División de Informática. Es Ingeniero en Sistemas Computacionales, Maestro en Ingeniería en Sistemas, y doctorante en Cómputo. Ha impartido conferencias y publicado en memorias de congresos nacionales e internacionales, así como en revistas indexadas y arbitradas. Actualmente está dirigiendo proyectos de investigación financiados por el Tecnológico Nacional de México; sus líneas de investigación incluyen inteligencia artificial, machine learning, aprendizaje profundo, procesamiento natural del lenguaje y visión artificial, miembro del Institute of Research Engineers and Doctors, y de la International Association of Engineers. Es asesor en Tecnologías de la información del Instituto Nacional Electoral y de la Asamblea Legislativa del Distrito Federal en la VII Legislatura; actualmente se encuentra dirigiendo proyectos de investigación para el Centro de Cooperación Academia-Industria del TESE.

Dr. Adolfo Meléndez Ramírez. Es miembro (M) de la IEEE desde el 2015 y es miembro de ACM desde 2013. Nació en Tlaxcala en 1965. Obtuvo el título de ingeniero en comunicaciones y electrónica en el Instituto Politécnico Nacional, Unidad Zacatenco. D.F., México en el año 1990. Obtuvo el grado de la Maestría en Administración y Desarrollo de negocios de la Universidad del Valle de Toluca en el 2011; obtuvo el grado de Doctor en Sistemas Computacionales por la Universidad Da Vinci en 2015. Del 2003 a la fecha se ha desempeñado como profesor investigador en la División de Ingeniería en Sistemas Computacionales en el Tecnológico de Estudios Superiores de Ecatepec, ha impartido conferencias y publicado en memorias de congresos nacionales e internacionales, y actualmente está dirigiendo proyectos de investigación financiados por la DGEST y el Tecnológico Nacional de México. También cuenta con artículos en revisión en diversas publicaciones arbitradas e indexadas. Sus líneas de investigación incluyen procesamiento paralelo, inteligencia artificial, reconocimiento de patrones, seguridad informática, minería de datos, internet de las cosas, sistemas distribuidos, sistemas embebidos. Es socio fundador de la empresa Pringel, S.A. de C.V. El M. en I.S.C. Meléndez es miembro activo de la Sociedad Mexicana de Tecnologías de la Información, Mecatrónica y Telemática, A.C.

Ing. Juan Manuel Stein: Es Candidato a doctor por la Universidad Da Vinci; actualmente es profesor de tiempo completo en el Tecnológico de Estudios Superiores de Ecatepec; ha publicado dos artículos en revistas internacionales y sus líneas de investigación están enfocadas en interfaces humano-computadora, reconocimiento de patrones y otros.

Referencias

[1] W. Liu et al., "SSD: Single shot multibox detector," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2016, Vol. 9905 LNCS, pp. 21–37.

[2] D. G. Lowe, "Object recognition from local scale-invariant features," Proc. Seventh IEEE Int. Conf. Comput. Vis., Vol. 2, No. 8, pp. 1150–1157, 1999.

[3] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li, "S3FD: Single Shot Scale-Invariant Face Detector," in Proceedings of the IEEE International Conference on Computer Vision, 2017.

[4] C. Ning, H. Zhou, Y. Song, and J. Tang, "Inception Single Shot MultiBox Detector for object detection," in 2017 IEEE International Conference on Multimedia and Expo Workshops, ICMEW 2017, 2017.

[5] P. He, W. Huang, T. He, Q. Zhu, Y. Qiao, and X. Li, "Single Shot Text Detector with Regional Attention," in Proceedings of the IEEE International Conference on Computer Vision, 2017.

[6] T. Lin, X. Zhao, and Z. Shou, "Single Shot Temporal Action Detection," in Proceedings of the 2017 ACM on Multimedia Conference - MM '17, 2017.

[7] Q. Song, F. Qian, M. Akdas, J. Meyer, and O. Boyraz, "Single Detector Single Shot High Resolution Imaging," Biomed. Eng. (NY)., 2009.

[8] Y. Zhang, H. Peng, and P. Hu, "CS341 Final Report: Towards Real-time Detection and Camera Triggering," 2017.

[9] A. Paszke et al., "Automatic differentiation in PyTorch," Adv. Neural Inf. Process. Syst. 30, No. Nips, pp. 1–4, 2017.

[10] V. Subramanian, Deep learning with PyTorch : a practical approach to building neural network models using PyTorch.

[11] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," Apr. 2018.

[12] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, and S. Yan, "Perceptual Generative Adversarial Networks for Small Object Detection."