Detección de carril utilizando barrido de ventanas en imágenes.

Enrique Alemán [ab], Miguel López-Montiel [a], M.C. Yoshio Rubio [a], M.C. Adolfo Esquivel Martínez [ab]

[a] Instituto Politécnico Nacional, Centro de Investigación y Desarrollo de Tecnología Digital (CITEDI)
[b] CETYS Universidad Campus Tijuana

Resumen

La detección de carril es una tarea fundamental en el desarrollo de vehículos autónomos y sistemas de asistencia al conductor (*DAS*), su importancia radica en el hecho de que permite aumentar la seguridad durante la conducción dado el objetivo general de mantener al vehículo dentro de los límites del carril. En este documento se describe la implementación de visión por computadora para llevar a cabo la detección de carril mediante un algoritmo elaborado con filtros de binarización, cambios de perspectiva y algoritmo de análisis de imagen por segmentos. Se utiliza un filtro Gaussiano para reducir detalle en la imagen, cambio de perspectiva, cambio de espacio de color *HSV*, etc. Se implementan la técnica de procesamiento de imagen por segmentos (sliding windows) para facilitar la tarea de detección. Finalmente, se calcula el ángulo de desviación del centro de la cámara al carril a través de un promedio a tres puntos de referencia, para así tener una estimación del ángulo de giro apropiado de la ruedas frontales para un ajuste apropiado de la posición del vehículo para que este se encuentre en el centro. El ambiente en el que probará el algoritmo es una vía de dos carriles a escala 1:10, la imagen se obtendrá de una cámara montado a un vehículo robótico de configuración geométrica *Ackermann* a la misma escala para semejar una conducción de la vida real.

Palabras clave—Vehículos autónomos, detección de carril, visión por computadora, conducción automática.

Abstract

Lane detection is a fundamental task in the development of autonomous vehicles and driver assistance systems (DAS), its radical importance lies in the fact that it allows to increase safety during driving given the general objective of keeping the vehicle within lane boundaries. This document, describes the implementation of computer vision to perform lane detection using an algorithm developed with binarization filters, perspective changes and image analysis algorithm by segments. A Gaussian filter is used to reduce details in the image, change of perspective, change of HSV color space, etc. The image analysis by segments (sliding windows) is used to facilitate the detection task. Finally, the angle of deviation from the center of the chamber to the lane through an average of three reference points is calculated, in order to have a correct angle of rotation of the front wheels for an appropriate adjustment of the position of the vehicle so that it is in the center. The environment in which the algorithm will be tested is a two-lane road at 1:10 scale, the image will be seen from a camera mounted on a robotic vehicle of Ackermann geometric steering configuration at the same scale to resemble a real-life driving.

Keywords— *Autonomous vehicles, lane detection, Computer vision, ADAS*

1. INTRODUCCIÓN

1.1 Antecedentes

En la actualidad el desarrollo de vehículos inteligentes significa dar solución a diferentes problemas tales como exceso de tráfico, alto índice de accidentes automovilísticos y a la evolución de los vehículos autónomos. Se han desarrollado diferentes mecanismos que se pueden considerar inteligentes y cuyo propósito es promover la buena conducción del vehículo (con o sin intervención directa del sistema), algunos ejemplos son *ACC (Autonomous Cruise Control), CACC (Cooperative Adaptive Cruise Control), LDW (Lane Departure Warning), FCW (Forward Collision Warning)* [1], es importante considerar que a este tipo de mecanismos se les conoce como *DAS* (Sistemas de Asistencia al Conductor por sus siglas en ingles) o *ADAS* (Sistemas Avanzados de Asistencia al Conductor por sus siglas en ingles) en caso de que la implementación sea más compleja y robusta. Aun cuando estos mecanismos funcionan de manera independiente e inteligente, solo un mecanismo capaz de unir todas las capacidades mencionadas a tal grado que la intervención humana no fuese necesaria, permitiría el desarrollo de vehículos totalmente autónomos.

En este trabajo se elabora un algoritmo de detección, con el que se podrá determinar qué tan alejado se encuentra el centro del automóvil con respecto al centro del carril. Este tipo de sistemas se han estado desarrollando dado el hecho del gran impacto que pueden tener en el área de ADAS y de vehículos autónomos, las aplicaciones van de simplemente detectar carriles durante una conducción, así como predecir cambios de carril, des-alineación del vehículo con respecto al dentro del carril o cálculo de la curvatura para poder maniobrar correctamente [2], un ejemplo de este tipo de algoritmos se puede observar en la Figura 1 en donde se detectan las líneas que delimitan el carril y se marcan en rojo.

Figura 1. Ejemplo del resultado de un algoritmo de detección de carril básico.

Imagen recuperada de https://github.com/dmlicht

El problema de detección se pretende resolver por medio de algoritmos de visión de computadora basados en modelos y características (sistema híbrido). Gracias a muchos algoritmos y filtros disponibles listos para implementación, es relativamente sencillo acelerar el desarrollo del algoritmo, sin embargo, la señal procesada es una imagen que puede tener ruido por las variaciones de iluminación, carriles mal pintados, desgastados o descoloridos, lo cual puede representar un reto. Este tipo de condiciones no ideales son el principal obstáculo durante la implementación de algoritmos robustos para detección de líneas de carril [2].

Otro reto es la variabilidad de condiciones a las que un vehículo en movimiento se expone, por ejemplo, personas o animales, obstáculos, otros vehículos y variación del diseño de los carriles pintados en el suelo (zona rural o urbana) [6], por lo que se necesita de otros algoritmo que operen en paralelo para tener un mejor desempeño.

Algunos ejemplos, desarrollados por prestigiosas instituciones que han servido como base teórica para mejoras y diseño de nuevos algoritmos son [3] [6]:

- **GOLD**: Utiliza detección a base de bordes
- **RALPH**: Busca controlar la posición lateral del vehículo.
- **YARF**: Se basa en el modelo geométrico de la vía.
- **AURORA**: En este caso una cámara RGB apunta hacia el suelo para detectar el color de las líneas del carril.

Las Figuras 2 y 3, muestran los esquemas y resultados obtenidos de diferentes algoritmos orientados al mismo objetivo en [2] y [3].

Figura 2. Implementación de algoritmo de detección de línea

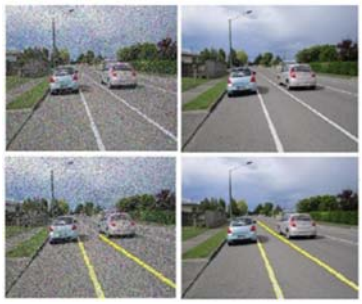


Imagen recuperada de Kaur, G., & Kumar, D. (2015). Lane detection techniques: A review. International Journal of Computer Applications, 112(10).

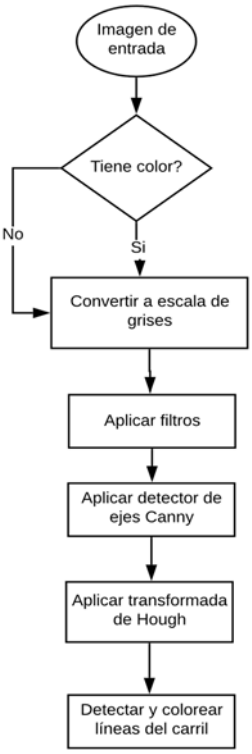


Figura 3. Implementación de algoritmo de detección de línea

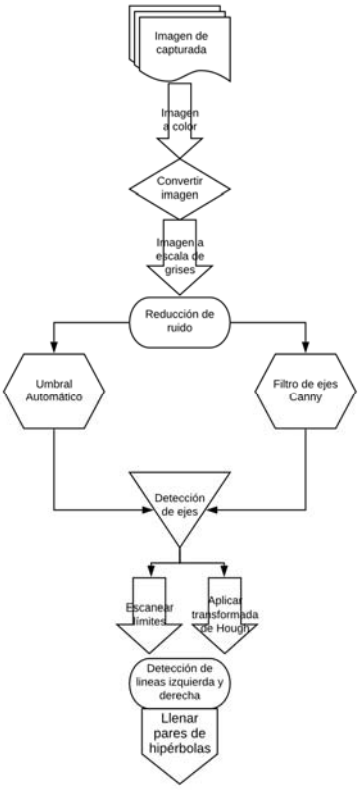


Imagen recuperada de A. A. Assidiq, O. O. Khalifa, M. R. Islam and S. Khan, "Real time lane detection for autonomous vehicles," 2008 International Conference on Computer and Communication Engineering, Kuala Lumpur, 2008, pp. 82-88. doi: 10.1109/ICCCE.2008.4580573

El problema de detección de carril se puede atacar por dos tipos de metodologías, una que se basa en detectar características y la otra se basa en modelos matemáticos que describen a los carriles. Cada una tiene ventajas y desventajas, por ejemplo, aquella basada en características depende de una señal ideal (imagen) ya que distintas condiciones como cambios de iluminación, carriles mal pintados, etc. pueden representar un problema para el procesamiento. La otra basada en modelos representa a los carriles como curvaturas modeladas matemáticamente y tienden a ser menos susceptibles al ruido de la señal, pero el modelo al no ser dinámico es difícil que este se adapte a todas las situaciones [2]. Para la implementación de nuestra propuesta se utiliza un algoritmo que, hasta cierto punto, hace uso de ambas metodologías para desarrollar uno adaptable a diferentes escenarios.

1.2 Barrido de ventanas

Para la detección de las líneas que definen el carril, se utiliza el algoritmo de barrido de ventanas. Este se caracteriza por ser computacionalmente costoso pero preciso al identificar cierta característica en una imagen. El algoritmo se define formalmente como se muestra en la ecuación (1)

$$R_{obj} = argmax_{R \subseteq I}\, f(R) \quad (1)$$

en donde *R* se extiende por todas las regiones de la imagen *I*.

El algoritmo de barrido de ventanas consiste en crear una región rectangular y desplazarla a lo largo de toda la imagen, con el propósito de analizar cada sección y así detectar alguna característica y al mismo tiempo conocer la localización (en la imagen) de cada característica. Normalmente se establece un tamaño fijo a la ventana a deslizar sobre la imagen. Entre más grande sea la ventana, menor es el costo de cómputo, pero menor es la resolución de la localización; por otra parte, entre más pequeña la ventana, mayor es el costo computacional pero mayor es la confiabilidad de la localización [7].

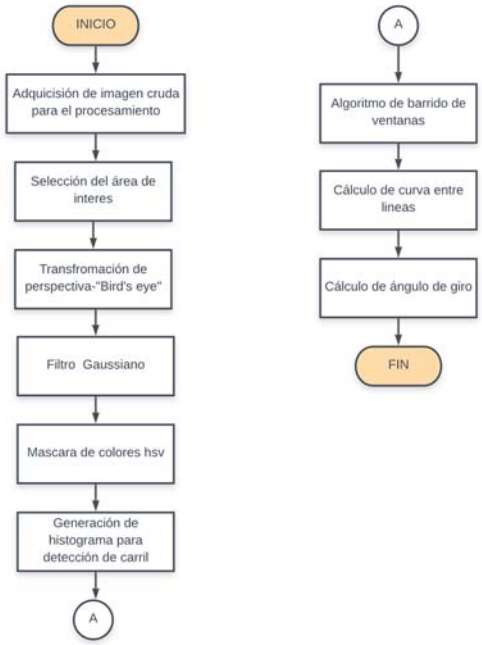
2. METODOLOGIA

La adquisición de la imagen proviene de un video grabado por medio de una cámara montada en un vehículo escala 1:10 que se desplaza por un carril igualmente a escala; la pista cuenta con dos carriles para un mismo sentido, estos estarán divididos por medio de líneas punteadas. La Figura 4 muestra un segmento de la pista en la que se desarrollará el algoritmo.

Figura 4. Vía a escala 1:10

A continuación, se presentará la descripción de la implementación para la detección de líneas del contorno de un carril que se resume en el diagrama que se muestra en la Figura 5.

Figura 5. Implementación de algoritmo de detección de línea [2]



El algoritmo propuesto cuenta con 10 etapas entre el inicio y fin; se desarrolla en Python 3 y algunas herramientas proporcionadas por la biblioteca OpenCV 4. A continuación describe cada parte del algoritmo.

1. **Adquisición de la imagen en bruto:** es la imagen capturada directamente de la cámara (un cuadro que conforma parte del video), esto se logra gracias a la librería OpenCV para Python, esta librería igualmente se usa en el pre-procesado de la imagen. Un ejemplo de imagen captada se muestra en la Figura 6.

Figura 6. Imagen en bruto capturada por cámara montada en el vehículo escala en el laboratorio de computo inteligente de alto rendimiento en CITEDI.

2. **Selección de área de interés:** se selecciona un área de interés que discrimine todo aquello que pueda considerase como ruido en el algoritmo; como la pista no cambiará de forma o tamaño es adecuado fijar el área de interés de manera constante. La Figura 7 muestra el área seleccionada.

Figura 7. Selección de área de interés para el procesamiento de la región deseada (ROI por sus siglas en ingles)

3. **Transformación de perspectiva-"Bird's eye":** analizar la imagen hasta este punto resulta un tanto difícil, aún con gran cantidad del ruido mitigado, por el hecho de que las líneas no se visualizan en un plano 2D pue se pueden apreciar una cierta profundidad, lo que dificulta el problema, para facilitar el análisis se procede a hacer un cambio de perspectiva al estilo bird's-eye para llevar a cabo un análisis en 2D justo como se muestra en la Figura 8 [8].

Figura 8. Cambio de perspectiva estilo Birds Eye

4. **Filtro Gaussiano:** este filtro se suele utilizar para reducir detalle y así eliminar ruido de la imagen, siendo el principal reto en el diseño de este tipo de algoritmos; en la Figura 9 se muestra el efecto de este filtro.

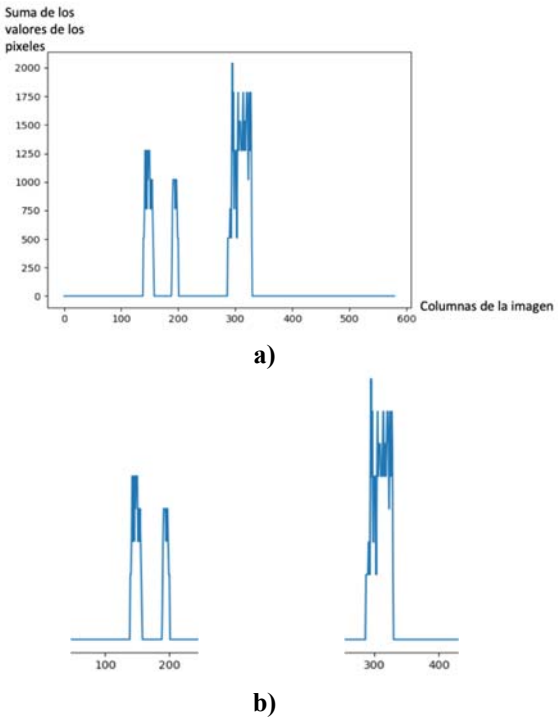
Figura 9. Filtro Gaussiano para reducción de ruido de la señal

5. **Mascara de colores HSV:** esta técnica permite hacer una binarización de la imagen dividiendo los pixeles mediante un umbral con un rango máximo y mínimo de valores en HSV, la calidad de identificación depende de cuantas máscaras se apliquen para así considerar más umbrales, para este caso se utilizaron 8 máscaras con diferentes umbrales para mitigar ruido y hacer mejor detección, el resultado se muestra en la Figura 10.

Figura 10. Aplicación de mascara de colores HSV

6. **Generación de histograma para detección de carril:** este es uno de los pasos más importantes para el desarrollo del algoritmo, dado que servirá una buena detección. En esta parte, la imagen binarizada produce un histograma considerando los valores y posiciones de los pixeles de la imagen, esto se puede ver en la Figura 11ª gracias a la librería de Python 3 MatplotLib para visalización.

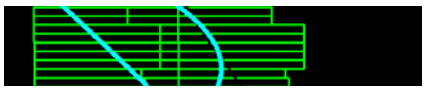
Figura 11. a) Generación de histograma a partir de imagen binarizada. b) División de histograma para obtener el valor máximo de cada distribución.



El histograma cuenta con varios valores pico que se pueden considerar como máximos locales, sin embargo, la utilidad del histograma es esencialmente permitir al algoritmo de barrido de ventanas (*sliding windows*) reconocer en dónde comenzar el barrido; Por lo cual, el histograma se divide en dos mitades (Figura 11b), para así encontrar el valor máximo global de cada mitad que viene a representar la posición de inicio de cada línea del carril.

7. **Algoritmo de barrido de ventanas:** este algoritmo consiste en generar secciones rectangulares que se deslizan a lo largo de la imagen para analizar cada sección y procesar el contenido de manera individual, en este caso para determinar si existen pixeles blancos, los cuales indican la presencia de las líneas que delimitan un carril. Gracias al paso anterior se obtienen puntos de referencia para eficientizar la búsqueda y así posteriormente generar polinomios que modelen los carriles como curvas a partir de interpolación polinomial de segundo orden [5], considerando el centro de cada ventana que encuadre un segmento de la línea del carril, con lo cual se obtiene un resultado como el de la Figura 12.

Figura 12. Detección de líneas del carril por medio de barrido de ventanas e interpolación polinomial.

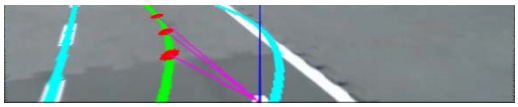


8. **Cálculo de curva entre líneas:** una vez se han modelado las líneas que definen los límites laterales del carril, se procede a estimar una línea imaginaria que represente el centro del carril en el cual debería de estar alineado el vehículo, esto se logra simplemente tomando el valor absoluto de la resta de los puntos de las líneas estimadas y sumándolos al lado izquierdo (o restándolos al lado derecho) para lograr el resultado que se muestra en la Figura 13.

Figura 13. Generación de curva intermedia del carril.

9. **Cálculo de ángulo de giro:** esta es la última parte del algoritmo y se enfoca en calcular el ángulo de giro que el vehículo debería de corregir para poderse alinear al centro del carril. El objetivo se logra al crear un punto de referencia central a la cámara y crear tres puntos a lo largo de la línea central que se ha generado en el paso anterior, con los cuales se calcula un ángulo entre el punto de referencia central y cada uno de los tres puntos para finalmente computar un promedio ponderado que represente el ángulo adecuado de giro. La Figura 14 muestra gráficamente el procedimiento descrito en la imagen resultante.

Figura 14. Cálculo de giro centrar al vehículo en el carril.



El procedimiento descrito con anterioridad se ha implementado en el lenguaje de programación Python 3 en conjunto con la librería OpenCV 4 y MatplotLib. En la próxima sección se procede a analizar los resultados obtenidos al aplicar el algoritmo en videos pregrabados.

3. RESULTADOS

Para analizar el desempeño del algoritmo propuesto es necesario contemplar métricas que permitan estimar qué tan confiable y eficiente es este. Los resultados consistirán en un análisis de la relación de cuadros que existen en un video y la cantidad de cuadros que no tuvieron una detección de carril exitosa, de manera que se pueda calcular el error porcentual dado por la fórmula (2).

$$\frac{Total\ de\ cuadros - (Cuadros\ con\ detección)}{Total\ de\ cuadros} \times 100 = Error\ porcentual \quad (2)$$

$$Cuadros\ con\ detección = Total\ de\ cuadros - Cuadros\ \text{sin}\ detección$$

Para la métrica se considerará el desempeño del algoritmo en 6 videos pregrabados en donde se puede ver la ruta que sigue el vehículo.

Los escenarios consisten en una misma vía con dos carriles, sin embargo, habrá una variación en la velocidad con la que se desplazará el vehículo para la grabación del video y una variación en el manejo (bueno o malo), los resultados correspondientes se mostrarán en la tabla 1.

Tabla 1.Evaluación del algoritmo de detección de carril.

	Cambio de carril	Conducción	Velocidad	Total de cuadros	Cuadros sin detección	Error porcentual (%)
1	Izquierda-Derecha	Movimientos Suaves	Rápida	120	2	1.666
2	Derecha-Izquierda	Movimientos Suaves	Rápida	116	4	3.448
3	Izquierda-Derecha	Movimientos Suaves	Lenta	288	6	2.083
4	Derecha-Izquierda	Movimientos Suaves	Lenta	284	7	2.464
5	Izquierda-Derecha	Movimientos Abruptos	Rápida	172	29	16.86
6	Derecha-Izquierda	Movimientos Abruptos	Rápida	296	42	14.18

Dados los resultados obtenidos en la tabla 1, es posible evaluar la eficacia del algoritmo implementado. Este algoritmo mostró tener un error menor al 5% independientemente de la velocidad, siempre y cuando hubiese una buena conducción como en los casos (1-4), el error aumento cuando hubo una mala conducción por el hecho de que la orientación del vehículo no permitía que este tuviese una buena percepción del carril y por tanto el

algoritmo no pudiera llevar a cabo una detección adecuada cuando no se hacían transiciones prudentes entre carriles como en los casos 5 y 6, de cualquier forma se obtuvo un error menor al 20 %, que para efectos de prototipado y vehículos a escala, no es un error muy significativo, sin embargo es posible deducir que una buena conducción permitiría reducir este error considerablemente.

4. CONCLUSIONES Y RECOMENDACIONES

4.1 Observaciones generales

El proyecto ha permitido explorar una aplicación de la visión de computadora orientada al desarrollo de vehículos autónomos, la cual es la detección de carriles por medio de procesamiento de imagen. En este caso se lograron los resultados deseados pues efectivamente se construyó un algoritmo que cumplía con el propósito, y aun cuando los resultados de las métricas vistos en la sección de resultados son aceptables para condiciones de simulación o prototipado con vehículos a escala, es necesario mejorar el rendimiento de la implementación para poder llevar este sistema a vehículos reales. Existen mejoras que se pueden realizar para eficientizar el algoritmo y hacerlo más robusto, sin embargo, las bases y fundamentos son suficientemente buenas y permiten dar solución al problema de manera adecuada.

La aportación principal de la implementación descrita fue, como se describió al principio, crear un algoritmo híbrido en donde se utilizó la metodología de características y de modelado para hacer una detección exitosa con la cual fuese posible crear una señal de retroalimentación para un sistema de control dedicado a mantener al vehículo en medio del carril, lo anterior fue exitosamente logrado y la metodología para obtener la señal del sistema es de buena calidad y fiabilidad.

4.2 Trabajo futuro

Este algoritmo se ha desenvuelto de manera aceptable en las condiciones en que se ha puesto a prueba utilizando métodos “tradicionales” de procesamiento de imágenes, sin embargo, es evidente resaltar que existe la posibilidad de mejorar el algoritmo con métodos inteligentes que se encuentran en tendencia como por ejemplo redes neuronales convolucionales. De igual forma, se buscará pasar de la simulación a la práctica utilizando el ángulo de giro como señal de retroalimentación para un sistema de control ya sea clásico o difuso para que el prototipo a escala sea capaz de navegar de manera autónoma en una vía con carriles bien definidos.

5. REFERENCIAS

[1] Wally Chen, Leon Jian and Sy-Yen Kuo, "Video-based on-road driving safety system with lane detection and vehicle detection," 2012 12th International Conference on ITS Telecommunications, Taipei, 2012, pp. 537-541.
doi: 10.1109/ITST.2012.6425237

[2] Kaur, G., & Kumar, D. (2015). Lane detection techniques: A review. International Journal of Computer Applications, 112(10).

[3] A. A. Assidiq, O. O. Khalifa, M. R. Islam and S. Khan, "Real time lane detection for autonomous vehicles," 2008 International Conference on Computer and Communication Engineering, Kuala Lumpur, 2008, pp. 82-88.
doi: 10.1109/ICCCE.2008.4580573

[4] Ganage, D. G., Nikam, N. S., & Wagh, S. A. (2019). Vision based driver alertness system for lane detection. International Journal of Engineering and Advanced Technology, 8(6), 1394-1397. doi:10.35940/ijeat.F8102.088619

[5] Bhushan, S., & Ganorkar, S. R. (2015). Real-Time Lane Detection For Driving System Using Image Processing. Int. J. for Research & Development in Technology (IJRDT), 5(1), 19-23.

[6] Yinghua He, Hong Wang and Bo Zhang, "Color-based road detection in urban traffic scenes," in IEEE Transactions on Intelligent Transportation Systems, vol. 5, no. 4, pp. 309-318, Dec. 2004.
doi: 10.1109/TITS.2004.838221

[7] C. H. Lampert, M. B. Blaschko and T. Hofmann, "Beyond sliding windows: Object localization by efficient subwindow search," 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, 2008, pp. 1-8.
doi: 10.1109/CVPR.2008.4587586

[8] Falaleev, N. (2017). *Bird's Eye View Transformation*. Retrieved from https://nikolasent.github.io/opencv /2017/05/07/Bird%27s-Eye-View-Transformation.html

AGRADECIMIENTOS:

Estas actividades se desarrollaron en el marco del proyecto de investigación SIP: 20190053, Sistemas Inteligentes Cuánticos, Parte 1. Director: Dr. Oscar Humberto Montiel Ross, del Instituto Politécnico Nacional.