

# REPORTE DEL SISTEMA DE DETECCIÓN Y CONTEO DE AUTOS

24. Noviembre, 2025

**INTEGRANTES DEL EQUIPO  
CON MATRICULA ESCOLAR:**

Elena Isabel Espriella  
Bustamante – 80187

Adrián Moisés  
Martínez Hernández–  
80325

Stella Isabel Casillas  
Ramírez – 38919

Carlos Jaasiel Morrison  
Ramírez – 80702

**NOMBRE DEL PROFESOR:**

Mtro. Emmanuel  
Ovalle Magallanes

**NOMBRE DE LA MATERIA:**

Modelado y  
procesamiento de  
imágenes

**SEMESTRE Y GRUPO:**

5to Semestre  
Grupo: 512

PARCIAL 3, PROYECTO  
FINAL

# ÍNDICE

**1** Panorama general del sistema

---

**2** Fundamento de la  
implementación

---

**3** Marco Teórico o Conceptual

---

**4** Descripción del Problema

**5** Metodología

---

**6** Funcionamiento del Sistema

---

**7** Análisis de desempeño

---

**8** Cierre técnico del proyecto

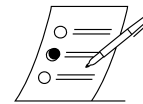
# 01 Panorama general del sistema

En la actualidad, muchos estacionamientos no logran satisfacer la demanda de los usuarios debido a la falta de información sobre la disponibilidad real de espacios. Para atender esta necesidad, desarrollamos un sistema capaz de detectar y contabilizar los vehículos que ingresan a un estacionamiento mediante un código implementado en Python y el uso de elementos esenciales como un video de la entrada de un estacionamiento público.



# 02 Fundamento de la implementación

El sistema aplica estas técnicas directamente al video del estacionamiento. El video se reduce y se convierte a escala de grises, se definen dos ROIs basadas en porcentajes del alto del cuadro y se obtiene un baseline de intensidad durante los primeros 40 fotogramas. Cuando la intensidad cambia más allá del umbral DELTA\_PLUMA durante MIN\_FRAMES consecutivos, se confirma el paso de un vehículo y se aplica un filtro de contraste en esa ROI usando funciones de mpi.py.



## METODOLOGÍA

La metodología se basó en el uso de técnicas de procesamiento digital de imágenes implementadas en mpi.py, aplicando filtros, conversiones de color y análisis por regiones de interés para detectar el cruce de vehículos en el video.



## RESULTADOS

Como resultado, el sistema logró identificar y contar autos de manera estable y precisa, validando cada evento mediante un ajuste de contraste aplicado en el momento exacto de la detección.

## 03 Marco Teórico o Conceptual

El procesamiento digital de imágenes permite analizar variaciones de intensidad en los píxeles para extraer información útil de fotos o videos. Se utilizan conceptos como conversión a escala de grises, filtrado mediante kernels, convolución y análisis por regiones de interés (ROIs), que ayudan a detectar cambios dentro de una escena sin necesidad de modelos avanzados de visión por computadora.

```
# ----- FUNCIONES AUXILIARES -----  
  
def get_gray_small(frame):  
    frame_small = cv2.resize(frame, (0, 0), fx=SCALE, fy=SCALE)  
    rgb = cv2.cvtColor(frame_small, cv2.COLOR_BGR2RGB)  
    gray = mpi.RGB2GRAY(rgb).astype(np.uint8)  
    return frame_small, gray
```

## 04 Descripción del Problema



En estacionamientos convencionales, los usuarios desconocen la disponibilidad real de espacios lo que genera saturación, filas innecesarias y tiempos de espera elevados. La falta de un sistema accesible capaz de monitorear y contabilizar los vehículos en tiempo real limita la eficiencia y la experiencia del usuario. El proyecto resuelve esta necesidad mediante un sistema capaz de detectar y contar autos usando únicamente procesamiento digital de imágenes.



## 05 Metodología

Para comenzar, se redujo el tamaño del video al 50% con el fin de acelerar el procesamiento sin comprometer la información relevante. Cada frame es convertido a RGB y posteriormente a escala de grises mediante la función `mpi.RGB2GRAY()`. Esto simplifica el análisis, pues trabajar con una sola banda de intensidad permite detectar cambios más claramente.

```
# ----- CONFIGURACIÓN -----  
  
BASE_DIR = os.path.dirname(os.path.abspath(__file__))  
VIDEO_PATH = os.path.join(BASE_DIR, "data", "pluma.mp4")  
  
SCALE = 0.5          # bajar tamaño del video (0.5 = 50%)  
CALIB_FRAMES = 40    # frames iniciales para calibrar "pluma cerrada"
```

El siguiente paso consistió en definir dos regiones de interés ubicadas verticalmente sobre el tramo donde pasan los vehículos. Ambas regiones abarcan las plumas de acceso, funcionando como “sensores virtuales” que detectan perturbaciones en la intensidad cuando un auto cruza. Las posiciones de estos ROIs se calcularon con porcentajes de la altura del video, lo que vuelve el sistema adaptable a otros videos sin necesidad de medir píxeles manualmente.

```
# ROIs de las dos plumas  
PLUMA1_Y1_F = 0.50    # pluma de adelante  
PLUMA1_Y2_F = 0.58  
  
PLUMA2_Y1_F = 0.40    # pluma de atrás  
PLUMA2_Y2_F = 0.48
```

Antes de empezar a detectar autos, el programa realiza una calibración automática como se ve en la primera imagen. Durante los primeros 40 frames, cuando las plumas están cerradas, el sistema toma el promedio de intensidad de cada ROI. Ese valor se guarda como baseline. Posteriormente, durante la ejecución normal, cualquier cambio significativo que se aleje de ese baseline es interpretado como movimiento causado por un auto.

La lógica de detección se basa en observar cuánto se desvía la intensidad actual respecto a la intensidad base. Si la diferencia rebasa cierto umbral (`DELTA_PLUMA`) durante varias imágenes consecutivas (`MIN_FRAMES`), entonces se confirma que un auto ha pasado. Esto evita falsos positivos por ruido, sombras, parpadeos o vibraciones de cámara.

```
DELTA_PLUMA = 20      # qué tanto debe cambiar la intensidad para decir "se abrió"  
MIN_FRAMES = 5        # mínimo de frames seguidos en estado "abierto" para contar
```

Cada vez que un ROI detecta una apertura (es decir, un auto cruzando), se incrementa el contador correspondiente y se activa un flag que indica que el evento ocurrió en ese frame. Con ese flag se aplica un filtro de contraste al ROI usando la función `mpi.convolucion2D()` con un kernel tipo sharpen. Este filtrado subraya visualmente el auto en el preciso momento en que es detectado, sin alterar el conteo ni afectar los demás frames.

```
# ROI pluma 1
roi1 = gray[pluma1_y1:pluma1_y2, :]
if evento_roi[0]:
    # aplicar filtro de convolución 2D con kernel de enfoque
    roi1_filt = mpi.convolucion2D(roi1, sharpen_kernel)
    roi1_show = cv2.cvtColor(roi1_filt, cv2.COLOR_GRAY2BGR)
else:
    roi1_show = cv2.cvtColor(roi1, cv2.COLOR_GRAY2BGR)

# ROI pluma 2
roi2 = gray[pluma2_y1:pluma2_y2, :]
if evento_roi[1]:
    roi2_filt = mpi.convolucion2D(roi2, sharpen_kernel)
    roi2_show = cv2.cvtColor(roi2_filt, cv2.COLOR_GRAY2BGR)
else:
    roi2_show = cv2.cvtColor(roi2, cv2.COLOR_GRAY2BGR)

cv2.imshow("ROI Pluma 1", cv2.resize(roi1_show, (0, 0), fx=0.8, fy=0.8))
cv2.imshow("ROI Pluma 2", cv2.resize(roi2_show, (0, 0), fx=0.8, fy=0.8))
```

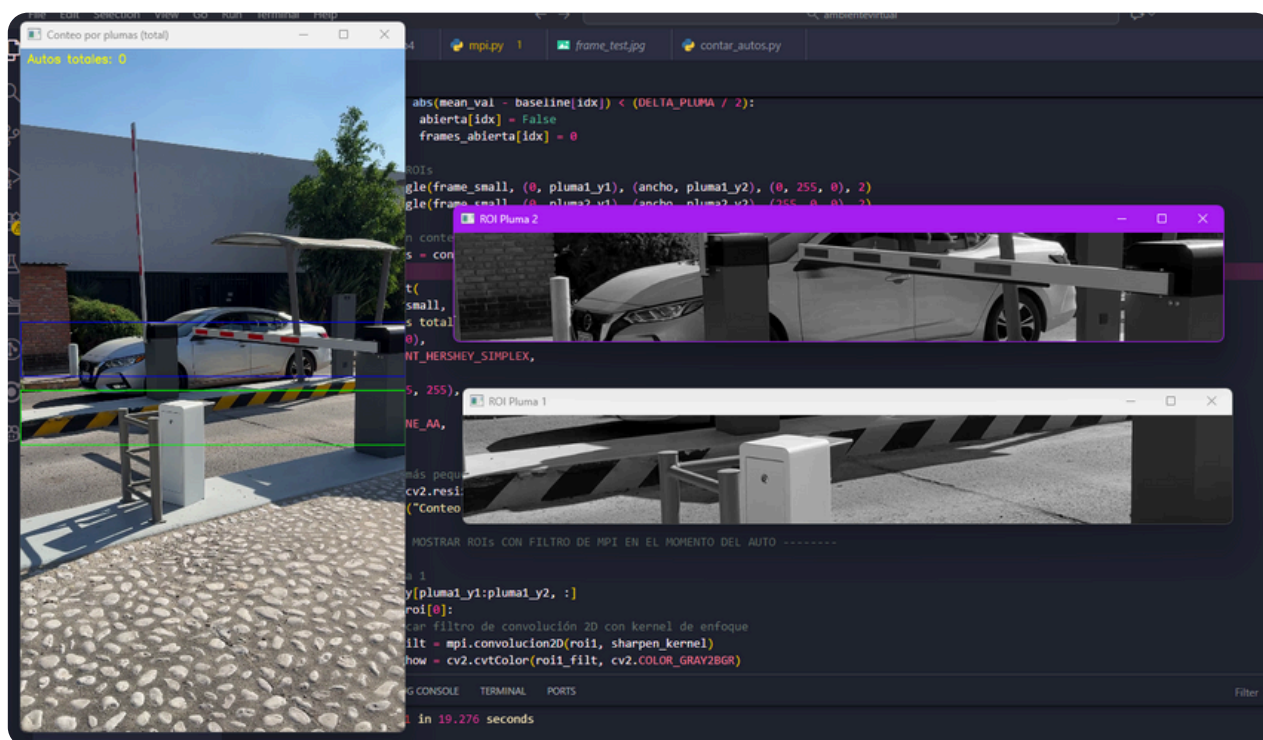


Todo el video se muestra en pantalla a un tamaño reducido para permitir verlo completo incluso en monitores pequeños. Además, cada ROI se presenta en ventanas separadas: tanto su vista normal en escala de grises como su versión filtrada cuando ocurre un evento. Esto permite validar fácilmente si el sistema está detectando correctamente.

## 06 Funcionamiento del Sistema

El sistema procesa el video cuadro por cuadro. En cada imagen reduce su tamaño, la convierte a escala de grises y analiza únicamente dos franjas horizontales del video. Cuando el valor promedio de un ROI cambia de manera suficiente y consistente, se interpreta que ha pasado un auto.

El video fue grabado con una cámara ubicada aproximadamente a 2 metros de la pluma, a una altura de 1.6 metros y con un ángulo lateral cercano a 35 grados respecto al flujo de vehículos. Esta posición permite observar claramente el cruce completo de los autos. Las ROIs se calculan como porcentajes de la imagen para asegurar que el sistema funcione igual en cualquier equipo sin ajustes manuales.



La detección no depende de formas, bordes ni modelos entrenados, sino únicamente del comportamiento de la intensidad dentro de las regiones. Cuando la detección se confirma, se suma al conteo total de autos y la parte correspondiente del ROI se remarca mediante un filtro de contraste. Finalmente, todo se visualiza en tiempo real: el conteo total, cada ROI, y el video original reducido.

# Análisis de desempeño

---



## RESULTADOS

Las pruebas se realizaron con el video original de un minuto, en el que los coches pasan muy cerca uno del otro y las vías están también próximas. El sistema, al detectar los vehículos uno a uno y sin generar duplicados, respondió correctamente. En ningún momento se detectaron alteraciones engañosas provocadas por luz o ruido.



## PRUEBAS

Se probó además con diversos niveles de reducción de video, distintos valores de umbrales de intensidad y diversas cantidades de fotogramas requeridos para confirmar una apertura. En todas las situaciones, el sistema se mantuvo preciso y estable; esto evidencia que la lógica basada en intensidad es eficaz y se mantiene estable en este tipo de escenas.

## TOTAL DE EVENTOS DETECTADOS

5

vehículos  
correctamente  
identificados



## VALIDACIÓN VISUAL

La verificación de que el sistema resaltaba el área cuando un auto pasaba por un ROI fue posible gracias a la visualización con filtro activado. Esto contribuyó a verificar visualmente la detección.

## FUNCIONAMIENTO

El sistema cumplió exitosamente con su propósito: detectar y contar autos usando únicamente técnicas de procesamiento de imágenes. Su funcionamiento se basa en la comparación de intensidades dentro de regiones de interés, lo cual resulta sorprendentemente efectivo aun sin emplear algoritmos sofisticados.

## MODIFICABILIDAD

Una ventaja importante es que la solución es completamente modificable; basta ajustar la posición de los ROIs y algunos parámetros para adaptarla a otro video. El hecho de que use funciones del archivo mpi.py reafirma el aprendizaje sobre filtros, convoluciones y análisis de imágenes de nivel básico.

## ALCANCE DEL MÉTODO

Aunque este método depende de la iluminación y requiere que los autos pasen por las zonas previstas, ofrece una forma eficiente y clara de extraer información significativa de un video real. El resultado final es un programa funcional, explicable y comprobable, que demuestra cómo el procesamiento digital de imágenes puede utilizarse para resolver problemas prácticos sin recurrir a modelos avanzados.

# Cierre técnico del proyecto

---

## BIBLIOGRAFÍA

☆ Gonzalez, R. C., & Woods, R. E. (2018). Digital Image Processing (4th ed.). Pearson.

<https://www.cl72.org/090imagePLib/books/Gonzales,Woods-Digital.Image.Processing.4th.Edition.pdf>

☆ Szeliski, R. (2022). Computer Vision: Algorithms and Applications (2nd ed.). Springer.

<https://library.huree.edu.mn/data/202295/2024-06-03/Computer%20Vision%20-%20Algorithms%20and%20Applications%202nd%20Edition,%20Richard%20Szeliski.pdf>

☆ Forsyth, D. A., & Ponce, J. (2011). Computer Vision: A Modern Approach (2nd ed.). Pearson.

<https://eclass.hmu.gr/modules/document/file.php/TM152/Books/Computer%20Vision%20-%20A%20Modern%20Approach%20-%20D.%20Forsyth%2C%20J.%20Ponce.pdf>

☆ OpenCV. (2023). OpenCV Documentation.

<https://docs.opencv.org/4.x/>

☆ Python Software Foundation. (2024). Python documentation.

<https://docs.python.org/3/>

☆ Russ, J. C. (2011). The Image Processing Handbook (6th ed.). CRC Press.

<https://www.taylorfrancis.com/books/mono/10.1201/b10720/image-processing-handbook-john-russ>

☆ Jähne, B. (2005). Digital Image Processing (6th ed.). Springer.

[https://aitskadapa.ac.in/e-books/CSE/DIGITAL%20IMAGE%20PROCESSING/Digital%20Image%20Processing%20\(%20PDFDrive%20\)%20\(1\).pdf](https://aitskadapa.ac.in/e-books/CSE/DIGITAL%20IMAGE%20PROCESSING/Digital%20Image%20Processing%20(%20PDFDrive%20)%20(1).pdf)

☆ Rosenfeld, A., & Kak, A. (1982). Digital Picture Processing (2nd ed.). Academic Press.

[https://www.researchgate.net/publication/2997098\\_Digital\\_Picture\\_Processing](https://www.researchgate.net/publication/2997098_Digital_Picture_Processing)

☆ Burger, W., & Burge, M. J. (2016). Digital Image Processing: An Algorithmic Introduction (2nd ed.). Springer.

<https://link.springer.com/book/10.1007/978-1-4471-6684-9>