FACULTAD DE CIENCIAS EXACTAS, INGENIERIA Y AGRIMENSURA
UNIVERSIDAD NACIONAL DE ROSARIO

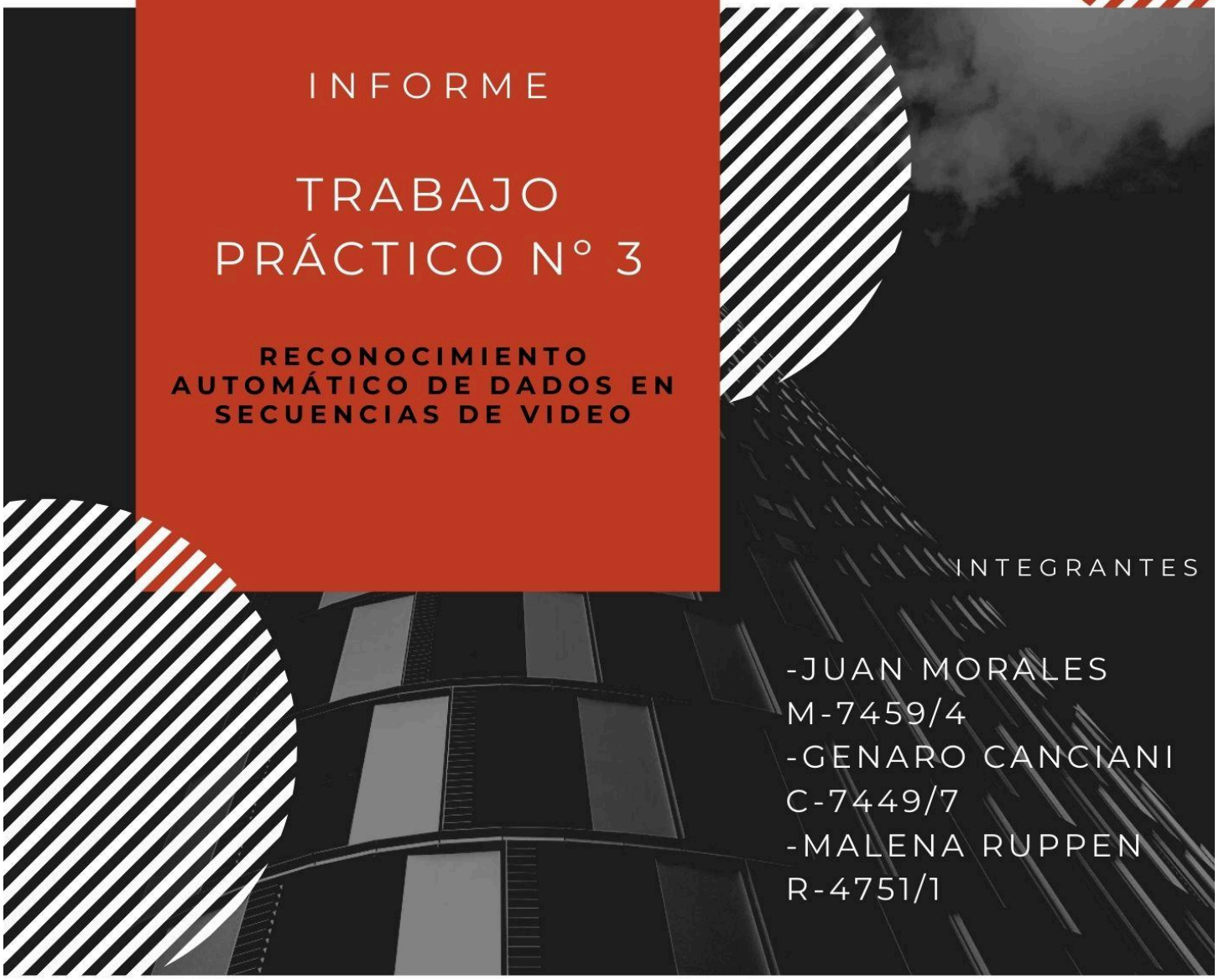
TECNICATURA UNIVERSITARIA EN INTELIGENCIA ARTIFICIAL

# PROCESAMIENTO DE IMÁGENES

INFORME

## TRABAJO PRÁCTICO N° 3

### RECONOCIMIENTO AUTOMÁTICO DE DADOS EN SECUENCIAS DE VIDEO




INTEGRANTES

-JUAN MORALES
M-7459/4
-GENARO CANCIANI
C-7449/7
-MALENA RUPPEN
R-4751/1


***PROFESORES: GONZALO SAD, JUAN MANUEL CALLE, JOAQUÍN ALLIONE.***

# Ejercicio: Detección de Dados y Lectura de Valores en Video

El objetivo de este ejercicio es desarrollar un algoritmo de visión por computadora capaz de procesar secuencias de video (tirada_n.mp4) donde se lanzan dados rojos sobre una superficie verde. El sistema debe operar de manera automática para resolver las siguientes consignas:

- **A. Detección de Estado Estacionario:** Determinar automáticamente los momentos del video en los que los dados han dejado de moverse (reposo).
- **B. Segmentación y Lectura:** Identificar la ubicación de cada dado y calcular el valor numérico de su cara superior contando los "pips" (puntos).
- **C. Generación de Resultados:** Exportar un nuevo video donde se visualicen los bounding boxes, etiquetas con el valor detectado y el estado del procesamiento.

# Análisis y Técnicas de Resolución

Para resolver el problema, se implementó una estrategia secuencial que abarca desde la segmentación por color hasta la detección de primitivas geométricas (círculos) para el conteo. A continuación se detallan las etapas:

## 1. Segmentación por Color (Extracción de Dados)

Problemas Enfrentados:

El principal desafío es aislar los dados del fondo. Si bien el fondo es verde uniforme, los dados son translúcidos y presentan brillos especulares (reflejos blancos) que pueden romper la máscara de segmentación.

Técnicas Utilizadas: Espacio de Color HSV

Se optó por convertir los frames de BGR a HSV (Matiz, Saturación, Valor). Dado que el color rojo se encuentra en los extremos del espectro cromático en OpenCV (0-10 y 170-180), se implementó una función extraer_dados_rojos que combina dos máscaras mediante una operación lógica OR (cv2.bitwise_or).

- **Rango Bajo:** H=[0, 10]
- **Rango Alto:** H=[170, 180]

Esto permite capturar todas las tonalidades del dado ignorando el fondo verde y las sombras oscuras.

## 2. Detección de Movimiento (Quietud)

Estrategia:

Procesar los dados mientras están rodando es ineficiente y propenso a errores debido al desenfoque de movimiento (motion blur). El algoritmo compara el área total de los contornos entre el frame actual y el frame anterior.

- **Lógica:** Se utiliza la función detectar_quietud. Si la diferencia absoluta entre la suma de las áreas de los contornos rojos del frame t y el frame t-1 es menor a un umbral experimental (45 píxeles), se asume que la escena está estática y se procede al reconocimiento.

### 3. Procesamiento Morfológico y Búsqueda de Contornos

Una vez obtenida la máscara binaria de los dados, se aplican técnicas para mejorar la definición de los objetos antes de buscar sus bordes:

1. **Detección de Bordes (Canny):** Se aplica el algoritmo de Canny sobre la máscara para resaltar los límites de los dados.
2. **Dilatación (Morphological Dilation):** Se aplica cv2.dilate para cerrar posibles huecos en los bordes causados por los brillos internos del dado translúcido, asegurando contornos cerrados.
3. **Filtrado por Área:** Al iterar sobre los contornos encontrados (cv2.findContours), se descartan aquellos cuya área no corresponda a la de un dado promedio (rango 4300 < area < 6400), eliminando así ruido o reflejos espurios.

### 4. Reconocimiento de Puntos (Hough Circles)

Problema:

Contar los puntos dentro de cada dado requiere distinguir círculos pequeños y difusos dentro de una región rectangular.

Técnica: Transformada de Hough para Círculos

Para cada dado detectado, se extrae un recorte (ROI) de la imagen original. Sobre este recorte se aplica una cadena de procesamiento específica:

1. **Conversión a Gris:** Para trabajar con intensidad de luz.
2. **Gaussian Blur:** Suavizado (kernel 9x9) para eliminar el ruido de textura del dado.
3. **Canny (Interno):** Para detectar los bordes de los puntos.
4. **HoughCircles:** Se utiliza cv2.HoughCircles con parámetros ajustados (minDist=8, minRadius=5, maxRadius=8) para detectar exclusivamente los puntos del dado.

La cantidad de círculos detectados corresponde directamente al valor numérico del dado.

## Resultados y Visualización

El algoritmo implementa una función de depuración (debug_visual) que permite visualizar, mediante matplotlib, las etapas intermedias del proceso para el primer video, facilitando el ajuste de parámetros.

**Etapas Visualizadas:**

1. **Frame Quieto:** Momento exacto donde el algoritmo decide procesar.
2. **Máscara de Color:** Resultado de la segmentación HSV.
3. **Detalle del Dado:** Visualización del recorte, el desenfoque y los círculos encontrados por Hough.

Salida Final:

El sistema genera archivos de video en la carpeta salidas/ donde se superponen rectángulos azules sobre los dados y etiquetas de texto con el valor contado, validando la eficacia de la solución propuesta.

# Conclusiones

La solución desarrollada demostró ser robusta para las condiciones presentadas.

- El uso de **doble rango HSV** fue crítico para segmentar correctamente los objetos rojos.
- La **detección de quietud basada en variación de áreas** resultó ser un método computacionalmente económico y efectivo para evitar procesar frames con *motion blur*.
- La **Transformada de Hough** probó ser superior al simple conteo de contornos internos para los puntos, ya que discrimina mejor por forma circular perfecta, ignorando manchas irregulares de luz.