Universidad Nacional de Rosario
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Tecnicatura Universitaria en Inteligencia Artificial
Procesamiento de Imágenes I - IA 4.4

TRABAJO PRÁCTICO N°3

GRUPO 6

INTEGRANTES:

Cima, Nancy Lucía
Longo, Gonzalo
Sumiacher, Julia

Índice

Problema 1

Descripción y Enunciado

Se tienen cuatro secuencias de video firada_1.mp4, tirada_2.mp4, tirada_3.mp4 y tirada_4.mp4 corresponden a tiradas de 5 dados. En la Figura 1 se muestran los dados luego de una tirada. En base a esas secuencias de video, se debe realizar lo siguiente:

a. Desarrollar un algoritmo para detectar automáticamente cuando se detienen los dados y leer el número obtenido en cada uno. Informar todos los pasos de procesamiento.

b. Generar videos (uno para cada archivo) donde los dados, mientras estén en reposo, aparezcan resaltados con un bounding box de color azul y ademas, agregar sobre los mismos el número reconocido.

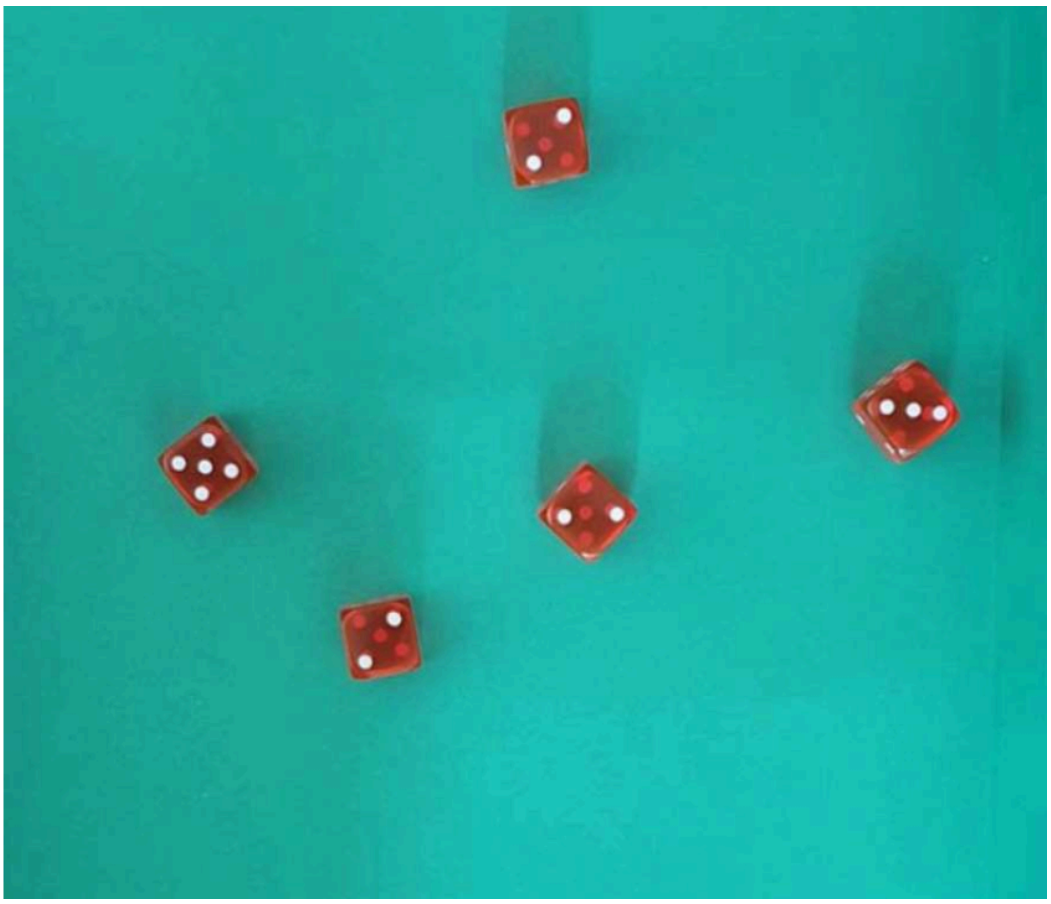

Figura 1: Dados luego de una tirada

Es decir, el objetivo principal es desarrollar una solución que permita detectar automáticamente el momento en que los dados se detienen en secuencias de video, para luego reconocer y resaltar el número mostrado por cada dado. Esto implica analizar cada cuadro del vídeo, identificar los dados en reposo y reconocer los números en sus caras superiores.

Desarrollo

A continuación, se detalla el desarrollo y las etapas implementadas.

Primero, es importante considerar que para este problema se utilizan bibliotecas como OpenCV, NumPy y Matplotlib para la manipulación, análisis y visualización de imágenes, respectivamente.

Para facilitar el desarrollo del algoritmo de detección automática de tiradas de dados se decidió realizar modularizar el código. A continuación, se detallan las funciones realizadas.

Funciones de Visualización

El sistema incluye funciones para facilitar la visualización de imágenes:

1. **imshow**: Simplifica la visualización de una imagen, permitiendo configurar aspectos como título, colorbar y ticks.
2. **subplot3 y subplot4**: Permiten mostrar 3 o 4 imágenes en un formato de subplot, útil para comparar diferentes etapas del procesamiento.

Carga y Procesamiento de Video

Por otro lado, la función **carga_frames** se encarga de cargar los frames de un video y almacenarlos en formato RGB. Esto permite procesar cada frame de forma individual.

Para este paso, primero se creó un script aparte de ejemplo donde, se desarrolló un código auxiliar para procesar videos y extraer cada fotograma como una imagen independiente.

Inicialmente, se asegura la existencia de un directorio llamado frames, donde se almacenarán los fotogramas extraídos. Si el directorio no existe, el programa lo crea automáticamente mediante la función os.makedirs.

Posteriormente, el script abre un archivo de vídeo (tirada_x.mp4) utilizando cv2.VideoCapture y obtiene información esencial sobre el video, como su ancho, alto y número de cuadros por segundo (FPS). Esto es útil para ajustar las dimensiones de cada fotograma y asegurar que el análisis se realice de manera consistente.

El proceso de lectura de los fotogramas se realiza en un bucle mientras el video esté correctamente cargado. Cada fotograma es redimensionado a un tercio de su tamaño original mediante cv2.resize, lo que reduce la carga de procesamiento sin comprometer la calidad visual necesaria para las pruebas. Los fotogramas son visualizados en tiempo real con cv2.imshow y simultáneamente almacenados en el directorio frames con nombres secuenciales (frame_0.jpg, frame_1.jpg, etc.), utilizando cv2.imwrite.

El bucle incluye un mecanismo para detener la ejecución manualmente al presionar la tecla q, asegurando flexibilidad durante la ejecución. Una vez que se procesan todos los fotogramas o se interrumpe el proceso, los recursos del vídeo y las ventanas abiertas son liberados con cap.release y cv2.destroyAllWindows.

Segmentación por Color

La función **mask_y_mask_inv_segHSV** realiza la segmentación de color en el espacio HSV, detectando píxeles que correspondan al color verde de la mesa. Genera dos máscaras:

- **Máscara binaria**: Identifica las regiones verdes.
- **Máscara inversa**: Resalta las regiones no verdes (dados y otros objetos).

En esta sección del proyecto, primero se hizo un script para calcular de forma dinámica los umbrales de valores HSV. La idea principal fue diseñar una herramienta interactiva que permita ajustar dinámicamente los umbrales de color mediante una GUI basada en OpenCV.

La función principal, denominada **segmentacion_dinamica**, genera una interfaz gráfica donde el usuario puede ajustar los valores mínimos y máximos de los canales Hue, Saturation y Value (H, S y V). Estos ajustes se reflejan en tiempo real sobre la imagen, permitiendo identificar de manera visual las áreas segmentadas según los colores seleccionados.

Adicionalmente, se incluyó la función **mask_y_mask_inv_segHSV**, que genera máscaras binarias e inversas para una imagen, basándose en los intervalos de color definidos. Esto permitió realizar una segmentación precisa de colores específicos, como verde y rojo, mediante pruebas realizadas en diferentes fotogramas. Los resultados se visualizaron en un formato comparativo de cuatro imágenes: la

imagen original, la máscara binaria, la máscara inversa y la segmentación final.

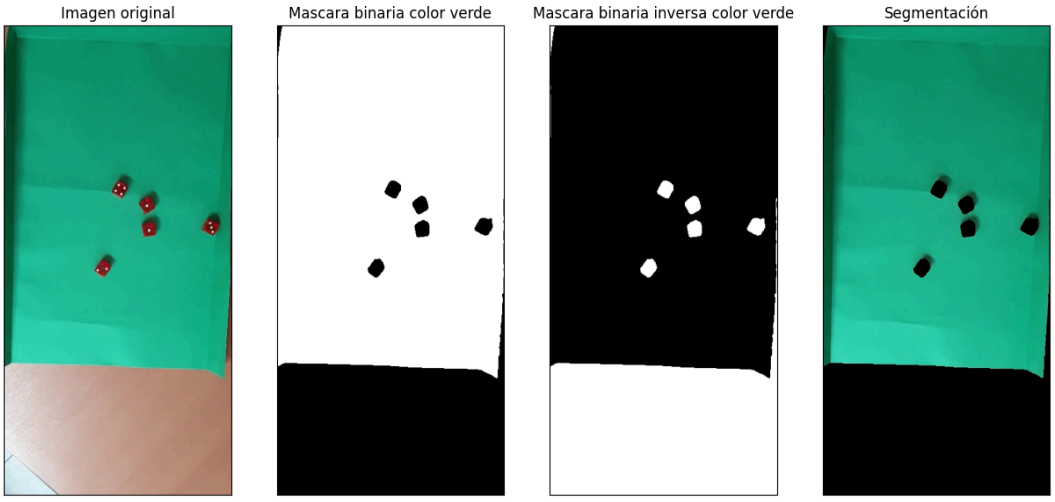

Figura 2: Mascaras para el color verde

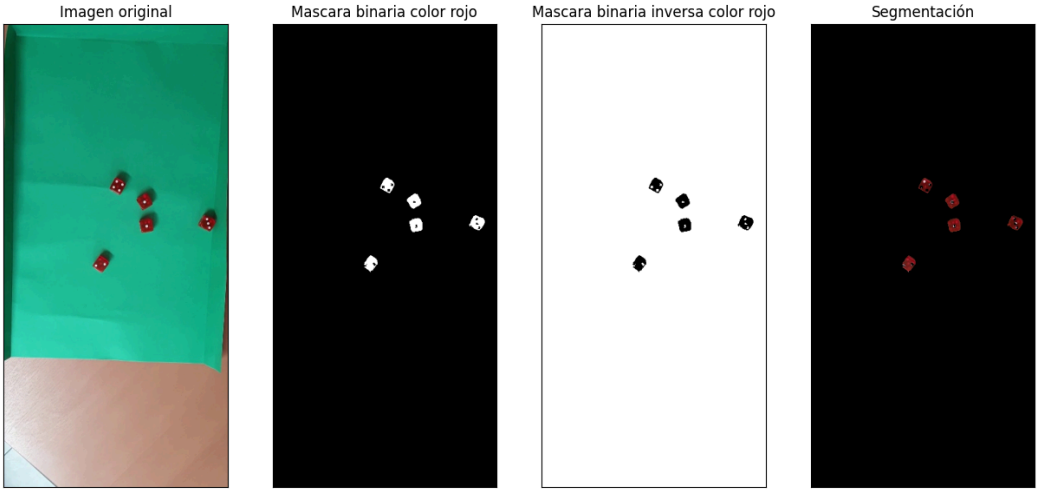

Figura 3: Mascaras para el color rojo

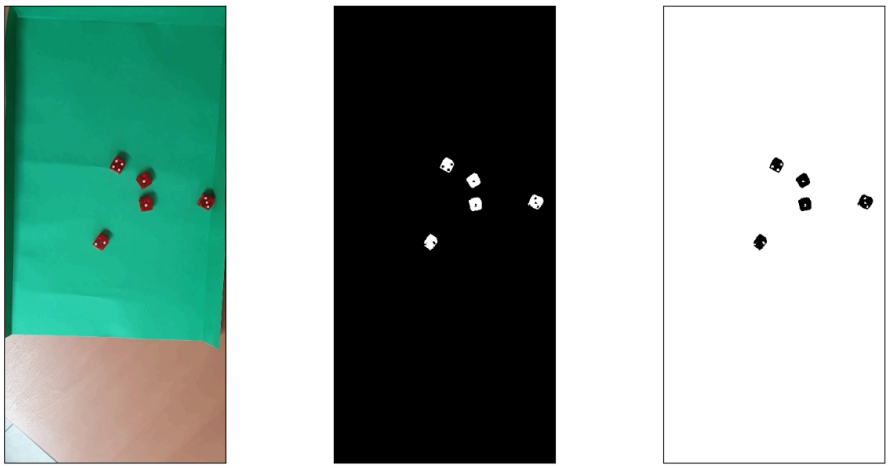

Figura 4: Máscara binaria mejorada

Finalmente, se utilizaron los umbrales obtenidos en el código principal del problema (Problema1.py) como entrada de la función **mask_y_mask_inv_segHSV**.

Filtrado de Regiones

El sistema implementa filtros basados en:

- **Área (filtrar_area)**: Filtra regiones conectadas cuya área esté fuera de un rango predefinido.
- **Relación de Aspecto (filtrar_relAsp)**: Filtra regiones cuya proporción de altura y ancho no sea cercana a 1 (formas cuadradas, como los dados).

Indicaremos cuando se aplican estos filtros a continuación.

Detección de Dados Estáticos

La función **dados_en_mesa** analiza una imagen (img) para determinar si contiene dados en una mesa bajo condiciones específicas:

1. La cantidad de píxeles verdes debe ser mayor que un umbral definido.
2. El área ocupada por los dados debe estar dentro de un rango aceptable.

Primero, utiliza una función auxiliar (mask_y_mask_inv_segHSV) para segmentar los píxeles verdes en la imagen.

Esto devuelve dos máscaras:

1. mask: contiene los píxeles verdes de la imagen.
2. mask_inv: contiene el complemento, es decir, los píxeles que no son verdes.

Se calcula el total de píxeles verdes en la imagen sumando los valores de la máscara mask. Este valor se usará más adelante como una de las condiciones para determinar si hay dados sobre la mesa.

También, se aplica filtrar_area para filtrar objetos en la máscara invertida (mask_inv) según su área. Solo se conservan regiones cuya área esté dentro del rango [4400, 6000], el cual corresponde aproximadamente al tamaño esperado de los dados.

Luego, se calculan las **componentes conectadas** en la máscara filtrada. Cada componente representa un objeto separado en la imagen, como un dado.

- num_labels indica el número total de etiquetas encontradas (incluyendo el fondo).
- cant_dados se calcula restando 1 (para descontar la etiqueta del fondo), representando el número de dados detectados.

Se define que el número de dados debe estar entre 1 y 6, considerando que puede haber más de un dado muy próximo o algún objeto del tamaño de un dado (como un dedo). Se establece un umbral (CANT_VERDE_TH) para la cantidad mínima de píxeles verdes necesarios. Esto asegura que la imagen incluye una cantidad significativa de área verde, representando la mesa.

Se evalúan las condiciones:

1. El número de dados está dentro del rango aceptado.
2. La cantidad de píxeles verdes supera el umbral.

Si ambas condiciones se cumplen, la función retorna True, indicando que los dados están sobre la mesa, que pueden estar girando y/o estáticos.

Filtrado de Frames Útiles

La función **filtrar_frames_dados_estaticos** está diseñada para procesar una secuencia de fotogramas de un video y seleccionar aquellos que contienen dados estáticos en la mesa, siguiendo estos pasos:

1. Filtra los frames que cumplen con un criterio específico (dados en la mesa, evaluados por una función llamada dados_en_mesa()).
2. Retorna solo la mitad superior de los frames útiles (se asume que esta parte tiene los dados estáticos).
3. Calcula y devuelve índices relacionados con la posición de los frames seleccionados.

Identificación de Dados y Cálculo de Puntos

La función **detecta_dados** detecta y clasifica los dados presentes en los frames estáticos. Para ello:

1. Detecta dados en un frame de referencia (el frame central de la lista).
2. Identifica las posiciones de los dados mediante máscaras (bounding boxes).
3. Marca estas posiciones en todos los frames estáticos.
4. Devuelve los frames con los dados identificados y la máscara de los bounding boxes.

Internamente define:

- indice_ref: Índice del frame central, considerado como el frame de referencia.
- frame_ref: Frame original seleccionado como referencia.
- frame_ref_c: Copia del frame de referencia para modificaciones.
- frame_ref_gray: Versión en escala de grises del frame de referencia, útil para operaciones posteriores.

Luego, genera máscaras para los dados usando la función mask_y_mask_inv_segHSV

Hecho esto, filtra los dados detectados utilizando las funciones filtrar_area y filtrar_relAsp. Así, pasa a detectar contornos utilizando cv2.connectedComponentsWithStats() sobre mask_inv.

Este paso devuelve:

- num_labels: Número de objetos detectados (incluyendo el fondo).
- labels: Imagen etiquetada, donde cada píxel tiene el ID del objeto al que pertenece.
- stats: Estadísticas de cada contorno, como posición (x, y), ancho, alto, y área.
- centroids: Centros de masa de los objetos detectados.

Hecho esto, para cada contorno:

1. Se extrae un bounding box definido por las coordenadas (x1, y1) y (x2, y2).
2. Se convierte la región del dado a escala de grises (dado_gray) y se aplica un suavizado (dado_gray_blur).
3. Se aplica un umbral binario para segmentar las caras del dado (mask).

Finalmente, se usa la misma técnica de componentes conectados para contar los puntos visibles en cada dado.

Así pasamos a dibujar bounding boxes y etiquetar. Para esto dibuja un rectángulo (bounding box) en mask_box para resaltar cada dado y escribe el número de puntos detectados en el dado.

Se usa la máscara mask_box para aplicar los mismos bounding boxes a todos los frames estáticos.

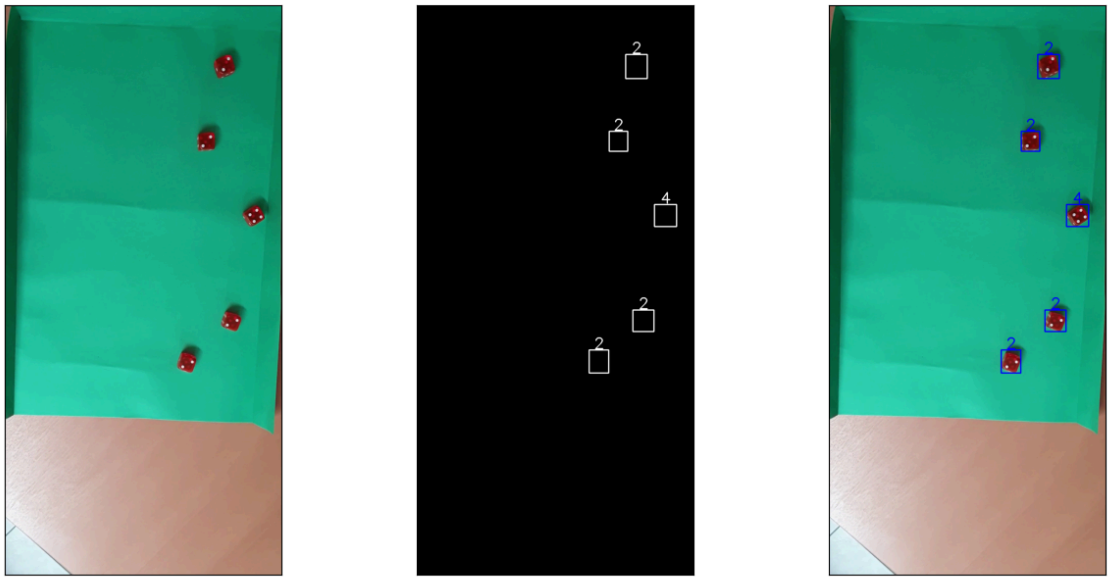

Figura 5: Dados segmentados y clasificados en un frame

Otro componente auxiliar desarrollado para el proyecto fue un script que permite leer un video, realizar anotaciones sobre sus fotogramas y generar un nuevo video con los cambios aplicados. Este proceso utiliza la biblioteca cv2 de OpenCV y sirve como base para pruebas de manipulación dinámica de videos.

El código comienza abriendo un archivo de vídeo (tirada_x.mp4) mediante cv2.VideoCapture, desde donde se obtienen sus propiedades básicas, como dimensiones (ancho y alto), cuadros por segundo (FPS) y el número total de fotogramas. Con esta información, se configura un objeto VideoWriter para escribir un nuevo video de salida (Video-Output.mp4) que mantenga las propiedades del video original. El codec utilizado es mp4v, garantizando compatibilidad y calidad adecuada.

Durante el procesamiento, cada fotograma es leído y anotado con un rectángulo rojo en coordenadas específicas (100,100) a (200,200). Para facilitar la visualización en tiempo real, los fotogramas son redimensionados a un tercio de su tamaño original y mostrados en una ventana de vista previa con cv2.imshow. Sin embargo, el video de salida conserva las dimensiones originales para mantener la integridad del contenido.

El bucle principal permite visualizar cada fotograma y escribirlo en el archivo de salida hasta que se procesen todos los fotogramas o el usuario interrumpa el proceso presionando la tecla q. Finalmente, se liberan los recursos asociados al video de entrada (cap), el video de salida (out) y las ventanas creadas durante la ejecución.

Este script ha sido esencial para realizar pruebas en la manipulación y anotación de videos, permitiendo evaluar la superposición de elementos gráficos y la generación de contenido derivado.

En resumen, una vez listas las diferentes funciones, se carga un video de tirada de dados y se crea una copia para no modificar las imágenes originales con bounding box.

Luego, se detectan los frames con dados y los índices con respecto a la lista frames. Estos índices nos servirán luego para grabar el video final, modificando los frames que solo estén dentro de esos índices y dejando el resto iguales. Esto se realiza mediante la función ‘filtrar_frames_dados_estaticos’.

Una vez obtenidos los frames, se identifican los dados y sus puntos en dichos frames mediante la función ‘detecta_dados’. Recuadraremos estos dados con bounding box en todos los frames para poder grabar el video.

Finalmente, se graba el video con los frames resultantes utilizando la función grabar_video.

Las imágenes procesadas, así como los resultados intermedios (máscaras, bounding boxes, etc.), se muestran de forma visual para validar cada paso del procesamiento.

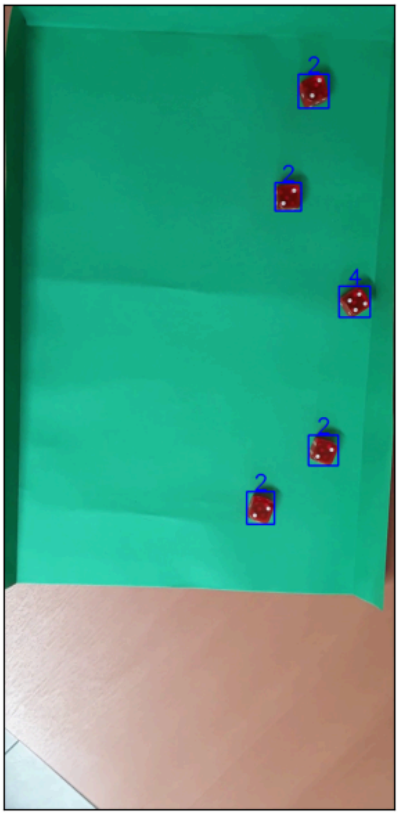

Figura 6: Frame del video final con dados segmentados y clasificados

Conclusión

Este trabajo demuestra el proceso de diseño e implementación de un sistema para la detección y análisis de dados utilizando técnicas de procesamiento de imágenes. A lo largo del desarrollo, se integraron herramientas como segmentación en el espacio HSV, detección de componentes conectados y uso de máscaras binarias, permitiendo identificar con precisión los dados y su posición en los frames seleccionados.

El método utilizado aprovecha un frame de referencia para reducir la complejidad del análisis en los demás frames estáticos, lo que optimiza los recursos computacionales y asegura una mayor consistencia en los resultados. Aunque algunos pasos, como la detección de números en los dados, requirieron ajustes manuales debido a limitaciones en ciertos algoritmos estándar, esto permitió un mejor entendimiento de los desafíos del procesamiento de imágenes.

El sistema desarrollado demuestra ser una herramienta efectiva para la detección automática de dados en un video. Este enfoque puede ser aplicado en diversos contextos, como juegos de mesa o análisis automatizado de actividades lúdicas, destacando su precisión y capacidad de adaptación a diferentes escenarios.