Universidad Nacional de Rosario
Facultad de Ciencias Exactas, Ingeniería y Agrimensura

IA4.4 Procesamiento de Imágenes

Trabajo Práctico N°3

Tecnicatura Universitaria en Inteligencia Artificial

Integrantes (Grupo 16):

Alsop, Agustín (A-4651/7)

Asad, Gonzalo (A-4595/1)

Castells, Sergio (C-7334/2)

Hachen, Rocío (H-1184/3)

Docentes:

Gonzalo Sad

Julián Álvarez

Juan Manuel Calle

Fecha: 09/12/2024

1. INTRODUCCIÓN

En el presente trabajo se desarrollan técnicas avanzadas de procesamiento de imágenes para analizar tiradas de dados grabadas en video. El objetivo es automatizar la detección de los momentos en que los dados se detienen, leer los valores visibles y generar videos de salida que resalten los dados en reposo y se muestre el número detectado.

2. PROBLEMA 1 – CINCO DADOS

2.1 DESCRIPCIÓN

El problema consiste en procesar cuatro videos de tiradas de cinco dados (`tirada_1.mp4`, `tirada_2.mp4`, etc.) sobre un paño verde, como se observa en la Figura 1. La tarea consiste en:

1. Desarrollar un algoritmo para detectar automáticamente cuando se detienen los dados y leer el número obtenido en cada uno.
2. Generar videos (uno para cada archivo) donde los dados sean señalados mediante bounding boxes de color azul y además, mientras estén en reposo, mostrar sobre los mismos los números detectados. También se debe incluir el resultado de la tirada.
3. Reconocer el resultado obtenido según las reglas del juego “Generala”.

Figura 1: Dados luego de una tirada.

2.2 RESOLUCIÓN

2.2.1 Delimitación de área de interés

La función `roiDetect` aplicada sobre el primer cuadro identifica automáticamente el área de interés (paño) y obtiene sus coordenadas. Para ello convierte la imagen al espacio de color LAB y detecta componentes conectadas en el canal A. Las coordenadas obtenidas se utilizan para recortar los cuadros posteriores, lo que permite enfocar el procesamiento en una región específica.

Figura 2: Área recortada.

Para los cuadros (frames) posteriores, se vuelve a emplear el espacio de color LAB, aislando nuevamente el canal A para resaltar los dados sobre el fondo verde.

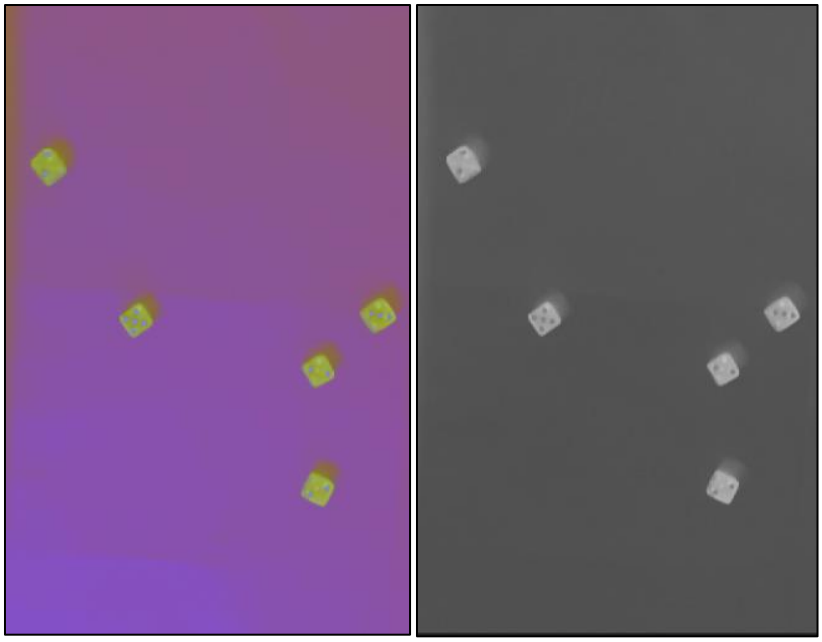

Figura 3: Canal A de CIELAB.

A continuación, se implementa la función `centroidsDetect`, encargada de identificar los cinco dados y calcular sus centroides y estadísticas. Esto se logra mediante un proceso que incluye umbralado y detección de componentes conectadas filtradas por área.

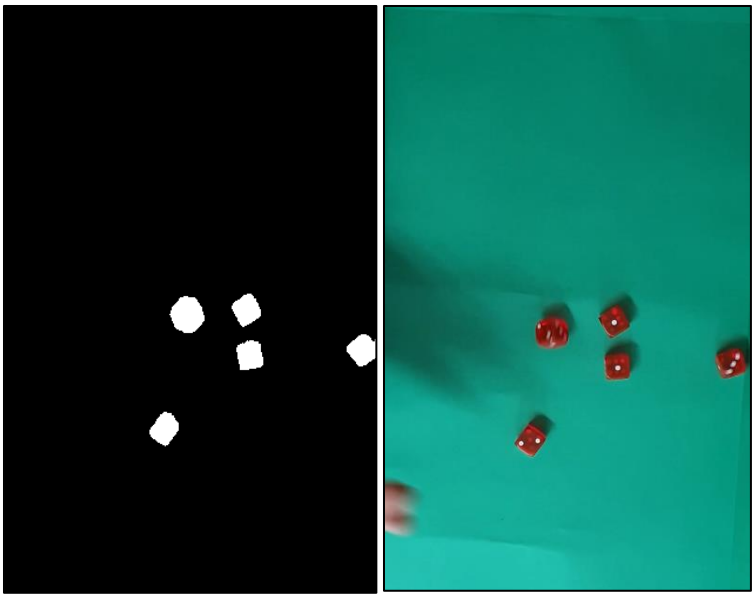

Figura 4: Imagen umbralada y cuadro original.

Con las estadísticas obtenidas, se trazará un rectángulo delimitador alrededor de cada dado identificado, incluso cuando estos se encuentren en movimiento al momento de su detección.

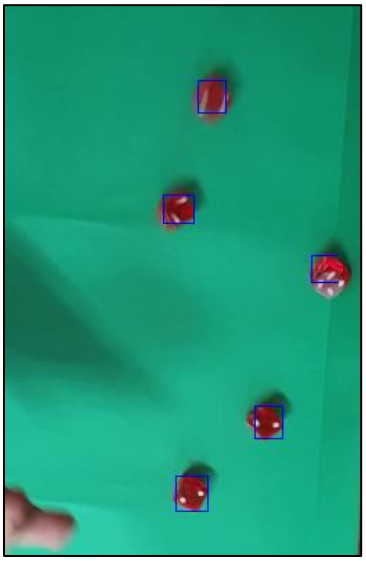

Figura 5: Cuadros delimitadores sobre dados en movimiento.

2.2.2 Detección de movimiento

Para identificar el número de cada dado, es necesario que este permanezca inmóvil. Para detectar esta condición, se emplea la función `motionDetector`, que compara las posiciones de los centroides de cada dado entre cuadros consecutivos. Además, a fin de evitar detecciones espurias, se utiliza un contador denominado `aux_mov`, el cual confirma la condición de quietud únicamente después de varios cuadros consecutivos sin movimiento.

2.2.3 Detección y clasificación de dados

Cuando los dados se encuentran en reposo, es posible proceder a analizar sus valores. La función `diceValue` identifica los puntos presentes en la cara superior de cada dado mediante operaciones morfológicas y análisis de componentes conectadas.

Figura 6: Recorte de un dado.

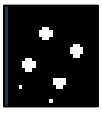

Figura 7: Dado tras aplicar umbralado.

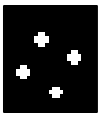

Figura 8: Número identificable tras aplicar apertura.

Los valores de cada dado se almacenan en una lista. Posteriormente, en cada cuadro y para cada dado, se verifica si su valor ya ha sido determinado; en caso afirmativo, el dato no se actualiza nuevamente. Este enfoque previene que el valor cambie al momento de recoger los dados, asegurando la consistencia de los resultados.

Acto seguido, se añade un texto sobre el rectángulo delimitador de cada dado, indicando el número correspondiente a cada uno de ellos.

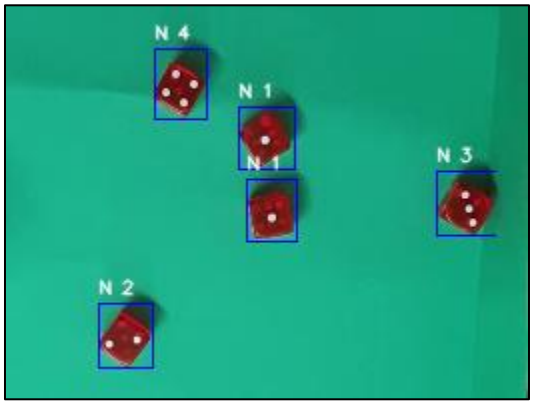

Figura 9: Dados detectados con sus respectivos valores.

2.2.4 Generación de resultados

La función `gameAnalyzer` analiza los valores de los dados y verifica si conforman una jugada válida, como Generala, Póker o Full. Esta información, es suministrada a la función `insertPicture` la cual insertará una imagen que comunica visualmente el resultado de la tirada.

Figura 10: Ejemplos de resultados posibles.

Finalmente, se ajustan las dimensiones de cada cuadro y se incluyen en el video de salida

Figura 11: Capturas de resultados finales con los dados detenidos.

2.3 FUNCIONES UTILIZADAS

2.3.1 roiDetect

Identifica automáticamente el área de interés (el paño verde donde se encuentran los dados) y devuelve sus límites como coordenadas. La función recibe como parámetros:

- `img(np.ndarray)`: Imagen de entrada en formato BGR.
- `percent(float)`: Porcentaje de margen a ajustar en los bordes del ROI. Por defecto es 5%.
- `thresh(int)`: Valor del umbral para la máscara binaria.
- `save(bool)`: Indica si se guarda la imagen procesada.

Funcionamiento:

1. **Conversión al espacio LAB:**
   La imagen de entrada se convierte al espacio LAB. Este espacio de color separa la luminosidad (L) de las cromaticidades (A - rojo-verde y B - azul-amarillo), lo que facilita segmentar colores específicos como el verde del paño.

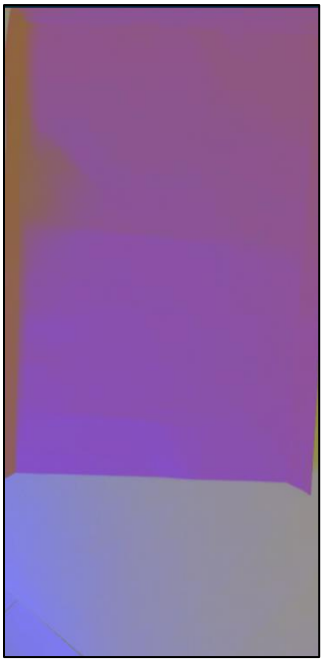

Figura 12: Imagen en espacio LAB.

2. **Umbralado:**
   Se aplica un umbral binario inverso al canal A, que corresponde al color rojo-verde. Esto resalta el paño verde mientras se descarta el fondo.

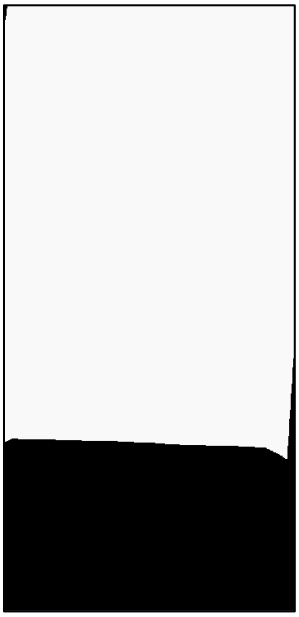

Figura 13: Imagen umbralada.

3. **Detección de componentes conectadas:**
   Utilizando `connectedComponentsWithStats`, se identifican todas las componentes conectadas en la imagen umbralada y se obtienen las coordenadas del área de interés, las cuales se ajustan con un margen especificado por el parámetro `percent`.

Figura 14: Área reconocida.

2.3.2 centroidsDetect

Devuelve las coordenadas de los centroides, las estadísticas de las regiones válidas (e.g., ancho, alto, área) y una bandera (`flag`) que indica si se encontraron exactamente 6 componentes válidas. La función recibe los siguientes parámetros:

- `img(np.ndarray)`: Imagen de entrada.
- `th_min(int)`: Valor de umbral inicial para obtener una máscara binaria.
- `min_area(int)`: Área mínima que debe tener una componente conectada para ser considerada válida.
- `max_area(int)`: Área máxima que puede tener una componente conectada para ser considerada válida. Debe ser menor o igual al área total de la imagen.
- `jump(int)`: Incremento en el umbral `'th_min'` en cada iteración si las condiciones no se cumplen.

Funcionamiento:

1. **Umbralado inicial:**
   Se aplica un umbral binario para segmentar los objetos en la imagen.
2. **Detección de Componentes conectadas**:
   Se implementa `connectedComponentsWithStats` y se verifica que la cantidad de labels detectadas sea igual a 6. Si esta condición no se cumple se reinicia el proceso incrementando el umbral un valor `jump`.
3. **Filtrado por área:**
   Las regiones detectadas se filtran según su área, se eliminan aquellos con área menor a `min_area` y mayor a `max_area`.
4. **Iteración con ajustes de umbral:**
   Si tras la etapa de filtrado no subsisten exactamente cinco componentes, se reinicia el proceso ajustando el umbral hasta obtener resultados válidos.

2.3.3 motionDetector

Compara las posiciones de los centroides entre cuadros consecutivos. Si el desplazamiento es menor a un umbral, se considera que no hay movimiento. Retorna `True` si se detecta movimiento, `False` en caso contrario. Sus parámetros son:

- `ant(list)`: Lista de coordenadas de los centroides en el cuadro anterior.
- `act(list)`: Lista de coordenadas de los centroides en el cuadro actual.
- `thresh(int)`: Umbral de desplazamiento para determinar si hay movimiento.

Funcionamiento:

1. **Ordenamiento de centroides:**
   Los centroides de los cuadros anterior y actual se ordenan para garantizar correspondencia entre ellos.
2. **Cálculo del desplazamiento:**
   Se calcula la distancia euclidiana entre centroides correspondientes en ambos cuadros. Si el desplazamiento de todos los centroides es menor a un umbral (`thresh`), se considera que no hay movimiento.

2.3.4 diceValue

Calcula el valor de un dado en una imagen a partir del conteo de puntos en su superficie. Recibe como parámetros:

- `img(np.ndarray)`: Imagen de entrada.
- `x_cord(int)`: Coordenada X superior izquierda de la ROI.
- `y_cord(int)`: Coordenada Y superior izquierda de la ROI.
- `width(int)`: Ancho de la ROI.
- `height(int)`: Altura de la ROI.

Funcionamiento:

1. **Recorte de la imagen:**
   Se recorta la imagen principal utilizando las coordenadas x e y, junto con el ancho y la altura proporcionados como argumentos.
2. **Umbralado y morfología:**
   Se aplica un umbral a la región de interés y se realizan operaciones morfológicas de apertura utilizando un kernel elíptico, con el objeto de aislar los puntos visibles en la cara del dado.
3. **Componentes conectadas:**
   Se detectan las componentes conectadas, lo que permite determinar el valor representado en el dado.

2.3.5 gameAnalyzer

Evalúa los valores de los dados y clasifica las jugadas de acuerdo con las reglas del juego Generala. La función recibe un único argumento:

- `dados(list)`: Lista de 5 enteros entre 1 y 6, que representan los valores obtenidos en los dados.

Funcionamiento:

1. **Conteo:**
   Se realiza un conteo de las apariciones de cada número en la lista proporcionada.
2. **Análisis del resultado:**
   Se evalúan las combinaciones posibles en función de los valores y sus frecuencias, determinando el resultado de la jugada según las siguientes condiciones:

   - *Generala*: Todos los valores son iguales.
   - *Póker*: Cuatro valores iguales y uno distinto.
   - *Full*: Dos grupos de valores iguales, uno con tres y otro con dos elementos.
   - *Escalera*: Secuencia de 5 valores consecutivos, ya sea [1, 2, 3, 4, 5] o [2, 3, 4, 5, 6].
   - *Escalera al As*: Secuencia de 5 valores consecutivos, [3, 4, 5, 6, 1].
   - *Nada*: Cualquier otra combinación no contemplada en las reglas anteriores.

2.3.6 setReset

Implementa un latch set-reset (SR) básico. Parámetros:

- `set(bool)`: Señal de entrada para establecer el estado a `True`.
- `reset(bool)`: Señal de entrada para reiniciar el estado a `False`.
- `q(bool)`: Estado anterior.

2.3.7 insertPicture

Inserta una imagen dentro de otra, asegurando que las dimensiones sean compatibles. Recibe como argumentos:

- `img(np.ndarray)`: Imagen base donde se insertará la nueva imagen.
- `pict(dict)`: Diccionario que contiene imágenes de referencia.
- `ref(str)`: Clave de la imagen que se desea insertar.
- `x_cord(int)`: Coordenada X superior izquierda para insertar la imagen.
- `y_cord(int)`: Coordenada Y superior izquierda para insertar la imagen.

Funcionamiento:

La función recibe una imagen base (`img`) y, utilizando las coordenadas `x_cord` e `y_cord` para definir la posición superior izquierda, inserta dentro de ella otra imagen obtenida de un diccionario (`pict`) a través de una referencia (`ref`).

2.4 ANÁLISIS DEL DESARROLLO

Si bien se pudieron aplicar procedimientos utilizados en el trabajo práctico anterior, se necesitaron realizar adecuaciones para el funcionamiento correcto bajo las condiciones específicas de este trabajo. Fue necesario realizar una detección del movimiento de los dados para garantizar que la clasificación de sus valores fuera precisa, ya que, de no hacerlo, los valores fluctuaban y no permanecían constantes, lo que resultaba en una clasificación errónea de la jugada. Además, se decidió calcular los valores de los dados solo después de confirmar que permanecían quietos durante varios cuadros consecutivos, ya que la sombra proyectada por la mano sobre la imagen afectaba la exactitud de los resultados.

La elección del espacio de color para lograr un buen contraste entre los dados y el fondo no fue compleja. Se aprovechó la complementariedad entre los colores rojo y verde, utilizando el canal A del espacio CIELAB, que resalta estos dos colores. Este enfoque facilitó la detección de los dados y la identificación de las áreas de interés.

2.5 CONCLUSIONES

El algoritmo desarrollado cumple con éxito los objetivos del trabajo:

- Detecta con precisión las posiciones de los dados y muestra sus valores cuando se detienen.
- Analiza los resultados obtenidos y los muestra en pantalla.
- Genera los videos de salida.

Trabajar con videos no presenta una dificultad significativamente mayor que hacerlo con imágenes, ya que su procesamiento se realiza cuadro por cuadro, tratándolos como imágenes individuales. No obstante, fue necesario tener en cuenta el movimiento de los objetos y determinar el momento exacto en el que realizar ciertas operaciones, como la detección del valor de los dados. Esta tarea introduce una nueva dimensión al problema: el tiempo, lo que implica la necesidad de comparar una imagen con sus cuadros previos.

Elegir un buen espacio de colores puede marcar la diferencia entre el éxito y el fracaso del algoritmo. En este trabajo, optar por CIELAB facilitó considerablemente obtener un buen contraste entre los dados y el fondo. Sin embargo, es importante señalar que los videos analizados contaban con fondos uniformes, una condición que podría no ser tan común en muchas situaciones reales.

Limitaciones y mejoras:

El sistema está optimizado para los videos proporcionados, por lo que no se puede garantizar un buen desempeño en otros videos del mismo tipo o en aquellos donde los colores y las condiciones de iluminación varíen.

Aunque se implementó un sistema de detección del movimiento con un contador para asegurar que los dados estén quietos antes de detectar su valor, podría mejorarse el algoritmo para manejar situaciones donde hay más ruido de movimiento o las sombras de las manos interfieren más. Una opción podría ser implementar un modelo de redes neuronales entrenadas específicamente para detectar el movimiento de los dados en situaciones más ruidosas.

Además, en lugar de depender únicamente de operaciones morfológicas y filtrado, el uso de redes neuronales entrenadas para detectar los dados y sus valores podría mejorar la precisión en situaciones complicadas, especialmente cuando los mismos no son perfectamente visibles o se encuentran en posiciones no estándar.

Actualmente, el algoritmo solo detecta jugadas asociadas al juego de “La Generala”. Sin embargo, sería posible verificar jugadas de otros juegos de dados mediante una extensión de las reglas de clasificación.