Trabajo Práctico N° 3
Procesamiento de Imágenes - UNR

Detección de Dados y Valores

Integrantes:

Julian Ignacio Göttig - **Legajo**: G-5922/6

Fecha: **15/12/2025**

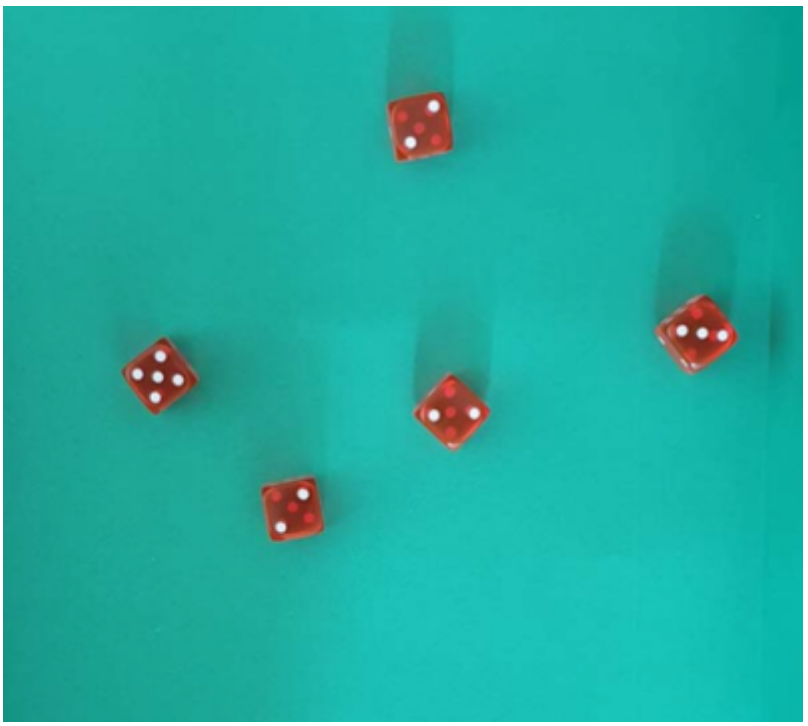
Detección de Dados y Valores

¿Qué estamos haciendo?

El objetivo principal de este Trabajo, es analizar cada tirada de dados, logrando obtener de las mismas la cantidad de dados arrojados y el valor final de cada uno.

Luego, una vez obtenida la información, dejamos almacenado en un video de salida la tirada original junto con un bounding-box en cada dado, mostrando su Valor y Nombre, junto con la sumatoria total de la tirada.

Para esto, buscamos hacerlo con la siguiente metodología:
1) Detectar el frame donde los dados permanecen estáticos
2) Detectar los bordes que tengan forma similar a Dados según su Relación de Aspecto y realizar el bounding-box pertinente
3) Dentro de cada bounding-box, buscamos la cantidad de Puntos, equivalente al valor que tiene el dado en ese momento.



¿Cómo lo estamos haciendo?

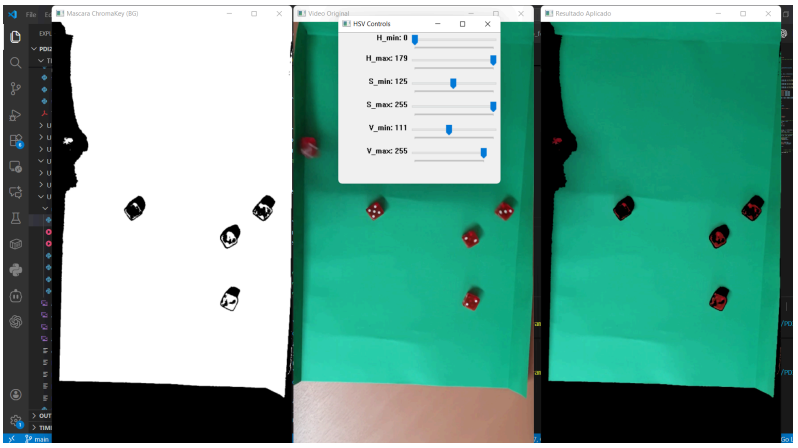
Visto que las tiradas son realizadas sobre un paño verde, y a su vez, todos los dados siempre quedan sobre el mismo (a pesar de que también se puede visualizar la mesa y la mano, entre otros), utilizamos el metodo de eliminar el fondo, filtrando el mismo a traves de una Mascara con el modelo de color HSV.

Para identificar cual es el rango HSV correcto que buscamos para aplicar la Máscara, hicimos inferencia de estos valores apoyándonos en un Display (posteriormente eliminado),

en el cual podíamos elegir valores Mínimos y Máximos de Tono, Saturación y Valor, y con esto, poder identificar Visualmente en qué rangos queríamos estar.

Primer aproximación:
Como podemos ver en la imagen, logramos eliminar el fondo, y detectar de buena forma el borde de los dados, el cual juegos de unos ajustes de valores, quedará con la forma correcta

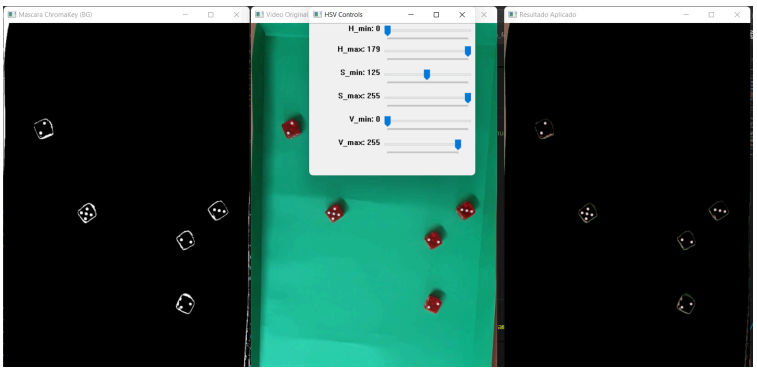
También se puede ver que en esta primer aproximación, estamos eliminando el color Rojo de los dados, sobre el cual más adelante haremos una aclaración



Segunda aproximación:

En este caso, ajustamos los valores necesarios, y además decidimos invertir la Mascara, para lograr así que el borde de los dados queden de color blanco, pudiendo ser detectados posteriormente

```
    mascara_original = cv2.inRange(frame_hsv, lower, upper)
    mascara = cv2.bitwise_not(mascara_original) #Invertimos
```



Detección de estaticidad:

Como siguiente paso, buscamos identificar donde comenzaban nuestros frames “Estáticos” para analizar la imagen

Para esto, establecimos valores umbrales a partir de los cuales consideramos que un frames se encuentra “Estático” respecto al anterior.

Este umbral, es de 15.000 píxeles, es decir, si un frame varía en >15.000 píxeles respecto al anterior, significa que sigue en “Movimiento”, de lo contrario, está “Estatico”, para esto, utilizamos una máscara de diferencia “mascara_diff” dejando en blanco (255) solo los pixeles que varían

A su vez, como segundo umbral, establecimos que la estaticidad debe mantenerse durante 10 frames continuos, en caso de hacerlo, consideramos que el video estúvo “Estatico” y obtenemos su **Primer Frame de los 10 estáticos** para analizar.

Detección de Dados:

Para este caso, apoyandonos en nuestro trabajo anterior, decidimos replicar las tecnicas de detección de bordes

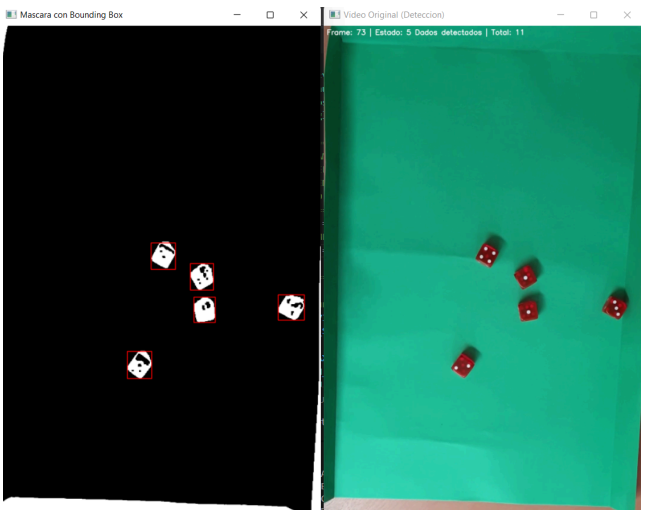
Una vez obtenido el Frame “estatico”, analizamos el mismo con nuestra función:

```
analizar_frame_estatico(frame, mascara_dados, frame_hsv):
```

En esta función utilizamos FindContourns, aplicando la función cv2.RETR_EXTERNAL, dado que de momento los bordes internos (puntos) no serán analizados.

En el código de ejecución principal, obtenemos la máscara de los dados aplicando un Kernel de Close de 5x5, y luego un Kernel de 3x3 con la función de Open.

Con esto, logramos “cerrar” los dados para que mantengan su forma cuadrada, y así poder obtener sus bordes con la morfología correcta.



Conteo de Puntos:

Para realizar el conteo, generamos una segunda máscara para puntos blancos, también utilizando HSV para rango de valores

Utilizando la función “analizar_dado”, realizamos el conteo de cuantos puntos tiene dentro cada dado y guardamos su valor en la variable “valor_dado” para luego mostrarlo tanto en consola como en el video de Salida.

Una vez obtenido los valores de los dados, guardamos los mismos en una lista llamada “resultados_dados”.

Videos de Salida:

Para cada video de salida, utilizamos la misma cantidad de FPS que su video original, para que estos se reproduzcan a la misma velocidad.

Luego de analizado el video y obtenidos los valores, en cada video de Salida dibujamos un bounding-box por dado, con las coordenadas obtenidas anteriormente, y por encima del mismo mostramos su Nombre (referencial) y su Valor

El usuario puede consultar estos videos dentro de la carpeta “tiradas_salidas” donde quedarán almacenados.

Extra:

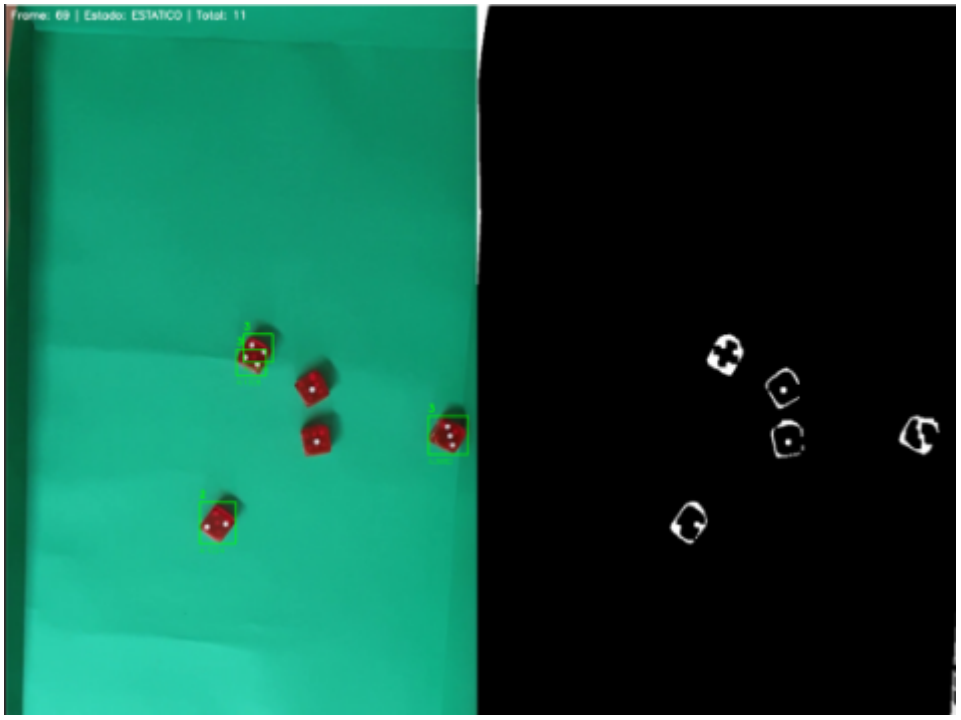
Dado que para el TP se solicitaba “informar y mostrar los resultados en cada una de las etapas de procesamiento.”, agregamos una variable “ver_mas”

La misma, se encuentra en la línea #14 de nuestro código, y descomentando la misma, la ejecución del código considerara unos videos extras a modo de mostrar el video original con las distintas máscaras aplicadas y los bounding-box a medida que los detecta

¿Qué problemas tuvimos?

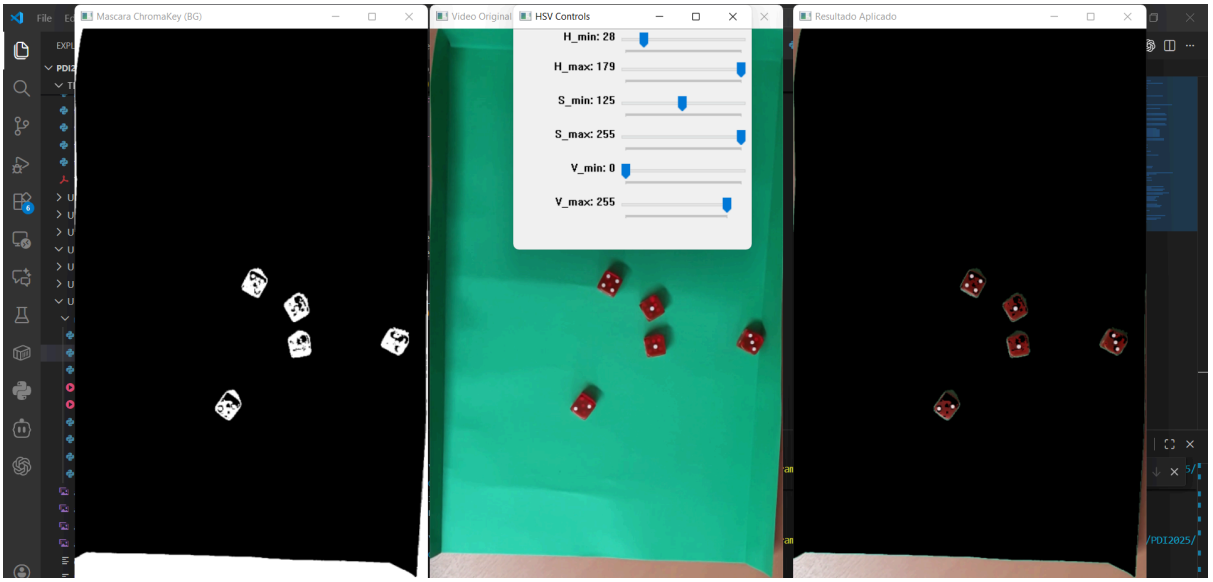
En primer lugar, la identificación exacta de los valores HSV a utilizar para eliminar el fondo verde requería de muchas iteraciones hasta llegar a su valor exacto, con lo cual, inteligentemente implementamos un Display para ir variando estos valores en tiempo real.

Por otro lado, luego de realizar la primera máscara, no estábamos logrando detectar correctamente los bordes de los dados, ya que (como se puede ver en imagen por debajo), los mismos no cerraban correctamente.



Para intentar lograr esto, buscamos aumentar el valor del Kernel de Closing aplicado, llevándolo incluso a un valor de 40x40, lo cual nos terminaba trayendo otras complicaciones como por ejemplo, que los distintos dados cercanos se tocaban entre sí.

Luego, identificamos que había una **opción B** para corregir esto:
Originalmente habíamos establecido el valor mínimo del Tono (H) en “0”, lo cual hacía que también se eliminara el color Rojo interno de nuestros dados, y que aumentando el valor del mismo ese tono Rojo comenzaba a aparecer.
Gracias a esta aparición del color, también al aplicar la máscara, el dado internamente tenía más valores blancos “rellenos” (imagen por debajo), lo cual favoreció que al pasar un Kernel (mucho mas chico, 5x5), los dados se cerraran completamente sin complicaciones.



Conclusión:

Durante el desarrollo se presentaron diversos desafíos, principalmente relacionados con la correcta selección de los rangos HSV y con la estabilidad de las máscaras utilizadas para la detección de los dados.

A través de pruebas iterativas y ajustes en los parámetros de segmentación y morfología, se logró una solución robusta que evita falsos positivos y mantiene la forma geométrica de los dados sin unir objetos cercanos.

Como trabajo futuro, podría evaluarse la adaptación del sistema a distintos fondos o condiciones de iluminación, así como la optimización del conteo de puntos mediante técnicas más avanzadas de análisis de forma.