TECNICATURA UNIVERSITARIA EN INTELIGENCIA ARTIFICIAL (TUIA)

PROCESAMIENTO DE IMÁGENES (PDI)

Docentes de la cátedra:

Dr. Ing. Gonzalo Sad

Ing. Facundo Reyes

Ing. Julian Álvarez

Trabajo Práctico 3

Estudiantes (GRUPO 15):

Max Eder

Martín L. Perrone Leone

Pablo Pistarelli

AÑO 2023 (2do CUATRIMESTRE), Rosario.

INTRODUCCIÓN

En el siguiente informe presentamos el proceso de trabajo realizado para el TP3.

En el mismo abordamos el problema en el que se nos presentaban 4 archivos de video y debíamos desarrollar un algoritmo para detectar automáticamente cuando se detienen los dados y leer el número obtenido en cada uno.

Luego, generar un video para cada archivo donde los dados, mientras están en reposo, aparezcan resaltados con un bounding box de color azul y además, agregar sobre los mismos el número reconocido.

Funciones definidas:

- filtar_color_verde(img): recibe como argumento de entrada una imagen y devuelve una versión de ésta donde solo se conservan los píxeles que caen dentro del rango de color verde definido por los umbrales en formato HSV. El paso a paso está explicado en el código fuente.
- filtrar_color_rojo(img,inf,sup): filtra y resalta áreas de color rojo de la imagen (argumento img), devolviendo una versión de ésta donde solo se conservan los píxeles que caen dentro del rango de color rojo definido por los umbrales, ajustables mediante los argumentos inf y sup, en formato HSV. El paso a paso está explicado en el código fuente.
- calcular_diferencias_bbox(stats1, stats2): calcula y devuelve las diferencias entre los bounding boxes de dos conjuntos de stats para comparar la posición y el tamaño en los diferentes frames.
- filtrar_por_area(componentes_filtrados): filtra los componentes conectados según su área, conservando solo aquellos cuyas áreas están dentro del rango establecido por area_minima y area_maxima.

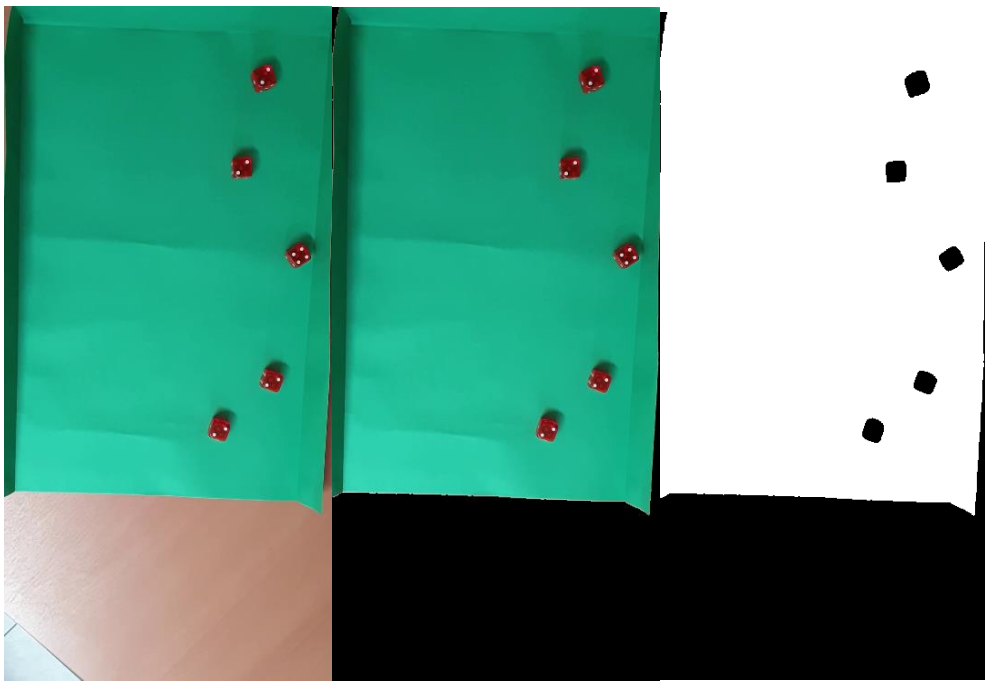
Lo primero que hicimos fue inicializar un objeto VideoCapture para leer el video y obtener las propiedades ancho, alto, FPS y número total de fotogramas.

Luego procesamos el video frame por frame aplicándole las siguientes operaciones a cada uno de ellos:

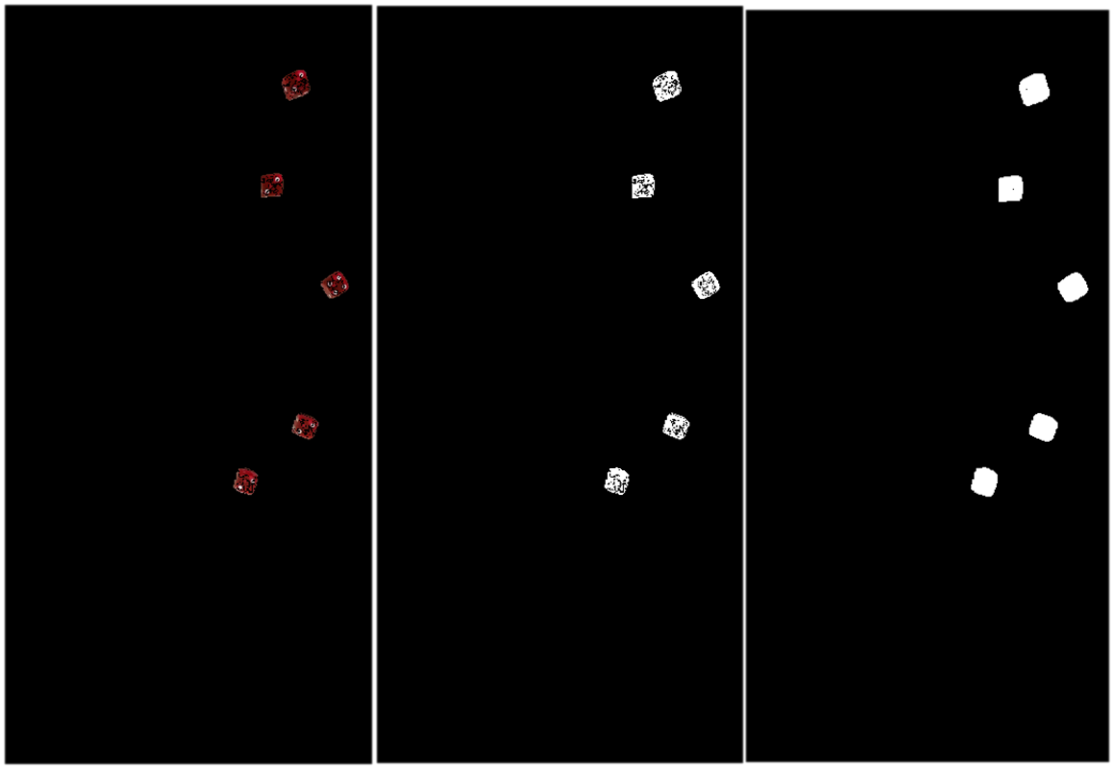
Redimensionamos a un tercio del tamaño para acelerar el procesamiento.

Aplicamos la función filtrar_color_verde() para obtener la máscara de áreas verdes de la imagen.

Utilizamos cv2.findcontours() y max() para obtener el contorno más grande (el paño) y generamos una máscara de color blanco.



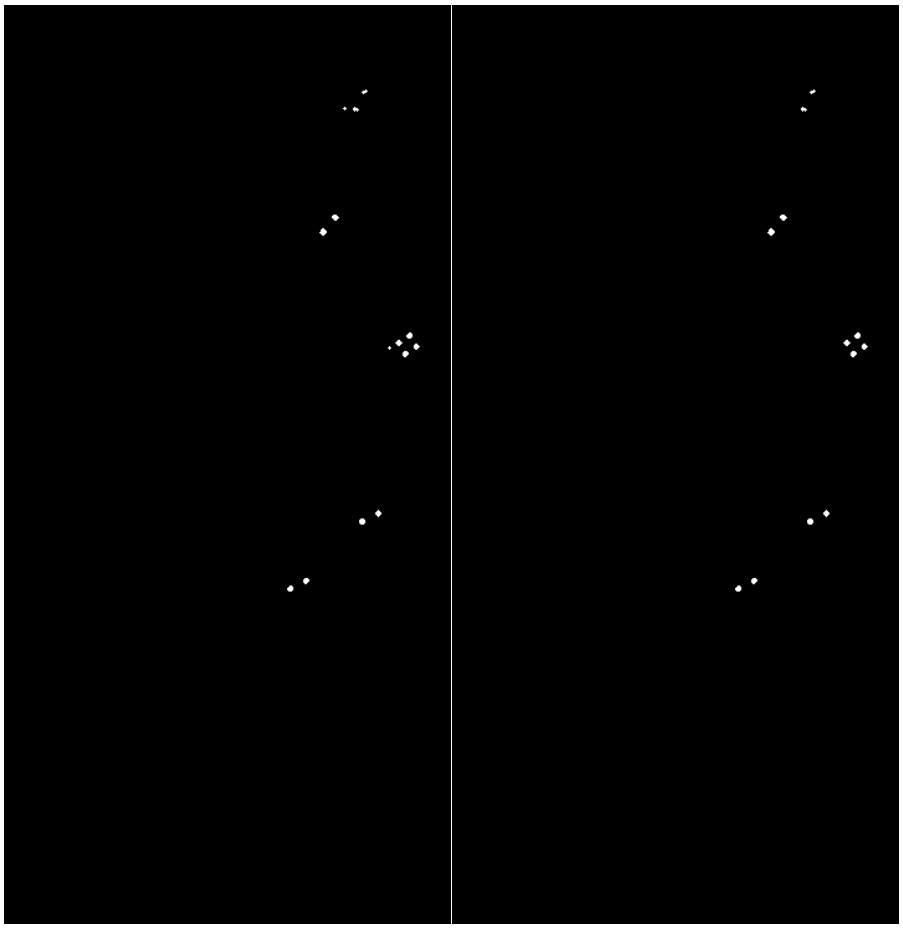
Aplicamos la función filtrar_color_rojo() para detectar áreas rojas dentro del área de interés definida por el contorno verde más grande. Se hace esto dos veces con diferentes umbrales para el color rojo y realizamos una operación OR bit a bit (cv2.bitwise_or()) entre ellas para obtener la máscara final de áreas rojas. Le aplicamos una dilatación a la máscara obtenida.



Luego utilizamos las etiquetas de componentes conectados para identificar los componentes. Si se encuentran exactamente 6 componentes, almacenamos la información relevante (máscara, número de frame, estadísticas y fotograma original) en listas para procesar las de interés posteriormente.

Recorremos la lista generada para los números de frames para buscar secuencias de tres frames consecutivos en los que las diferencias entre los bounding boxes de los componentes conectados sean menores que 2. Cuando se encuentran tales secuencias, almacenamos los índices correspondientes porque consideramos que no hay cambios o hay cambios mínimos en la posición de los dados entre frames consecutivos y con esto filtramos todas las listas que usaremos para el procesamiento.

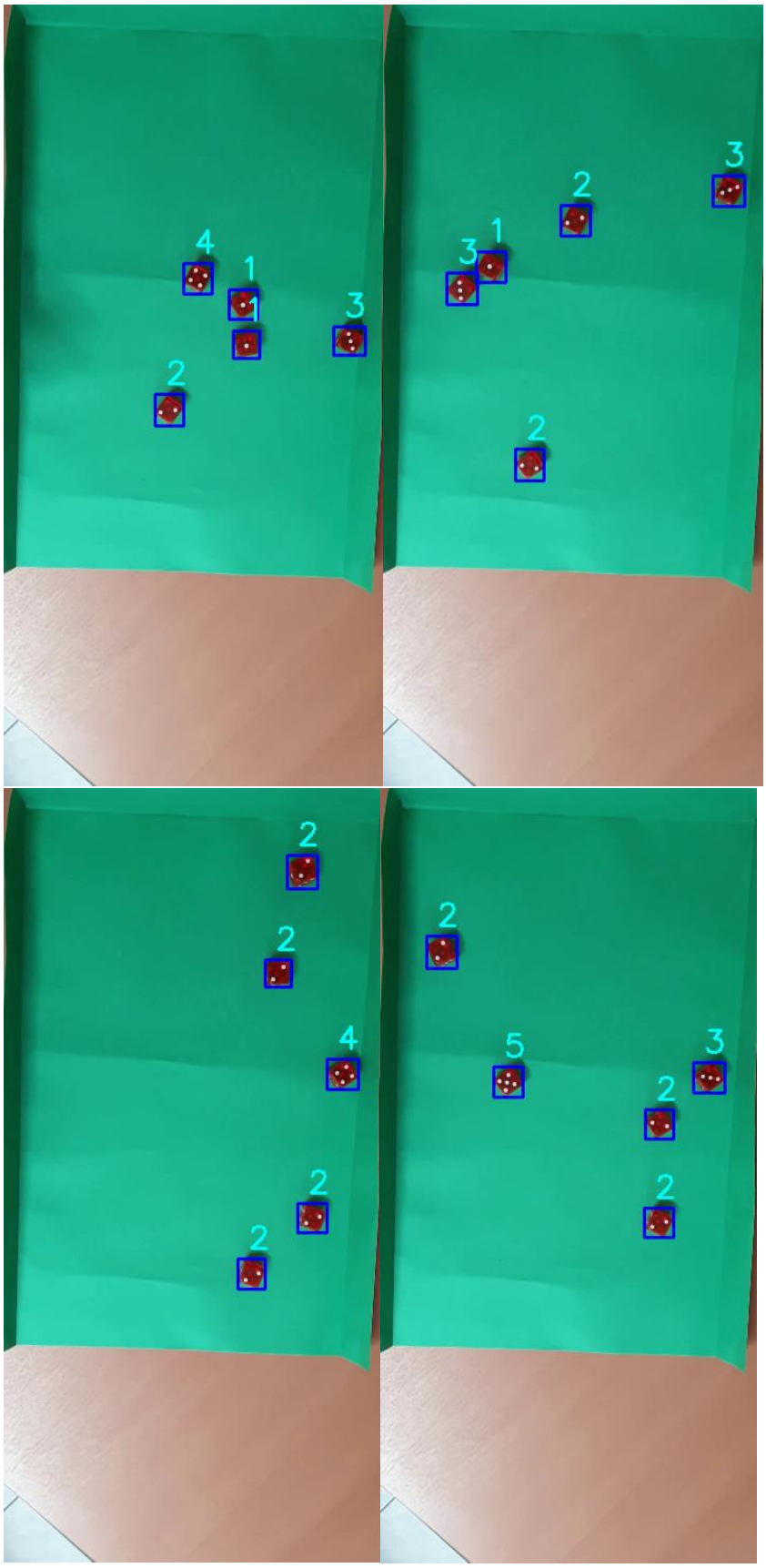
Luego, para cada uno de los frames, creamos una máscara para aislar cada dado, recortamos su región, la convertimos a escala de grises, le aplicamos umbralado para obtener los puntos, los dilatamos y los contamos.



El siguiente paso es tomar el video original, y en los frames que están dentro de la lista que identifica que los dados están quietos, les grabamos los bounding boxes y el valor de cada dado.

Por último mostramos el video definitivo.

En la siguiente página se muestra un frame de cada video con la información agregada.


4
1
3
2
3
2
1
3
2
2
2
2
4
2
2
2
2
5
3
2
2

Conclusión: nos resultó muy interesante trabajar sobre videos, desglosarlos en frames y editarlos y nuevamente tuvimos una muestra de aplicación de estas tecnologías. Esta vez el trabajo nos resultó un poco más sencillo porque requería de bastante de programación y ya conocíamos las técnicas de dilatación y umbralado para trabajar con las máscaras, pero eso no nos eximió de esfuerzo importante en uno de los videos en que en un dado nos leía un número de más y en otro un número de menos por lo que tuvimos que ingeniárnosla y probar mucho hasta dar en la tecla y que por fin los lea bien.