# PROCESAMIENTO DE IMÁGENES

## Universidad Austral Rosario

Trabajo Práctico Final: Cinco Dados

Profesor: Gonzalo Sad

Grupo nro. 2

**Integrantes**: Larreta Ricardo, Adolfo Moyano, Martin Jesús Fernández, Oriana Medina, Grippo Luisina, Drago Braian.

**Objetivo general del trabajo:** tomando como input una serie de videos de lanzamiento de dados, se propone elaborar un script documentado a través del cual se detecten e identifiquen los dados en el momento de reposo y se generen videos como output en los cuales se muestren los dados resaltados y el numero obtenido de cada uno.

En el siguiente informe se resume la metodología llevada adelante en función del cumplimiento del objetivo.

**Estrategia**:

Se planteó un enfoque compuesto por dos pasos integrales:

1. Inicialmente se buscó detectar que la imagen se encuentre en un estado de reposo. En este sentido, identificar directamente la detección de únicamente los dados en un estado de reposo resultó ser más práctico en términos de resolver problemas de iluminación, sombras y el movimiento de manos.

   Como métrica general para detectar el reposo se calcula el RMSE entre un frame anterior y el actual. (Luego, en una segunda instancia se realiza una optimización de este criterio utilizando una máscara binaria que representa la posición de los dados en lugar del frame RGB)

2. Una vez resuelta la detección, se procede a identificar los valores de cada dado.

**Descripción de los pasos ejecutados y los resultados parciales y finales** (en el script se encuentran los detalles documentados)**:**

1. **Importación de Librerías**
   Esencialmente se utilizaron las siguientes librerías para la realización del trabajo: **numpy y cv2**

2. **Bucle principal y secundario**
   El script integral consiste en un bucle que procesa de manera iterativa los videos originales extrayendo algunas propiedades significativas y realizar los procesamientos correspondientes frame a frame.

   Al inicio del bucle se definen algunas variables importantes que se utilizan a lo largo del programa:

   Máscaras:
   Se genera una máscara en tres canales inicialmente en negro sobre la cual se mostrarán luego los dados en blanco. Posteriormente se guarda la máscara de los dados del frame anterior al que se está procesando. Esto se crea para luego calcular la métrica de RMSE (**mask_dados** y **mask_dados_ant**)

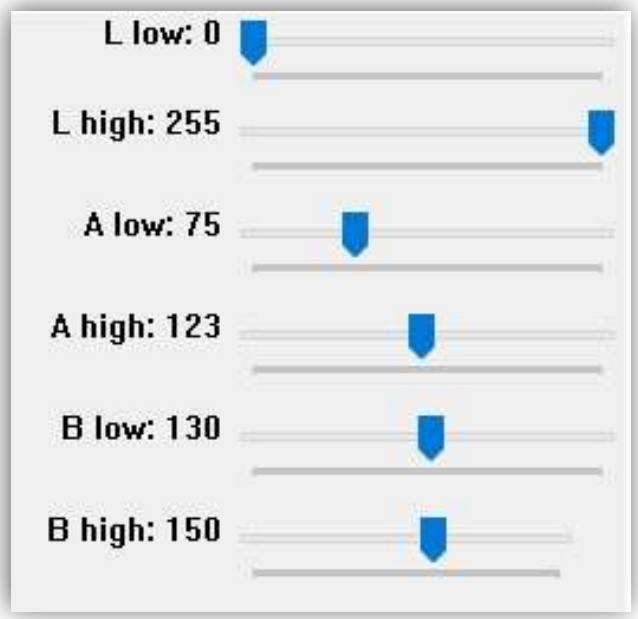
Frames:
Se genera una copia del frame auxiliar a modificar en los procesamientos sucesivos (**frame_pre**). Además, se define otra copia del frame la cual se exporta en el video de salida y se le agregan anotaciones (**frame_out**)

3. **Filtrado de frames**

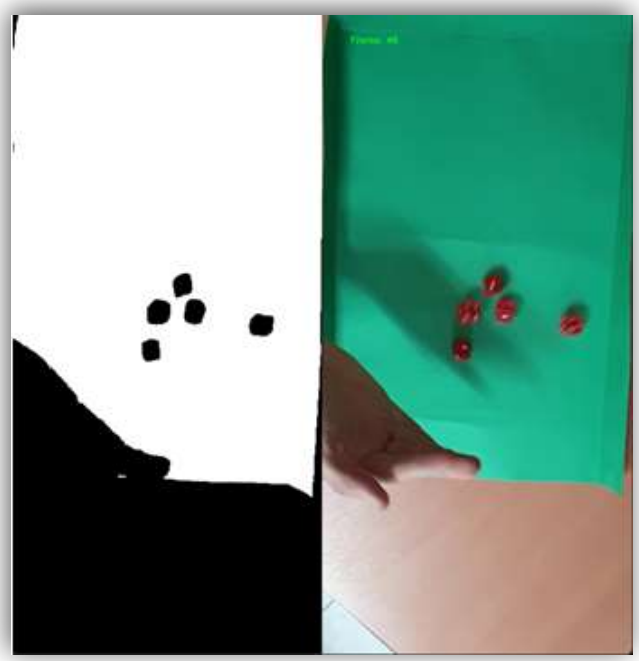
A continuación, se utilizan como inputs los parámetros encontrados en el Script anexo de **filtrado cielab**. En este sentido, inicialmente se utilizó este código para procesar un archivo de video y mostrarlo junto con algunos controles de filtrado en una interfaz gráfica de usuario (GUI). (comentario: en el script **filtrado_cielab** se definen rangos de valores para las barras de seguimiento y los valores alto y bajo para cada uno de los canales de color L, A y B en el espacio de color LAB. LAB: Tono, Luminosidad(oscuro/claro) y pureza(apagado/vivo).

El espacio de color LAB se usa comúnmente en el procesamiento de imágenes para representar los colores de una manera que es más uniforme desde el punto de vista de la percepción que el espacio de color RGB). Este espacio es menos susceptible a cambios de luminosidad.

A continuación, se muestra una imagen con los parámetros seleccionados:




Mediante la función **cv2.cvtColor** se convierte el frame de RGB al espacio LAB. La función cv2.inRange luego crea una máscara binaria donde los colores que se encuentran dentro del rango de color son negros y los que están fuera del rango son blancos. **Particularmente, se buscó representar todo objeto rojo (dados) en negro y todo el resto (lo que no es rojo) se vea blanco**.

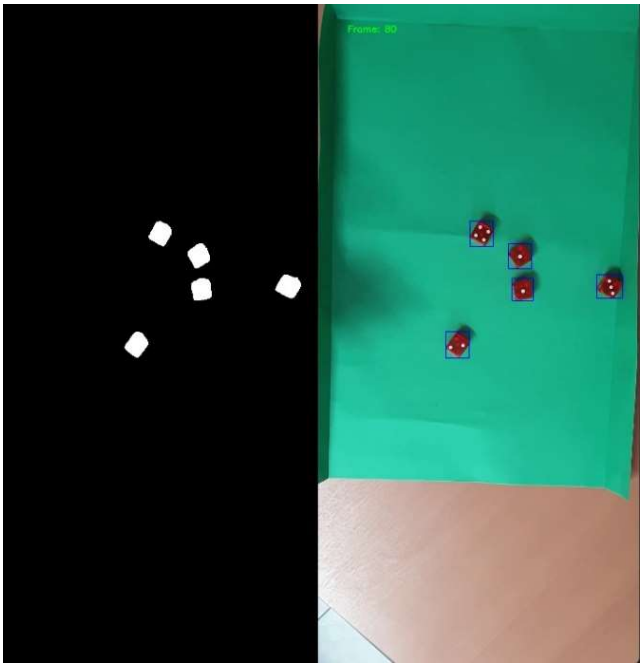


Ventaja de usar este filtrado: la sombra de la mano no afecta la identificación de los dados.

4. **Identificación de Contornos**:
   Inicialmente encontramos los contornos en la imagen con umbral binario
   **contours, hierarchy = cv2.findContours(frame_threshold, cv2.RETR_LIST, cv2.CHAIN_APPROX_NONE)**

   Se recorren cada uno de los contornos que se encuentran en la imagen con umbral y se extraen varias propiedades de cada contorno (área, relación de aspecto y coordenadas del rectángulo delimitador). Posteriormente, se verifica si el contorno cumple con ciertos criterios para ser un objeto "dados" (lo que buscamos detectar).
   Al cumplir los criterios definidos se dibuja un cuadro delimitador alrededor del objeto en el frame actual mediante cv2.rectangle. Además, se rellena el contorno con píxeles blancos en la imagen mask_dados mediante cv2.drawContours y se incrementa un contador para el número de objetos "dados". encontrado (**count_dados**).
   Detalle de los descriptores geométricos utilizados: - Área y Relación de Aspecto (Alargamiento):

```
if (area > 4000 and area < 6000 and aspect_ratio>=0.85 and aspect_ratio<=1.15):
   _ = cv2.rectangle(frame_out, (x,y), (x+w,y+h), color=(255, 0, 0), thickness=2)
   _ = cv2.drawContours(mask_dados, contours, contourIdx=ii, color=(255, 255, 255),
thickness=cv2.FILLED)
count_dados += 1
```




5. **Cálculo de RMSE**

   Acorde a lo mencionado anteriormente se procede a ir calculando el RMSE entre **mask_dados actual y mask_dados_ant (máscara anterior).**

   La condición es que, si se tiene un frame previo, se compara el mismo con el actual vía RMSE y se determina si la imagen está en reposo.

   Además, se establece la condición de cálculo de RMSE únicamente si el conteo de dados es igual a 5.

   Finalmente, si la cantidad de dados es 5 y el RMSE es menor a 10, implica reposo y se recuadra con un rectángulo amarillo el frame. A partir de aquí, iniciaría el procesamiento para la determinación de los valores de los dados, que se detalla en el siguiente punto.

   **if count_dados==5 and rmse < 10:**
   **reposo = True**
   **cv2.rectangle(frame_out, (5,5), (width-5,height-5), (0,255,255), 5)**

6. **Determinación de los valores de los dados**

   En este punto se parte de la condición de reposo obtenida en el punto 5 y se realizan tres procesamientos.

   Aquí se parte de la premisa de delimitar las fronteras entre fondo de imagen con los dados y además los dados propiamente dichos con sus puntos. Es decir: Fondo blanco, dados negros, puntos blancos.

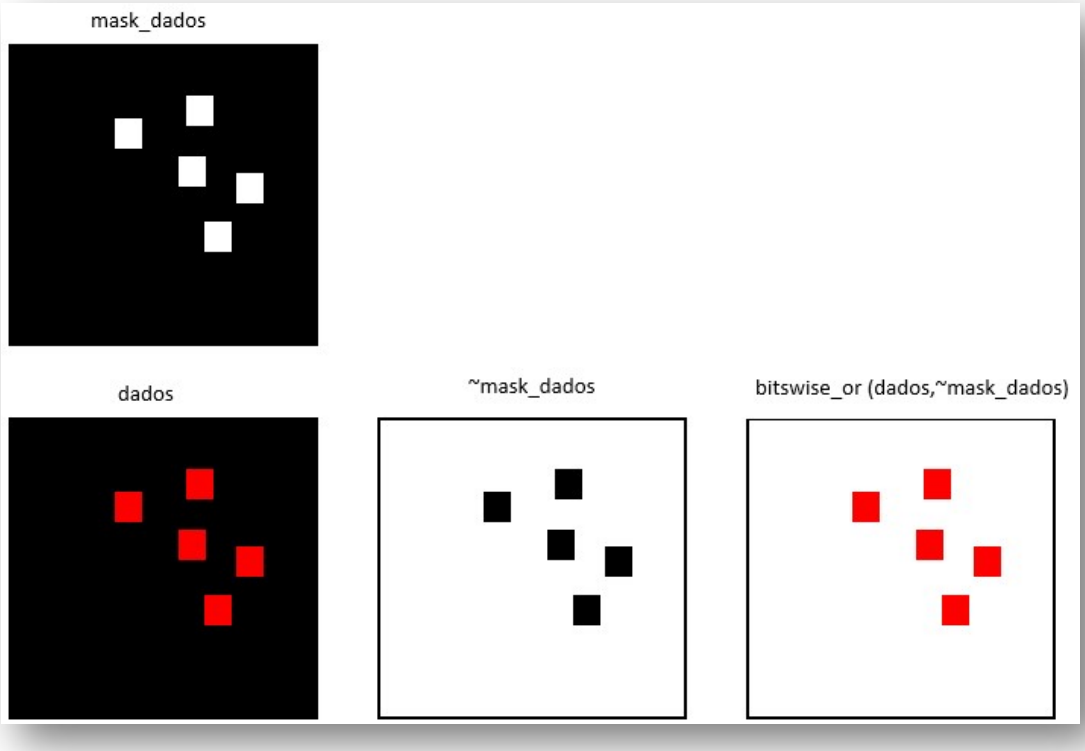
   - obtenemos una imagen toda negra con los dados originales

   **dados = cv2.bitwise_and(frame, frame, mask=cv2.cvtColor(mask_dados, cv2.COLOR_BGR2GRAY))**

- Se aplica una operación OR entre la máscara inversa y el resultado anterior con el objetivo de lograr un fondo blanco con los dados originales.

**frame_pre = cv2.bitwise_or(dados, ~mask_dados)**

Esquema representativo del procesamiento realizado en este punto:




Luego, se aplican filtros de mediana, umbralado y clausura para mejorar la identificación de los puntos.
Finalmente, se procede a encontrar los contornos (dados y puntos dentro de los dados).

**Filtro de Mediana**

frame_pre = cv2.medianBlur(frame_pre, 7)

**Umbralado**

umbral, frame_pre = cv2.threshold(frame_pre, thresh=150, maxval=255, type=cv2.THRESH_BINARY)

A continuación, la imagen binaria se somete a un **cierre morfológico** para rellenar los huecos y suavizar los bordes.

**Clausura**

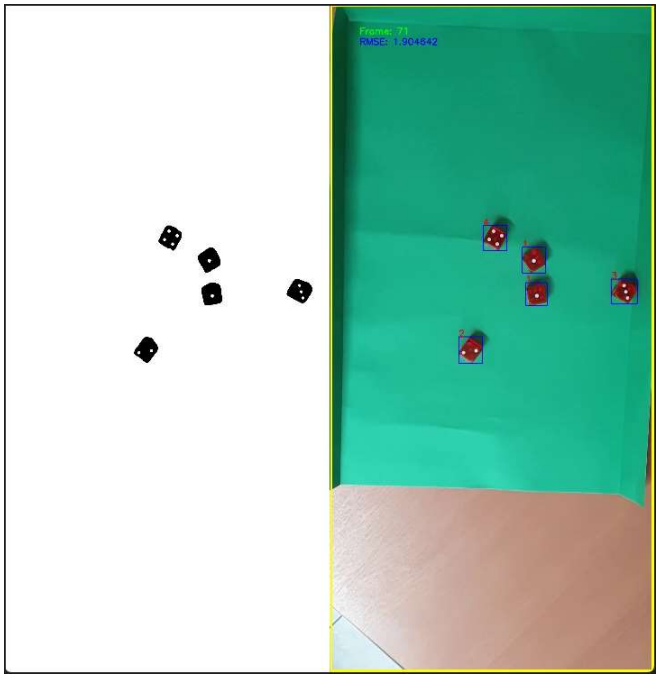
frame_pre = cv2.morphologyEx(frame_pre, cv2.MORPH_CLOSE, cv2.getStructuringElement(cv2.MORPH_RECT,(3,3)))

**Contorno**

Luego de los procesamientos efectuados, las formas finales solo responden a dados y puntos, por lo cual considerando a los dados como hijos del contorno principal (hierarchy[0][ii][3]==0), se procede a contar los contornos hijos de cada dado, es decir, los puntos.

## 7. Visualización

Se procede a visualizar las imágenes finales incorporando (mediante la función cv2.putText) los números de cada dado




## 8. Optimización del cálculo de reposo

Se plantea la realización de una nueva versión del script anterior con una serie de mejoras. El objetivo principal perseguido está centrado en evitar el calculo del RMSE entre dos imágenes y utilizar la información provista por los contornos de los dados para definir el estado de reposo.

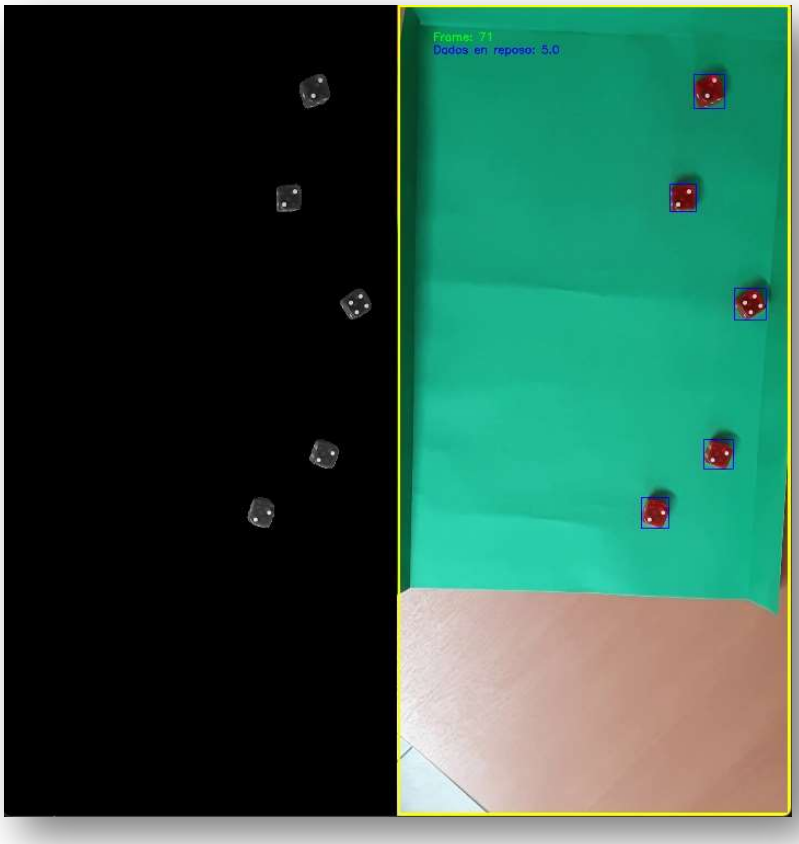
En primer lugar, todos los parámetros utilizados y ajustados para procesar correctamente estos cuatro videos se definen al principio del programa. De esta manera, se mejora su reproducibilidad y extensión a otros videos similares. Los parámetros en cuestión son:

TOL_REPOSO: tolerancia para detectar movimiento de los dados en función del cambio de su posición.
LOW_COLOR: nivel de color inferior del espacio CIELAB para filtrar los dados.
HIGH_COLOR: nivel de color superior del espacio CIELAB para filtrar los dados.
LOW_AREA: umbral inferior del área del contorno que se identifica como dado.
HIGH_AREA: umbral superior del área del contorno que se identifica como dado.
LOW_ASPECT_RATIO: límite inferior del alargamiento del contorno que se identifica como dado.

HIGH_ASPECT_RATIO: límite superior del alargamiento del contorno que se identifica como dado.
MEDIAN_BLUR: tamaño del filtro utilizado previo a la detección de los puntos de los dados.
THRESHOLD: umbral utilizado para binarizar la imagen previa a la detección de los puntos de los dados.
CLOSE_ELEMENT: Elemento estructural utilizado al aplicar la operación de clausura previo a la búsqueda de contornos de los puntos de los dados.
FRAME_STEP: frecuencia de actualización del frame anterior para la comparación con el frame actual y la detección de movimiento.
Videos: lista con los archivos de video a procesar.

En segundo lugar, se actualiza el método para calcular el estado de reposo de los dados.
La comparación se realiza sobre los cambios de posición del vértice superior del bounding box de los objetos identificados como dados. Para esto, se guarda en dos listas (**ls_dados** y **ls_dados_ant**) los contornos identificados siguiendo el mismo criterio que el programa anterior. Los parámetros **TOL_REPOSO** y **FRAME_STEP** son los que definen que secuencia de imágenes se consideran en reposo.

Detectado el reposo, se procede trabajar con una imagen de los dados en fondo negro, producto de utilización de la máscara de dados confeccionada anteriormente.



Luego de procesar y umbralar, la imagen resultante cuenta ahora con solo los puntos blancos de los dados. A partir de ahí se buscan los contornos de esos puntos y a través de la obtención de medidas invariantes de momentos, se determina el centroide de cada punto. Para cada uno de estos puntos, se recorren todos los contornos de los dados obtenidos en la etapa anterior y se verifica si el centroide está contenido en el dado. De ser así se incrementa un contador para ese dado que al finalizar esta etapa se muestra en el video final.