Detección y Clasificación de Monedas y Dados

Desarrollamos un algoritmo capaz de **detectar monedas y dados**, clasificarlos correctamente y, en el caso de los dados, **determinar cuántos puntos tienen en la cara superior**, todo a partir de una única imagen con iluminación no uniforme.

Qué hicimos

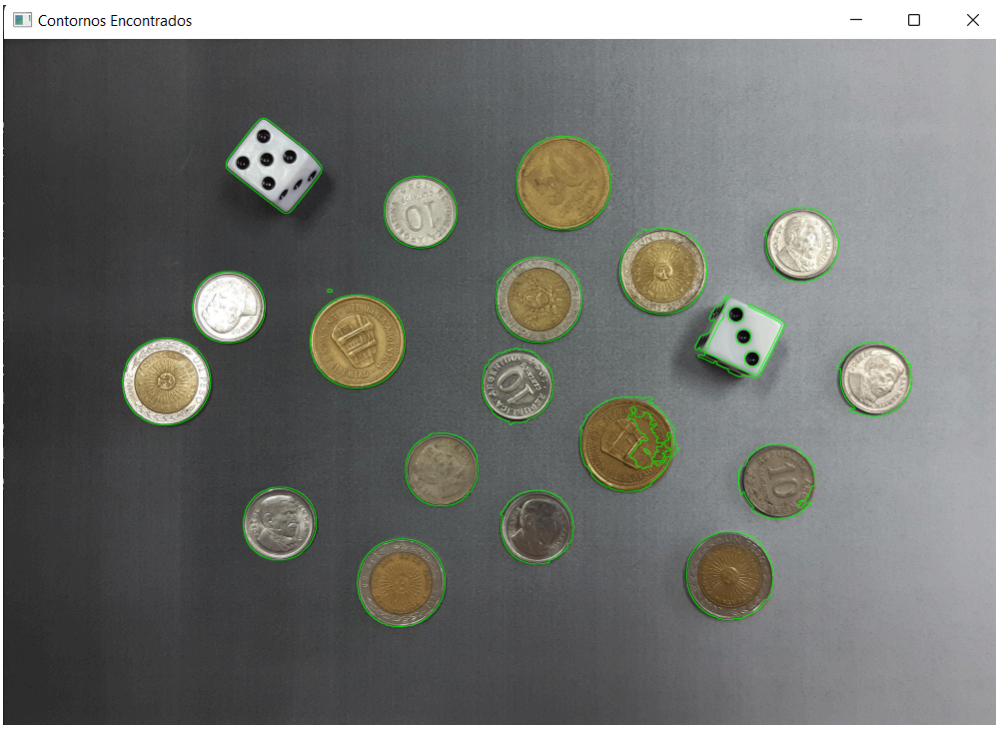
Primero cargamos la imagen y la convertimos a escala de grises para simplificar el procesamiento. Aplicamos un **suavizado Gaussiano** y el detector de bordes **Canny**, lo que permitió resaltar los límites de monedas y dados.
Después realizamos dos operaciones morfológicas importantes: una **dilatación**, para conectar bordes cercanos, y un **cierre** con un kernel grande, cuyo propósito fue rellenar huecos y dejar cada objeto completamente cerrado.

Con esta imagen ya “limpia”, detectamos contornos usando únicamente los contornos externos (RETR_EXTERNAL) y luego los filtramos por área.
Posteriormente clasificamos cada contorno en **moneda o dado**, analizando su **circularidad** y la **relación de aspecto**. Para las monedas, además clasificamos su valor aproximado según el **ancho del bounding box**, que actúa como una estimación del diámetro.

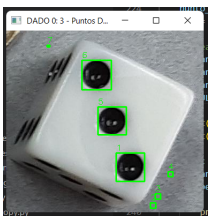
Finalmente, para cada dado obtuvimos un ROI y realizamos un proceso de segmentación específico para detectar sus puntos. Luego de filtrar contornos por **área** y **circularidad**, contamos los puntos detectados y los mostramos sobre la imagen.

Qué problemas tuvimos

Uno de los primeros problemas fue que los bordes de muchas monedas eran irregulares debido a la iluminación y a la textura del fondo, lo que hacía que los contornos quedarán abiertos o incompletos. Esto generaba errores tanto en la detección como en la clasificación. Como se ve en la siguiente imagen, en las **Primeras Etapas,** los bordes de las monedas no eran correctos



Otro problema fue distinguir correctamente los puntos de los dados: algunos se detectaban como manchas y otros se perdían por falta de contraste o por pequeñas rupturas en la segmentación. Además, cada dado ocupa un área diferente, por lo que el tamaño de los puntos varía según la región recortada.



También surgió un inconveniente inicial al intentar clasificar monedas pequeñas (10 centavos) que tenían contornos bastante deformados; su circularidad no alcanzaba para clasificarlas correctamente.

Qué técnicas aplicamos para resolverlo

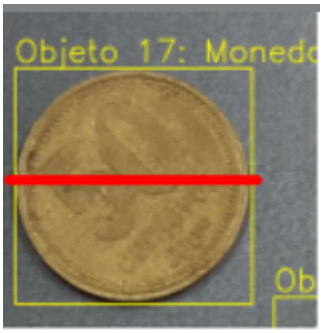
Para corregir la fragmentación de los contornos, aplicamos una **dilatación** y posteriormente una **clausura morfológica** con un kernel grande, lo que aseguró que cada moneda y cada dado quedaran como regiones sólidas y separadas del fondo.

Ademas, utilizamos la función de findContourns de RETR_EXTERNAL para localizar los bordes externos sin tomar los internos

Para distinguir monedas de dados utilizamos dos medidas geométricas:

- **Circularidad**: útil para detectar objetos casi circulares (monedas).
- **Relación de aspecto** del bounding box: los dados tienden a tener aspecto más cuadrado y circularidad más baja.

Para clasificar monedas por valor aplicamos una técnica simple pero efectiva: usar el **ancho del bounding box** como aproximación al diámetro real, estableciendo umbrales entre los distintos tipos.



En cuanto a los puntos de los dados, utilizamos:

- **Umbralización Otsu inversa** para separar los puntos oscuros.
- **Apertura morfológica** para eliminar manchas pequeñas que generaban ruido.
- **Clausura** para cerrar los círculos de los puntos y obtener contornos más definidos.
- **Filtrado por área y circularidad**, asegurando que solo se detecten puntos reales y no bordes/puntos laterales o manchas internas.

Estas técnicas permitieron lograr una detección consistente tanto de monedas como de puntos en los dados, incluso con diferencias de iluminación y tamaños.