Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Tecnicatura Universitaria en Inteligencia Artificial

PROCESAMIENTO DE IMÁGENES I

IA 4.4

Año: 2025

Integrantes:

- Tapia, Fabrizio Joaquín T-3095/3
- Mezzano, Florencia M-7463/2

Fecha de entrega: 14/12/2025

Docentes:

- Gonzalo Sad
- Juan Manuel Calle
- Joaquín Allione

Informe Técnico

1. Introducción

En este trabajo práctico abordamos el análisis de cuatro videos de tiradas de dados. El objetivo fue, primero, identificar automáticamente un momento de reposo confiable para poder trabajar con una imagen estable, y luego construir un pipeline que detecte los dados y cuente sus pips de forma consistente. Finalmente, extendimos el resultado para generar un video de salida donde, únicamente mientras los dados están quietos, se muestre cada dado con su bounding box, un identificador y el valor detectado.

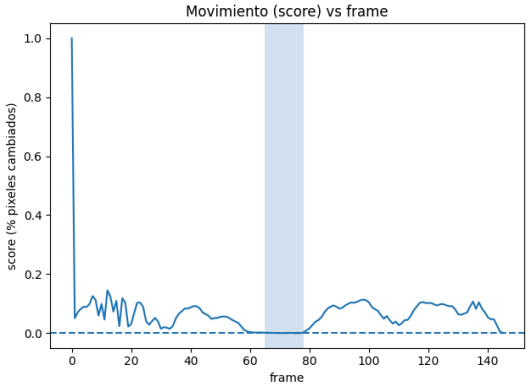
2. Parte A

I. Detección de reposo

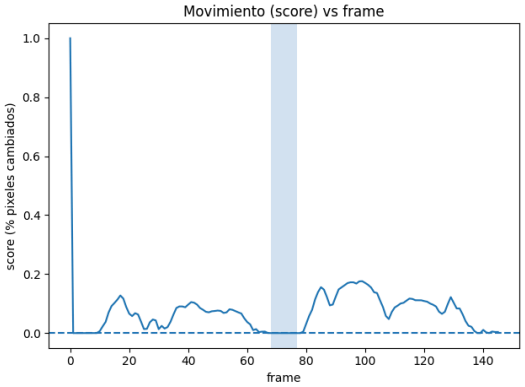
El primer paso fue detectar automáticamente cuándo el video entraba en reposo. Para eso calculamos un “score de movimiento” frame a frame, comparando imágenes consecutivas con una diferencia absoluta y midiendo qué proporción de píxeles cambiaba por encima de un umbral. Al principio, al visualizar los mapas de diferencia, se veían casi negros y pensamos que era un error; luego entendimos que era esperable, porque en reposo la diferencia entre frames es mínima. Por esa razón decidimos no conservar esas imágenes intermedias y quedarnos solo con evidencias más útiles, como el gráfico del movimiento y el frame final elegido.

Al probar el método en los cuatro videos, en varios casos no aparecían segmentos de reposo. Esto no se debió a un problema de lectura sino a que el criterio era demasiado estricto frente a cambios pequeños del video (ruido, compresión o ajustes automáticos de la cámara). Lo resolvimos achicando el frame antes de medir el movimiento para hacerlo más estable y sumamos un “fallback” en el percentil del umbral, probando valores más tolerantes cuando no se detectaba reposo. Con eso logramos detectar reposo en todas las tiradas.

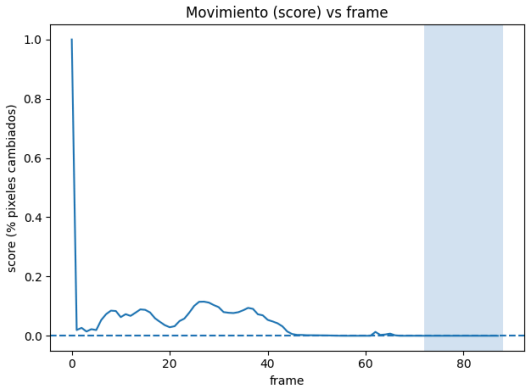
En la tirada 2 surgió un caso particular: había dos reposos y el primero no correspondía al estado final de los dados. En vez de hardcodear una excepción, iteramos por los segmentos detectados y seleccionamos automáticamente el primero que permitía detectar los cinco dados, manteniendo el enfoque general y sin agregar complejidad innecesaria.



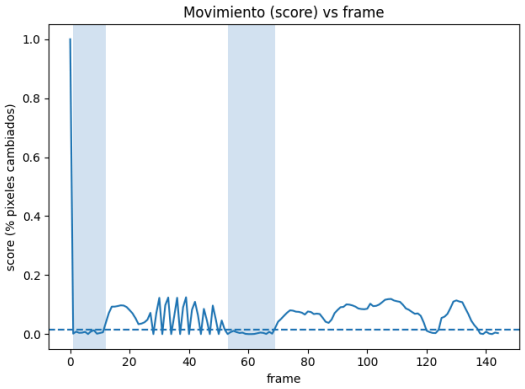
Score vs Frames: Tirada 1



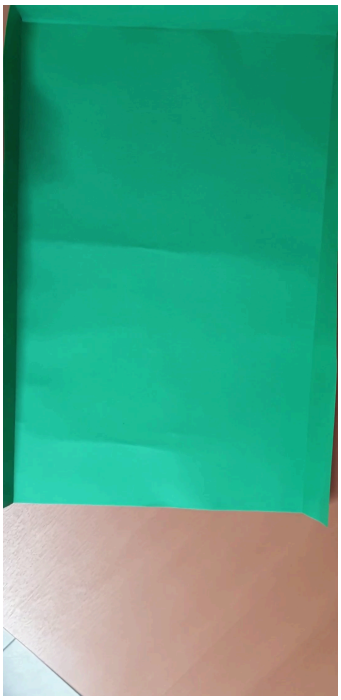
Score vs Frames: Tirada 2



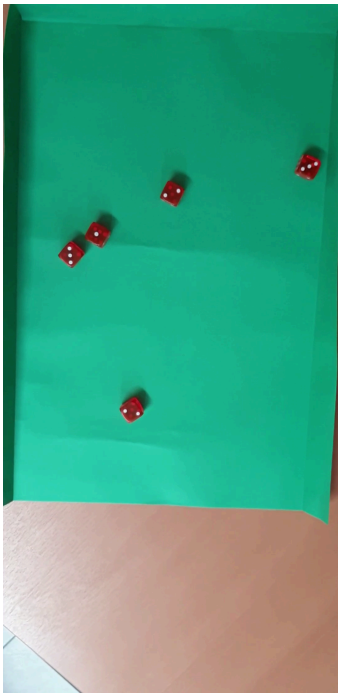
Score vs Frames: Tirada 4



Score con dos reposos: Tirada 2



Primer Reposo: Tirada 2

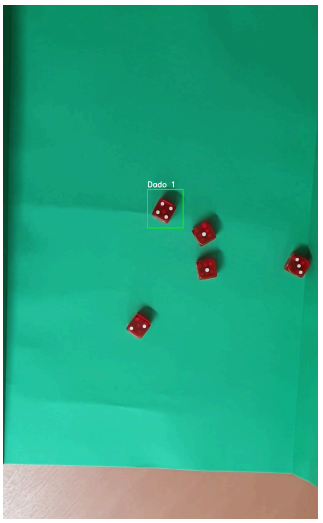


Segundo reposo: Tirada 2

II. Detección de dados

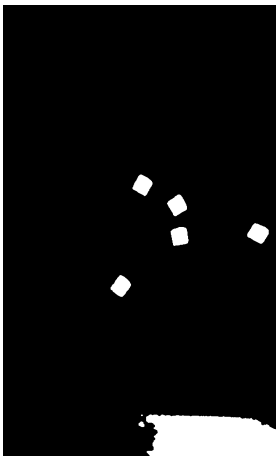
Una vez detectado el reposo, pasamos a localizar los dados. Al principio intentamos hacerlo con bordes y contornos, pero aparecía un falso positivo grande en la parte inferior del frame (producido por la mesa o el borde de la escena) y, además, la detección no era estable: en varias pruebas no lograba reconocer todos los dados y en algunos casos terminaba detectando solo uno. Para estabilizarlo, aplicamos un recorte porcentual fijo que eliminó esa zona inferior y dejó únicamente el área útil.

Falsa detección positiva de dado

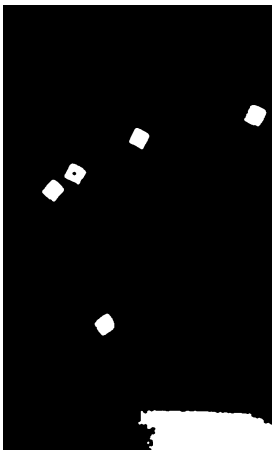


Detección ineficiente

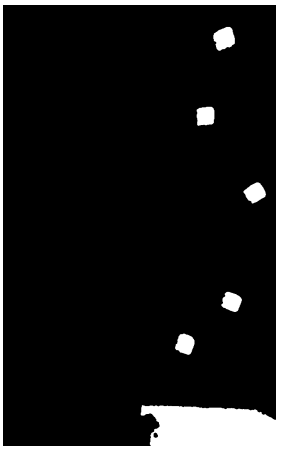
Luego mejoramos la detección cambiando de estrategia: en lugar de depender de bordes, segmentamos el color rojo de los dados usando HSV (HUE), obteniendo una máscara mucho más limpia y consistente. A partir de esa máscara buscamos contornos y filtramos candidatos por tamaño relativo y forma aproximadamente cuadrada, lo que redujo errores y evitó que elementos del fondo se confundieran con dados. También ajustamos la depuración visual para que las imágenes se vieran en un tamaño cómodo, redimensionando solo para mostrar sin alterar el procesamiento.



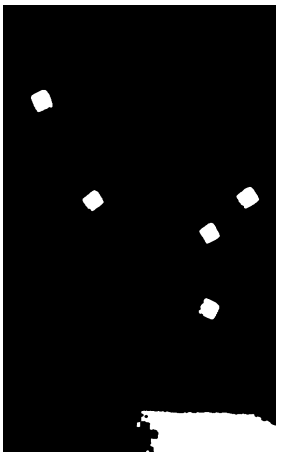
Máscara binaria: Tirada 1



Máscara binaria: Tirada 2



Máscara binaria: Tirada 3



Máscara binaria: Tirada 4

Finalmente, pensando en el punto b, corregimos un detalle clave: las bounding boxes se calculaban en el ROI recortado y quedaban corridas al dibujarlas sobre el frame completo. La solución fue conservar los offsets del recorte y convertir las coordenadas a valores globales, asegurando que las cajas quedaran correctamente alineadas en el video final.

III. Conteo de pips

Para el conteo de pips reutilizamos una idea de trabajos previos: binarizar, limpiar con morfología y filtrar contornos por tamaño y circularidad. Al principio, con un enfoque basado solo en Otsu sobre gris, algunos pips no se detectaban bien, especialmente en casos límite. Ajustamos esto hacia un criterio más estable para esta escena: detectar pips en HSV como regiones de baja saturación y alto valor dentro del dado, y luego aplicar los filtros geométricos. También evitamos hardcodear áreas absolutas, que dependen del tamaño del crop y del zoom, y pasamos a definir rangos de área relativos al área del ROI del dado. Con esa mejora logramos resultados correctos casi en la totalidad de los casos; en el caso aislado donde faltaba un pip, el problema venía de la propia máscara del dado que dejaba un “agujero” justo en la zona clara del pip. Lo solucionamos fortaleciendo la máscara del dado con una dilatación suave, de modo que el pip no quedara fuera del área válida.

---

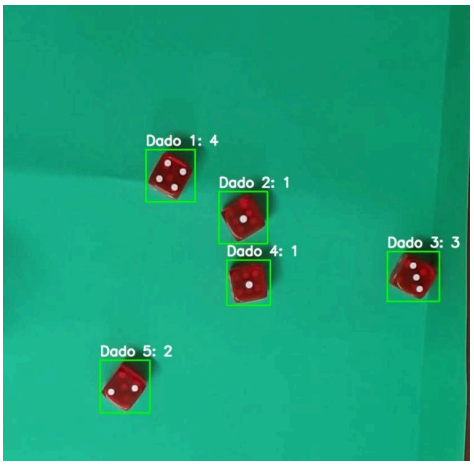
3. Parte B

I. Generación de video anotado en reposo

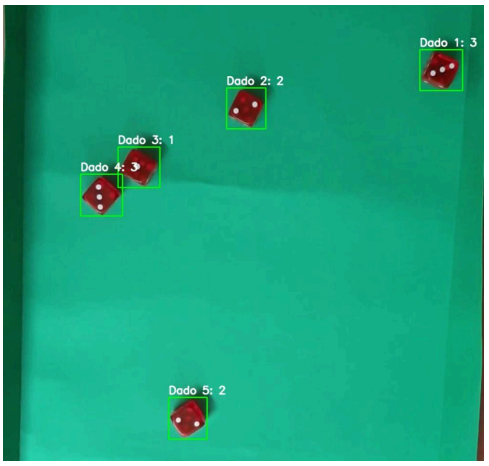
En la segunda parte debíamos generar un video por archivo donde, mientras los dados estén en reposo, aparezcan dibujados su bounding box, un nombre y el valor obtenido. Para hacerlo eficiente evitamos recalcular detección y conteo en cada frame. En cambio, reutilizamos la detección de reposo: para cada segmento de reposo tomamos un frame representativo, detectamos dados y valores una única vez, y guardamos esas anotaciones

para ese segmento. Luego recorremos el video frame a frame y, si el frame pertenece a un segmento anotado, dibujamos las cajas y textos sobre el frame antes de escribirlo al video de salida. De este modo la anotación es estable durante todo el período quieto y el proceso es rápido.

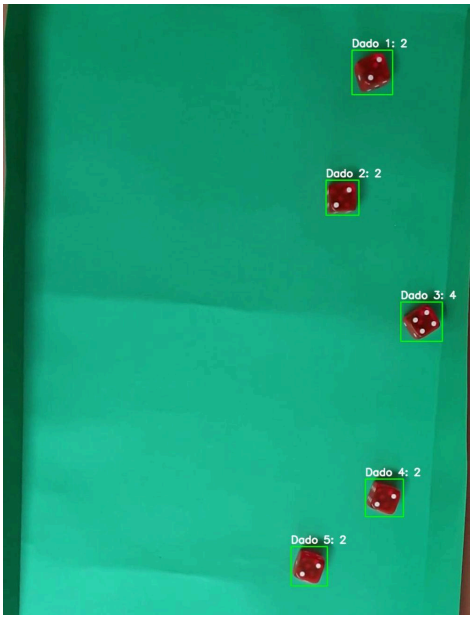
Durante las pruebas en algunas tiradas las cajas aparecían por un intervalo demasiado corto aun cuando visualmente los dados ya estaban quietos. Lo resolvimos agregando un margen de frames al momento de decidir si un frame pertenece al segmento de reposo, extendiendo levemente el inicio y el final del reposo para que el overlay coincida mejor con lo que se percibe en el video. Con estas correcciones, cada video de salida queda limpio: durante el movimiento no se muestra nada y, durante reposo, cada dado aparece identificado con su bounding box, su nombre y su valor.



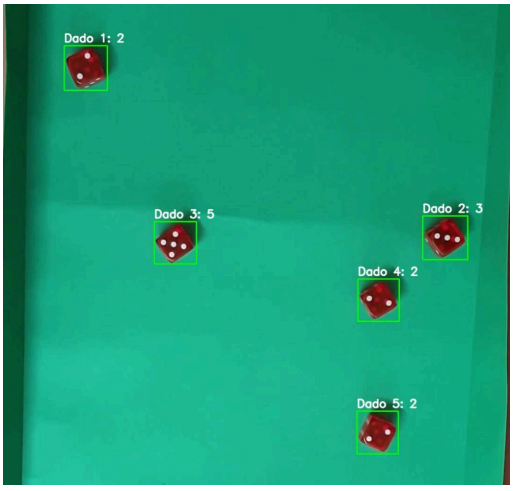
Recorte Bounding Box Tirada 1



Recorte Bounding Box Tirada 2



Recorte Bounding Box Tirada 3



Recorte Bounding Box Tirada 4

4. Conclusión

En este trabajo logramos construir un pipeline completo y consistente para analizar las tiradas: primero identificamos automáticamente los momentos de reposo, luego detectamos los cinco dados de forma estable y contamos sus pips con buen nivel de acierto, evitando depender de parámetros frágiles ante cambios de escala. A partir de esa base, generamos un video de salida por cada archivo donde, durante el reposo, se muestran las cajas, el identificador y el valor de cada dado, cumpliendo el objetivo del enunciado. En el proceso fuimos corrigiendo problemas reales de datos y de implementación (ruido del video, falsos positivos, reposos múltiples y coordenadas en ROI), priorizando siempre una solución simple, verificable y fácil de justificar con las evidencias visuales del paso a paso