TRABAJO PRÁCTICO INTEGRADOR N°2

IA4.4 Procesamiento de Imágenes

Tecnicatura Universitaria en Inteligencia Artificial

Trabajo realizado por:
Katia Otterstedt, Lautaro Florenza y Sebastián Palacio

Problema 1 - Detección y clasificación de monedas y dados

Introducción:

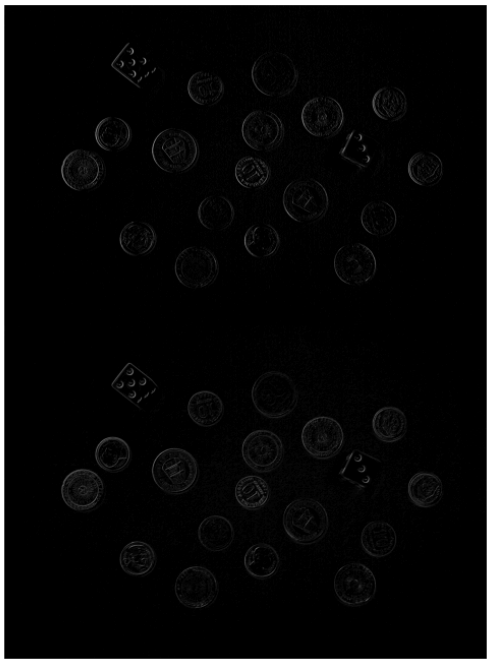
En este ejercicio trabajamos con la imagen del archivo monedas.jpg, la cual consiste en un fondo de intensidad no uniforme sobre el cual se hallan dados y monedas de distinto valor y tamaño. Nuestro objetivo fue elaborar un algoritmo capaz de procesar la imagen para clasificar los distintos tipos de monedas, realizando un conteo automático; y determinar el número de la cara superior de cada dado, igualmente realizando un conteo automático.

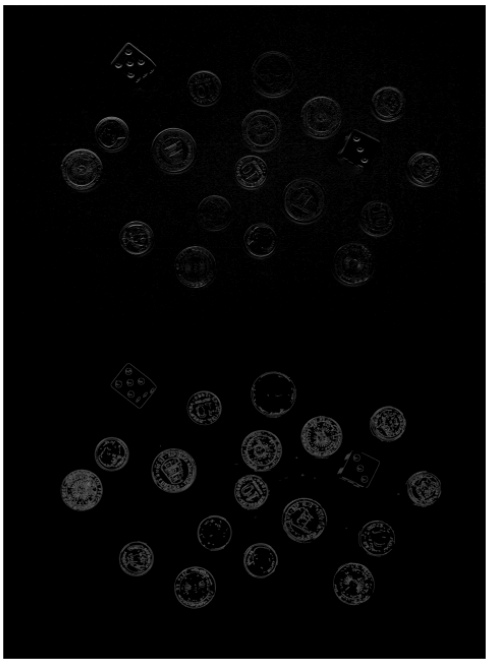
Desarrollo:

El primer paso fue importar la imagen, pasarla a escala de grises, para poder procesarla con algoritmos de detección de bordes como Sobel y Canny; aplicar un filtro gaussiano, para reducir el ruido previo a la detección de bordes, intentando evitar la detección de bordes falsos.
Los parámetros seleccionados para Canny fueron:
- Umbral inferior: 35
- Umbral superior: 90
- Kernel gaussiano: 7x7

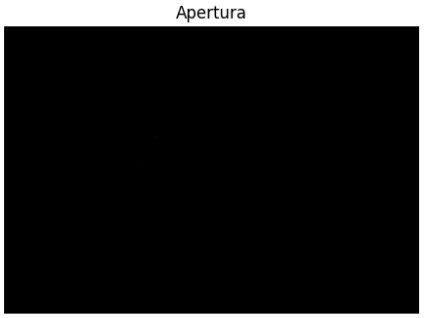
Estos valores se determinaron experimentalmente para maximizar la detección de bordes relevantes (monedas y dados) minimizando el ruido de fondo.

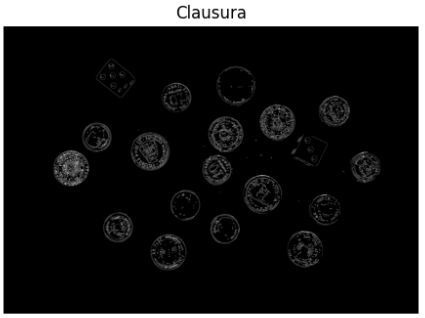
Utilizamos Sobel de forma exploratoria, pero seleccionamos el resultado de Canny para continuar con el procesamiento, dado que en adición al cálculo de gradiente que realiza Sobel, también adelgaza los bordes y realiza un proceso de umbralización con histéresis.

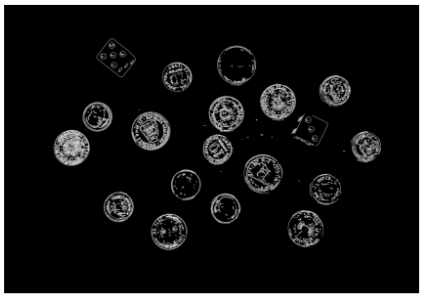




Una vez detectados los bordes, utilizamos un kernel circular de 3x3 píxeles para realizar operaciones morfológicas. Aplicamos un proceso de clausura, obteniendo la imagen que posteriormente se utilizará para la detección de dados. Luego dilatamos esta imagen, para lograr que los círculos se vuelvan más visibles para posteriormente aplicar la transformada de Hough en la detección de monedas.





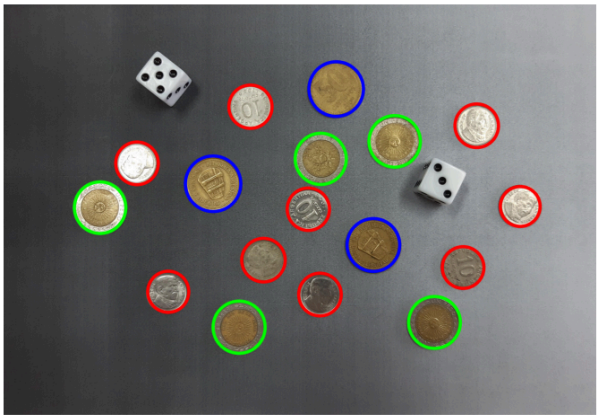


Con la imagen dilatada en el paso anterior, aplicamos la transformada de Hough para detectar círculos. Utilizamos una distancia mínima de 300 píxeles, para evitar detectar la misma moneda más de una vez, y un rango de tamaños de entre 50 y 200 píxeles para poder detectar monedas de distintos tipos.

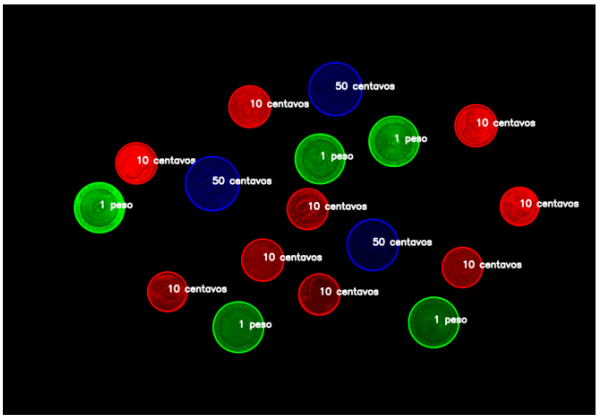
Círculos Detectados

Realizamos entonces la clasificación de las monedas detectadas en categorías en función de sus radios. Determinamos un rango de 120-145 píxeles para las monedas de 10 centavos, 145-170 píxeles para las monedas de 1 peso y 170-500 píxeles para las monedas de 50 centavos.
Para cada tipo de moneda utilizamos un color distinto, y mostramos la detección de monedas de distintos tipos junto a su valor.

Clasificación



Conteo



Utilizamos estos valores obtenidos para realizar el conteo automático de la suma total de dinero contenida en las monedas, y mostramos el resultado desglosado por consola:

```
================================================
RESULTADOS DE MONEDAS
================================================
10c : 9 moneda(s) | $  0.90
1p  : 5 moneda(s) | $  5.00
50c : 3 moneda(s) | $  1.50
------------------------------------------------
TOTAL: $7.40
================================================
```

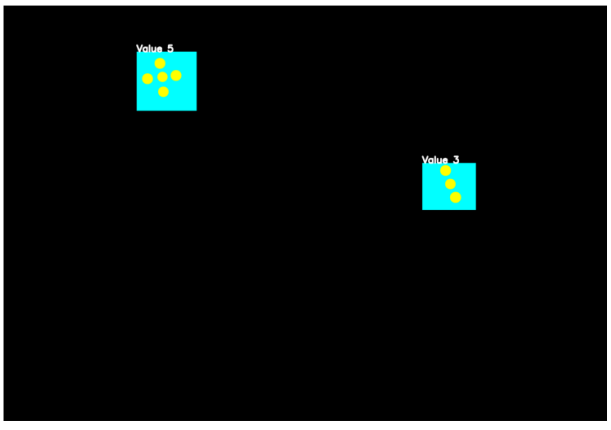
Continuamos con la detección de dados. Para ello, tomamos la máscara de clausura y “eliminamos” las monedas de la misma, dibujando círculos negros por encima de ellas. De esta forma obtuvimos una máscara que sólo contuviera los dados. La dilatamos de forma iterativa, 10 veces, para obtener dados más sólidos.

Utilizando la máscara obtenida en el paso anterior, aplicamos componentes conectadas para filtrar por área, determinando el umbral 10000, detectando de esta forma únicamente dados completos, y descartando el ruido. Para verificar que el proceso hubiera sido exitoso, generamos un recuadro alrededor de cada label detectado que tuviera un área mayor a 10000 y confirmamos que efectivamente así fue.

Dados Detectados

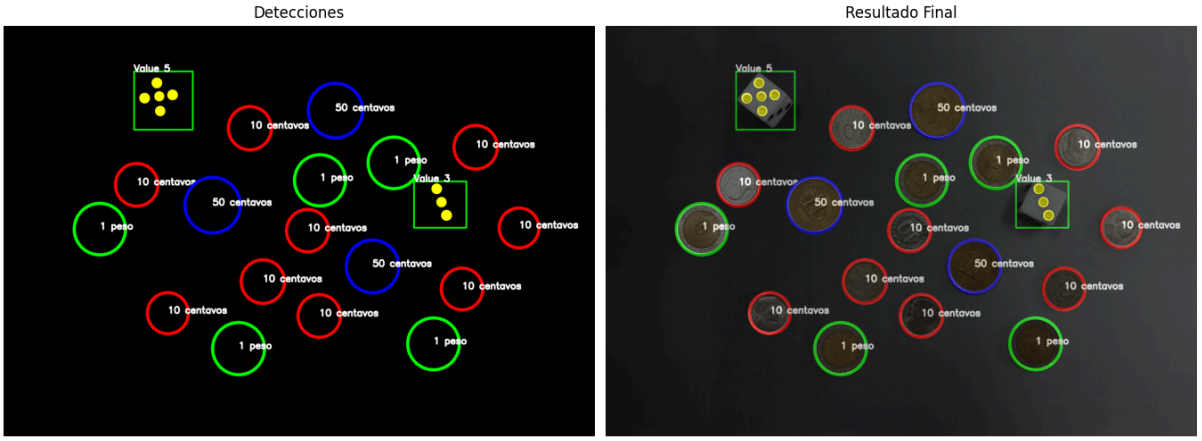
Finalmente, restaba detectar la cantidad de puntos presente en cada dado, y así poder contar automáticamente el valor de los mismos. Analizamos cada dado por separado, aplicando Hough nuevamente, pero esta vez buscando círculos más pequeños, de entre 30-40 píxeles, que es el tamaño aproximado de los puntos del dado. A su vez, también seleccionamos una distancia mínima menor (de 12 píxeles), debido a que estos puntos se encuentran mucho más cerca entre sí que las monedas. Para cada dado, contamos la cantidad de círculos (puntos) detectados y le asignamos dicho valor al dado.

Dados con Valores




```
==================================================
RESULTADOS DE DADOS
==================================================
Dado 1: Valor = 5
Dado 2: Valor = 3
==================================================
```

Visualización final de las monedas y dados detectados, con sus respectivos valores:



Conclusión:

Concluimos que la detección de bordes seguida de la aplicación de Hough y un filtro en función del tamaño es un método eficaz para determinar los valores de distintos tipos de monedas y poder cuantificar el valor total de las mismas en una fotografía. Un procedimiento similar, ajustando los parámetros del procesamiento, puede utilizarse para detectar los valores de la cara superior de dados y cuantificar su valor.
La principal dificultad con la que tuvimos que trabajar fue la selección de los parámetros a utilizar, como por ejemplo el tamaño del elemento estructural utilizado durante las operaciones morfológicas, y los distintos parámetros de entrada tanto del algoritmo de Canny como para la transformada de Hough.

Problema 2 - Detección de patentes

Introducción:

El objetivo de esta segunda parte del trabajo práctico es procesar doce imágenes que representan la vista anterior o posterior de diversos vehículos, detectando automáticamente la placa patente, y segmentándola. Posteriormente se busca implementar un algoritmo que segmente los caracteres de la placa patente determinada con anterioridad.

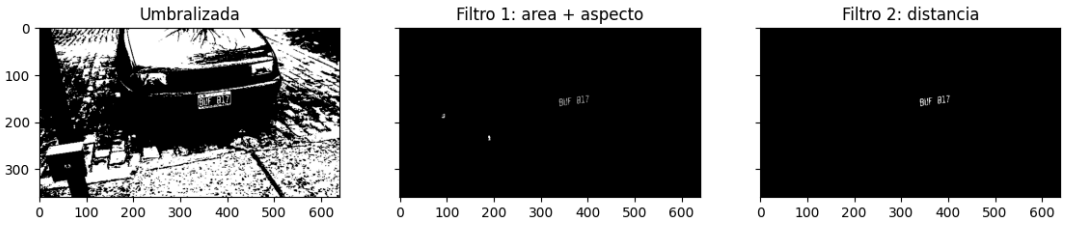
Desarrollo:

Para la detección de patentes, se utilizaron parámetros específicos para cada fotografía, de modo de poder detectar tanto las patentes como las letras de forma específica. Se generaron 12 configuraciones teniendo en cuenta las distintas condiciones de cada imagen: iluminación, ángulo de la foto, color de fondo y contraste entre patente y fondo.
Se determinaron los parámetros de: umbral de binarización, área en píxeles de los caracteres, relación de aspecto entre alto y ancho y la separación máxima entre letras, personalizados para cada imagen.
Un proceso similar se utilizó para la detección de la forma completa de la patente,

aumentando el tamaño del área y modificando el radio ancho/alto, dado que la patente completa tiene un perfil más horizontal que la letra.

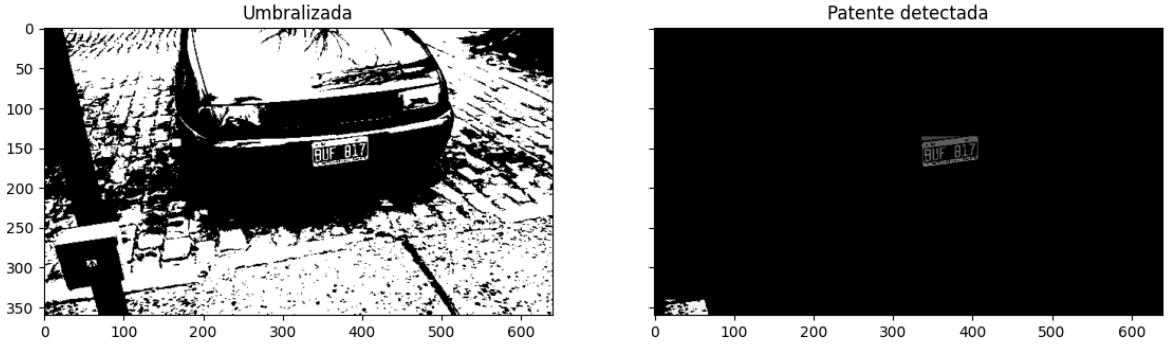
Lo primero que hicimos fue cargar la imagen en escala de grises, junto con los parámetros correspondientes a la misma. La binarizamos en función del umbral predefinido, obtuvimos las componentes conectadas, y procesamos las labels obtenidas, descartando la correspondiente al fondo.
El filtro 1 se relaciona con las componentes detectadas en función de los parámetros elegidos: relación alto/ancho y área. Se eliminaron las componentes muy grandes, dado que las mismas no correspondían a letras, las muy pequeñas, correspondientes al ruido, y las que fueran muy anchas o muy delgadas. Las que pasaron los filtros fueron consideradas letras.
El filtro 2 se relaciona con la distancia entre componentes: si el componente se encontraba aislado, no se consideraba como letra, si su distancia era menos a la distancia máxima determinada, es decir, si se encontraba cerca de al menos un componente, se incluía este componente como letra. De esta forma, se mantuvieron sólo aquellas componentes que formaran grupos.
A continuación se encuentra el resultado para una de las imágenes del ejemplo. Como se puede observar, con la detección a partir de área y aspecto se obtiene un resultado bastante sólido, muy similar al obtenido al aplicar adicionalmente el filtro de distancia.

Imagen 1 - Caracteres



Para la segmentación de formas completas (patente entera). En este caso, el algoritmo tiene un procesamiento muy similar, se umbraliza la imagen, se aplica componentes conectados y se filtra. Se utiliza sólo el filtro de área y aspecto, no se incorpora el filtro de distancia.

Imagen 1 - Forma completa



Conclusión:

Como conclusión determinamos que el algoritmo desarrollado tiene limitaciones para detectar las patentes de las imágenes 3 y 12. Dedujimos que esto se debía al hecho de que el color de la patente es muy similar al del auto, lo que no permite umbralizar correctamente la patente con respecto al fondo.