Facultad de
Ciencias Exactas,
Ingeniería y Agrimensura

FCEIA

TRABAJO PRÁCTICO N° 2

PROCESAMIENTO DE IMÁGENES

INTEGRANTES:

- ALMADA, Manuela - A-4692/6
- CHIAPPE, Maximiliano - C-7542/6
- LENARDUZZI, Juan Sebastián - L-3398/7

DOCENTES:

- ALLIONE, Joaquín
- CALLE, Juan Manuel
- SAD, Gonzalo

Pasos Previos:

Para comenzar el trabajo práctico, descargamos todo el material proporcionado por la cátedra, disponible en el repositorio de Git indicado.

Todos los elementos fueron almacenados dentro de una misma carpeta, la cual incluye el script **“Tp2_PDI”**, encargado de resolver los problemas planteados.

Script

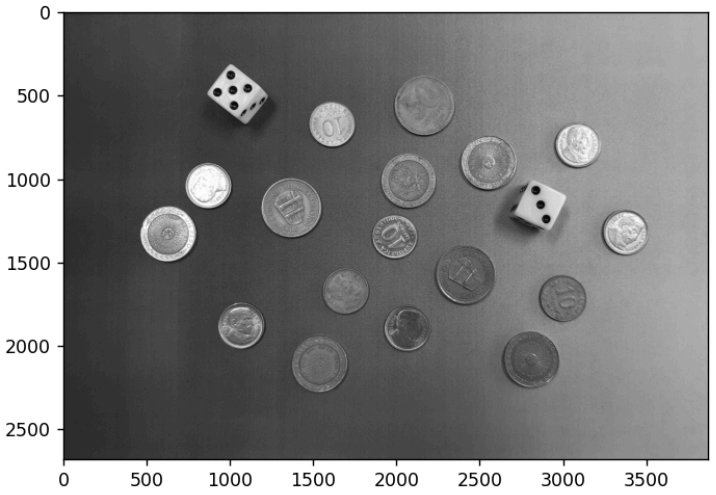
1. **Importación de librerías:**
   Se importaron las funciones necesarias para el procesamiento de imágenes y la visualización de resultados.

2. **Carga de las imágenes:**
   Se descargaron las distintas imágenes que contienen las patentes y la que posee las monedas y dados.

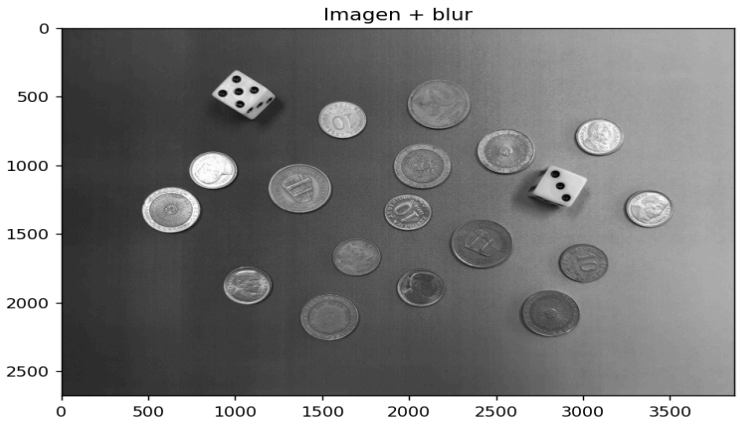
3. **Desarrollo de los problemas**:

Problema 1: Detección de monedas y dados

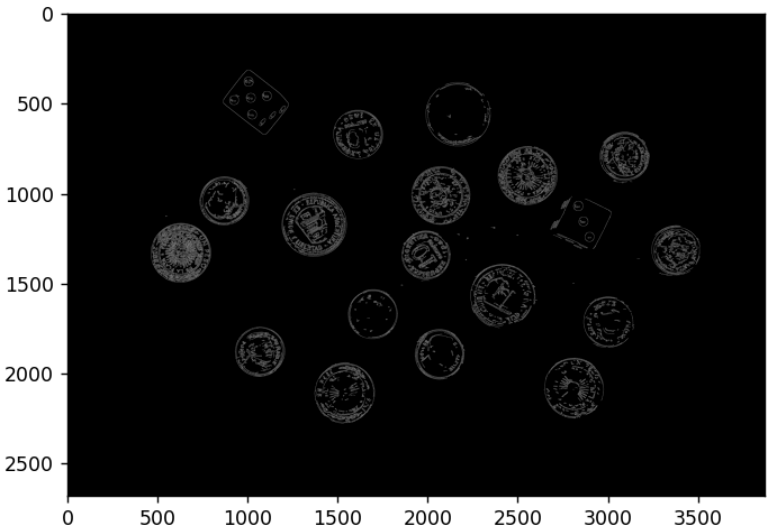
Carga y Visualización: Se cargó la imagen monedas.jpg en escala de grises. Se verificaron sus propiedades básicas (img.shape, img.min(), img.max()).



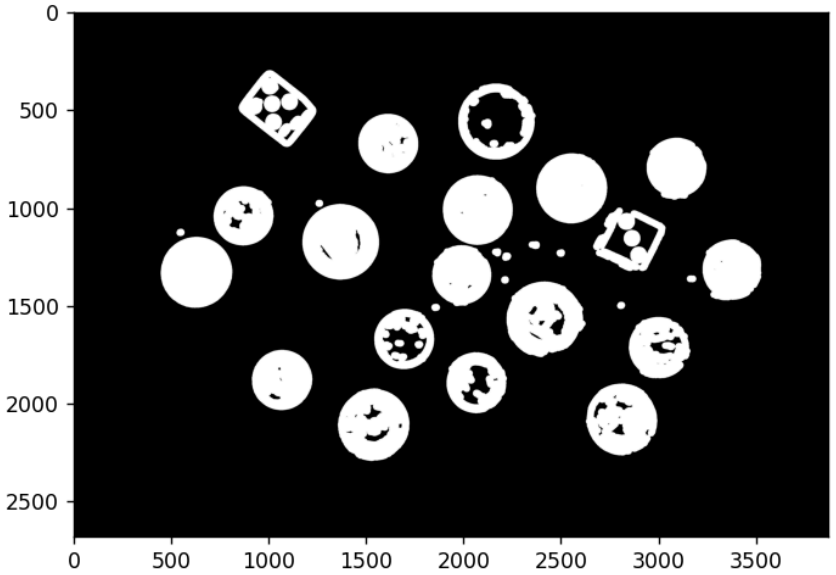
Suavizado Gaussiano: Se aplicó un filtro **Gaussiano** (cv2.GaussianBlur con kernel de 3,3) para reducir el ruido fino y facilitar la detección de bordes en la siguiente etapa, mejorando la calidad de la segmentación.



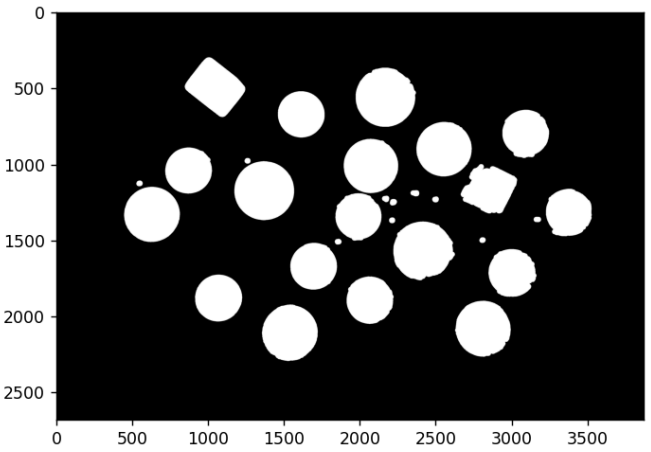
Detección de bordes:
Se aplicó el detector de bordes Canny para localizar los límites de los objetos



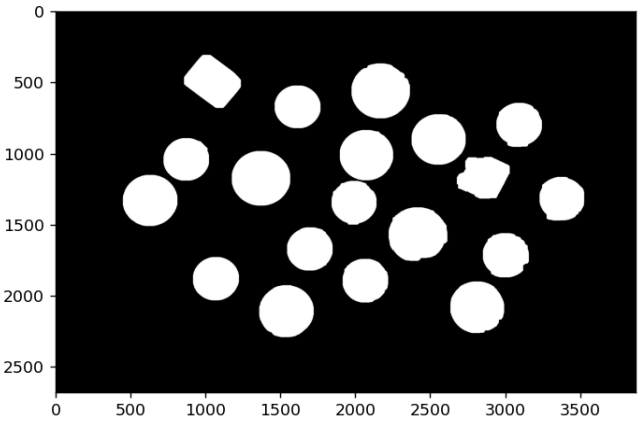
Luego, se utilizó la operación de dilatación con un kernel de (35,35). El propósito de esto es conectar pequeños huecos en los contornos y engrosar los bordes para facilitar el relleno posterior.



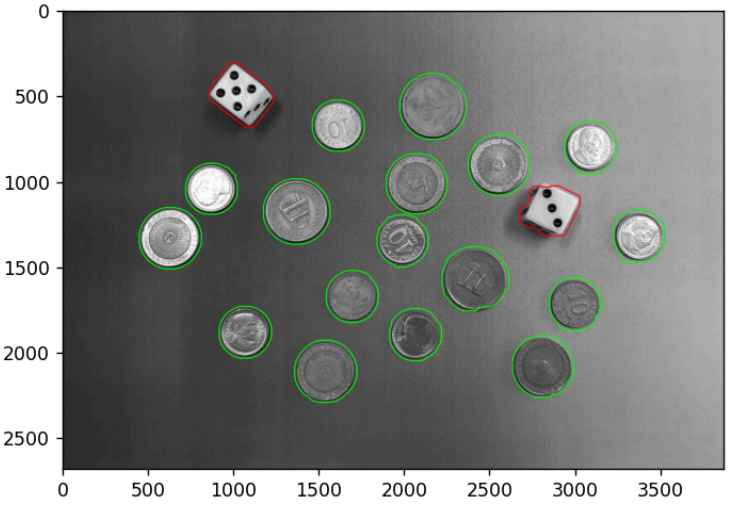
Para el mismo, se utilizó la función ‘cv2.floodFill’, creando los blobs para la segmentación.



Por último, se realizó una apertura morfológica con un kernel (50,50). Esto suaviza los contornos y rompe puentes finos entre objetos.



Conteo y clasificación inicial (Circularidad):
Una vez obtenida la máscara binaria (Aop), se usó el análisis de contornos ‘cv2.findContours’ para aislar el objeto. Se calculó la circularidad para cada contorno detectado y luego, aquellos con una circularidad mayor a 0.85 se clasificaron como Monedas (verde) y el resto como Dados (rojo)



Clasificación de Monedas y Conteo:
Se clasificaron las monedas en grupos (chica, mediana, grande) usando el área como descriptor.
Se definieron umbrales basados en los valores de área observados en el conjunto de datos:

- Chica: Área < 80.000
- Mediana: 100.000 < Área < 110.000
- Grande: Área > 110.000

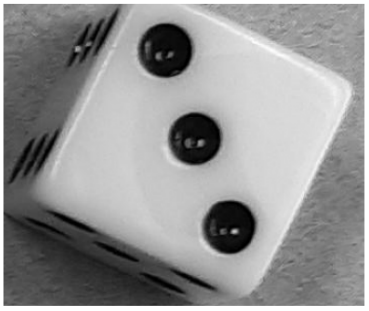
```
Moneda 0 | Área=105086 | Tipo=mediana
Moneda 1 | Área=103358 | Tipo=mediana
Moneda 2 | Área=74709 | Tipo=chica
Moneda 3 | Área=74517 | Tipo=chica
Moneda 4 | Área=76192 | Tipo=chica
Moneda 5 | Área=74083 | Tipo=chica
Moneda 6 | Área=116615 | Tipo=grande
Moneda 7 | Área=72240 | Tipo=chica
Moneda 8 | Área=74188 | Tipo=chica
Moneda 9 | Área=103882 | Tipo=mediana
Moneda 11 | Área=119259 | Tipo=grande
Moneda 12 | Área=73206 | Tipo=chica
Moneda 13 | Área=100355 | Tipo=mediana
Moneda 14 | Área=102134 | Tipo=mediana
Moneda 15 | Área=74592 | Tipo=chica
Moneda 16 | Área=73252 | Tipo=chica
Moneda 17 | Área=118776 | Tipo=grande
```

Luego, se realizó un conteo automático, mediante el diccionario ‘conteo’, para registrar el número de monedas encontradas por categoría.

```
Conteo final por tipo:
>>> print(conteo)
{'chica': 9, 'mediana': 5, 'grande': 3}
```

Determinación del valor del dado y conteo:
Se aisló la ROI de cada dado detectado, creando una sub-imagen para un procesamiento localizado.

Dado 10 - ROI original



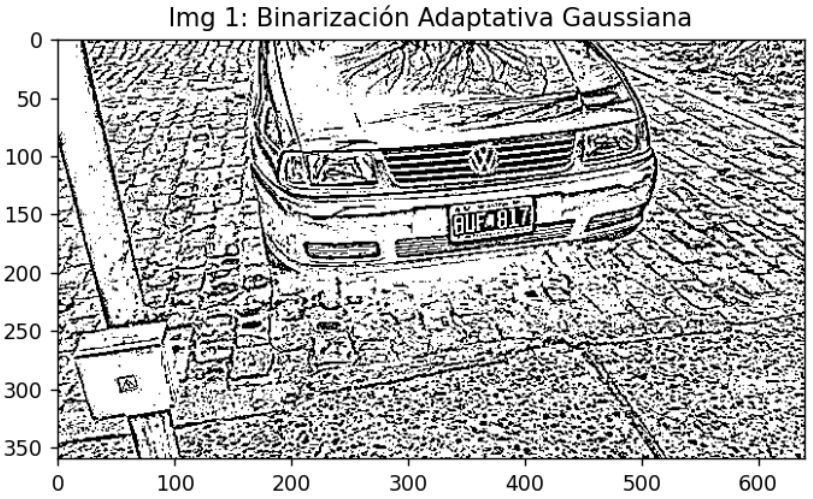
Dado 18 - ROI original

Se aplicó un suavizado extra (7,7) solo en el ROI para eliminar el ruido interno del dado, sin afectar al resto de la imagen.
Se utilizó el umbral adaptativo para segmentar los pips oscuros.
Luego, se aplicó una apertura con kernel (5,5) para eliminar cualquier ruido residual que no sean los puntos reales del dado.
Para el conteo final, se utilizó ‘cv2.findContours’ para detectar cada punto individual.
Luego, se aplicó un filtro estricto por área para garantizar que solo los puntos reales sean contados. El valor final del dado es la cantidad de contornos que superaron este filtro.

```
Dado 10: 3 puntos
Dado 18: 5 puntos
```

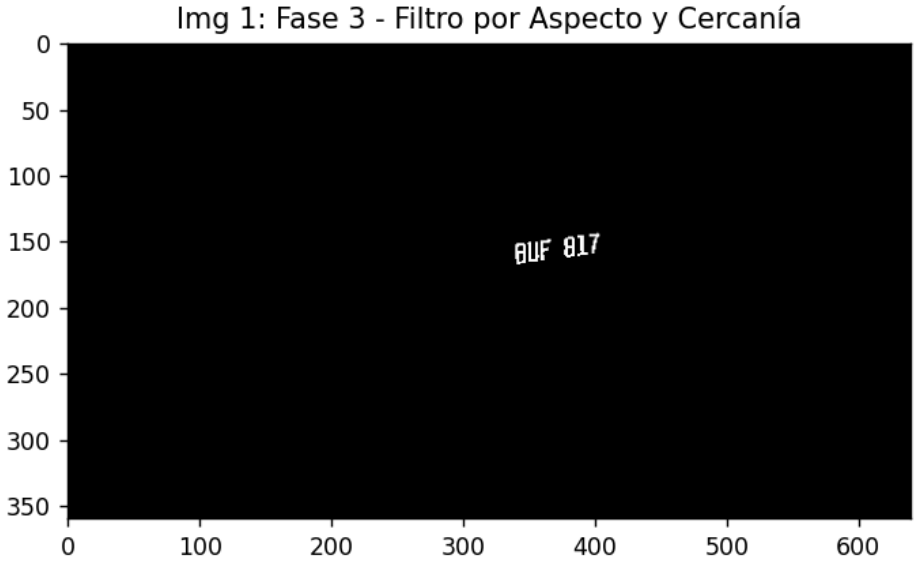
Problema 2: Detección de patentes

Preprocesamiento: Carga y Binarización Robusta:
La primera etapa consistió en estandarizar las doce imágenes (img01.png a img12.png) y prepararlas para el análisis de componentes.
Esta preparación consistió en la conversión de las imágenes a escala de grises.
Luego, dada a la alta variabilidad en la iluminación, sombras y reflejos presentes en el conjunto de datos, se descartó el umbral manual en favor de la Binarización adaptativa Gaussiana (‘cv2.adaptiveThreshold). Esta técnica calcula un umbral localmente (en un vecindario de 9x9 pixeles), permitiendo que el algoritmo aísle los caracteres claros de la patente incluso en áreas con iluminación desigual.
Se observó que las imágenes 6 y 10 presentaban ruido o desconexiones que no se resolvían con la binarización estándar, por lo que se aplicaron correcciones puntuales para estas imágenes.



Segmentación de caracteres mediante filtros heurísticos:
La segmentación del problema se basó en el análisis de componentes conectados, asumiendo que un carácter de patente válido debe cumplir con criterios geométricos y estar agrupado espacialmente con otros.

- *Filtro geométrico*: Se identificaron los blobs iniciales y se filtraron por dos características:
    - Relación de aspecto:Los caracteres de patente son consistentemente más altos que anchos. Se conservaron los componentes cuya relación **Alto/Ancho** se encontraba en el rango **[1.5, 3.0]**
    - **Área:** Para eliminar ruido muy fino o componentes que son demasiado grandes (como sombras o reflejos), se definió un rango de área de **[20, 110] píxeles**. Solo los objetos que satisfacen *ambos* criterios (área y aspecto) pasaron a la siguiente fase
- *Filtro de proximidad (Vecindad de centroides)*:
    - Se utiliza el hecho de que todos los caracteres de una patente están muy juntos. Solo se conservó un componente si su centroide se encontraba a una distancia máxima de 80.0 pixeles de otro componente que también haya superado el filtro geométrico.
    - Este filtro, actúa como agrupador, eliminando el ruido disperso
    - La imágen resultante (imagen_filtrada_cercania) contiene exclusivamente los pixeles de los caracteres de la patente, completamente aislados de la escena.



Detección de la patente y visualización:
Una vez que se tienen los píxeles de los caracteres aislados, se realizó la detección de la placa completa.

- Bounding Box de la placa completa: se calcula el Bounding Box mínimo (cv2.boundingRect) que engloba todos los píxeles restantes en la imagen filtrada. Este recuadro delimita la ubicación de la patente en la imagen original.
- *Segmentación de la placa*: Se utiliza el Bounding box calculado (con un padding de 5 píxeles) para realizar el recorte de la imagen original a color, obteniendo la patente completamente segmentada.
- **Visualización**: Por último, se visualizan los resultados.
    - El bounding box azul, indica la detección de la patente completa
    - El bounding box verde, se dibuja alrededor de cada componente individual que superó el filtro de cercanía, logrando la segmentación individual de los caracteres.

Img 1: Fase 4 - BBox Patente Completa

Img 1: Fase 6 - BBox de Caracteres Válidos