

## **LAB 1**

LAB 1: <https://colab.research.google.com/drive/1YsN2t4wLdM8y0Vc0tPzaClcdB-KDAwDZ#scrollTo=OpWvyuUiPf9Z>

**mini laboratorio de visión por computadora** para que la compu aprenda a **detectar frutillas en una foto** usando colores.

👉 Lo que pasa en cada paso:

### **1. Cargar la imagen**

→ Ejemplo: le damos una foto con bananas, uvas y frutillas.

### **2. Entender cómo la compu ve los colores**

→ Cada píxel es solo tres números: (Rojo, Verde, Azul).

- Una frutilla típica sería algo como **(200, 30, 40)** → mucho rojo, poco verde/azul.
- Una hoja verde sería algo como **(50, 180, 60)** → mucho verde.
- Un plátano amarillo sería **(220, 220, 30)** → rojo y verde altos, azul bajo.

### **3. Definir una regla**

→ Si el píxel tiene **Rojo alto** y **Verde + Azul bajos**, lo marcamos como frutilla.

Ejemplo:

- (200, 30, 40) ☒ frutilla
- (50, 180, 60) ☐ hoja verde

Píxeles Examinados - ¡Cada círculo es un punto específico!



#### 4. Aplicar la regla a toda la imagen

→ Se genera una “máscara”:

- a. Blanco = frutilla
- b. Negro = no frutilla

#### 5. Ver el resultado

→ Se muestra una imagen con solo las frutillas destacadas, todo lo demás en negro.

#### 6. Experimentar con los valores

→ Si bajas el umbral del rojo, detecta más cosas (pero puede confundir tomates o sombras).

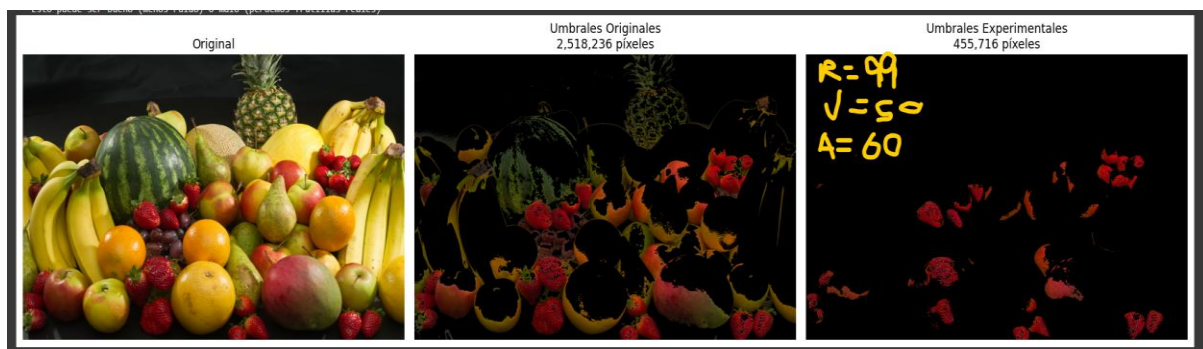
→ Si lo subes, detecta menos (quizás pierde frutillas más oscuras).

En pocas palabras:

La compu no sabe qué es una frutilla, solo mira **números de colores**.

Nosotros le damos una regla “si es bien rojo y poco verde/azul = frutilla”, y así puede detectarlas en la foto

### Experimento



### **LAB 2**

[https://colab.research.google.com/drive/1Kcr9mDfYgoFsA378CI9sFPDOt\\_jGznUk#scrollTo=cell-connections](https://colab.research.google.com/drive/1Kcr9mDfYgoFsA378CI9sFPDOt_jGznUk#scrollTo=cell-connections)

La idea es agarrar una imagen, analizarla y extraer solo los objetos que nos interesan (en este caso, flores rojas) usando criterios de color. Te lo desgloso:

## 1. Preparación

- Se cargan librerías para manejar imágenes (OpenCV, numpy, matplotlib) y para hacer experimentos interactivos (ipywidgets).
- Se define funciones auxiliares para ver info de la imagen, histogramas de color y mostrar comparaciones de imágenes.

## 2. Carga y análisis inicial

- Se descarga y carga una imagen de flores.
- Se convierte de **BGR (OpenCV)** a **RGB (matplotlib)** para que los colores se vean correctos.
- Se muestran los canales rojo, verde y azul por separado, y se analizan histogramas para entender qué colores predominan.

## 3. Segmentación por color

- **Se definen umbrales:** por ejemplo, “rojo > 150, verde < 50, azul < 50” para identificar flores rojas.
- Se crean **máscaras binarias:** píxel blanco = cumple condición, negro = no cumple.
- Se combinan las máscaras de los tres canales usando **AND lógico** para extraer solo los píxeles que realmente son flores rojas.

## 4. Post-procesamiento

- Se limpia la máscara eliminando ruido (píxeles aislados que no corresponden a la flor).
- Se puede calcular un **rectángulo delimitador** o usar **contornos** para ver la forma exacta de las flores.

## 5. Experimentos interactivos

- Se puede jugar con los umbrales para ver cómo cambia la segmentación.
- Permite probar qué pasa si somos más estrictos o relajados con los colores.

## 6. Extensiones y aplicaciones

- Analizar la composición de colores en cualquier imagen.
- Aplicable a agricultura, medicina, control de calidad industrial, visión por computadora.
- Se muestra cómo la **cuantización** (reducir la precisión de los colores) afecta la segmentación.

## 7. Conceptos clave que toca

- Segmentación, máscara binaria, umbral, canal de color, histograma, operaciones lógicas, contornos, rectángulo delimitador, post-procesamiento, ruido, cuantización.

💡 **En resumen:** el Colab te enseña cómo **aislar objetos de interés en una imagen según su color**, limpiar resultados y analizar su forma y tamaño. Es un laboratorio completo desde la teoría hasta experimentos prácticos y visuales.

### **LOCALIZAR PÍXELES**

<https://colab.research.google.com/drive/1aadRhUk2KyXS4vC7zopauUnzXO7VuAFs#scrollTo=a1TG8uLVVO6G>

### **Preparación**

1. Se importan librerías clave: `numpy`, `matplotlib`, `cv2` y `ipywidgets`.
2. Se descarga una imagen de ejemplo de frutas y se convierte de BGR a RGB (porque OpenCV usa BGR por defecto).
3. Se imprime la forma de la imagen (`alto x ancho x canales`).

## Método 1: Imagen con ejes y grilla

- Muestra la imagen con ejes visibles y grilla para leer coordenadas.
- Útil para identificar visualmente píxeles de interés.
- **Flujo:**
  - Ves el objeto en la imagen.
  - Lees coordenadas aproximadas en los ejes.
  - Usas esas coordenadas como `imagen[Y, X]`.
- ☒ Simple, confiable y siempre funciona en Colab.

## Método 2: Explorador interactivo con widgets


- Crea sliders para moverte por la imagen (X, Y) y ajustar el zoom (Radio).
- Muestra:
  - Imagen completa con el píxel seleccionado marcado.
  - Zoom de la región alrededor del píxel.
  - Valores RGB del píxel seleccionado.
- ☒ Interactivo y práctico sin depender de eventos de mouse que Colab bloquea.

## Método 3: Análisis de regiones predefinidas


- Define zonas típicas donde suelen estar ciertos objetos (ej.: frutillas abajo-centro, manzanas en el centro).
- Muestra:
  - Imagen con puntos marcados y etiquetados.
  - Valores RGB y código para acceder a cada píxel.
- ☒ Rápido y muy útil cuando sabes de antemano las áreas de interés.

## Método 4: Búsqueda manual sistemática

- Prueba una lista de coordenadas específicas.
- Muestra:
  - Imagen con los puntos marcados.

- Valores RGB.
- Gráfico de barras con valores RGB de cada píxel.
-  Infalible y da control total sobre qué coordenadas analizar.

## Método 5: Generador de código automático

- Una vez que identificaste coordenadas de interés, genera automáticamente **código para segmentar esos objetos por color**.
- Calcula:
  - Valores mínimos, máximos y promedio RGB de los píxeles seleccionados.
  - Umbrales de color con margen de seguridad.
- Genera un snippet que:
  - Extrae los canales RGB.
  - Aplica condiciones sobre los umbrales.
  - Crea máscara binaria y muestra la región detectada.
-  Automatiza segmentación y ahorra tiempo de programación.

## Resumen final

- **Siempre funcionan en Colab:** Método 1 (visual), 2 (widgets), 3 (regiones predefinidas), 4 (manual), 5 (generador de código).
- **No funciona en Colab:** eventos de mouse, callbacks de matplotlib, interactividad avanzada.

## Recomendación corporativa:

- Explora coordenadas visualmente con Método 1.
- Usa Método 5 para automatizar la segmentación y análisis de píxeles.