Image Segmentationn

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Load the image
image = cv2.imread('/content/mahadev.jpg')
if image is None:
    raise ValueError("Could not load the image. Please check the file
path.")

# Convert the image to grayscale
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Apply Gaussian blur to reduce noise and improve segmentation
blurred_image = cv2.GaussianBlur(gray_image, (5, 5), 0)

# Apply Otsu's thresholding method for segmentation
_, threshold_image = cv2.threshold(blurred_image, 0, 255,
cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# Find contours
contours, hierarchy = cv2.findContours(threshold_image,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Draw the contours on the original image
contour_image = image.copy()
cv2.drawContours(contour_image, contours, -1, (0, 255, 0), 2)

# Function to resize images to a consistent size
def resize_image(img, width=300):
    aspect_ratio = img.shape[1] / img.shape[0]
    height = int(width / aspect_ratio)
    return cv2.resize(img, (width, height))

# Function to add labels to images
def add_label(image, label):
    # Create a white bar at the top of the image for the label
    h, w = image.shape[:2]
    label_bar = np.ones((30, w, 3), dtype=np.uint8) * 255

    # Add text to the label bar
    font = cv2.FONT_HERSHEY_SIMPLEX
    font_scale = 0.6
    thickness = 1
    text_size = cv2.getTextSize(label, font, font_scale, thickness)[0]
    text_x = (w - text_size[0]) // 2
```

```
    text_y = 20
    cv2.putText(label_bar, label, (text_x, text_y), font, font_scale,
(0, 0, 0), thickness)

    # Combine label bar and image
    return np.vstack((label_bar, image))

# Resize all images
images = [
    resize_image(image),
    resize_image(cv2.cvtColor(gray_image, cv2.COLOR_GRAY2BGR)),
    resize_image(cv2.cvtColor(threshold_image, cv2.COLOR_GRAY2BGR)),
    resize_image(contour_image)
]

# Add labels to images
labels = ['Original', 'Grayscale', 'Threshold', 'Contours']
labeled_images = [add_label(img, label) for img, label in zip(images,
labels)]

# Create horizontal concatenation of images
result = np.hstack(labeled_images)

# Display the combined result
cv2_imshow(result)
```


```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Load the image
```

```
image = cv2.imread('/content/mahadev.jpg')
if image is None:
    raise ValueError("Could not load the image. Please check the file
path.")

# Convert to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Apply threshold to convert the image to binary
_, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)

# Remove noise using morphological transformations (opening)
kernel = np.ones((3, 3), np.uint8)
opening = cv2.morphologyEx(binary, cv2.MORPH_OPEN, kernel,
iterations=2)

# Get sure background by dilating the result
sure_bg = cv2.dilate(opening, kernel, iterations=3)

# Distance transform to get sure foreground
dist_transform = cv2.distanceTransform(opening, cv2.DIST_L2, 5)
_, sure_fg = cv2.threshold(dist_transform, 0.7 * dist_transform.max(),
255, 0)

# Subtract sure foreground from sure background to get unknown region
sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg, sure_fg)

# Markers for watershed: label sure_fg as 1, unknown region as 0
_, markers = cv2.connectedComponents(sure_fg)

# Add one to all labels to ensure background is not labeled as 0
markers = markers + 1

# Mark the unknown region as 0
markers[unknown == 255] = 0

# Apply watershed
markers = cv2.watershed(image, markers)

# Create a copy of the original image for visualization
result_image = image.copy()

# Mark the boundaries in red on the result image
result_image[markers == -1] = [0, 0, 255]

# Prepare markers visualization
markers_viz = cv2.normalize(markers.astype(np.float32), None, 0, 255,
cv2.NORM_MINMAX)
```

```
markers_viz = markers_viz.astype(np.uint8)
markers_colored = cv2.applyColorMap(markers_viz, cv2.COLORMAP_JET)

# Function to resize images to a consistent size
def resize_image(img, width=300):
    aspect_ratio = img.shape[1] / img.shape[0]
    height = int(width / aspect_ratio)
    return cv2.resize(img, (width, height))

# Function to add labels to images
def add_label(image, label):
    # Create a white bar at the top of the image for the label
    h, w = image.shape[:2]
    label_bar = np.ones((30, w, 3), dtype=np.uint8) * 255

    # Add text to the label bar
    font = cv2.FONT_HERSHEY_SIMPLEX
    font_scale = 0.6
    thickness = 1
    text_size = cv2.getTextSize(label, font, font_scale, thickness)[0]
    text_x = (w - text_size[0]) // 2
    text_y = 20
    cv2.putText(label_bar, label, (text_x, text_y), font, font_scale,
(0, 0, 0), thickness)

    # Combine label bar and image
    return np.vstack((label_bar, image))

# Resize all images
images = [
    resize_image(image),
    resize_image(cv2.cvtColor(binary, cv2.COLOR_GRAY2BGR)),
    resize_image(cv2.cvtColor(opening, cv2.COLOR_GRAY2BGR)),
    resize_image(cv2.cvtColor(sure_bg, cv2.COLOR_GRAY2BGR)),
    resize_image(cv2.cvtColor(sure_fg, cv2.COLOR_GRAY2BGR)),
    resize_image(cv2.cvtColor(unknown, cv2.COLOR_GRAY2BGR)),
    resize_image(markers_colored),
    resize_image(result_image)
]

# Add labels to images
labels = ['Original', 'Binary', 'Opening', 'Sure BG', 'Sure FG',
'Unknown', 'Markers', 'Final']
labeled_images = [add_label(img, label) for img, label in zip(images,
labels)]

# Create horizontal concatenation of images
result = np.hstack(labeled_images)
```

```
# Display the combined result
cv2_imshow(result)
```