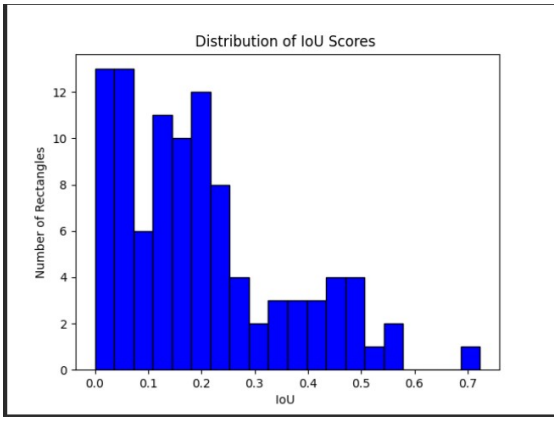
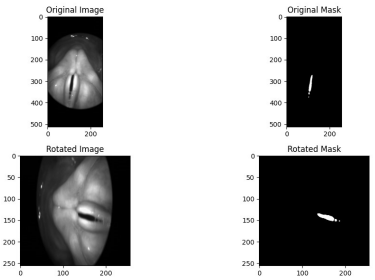
Name: Rutvi shah
Matriculation:23159043
Idm :to20raje




```python
import numpy as np
import matplotlib.pyplot as plt
import flammkuchen as fl

1 usage
def calculate_iou(rect1, rect2):
    def to_numeric(rect):
        try:
            return tuple(map(int, rect))
        except ValueError:
            # Handle non-numeric values, for example, by skipping the rectangle
            return None

    rect1_numeric = to_numeric(rect1)
    rect2_numeric = to_numeric(rect2)

    if rect1_numeric is None or rect2_numeric is None:
        return 0  # Return 0 for non-numeric rectangles

    x1, y1, w1, h1 = rect1_numeric
    x2, y2, w2, h2 = rect2_numeric

    intersection_x = max(0, min(x1 + w1, x2 + w2) - max(x1, x2))
    intersection_y = max(0, min(y1 + h1, y2 + h2) - max(y1, y2))

    intersection_area = intersection_x * intersection_y
    union_area = w1 * h1 + w2 * h2 - intersection_area
```

```python
import numpy as np
import matplotlib.pyplot as plt
import flammkuchen as fl

1 usage
def calculate_iou(rect1, rect2):
    def to_numeric(rect):
        try:
            return tuple(map(int, rect))
        except ValueError:
            # Handle non-numeric values, for example, by skipping the rectangle
            return None

    rect1_numeric = to_numeric(rect1)
    rect2_numeric = to_numeric(rect2)

    if rect1_numeric is None or rect2_numeric is None:
        return 0  # Return 0 for non-numeric rectangles

    x1, y1, w1, h1 = rect1_numeric
    x2, y2, w2, h2 = rect2_numeric

    intersection_x = max(0, min(x1 + w1, x2 + w2) - max(x1, x2))
    intersection_y = max(0, min(y1 + h1, y2 + h2) - max(y1, y2))

    intersection_area = intersection_x * intersection_y
    union_area = w1 * h1 + w2 * h2 - intersection_area
```




```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
import albumentations as A
from albumentations.pytorch import ToTensorV2

# Set random seed
np.random.seed(23159043)  # Use your matriculation number as the random seed

# List of paths to images and masks
image_paths = [
    "C:/Users/rutvi/OneDrive/Desktop/Masters/Semester 3/Data science survival skills/Exercise/Assignment 6/Mini_BAGLS_dataset
    "C:/Users/rutvi/OneDrive/Desktop/Masters/Semester 3/Data science survival skills/Exercise/Assignment 6/Mini_BAGLS_dataset
    # Add more paths as needed
]

mask_paths = [
    "C:/Users/rutvi/OneDrive/Desktop/Masters/Semester 3/Data science survival skills/Exercise/Assignment 6/Mini_BAGLS_dataset
    "C:/Users/rutvi/OneDrive/Desktop/Masters/Semester 3/Data science survival skills/Exercise/Assignment 6/Mini_BAGLS_dataset
    # Add more paths as needed
]

# Loop over each image-mask pair
for image_path, mask_path in zip(image_paths, mask_paths):
    # Load the image and mask
    image = cv2.imread(image_path)
    mask = cv2.imread(mask_path, cv2.IMREAD_GRAYSCALE)

    # Define Albumentations transforms
    transform = A.Compose([
        A.RandomRotate90(),
        A.HorizontalFlip(),
        A.VerticalFlip(),
        A.RandomBrightnessContrast(),
        A.RandomGamma(),
        A.Resize(256, 256),  # Resize the image and mask to a common size
        ToTensorV2(),  # Convert image and mask to PyTorch tensors
    ])
```

```python
# Apply transforms to the image and mask
transformed = transform(image=image, mask=mask)
transformed_image = transformed['image']
transformed_mask = transformed['mask']

# Convert PyTorch tensor to NumPy array for visualization
transformed_mask = transformed_mask.squeeze().cpu().numpy()

# Display the original and augmented images and masks side by side
plt.figure(figsize=(12, 6))

# Original Image and Mask
plt.subplot(2, 2, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title('Original Image')

plt.subplot(2, 2, 2)
plt.imshow(mask, cmap='gray')
plt.title('Original Mask')

# Augmented Image and Mask
plt.subplot(2, 2, 3)
plt.imshow(transformed_image.transpose(0, 1).transpose(1, 2))
plt.title('Rotated Image')

plt.subplot(2, 2, 4)
plt.imshow(transformed_mask, cmap='gray')
plt.title('Rotated Mask')

plt.tight_layout()
plt.show()
```