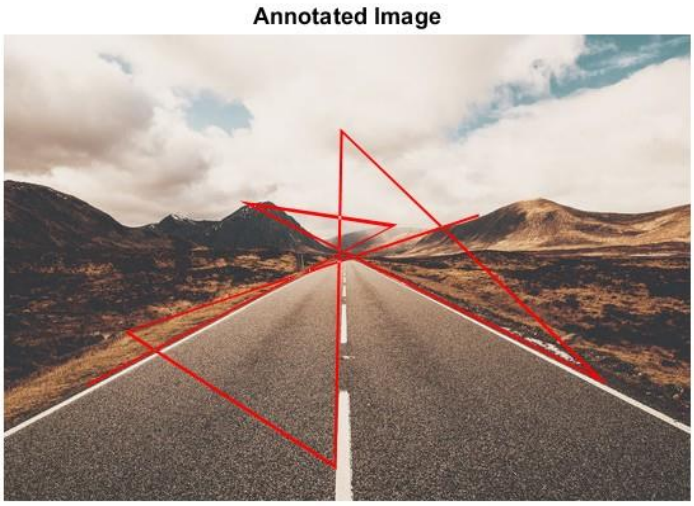
Lane Detection

```
function laneDetection_start
    img = imread('highway.jpg');
    figure; imshow(img); title('Original Image');
    laneMask = createLaneMask(img);
    skeletonizedMask = bwmorph(laneMask, 'thin', Inf);

    [H, theta, rho] = hough(skeletonizedMask);
    P = houghpeaks(H, 5, 'Threshold', 0.3 * max(H(:)), 'NHoodSize', [31 31]);
    lines = houghlines(skeletonizedMask, theta, rho, P, 'FillGap', 80,
'MinLength', 150);

    posArray = getVizPosArray(lines);

    annotatedImg = insertShape(img, 'line', posArray, 'LineWidth', 2, 'Color',
'red');

    figure; imshow(annotatedImg); title('Annotated Image');
end

function laneMask = createLaneMask(img)

    grayImg = rgb2gray(img);
    blurredImg = imgaussfilt(grayImg, 3);
    edgeImg = edge(blurredImg, 'canny');

    [rows, cols, ~] = size(img);
    ROI = [cols/2, 0; cols, rows; 0, rows];

    laneMask = poly2mask(ROI(:,1), ROI(:,2), rows, cols);
    laneMask = laneMask & edgeImg;
end

function posArray = getVizPosArray(lines)
    posArray = zeros(length(lines)*2, 2);

    for k = 1:length(lines)
        xy = [lines(k).point1; lines(k).point2];
        posArray((k-1)*2+1:k*2, :) = xy;
    end
end
```

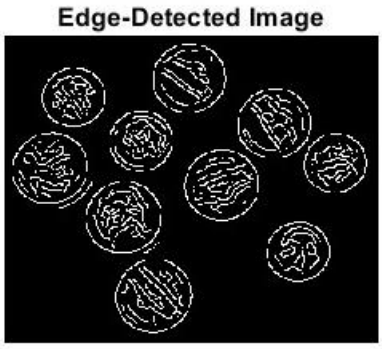
Output:



Edge Detection

```
I = imread('Coins.jpg');
if size(I, 3) == 3
    I = rgb2gray(I);
end
I_smoothed = imgaussfilt(I, 1);

edge_image = edge(I_smoothed, 'Canny');

figure;
subplot(1, 2, 1);
imshow(I);
title('Original Image');

subplot(1, 2, 2);
imshow(edge_image);
title('Edge-Detected Image');
```

Output:



Object Detection

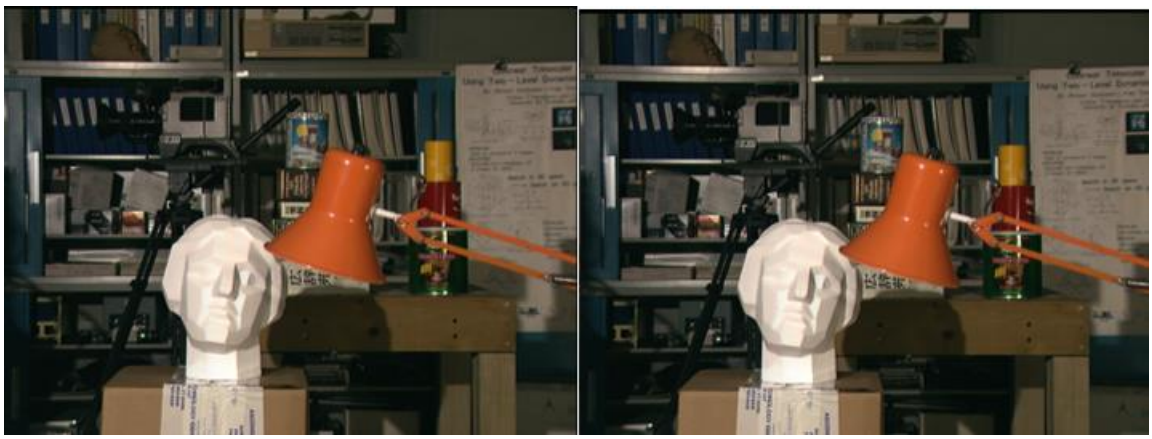
```
import cv2
bg_subtractor = cv2.createBackgroundSubtractorMOG2()
cap = cv2.VideoCapture("C:/Users/ANJALI/Downloads/pexels_videos_2099536
(1080p).mp4")
min_area_threshold = 1000
fps = int(cap.get(cv2.CAP_PROP_FPS))
frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fource= cv2.VideoWriter fource('XVID') out = cv2.VideoWriter('output_video.avi,
fourcc, fps, (frame_width, frame_height))
whilecap.isOpened():
ret, frame cap.read()
if not ret:
break
fg_maskbg_subtractor.apply(frame)
fg_mask = cv2.medianBlur (fg_mask, 5)
contours, _ = cv2.findContours (fg_mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE
for
contour in contours: if cv2.contourArea (contour) >min_area_threshold:
x, y, w, h = cv2.boundingRect(contour)
cv2.rectangle(frame, (x, y), (x+w, y + h), (0, 255, 0), 2)
out.write(frame)
cv2.imshow("Object Detection, frame)
if cv2.waitKey(1) & 0xFF== ord('q'):
break
cap.release()
out.release()
cv2.destroyAllWindows()
```

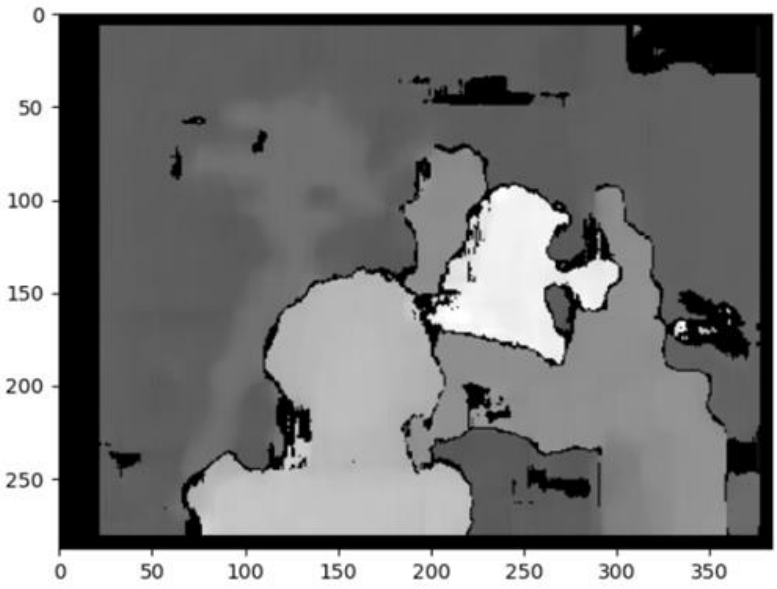
Output:

Depth Estimation

```
% Load left and right stereo images
left_image = imread('sceneLeft.jpg');
right_image = imread('sceneRight.jpg');

left_gray = rgb2gray(left_image);
right_gray = rgb2gray(right_image);

disparity_range = [-16, 16];
disparity_map = disparity(left_gray, right_gray, 'BlockSize', 15,
'DisparityRange', disparity_range);

disparity_map(disparity_map == 0) = NaN;

baseline_distance = 100;
focal_length = 100;
depth_map = (focal_length * baseline_distance) ./ disparity_map;

% Display the depth map
figure;
imshow(depth_map, []);
title('Depth Map');

% Optionally, visualize the disparity map
figure;
imshow(disparity_map, []);
title('Disparity Map');
```

Output:




0
50
100
150
200
250
0
50
100
150
200
250
300
350