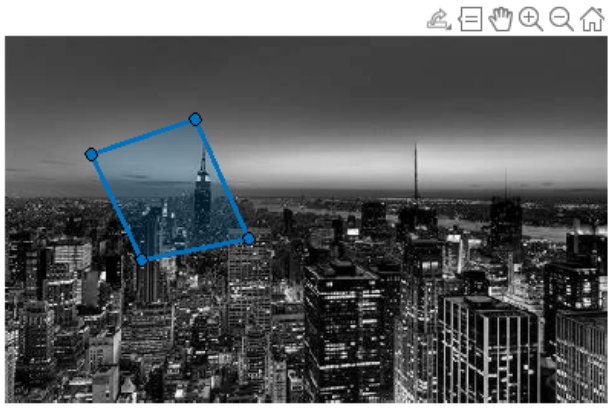
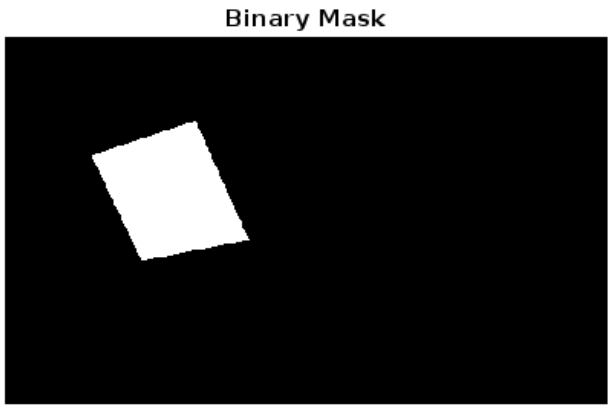
```
% Read the input image
I = imread('City.jpg'); % Replace 'image.jpg' with your image file

% Convert to grayscale if the image is RGB
if size(I, 3) == 3
    I_gray = rgb2gray(I);
else
    I_gray = I;
end

% Create a binary mask manually or automatically
imshow(I_gray);
h = drawpolygon; % Use this function to manually select the ROI
```



```
mask = createMask(h);

% Display the binary mask
imshow(mask);
title('Binary Mask');
```

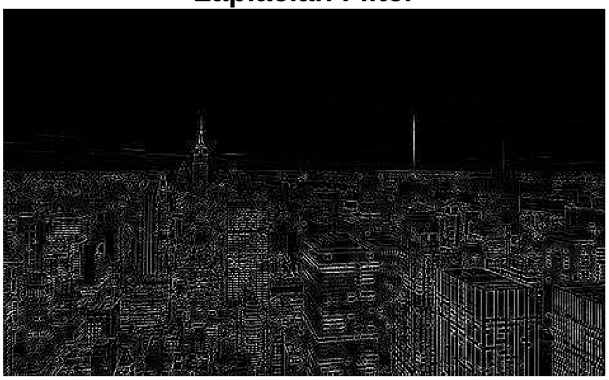



```
%2

% Apply Gaussian filter
sigma = 2; % Standard deviation for Gaussian filter
I_gaussian = imgaussfilt(I_gray, sigma);

% Display the result
figure;
imshow(I_gaussian);
title('Gaussian Filter');
```

```
% Define a 3x3 averaging kernel
h_avg = fspecial('average', [3 3]);

% Apply average filter
I_average = imfilter(I_gray, h_avg);

% Display the result
figure;
imshow(I_average);
title('Average Filter');
```


```
%3

% Apply Laplacian filter
h_laplacian = fspecial('laplacian', 0.2);
I_laplacian = imfilter(I_gray, h_laplacian, 'replicate');

% Display the result
figure;
imshow(I_laplacian, []);
title('Laplacian Filter');
```

Laplacian Filter



```
% prewitt

% Apply Prewitt filter for edge detection
I_prewitt_x = imfilter(I_gray, fspecial('prewitt'), 'replicate');  %
Horizontal edges
I_prewitt_y = imfilter(I_gray, fspecial('prewitt')', 'replicate'); %
Vertical edges

% Combine horizontal and vertical edges by adding them together
I_prewitt = I_prewitt_x + I_prewitt_y;

% Display the result
figure;
imshow(I_prewitt, []);
title('Prewitt Filter');
```

Prewitt Filter