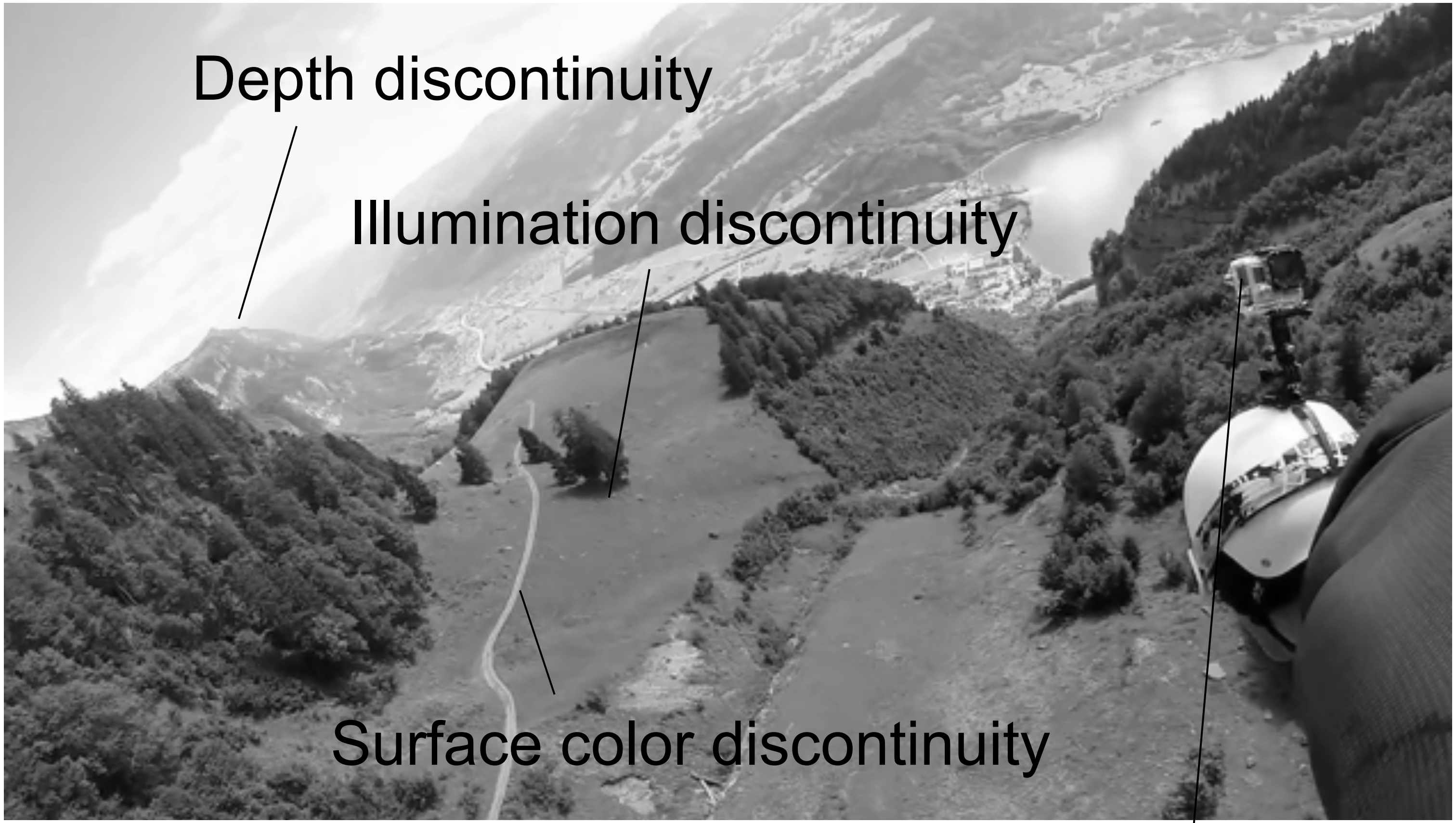
Edge Formation Factors



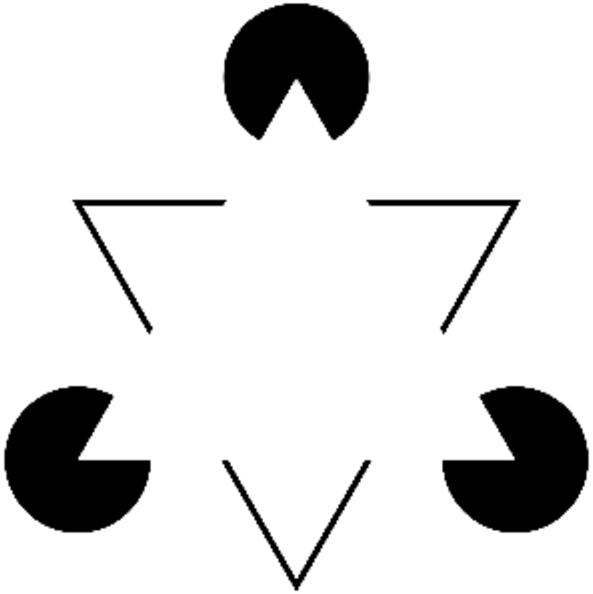
Depth discontinuity

Illumination discontinuity

Surface color discontinuity

Surface normal discontinuity

Kanizsa Triangle

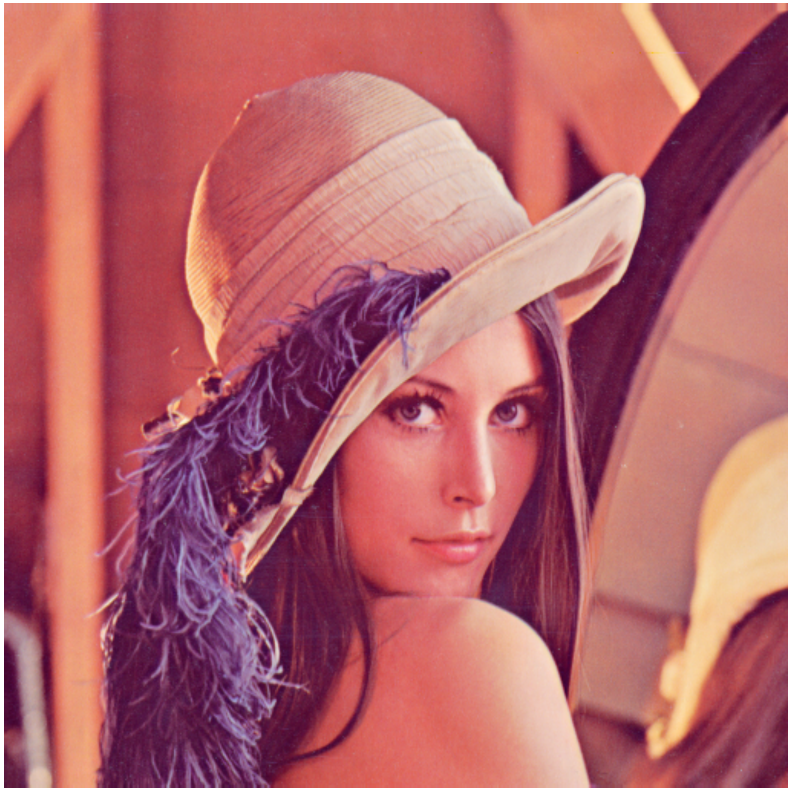


Canny Edge Detection

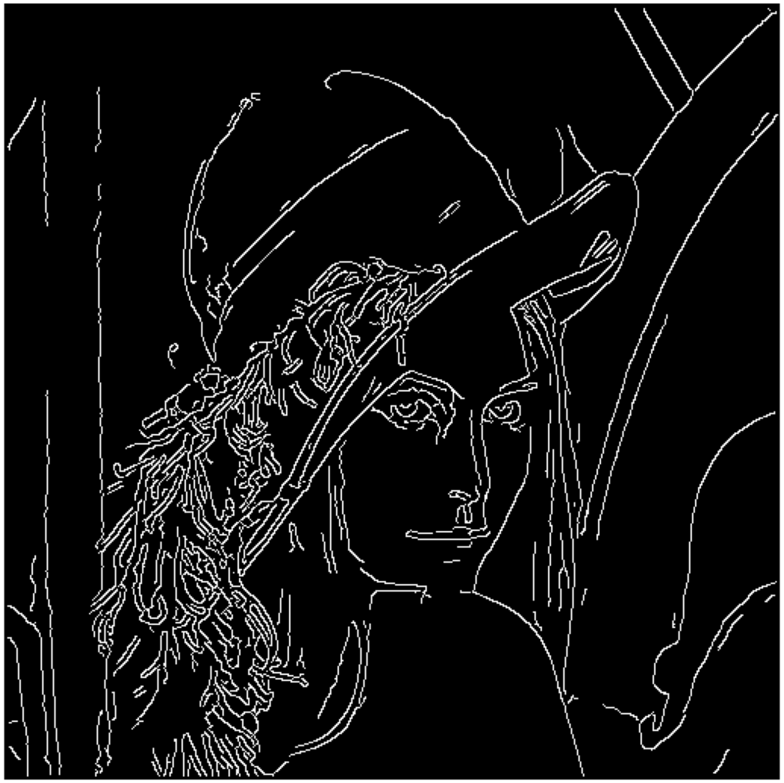
$$B(i,j) = \begin{cases} 1 & \text{if } I(i,j) \text{ is edge} \\ 0 & \text{if } I(i,j) \text{ is not edge} \end{cases}$$

Objective: to localize edges given an image.

Binary image indicating edge pixels



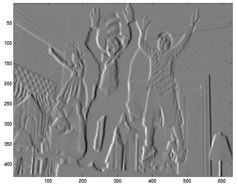
Original image, I



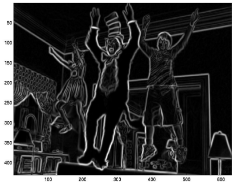
Edge map image, B

Canny Edge Detection

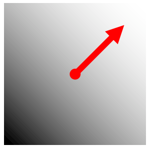
1. Filter image by derivatives of Gaussian



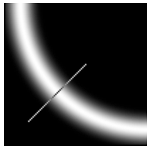
2. Compute magnitude of gradient



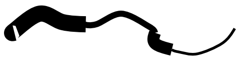
3. Compute edge orientation



4. Detect local maximum



5. Edge linking



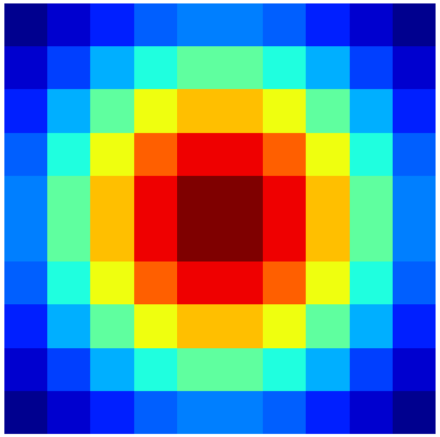
1) Compute Image Gradient

the first order derivative of Image I in x, and in y direction

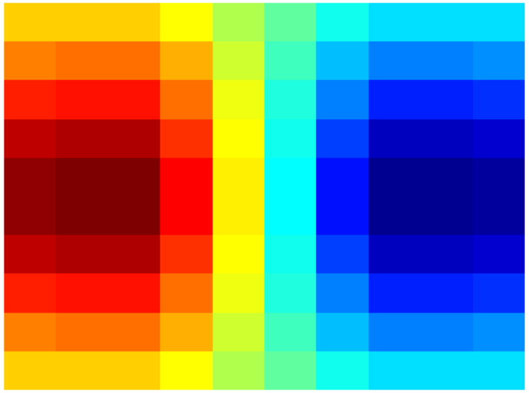
Edge Detection, Step 1,
Filter out noise and compute derivative:

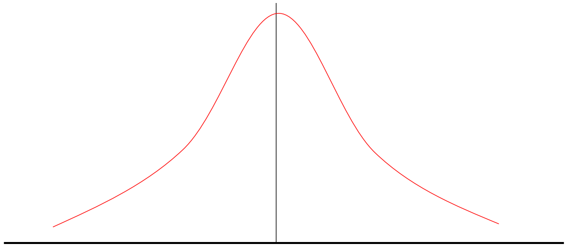
$$\left(\frac{\delta}{\delta x} \otimes G\right)$$

Gradient of Gaussian

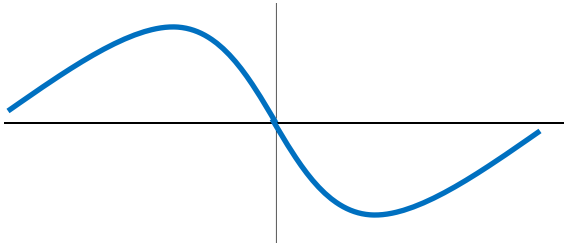


$$\frac{\delta}{\delta x} \longrightarrow$$



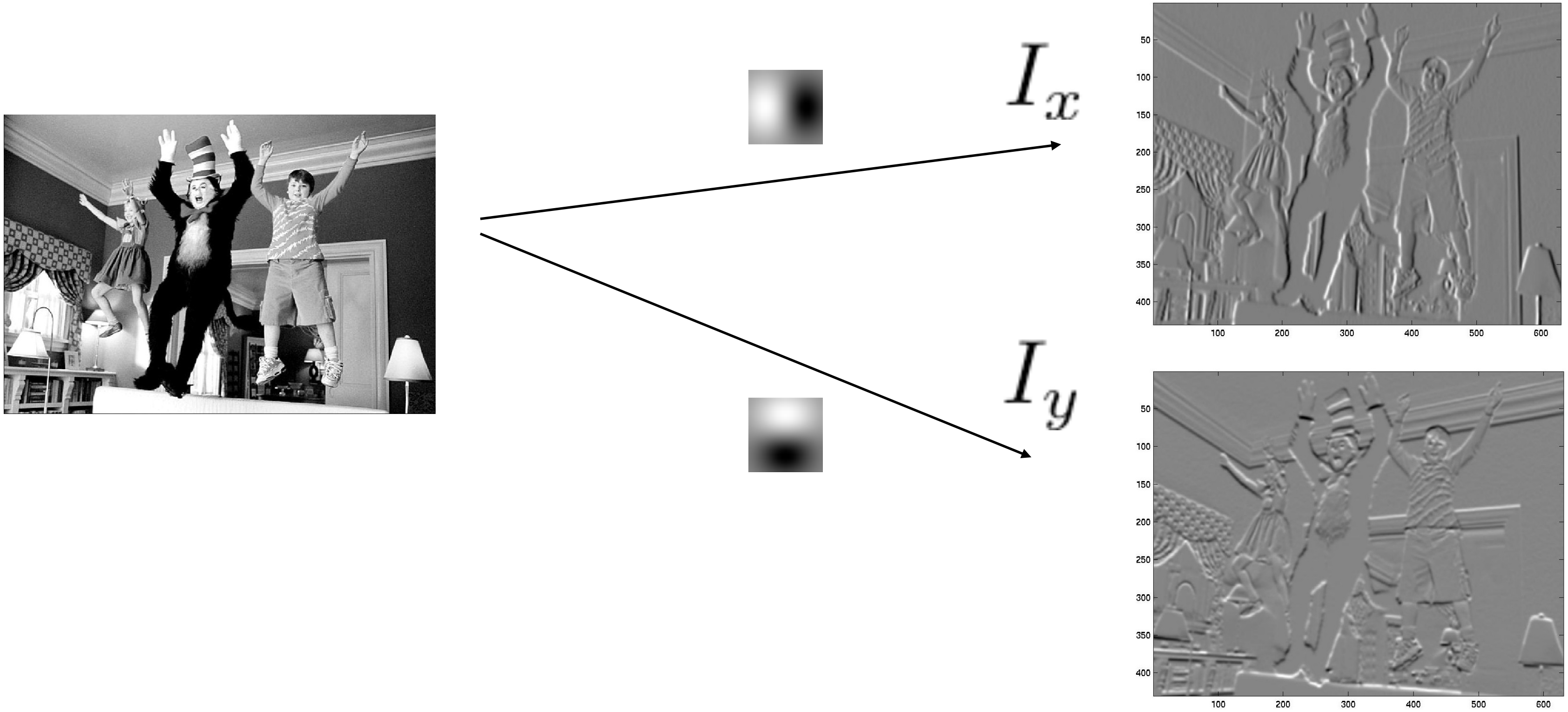


$$\frac{\delta}{\delta x} \longrightarrow$$



Edge Detection, Step 1,
Filter out noise and compute derivative:

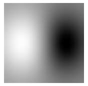
Image $\otimes\left(\frac{\delta}{\delta x}\otimes G\right)$ **Smoothed Derivative**



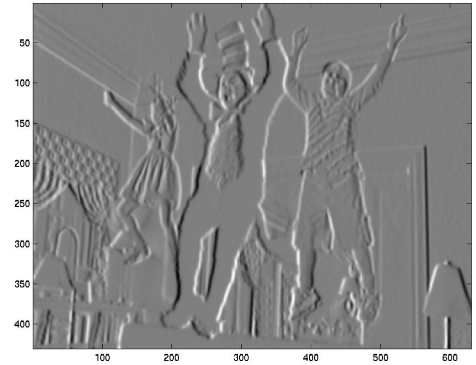
Edge Detection, Step 1,
Filter out noise and compute derivative:

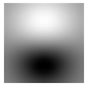
Python:

```
dx, dy = np.gradient(G, axis = (1,0))
Ix = signal.convolve2d(I,dx,'same')
Iy = signal.convolve2d(I,dy,'same')
```

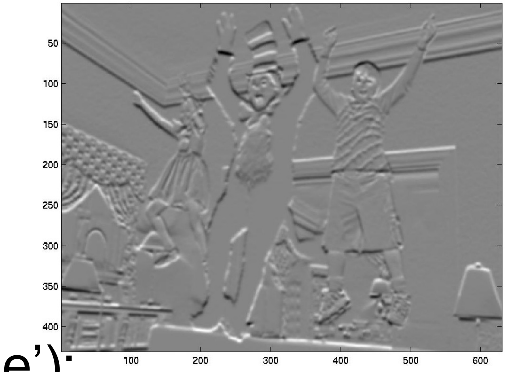


I_x





I_y



MATLAB:

```
>> [dx,dy] = gradient(G); % G is a 2D gaussain
>> Ix = conv2(I,dx,’same’); Iy = conv2(I,dy,’same’);
```

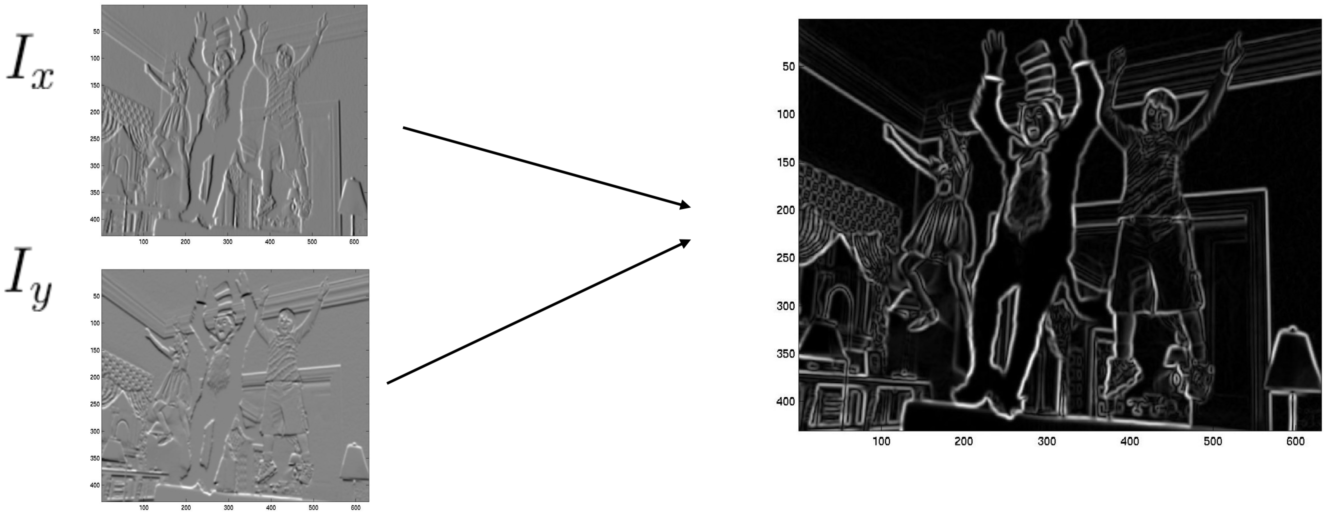
Edge Detection: Step 2
Compute the magnitude of the gradient

Python:

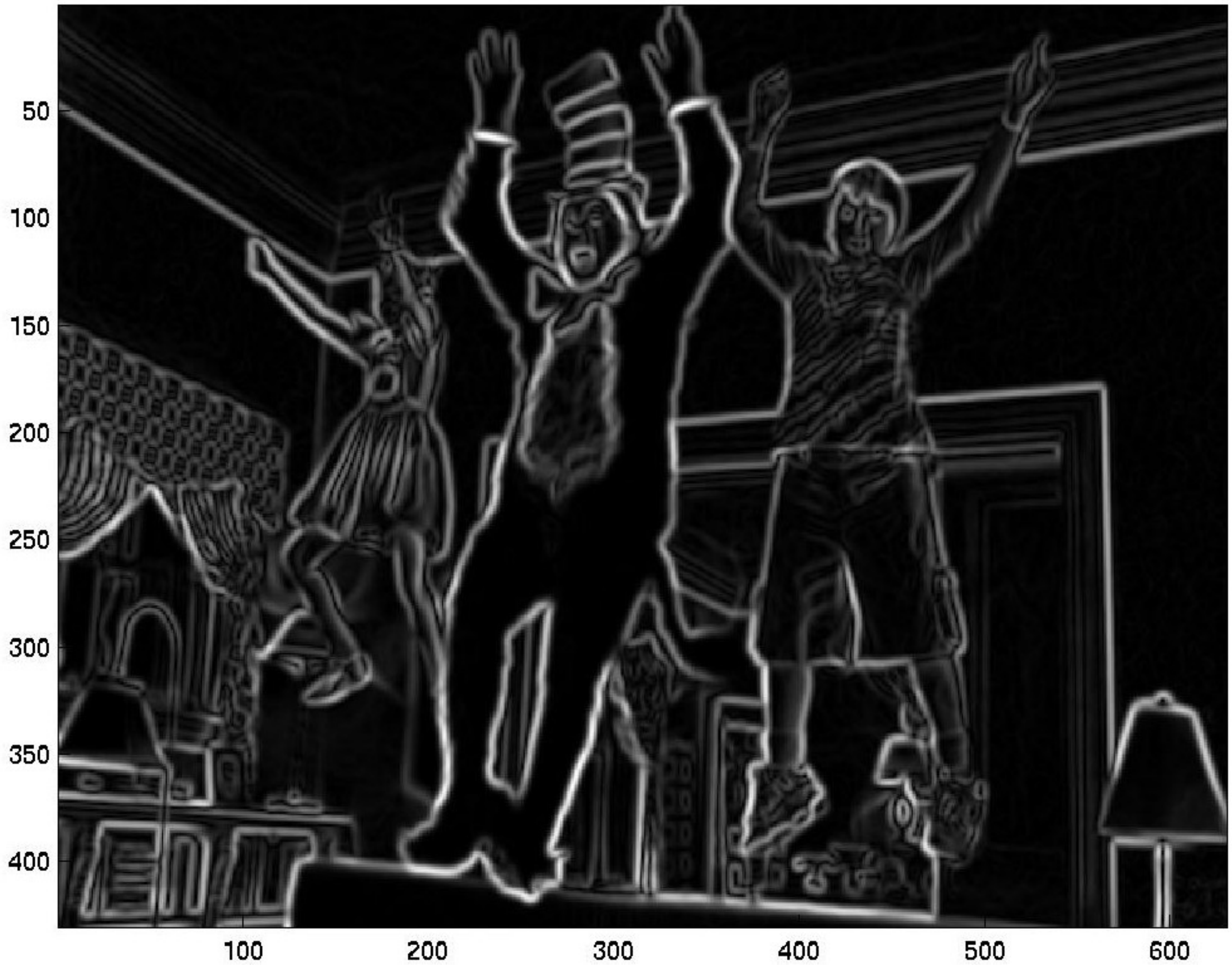
```
Im = math.sqrt(Ix*Ix + Iy*Iy);
```

Matlab:

```
>> Im = sqrt(Ix.*Ix + Iy.*Iy);
```



We know roughly where are the edges, but we need their precise location.


50
100
150
200
250
300
350
400
100
200
300
400
500
600