Why edge detection?
Goal is to have a first pre-processing step to recognise objects in an image, or to do different kinds of processing inside and outside of an object, for instance smoothing inside an object without blurring the edges.

Problem: How can recognize edges in an image?

Approach: compute a derivative/ a gradient of the intensity values of the image, and find the positions where the magnitude of this derivative is particularly large. The gradient of the intensity function $f(x, y)$ is:

$$\nabla f(x, y) = \frac{\partial f(x, y)}{\partial x} i_x + \frac{\partial f(x, y)}{\partial y} i_y$$

where $i_x$ and $i_y$ are vectors in the direction of $x$ and $y$ resp.

The effect of this function can be seen in the following picture (from J. Lims Book):
Here we can see that the 2\textsuperscript{nd} derivative can increase the accuracy of the position finding for the edges, because we can determine the positions of the zeros more precisely more easily than an extremum.

This following scheme is based on the first derivatives:
We are in the (space) discrete case here, so strictly speaking we cannot compute derivatives, which would require infinitesimally small distances $dx$. The smallest distance we have is a pixel distance. If we set this distance to one, the derivative is approximated by the difference between the values of two neighbouring pixels. This leads us to linear filters, particularly high pass filters. Examples can be seen with the following impulse responses of edge detection filters:
In this way we can filter out certain directions, which in some applications might be desirable, but in other applications we would like to have all edges. In that case we can take the magnitude of our earlier gradient:

$$|\nabla f(x, y)| = \sqrt{\left(\frac{\partial f(x, y)}{\partial x}\right)^2 + \left(\frac{\partial f(x, y)}{\partial y}\right)^2}$$

This is the length of the gradient vector, and a measure of the steepness of our edge, independent of its directions. This can be approximated in our discrete case using our horizontal and vertical edge detector filters in above image, which take the roles of the horizontal (dx) and vertical (dy) derivations:
\[ |\nabla f(x, y)| \rightarrow \sqrt{\left[f_x(n_1, n_2)\right]^2 + \left[f_y(n_1, n_2)\right]^2} \]

where \( n_1, n_2 \) are the pixel indices in directions \( x \) and \( y \), resp.

Another example of approximating the length of the gradient can be found in the following image, with the so-called Robert's edge detector:

![Image of Robert's edge detector](image.png)

**Figure 8.29** Impulse responses of filters used in Robert's edge detection method. The method is based on comparison of

\[ \sqrt{(f(n_1, n_2) * h_x(n_1, n_2))^2 + (f(n_1, n_2) * h_y(n_1, n_2))^2} \]

with a threshold.

There are many variations of the edge detection methods discussed in this chapter. For example, we could use a different nonlinear combination of \( \partial f(x, y) \). Here we again have orthogonal derivations/differences, but instead of along the horizontal and vertical axis it is along the diagonals. An example of applying the horizontal and vertical edge detectors to an image can be seen in this picture:
Figure 8.30  Edge maps obtained by directional edge detectors. (a) Image of $512 \times 512$ pixels; (b) result of applying a vertical edge detector; (c) result of applying a horizontal edge detector.
Another, similar edge detector is the so-called Sobel edge detector, that can be seen in the following image (all from: J. Lim, Two-Dimensional Signal and Image Processing):

![Sobel Edge Detector Diagram](image)

**Figure 8.28** Approximation of (a) $\frac{\partial f(x, y)}{\partial x}$ with $f(n_1, n_2) \ast h_x(n_1, n_2)$; (b) $\frac{\partial f(x, y)}{\partial y}$ with $f(n_1, n_2) \ast h_y(n_1, n_2)$. Sobel’s edge detection method is based on comparison of $\sqrt{(f(n_1, n_2) \ast h_x(n_1, n_2))^2 + (f(n_1, n_2) \ast h_y(n_1, n_2))^2}$ with a threshold.

The following image shows a comparison between the Sobel and the Roberts edge detector:
Laplacian-Based Methods:

The laplacian is defined as

\[ \nabla^2 f(x, y) = \nabla (\nabla f(x, y)) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2} \]

Instead of approximating the gradient, we now approximate the laplacian. First we approximate the second derivative in direction \( x \):

\[ \frac{\partial^2 f(x, y)}{\partial x^2} \rightarrow f_{xx}(n_1, n_2) = f(n_1 + 1, n_2) - 2f(n_1, n_2) + f(n_1 - 1, n_2) \]

using a similar approximation for \( y \), we can now approximate the Laplacian:
\[ \nabla^2 f(x, y) \rightarrow \nabla^2 f(n_1, n_2) = f_{xx}(n_1, n_2) + f_{yy}(n_1, n_2) \]

The zeros of the approximated Laplacian now gives us the positions of the edges.