# Part I

# Feature-based image analysis

Analyzing images using feature-based representations is central to many applications. We have chosen three topics that we will cover. First we will work with scale-space for detecting image features independently of scale. Specifically, we will focus on scale-space blob detection. Then we will investigate features as a basis for pixel classification used for image segmentation. Finally, we will work with feature-based image registration.

# 2 Scale-space

Methods from scale-space allow scale invariant detection of image structures. This means that we can find features like blobs (binary large objects), corners, ridges, edges, and other structures at different scale. When we talk about image features like corners and edges, it is not corners or edges of the physical depicted objects, but corners and edges in the image intensities. To visualize this, you can think of a 2D image as a landscape, with pixel intensities corresponding to height measurements at regularly placed positions. In this landscape, an edge is a line where high abruptly changes. A corner will be a height-change point where two (more or less) orthogonal edges meet, and other types of features can be described in the same way.

Using a feature-based image representation is convenient, because we break the image up into manageable parts that are more descriptive than the individual pixels. Scale invariance, which means that we characterize (make a mathematical description) the same feature shown at different scale in two images, is also very convenient. In e.g. computer vision where images of the same object are often captured from difference distance, it is typically a desired property to be able to measure the features independent of its scale. But it also allows us to measure image structures that are different in size for example from microscope images, as we will be working with here.

Here we will base our work on the article of Lindeberg[1] that gives an introduction to scale-space theory. Scale-space representation has made the basis for a range of image analysis methods and is extensively used in computer vision. In the exercise you will implement scale-space blob detection and use it for detecting and measuring the size of fibres that are imaged using X-ray CT.

[1] Tony Lindeberg. Scale-space: A framework for handling image structures at multiple scales. 1996

The computation of scale-space is done by representing image features at all scales at once and detect features based on criteria that is independent of the scale. We will work with the Gaussian scale-space, and the analysis is in practice done by smoothing the image using a Gaussian filter. In Lindeberg[2] the scale-space representation is defined for a general $N$-dimensional signal $f : \mathbb{R}^N \to \mathbb{R}$, but we will use it

[2] Tony Lindeberg. Scale-space: A framework for handling image structures at multiple scales. 1996

for a 2D image $I : \mathbb{R}^2 \to \mathbb{R}$. For a 2D image, its Gaussian scale-space representation is $L : \mathbb{R}^2 \times \mathbb{R}_+ \to \mathbb{R}$, which in practice becomes a 3D object, with the two spatial image dimensions $(x, y)$ and the scale in the third dimension. Since scale is obtained by smoothing with a Gaussian, the variable determining the degree of smoothing is the variance $t$. Also the standard deviation $\sigma = \sqrt{t}$ is used in the article, but here we have simplified the notation and use only the variance $t$.

The Gaussian scale-space $L$ is defined for $N$-dimensional signals by

$$L(x;t) = \int_{\xi \in \mathbb{R}^N} f(x-\xi) g(\xi;t) d\xi \tag{2.1}$$

with $g : \mathbb{R}^N \times \mathbb{R}_+ \to \mathbb{R}$ being the $N$-dimensional Gaussian kernel

$$g(x;t) = \frac{1}{(2\pi t)^{N/2}} e^{-(x_1^2 + \cdots + x_N^2)/(2t)}. \tag{2.2}$$

In practice we will work with the Gaussian scale-space for 2D images on a discrete set of pixels. Therefore, we can write the Gaussian scale-space (ignoring boundary issues) as

$$L(x,y;t) = \sum_{-\gamma}^{\gamma} \sum_{-\delta}^{\delta} I(x-\gamma, y-\delta) g(\delta, \gamma; t) \tag{2.3}$$

where $g : \mathbb{R}^2 \times \mathbb{R}_+ \to \mathbb{R}$ is the 2D Gaussian kernel

$$g(x,y;t) = \frac{1}{2\pi t} e^{-(x^2+y^2)/(2t)}. \tag{2.4}$$

Computing the scale-space is done at a discrete set of steps, and we have the start condition with $t = 0$ defined as $L(x, y; 0) = I(x, y)$.

For feature detection, we need to compute the derivatives of a scale-space representation. Note that this is conveniently achieved by convolving an image with a kernel that is a derivative of a Gaussian. Blob detection uses second order derivatives, more precisely the Laplacian of a Gaussian $\nabla^2 L = L_{xx} + L_{yy}$ which gives a high response where there is a blob in the image. To detect blobs, we need to find local maxima and minima of the Laplacian. Some local maxima and minima will, however, be very weak and they should not be detected as a blob. Therefore, low responses of the Laplacian of the Gaussian should not be included. These low-response blobs are excluded by not including blobs that have an absolute Laplacian response lower than a certain threshold.

What we still need to do is to ensure that we can detect blobs across different scales. The image in scale-space representation is increasingly smoothed, and with increasing scale $t$, pixels will change their intensity value towards the average value of the image. Therefore, the absolute values of derivatives will become smaller when increasing $t$. For blob

detection, this means that the magnitude of the local maxima and minima in the scale-space of the Laplacian $\nabla^2 L$ will decrease and this smoothing must be compensated. The compensation factors for different features are given in Lindeberg[3] and for the blob feature it is $t$ such that the scale normalized Laplacian of Gaussian is $t\nabla^2 L$.

[3] Tony Lindeberg. Scale-space: A framework for handling image structures at multiple scales. 1996

## 2.1 *Exercise 2 – part I, Scale-space blob detection*

In this part of the exercise you will implement scale-space blob detection with the purpose of detecting and measuring glass fibres from images of a glass fibre composite. An image example is given in Figure 2.1, that shows a polished surface of a glass fibre composite sample, where individual fibres can be seen. Since these fibres are relatively circular we will model them as circles. This means that we must find their position (center coordinate) and diameter, and for this we will use the scale-space blob detection. After having computed the fibres parameters, we will carry a statistical analysis of the results.

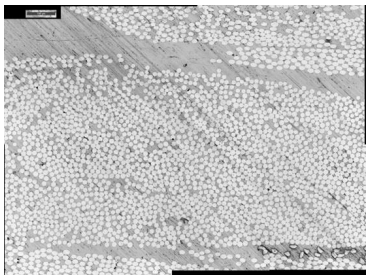

Figure 2.1: Example of fibre image acquired using an optical microscope.

### 2.1.1 *Computing Gaussian and its second order derivative*

We will approach this analysis in steps that lead to the final algorithm. First we will use synthetic data to develop and test our algorithm, and after that we will carry out the analysis on the real images.

Since we focus on blob detection, we must have a Gaussian kernel and its second order derivative. Some convolution libraries have already implemented the second order derivative of a Gaussian that you are welcome to use for the exercise. But we will anyhow start investigating the second order derivative of a Gaussian, which you will be using for blob detection.

The Gaussian is separable and we can employ 1D filters for our analysis, which you will compute now. The 1D Gaussian is given by

$$g(x) = \frac{1}{\sqrt{2\pi t}} e^{\frac{-x^2}{2t}} \ . \tag{2.5}$$

*Suggested procedure*

1. Derive (analytically) the second order derivative of the Gaussian

$$\frac{\mathrm{d}^2 g}{\mathrm{d}x^2} \ .$$

2. Implement a function that takes the variance $t$ as input and outputs a filter kernel of $g$ and $\mathrm{d}^2 g/\mathrm{d}x^2$. You should use a filter kernel with a size of at least $\pm 3\sqrt{t}$. Why? (*Hint:* Set a variable $\nu = \lceil 3\sqrt{t} \rceil$, make an array with the integer values $[-\nu, -\nu+1, ..., \nu-1, \nu]$, and compute the Gaussian on these values.)

3. Try the filter kernel on the synthetic test image `test_blob_uniform.png` and inspect the result.

### 2.1.2 *Detecting blobs at one scale*

Here you going to implement a function to detect blobs at a single scale. Blobs can be found as spatial maxima (dark blobs) or minima (bright blobs) of the scale-space Laplacian

$$\nabla^2 L = L_{xx} + L_{yy} \, . \tag{2.6}$$

*Suggested procedure*

1. Compute the Laplacian at one scale using the synthetic test image `test_blob_uniform.png`.
2. Create a function that detects the coordinates of maxima and minima in the Laplacian image (detect blobs), and that has an absolute value of the Laplacian larger than some threshold.
3. Plot the center coordinates and circles outlining the detected blobs. The radius of the circles should be $\sqrt{2t}$.
4. Try varying $t$ such that the blobs in `test_blob_uniform.png` are exactly outlined.

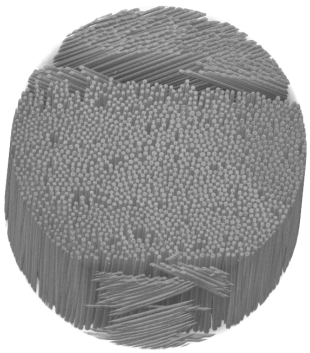

Figure 2.2: Visualization the 3D fibers scanned with the high resolution X-ray CT-scanner.

### 2.1.3 *Detecting blobs on multiple scales*

Now you should extend the blob detection at a single scale to multiple scales. To find blobs at multiple scales, we must use the scale-space representation. This can conveniently be done by representing $\nabla^2 L$ as a 3D array (volumetric image).

*Suggested procedure*

1. Decide on scales at which the Laplacian must be computed. A good idea is to make it equal steps in $t$. Remember that the radius of the blobs are $\sqrt{2t}$, so you can look at the size of the structures that you want to detect to decide a good range of scales.
2. Compute the scale normalized scale-space Laplacian $t\nabla^2 L$ for the test image `test_blob_uniform.png`. It is very important that you remember to scale normalize, i.e. that you multiply the Laplacian $\nabla^2 L$ by $t$ to be sure to detect the correct scales.
3. Find coordinates and scales of maxima and minima in this scale-space and plot the detected blobs on top of the image. What are the detected scales and what is the diameter of the blobs?

4. Detect blobs in the test image `test_blob_varying.png`.
5. Verify that you detected the blobs at the correct scale by showing an image where you plot circles with a diameter of $\sqrt{2t}$ on top of the detected blobs.

### 2.1.4 *Detecting blobs in real data*

We will now continue with the real images of fibers. The fibre data is obtained using different scanning methods including scanning electron microscopy (`SEM.png`), optical microscopy (`Optical.png`), synchrotron X-ray CT (`CT_synchrotron.png`), and three resolutions of laboratory X-ray CT (`CT_lab_high_res.png`, `CT_lab_med_res.png`, `CT_lab_low_res.png`). The CT data is a single slice very close to the top, so we assume the data to be from the same part of the sample, and this allows us to directly compare the fibers. We will do this comparison in next exercise, but in this we will compute the fiber location and their diameter. In Figure 2.2 you can see a visualization of the fibre data from the high resolution X-ray CT scan.

We start by testing the blob-detection on this real data.

*Suggested procedure*

1. Run your blob-detection function from above on a cut-out example of one of the images. It is important that you tune your parameters to get the best possible results.

### 2.1.5 *Localize blobs*

It turns out, that it is difficult to detect blobs in the Laplacian scale-space in the fiber image, such that all fibers are found. To overcome this, we will detect the fibers as maxima in a Gaussian smoothed image. Since the fibers are almost the same size, we can use a single scale of the Gaussian to detect the fiber centers.

*Suggested procedure*

1. Smooth an image of fibers with a Gaussian and visualize the result.
2. Find locations of maxima in this image and plot the positions on top of the original image.
3. Compute the Laplacian scale-space for the image.
4. Find the scale of each fibre as the minimum over scales at the fiber locations.
5. Plot circles according to the found scale on top of the original image.

6. Detect fibers in all six fiber images. Save the locations and diameters.

In the exercise in Week 4, where you will work with feature-based image matching, you will use the results obtained in this exercise. So, it will be possible to continue working on the parts that you did not finish here in Week 4.