Assignment: Feature Detection and Scale-Space Theory
Signal and Image Processing

March 4, 2020

Herein you will find both the hands-on exercises to help you learn the week's material. You can work on this and submit your solution (report and code) as a GROUP. This assignment does not count towards your grade, but have to submitted in order to pass the course.

Report Guidelines

- Your answers should be complete, but concise.
- Deliverables should be added to the report. Make sure you actually answer all parts of the questions.
- If you are asked to comment about the result, this usually means we are looking for 1 - 2 sentences that show you have understood the outcome.
- Any figures (images, graphs etc) should be at a size suitable for their purpose. For example, if you want to show pixel-level differences, then the pixels should be large enough to be seen. Use a zoomed portion of the image if necessary.
- ALL figures, images, graphs, tables etc should be numbered, (e.g. Figure 1, Table 3), and include at least a minimal caption (e.g. "Figure 1: Original image rotated by 45 degrees").
- All graph axes need to be labelled, and axes scales included.
- Images should include a labelled colour scale (a 'colorbar') where appropriate.
- All text in legends, labels, titles, scales etc should be readable, e.g. be set in a readable font size.
- When asked to show your code, this usually means code 'snippets'. That is, the few important lines of code that are the most relevant for the question. Unless specified otherwise, exclude the 'book-keeping' code (e.g. import statements, setting paths, loading/saving data etc).
- You should always submit your report as a separate PDF file and your complete code (.py files) in a compressed archive (we can uncompress .zip or .tar.gz - NOT .rar format).

1. Fixed scale feature detectors

1.1. Use the `skimage.feature.canny` function on the `hand.tiff` image. Try different settings for the parameters `sigma`, `low_threshold`, and `high_threshold` of the `canny` function.
Deliverables: Include an illustration showing the results of the different settings and explain what the effect is of each of the parameters based on these results.

1.2. Use the `skimage.feature.corner_harris` function on the `modelhouses.png` image to compute a Harris corner response image (a.k.a. feature map). Try different settings for the parameters `sigma` and `k` for `method='k'` of the `corner_harris` function. Also try to fix `sigma` and change method to `method='eps'` and try different values of the `eps` parameter.
Deliverables: Include an illustration showing the results of the different settings and explain what the effect is of each of the parameters based on these results.

1.3. Write a Python function that finds local maxima in the feature map generated by the `skimage.feature.corner_harris` function by using the function `skimage.feature.corner_peaks`.
Deliverables: Include in the report the code for your function. Apply this function to the `modelhouses.png` image and create a figure of the resulting corner points overlaid on the `modelhouses.png` image. Remember to indicate your choice of parameter settings in the caption of the figure.

2. Scale-space features and detectors

2.1. Consider a Gaussian kernel

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \ . \tag{1}$$

The convolution of a Gaussian with itself is also a Gaussian and we have that

$$G(x, y, \sigma) * G(x, y, \tau) = G(x, y, \sqrt{\sigma^2 + \tau^2}) \ . \tag{2}$$

Consider the function made from this analytical expression,

$$B(x, y) = G(x, y, \sigma) \ . \tag{3}$$

We will use this function as a synthetic model of a blob in an image.
Using Python, make a discrete image from the function in eq. (3) for some fixed σ by sampling the x, y axes appropriately. Visually illustrate your solution and eq. (2) by calculating images from its scale-space,

$$I(x, y, \tau) = B(x, y) * G(x, y, \tau) \ . \tag{4}$$

Deliverables: Provide a code snippet and explanation of your solution as well as illustrations of your solution.

2.2. Next you should prove that the expression in eq. (2) is correct.
Deliverables: Mathematical proof with essential steps and explanations.

2.3. Consider the 2-dimensional scale normalized partial derivatives of order $m+n$ at scale τ,

$$I_{x^m y^n}(x, y, \tau) = \tau^{\gamma(m+n)} \frac{\partial^{m+n} I(x, y, \tau)}{\partial x^m \partial y^n}, \tag{5}$$

where $\gamma \in \mathbb{R}$ is a parameter of the scale normalization and $I(x, y, \tau)$ is the scale space of an image. Now consider the scale normalized Laplacian of the blob image defined in eq. (3),

$$H(x, y, \tau) = I_{xx}(x, y, \tau) + I_{yy}(x, y, \tau), \tag{6}$$

Using $\gamma = 1$, solve the following:

i. **Deliverables:** Write the closed form expression for $H(x, y, \tau)$ including the essential steps and explanation in the development of the expression.

ii. Consider the point $(x, y) = (0, 0)$ and derive analytically the scale(s), τ, for which $H(0, 0, \tau)$ is extremal. A CAS tool such as Maple (or Wolfram Alpha) may be helpful, or simply solve it by hand.
Deliverables: Mathematical derivation with essential steps and explanations. Characterize these extremal point(s) in terms of maximum, saddle, and minimum in (x, y, τ).

iii. **Deliverables:** Confirm your result in Python by plotting $H(0, 0, \tau)$ as a function of τ using the expression from 2.3.i.

iv. Locating the maxima and minima (x, y, τ) in the scale-space of eq. (6) applied to images $I(x, y)$ in general is called blob detection with scale selection. Write a Python script that detects the 20 maxima and minima in the scale-space of the `sunflower.tiff` image with the largest absolute value of eq. (6). Indicate each detected point and corresponding detection scale τ with a dot and a circle centred on the point of detection with a radius proportional to τ. Choose different colors for the circle and point so you can distinguish maxima from minima.
Deliverables: An illustration of your results and code snippets showing essential steps in your implementation. What image structure does maxima of eq. (6) represent, and what image structure does minima represent?

2.4. Consider this model of a vertical soft edge going through the origin $(0, 0)$,

$$S(x, y) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x')^2}{2\sigma^2}} \, dx' \tag{7}$$

for some constant σ that defines the smoothness / softness across the edge. Note that it is not a mistake that y does not appear on the right hand side of the equation. Consider also its scale-space,

$$J(x, y, \tau) = S(x, y) * G(x, y, \tau), \tag{8}$$

and the scale-normalized squared (spatial) gradient magnitude operator

$$\begin{aligned} \|\nabla J(x, y, \tau)\|^2 &= J_x^2(x, y, \tau) + J_y^2(x, y, \tau) \\ &= \tau^{2\gamma} \left(\frac{\partial J(x, y, \tau)}{\partial x} \right)^2 + \tau^{2\gamma} \left(\frac{\partial J(x, y, \tau)}{\partial y} \right)^2 . \end{aligned} \tag{9}$$

Using Python, make a discrete image from the function in eq. (7) for some fixed σ by sampling the x, y axes appropriately (Hint: Remember that the discrete approximation of an integral is a sum. You might want to use something like the `cumsum` function along the x axis). Visualize the scale-space eq. (8) of this soft edge model by computing some images at a couple of scales τ of your choice.
Deliverables: Provide a code snippet and explanation of your solution as well as illustrations of your solution.

2.5. Using t the scale-normalized squared gradient magnitude operator eq. (9) and $\gamma = \frac{1}{2}$, solve the following:

i. **Deliverables:** Write the closed form expression for $\|\nabla J\|^2$ for the soft edge model eq. (7) including the essential steps and explanation in the development of the expression. Hint: You might need the fundamental theorem of calculus.

ii. Derive analytically the scale, τ, for which $\|\nabla J\|^2$ is maximal in the point $(x, y) = (0, 0)$.
Deliverables: Mathematical derivation with essential steps and explanations. Is this a maximum in scale-space (x, y, τ)?

iii. **Deliverables:** Confirm your result in Python by plotting $\|\nabla J(0, 0, \tau)\|^2$ as a function of τ using the expression from 2.5.i.

iv. The maxima in (x, y, τ) of (9) for any image $I(x, y)$ defines edge detection with scale-selection. Implement this scale-space edge detector in Python. In the `hand.tiff` image, detect the 100 maxima with the largest value of eq. (9) and indicate the point of detection and scale by circles with a radius proportional to the detection scale τ.
Deliverables: An illustration of your results and code snippets showing essential steps in your implementation.