Computer Vision TP5: 
Harris Corners and Matching

Martin de La Gorce

The goal will be to implement an harris corner detector and match the harris corners of two images using the sum of square differences.

1 some reformulation of the harris detector

- Given some \( \sigma \), we start by defining \( w_\sigma \) the gaussian with standard deviation \( \sigma \) with truncated support outside the square of size \( 6\sigma \):

\[
  w_\sigma(x, y) = \frac{1}{Z} \exp \left( -\frac{(x^2 + y^2)}{2\sigma^2} \right) \quad \text{if} \quad \max(|x|, |y|) < 3\sigma, 0 \quad \text{otherwise}
\]

and \( Z \) a normalization coefficient such that \( \sum_{x,y} w_\sigma(x, y) = 1 \) (1)

- Using such a gaussian with \( \sigma \) we compute a smoothed image \( I_{\text{smooth}} \) that corresponds to a smoothed version of the image \( I \) using a Gaussian filter with standard deviation \( \sigma_1 = 2 \)

\[
  I_{\text{smooth}} = I * w_{\sigma_1}
\]

- We compute \( I_x \) and \( I_y \) the gradient of the smoothed image \( I_{\text{smooth}} \)

- Given an image \( I \) we compute the harris score for each pixel \((i, j)\) of the image a matrix \( M_{ij} \) using

\[
  M_{ij} = \sum_{x,y} w_{\sigma_2}(x-i, y-j) \begin{bmatrix} I_x(x, y)^2 & I_x(x, y)I_y(x, y) \\ I_x(x, y)I_y(x, y) & I_y(x, y)^2 \end{bmatrix}
\]

With the parameter \( \sigma_2 = 3 \) controlling the size of the window \((6\sigma_2)\) used as context when computing the harris score. We introduce four images
\[ m_{11}, m_{12}, m_{21}, m_{22} \] such that
\[
M_{ij} = \begin{bmatrix}
m_{11}(i, j), m_{12}(i, j) \\
m_{21}(i, j), m_{22}(i, j)
\end{bmatrix}
\]
We get
\[
m_{11} = \sum_{x,y} w_{\sigma_2}(x - i, y - j) I^2(x, y)
\]
\[ m_{11} \] is the convolution of the image \( I^2(x, y) \) by a gaussian kernel i.e
\[
m_{11} = w_{\sigma_2} \ast I^2_x
\]
I also have \( m_{12} = m_{21} = w_{\sigma_2} \ast (I_x I_y) \) and \( m_{22} = w_{\sigma_2} \ast I^2_y \). The Harris score writes
\[
R(i, j) = \det(M_{ij}) - k(\text{trace}(M_{ij}))^2
\]
with \( k = 0.06 \). We have \( \det(M_{ij}) = m_{11}(i, j)m_{22}(i, j) - m_{12}(i, j)m_{21}(i, j) \) and \( \text{trace}(M_{ij}) = m_{11}(i, j) + m_{22}(i, j) \) therefore using element-wise operators on arrays we get
\[
R = m_{11}m_{22} - m_{12}m_{21} - k(m_{11} + m_{22})^2
\]

2 Exercise

1. Implement a function smoothGrad that computes the smoothed gradients \( I_x \) and \( I_y \) given an image \( I \) and a standard deviation \( \sigma \)

2. Implement a function HarrisScore that computes the harris score image \( R \) given an image \( I \), \( \sigma_1, \sigma_2 \) and \( k \)
3. Implement a function `HarrisCorners` that calls `HarrisScore` to get the Harris score image $R$, find local maximums that are above 0.005 times the global maximum of $R$ and that local maximums in a radius of 2 pixels using `skimage.feature.peak.peak_local_max` (see hough transform code from previous programming exercise) and return the list of peaks. If you display the list of peaks you should get:

![corners image 1](image1.png)  ![corners image 2](image2.png)

4. Implement a function `SSDTable` that takes two array of patches respectively of size $M_1 \times N \times N$ and $M_2 \times N \times N$ and computes a matrix $D$ with $D_{ij}$ the sum of square differences between the intensities of patch $i$ in the first list of patches and $j$ in the second list of patches.

5. Implement a function `NCCTable` that takes two array of patches respectively of size $M_1 \times N \times N$ and $M_2 \times N \times N$ and computes a matrix $D$ with $D_{ij}$ equal to one minus the cross correlation between patch $i$ in the first list of patches and $j$ in the second list of patches. In order to avoid removing the mean and dividing by the norm of the match for each pair of patches, compute first centered and normalized patches (see slides) and then you only need to compute scalar product between patches.

6. Using the function `extractMatches` and `displayMatches2` display the matches between the two images using either the score given by `SSDTable` or `NCCTable`. You should get something similar to the image next page.