جـــامعة هواري بومـــدين للعلـــوم و التكنـــولوجيـــا
Université des Sciences et de Technologie Houari Boumediene
Faculté d'Informatique

Année 2025/2026
Master Informatique Visuelle
Vision Artificielle

Série de travaux pratiques n°1
Vision Artificielle

Exercice 1.

Le code du prog1.py permet d'appliquer la convolution d'une image avec un filtre Gaussien : $g(x,y) = f(x,y) \otimes G_\sigma$

1- Transformer le code fourni en prog2.py qui réalise les itérations suivantes :

$g(x,y) = f(x,y) \otimes G_\sigma$
For i=1 to n Do
    $g(x,y) = g(x,y) \otimes G_\sigma$
EndDo

La valeur de sigma est communiquée comme paramètre à la fonction : cv.GaussianBlur. Voir en fin de la série la syntaxe de l'appel à la fonction.

Convolution with a Gaussian is a linear operation, so a convolution with a Gaussian kernel followed by a convolution with again a Gaussian kernel is equivalent to convolution with the broader kernel. *Of course we can concatenate as many blurring steps as we want to create a larger blurring step.* [Réf: https://pages.stat.wisc.edu/~mchung/teaching/MIA/reading/diffusion.gaussian.kernel.pdf]

Exercice 2

Le programme en Python fourni « Sift_Detect_Draw.py » permet de détecter et visualiser les points SIFT sur l'image. Changez ce code pour lire deux images (query, train) et calculer les descripteurs SIFT.
Visualisez les attributs du point et son descripteur. A noter que l'image requête (query) est un objet de l'image modèle (train).

Il s'agit ensuite de rechercher l'image requête dans l'image modèle en utilisant la mise en correspondance des descripteurs SIFT. Deux codes sont fournis pour utilisation et compréhension : BruteForceMatchingSorting.py et BruteForceMatching.py
Les figures 1 et 2 illustrent les images à traiter et le résultat recherché.

Figure 1. Images requête et modèle (query and train)

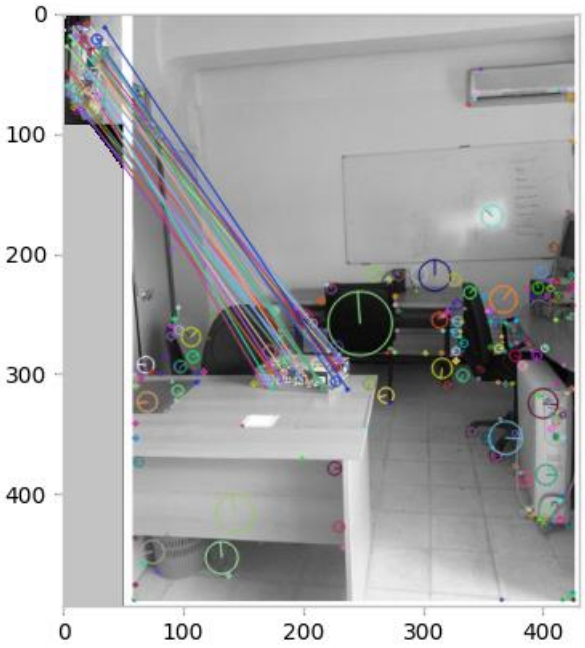

Figure 2. Output recherché

Nous disposons d'une base d'images de véhicules autorisés à entrer au parking d'un établissement. Un traitement automatique ou manuel nous permet d'isoler les plaques d'immatriculation. Les données récupérées constituent la dataset « train ».
Nous voulons vérifier si le véhicule qui se présente en entrée du parking est autorisé ou non. Pour cela nous ferons une identification de son matricule.
Une des solutions à étudier consiste à vérifier si le descripteur SIFT à lui seul permet de réaliser cette tâche. Il s'agit de vérifier si l'image requête (test) contient une région identique à une région (matricule) modèle.

Travail demandé :

Pour les images fournies (query, train), fournir le code qui recherche pour chaque image requête l'image modèle la contenant.

Annexe:

void cv::GaussianBlur(InputArray src, OutputArray dst, Size ksize, double sigmaX, double sigmaY = 0, int borderType =BORDER_DEFAULT)

Python:
cv.GaussianBlur(src, ksize, sigmaX[, dst[, sigmaY[, borderType]]]) -> dst
#include <opencv2/imgproc.hpp>

Blurs an image using a Gaussian filter.

The function convolves the source image with the specified Gaussian kernel. In-place filtering is supported.

Parameters

Src	input image; the image can have any number of channels, which are processed independently, but the depth should be CV_8U, CV_16U, CV_16S, CV_32F or CV_64F.
Dst	output image of the same size and type as src.
Ksize	Gaussian kernel size. ksize.width and ksize.height can differ but they both must be positive and odd. Or, they can be zero's and then they are computed from sigma.
sigmaX	Gaussian kernel standard deviation in X direction.
sigmaY	Gaussian kernel standard deviation in Y direction; if sigmaY is zero, it is set to be equal to sigmaX, if both sigmas are zeros, they are computed from ksize.width and ksize.height, respectively (see getGaussianKernel for details); to fully control the result regardless of possible future modifications of all this semantics, it is recommended to specify all of ksize, sigmaX, and sigmaY.
borderType	pixel extrapolation method, see BorderTypes. BORDER_WRAP is not supported.