جامعة هواري بومدين
للعلوم والتكنولوجيا
USTHB

Année 2023/2024
Master Informatique Visuelle

Série de travaux pratiques n°1
Vision Artificielle

Exercice 1 :

1- Le programme homography1.py permet de sélectionner dans une image (indiquée dans le code source) une zone définie par quatre points localisés par l'utilisateur par clic de la souris. Les points sont sélectionnés dans le sens des aiguilles d'une montre. La zone est ensuite mappée avec une homographie dans un rectangle (width, heigth) (voir figure 1).

Décrire le code fourni et appliquez le sur des images acquises par votre caméra.

2- Le programme homography2.py permet de placer une image source (argument 1) dans une zone de l'image destination (argument 2) (voir figure 2). Appliquez différentes déformations pour avoir différents résultats.

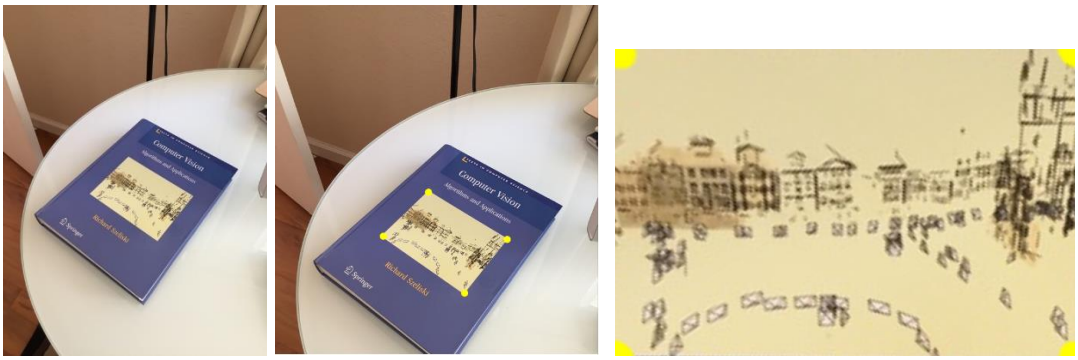

Figure 1. De gauche à droite : Image source, localisation de la zone (coins en jaune), résultat obtenu.

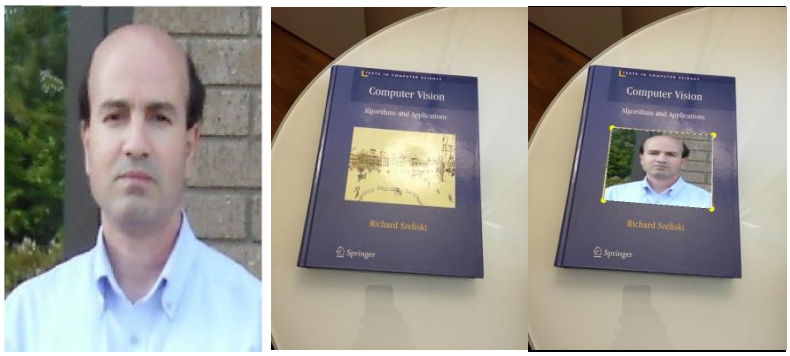

Figure 2. De gauche à droite : Image source (argument 1), image destination (argument 2), résultat d'insertion de l'image source dans la région délimitée par l'utilisateur.

Exercice 2 :

Le programme stitch.py permet de réaliser une image panoramique à partir de deux images (image droite et image gauche).

1- Appliquer le programme sur une paire d'images
2- Appliquer le programme sur une séquence d'images prises par une caméra en rotation.
3- Réalisez une image panoramique avec vos propres images (3 au minimum).

Tools for OpenCV-Python programming

Finding Homography

FindHomography(): Finds a perspective transformation between two planes.

```
Mat cv::findHomography   (InputArray _srcPoints,
                          InputArray_dstPoints,
                          int  method = 0,
                          double          ransacReprojThreshold = 3,
                          OutputArray mask = noArray(),
                          const int       maxIters = 2000,
                          const double confidence = 0.995
                          )
```

Python:
cv.findHomography(srcPoints, dstPoints[, method[, ransacReprojThreshold[, mask[, maxIters[, confidence]]]]]) -> retval, mask

Parameters
srcPoints: Coordinates of the points in the original plane, a matrix of the type CV_32FC2 or vector<Point2f> .
dstPoints: Coordinates of the points in the target plane, a matrix of the type CV_32FC2 or a vector<Point2f> .
method: Method used to compute a homography matrix. The following methods are possible:

- 0 - a regular method using all the points, i.e., the least squares method
- RANSAC - RANSAC-based robust method
- LMEDS - Least-Median robust method
- RHO - PROSAC-based robust method

ransacReprojThreshold: Maximum allowed reprojection error to treat a point pair as an inlier (used in the RANSAC and RHO methods only). That is, if $\|dstPoints_i - convertPointsHomogeneous(H * srcPoints_i)\|^2 > ransacReprojThreshold$ then the point i is considered as an outlier. If srcPoints and dstPoints are measured in pixels, it usually makes sense to set this parameter somewhere in the range of 1 to 10.
Mask: Optional output mask set by a robust method (RANSAC or LMeDS). Note that the input mask values are ignored.
maxIters: The maximum number of RANSAC iterations.
Confidence: Confidence level, between 0 and 1.

Apply a perspective transformation (Warping) to an image.

warpPerspective():

```
void cv::warpPerspective    (
                        InputArray    src,
                        OutputArray dst,
                        InputArray    M,
                        Size    dsize,
```

```
                    int      flags = INTER_LINEAR,
                    int      borderMode = BORDER_CONSTANT,
                    const Scalar &borderValue = Scalar()
)
```

Python:
cv.warpPerspective(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]]) -> dst

The function warpPerspective transforms the source image using the specified matrix:

$$\texttt{dst}(x,y) = \texttt{src}\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}\right)$$

Parameters
Src: input image.
Dst: output image that has the size dsize and the same type as src .
M: 3×3 transformation matrix.
dsize: size of the output image.
flags: combination of interpolation methods (INTER_LINEAR or INTER_NEAREST) and the optional flag WARP_INVERSE_MAP, that sets M as the inverse transformation (dst→src).
borderMode: pixel extrapolation method (BORDER_CONSTANT or BORDER_REPLICATE).
borderValue: value used in case of a constant border; by default, it equals 0.