Corrigé Série 3

Calibrage

Exercice 1

1. Etant donné un point A de coordonnées (5,7) et un angle de rotation de 30 degrés. Calculer les coordonnées de ce point après rotation et donner la matrice de rotation.

$$\begin{bmatrix} \cos 30° & -\sin 30° \\ \sin 30° & \cos 30° \end{bmatrix} \cdot \begin{bmatrix} 5 \\ 7 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 \\ 0.5 & 0.866 \end{bmatrix} \cdot \begin{bmatrix} 5 \\ 7 \end{bmatrix} = \begin{bmatrix} 0.83 \\ 8.56 \end{bmatrix}$$

2. Etant donné un point A (4,2) et un point B (-2,4), trouver l'angle de rotation qui permet d'amener le point A sur le point B ainsi que :

- la matrice de rotation qui permet de passer du point A au point B

$$\begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 2 \end{bmatrix} = \begin{bmatrix} -2 \\ 4 \end{bmatrix} \qquad \begin{aligned} 4\cos\alpha - 2\sin\alpha &= -2 \\ 4\sin\alpha + 2\cos\alpha &= 4 \end{aligned} \qquad \alpha = 90° \qquad \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

- la matrice de rotation qui permet de passer du point B au point A

$$\begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \cdot \begin{bmatrix} -2 \\ 4 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix} \qquad \begin{aligned} -2\cos\alpha - 4\sin\alpha &= 4 \\ -2\sin\alpha + 4\cos\alpha &= 2 \end{aligned} \qquad \alpha = -90° \qquad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

On constate qu'une rotation dans le sens négatif possède une matrice transposée par rapport à celle qui décrit une rotation du même angle dans le sens positif.
Une matrice de rotation transposée est identique à la matrice inversée.

Exercice 2

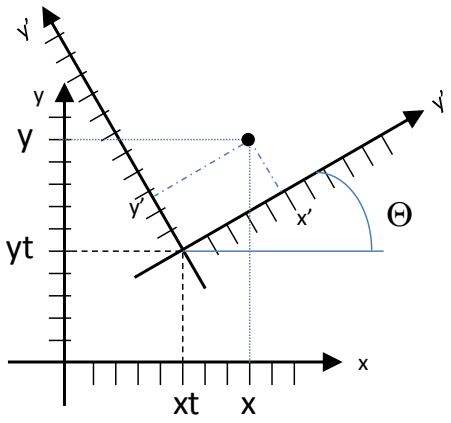
On donne un point A de coordonnées (x,y) dans le référentiel 0xy. Le référentiel Ox'y' a été translaté de (xt,yt), puis tourné de Θ.

1. Trouver la matrice de rotation et le vecteur de translation qui permettent de changer de référentiel.

2. Donnez cette transformation en coordonnées homogènes

3. Quels sont les coordonnées de A point dans le référentiel Ox'y'

4. Qu'en est-il de changement de référentiel inverse ?



Pour valider les calculs : A=(x,y)=(8,10), T=(tx,ty)=(5,5), Θ=30 degrés et A'=(5,2.8)

A. Translation de (-xt,-yt), puis une rotation de -30deg

1. Trouver la matrice de rotation et le vecteur de translation qui permettent de changer de référentiel.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \left(\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -x_t \\ -y_t \end{pmatrix} \right)$$

2. Quels sont les coordonnées de A point dans le référentiel Ox'y'

$$\begin{pmatrix} \cos 30 & \sin 30 \\ -\sin 30 & \cos 30 \end{pmatrix} \begin{pmatrix} 8-5 \\ 10-5 \end{pmatrix} = \begin{pmatrix} 5.10 \\ 2.83 \end{pmatrix}$$

3. Donnez cette transformation en coordonnées homogènes

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x-x_t \\ y-y_t \\ 1 \end{pmatrix}$$

4. Qu'en est-il de changement de référentiel inverse ?

$$\begin{pmatrix} \cos\theta & +\sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} x-x_t \\ y-y_t \\ 1 \end{pmatrix}$$

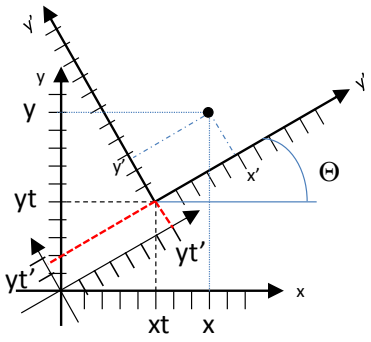
$$\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ +\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} + \begin{pmatrix} -x_t \\ -y_t \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ +\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} + \begin{pmatrix} x_t \\ y_t \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} \cos 30 & -\sin 30 & 0 \\ \sin 30 & \cos 30 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 5.10 \\ 2.83 \\ 1 \end{pmatrix} + \begin{pmatrix} 5 \\ 5 \\ 0 \end{pmatrix} = \begin{pmatrix} 8 \\ 10 \\ 1 \end{pmatrix}$$

B. Une rotation de -30deg puis une translation de (-xt', -yt')

1. Trouver la matrice de rotation et le vecteur de translation qui permettent de changer de référentiel.



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} x_t' \\ y_t' \end{pmatrix} \text{ avec } \begin{pmatrix} x_t' \\ y_t' \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_t \\ y_t \end{pmatrix}$$

2. Quels sont les coordonnées de A point dans le référentiel Ox'y'

$$\begin{pmatrix} x_t' \\ y_t' \end{pmatrix} = \begin{pmatrix} \cos 30 & \sin 30 \\ -\sin 30 & \cos 30 \end{pmatrix} \begin{pmatrix} 5 \\ 5 \end{pmatrix} = \begin{pmatrix} 6.83 \\ 1.83 \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos 30 & \sin 30 \\ -\sin 30 & \cos 30 \end{pmatrix} \begin{pmatrix} 8 \\ 10 \end{pmatrix} - \begin{pmatrix} 6.83 \\ 1.83 \end{pmatrix} = \begin{pmatrix} 11.93 \\ 4.66 \end{pmatrix} - \begin{pmatrix} 6.83 \\ 1.83 \end{pmatrix} = \begin{pmatrix} 5.10 \\ 2.83 \end{pmatrix}$$

3. Donnez cette transformation en coordonnées homogènes

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta & -x_t' \\ -\sin\theta & \cos\theta & -y_t' \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

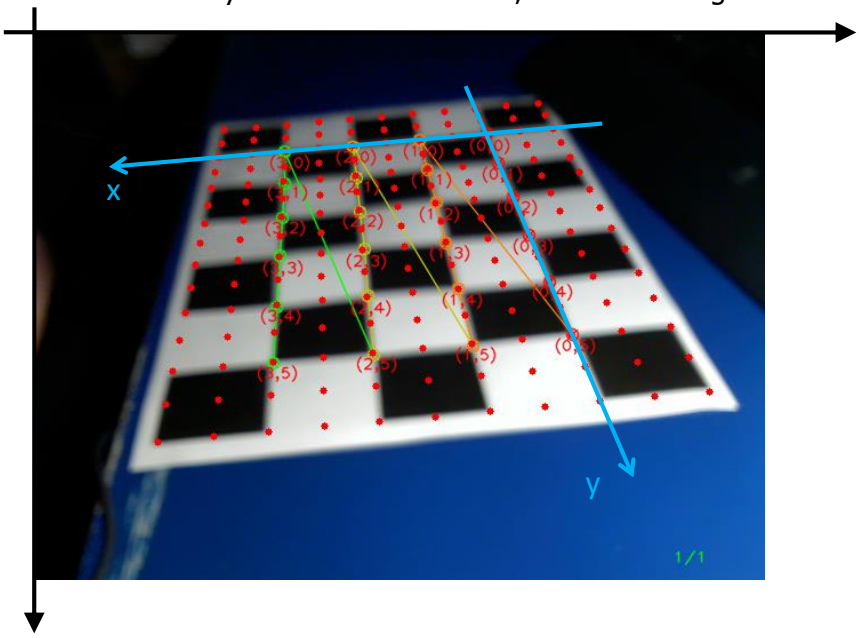
Exercice 3

Etudiez, critiquez et utilisez le code fourni en exemple dans OpenCV pour calibrer votre caméra. Déterminez les paramètres intrinsèques et extrinsèques.
Modifiez ce code afin de l'améliorer et de :

- Vérifier que la matrice homographique donne les résultats voulus en calculant les coordonnées à l'écran en fonction des coordonnées du chessboard.
- Calculez la focale en mm
- Déterminez la correction radiale apportée au pixel 100,100
- Déterminez la correction tangentielle apportée au même pixel.

Matrice d'homographie

La figure suivante montre deux systèmes de références, l'un lié à l'image et l'autre lié au damier.



La matrice d'homographie permet de passer des coordonnées exprimées dans un repère à l'autre. Cette matrice est déterminée en connaissant les coordonnées des points dans le repère échiquier {(0,0),…(3,5)} et leurs correspondants dans le repère image. Avec l'image ci-dessus, OpenCV donne la matrice suivante.

$$H = \begin{bmatrix} -58.801 & -15.917 & 395.909 \\ 4.931 & 17.691 & 90.226 \\ -0.001 & 0.066 & 1.000 \end{bmatrix}$$

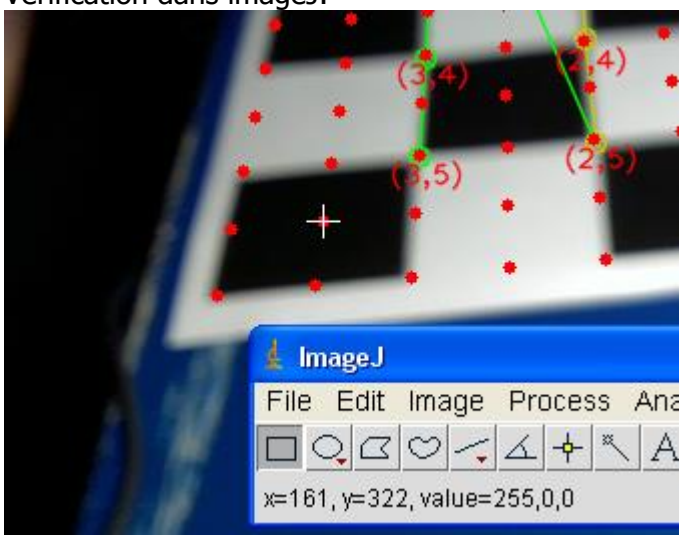
Par exemple, prenons le point (3.5, 5.5) en coordonnée échiquier et déterminons le point en coordonnées images:

$$\begin{bmatrix} x \\ y \\ s \end{bmatrix}_{image} = H \cdot \begin{bmatrix} x \\ y \\ s \end{bmatrix}_{échiquier} = \begin{bmatrix} -58.801 & -15.917 & 395.909 \\ 4.931 & 17.691 & 90.226 \\ -0.001 & 0.066 & 1.000 \end{bmatrix} \cdot \begin{bmatrix} 3.5 \\ 5.5 \\ 1 \end{bmatrix} = \begin{bmatrix} 102.560 \\ 204.784 \\ 0.635 \end{bmatrix}$$

Pour revenir aux coordonnées homogènes il faut diviser par s (0.635). Et on obtient :

$$\begin{bmatrix} x/s \\ y/s \\ s/s \end{bmatrix}_{image} = \begin{bmatrix} 161.525 \\ 322.521 \\ 1 \end{bmatrix}_{image}$$

Vérification dans imageJ:



On obtient bien les coordonnées 161, 322 dans le repère lié à l'image

Code OpenCV:

Acquisition de points « images » correspondant au coordonnées « échiquier »

```
void acquisition(int image_count, CvSize board_size, IplImage *view,
            CvCapture* capture,
                  CvPoint2D32f* image_points_buf,
            CvSeq* image_points_seq,
            clock_t* prev_timestamp)
{
      IplImage *view_gray=0;
CvSize    img_size;
      int       found;
      int       blink = 0;
      int       count =0;

      img_size = cvGetSize(view);
found = cvFindChessboardCorners( view, board_size, image_points_buf,
                              &count, CV_CALIB_CB_ADAPTIVE_THRESH );
      view_gray = cvCreateImage( cvGetSize(view), 8, 1 );
cvCvtColor( view, view_gray, CV_BGR2GRAY );
cvFindCornerSubPix( view_gray, image_points_buf, count,
            cvSize(11,11), cvSize(-1,-1),
            cvTermCriteria(CV_TERMCRIT_EPS+CV_TERMCRIT_ITER, 30, 0.1)
            );
      cvReleaseImage( &view_gray );

      int delay = 1000;
      if( found && (clock() - *prev_timestamp > delay*1e-3*CLOCKS_PER_SEC) )
{
            cvSeqPush( image_points_seq, image_points_buf );
            *prev_timestamp = clock();
            blink = 1;
      }else{
            blink = 0;
      }

      cvDrawChessboardCorners( view, board_size,
                              image_points_buf, count, found );

      // Prepare text to draw on image
      CvFont  font        = cvFont( 1, 1 );
      CvPoint text_origin;
      int     base_line   = 0;
      CvSize  text_size   = {0,0};
      char    str[100];

      cvGetTextSize( "100/100", &font, &text_size, &base_line );
      text_origin.x = view->width  - text_size.width - 10;
      text_origin.y = view->height - base_line       - 10;

      if( image_count > 0 )
            sprintf( str, "%d/%d", image_points_seq ? image_points_seq->total : 0,
image_count );
      else
            sprintf( str, "%d/?", image_points_seq ? image_points_seq->total : 0 );
      cvPutText( view, str, text_origin, &font, CV_RGB(0,255,0));
      if( blink )cvNot( view, view );

}
```

```
void findHomography( CvSeq* image_points_seq,
                     CvSize board_size,
                     float square_size, CvMat* matHomography)
{

    int point_count = board_size.width*board_size.height;

    CvMat* image_points  = cvCreateMat( 1, point_count, CV_32FC2 );
    CvMat* object_points = cvCreateMat( 1, point_count, CV_32FC2 );


     int  j, k;

    CvSeqReader reader;
    cvStartReadSeq( image_points_seq, &reader );

    // initialize arrays of points
    CvPoint2D32f* src_img_pt =  (CvPoint2D32f*)reader.ptr;
    CvPoint2D32f* dst_img_pt = ((CvPoint2D32f*)image_points->data.fl);
    CvPoint2D32f* obj_pt     = ((CvPoint2D32f*)object_points->data.fl);

    for( j = 0; j < board_size.height; j++ )
       for( k = 0; k < board_size.width; k++ )
       {
           *obj_pt++     = cvPoint2D32f(j*square_size, k*square_size);
           *dst_img_pt++ = *src_img_pt++;
       }
    CV_NEXT_SEQ_ELEM( image_points_seq->elem_size, reader );

    cvFindHomography(object_points, image_points, matHomography,CV_RANSAC, 5 );

    cvReleaseMat( &object_points );
    cvReleaseMat( &image_points );

}
```

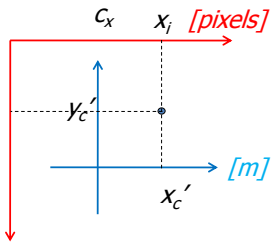
```
void useHomography( CvSeq* image_points_seq,
                    CvSize board_size,
                    float square_size, IplImage *view, CvMat* matHomography)

{
      CvMat*  ptScreen         = cvCreateMat( 3, 1, CV_32FC1 );
      float*  ptScreen_data    = ptScreen->data.fl;

      CvMat*  ptChessBord      = cvCreateMat( 3, 1, CV_32FC1 );
      float*  ptChessBord_data = ptChessBord->data.fl;

      float* matHomography_data = matHomography->data.fl;


      printf("Matrix|%9.3f %9.3f %9.3f|\n       |%9.3f %9.3f %9.3f|\n
      |%9.3f %9.3f %9.3f|\n",
            matHomography_data[0], matHomography_data[1], matHomography_data[2],
            matHomography_data[3], matHomography_data[4], matHomography_data[5],
            matHomography_data[6], matHomography_data[7],
      matHomography_data[8]);

      for( float m = -1.; m <= board_size.height; m=m+0.5 )
      {
            for( float n = -1.; n <= board_size.width; n=n+0.5 )
            {
                  ptChessBord_data[0] = m*square_size;
                  ptChessBord_data[1] = n*square_size;
                  ptChessBord_data[2] = 1.0;

                  cvMatMul( matHomography, ptChessBord, ptScreen);

                  printf("M[%4.1f %4.1f %4.1f]t = [%7.3f %7.3f %7.3f]   =>
                        [%7.3f %7.3f %7.3f]\n)",
                  ptChessBord_data[0], ptChessBord_data[1], ptChessBord_data[2],
                  ptScreen_data[0],    ptScreen_data[1],    ptScreen_data[2],
                  ptScreen_data[0]/ptScreen_data[2],
                  ptScreen_data[1]/ptScreen_data[2],
                  ptScreen_data[2]/ptScreen_data[2]);

            }
      }

      cvReleaseMat( &point_counts );
      cvReleaseMat( &ptScreen );
      cvReleaseMat( &ptChessBord );
}
```

Correction de la distorsion

Matrice intrinsèque:

$$T_{Intrinsèque} = \begin{pmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{pmatrix} \text{ avec } \begin{cases} (C_x, C_y) \text{ le centre principal de l'image} \\ f_x \text{ et } f_y \text{ les focales exprimées en pixels} \end{cases}$$

Cette matrice permet de passe du référentiel lié à la caméra au référentiel lié à l'image



$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{pmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{bmatrix} x_c' \\ y_c' \\ 1 \end{bmatrix}$$

Correction de la distorsion

Avant d'applique la correction de distorsion tangentielle et radiales, il faut passer aux coordonnées liées à la caméra.

$$\begin{aligned} x_i &= x_c' \cdot f_x + C_x \\ y_i &= y_c' \cdot f_y + C_y \end{aligned} \Rightarrow \begin{aligned} x_c' &= (x_i - C_x)/f_x \\ y_c' &= (y_i - C_y)/f_y \end{aligned}$$

La correction de la distorsion est composée des corrections radiale et tangentielle décrites ci-dessous :

$$\begin{pmatrix} x_{corrigé} \\ y_{corrigé} \end{pmatrix} = \underbrace{(1 + k_1 r^2 + k_2 r^4)}_{\text{correction radiale}} \cdot \begin{pmatrix} x_c' \\ y_c' \end{pmatrix} + \underbrace{\begin{pmatrix} 2 \cdot p_1 \cdot x_c' \cdot y_c' + p_2 (r^2 + 2x_c'^2) \\ 2 \cdot p_2 \cdot x_c' \cdot y_c' + p_1 (r^2 + 2y_c'^2) \end{pmatrix}}_{\text{correction tangentielle}} \text{ avec } r^2 = x_c'^2 + y_c'^2$$

Puis il faut revenir aux coordonnées écran :

$$\begin{aligned} x_i'' &= x_{corrigé} \cdot f_x + C_x \\ y_i'' &= y_{corrigé} \cdot f_y + C_y \end{aligned}$$

Application

Pour 10 images, j'ai obtenu de OpenCV les informations suivantes :

$$T_{Intrinsèque} = \begin{pmatrix} 651.65 & 0 & 313.25 \\ 0 & 654.94 & 264.50 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{et} \quad \begin{aligned} k_1 &= 0.191 \\ k_2 &= -0.264 \\ p_1 &= 0.0121 \\ p_2 &= -0.0134 \end{aligned}$$

Pour le pixel (100,100) on trouve :

xi	**100**	**pixels**
yi	**100**	**pixels**
x'c	-0.32724933	mm
y'c	-0.25117707	mm
r2	*0.17018204*	*mm*
r4	*0.02896193*	*mm*
k1r2	*0.03257571*	*mm*
k2r4	*-0.00765361*	*mm*
1+k1r2+k2r4	*1.02492209*	*mm*
xrad= x'c*(1+…)	-0.33540507	mm
yrad= y'c*(1+…)	-0.24506942	mm
xtang	-0.00316057	mm
ytang	0.00139323	mm
x'c (corrigé)	-0.33856564	mm
y'c (corrigé)	-0.24646265	mm
xi_corr	**92.6256994**	**pixels**
yi_corr	**103.087649**	**pixels**
correction_x	**7.374**	**pixels**
correction_y	**-3.088**	**pixels**

Dans ce cas, le pixel (100,100) est déplacé aux coordonnées (93,103)

Calcul de la focale

La matrice intrinsèque indique la focale en pixels, dans le cas ci-dessus elle vaut environ 650 pixels.
Il faut connaitre la taille des pixels en mm pour avoir la distance focale en mm. Les paramètres de la caméra utilisée (Logitech C300, 1.3MP) sont introuvables. Mais, on estime pour une caméra de 2MP un ordre de grandeur est de 2.8 micron/pixels. D'autre part, la caméra Logitech Quickcam Pro 9000 de MP, on trouve une distance focale de 3.7 mm.

La focale estimée pour la caméra C300 vaut : 650 x 2.8um = 1820 um = 1.8 mm. L'ordre de grandeur est correct. De plus la taille des pixels d'une caméra 1.3 MP est plus grande que ceux d'une caméra 2MP. Ainsi la focale est certainement plus grande que 1.8 mm