

In[1]:=

IFT 6145 – Vision tridimensionnelle

IFT 6145 – Vision tridimensionnelle

TP2 - Énoncé

Étudiant: Zhibin Lu

■ Partie 7 : Calibrage d'une vraie caméra (5 points)

Utilisez le code de calibration planaire que vous avez réalisé à la partie 5 et calibrer une vraie caméra de votre choix, et affichez la caméra en 3D avec un objet connu pour montrer que c'est bien la bonne calibration. Assurez-vous qu'il n'y a pas trop de distorsion radiale, ça ira mieux.

Je vais calibrer le caméra de mon iPhone7.

some general functions:

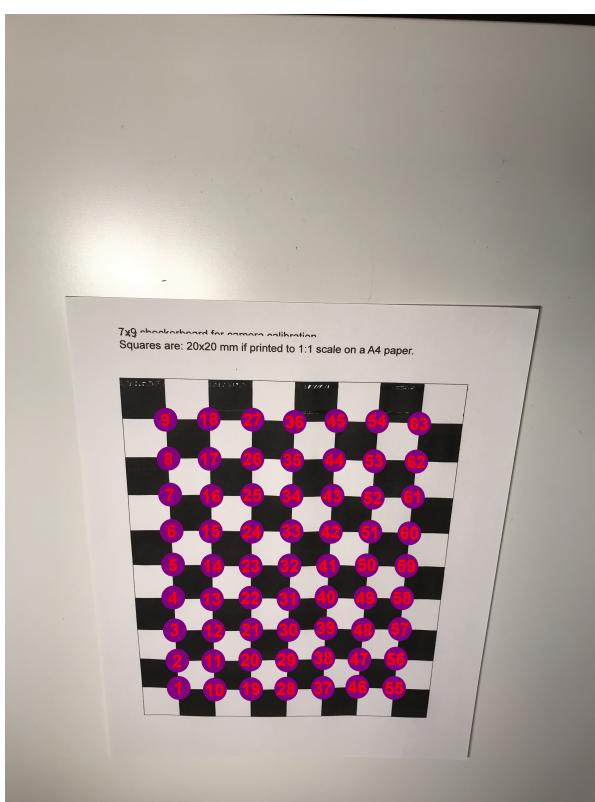
In[2]:=

```
e2p[v_] := Append[v, 1];
           ↳ appose
p2e[v_] := v[[1 ;; -2]]/v[[-1]];
cornersHomogene[im_, homo_] := Module[{w, h, c}, (
           ↳ module
```

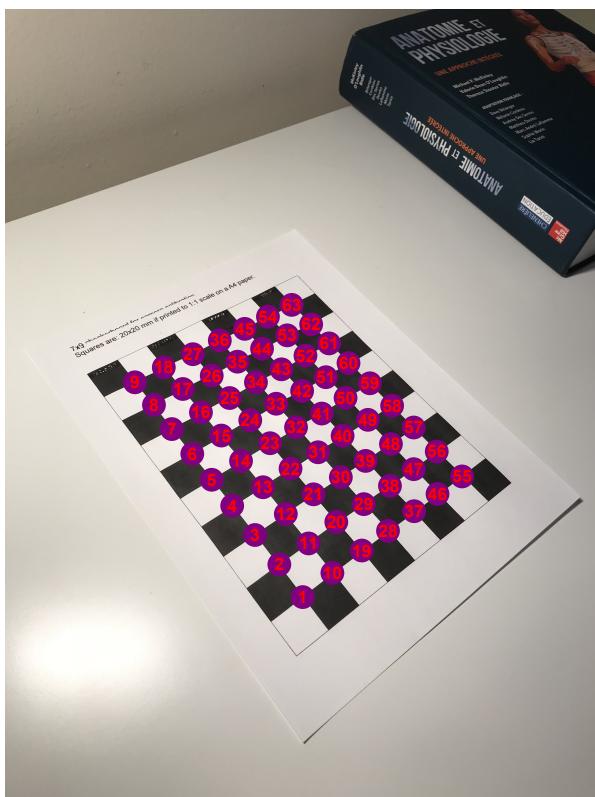
Out[•]=



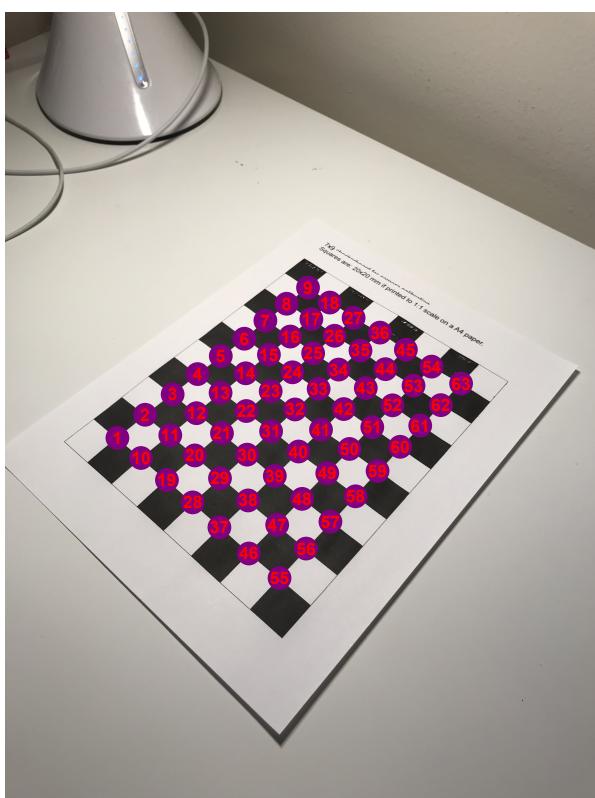
Out[•]=



Out[•]=



Out[•]=

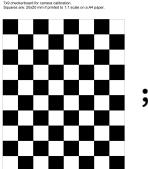


Obtenir les points du damier original. Afficher les points originaux:

```
In[8]:= {w, h} = ImageDimensions[img3];
          [dimensions d'image]

Print["Size of image: ", w, " , ", h];
          [imprime]

A4h = 279; (*the height mm paper *)
(*A4w=215.4; the width mm paper*)
A4w = A4h * w / h; (*the height mm of paper correspondant the scale of camera*)
```



```
img00 = ;
```

```
{A4pixW, A4pixH} = ImageDimensions[img00];
          [dimensions d'image]

img0 = ImageResize[ImageCrop[img00, {A4pixW, h * A4pixW / w}], {w, h}];
          [redimensionn... [recadre image]

(*la taille entre les cases real sur ma feuille imprime!*)
(*167/8,209/10*)
xStep = 167 / 8;
yStep = 209 / 10;
pc = Table[{xStep * i, yStep * j}, {i, -3, 3}, {j, -4, 4}];
          [table]

ListPlot[pc]
          [tracé de liste]

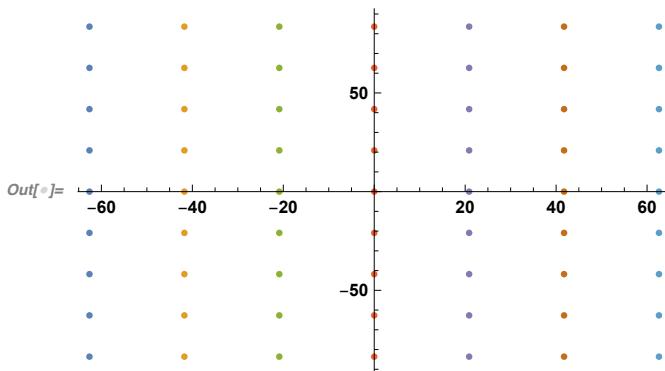
(*pc//N//MatrixForm*)
          [v... [apparence matricielle]

pc = Flatten[pc, {{1, 2}, {3}}] // N;
          [aplatis]          [valeur numérique]

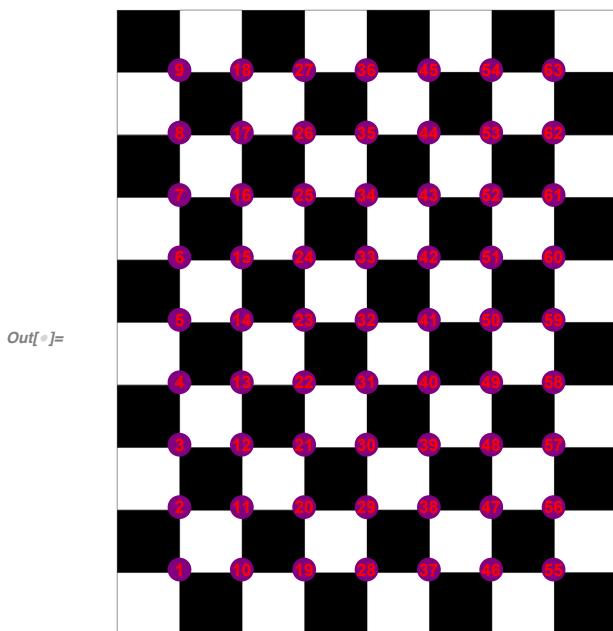
(*{Min[pc[[All,1]],Min[pc[[All,2]]]}*)
          [minimum tout]  [minimum tout]

pi0 = 15.2 * (# + {96.9, 139.6}) & /@ pc;
showImageAndPoints[img0, pi0]

Size of image: 3024 , 4032
```



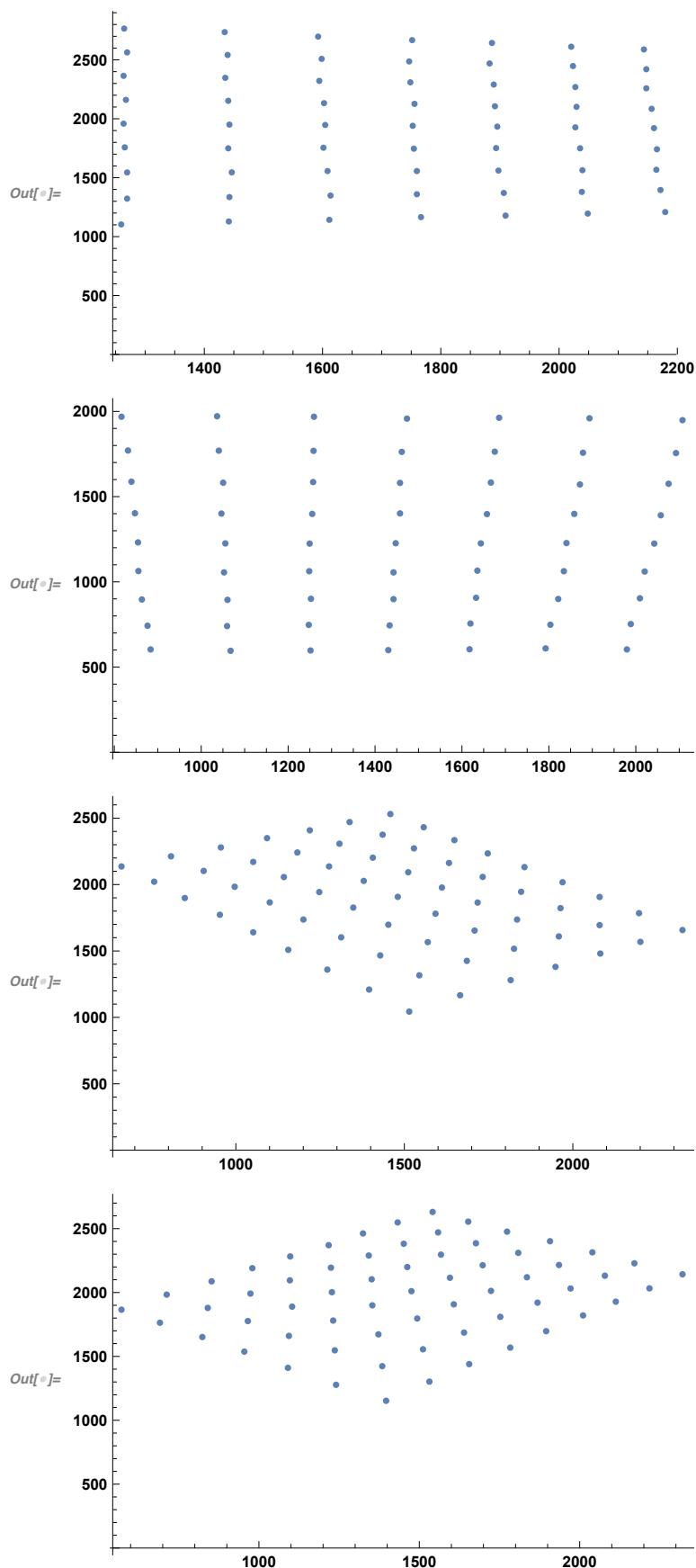
1x100mm grid for camera calibration.
Squares are: 20x20 mm if printed to 1:1 scale on a A4 paper.



Afficher les point qu'on a choisi:

```
In[•]:= PIs = {pi3, pi4, pi5, pi6};
(*all the pc are the same, but the order are different*)
ListPlot[PIs[[1]]]


```



Trouver les homographies H:

```
In[]:= Hs = Table[FindGeometricTransform[PIs[[i]], pc][[2]], {i, 1, Length[PIs]}];
          |table |trouve transformée géométrique |longueur
(*Hs=FindGeometricTransform[pc,PIs][[All,2]];*)
          |trouve transformée géométrique |tout
HsMatrix = TransformationMatrix /@ Hs;
          |matrice de transformation
Print["All homographies Matrix:"];
          |imprime |tout
MatrixForm /@ HsMatrix
          |apparence matricielle
```

All homographies Matrix:

$$\text{Out}[]= \left\{ \begin{pmatrix} 8.97877 & 0.182713 & 1754.42 \\ 1.72773 & 9.26458 & 1932.49 \\ 0.00111775 & 0.000172171 & 1. \end{pmatrix}, \begin{pmatrix} 9.54304 & -1.19083 & 1447.92 \\ 0.123412 & 6.8527 & 1229.79 \\ 0.000096609 & -0.000974403 & 1. \end{pmatrix}, \begin{pmatrix} 8.043 & -3.68114 & 1484.8 \\ 5.89962 & 7.57597 & 1899.93 \\ 0.00106983 & 0.000964207 & 1. \end{pmatrix}, \begin{pmatrix} 4.91125 & 7.5661 & 1477.66 \\ -6.59262 & 7.76 & 2005.7 \\ -0.000944217 & 0.00136174 & 1. \end{pmatrix} \right\}$$

- Utiliser l'image de la conique absolue ω pour récupérer les paramètres internes (matrice K), à l'aide de la décomposition de Cholesky.

$$K = \begin{pmatrix} 3995.77 & 0. & 1681.83 \\ 0. & 3995.77 & 2317.51 \\ 0. & 0. & 1. \end{pmatrix}$$

- Récupérer, pour chaque image, la rotation R et translation T.

```
In[]:= decomposeRC[K_, H_] := Module[{N, rho, r1, r2, r3, R, c},
  N = Inverse[K].H;
  rho = 1/Norm[N[[All, 1]]];
  r1 = rho * N[[All, 1]];
  r2 = rho * N[[All, 2]];
  r3 = Cross[r1, r2];
  R = Transpose[{r1, r2, r3}];
  c = -rho * Transpose[R].N[[All, 3]];
  {R, c}
]

allRC = decomposeRC[K, #] & /@ HsMatrix;
Print["All Rotation R:"];
Imprime[tout]
allR = proj2Dvers3D[ TransformationFunction[#] & /@ allRC[[All, 1]]]
Print["All Translate T:"];
Imprime[tout]
allT = TranslationTransform[#] & /@ allRC[[All, 2]]
All Rotation R:
Out[]= {TransformationFunction[{{0.841972, -0.012673, -0.565366, 0}, {-0.10232, 1.05151, -0.0754145, 0}, {0.52973, 0.0815956, 0.884047, 0}, {0, 0, 0, 1.}}], TransformationFunction[{{0.999097, 0.0477101, -0.0354588, 0}, {-0.0107019, 0.970376, 0.416272, 0}, {0.0411147, -0.414685, 0.97001, 0}, {0, 0, 0, 1.}}], TransformationFunction[{{0.751894, -0.638581, -0.140032, 0}, {0.411883, 0.643234, -0.677586, 0}, {0.51479, 0.463963, 0.746664, 0}, {0, 0, 0, 1.}}], TransformationFunction[{{0.746138, 0.605688, -0.0869117, 0}, {-0.505638, 0.528572, -0.728436, 0}, {-0.433139, 0.624669, 0.700646, 0}, {0, 0, 0, 1.}}]}
```

All Translate T:

$$\text{Out}[=] = \{\text{TransformationFunction}\left[\left(\begin{array}{ccc|c} 1. & 0 & 0 & -262.972 \\ 0 & 1. & 0 & 9.4576 \\ 0 & 0 & 1. & -417.547 \\ \hline 0 & 0 & 0 & 1. \end{array}\right)\right],$$

$$\text{TransformationFunction}\left[\left(\begin{array}{ccc|c} 1. & 0 & 0 & 6.15362 \\ 0 & 1. & 0 & 290.088 \\ 0 & 0 & 1. & -365.473 \\ \hline 0 & 0 & 0 & 1. \end{array}\right)\right],$$

$$\text{TransformationFunction}\left[\left(\begin{array}{ccc|c} 1. & 0 & 0 & -209.158 \\ 0 & 1. & 0 & -206.058 \\ 0 & 0 & 1. & -396.681 \\ \hline 0 & 0 & 0 & 1. \end{array}\right)\right],$$

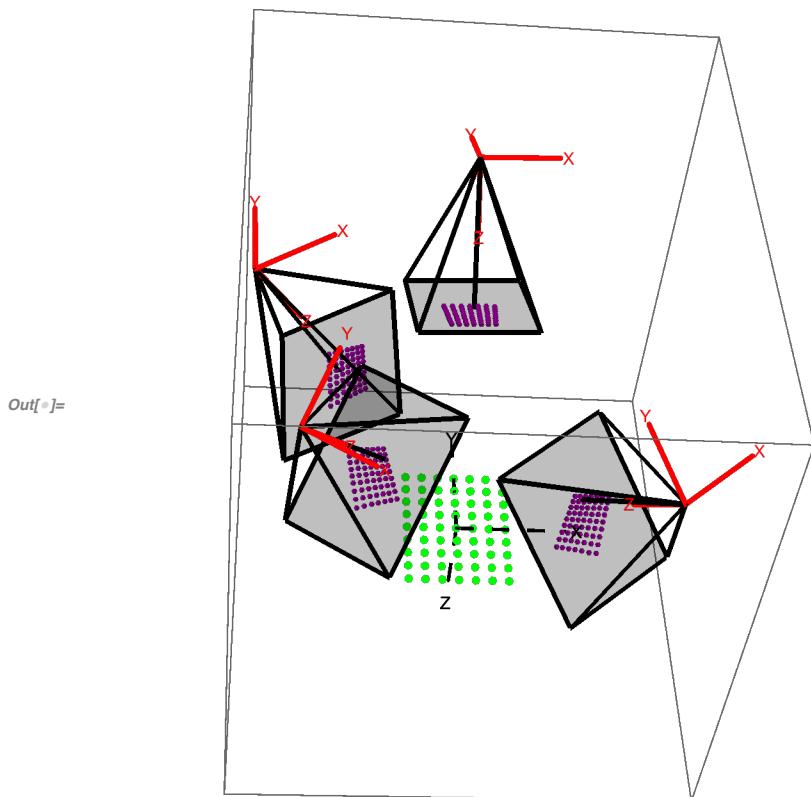
$$\text{TransformationFunction}\left[\left(\begin{array}{ccc|c} 1. & 0 & 0 & 198.082 \\ 0 & 1. & 0 & -253.435 \\ 0 & 0 & 1. & -349.519 \\ \hline 0 & 0 & 0 & 1. \end{array}\right)\right]\}$$

- Illustrer graphiquement le plan de calibrage dans le monde et les 7 caméras ensembles. Seul le plan (avec des points dessus) et les caméras sans les points images sont requis.

```

dist = -200;
allMext = MapThread[externe, {allR, allT}];
    [applique en enfilade
mint = TransformationFunction[K];
    [fonction de transformation
(*mint=proj2Dvers3D[interne[w/2,h/2,f]] ;*)
allM = proj2Dvers3D[mint].# & /@ allMext;
allIM = InverseFunction /@ allM;
    [fonction inverse
pPlanCameraAuMonde =
  Table[allIM[[k]] /@ (e2p[#] * dist & /@ PIs[[k]]), {k, 1, Length[allIM]}];
    [table [longueur
(*drawPlan[m_,mext_]:=geomCamera[w,h,dist,m]*)
drawPlan[m_, mext_] := geomCameraWithAxes[w, h, dist, m, mext]
Show[
[montre
  Graphics3D[{[graphique 3d
    PointSize[0.01], Purple,
      [taille des points [violet
    Point /@ pPlanCameraAuMonde,
      [point
    (*Table[Point[{-6.3+x*13,-6.3+y*13,0}],{y,0,5},{x,0,8}];*)
      [table [point
    (*all the pc are the same, but the order are different*)
    PointSize[0.015], Green,
      [taille des points [vert
    Point[Append[#, 0] & /@ pc]
      [point [appose
  }],
  MapThread[drawPlan, {allM, allMext}]
[applique en enfilade
]

```



1. Après le calibrage, les paramètres du camera de iphone7 sont obtenus: $f = 3995.77$, $w = 3,363.66$, $h = 4,635.02$. Étant donné qu'il y a certain écart de position pour le centre du capteur du petit caméra au moment de la fabrication, le résultat du calibration est un petit peu différent de la taille originale.

2. Selon le calcul dessous, j'ai obtenu l'angle horizontal de la caméra et l'angle vertical.

$$180 \times 2 \times \arctan(1681.83 / 3995.77) / 3.14159 = 45.65 \text{ degrés}$$

$$180 \times 2 \times \arctan(2317.51 / 3995.77) / 3.14159 = 60.23 \text{ degrés}$$

3. Ces quatre positions affiché dessus correspondent bien aux positions lorsque j'ai pris ces photos.