NSR18 – Rapport de mini-projet : Réalité augmentée.

Alexandre Houet
Xavier Lagorce

Novembre 2010

Table des matières

1 Présentation du projet

Le but du projet est de réaliser une application de vision par ordinateur. On va donc réaliser un programme de réalité augmentée : ce programme devra être capable de rajouter des objets virtuels dans une scène réelle photographiée.

Les données fournies pour parvenir à la résolution du problème seront :

- Une photographie d'une mire 3D, pour assurer l'étalonnage de la caméra utilisée pour prendre les images.
- Une photograpie de la scène dans laquelle rajouter des objets virtuels.

Le programme terminé va rajouter une vue 3D d'une maison (en mode « fil de fer ») sur un plan qui sera contenu dans l'image. Ce plan sera identifié par un quadrillage qui permettra son identification.

2 Approche mathématique

2.1 Principe

Pour pouvoir ajouter un objet virtuel dans notre scène photographiée, nous avons besoin d'identifier la matrice de projection associée à la caméra dans la scène en question.

Pour identifier cette matrice de projection, il faudrait connaître un certain nombres de points dans la scène. Mais pour que cette identification soit correcte, ces points doivent être placés dans toutes les directions de l'espace, or nous ne connaissons que des points contenus dans un plan. Utiliser ces points pour déterminer la matrice de projection n'est donc pas possible car nous ne possédons pas de données sur le troisième axe de notre repère.

Pour résoudre ce problème, nous allons donc utiliser une particularité du problème qui nous est posé. En effet, l'opération qui transforme les points du plan vers l'image est aussi une homographie. L'intérêt réside dans le fait que cette homographie n'est pas indépendante de la projection que nous recherchons. Elle correspond en fait à une restriction de la projection de la caméra aux points possédant une coordonnée $Z = 0$.

Voyons en détail comment nous allons procéder.

2.2 Identification de la caméra

Pour commencer, nous avons besoin de connaître les propriétés de la caméra utilisée. Ces caractéristiques sont regroupées dans la matrice de calibrage de la caméra $\mathbf{K}$:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Cette matrice contient les focales de la caméra selon les 2 directions de l'image ainsi que le décalage de son centre optique par rapport au système de coordonnées de l'image.

Lorsque l'on place notre caméra dans notre scène, on effectue une translation ainsi qu'une rotation dans l'espace de cette dernière. La matrice de projection **P** associée à notre scène va donc devenir :

$$\mathbf{P} = \mathbf{K}\,[\mathbf{R}|\mathbf{t}] \tag{2}$$

Où **R** est une matrice de rotation dans l'espace et **t** correspond à la translation de la caméra. Identifier **P** nous permet donc de remonter à **K**.

2.2.1 Identification de la matrice de projection

Pour identifier la matrice de projection **P**, nous avons besoin d'un ensemble de points placés dans l'espace de manière à décrire complètement le repère représentatif de la scène. Nous allons donc photographier une mire 3D. Celle-ci est constituée de 3 plans orthogonaux contenant chacun 16 points.

On obtient donc un ensemble d'équations de la forme :

$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \equiv \mathbf{P} \begin{pmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{pmatrix} \tag{3}$$

Ces données vont alors nous permettre d'identifier la matrice **P** suivante :

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \tag{4}$$

qui nous donne le système suivant :

$$\mathbf{A}\mathbf{p} = 0 \tag{5}$$

où **p** est un vecteur colonne regroupant les p_{ij} et où les lignes de A sont de la forme :

$$\begin{bmatrix} A_{3i} \\ A_{3i+1} \\ A_{3i+2} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & -X_i & -Y_i & -Z_i & -1 & y_iX_i & y_iY_i & y_iZ_i & y_i \\ X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -x_iX_i & -x_iY_i & -x_iZ_i & -x_i \\ -y_iX_i & -y_iY_i & -y_iZ_i & -y_i & x_iX_i & x_iY_i & x_iZ_i & x_i & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6}$$

Pour résoudre ce système, il nous faut au moins 4 points, l'utilisation de plus de points nous permet d'améliorer la précision du résultat et de potentiellement réduire l'influence du bruit sur l'estimation de la position des points d'intérêts dans l'image.

2.2.2 Identification de la matrice de calibrage

La section précédente nous a permis d'obtenir la matrice de projection **P** associée à la caméra. D'après l'équation 2, on peut alors extraire de **P** la matrice **K** qui nous intéresse.

En effet, la matrice **P** est le produit d'une matrice triangulaire supérieure (**K**) par une matrice orthogonale ([**R**|**t**]). Si on effectue une décomposition RQ sur la matrice **P**, on va donc pouvoir en déduire **K**.

2.3 Obtention de la matrice de projection de la scène

Maintenant que l'on connaît la matrice de calibrage de la caméra, nous allons pouvoir utiliser la correspondance entre la projection **P** et l'homographie **P** qui relie les points du plan considéré dans le problème et les points correspondants sur l'image.

2.3.1 Identification de l'homographie du plan vers image

Pour cela, on va devoir par commencer identifier l'homographie **H**. On va utiliser la même technique que pour identifier **P** à la section 2.2.1 à partir des équations :

$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \equiv \mathbf{H} \begin{pmatrix} X_i \\ Y_i \\ 1 \end{pmatrix} \tag{7}$$

$$\mathbf{P} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{8}$$

$$\mathbf{Ah} = 0 \tag{9}$$

où **h** est un vecteur colonne regroupant les h_{ij} et où les lignes de A sont de la forme :

$$\begin{bmatrix} A_{3i} \\ A_{3i+1} \\ A_{3i+2} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & -X_i & -Y_i & -1 & y_iX_i & y_iY_i & y_i \\ X_i & Y_i & 1 & 0 & 0 & 0 & -x_iX_i & -x_iY_i & -x_i \\ -y_iX_i & -y_iY_i & -y_i & x_iX_i & x_iY_i & x_i & 0 & 0 & 0 \end{bmatrix} \tag{10}$$

Il est à noter que nous ne pouvons déterminer **H** qu'à une constante multiplicative λ près.

2.3.2 Déduction de la matrice de projection

Pour les points du plan qui nous intéresse ici, on sait d'une part que :

$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \equiv \mathbf{H} \begin{pmatrix} X_i \\ Y_i \\ 1 \end{pmatrix}, \tag{11}$$

et d'autre part que :

$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \equiv \mathbf{P} \begin{pmatrix} X_i \\ Y_i \\ 0 \\ 1 \end{pmatrix}. \tag{12}$$

On peut donc dire que, puisque **P** et **H** correspondent à la même transformation pour ces points :

$$\mathbf{H} \begin{pmatrix} X_i \\ Y_i \\ 1 \end{pmatrix} = \mathbf{P} \begin{pmatrix} X_i \\ Y_i \\ 0 \\ 1 \end{pmatrix}. \tag{13}$$

On obtient donc la relation suivante :

$$\mathbf{H} = \mathbf{K} \left[\begin{array}{ccc|c} \mathbf{r_1} & \mathbf{r_2} & & \mathbf{t} \end{array} \right] \tag{14}$$

Comme on connaît $\mathbf{H}$ à une constante multiplicative λ près, on obtient :

$$\lambda \left[\begin{array}{ccc|c} \mathbf{r_1} & \mathbf{r_2} & & \mathbf{t} \end{array} \right] = \mathbf{K}^{-1}\mathbf{H} \tag{15}$$

Or, on sait que $\mathbf{R}$ est une matrice de rotation, ses trois colonnes sont donc orthogonales et son déterminant est unitaire. On a donc :

$$\begin{cases} \lambda^2 \mathbf{r_3} & = & (\lambda \mathbf{r_1}) \times (\lambda \mathbf{r_2}) \\ \left| \mathbf{r_1} \quad \mathbf{r_2} \quad \mathbf{r_3} \right| & = & 1 \end{cases} \tag{16}$$

La relation sur le déterminant nous permet donc de déterminer λ puisque :

$$\left| \lambda \mathbf{r_1} \quad \lambda \mathbf{r_2} \quad \lambda^2 \mathbf{r_3} \right| = \lambda^4 \tag{17}$$

On connait donc λ et r_3, on remultipliant par $\mathbf{K}$ on peut en déduire $\mathbf{P}$.

2.4 Ajout de l'objet virtuel dans la scène

Maintenant que nous connaissons la matrice de projection $\mathbf{P}$, nous pouvons utiliser la relation suivante pour tracer notre objet virtuel dans l'image réelle :

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \equiv \mathbf{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{18}$$

3 Résolution sous Matlab

3.1 Gestion des images

3.1.1 Utilisation d'une webcam

Des recherches ont été faites pour utiliser directement une webcam dans Matlab et ainsi pouvoir réaliser l'opération de réalité augmentée dans un flux vidéo. Depuis Matlab 2010b, les webcams sont utilisables sous tous les systèmes d'exploitation[1].

Ouvrir la webcam est plutôt simple :

```
vidobj = videoinput('linuxvideo', 1, 'YUYV_320x240');
set(vidobj, 'ReturnedColorSpace', 'grayscale');
```

Il est alors intéressant de paramétrer le déclenchement manuel de la webcam pour la prise d'une image. Cela permet de garder la webcam active un certain temps avant de prendre l'image et donc de lui laisser le temps d'adapter sa luminosité à la lumière ambiante :

[1] Soit Linux, MacOS et Windows.

```
triggerconfig(vidobj, 'manual');
vidobj.FramesPerTrigger = 1;
```

On peut alors récupérer une image comme ceci :

```
start(vidobj);
trigger(vidobj);
wait(vidobj, 3);
image = getdata(vidobj, 1);
```

On peut alors créer une figure contenant un objet video associé à une fonction qui sera appelée périodiquement pour mettre à jour le contenu de la figure. On peut alors se servir de ce système pour traiter chaque image d'un flux vidéo et y ajouter des objets virtuels.

3.1.2 Chargement d'images

Si on ne veut pas utiliser de webcam, on peut aussi utiliser la fonction `imread` qui permet de charger divers types d'images et d'obtenir une matrice représentant l'image chargée.

3.2 Mesures dans les images

la détermination des points d'intérêts dans les images sera fait par l'utilisateur qui va cliquer dans l'image les points demandés par le programme[2]. Ceci rend donc le pointage quelque peu imprécis et serait préférable de la faire de manière automatique.

De plus, ce pointage par l'utilisateur nous interdit de travailler sur un flux video. Le programme se contentera donc de traiter des images statiques. La calibration de la caméra pourra donc être réalisée à chaque utilisation du programme ou bien chargée depuis un fichier pour ne réaliser que la partie réalité augmentée.

3.3 Implémentation de la méthode de résolution

Le code Matlab commenté est fourni en annexe de ce rapport. Il correspond à l'implémentation directe de la méthode décrite dans la partie précédente.

La première partie du code correspond à la calibration de la caméra et à la détermination de la matrice $\mathbf{K}$. Il est à noter que la résolution du système d'équations est effectuée grâce à une décomposition en valeurs singulières[3].

Une fois le calcul de la projection effectué, le code reprojette les points qui ont servi à déterminer ladite projection et calcule l'erreur quadratique moyenne associée pour que l'on puisse vérifier de la pertinence du résultat obtenu.

Les points reprojetés sont de plus tracés sur l'image pour un retour graphique. Est de plus ajouté le cube correspondant à la portion du repère objet identifié par les points de mesure. Ceci permet de mieux visualiser la projection obtenue.

[2] Le point à cliquer est indiqué dans le titre de la figure.
[3] Au travers de la fonction Matlab SVD.

La seconde partie du code va s'occuper de déterminer la matrice de projection dans l'image à traiter puis ensuite de rajouter des objets virtuels à la scène.

On commence par reprojeter les points utilisés dans le calcul de la projection pour déterminer l'exactitude de la matrice $\mathbf{P}$ obtenue. L'erreur quadratique moyenne est calculée et les points reprojetés sont affichés pour comparaison graphique du résultat. On trace de même le même ensemble de points décalés vers le haut et vers le bas, ce qui permet de visualiser le résultat obtenu dans $\mathbf{P}$ pour la coordonnée Z.

On termine par dessiner une maison en « fil de fer » ce qui est le but de tout le programme.

4 Résultats

4.1 Avec utilisation d'une webcam

Les résultats obtenus avec des images prises depuis une webcam ne sont pas très bons. Ceci peut venir de fait que les images obtenues ont une résolution assez faible, ce qui augmente nos erreurs de calculs.

4.2 Avec utilisation d'un appareil photo

Pour pallier au problème de la webcam, on va utiliser des photographies réalisées avec un appereil réflex donnant des images de bien meilleure qualité. Dans ce cas, on fera attention à prendre toutes les photos avec une focal fixe et un objectif qui ne déforme pas les images.

On peut voir dans la figure 1 les images utilisées pour la calibration de la caméra et pour la réalité augmentée. On peut noter que les points placés aux nœuds de la mire permettaient de mieux les visualiser sur les images récupérées au travers de la webcam.

La figure 2 montre le résultat obtenu sur la mire après calibration de la caméra. On obtient une erreur quadratique moyenne de reprojection des points de $2,4192$ pixels ce qui semble très correct.

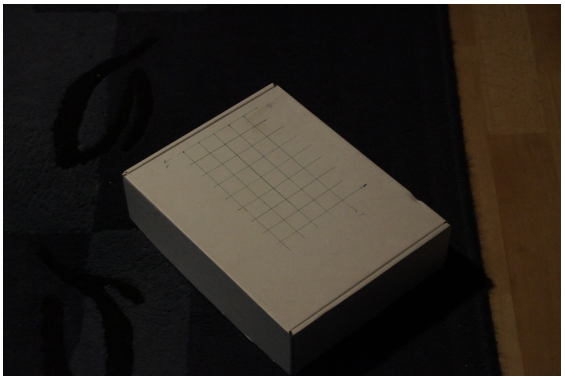
On obtient de plus la matrice $\mathbf{K}$ suivante :

$$\mathbf{K} = \begin{bmatrix} 253,9 & 1,32 & 2333 \\ 0 & 250,5 & 1417 \\ 0 & 0 & 1 \end{bmatrix} \tag{19}$$

Le programme est ensuite appliqué à 2 images sur lesquelles on va ajouter un objet virtuel. On peut voir le résultat sur les figures 3 et 4. On peut obsever sur ces images que la maison est effectivement projetée sur l'image au bon emplacement mais que sa position dans l'espace selon l'axe Z semble déformée.

Une des causes de cette déformation peut se trouver dans le bruit de mesure associé au pointage par l'utilisateur des points dans l'image et pourrait éventuellement être amélioré par l'identification automatique de ces points avec une précision qui serait alors sub-pixellaire.

(a) Mire 3D



(b) image test 1

(c) image test 2

FIG. 1 – Images utilisées pour les tests du programme

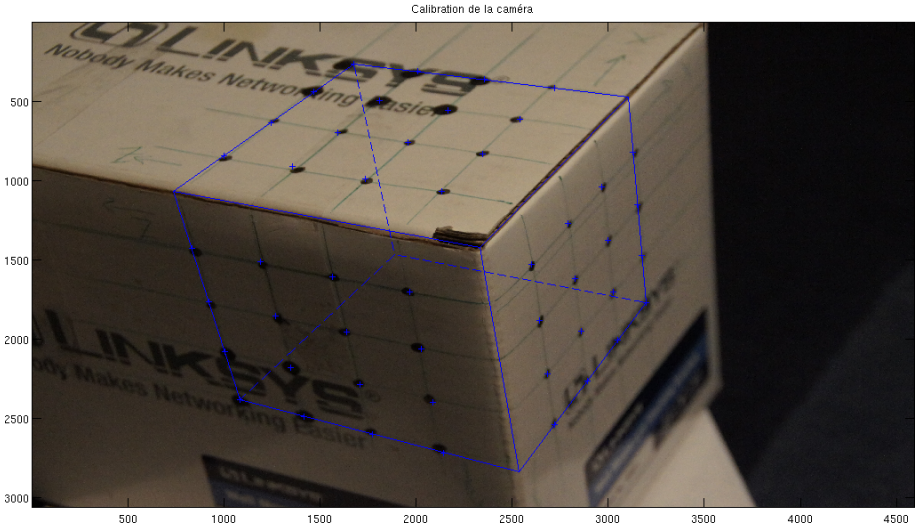


FIG. 2 – Mire 3D et reprojection après calibration de la caméra

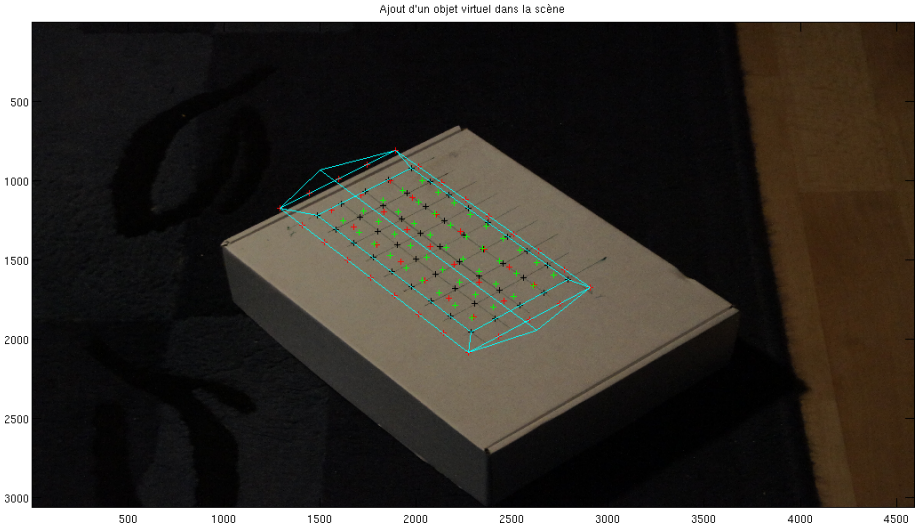


FIG. 3 – Réalité augmentée sur l'image 1 - Erreur de reprojection : $0,9031$ pixels

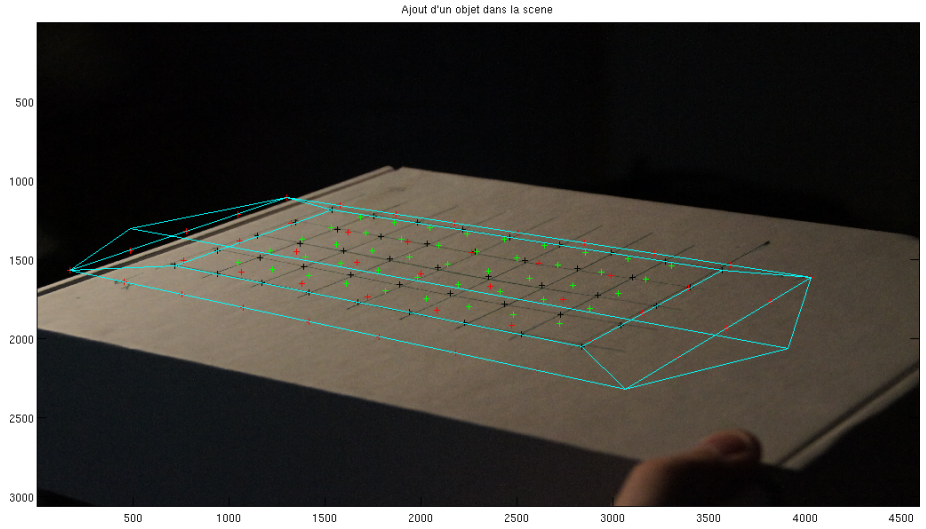


FIG. 4 – Réalité augmentée sur l'image 2 - Erreur de reprojection : $2,1477$ pixels

A Code source Matlab

A.1 Décomposition RQ

Listing 1 – rq.m

```
function [R Q]=rq(A)

[m n] = size(A);
if m>n
    error('RQ: Number of rows must be smaller than column');
end

[Q R]=qr(flipud(A).');
R=flipud(R.');
R(:,1:m)=R(:,m:-1:1);
Q=Q.';
Q(1:m,:)=Q(m:-1:1,:);

end
```

A.2 Programme complet

Listing 2 – realiteAugmentee.m

```
close all
clear all

% Cette partie permet de configurer l'utilisation d'une webcam.
% Elle est commentee, car nous utiliserons des images issues d'un
% appareil photo dans la suite
%-------------------------------------------------------------------
% Configuration du peripherique d'acquisition :
%vidobj = videoinput('linuxvideo', 1, 'YUYV_320x240');
%set(vidobj, 'ReturnedColorSpace', 'grayscale');

% Configuration du declencheur pour ne prendre qu'une image
%triggerconfig(vidobj, 'manual');
%vidobj.FramesPerTrigger = 1;
%-------------------------------------------------------------------

% Indique si la calibration doit etre effectuee ou non
calib = 0;

% Calibration de la camera si necessaire
if calib == 1,
    % Pour recuperer une image depuis la webcam
```

```
 %start(vidobj);
 %trigger(vidobj);
 %wait(vidobj, 3);
 %image = getdata(vidobj, 1);
% Pour recuperer une image webcam prealablement sauvegardee
 %load('im_calib.mat');
% Pour lire une photographie
im_calib = imread('images/mire.jpg');

% Points traces sur la mire 3D
%   plan YZ   plan XY   plan XZ
X = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   2 2 2 2 4 4 4 4 6 6 6 6 8 8 8 8   2 2 2 2 4↩
     4 4 4 6 6 6 6 8 8 8 8];
Y = [2 4 6 8 2 4 6 8 2 4 6 8 2 4 6 8   2 4 6 8 2 4 6 8 2 4 6 8 2 4 6 8   0 0 0 0 0↩
     0 0 0 0 0 0 0 0 0 0 0];
Z = [2 2 2 2 4 4 4 4 6 6 6 6 8 8 8 8   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   2 4 6 8 2↩
     4 6 8 2 4 6 8 2 4 6 8];

% On demande a l'utilisateur de cliquer sur les points de la mire
figure(1), imagesc(im_calib), colormap(gray), title('Calibration');
xp = zeros(size(X));
yp = zeros(size(Y));
for i=1:size(X,2)
    title(['Calibration point : X=' int2str(X(i)) ' Y=' int2str(Y(i)) ' Z=' ↩
        int2str(Z(i))]);
    [xi,yi]=ginput(1);
    xp(i)=xi; yp(i)=yi;
end
title('Calibration de la camera');

% On remplit la matrice representative du systeme a resoudre pour
% trouver la matrice de projection
Ap = zeros(3*size(xp,2), 12);
for i=1:size(xp,2)
    Ap(3*(i-1)+1,:) = [0 0 0 0 -X(i) -Y(i) -Z(i) -1 yp(i)*X(i) yp(i)*Y(i) yp(i)*Z(↩
        i) yp(i)];
    Ap(3*(i-1)+2,:) = [X(i) Y(i) Z(i) 1 0 0 0 0 -xp(i)*X(i) -xp(i)*Y(i) -xp(i)*Z(i↩
        ) -xp(i)];
    Ap(3*(i-1)+3,:) = [-yp(i)*X(i) -yp(i)*Y(i) -yp(i)*Z(i) -yp(i) xp(i)*X(i) xp(i)↩
        *Y(i) xp(i)*Z(i) xp(i) 0 0 0 0];
end

% On resout le systeme grace a une decomposition en valeurs singulieres
[U,S,V]=svd(Ap);
p = V(:,end);
P = zeros(3,4);
P(1,:)=p(1:4);P(2,:)=p(5:8);P(3,:)=p(9:12);

% Puis on determine K a partir de P
[K Rt] = rq(P);
% normalisation
K = 1/K(3,3)*K(1:3,1:3);
```

```
    % On calcule alors l'erreur de reprojection sur P pour verifier
    % que notre matrice est correcte
    xip=xp; yip=yp;
    for i=1:size(X,2)
        pim = P*[X(i) Y(i) Z(i) 1].';
        pim = pim/pim(3);
        xip(i) = pim(1);
        yip(i) = pim(2);
    end
    err2m_p = sqrt(sum((xip-xp).*(xip-xp) + (yip-yp).*(yip-yp)))/length(xp)
    % On dessine la mire
    hold on;
    plot(xip,yip,'b+');
    % On dessine le cube de la mire
    X = [0 8 8 0 0 0 0 0 8 8 0   8 8 0  8 8];
    Y = [0 0 8 8 0 0 8 8 0 0 0 0   0 8 8  8 8];
    Z = [0 0 0 0 0 8 8 0 0 0 8 8   8 8 8  8 0];
    xi=X; yi=Y;
    for i=1:size(X,2)
        pim = P*[X(i) Y(i) Z(i) 1].';
        pim = pim/pim(3);
        xi(i) = pim(1);
        yi(i) = pim(2);
    end
    plot(xi(1:12),yi(1:12),'b-');
    plot(xi(13:15),yi(13:15),'b--');
    plot(xi(16:17),yi(16:17),'b--');
else
    % Si on ne refait pas la calibration, il faut charger la matrice K
    % depuis un fichier
    load('camera.mat');
end

% ---------------------------------------
% Realite augmentee
% ---------------------------------------

% Prise d'une image depuis la webcam
 %start(vidobj);
 %trigger(vidobj);
 %wait(vidobj, 3);
 %image = getdata(vidobj, 1);
% Recuperation d'une image prealablement sauvegardee
 %load('image.mat');
% Recuperation d'une photographie
image = imread('images/image1.jpg');

% Points d'interet du plan sur lequel ajouter notre objet
X = 2*[0 1 2 3 4  0 1 2 3 4  0 1 2 3 4  0 1 2 3 4  0 1 2 3 4  0 1 2 3 4  0 ↩
    1 2 3 4  0 1 2 3 4];
Y = 2*[0 0 0 0 0  1 1 1 1 1  2 2 2 2 2  3 3 3 3 3  4 4 4 4 4  5 5 5 5 5  6 6 6 6 6  7 ↩
    7 7 7 7  8 8 8 8 8];
Z = zeros(1,9*5);
```

```
% On demande a l'utilisateur de cliquer sur les points du plan
figure(2), imagesc(image), colormap(gray), title('Reperage du support');
xh = zeros(size(X));
yh = zeros(size(Y));
for i=1:size(X,2)
    title(['Recuperation point : X=' int2str(X(i)) ' Y=' int2str(Y(i)) ' Z=' int2str(Z↩
        (i))]);
    [xi,yi]=ginput(1);
    xh(i)=xi; yh(i)=yi;
end
title('Ajout d''un objet dans la scene');

% On remplit la matrice representative du systeme a resoudre pour
% trouver la matrice d'homographie
Ah = zeros(3*size(xh,2), 9);
for i=1:size(xh,2)
    Ah(3*(i-1)+1,:) = [0 0 0 -X(i) -Y(i) -1 yh(i)*X(i) yh(i)*Y(i) yh(i)];
    Ah(3*(i-1)+2,:) = [X(i) Y(i) 1 0 0 0 -xh(i)*X(i) -xh(i)*Y(i) -xh(i)];
    Ah(3*(i-1)+3,:) = [-yh(i)*X(i) -yh(i)*Y(i) -yh(i) xh(i)*X(i) xh(i)*Y(i) xh(i) 0 0 ↩
        0];
end
% On resout le systeme grace a une decomposition en valeurs singulieres
[U,S,V]=svd(Ah);
h = V(:,end);
H = zeros(3,3);
H(1,:)=h(1:3);H(2,:)=h(4:6);H(3,:)=h(7:9);
H = H/H(end,end);

% Calcul de l'erreur de reprojection sur H
xih=xh; yih=yh;
for i=1:size(X,2)
    pim = H*[X(i) Y(i) 1].';
    pim = pim/pim(3);
    xih(i) = pim(1);
    yih(i) = pim(2);
end
err2m_h = sqrt(sum((xih-xh).*(xih-xh) + (yih-yh).*(yih-yh)))/length(xh)

% On determine [R|t] a partir de H
Pt = inv(K)*H;
Pk = zeros(3,4);
Pk(:,1) = Pt(:,1);
Pk(:,2) = Pt(:,2);
Pk(:,4) = Pt(:,3);
Pk(:,3) = cross(Pk(:,1),Pk(:,2));

% Normalisation de inv(K)*P
% (H connue a un facteur multiplicatif lambda pres)
lambda = nthroot(det(Pk(:,1:3)),4);
Pk(:,1:2)=Pk(:,1:2)./lambda;
Pk(:,3)=Pk(:,3)./(lambda*lambda);
Pk(:,4)=Pk(:,4)./lambda;
```

```
% Retour a P
P = K*Pk;

% Trace des valeurs pointees
Xi=X; Yi=Y; Z = 0*ones(size(X));
for i=1:size(X,2)
    pim = P*[X(i) Y(i) Z(i) 1].';
    pim = pim/pim(3);
    Xi(i) = pim(1);
    Yi(i) = pim(2);
end
hold on;
plot(Xi, Yi, 'k+');

% Trace du plan superieur
Z = 0.7*ones(size(X));
for i=1:size(X,2)
    pim = P*[X(i) Y(i) Z(i) 1].';
    pim = pim/pim(3);
    Xi(i) = pim(1);
    Yi(i) = pim(2);
end
hold on;
plot(Xi, Yi, 'r+');

% Trace du plan inferieur
Z = -0.7*ones(size(X));
for i=1:size(X,2)
    pim = P*[X(i) Y(i) Z(i) 1].';
    pim = pim/pim(3);
    Xi(i) = pim(1);
    Yi(i) = pim(2);
end
hold on;
plot(Xi, Yi, 'g+');


% Trace d'un objet 3D :
X = [0 8 8 0 0  0  8  8  0  0  0   4  8 8   4 0  0   4   4 8 8  8];
Y = [0 0 0 0 0 16 16 16 16 16 16  16 16 0   0 0 16  16   0 0 0 16];
Z = 0.7*[0 0 1 1 0  0  0  1  1  0  1 1.5  1 1 1.5 1  1 1.5 1.5 1 0  0];
Xi = zeros(size(X));
Yi = zeros(size(Y));
for i=1:size(X,2)
    pim = P*[X(i) Y(i) Z(i) 1].';
    pim = pim/pim(3);
    Xi(i) = pim(1);
    Yi(i) = pim(2);
end
hold on;
plot(Xi, Yi, 'c');
```