ROB313 - PERCEPTION POUR LES SYSTÈMES AUTONOMES

TP n°1 Estimation d'homographies et Reconstruction de Panoramas

13 décembre 2019

Gabriel Henrique Riqueti

Victor Kenichi Nascimento Kobayashi

ENSTA IP Paris

Question 1 - Rectication d'Images

Tout d'abord, nous avons mis en oeuvre sur matLab la fonction `homography2D`représentée par la Figure 1.

```
function H = homography2d(varargin)
    [x1, x2] = checkargs(varargin(:));
    % Attempt to normalise each set of points so that the origin
    % is at centroid and mean distance from origin is sqrt(2).
    [x1, T1] = normalise2dpts(x1);
    [x2, T2] = normalise2dpts(x2);

    n = size(x1, 2);
    A = zeros(2*n,9)

    for pt = 1:n
        A(2*(pt-1)+1,4:6) = -x2(3,pt)*x1(:,pt)';
        A(2*(pt-1)+1,7:9) = +x2(2,pt)*x1(:,pt)';
        A(2*(pt-1)+2,1:3) = +x2(3,pt)*x1(:,pt)';
        A(2*(pt-1)+2,7:9) = -x2(1,pt)*x1(:,pt)';
    end
    [U, S, V] = svd(A);

    h = V(:, 9);
    H = eye(3);
    H = reshape(h, 3, 3);
    % Denormalise
    H = T2\H*T1;
```

FIGURE 1: Fonction Homography 2d

Ensuite, concernant la partie de sélection des points. En raison de la facilité d'identification, les points choisis ont été les coins du carré blanc au dessus du carré orange. En plus, comme les points ne sont pas collinéaires et on a quatre points, la matrice H est contrainte et une solution pour H est possible.

Par ailleurs, le code pour le script qui concerne cette question est représenté par la Figure 3.

FIGURE 2: Rectification de cadre de Pompei.

```
% Charge l'Image
imrgb = double(imread('Pompei.jpg'))/255;

% Afficher l'Image
imagesc(imrgb);

% Matrice Points d'Origine (3x4)
PtO = [143 112 370 357;45 254 262 44;1 1 1 1];

% Matrice Points de Destination (3x4)
PtD = [143 143 370 370;45 255 255 45;1 1 1 1];

% Fonction pour estimer l'homographie nécessaire pour redresser votre image
H = homography2d(PtO,PtD);

% Affiche l'image transformée :
imagesc(vgg_warp_H(imrgb,H));
```

FIGURE 3: Script pour la question 1

Question 2 - Assemblage d'Images

L'usage de plus de quatre points engendre l'algorithme Direct Linear Transformation (DLT) à avoir plus de contraintes que de valeurs de H à déterminées. Grâce au bruit, on

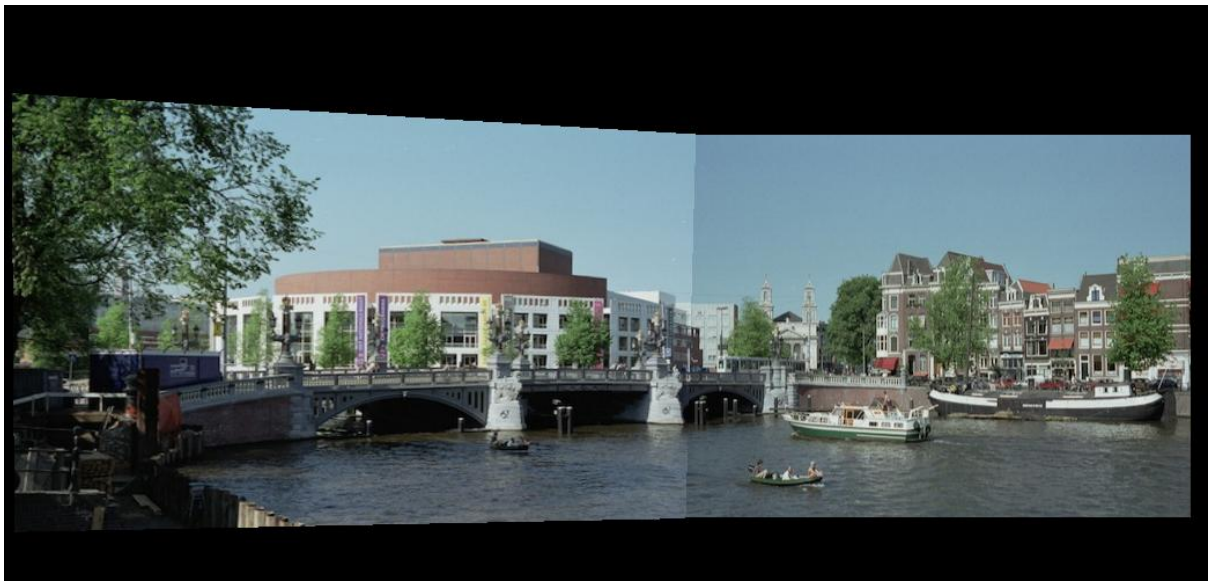

FIGURE 4: Assemblage de deux images comme panorama d'Amsterdam.

peut n'avoir plus une transformation projective qui satisfasse ces conditions. Dans ce cas, DLT trouve la meilleur transformation projective dans le sens que une fonction de coût soit minimisée.

Dans la Figure 4, on voit l'assemblage des images `Amst-1.JPG` et `Amst-2.JPG`. On a utilisé quatre points pour générer cette image panorama, deux coins du bâtiment blanc et deux coins dans le support de la pont. On a généré autre panorama avec 7 points et le résultat était le même à cause du faible bruit. Quand le bruit est plus grand, on voit une image qui n'est plus un quadrilatère et qui a des discontinuités.

Concernant l'affichage de trois images, on a exécuter deux transformations projectives de les images `Amst-1.JPG` et `Amst-3.JPG` afin de les afficher dans le plan de l'image `Amst-2.JPG`.

Finalement, pour l'assemblage de deux et trois images nous nous sommes appuyé sur les fonctions `homography`, `vgg_warp_H` et `max` afin de faire une seule image à partir d'autres. On remarque que l'usage de la fonction `max` éclaire grâce que selon l'encodage RGB des images, les composants avec les valeurs plus forts sont les plus clairs. Le main fonction qui est représenté par la Figure 5 et le résultat final de l'assemblage de 2 images et de 3 images

```
% Points d'origine et de destin à résoudre
if nPts == 4
    Pta  = [[435;136;1],[471;135;1],[494;246;1],[466;247;1]];
    Ptab = [[1;136;1],  [39;136;1], [60;248;1], [31;249;1]];
else
    Pta  = [[435;136;1],[471;135;1],[494;246;1],[466;247;1],[481;237;1],[440;157;1],[440;184;1]];
    Ptab = [[1;136;1],  [39;136;1], [60;248;1], [31;249;1], [45;238;1], [6;158;1],  [6;185;1]];
end
Ptc  = [[19;142;1], [57;142;1], [66;215;1], [36;217;1]];
Ptcb = [[400;144;1],[436;143;1],[446;216;1],[417;217;1]];

% Estimation de l'homographie par Direct Linear Transformation
Ha = homography2d(Pta,Ptab);
Hc = homography2d(Ptc,Ptcb);

bbox = [xmin, xmax, ymin, ymax];

% Application des l'hommographies
ima_warped = vgg_warp_H(ima, Ha, 'linear', bbox);
imb_warped = vgg_warp_H(imb, eye(8), 'linear', bbox);
if nImgs == 3
    imc_warped = vgg_warp_H(imc, Hc, 'linear', bbox);
end

% Présentation du résultat
im_fused = max(ima_warped, imb_warped);
if nImgs == 2
    imwrite(im_fused,'Amst-12.JPG');
    imagesc(im_fused);
elseif nImgs == 3
    im_fused = max(im_fused, imc_warped);
    imwrite(im_fused,'Amst-123.JPG');
    imagesc(im_fused);
end
```

FIGURE 5: Code pour l'assemblage d'images.

par la Figure 4 et Figure 6, respectivement.

FIGURE 6: Assemblage de trois images comme panorama d'Amsterdam.