



Compte rendu

Projet : Réalité Augmentée

MU4RBI09 : Vision par ordinateur

Chargé de TP : Xavier Clady

24/12/2022

Étudiants :

Leo Bellali, 28609955

Yousri Aboudaoud, 28712992

Parcours : M1 ISI | CMI EEA

Groupe de TP : 1

"Computer Vision is the process of discovering from images what is present in the world, and where it is."

(David Marr)

I. INTRODUCTION	4
II. Méthodes	5
III. Conclusion	9
Bibliographie	10

I. INTRODUCTION

Dans le cadre de l'unité d'enseignement de vision par ordinateur, nous avons eu l'opportunité de conduire un projet avec pour sujet la réalisation d'une solution de réalité augmentée.

La réalité augmentée représente la superposition de la réalité et d'images ou de sons en temps réel. Ceci, rendu possible par un système informatique.

La réalité augmentée trouve de nombreuses applications notamment dans les sciences, le divertissement, l'éducation et l'enseignement, les réseaux sociaux, et bien d'autres.

En voici quelques exemples :



e-marketing.fr : Comment exploiter le potentiel de la réalité augmentée ? [1]



Laval Virtual : La première plateforme de maintenance en réalité augmentée [2]

Concrètement il nous faut projeter un cube virtuel en 3 dimensions sur une séquence d'images ou une vidéo.

Pour cela, il nous faut passer par plusieurs étapes, telles que :

- la calibration
- le calcul de l'homographie
- l'estimation de la pose
- le tracking de primitives
- les expériences / l'évaluation

Tout ce travail est expliqué dans le présent rapport.

Note : ce rapport vient compléter les fichiers fournis avec. Pour simplifier sa lecture et la compréhension du code, nous avons résumé ici les parties les plus essentielles de notre réflexion. Nous avons également commenté notre code et l'avons rendu le plus explicite possible.

II. Méthodes

1) Calibration

Le fonctionnement d'une caméra repose sur le même principe qu'une "boîte noire", qu'on appelle la "sténopé" :

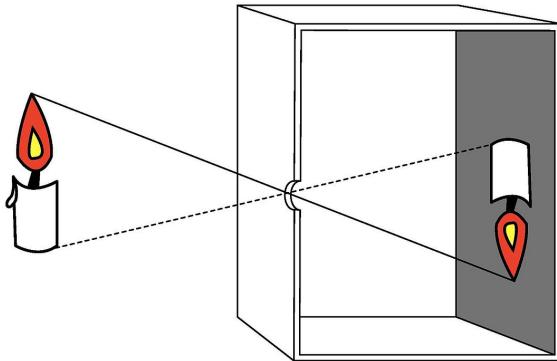


Figure 1 Sténopé [3]

Ici les points à l'intérieur (m) de la boîte noire sont la "projection image" de ceux à l'extérieur (M).

Faire la calibration de la caméra signifie déterminer son modèle géométrique(ie. Ses paramètres intrinsèques; compte du fait que la caméra que nous utilisons n'est pas calibrée), c'est-à-dire la fonction qui donne m en fonction de M .

On peut écrire : $M_{cam} = R \cdot M_{monde} + T$

$$\Rightarrow \tilde{M}_{cam} = (R \cdot T) \tilde{M}_{monde}$$

Le couple $(R \cdot T)$ étant le "pose" de la caméra

On peut également écrire : $\tilde{M}_{image} = K \cdot \tilde{M}_{cam}$, où

K désigne la matrice de paramètres "intrinsèques" à la caméra

Ainsi on obtient : $\tilde{M}_{image} = K \cdot (R \cdot T) \tilde{M}_{monde}$

$$\Rightarrow \tilde{M}_{image} = P \cdot \tilde{M}_{monde} \text{ où } P \text{ est un application linéaire à déterminer}$$

Pour cela on utilise une série de points m et M connus. Le point M est donné par une mire et m , projection de M , est extrait par traitement d'image.

Afin d'obtenir la matrice K nous avons utilisé l'application "camera calibrator" de la toolbox "computer vision" de Matlab.

Nous avons créé une première mire que nous avons affichée sur un écran. Par la suite, nous avons pris une série de 20 photos de cette mire, suivant des angles différents autour de l'écran pour obtenir différents points de vue du même objet 2D.

Nous importons ensuite ces images dans l'outil camera calibrator de Matlab. Nous obtenons alors une approximations des paramètres de notre caméra ainsi qu'une marge d'erreur(cf. Figure 2).

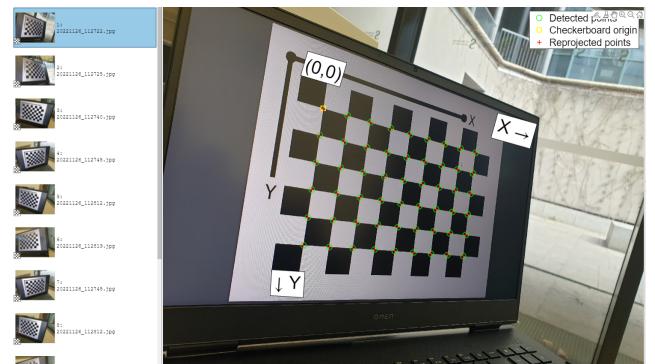


Figure 2 Calibration de la caméra

Nous avons alors éliminé quelques photos dont la marge d'erreur était trop élevée pour extraire nos paramètres avec le maximum de précision, nous nous sommes servies de la courbe d'erreur (cf. Figure 3) afin de supprimer les photos ayant un nombre de d'erreurs de reprojection élevé.

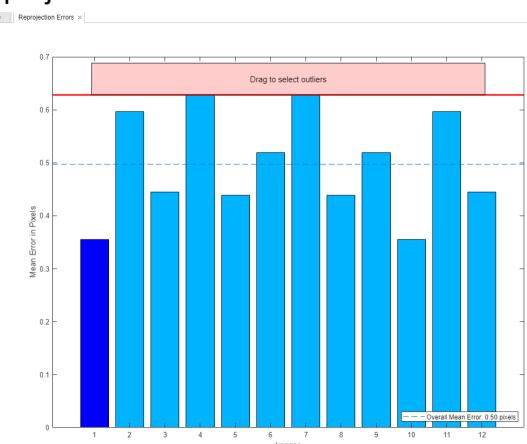


Figure 3 Erreur Moyenne de reprojection

Nous avons fait en sorte de prendre des points de vues différents sur le pattern (cf. Figure 4)

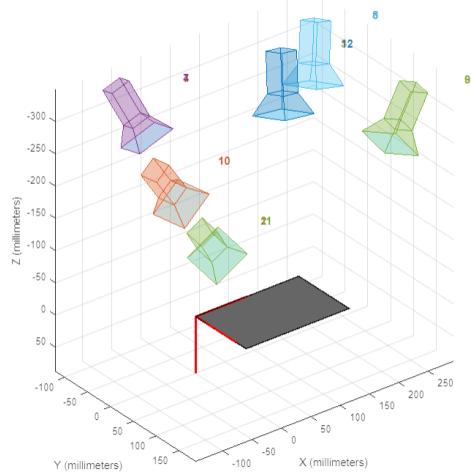


Figure 4 Pattern centric view

Pour construire notre matrice K, nous injectons les paramètres dans le notebook ipython "Homography" fournis avec le présent rapport, dans la section "Building the K matrix". Ce dernier a pour but de nous retourner la matrice K à partir des paramètres fournis par Matlab (Cf. Figure 4)

Building the K matrix

```
# Building the K matrix
#From the calibration of the camera we've done using matlab we found that
#all length units are in mm
Focal_length_X = 3146.7211
Focal_length_Y = 3150.2599
Principal_point_X = 1960.7933
Principal_point_Y = 1503.0496

K=np.zeros((3,3))
K[0][0]=Focal_length_X
K[1][1]=Focal_length_Y
K[0][2]=Principal_point_X
K[1][2]=Principal_point_Y
K[2][2]=1

print(K)

[[3.1467211e+03 0.000000e+00 1.9607933e+03]
 [0.000000e+00 3.1502599e+03 1.5030496e+03]
 [0.000000e+00 0.000000e+00 1.000000e+00]]
```

Figure 5 Création de la matrice K

On ne peut malheureusement pas encore réaliser de projection sans prendre en compte le couple (R T).

Ce qui introduit le concept d'homographie, qui permet de relier deux plans par une transformation linéaire (H).

2) Homographie

Pour estimer cette transformation représentée par une matrice (H), il nous faut connaître les m et m', étant les entrées et sortis d'une transformation linéaire. Ces entrées et sorties sont représentées par les coordonnées en x et y des points.

Pour cela, nous avons mesuré une série de points prise sur notre mire (m) et avons extrait leur coordonnée en pixel via l'outil prévu à cet effet de Matlab (m').

La matrice H résulte alors de l'application de la méthode de la DLT, réalisée par le notebook ipython "Homography" fournis avec le présent rapport.

Pour projeter des points simulant une forme en 3 dimensions sur un image en 2 dimensions, nous avons besoin de passer par une transformation rigide tridimensionnelle.

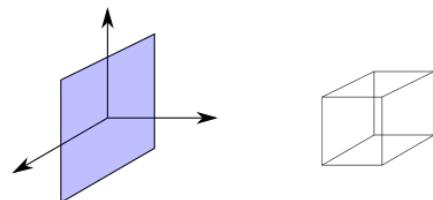


Figure 6 :Objectif, projeté un cube sur une surface 2D[4]

En utilisant l'homographie que nous avons extraite, voici un exemple de résultat que nous pourrions obtenir

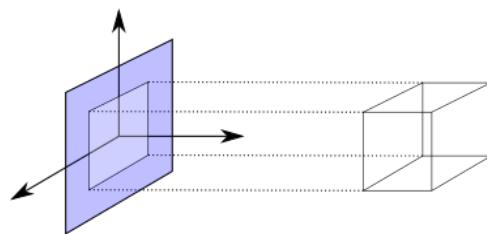


Figure 7 :Résultat apporté par l'homographie [4]

En effet, l'homographie ne prend pas en compte l'aspect tridimensionnel d'un objet. Ici

nous cherchons à ce que le cube apparaisse sur l'image comme tel :

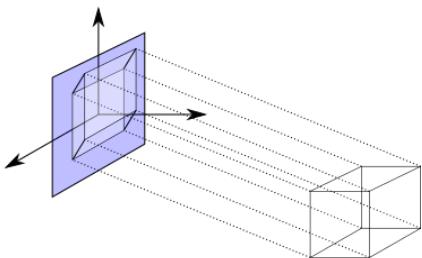


Figure 8: Résultat apporté par la matrice de projection [4]

3) Estimation de la pose

Afin de calculer la pose de la caméra qui nous permettre d'obtenir le résultat de la figure 5, nous utilisons trois données :

- L'homographie est connue
- La matrice K est connue
- La matrice de rotation de la matrice de projection est réversible et son déterminant est égale à 1

Dans la section “Building the 2D components of the projection matrix” du notebook “Homography” attaché à ce rapport (Cf. Figure 9), on trouve les calculs mathématiques permettant de déterminer les composantes bidimensionnelles de la matrice de projection à un facteur près.

```

1 #reverse the K matrix to get the results of K^-1*H
2 reverse_K = np.linalg.inv(K)
3
4
5
6 homography_mult_Reverse_K=np.matmul(reverse_K,H)
7 print(homography_mult_Reverse_K)
8 R1_tilde=np.array(homography_mult_Reverse_K[:,0]) #The first column
9 R2_tilde=np.array(homography_mult_Reverse_K[:,1]) #The second column
10 Trans_tilde=np.array(homography_mult_Reverse_K[:,2]) #The third column
11
12 #promoting the arrays from row to column ones
13 R1_tilde = R1_tilde[:, np.newaxis]
14 R2_tilde = R2_tilde[:, np.newaxis]
15 Trans_tilde = Trans_tilde[:, np.newaxis]
16

```

Figure 9: Building the 2D components of the projection matrix

Puis, dans les sections suivantes, nous nous retrouverons avec deux solutions possibles, dépendamment du signe du facteur dont on a évoqué plus haut. Après avoir calculé le déterminant des matrices de rotations propres à chaque solution, nous sélectionnons celle dont

le déterminant est égal à 1 (Compte tenu de la donnée N°3, utilisée pour déterminer la matrice de projection), l'autre solution constitue une matrice de projection liée à une caméra qui se situerait derrière la mire, ce qui physiquement impossible.

Après les premiers essais nous obtenons une matrice dont les coefficients étaient de l'ordre de 10^6 . Cette erreur provenait des mesures réalisées en mm et non en m.

En premier lieu, Nous utilisons la mire de la figure 10, Nous obtenons une matrice de projection (présente dans le notebook “Homography”).

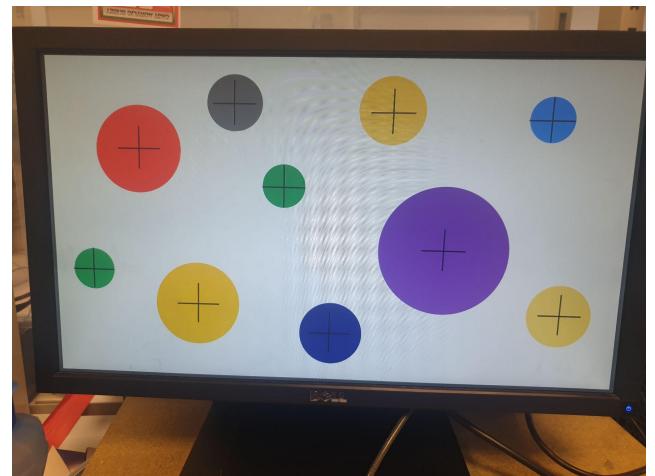


Figure 10: Pattern initial

Nous appliquons la matrice de projection résultante sur matlab en choisissant cinq points afin d'obtenir un carré complet (Cf. Figure 11) en utilisant la commande plot de Matlab et en appliquant la matrice de projection à des points monde dont la coordonnée Z est non nulle.



Figure 11: Projection Cube sur un plan 2D

4) Normalisation

Avant de calculer l'homographie, il est impératif de normaliser les données selon [5], en effet nous pouvons comprendre qu'involontairement (l'homographie 2D dépend du cadre de coordonnées dans lequel les points sont exprimés et certains systèmes de coordonnées sont en quelque sorte meilleurs que d'autres).

Sans normalisation les points x, x' d'une image typique sont de l'ordre($x,y,w)=(100,100,1)$ c'est à dire que x,y sont beaucoup plus grands que w . Dans A les entrées xx', xy', yx', yy' seront de l'ordre de 10^4 , les entrées xw', yw' de l'ordre de 10^2 , et les entrées ww' seront l'unité. Remplacer \tilde{A} par \tilde{A} signifie que certaines entrées sont augmentées et d'autres diminuées de telle sorte que la somme carrée des différences de ces changements soit minimale (et que la matrice résultante ait un rang 8). Cependant, et c'est là le point essentiel, augmenter le terme ww' de 100 signifie un énorme changement dans les points de l'image, alors qu'augmenter le terme xx' de 100 ne signifie qu'un léger changement. C'est la raison pour laquelle toutes les entrées de A doivent avoir une magnitude similaire et que la normalisation est essentielle.

L'effet de la normalisation est lié au nombre d'états de l'ensemble des équations DLT, ou plus précisément au rapport $d/dn-1$ de la première à l'avant-dernière valeur singulière de la matrice d'équation A. Ce point est étudié plus en détail dans le cadre de l'étude de l'effet de la normalisation. Ce point est étudié plus en détail dans [5]]. La normalisation est une étape essentielle !

Algorithme :

(i) Normalisation de x : calculer une transformation de similarité T, composée d'une translation et d'une mise à l'échelle, qui amène les points x_i à un nouvel ensemble de points $x_{i\tilde{}}$ tel que le centroïde des points $x_{i\tilde{}}$ est la coordonnée origine(0,0), et leur distance moyenne de l'origine est racine carrée(2)

- (ii) Normalisation de x' : calculer une transformation similaire T' pour les points de la seconde image, en transformant les points x_i en $x_{i\tilde{}}$.
- (iii) DLT : appliquer DLT aux correspondances $x_{i\tilde{}} \leftrightarrow x_{i\tilde{}}$ pour obtenir une homographie $H\tilde{}$
- (iv) Dénormalisation : SET $H = T'^{-1}H\tilde{}T$

Nous avons constaté une légère amélioration suite à l'implémentation de cette modification.

5) Tracking

Un élément figurant dans le cahier de charge est une vidéo dans laquelle le cube 3D a été projeté. Pour faire cela nous devons être capable de suivre les points d'intérêts, et pour ce faire il existe diverses méthodes. La première étant celle d'extraire les caractéristiques de type "MinEigine", après avoir testé cette méthode, nous nous sommes rendus compte qu'elle n'était pas efficace avec la mire que nous avions créée (Cf. Figure 10), en effet cette méthode (dite de Lucas Kanade) extrait des points dont les caractéristiques sont bien spécifiques, ce qui nous donne des résultats forts peu convenant (un bon tracking au début de la séquence d'images, puis une divergence des points désirés). Nous avons opté plus tard pour une mire dont les points désirés sont des disques plus petits et au nombre de quatre (Cf. Figure 12).

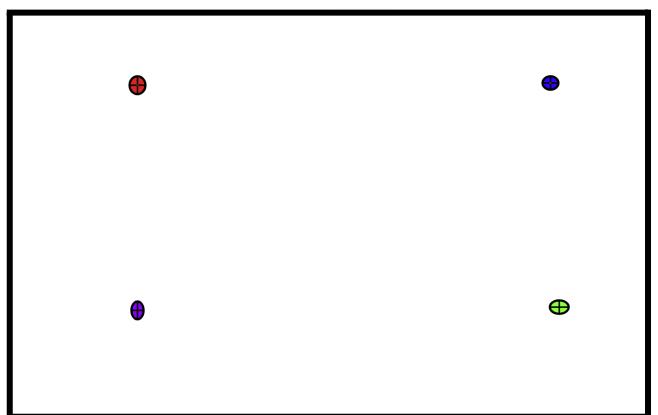


Figure 12: Mire finale

Nous avons d'autre part modifier les caractéristiques à suivre, en utilisant les KAZE

features (qui sont les plus performants d'après nos expériences). Nous obtenons des solutions satisfaisantes pour les points suivis(Cf. Figure 13).

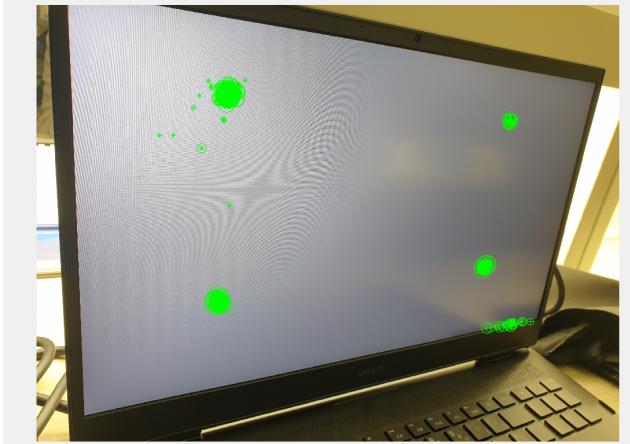


Figure 13: En vert les points ayant le plus de caractéristiques

Cependant, deux problèmes résultent de ces modifications, tout d'abord le nombre de points sélectionnées est disproportionné à nos besoins (quatre points suffisent afin de projeter un cube) Une solution serait de calculer la moyenne des coordonnées de chaque groupe de points. Un autre problème étant, le suivi de points diverge à un certain moment, en effet si l'on choisit de traiter une séquence d'image force est il de constater que la distance entre les images ne doit pas être grande, autrement le tracker ne pourra plus suivre les points puisqu'il utiliser la nème frame pour détecter les points dans la n+1 frame.

La solution la plus optimale que nous ayons trouvé pour le premier est d'utiliser la commande `ginput` de Matlab, et quant au second problème la solution est de prendre une vidéo du pattern au lieu d'une séquence d'image(même si cette vidéo pourrait contenir des problèmes liés à la mauvaise compression).

Cette solution nous a donné de bon résultats, disponibles en téléchargeant la vidéo suivante puis la sauvegarder sous le nom “Pattern.mp4”, le lien :

<https://www.swisstransfer.com/d/27369a9d-1694-4a5f-988d-0f0597582ee1>

Puis lancer le fichier matlab “Augmented_REALITY”.

Comme on peut le voir les points d'intérêts, à savoir les centres des cercles sont suivis tout au long des 358 frames.

Après avoir récupéré les coordonnées de ces images, nous procédons à la projection de ces points sur la vidéo. D'abord, nous exportons les coordonnées sous forme homogène au Notebook “ProjectionMatrixVideo” dans lequel une boucle “for” calcule l'homographie des coordonnées associées à chaque frame, puis applique la matrice de projection et l'homographie, tout en normalisant les données (avant le calcul de l'homographie et après son application), puis les données seront visualisées sur les frames.

III. Conclusion

Ce projet a été enrichissant surtout par la pédagogie qui y a été appliquée. Nous avons effectué un bon nombre de recherches, effectué des croisements entre les sources afin de vérifier la véracité des données. Nous cherchions chaque mot clé pouvant nous amener vers la fin du projet, et nous avons appris énormément de notions sur plusieurs domaines (Géométrie projective, traitement d'image...), des connaissances qui nous seront utiles pour notre parcours.

Par manque de temps, nous n'avons malheureusement pas pu visualiser le résultat final sur la vidéo.

Nous avons réussi à effectuer la calibration, calculer l'homographie, la matrice de projection; nous avons réussi à suivre les points d'intérêt, à projeter un cube sur le plan 2D.

Bibliographie

I. Introduction

[1] :

<https://www.e-marketing.fr/Thematique/cross-canal-1094/Breves/Comment-exploiter-potentiel-realite-augmentee-343772.htm>

[2] :

<https://blog.laval-virtual.com/la-premiere-plateforme-de-maintenance-en-realite-augmentee/>

II. Méthodes

1. Calibration

[3] :

<https://fr.wikipedia.org/wiki/St%C3%A9rioscopie>

2. Homographie

[4]

<https://openclipart.org/detail/324480/projections>

4. Normalisation

[5] Multiple View Geometry in computer vision, 2nd Ed., R. Hartley and A. Zisserman, 2003, Cambridge, section 4.4.4, pp.107