TP VSION
Réalité augmentée

OPPMANN Thomas
LESDALONS Christopher

CENTRALE NANTES

SOMMAIRE

INTRODUCTION

L'objectif de ce TP est de créer une petite application de réalité augmentée en C++ s'appuyant sur les bibliothèques *OpenCV* et *AruCo*.

Dans un premier temps, nous réaliserons nos premiers pas avec Aruco en écrivant un simple programme qui détectera les marqueurs. Nous améliorerons ensuite ce programme à l'aide des fichiers téléchargés sur Hippocampus pour afficher des cubes en fil de fer au dessus des marqueurs.

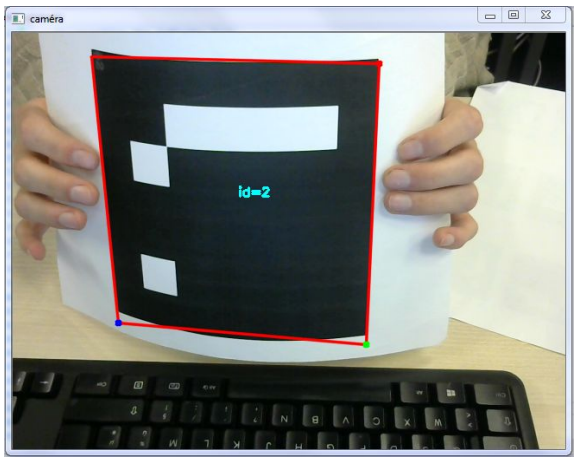
Enfin, nous avons codé une application dessinant des personnages sur les marqueurs, et nous les avons fait sauter sur place.

ARUCO : PREMIER PROGRAMME

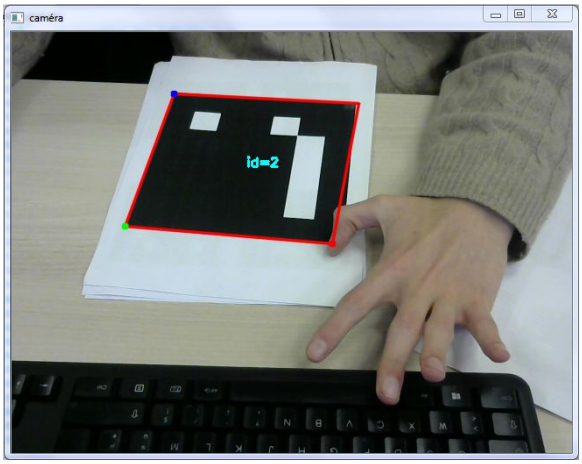
Nous avons créé un premier programme capable de détecter les marqueurs. Celui-ci s'appuie sur la webcam et affiche en temps réel l'identification du marqueur.

Nous avons ensuite testé les limites de la détection des marqueurs :

- Feuille du marqueur plié : le marqueur reste détecté si la courbure de la feuille n'est pas trop importante

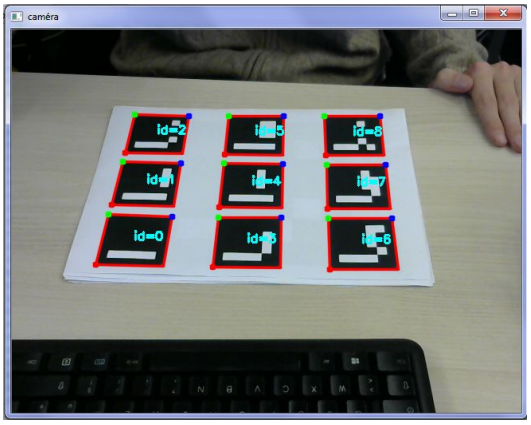


- Marqueur partiellement masqué : le marqueur reste détectable s'il est très légèrement masqué (pas plus que 2-3 cm²). Si l'obstruction est trop importante, celui-ci n'est plus détecté.



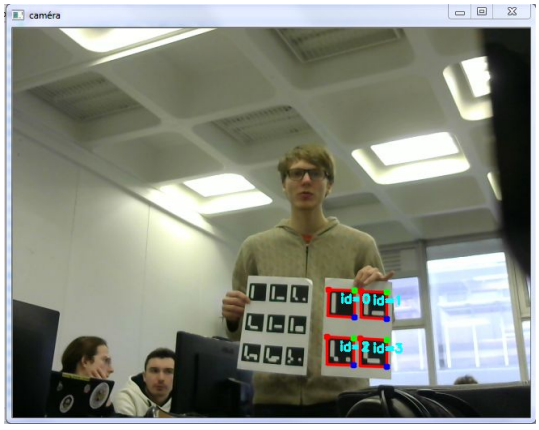
- Angle de la caméra : le marqueur peut être détecté malgré un angle entre la normal du marqueur et la direction de la caméra très important.

- Multiples marqueurs : plusieurs marqueurs peuvent être détectés simultanéments sans aucun soucis.



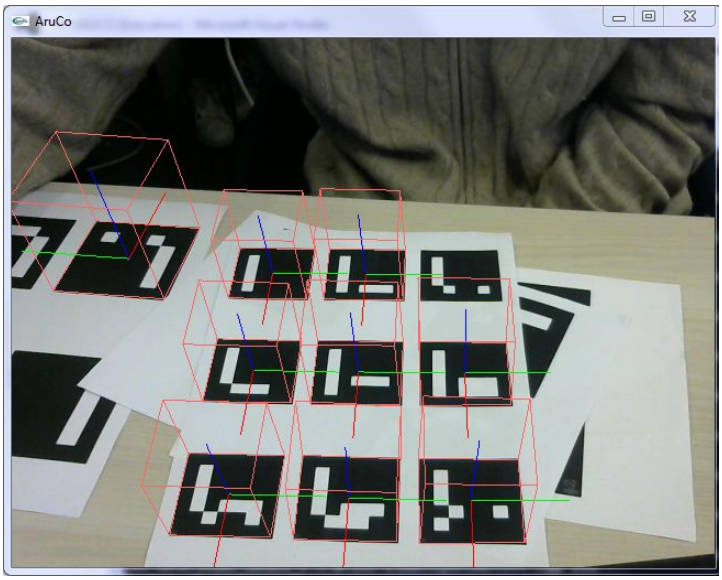
- Multiples marqueurs identiques : si plusieurs marqueurs sont identiques, seulement un marqueur par type sera reconnu.

- Distance : les marqueurs sont détectés s'ils sont relativement proche de la caméra. La limite se trouve environ à 2m.



PREMIERE AUGMENTATION

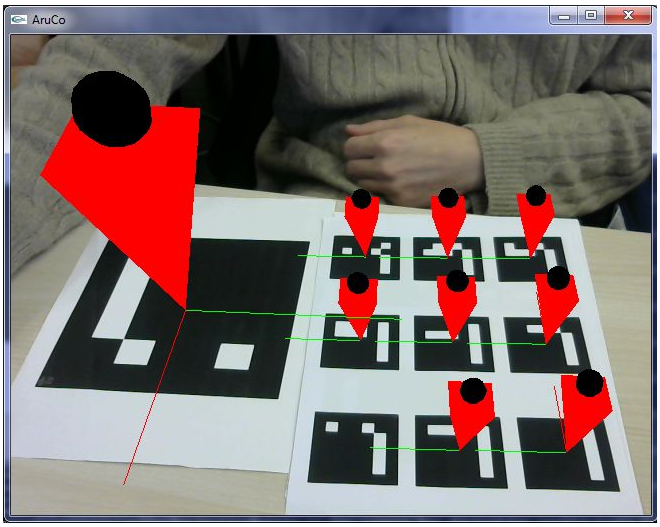
Nous avons complété l'application permettant le dessin de cubes en fil de fer au dessus des marqueurs. Etant donné que le programme dessine les cubes seulement si le marqueur est détecté, le dessin des cubes est soumis aux mêmes limites que celles décrites dans la partie précédente. On remarque par exemple sur la photo ci-dessous qu'un cube n'a pas été dessiné parce que le même marqueur est déjà présent ailleurs sur l'image.

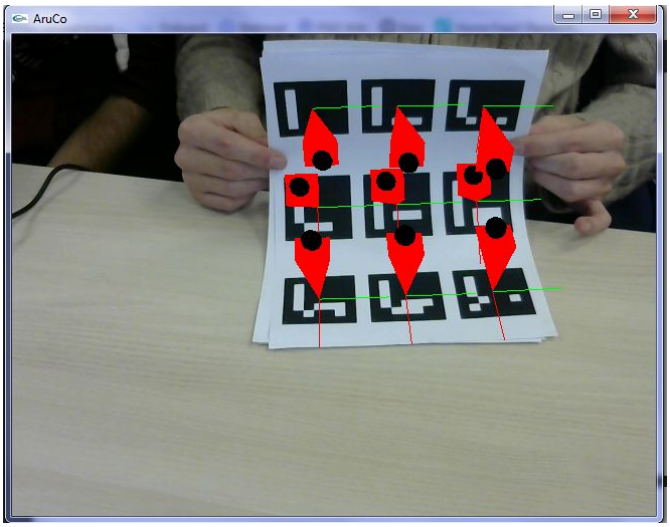


Ce programme s'appuie sur la fonction *drawScene* de la classe Aruco. Cette fonction permet l'affichage des cube. Cette fonction procède ainsi :

1. Vérifie si l'image a bien été chargée
2. On se place dans le repère de la caméra
3. Pour chaque marqueur détecté on dessine le cube. La face inférieure du cube correspond au marqueur.

Nous avons ensuite remplacé le dessin du cube par un petit personnage pyramidale. Nous constatons que la rotation des marqueurs est relativement stable.





Le code pour l'affichage du personnage est écrit ci-dessous.

```
void affichePerso() {
    //Corps
    glColor3f(1, 0, 0);
    glBegin(GL_QUADS);
    for (int i = 4; i>= 1; i--) {
        glVertex3f(personnage[i].x, personnage[i].z, personnage[i].y);
    }
    glEnd();
   for (int i=0; i<4; i++){
       glBegin(GL_TRIANGLES);
    for (int j=0; j<3;j++){
        glVertex3f(personnage[facePersonnage[i][j]].x, personnage[facePersonnage[i][j]].z, personnage[facePersonnage[i][j]].y);
    }
        glEnd();
   }

   //Tête
   glColor3f(0.0f, 0.0f, 0.0f);
   glPushMatrix();
   glTranslatef(0.0f ,0.0f, 0.115f);
   glutSolidSphere(0.015f,10,10);
   glPopMatrix();
};
```

```
// For each detected marker
for (unsigned int m=0;m<m_Markers.size();m++)
{
   m_Markers[m].glGetModelViewMatrix(modelview_matrix);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   glLoadMatrixd(modelview_matrix);

   // Disabling light if it is on
   GLboolean lightOn = false;
   glGetBooleanv(GL_LIGHTING, &lightOn);
   if(lightOn) {
      glDisable(GL_LIGHTING);
   }

   // Drawing axis
   drawAxis(m_MarkerSize);

   affichePerso();
   // Re-enabling light if it is on
   if(lightOn) {
      glEnable(GL_LIGHTING);
   }

   glPopMatrix();
}
```

```
typedef struct {
   float x;
   float y;
   float z;
} vertex;

// Création des sommets d'un perso
static vertex personnage[5]={
   {0.0f,0.0f,0.0f},
   {0.025,0.1f,0.025},
   {0.025,0.1f,-0.025},
   {-0.025,0.1f,-0.025},
   {-0.025,0.1f,0.025}
};

// Création des faces d'un perso
static int facePersonnage[4][3] ={
   {0,1,2},
   {0,2,3},
   {0,3,4},
   {0,4,1},
};
```

APPLICATION DE REALITE AUGMENTEE

Nous avons créer une petite application faisant sauter nos personnages sur les marqueurs. Cependant, plus il y a de personnages sur l'écran, plus ceux-ci seront "heureux" et sauteront plus haut et plus rapidement. Un personnage seul ne sautera pas du tout.

Mode d'emploi : il suffit de filmer les marqueurs avec une webcam. Une feuille avec de nombreux marqueurs est préférable. Masquez différents marqueurs avec vos mains ou tout autre objet et observez les changements.

Par manque de temps, l'animation de saut est très basique. Le personnage s'élève d'une certaine hauteur puis, arrivé en haut, il se téléportera à ses coordonnées d'origine ; la redescente du personnage n'a pas été codée. De plus, nous comptions également intégrer une rotation du personnage sur lui même pendant son saut. Cette fonctionnalité de fonctionne néanmoins pas encore.