Rapport TP de Analyse d'Image et Réalité Virtuelle

Juan Gabriel Barros Gavilanes

1 Description du travail.

Le travail consiste en cinq points à faire :

1. Afficher un flux vidéo
2. Afficher le modèle en 3D
3. Calibrage
4. Manipuler le object 3D
5. Bonus

2 Développement

Afficher un flux vidéo

Pour ce point la consigne est :
Ecrire un programme OpenGL/GLUT/OpenCV qui affiche un flux vidéo. La documentation OpenCV vous sera utile (cf. HighGUI Reference Manual / section Video I/O functions).

En concernant cette partie, on a trouvé des documents qui présentent des façons faciles de afficher un flux vidéo, même qu'il soit provenant de une camera ou de un fichier de vidéo (cvCaptureFromAVI). Pour le cas de ce travail on a utilisé la fonction cvCaptureFromCAM.

```
1  int main(int argc, char** argv){
2      CvCapture* capture = 0;
3      capture = cvCaptureFromCAM(argc == 2?argv[1][0]-'0':0);
4      if( !capture ) return -1;
5      cvNamedWindow( "Captura_de_video", 0 );
6      for(;;) {
7              int c;
8              IplImage* frame=cvQueryFrame(capture);
```

```
9               if(!frame) break;
10
11               cvShowImage( "Captura_de_video", frame);
12               c=cvWaitKey(30); // run at ~20-30fps speed
13               if( (char)c == 27 ) break;
14      }
15      cvReleaseCapture(&capture);
16      cvDestroyWindow("Captura_de_video");
17  }
```

Le problème principal a était trouver un bon exemple dans la documentation ou dans l'Internet.

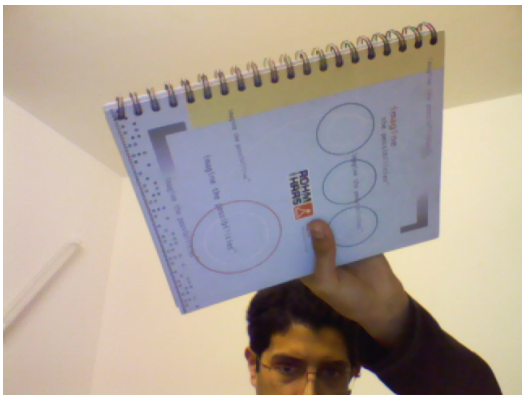

FIGURE 1 – image pris avec camera

Afficher le modèle en 3D

Pour ce point la consigne est :
Notre modèle 3D sera une boite de ramettes de feuille A4 (on a plein à l'école !). Modéliser la boite en OpenGL et afficher-là dans une fenêtre indépendante du flux vidéo.

Pour modéliser la boite, on utilise des carrés en respectant les relations entre les dimensions de chaque bord. Il existe un bord mayeur, autre moyen et un bord mineur. Pour modéliser ce volume on utilise le code suivant :

```
1   /*opengl variables*/
2   const float CAJA_LARGO = 8.0f;
3   const float CAJA_ANCHO = 2.0f;
4   const float CAJA_ALTO = 5.0f;
5
6           glBegin(GL_QUADS);
7                //Top face //jaune
8           glColor3f(1.0f, 1.0f, 0.0f);
9           glNormal3f(0.0, 1.0f, 0.0f);
```

```
10          glVertex3f(-CAJA_LARGO / 2, CAJA_ANCHO / 2, CAJA_ALTO );
11          glVertex3f(-CAJA_LARGO / 2, CAJA_ANCHO / 2, CAJA_ALTO / 2);
12          glVertex3f(CAJA_LARGO / 2, CAJA_ANCHO / 2, CAJA_ALTO / 2);
13          glVertex3f(CAJA_LARGO / 2, CAJA_ANCHO / 2, CAJA_ALTO );
14           //Front face //bleu
15          glNormal3f(0.0, 0.0f, 1.0f);
16          glColor3f(0.0f, 0.0f, 1.0f);
17          glVertex3f(-CAJA_LARGO / 2, -CAJA_ANCHO / 2, CAJA_ALTO / 2);
18          glVertex3f(CAJA_LARGO / 2, -CAJA_ANCHO / 2, CAJA_ALTO / 2);
19          glVertex3f(CAJA_LARGO / 2, CAJA_ANCHO / 2, CAJA_ALTO / 2);
20          glVertex3f(-CAJA_LARGO / 2, CAJA_ANCHO / 2, CAJA_ALTO / 2);
21              //Back face
22          glNormal3f(0.4, 0.3f, 1.0f);
23          glColor3f(0.0f, 0.5f, 0.5f);
24          glVertex3f(-CAJA_LARGO / 2, -CAJA_ANCHO / 2, CAJA_ALTO );
25          glVertex3f(-CAJA_LARGO / 2, CAJA_ANCHO / 2, CAJA_ALTO );
26          glVertex3f(CAJA_LARGO / 2, CAJA_ANCHO / 2, CAJA_ALTO );
27          glVertex3f(CAJA_LARGO / 2, -CAJA_ANCHO / 2, CAJA_ALTO );
28          glEnd();
```

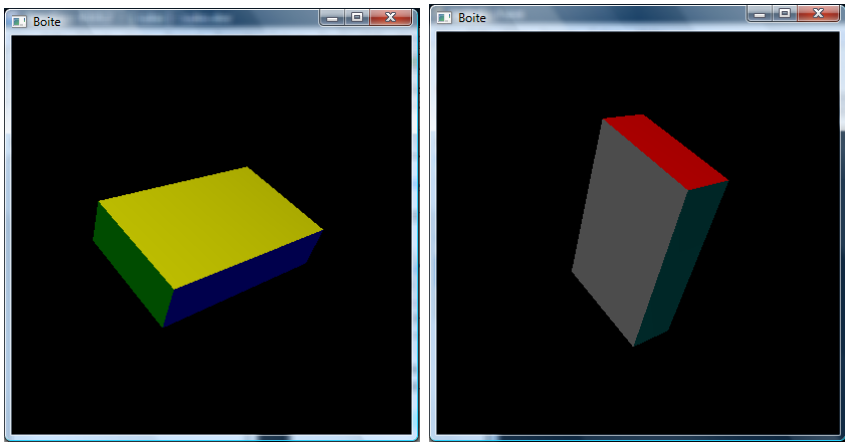

FIGURE 2 – boite modélise

Boite : Le problème principal a était trouver un bon exemple dans la documentation ou dans l'Internet. Parce que ils existent des autres façons de modéliser une boite. Par exemple avec le command `glutWireCube(2);` ou `glutSolidCube(1);` mais avec une transformation d'escale en respectant les relations `glScaled(1.5,1.0,.5)`, mais avec cette approche c'est difficile de trouver et manipuler les coordonnées de chaque bord et carré.

Affichage : autre problème était de trouver le bonne lieu pour le command `glutMainLoop()` car celui-ci donne le control aux fonctions OpenGL et on a besoin d'une loop de control pour le programme OpenCV. On a trouvé la solution en utilisant la fonction idle de OpenGL comme le control pour OpenCV aussi.

Calibrage

Pour ce point la consigne est :
L'objectif de cette étape est de placer notre modèle sous le même angle du vue que notre vraie boite. Pour cela nous allons, dans un premier temps, procéder de manière supervisé. En analysant rapidement notre objet, un recalage 2D/2D (une homographie) est suffisant. Voilà, par exemple, une façon de faire :

- *Mettez en correspondance les 4 coins de la face du haut (ceux de l'image) avec les même sommets de votre modèle 3D.*
- *Calculer la matrice d'homographie (cf. OpenCV Reference Manual / section Camera calibration).*
- *Appliquer la matrice calculée comme matrice de transformation de votre objet. Il faut remarquer que ce n'est qu'une face qui a besoin d'être transformée, les autres étant tracées à partir de celle-là. Attention, cette matrice contient à la fois les transformations de vue et de projection !*

REMARQUE : La transformation que vous avez calculée est une transformation bijective. Il suffit d'inverser la matrice obtenue pour passer de l'espace du modèle à l'espace de l'image. Voilà de quoi déplacer un élément virtuellement sur votre modèle et le voir également se déplacer sur votre vidéo !

On a utilisé une exemple de calcul de matrice homographie, pour mettre en correspondance les 4 points. On a adapté une algorithme de reconnaissance de carrés en OpenCV pour trouver une de la boite. On utilisé Canny et threshold méthodes après de un changement dans la taille pour réduire le bruit. Trouver les points en OpenCV est un peu facile malgré les plusieurs paramètres à tester.

Dans la section de code suivant on peut observer qu'ils existent plusieurs méthodes pour manipuler la projection et le paramètre MODELVIEW. Les lignes documentées sont valides pour autres exemples. La matrice H_GL est laquelle a toute l'information de l'homographie.

```
1  glMatrixMode(GL_PROJECTION);
2  glLoadIdentity();
3          //gluPerspective(45.0, (float)ancho / (float)alto, 1.0, 200.0);
4  glOrtho(0,ancho, 0, alto, -1, 1);
5        glMatrixMode(GL_MODELVIEW);
6          //glLoadIdentity();
7  glLoadMatrixf(H_GL);
```

Le principal problème dans cette partie est notamment le chevauchement entre les carrés qui conforment la boite 3D. On a besoin de plus practice avec les projections et OpenGL pour obtenir un mieux travail.

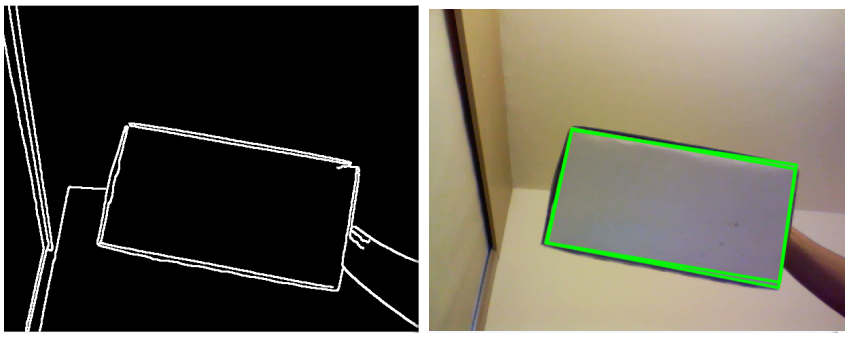

FIGURE 3 – point détection en utilisant Canny filtre.

FIGURE 4 – projection de points en utilisant homographie et aussi modèle 3D.

Manipuler le object 3D

On arrive d'une manière très primitive de manipulation de l'objet 3D avec le travail réalisé dans la section précédente. Mais avec les mêmes problèmes à résoudre.

3 Conclusion

L'auteur de ce travail pense que c'est une bonne introduction pour le monde 3D, et le traitement d'images et fluxes de vidéo avec OpenCV. De toute façon je considère que c'est un peut difficile si on n'a pas la connaissance de OpenGL.