TP Réalité Augmentée

ABERKANE Salwa

BERTRAND Paul

1 Introduction

L'objectif de ce TP est de réaliser une application de réalité augmentée en utilisant la bibliothèque OpenCV que nous avons déjà rencontré lors des précédents TP et la bibliothèque ArUco, basée sur OpenCV et qui est dédiée à la réalité augmentée. La première partie du TP est consacrée à la détection de marqueurs. Dans un second temps, nous chercherons à appliquer une augmentation sur un marqueur en utilisant GLUT. Enfin, nous proposerons une application de réalité augmentée utilisant les différents aspects abordés dans les parties précédentes.

2 Mise en place de ArUco et OpenCV

La première étape pour la création du projet a été d'inclure les bibliothèques ArUco et OpenCV. OpenCV a pu être inclus grâce aux feuilles de propriétés que nous avions créé pour les TP précédents. Pour ce qui est de l'inclusion d'ArUco, nous avons modifié les propriétés du projet afin d' ajouter les répertoires d'inclusion et d'en-tête de ArUco.

3 Détection des marqueurs avec ArUco

Dans un premier temps, nous cherchons détecter les marqueurs à dans les images que nous recevons de la caméra. Cela est réalisé grâce à la classe *MarkerDetector* fournie par ArUco.

Pour commencer, nous essayons de détecter un marqueur directement sur une image de marqueur. Nous remarquons que la détection du marqueur n'a pas lieu. Cela s'explique par le fait qu'ArUco détecte les marqueurs seulement lorsqu'ils sont entourés de blanc, ce qui n'est pas le cas pour les autres marqueurs, qui sont imprimés sur du papier blancs.

Ensuite, nous réalisons la détection en temps réel sur l'image de les images reçues par la caméra. Pour cela, nous écrivons d'abord un bout de code qui permet d'afficher l'image reçue par la caméra. Puis, en fonction de la détection ou non d'un marqueur, nous dessinons des carrés pour les repérer à l'aide de la méthode *draw*. Les résultats obtenus sont exposés figures 1 à 4. On remarque que la détection fonctionne généralement bien, même lorsque la caméra a un angle de vue incliné.Il arrive toutefois que le marqueur ne soit pas détecté : quand l'angle de vue est trop incliné, que la feuille qui contient le marqueur est tordue, ou que le marqueur est proche du bord de l'image (figure 3).

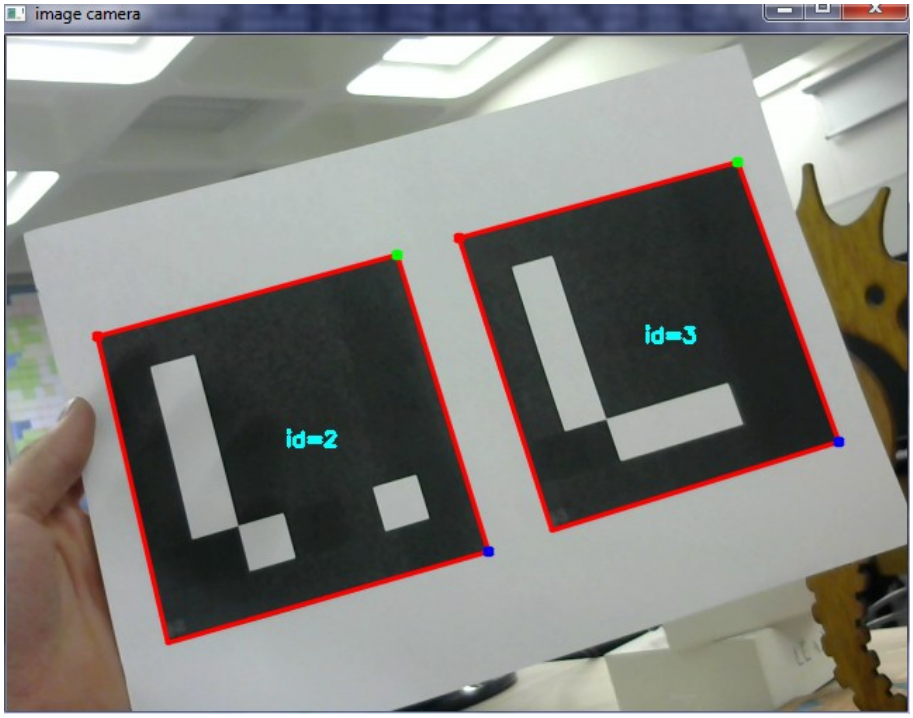

Figure 1: Détection des marqueurs

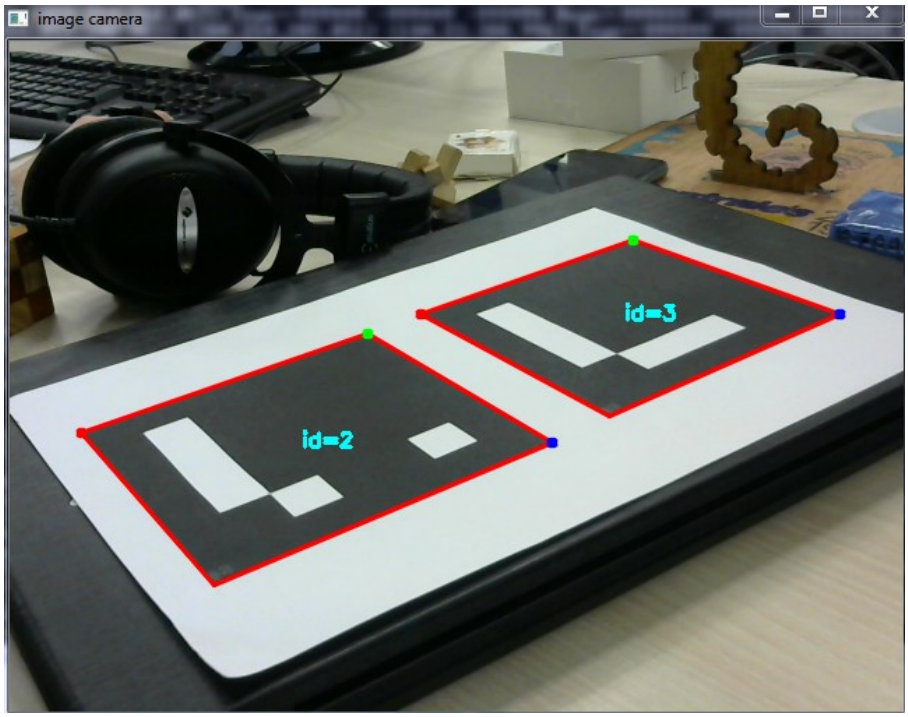

Figure 2: Détection des marqueurs avec un angle de vue

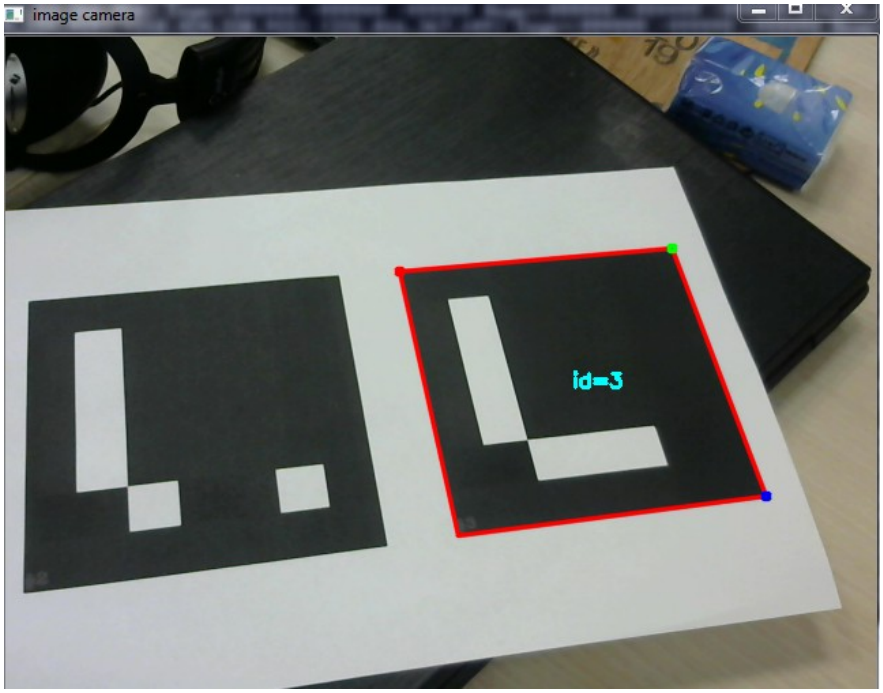

Figure 3: Détection d'un marqueur et non détection de l'autre

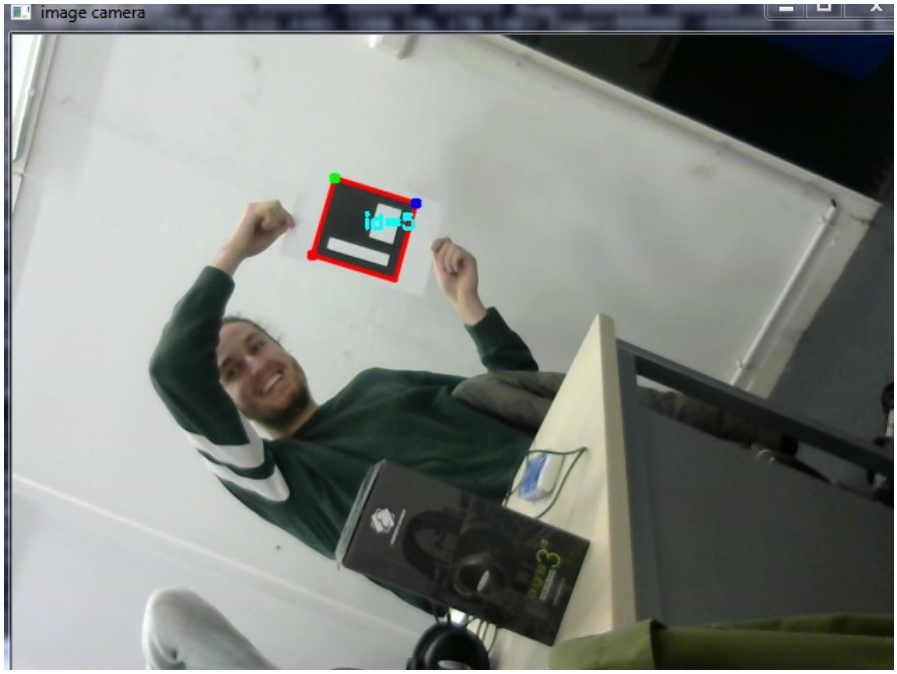

Figure 4: Détection d'un marqueur à plus grande distance

4 Première augmentation

Grâce au code fourni dans le sujet du TP, nous avons pu réaliser une première augmentation de l'image de la caméra en affichant un cube 3D sur les marqueurs. Le résultat obtenu est affiché figure 5. Pour obtenir cet affichage, ArUco la méthode *drawScene*. Celle-ci permet de parcourir les marqueurs détectés, et, pour chacun d'entre eux, dessiner les axes x, y et z grâce à la fonction *drawAxis* puis de dessiner un cube en fil de fer au centre de ce marqueur et de même taille.

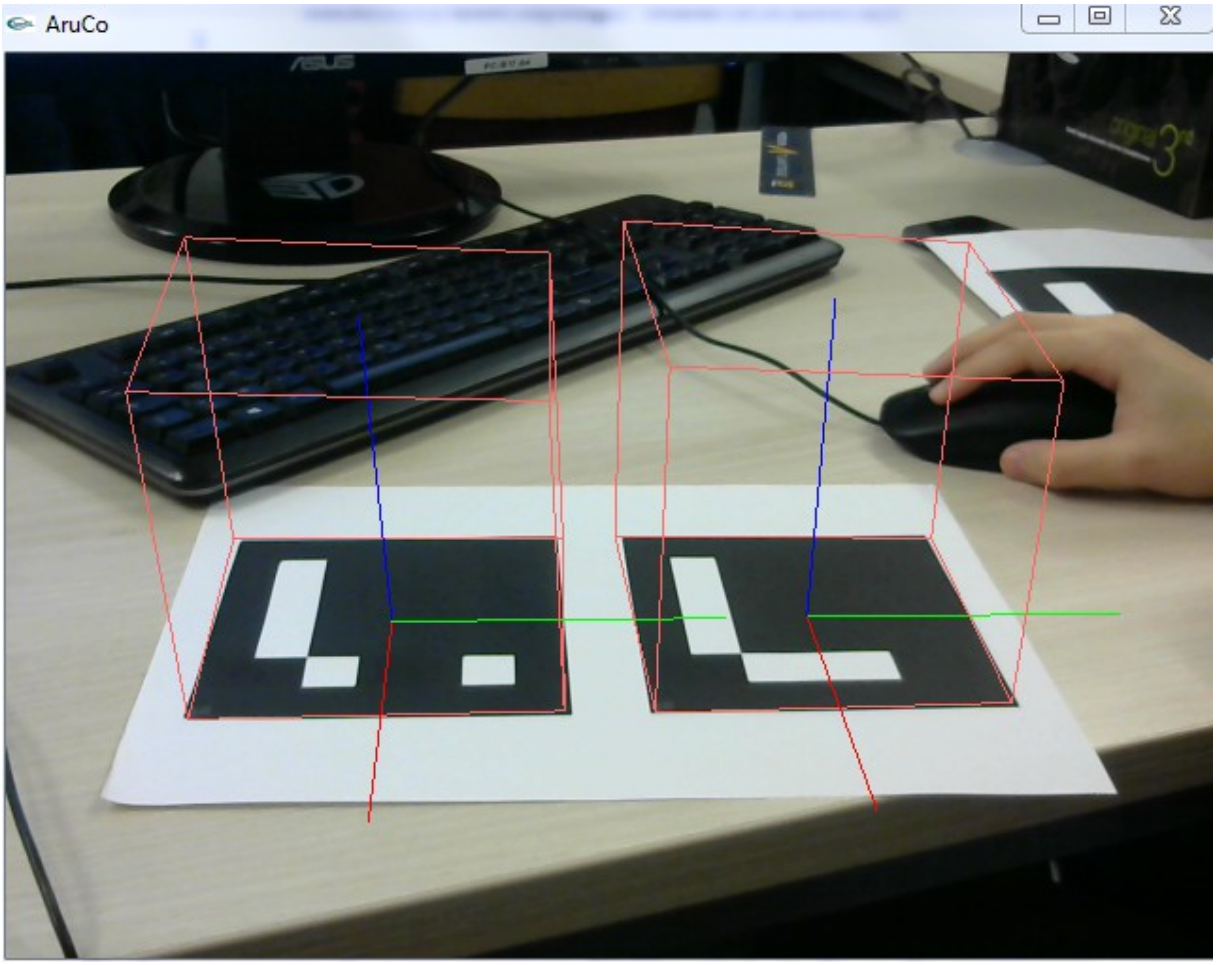

Figure 5: Première augmentation

Nous avons ensuite modifié la forme affichée sur le marqueur en modifiant la fonction *drawScene* de ArUco. Nous avons cherché à créer un carré flottant au dessus du marqueur. Pour cela, nous avons changé les valeurs de la translation effectuée avant le dessin de la forme, puis créé un carré de même côte de que marqueur. Le résultat obtenu est présenté figure 6 et 7, et le code utilisé est figure 8.

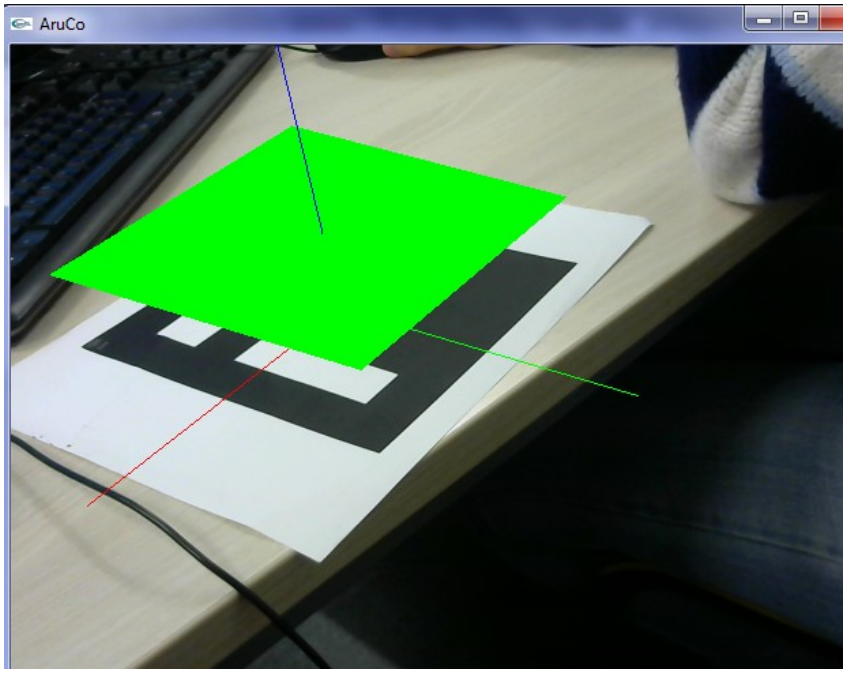

Figure 6: Carré flottant au dessus du marqueur

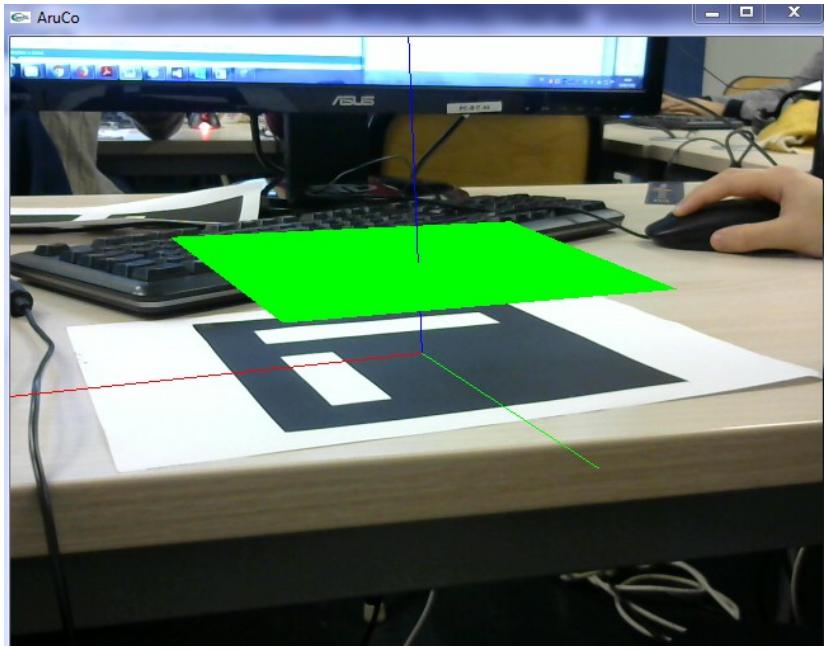

Figure 7: Carré flottant au dessus du marqueur

```
// Drawing axis
drawAxis(m_MarkerSize);

// Drawing a cube
glColor3f(1,0.4f,0.4f);
///translater le carré pour qu'il soit flottant au dessus de marqueur
glTranslatef(-m_MarkerSize/2, -m_MarkerSize/2, m_MarkerSize/4);


glPushMatrix();

//////création d'un carré de côté m_MerkerSize
glColor3f(0,1,0);
glBegin(GL_POLYGON);
glVertex3f(0.0f, 0.0f, 0.0f);
glVertex3f(m_MarkerSize,0.0f, 0.0f);
glVertex3f(m_MarkerSize,m_MarkerSize, 0.0f);
glVertex3f(0.0f,m_MarkerSize, 0.0f);
glEnd( );
```

Figure 8: Code utilisé pour implémenter le carré flottant

5 Application de Réalité Augmentée

Nous cherchons maintenant à réaliser une petite application de réalité augmentée utilisant au moins 2 marqueurs. Nous avons choisi d'implémenter le jeu suivant : lorsque l'on utilise les 2 marqueurs 2 et 3 et que les deux sont visibles, si l'on occulte le marqueur 3, le carré flottant au dessus du marqueur s'élève un peu plus haut. Plus on fait d'occlusions successives sur le marqueur 3, plus le carré du marqueur 2 s'élève. Si le marqueur 2 est occulté, son carré flottant aura à nouveau sa valeur initiale la fois suivante où il sera visible. Nous avons aussi fait en sorte dans l'application de modifier la couleur du carré en fonction de la taille du marqueur. Des captures de l'application sont présentées figures 9 à 11.

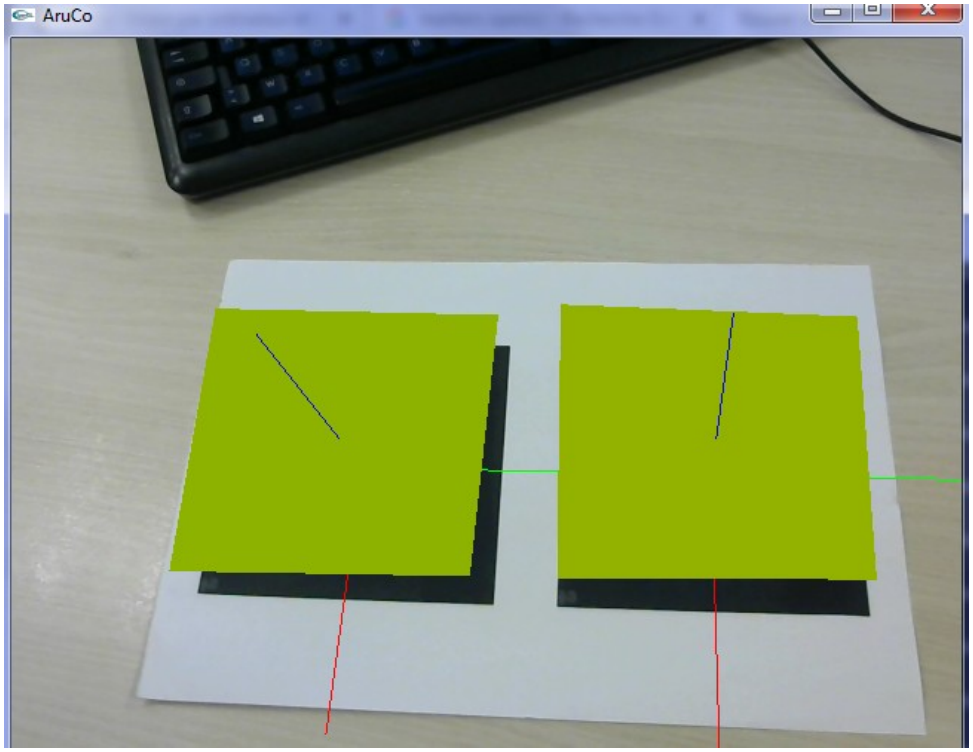

Figure 9: État initial de l'application

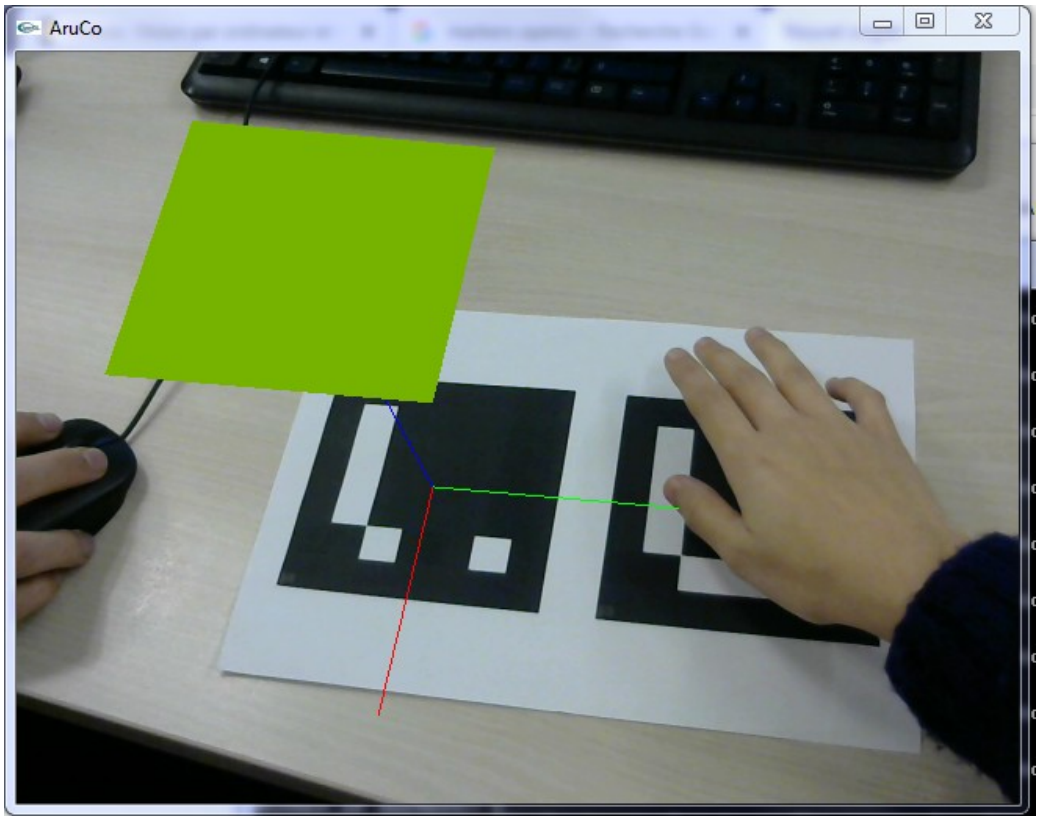

Figure 10: Montée du carré flottant lors de l'occlusion du marqueur 3

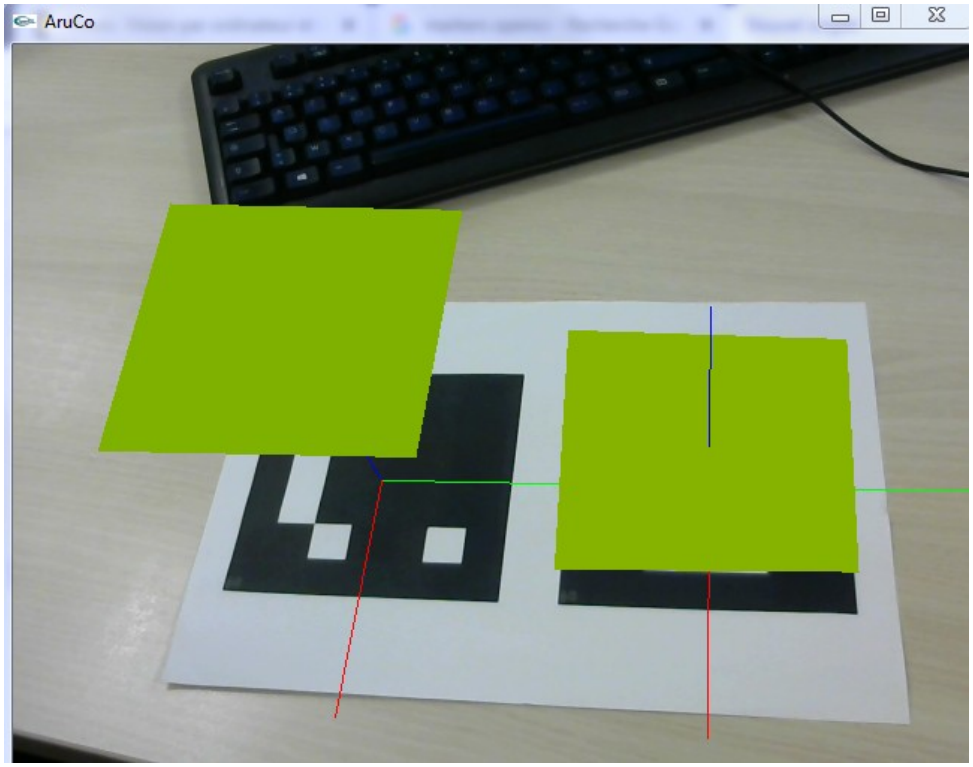

Figure 11: État de l'application en cours de jeu