Scanner 3D Volant de Bâtiments

GUO Kai
ZHANG Zhengmeng
SUN Xuan

Table des matières

1 Présentation du projet

1.1 Contexte

Ce projet a pour objectif de réaliser un scanner 3D automatique de bâtiments. Le principe est le suivant : un drone volant acquière des images HQ des différentes faces et toits d'un bâtiment. Les images prises servent à reconstruire une image 3D du bâtiment.

1.2 Domaine d'application ciblée

L'arpentage des bâtiments, la topographie, la protection des reliques culturelles.

2 Solution

Notre solution est divisé en 3 parties.

1) Première étape :
On tient un Smartphone 1 dans la main, et on met un Smartphone 2 sur le drône. On vois sur l'écran de Smartphone 1 ce que la caméra de Smartphone 2 voit avec un streaming vidéo de faible qualité. Avec le smartphone 1, on peut aussi envoyer les commandes au Smartphone 2 pour que le Smartphone 2 prenne des photos de haute qualité. Toutes ces photo de haute qualité vont être stocké dans le Smartphone 2.

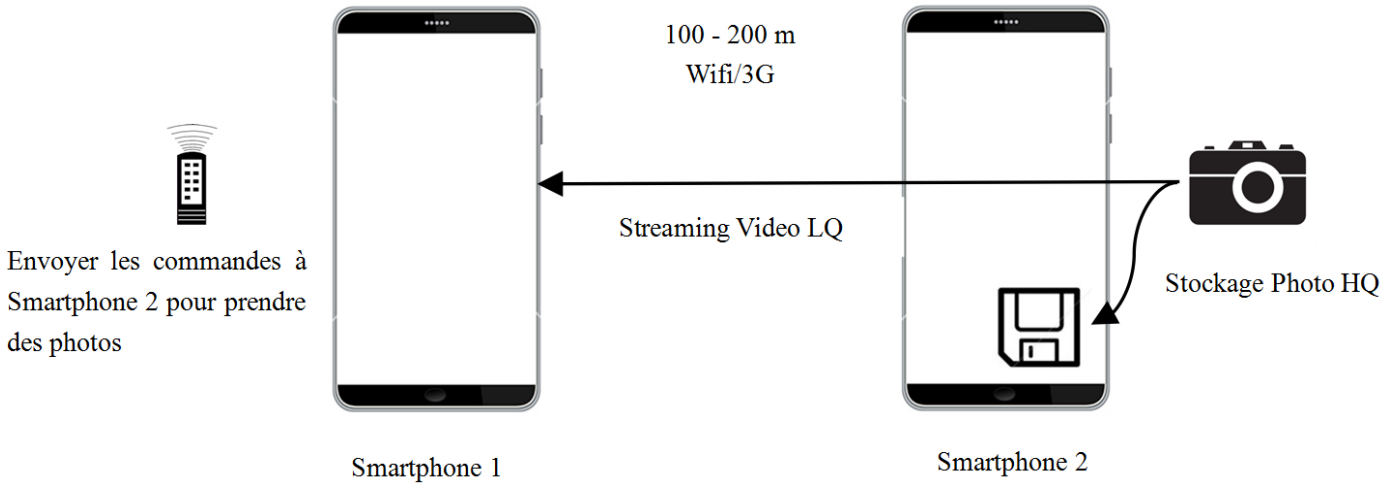

Figure 1 : shéma du fonctionnement photo

2) Deuxième étape :
Dans cette étape, avec les 2 smartphone dans l'étape 1,on ajouté an arduino. On peut envoyer des commandes pour régler l'attitude du drône depuis le smartphone 1. Ces commandes vont

d'abord être reçu par le smartphone 2. Ensuite, selon les commandes qu'il a reçu, le smartphone 2 va envoyer ses commandes a l'arduino.

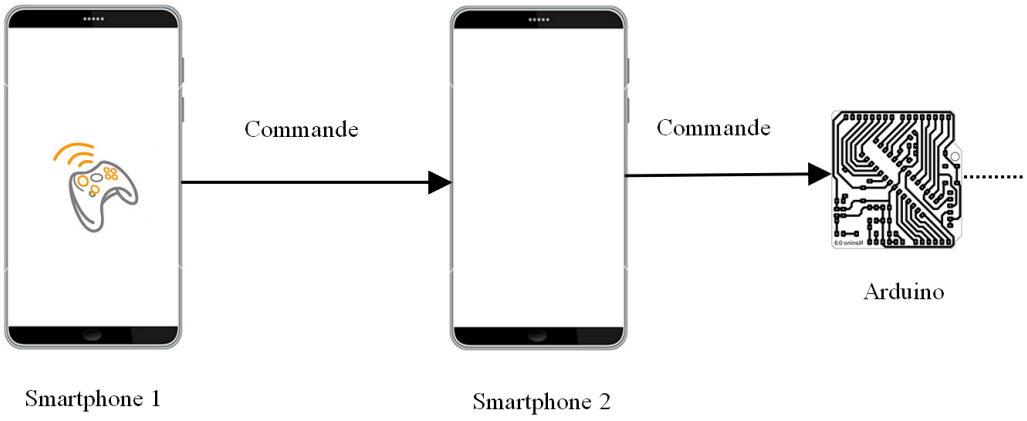

Figure 2 : shéma du fonctionnement contrôle

Ensuite, il faut joindre 1) et 2) dans une seule application.

3)Troisième étape
La étape finale est complétée en PC. Après obtenir

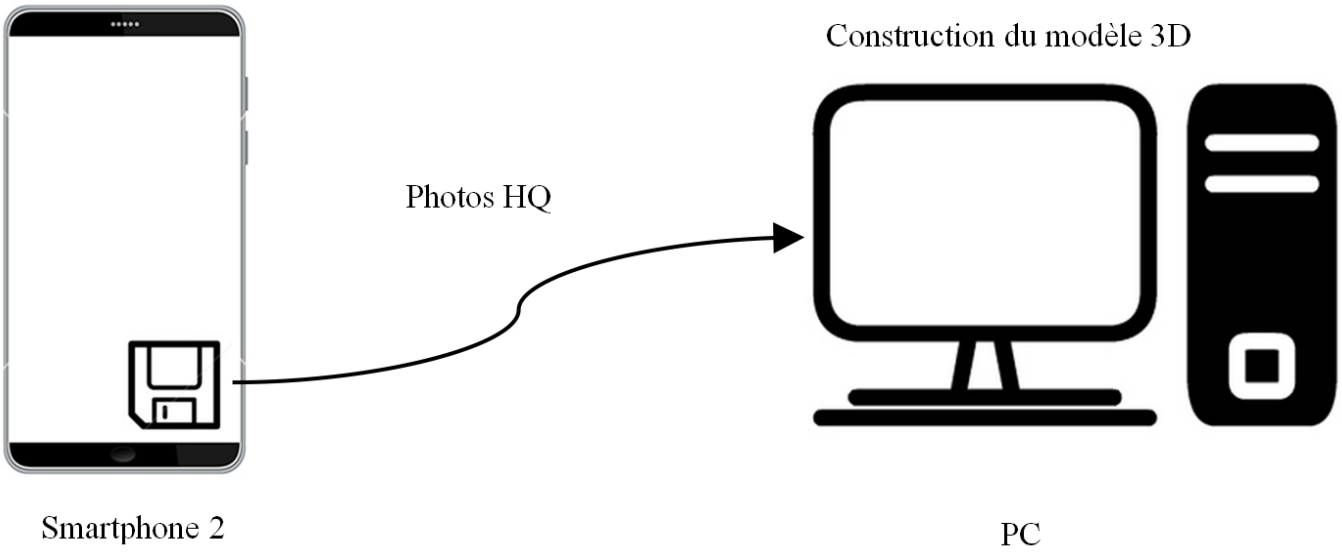

Figure 3 : shéma du fonctionnement de la construction du modèle

3 Architecture

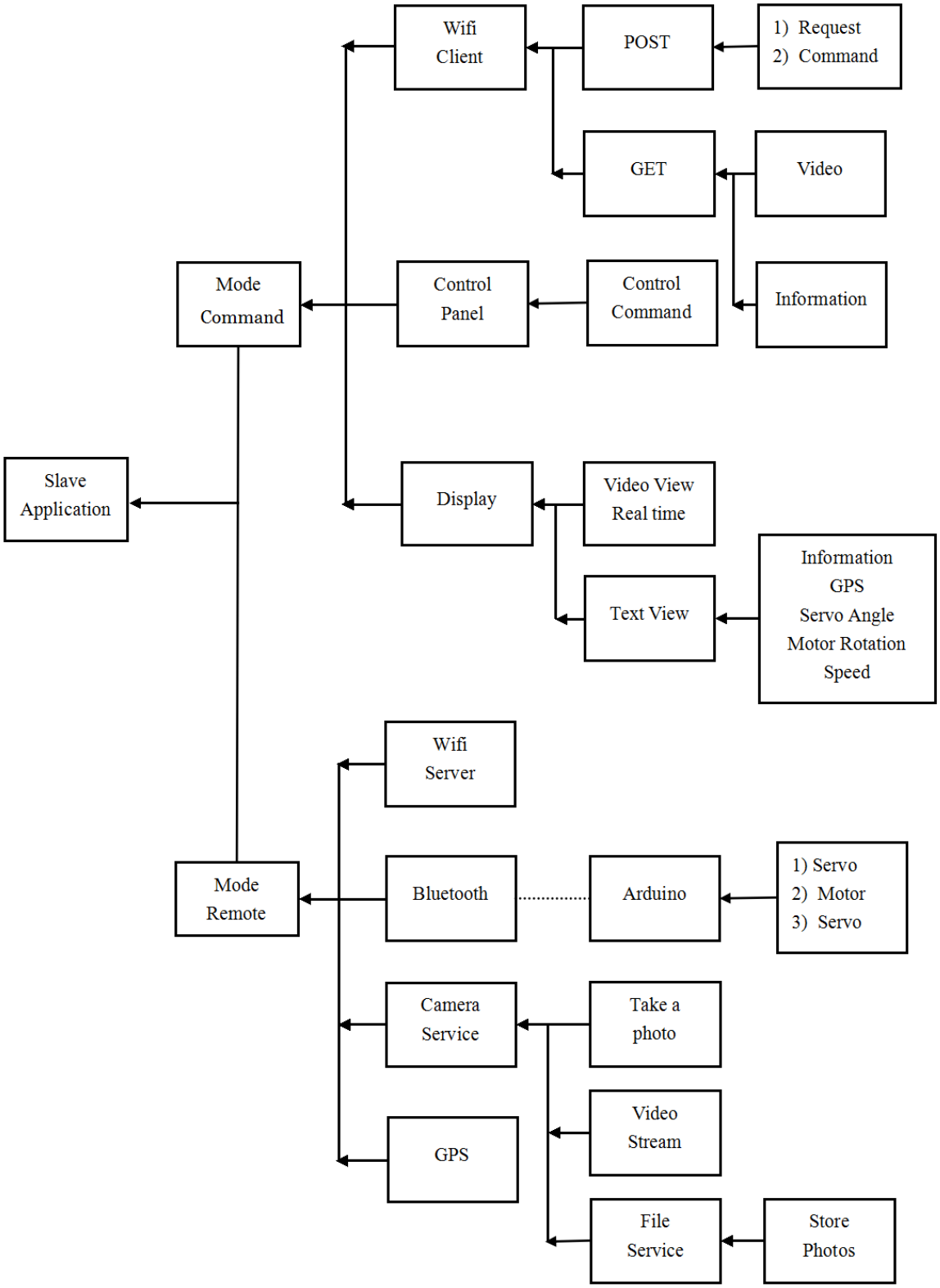

Figure 4 : diagramme du structure de l'application

4 Les Composants Logiciels et Matériels Utilisés

- Arduino uno: Arduino Uno utilise un ATmega328 comme les derniers modèles de Duemilanove, mais alors que le Duemilanove utilisait une puce FTDI pour la programmation via un connecteur USB, le Uno utilise une puce ATmega8U2 programmé comme un convertisseur série
- Pololu 6pin
- Intel az210b: portable android, version du système : 2.3.7
- Adt eclipse: le développement d'application d'Android
- Visual Studio:le développement du programme de traitement d'image
- OpenCV: une bibliothèque graphique libre, spécialisée dans le traitement d'images en temps réel.
- OpenGL:un ensemble normalisé de fonctions de calcul d'images 2D ou 3D lancé par Silicon Graphics.
- Libsdl:SDL fournit un accès de bas niveau de l'audio, clavier, souris, joystick, et le matériel graphique via OpenGL et Direct3D.
- POSIX threads:un sous-standard de la norme POSIX décrivant une interface de programmation permettant de gérer des threads.

5 Travail Réalisé

5.1 Mode Remote

Bluetooth:

Camera:

```
private void takePicOnPicMode() {
    // TODO Auto-generated method stub
    iCamera.takePicture(null, null, new PictureCallback() {

        @Override
        public void onPictureTaken(byte[] data, Camera camera) {
            // TODO Auto-generated method stub
            try {
                Bundle tempData = new Bundle();
                tempData.putByteArray("byte", data);//save picture binary data
                savePhoto(data);//save pic data into SD card
                Toast.makeText(getApplicationContext(), "Success", Toast.LENGTH_SHORT).show();
            } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
            }
        }
    });
    camIntent = new Intent("android.media.action.IMAGE_CAPTURE");
```

```java
private File savePhoto(byte[] data){
    // TODO Auto-generated method stub
    File fileFolder = new File(pic_Dir);
    if (!fileFolder.exists()) {//if this dir doesn't exist  creat
        fileFolder.mkdir();
    }
    File picFile = new File(fileFolder, get_Pic_Name());

    try {
        fileOutputStream = new FileOutputStream(picFile);
        fileOutputStream.write(data);
        fileOutputStream.close();
    } catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        return null;
    }
    return picFile;

}
```

GPS:

```java
private void startGPS() {
    // TODO Auto-generated method stub
    if (!GPS_State) {
        // TODO Auto-generated method stub
        /*
         * launch the GPS service and display in the gps tag
         */
        myGpsLocationListener.onProviderEnabled(LocationManager.GPS_PROVIDER
        launceGPS(myGpsLocationListener);
        GPS_State = true;
        Remote_Terminal_GPS.setText(R.string.Remote_State_GPS_ON);
        Remote_Terminal_GPS.setTextColor(android.graphics.Color.GREEN);
    }
}

private void stopGPS() {
    // TODO Auto-generated method stub
    myGpsLocationListener.onProviderDisabled(LocationManager.GPS_PROVIDER);
    //myLocationManager.removeUpdates(myGpsLocationListener);
    Remote_Terminal_GPS.setText(R.string.Remote_State_GPS_OFF);
    Remote_Terminal_GPS.setTextColor(android.graphics.Color.RED);
    Remote_GPS_info.setText(R.string.Remote_GPS_info_TURN_OFF);
    GPS_State = false;
}
```

5.2 Mode Command

Arduino:

```
#include <SoftwareSerial.h>
#include <Servo.h>

//definition of servo
Servo servo_1;
Servo servo_2;
//Servo servo_3;
SoftwareSerial btSerial(10, 11); // RX, TX
char buf[128];//receiving and sending buffer

void setup(){
  servo_1.attach(3);//attaching the servo_1 on pin 3
  servo_2.attach(6);
  //servo_3.attach(9);

  Serial.begin(9600);
  Serial.println("ready!");
  btSerial.begin(9600);

```

Control panel:

```
@Override
public void onClick(View v) {
    // TODO Auto-generated method stub
    switch (v.getId()) {
    case R.id.btn_play:
        startListning();
        startStreaming();
        break;
    case R.id.btn_capture:
        catchSnapShot();
        break;
    case R.id.reset:
        URGENTINTERRUPT = true;
        break;
    default:
        break;
    }
}
```

```
public boolean onTouch(View v, MotionEvent event) {
    // TODO Auto-generated method stub
    switch (v.getId()) {
    case R.id.Control_Left:
        if (!URGENTINTERRUPT) {
            try {
                onTouchSemaphore.acquire();
                while (event.getAction() == MotionEvent.ACTION_DOWN) {
                    sendHigh(R.id.Control_Left);
                }
                onTouchSemaphore.release();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }else {
            resetALL();
            break;
        }
        break;
```

5.3 Opération des fichiers

```
private void clearResource() {


/*
 * build a directory in device intern memory and put in some app
 */
private void buildResource() {

/*
 * copy files from a specified directory
 */
private void copyResourceFile(int rid, String targetFile)
```

5.4 Construction du modèle 3D

Dans cette partie, Nous mettons traitement des photos obtenus et constuire des modèles 3D. Nous utilisons un logiciel open-source *insight3d*, mais Ce logiciel nécessite une haute qualité et le bon angle en prenant des photos. Afin de répondre à nos besoins, nous avons modifié l'algorithme pour le rendre plus facile à traiter des photos de mauvaise qualité que nous avons obtenu.

Procédure de traitement d'images est divisé en les étapes suivantes:

5.4.1 Traitement des photos obtenus

- Recherche les points-clés avec l'algorithme SIFT

SIFT(Scale-invariant feature transform) c'est un algorithme utilisé dans le domaine de la vision par ordinateur pour détecter et identifier les éléments similaires entre différentes images numériques.

- Faire correspondre des points-clés
- Optimisation
- Normalisation et étalonnage des paires d'images
- Obtenir les positions, les orientations et les paramètres de caméra


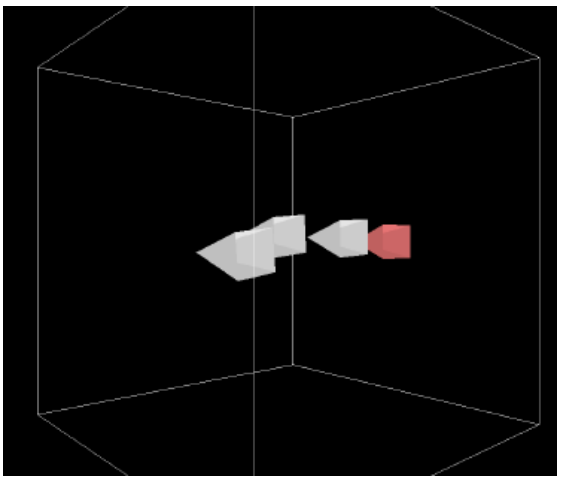

Figure 5 : les points clefs dans une photo
Figure 6 : les paramètres de caméra (angle)

5.4.2 Constuire le nuage des points 3D

- Triangulation avec l'algorithme Delaunay
- Obtenir l'angle nécessaire pour construire les modèles en 3D

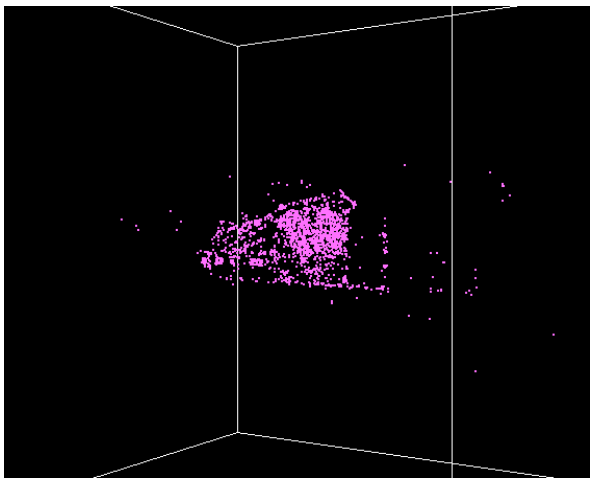
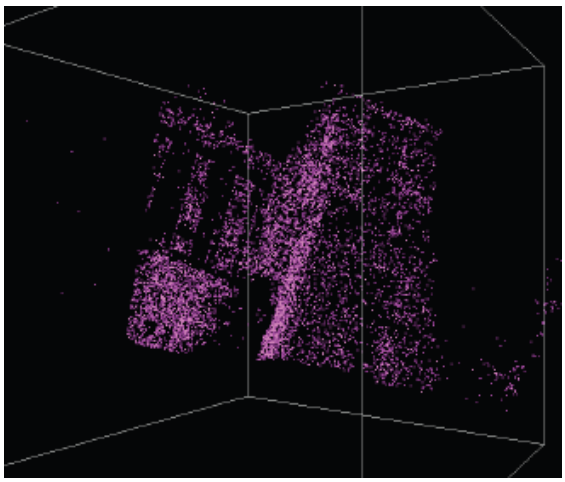

Figure 7 , Figure 8 : nuage des points 3D

5.4.3 Modélisation

- Construire manuellement des polygones
- Générer des textures

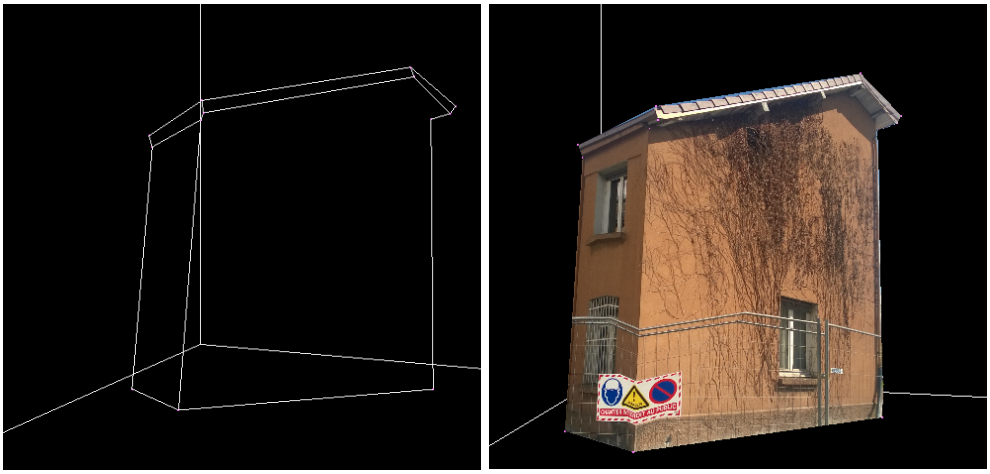

Figure 9,FIgure10 : génération des textures

6 Les Problèmes Rencontrés

- Crashes de programme après cliquer un bouton
- Oublier de définir l'activity dans androidMenifest

etc.

7 La Conclusion

Ce projet nous a permis de découvrir un certain nombre de notions. Tout d'abord nous approfondi largement nos connaissance en traitement d'image et développement Android. Ensuite, nous avons découvert comment relier une partie logicielle avec une partie matérielle.