Simon-Olivier Vaillancourt

Projet De Programmation

420-67P-SI

TP1 Partie 1

Documentation de l’environnement de développement

Travail présenté à

Martin Carignan, Martin Client

Département de l’informatique

Cégep de Sept-Îles

11 mars 2022

Table des matières

[Présentation des drones 3](#_Toc99043143)

[DJI Matrice 100 3](#_Toc99043144)

[DJI Mavic 2 Entreprise (Zoom) 4](#_Toc99043145)

[Choix du drone 5](#_Toc99043146)

[Installation d’applications 5](#_Toc99043147)

[DJI Assistant 2 (Entreprise Series) 5](#_Toc99043148)

[Android Studio 8](#_Toc99043149)

[Installation de librairies et SDK 12](#_Toc99043150)

[DJI SDK 12](#_Toc99043151)

[OpenCV 12](#_Toc99043152)

# Présentation des drones

Dans le cadre du cours, deux modèles de drone ont été présenté aux étudiants. Les deux modèles font partie de la gamme de fabrication de l’entreprise DJI, une entreprise chinoise créé en 2006 qui mène le marché de drone de loisir, professionnel et pour entreprise.

## DJI Matrice 100

Le Matrice 100 de chez DJI est un quadricoptère sortie en 2015 au prix d’environ $5000 CAD. En date d’aujourd’hui (14 mars 2022), le matrice 100 n’est plus en production et est considéré comme désuet par DJI. Cependant, cela n’empêche pas le fait que ce drone est plus que capable d’accomplir les missions de ce projet.

Voici les caractéristiques principales du DJI Matrice 100 :

|  |  |
| --- | --- |
|  |  |
| **Ordinateur de bord** | Oui (Manifold) |
| **Porté de vol (km)** | 3.5 |
| **Autonomie des batteries (min)** | 22 |
| **Capacité des batteries (mAh)** | 5700 (TB48D), 4500 (TB47D) |
| **Poids (g)** | 2355 |
| **Qualité de la caméra (Mpx)** | 12.4 |
| **Température supportées (°C)** | -10 à 40 |
| **Vitesse maximum (m/s)** | 5 |

Une image contenant périphérique

Description générée automatiquement

## DJI Mavic 2 Entreprise (Zoom)

Le Mavic 2 Entreprise de chez DJI est un quadricoptère sortie en 2018 au prix d’environ $3500 CAD.

Voici les caractéristiques principales du DJI Mavic 2 Entreprise :

|  |  |
| --- | --- |
|  |  |
| **Ordinateur de bord** | Non |
| **Porté de vol (km)** | 8 |
| **Autonomie des batteries (min)** | 30 |
| **Capacité des batteries (mAh)** | 3850 |
| **Poids (g)** | 905 |
| **Qualité de la caméra (Mpx)** | 20 |
| **Température supportées (°C)** | -10 à 40 |
| **Vitesse maximum (m/s)** | 5 |



# Choix du drone

Pour la réalisation de ce projet, j’ai choisi de travailler avec le DJI Mavic 2 Entreprise. J’ai choisi de travailler avec ce drone pour deux raisons principales. Premièrement, le Mavic 2 Entreprise est compatible avec le Mobile SDK de DJI, ce qui veut dire que je peux développer la solution sur un téléphone intelligent Android. Je possède beaucoup d’expérience sous ce type de développement et j’aime beaucoup travailler avec l’environnement Android, donc ce n’était pas un choix très difficile pour moi. Deuxièmement, j’ai trouvé que la documentation du Mobile SDK était beaucoup plus complète que celle du OnBoard SDK, la technologie utilisée par le Matrice 100. De plus, je pourrais ajouter que le Mavic possède une meilleure caméra que le Matrice 100. En revanche, cela ne sera pas vraiment utile puisque les deux drones possèdent des caméras amplement suffisantes au développement du projet. Pour finir, le Mavic 2 Entreprise est beaucoup plus petit et compact que son adversaire, ce qui apporte un petit plus à mon choix puisque le rangement du drone sera moins long et plus facile.

# Installation d’applications

Deux logiciels sont indispensables au développement du Mavic 2 Entreprise, soit DJI Assistant 2 et Android Studio. Ici, je vais expliquer le processus d’installation de ces deux applications. De plus, il est important de noter que le développement est fait à partir de Windows 10 version 21H1.

## DJI Assistant 2 (Entreprise Series)

DJI Assistant 2 est le logiciel de simulation du Mavic 2 Entreprise. Il permet entre autres de se connecter au drone et à sa mannette et de les mettre à jour. Voici le lien de téléchargement du logiciel : [DJI Assistant 2 Entreprise Series (Windows)](https://dl.djicdn.com/downloads/dji_assistant/20211015/DJI+Assistant+2+(Enterprise+Series)+V2.1.2.exe). À noter que j’utilise présentement la version 2.1.2 de DJI Assistant 2.

Une fois le logiciel téléchargé, lancez l’exécutable de l’installeur dans vos téléchargements et suivez le guide d’installation :

Une image contenant texte

Description générée automatiquement

Une image contenant texte

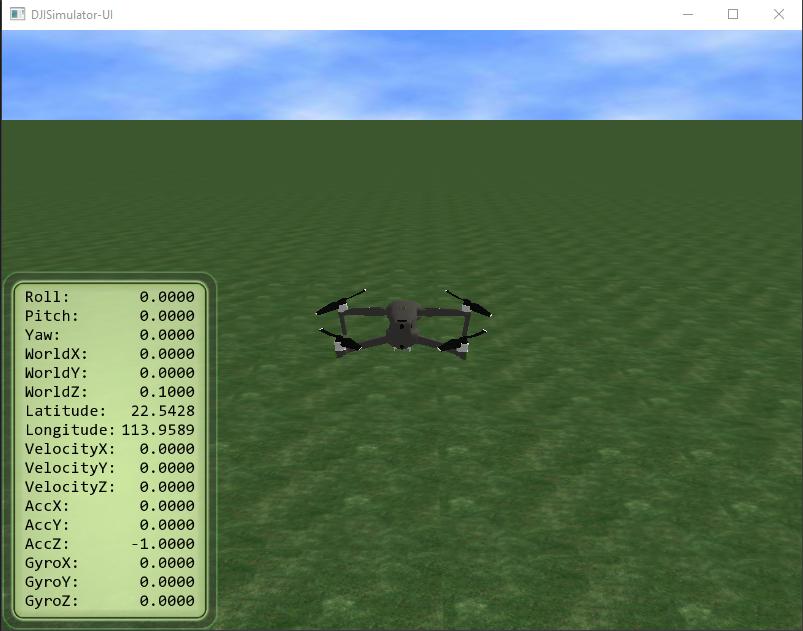
Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une fois l’installation terminée, lancez le logiciel. Assurez-vous que la manette du drone est connectée en USB à votre poste de travail, et sélectionnez-la dans DJI Assistant 2. Ensuite, dans l’onglet mise à jour, installez les dernières mises à jour disponible, cela peut prendre du temps. Maintenant, assurez-vous que le Mavic 2 Entreprise est branché en USB sur votre poste, et sélectionnez-le dans DJI Assistant 2. Encore une fois, allez dans l’onglet mise à jour et installez les derniers patchs disponibles.

Pour utiliser l’environnement de simulation de DJI Assistant 2, sélectionnez le drone et déplacez vous dans l’onglet simulation. Finalement, cliquez sur le bouton démarrer et la fenêtre de simulation apparaitra.



## Android Studio

Android Studio est un IDE qui permet de développer des applications Android en java et en kotlin. Il est nécessaire pour écrire le code, le compiler et l’installer sur la manette du drone. Voici le lien de téléchargement du logiciel : [Android Studio Bumblebee 2021.1.1 Patch 2](https://redirector.gvt1.com/edgedl/android/studio/install/2021.1.1.22/android-studio-2021.1.1.22-windows.exe). À noter que j’utilise la version Bumblebee 2021.1.1 Patch 2. La version d’Android Studio peut avoir un impact sur l’installation des dépendances du projet, il faut donc y faire attention.

Une fois le logiciel téléchargé, lancez l’exécutable de l’installeur dans vos téléchargements et suivez le guide d’installation :

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une fois l’installation terminée, vous êtes prêt à utiliser Android Studio.

# Installation de librairies et SDK

## DJI Mobile SDK

Le SDK de DJI est un outil de développement qui permet d’utiliser les fonctionnalités du drone. Il est indispensable au développement. Pour le développement du Mavic 2 Entreprise, j’utiliserais le Mobile SDK. Le Mobile SDK, comme son nom l’indique, est utilisé pour les applications mobiles (Android, IOS).

Dans ce guide d’installation, je suivrais la procédure mise en place par DJI : <https://developer.dji.com/document/76942407-070b-4542-8042-204cfb169168>. Réfèrez-vous au guide original si vous avez des interrogations.

### Création d’une application (Optionel)

Si vous avez déjà créer une application, ignorez cette étape. Pour créer une application, ouvrez Android Studio. Sélectionnez « Activité vide » et cliquez sur « Continuer ». Cliquez sur « Nouveau Projet » et entrez le nom de votre application et le nom de son package Android. Sélectionner « Java » comme language de l’application et utiliser 23 comme version de SDK Android minimum. La version de SDK Android est en lien avec la version d’Android minimum que doit avoir l’appareil qui utilisera l’application. Finalement, cliquez sur « Terminer ». Le projet est mainetant créer et vous pouvez passer à la prochaine étape.

### Configurer le script Gradle de l’application

La configuration Gradle de l’application doit être suivis à la lettre pour permettre à l’application et au Mobile SDK de DJI d’être compatible.

Voici le code à copier dans App: build.gradle :

plugins **{** id 'com.android.application'  
**}**android **{** compileSdk 30  
 buildToolsVersion '30.0.3'  
 useLibrary 'org.apache.http.legacy'  
  
 defaultConfig **{** applicationId "\*VOTRE PACKAGE ICI\*"  
 minSdk 23  
 targetSdk 30  
 multiDexEnabled true  
  
 versionCode 1  
 versionName "1.0"  
  
 ndk **{** abiFilters 'armeabi-v7a', 'arm64-v8a'  
 **}** testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  
 **}** buildTypes **{** debug **{** shrinkResources false  
 minifyEnabled false  
 proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  
 **}** release **{** shrinkResources false  
 minifyEnabled true  
 proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  
 **}  
 }** // DJI SDK  
 packagingOptions **{** doNotStrip "\*/\*/libdjivideo.so"  
 doNotStrip "\*/\*/libSDKRelativeJNI.so"  
 doNotStrip "\*/\*/libFlyForbid.so"  
 doNotStrip "\*/\*/libduml\_vision\_bokeh.so"  
 doNotStrip "\*/\*/libyuv2.so"  
 doNotStrip "\*/\*/libGroudStation.so"  
 doNotStrip "\*/\*/libFRCorkscrew.so"  
 doNotStrip "\*/\*/libUpgradeVerify.so"  
 doNotStrip "\*/\*/libFR.so"  
 doNotStrip "\*/\*/libDJIFlySafeCore.so"  
 doNotStrip "\*/\*/libdjifs\_jni.so"  
 doNotStrip "\*/\*/libsfjni.so"  
 doNotStrip "\*/\*/libDJICommonJNI.so"  
 doNotStrip "\*/\*/libDJICSDKCommon.so"  
 doNotStrip "\*/\*/libDJIUpgradeCore.so"  
 doNotStrip "\*/\*/libDJIUpgradeJNI.so"  
 exclude 'META-INF/rxjava.properties'  
 **}** compileOptions **{** sourceCompatibility JavaVersion.*VERSION\_1\_8* targetCompatibility JavaVersion.*VERSION\_1\_8* **}  
}**dependencies **{** implementation fileTree(dir: 'libs', include: ['\*.jar'])  
 implementation 'androidx.multidex:multidex:2.0.1'  
 implementation 'com.squareup:otto:1.3.8'  
  
 // DJI SDK  
 implementation('com.dji:dji-sdk:4.16', **{** exclude module: 'library-anti-distortion'  
 exclude module: 'fly-safe-database'  
 **}**)  
 compileOnly 'com.dji:dji-sdk-provided:4.16'  
  
 implementation 'com.jakewharton:butterknife:10.0.0'  
 annotationProcessor 'com.jakewharton:butterknife-compiler:10.0.0'  
 implementation 'com.google.guava:guava:29.0-android'  
 implementation 'org.apache.commons:commons-math3:3.6.1'  
 implementation 'com.davemorrissey.labs:subsampling-scale-image-view:3.10.0'  
 implementation 'io.reactivex.rxjava2:rxjava:2.2.4'  
 implementation 'io.reactivex.rxjava2:rxandroid:2.1.0'  
  
 implementation "com.google.android.material:material:1.3.0"  
 implementation 'androidx.appcompat:appcompat:1.2.0'  
 implementation 'androidx.core:core:1.0.0'  
 implementation 'androidx.constraintlayout:constraintlayout:1.1.3'  
 implementation 'androidx.recyclerview:recyclerview:1.0.0'  
 implementation 'androidx.lifecycle:lifecycle-extensions:2.0.0-rc01'  
 implementation 'androidx.annotation:annotation:1.0.0'  
 implementation 'androidx.lifecycle:lifecycle-common-java8:2.3.1'  
 implementation 'androidx.lifecycle:lifecycle-runtime:2.3.1'  
 implementation 'androidx.lifecycle:lifecycle-process:2.3.1'  
 implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1"  
  
 implementation "androidx.lifecycle:lifecycle-extensions:2.2.0"  
**}**

Après avoir modifier le fichier gradle, appuyez sur « Synchroniser le projet avec Gradle » pour appliquer les modifications et les compiler.

### Configuration du Manifest

Pour avoir accès aux fonctionnalités du Mobile SDK, il faut avoir une clé d’authentification. Pour ce faire, il faut avoir un compte DJI Developper, vous pouvez vous inscrire gratuitement ici : <https://account.dji.com/register?appId=dji_sdk&backUrl=https%3A%2F%2Fdeveloper.dji.com%2Fuser&locale=en_US>. Une fois connecté, il faut créer une clé pour votre application. Pour ce faire, dans la section « Applications » de votre centre de développeur, cliquez sur « Créer une application ». Remplissez ensuite les informations demandées et vous obtiendrez votre clé. Une fois la clé obtenue, rendez-vous dans le fichier AndroidManifest.xml de votre application Android. Insérez ceci dans le tag « application » afin de configurer votre clé développeur :

<meta-data  
 android:name="com.dji.sdk.API\_KEY"  
 android:value="VOTRE CLÉ DJI ICI"/>

Maintenant, le Mobile SDK requiert certaine permissions Android pour fonctionner correctement. Toujoujours dans AndroidManifest.xml, insérez ceci dans le tag « manifest » :

<uses-permission android:name="android.permission.BLUETOOTH" />  
<uses-permission android:name="android.permission.BLUETOOTH\_ADMIN" />  
<uses-permission android:name="android.permission.VIBRATE" />  
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  
<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />  
<uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE" />  
<uses-permission android:name="android.permission.RECORD\_AUDIO" />  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />  
  
<uses-feature android:name="android.hardware.camera" />  
<uses-feature android:name="android.hardware.camera.autofocus" />  
<uses-feature  
 android:name="android.hardware.usb.host"  
 android:required="false"/>  
<uses-feature  
 android:name="android.hardware.usb.accessory"  
 android:required="true"/>

### Création de la classe centrale

La classe centrale de l’application est une classe qui permet de centraliser des données afin de les rendre disponible partout dans l’application. En faisant un clique-droit sur le dossier « Java\\*nom de votre package\*, cliquez sur « Nouvelle -> Classe Java » et entrez le nom désiré. Faite hériter votre nouvelle classe de la classe « Application » et copiez le code suivant à l’intérieur de la classe:

@Override  
protected void attachBaseContext(Context paramContext) {  
 super.attachBaseContext(paramContext);  
 Helper.*install*(MavicMissionApp.this);  
}

### Processus d’authentification

À chaque fois qu’un usager lancera votre application, celle-ci devra s’authentifier au près de DJI avant de pouvoir utiliser le Mobile SDK. Ce processus se fera dans la classe « MainActivity.java ». Premièrement, il faut déclarer quelques variables utiles au processus d’authentification :

private static final String TAG = MainActivity.class.getName();

public static final String FLAG\_CONNECTION\_CHANGE = "dji\_sdk\_connection\_change";

private static BaseProduct mProduct;

private Handler mHandler;

Ensuite, il faut déclarer la liste des permissions requise par votre application :

private static final String[] REQUIRED\_PERMISSION\_LIST = new String[]{

Manifest.permission.VIBRATE,

Manifest.permission.INTERNET,

Manifest.permission.ACCESS\_WIFI\_STATE,

Manifest.permission.WAKE\_LOCK,

Manifest.permission.ACCESS\_COARSE\_LOCATION,

Manifest.permission.ACCESS\_NETWORK\_STATE,

Manifest.permission.ACCESS\_FINE\_LOCATION,

Manifest.permission.CHANGE\_WIFI\_STATE,

Manifest.permission.WRITE\_EXTERNAL\_STORAGE,

Manifest.permission.BLUETOOTH,

Manifest.permission.BLUETOOTH\_ADMIN,

Manifest.permission.READ\_EXTERNAL\_STORAGE,

Manifest.permission.READ\_PHONE\_STATE,

};

private List<String> missingPermission = new ArrayList<>();

private AtomicBoolean isRegistrationInProgress = new AtomicBoolean(false);

private static final int REQUEST\_PERMISSION\_CODE = 12345;

Maintenant, il faut indiquer à l’application de démarrer le processus d’authentification à la création de l’activité. Insérez ce bout de code dans la fonction « onCreate » :

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {

checkAndRequestPermissions();

}

Il faut maintenant ajouter les fonctions d’authentification:

private void checkAndRequestPermissions() {

// Check for permissions

for (String eachPermission : REQUIRED\_PERMISSION\_LIST) {

if (ContextCompat.checkSelfPermission(this, eachPermission) != PackageManager.PERMISSION\_GRANTED) {

missingPermission.add(eachPermission);

}

}

// Request for missing permissions

if (missingPermission.isEmpty()) {

startSDKRegistration();

} else if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {

showToast("Need to grant the permissions!");

ActivityCompat.requestPermissions(this,

missingPermission.toArray(new String[missingPermission.size()]),

REQUEST\_PERMISSION\_CODE);

}

}

@Override

public void onRequestPermissionsResult(int requestCode,

@NonNull String[] permissions,

@NonNull int[] grantResults) {

super.onRequestPermissionsResult(requestCode, permissions, grantResults);

// Check for granted permission and remove from missing list

if (requestCode == REQUEST\_PERMISSION\_CODE) {

for (int i = grantResults.length - 1; i >= 0; i--) {

if (grantResults[i] == PackageManager.PERMISSION\_GRANTED) {

missingPermission.remove(permissions[i]);

}

}

}

// If there is enough permission, we will start the registration

if (missingPermission.isEmpty()) {

startSDKRegistration();

} else {

showToast("Missing permissions!!!");

}

}

private void startSDKRegistration() {

if (isRegistrationInProgress.compareAndSet(false, true)) {

AsyncTask.execute(new Runnable() {

@Override

public void run() {

showToast("registering, pls wait...");

DJISDKManager.getInstance().registerApp(MainActivity.this.getApplicationContext(), new DJISDKManager.SDKManagerCallback() {

@Override

public void onRegister(DJIError djiError) {

if (djiError == DJISDKError.REGISTRATION\_SUCCESS) {

showToast("Register Success");

DJISDKManager.getInstance().startConnectionToProduct();

} else {

showToast("Register sdk fails, please check the bundle id and network connection!");

}

Log.v(TAG, djiError.getDescription());

}

@Override

public void onProductDisconnect() {

Log.d(TAG, "onProductDisconnect");

showToast("Product Disconnected");

notifyStatusChange();

}

@Override

public void onProductConnect(BaseProduct baseProduct) {

Log.d(TAG, String.format("onProductConnect newProduct:%s", baseProduct));

showToast("Product Connected");

notifyStatusChange();

}

@Override

public void onComponentChange(BaseProduct.ComponentKey componentKey, BaseComponent oldComponent,

BaseComponent newComponent) {

if (newComponent != null) {

newComponent.setComponentListener(new BaseComponent.ComponentListener() {

@Override

public void onConnectivityChange(boolean isConnected) {

Log.d(TAG, "onComponentConnectivityChanged: " + isConnected);

notifyStatusChange();

}

});

}

Log.d(TAG,

String.format("onComponentChange key:%s, oldComponent:%s, newComponent:%s",

componentKey,

oldComponent,

newComponent));

}

@Override

public void onInitProcess(DJISDKInitEvent djisdkInitEvent, int i) {

}

@Override

public void onDatabaseDownloadProgress(long l, long l1) {

}

});

}

});

}

}

private void notifyStatusChange() {

mHandler.removeCallbacks(updateRunnable);

mHandler.postDelayed(updateRunnable, 500);

}

private Runnable updateRunnable = new Runnable() {

@Override

public void run() {

Intent intent = new Intent(FLAG\_CONNECTION\_CHANGE);

sendBroadcast(intent);

}

};

private void showToast(final String toastMsg) {

Handler handler = new Handler(Looper.getMainLooper());

handler.post(new Runnable() {

@Override

public void run() {

Toast.makeText(getApplicationContext(), toastMsg, Toast.LENGTH\_LONG).show();

}

});

}

Le Mobile SDK est maintenant installé et fonctionnel. L’authentification au près de DJI se fera automatiquement au démarrage de votre application.

## OpenCV

OpenCV est une librairie open-source qui permet de faire du traitement d’image en temps réelle. OpenCV est dispobible sous plusieurs plateformes différentes, dont java.

Dans ce guide d’installation, je vais démontrer comment utiliser OpenCV dans un projet Android à partir d’Android Studio.

### Téléchargement

Pour commencer, il faut télécharger la librairie OpenCV. Pour ce faire, rendez-vous sur le site web officiel ici : <https://opencv.org/releases/>. Ensuite, sélectionnez la version d’OpenCV que vous voulez télécharger et appuyez sur « Android ».

N.B : Je recommende d’utiliser la version 4.5.2, sortit le 2021-04-02, puisque des problèmes sont susceptibles de se produire lors de l’importation dans Android Studio.

Une fois le dossier téléchargé, extrayez-le sur votre bureau.

### Importation dans Android Studio

Commencez par ouvrir le dossier que vous avez extrait sur votre bureau et copiez le dossier « sdk ». Ensuite, dirigez-vous dans le dossier de votre projet Android et collez le dossier « sdk » d’OpenCV. Ouvrez maintenant votre projet dans Android Studio. Ouvrez le fichier « settings.gradle » et insérez le texte suivant à la fin du fichier :

include ':sdk'

Sauvegardez le fichier et synchronisez les fichiers gradle. Ensuite, rendez dans « Fichier -> Structure du projet » et appuyez sur dépendances. Sélectionnez « app », appuyez sur le « + » et choissisez « Dépendance de module ». Dans la nouvelle fenêtre, cochez « sdk » et appuyez sur « OK ». Au retour dans la fenêtre de structure du projet, cliquez sur « Appliquer ». Attendez la synchronisation des fichiers gradle et retourner dans la fenêtre de structure du projet, dans l’onglet « Module ». Cliquez sur « sdk » et sélectionnez « 26 » au champ « Compile Sdk Version » et « 30.0.3 » au champ « Build Tools Version », puis cliquez sur « Appliquer ».

### Vérification de l’installation

Pour vérifier que l’installation de la librairie d’OpenCV se soit installée correctement dans votre projet Android, rendez-vous dans un fichier java d’une activité Android, puis collez le code suivant à l’intérieur de la classe :

private static final String *TAG* = "MainActivity";  
  
static {  
 if (OpenCVLoader.initDebug())  
 Log.*d*(*TAG*, "OpenCV installé correctement.");  
 else  
 Log.*d*(*TAG*, "Erreur d'installation d'OpenCV.");  
}

Si le texte « OpenCV installé correctement. » s’affiche dans les logs de votre application, l’installation a réussi et OpenCV est prêt à être utilisé dans votre application Android.