Face filters

Brad Mihai

Petrache Robert – Sorin

Introducere

Pentru acest proiect am dezvoltat aplicația Face Filter. Aceasta se vrea a fi o clona a sistemului de filtre din SnapChat. Aplicația a fost scrisa folosind Kotlin datatorită eficienței ridicate si folosește ca API pentru cameră CameraView v2 (notario1) si ML Kit din Firebase pentru detecția feței si a punctelor cheie de pe față.

Aplicația oferă posibilitatea de a alege camera folosită (front/back) si un număr de filtre.

Kotlin

În iulie 2011, JetBrains a dezvăluit proiectul Kotlin, un nou limbaj pentru JVM (Java Virtual Machine), care fusese în curs de dezvoltare timp de un an. Kotlin este un limbaj de programare cross-platform cu o eficiență ridicată a timpului de executare. Kotlin este susținut oficial de Google pentru dezvoltarea mobilă pe Andorid. De la lansarea Android studio 3.0 în octombrie 2017, Kotlin este inclus ca o alterativă la compilatorul Java standard.



Fig.1 Kotlin logo

Exemplu Hello, World! Kotlin:

// Hello, World! example

fun main(args: Array<String>) {

val scope = "World"

println("Hello, $scope!")

}

Camera View

CameraView este o bibliotecă de nivel înalt bine documentată, care facilitează capturarea imaginilor și a videoclipurilor, abordând majoritatea problemelor și necesităților comune, ce permite flexibilitate acolo unde este nevoie.

* Rapid și fiabil
* Suport gesturi
* Suport pentru procesarea cadrelor
* OpenGL preview
* Funcțiile takePicture și takeVideo ce oferă o calitate ridicată
* Instantanee super-rapide cu takePicture și takeVideo
* Dimensionare inteligentă: creați o vizualizare de tip CameraView de orice dimensiune
* Control HDR, bliț, zoom, balans de alb, expunere, locație, desen grilă și multe altele
* Funcționează de la versiunea 15 al API



Fig.2 CameraView logo

Firebase

Firebase este o platformă de dezvoltare a aplicațiilor mobile și web dezvoltată de Firebase, Inc. în 2011, achiziționată apoi de Google în 2014. Începând cu octombrie 2018, platforma Firebase are 18 produse ,care sunt folosite de 1,5 milioane de aplicații.

Firebase Inc. a obținut finanțarea pentru proiect în mai 2012. În octombrie 2014, Firebase a fost achiziționat de Google. În octombrie 2015, Google a achiziționat Divshot pentru a o îmbina cu echipa Firebase. De la achiziție, Firebase a crescut în Google și și-a extins serviciile pentru a deveni o platformă unificată pentru dezvoltatorii de dispozitive mobile. Firebase se integrează acum cu diverse alte servicii Google pentru a oferi produse și mijloace largi dezvoltatorilor. În ianuarie 2017, Google a achiziționat Fabric și Crashlytics de pe Twitter pentru a se alătura acestor servicii echipei Firebase. Firebase a lansat Cloud Firestore, o bază de date pentru documente, în octombrie 2017.

ML Kit este un kit SDK mobil care aduce experiența de învățare a machine learning Google în aplicațiile Android și iOS într-un pachet puternic, dar ușor de utilizat. Indiferent dacă sunteți nou sau experimentat în procesul de machine learning, puteți implementa funcționalitatea de care aveți nevoie doar în câteva rânduri de cod. Nu este nevoie să aveți cunoștințe profunde despre rețelele neuronale sau optimizarea modelului pentru a începe. Pe de altă parte, dacă sunteți un dezvoltator de ML cu experiență, kitul ML oferă API-uri convenabile care vă ajută să utilizați modelele personalizate TensorFlow Lite în aplicațiile dvs. mobile.



Fig.3 Firebase logo

Face Filter

Aplicația imită sistemul de filtre folosite in Snapchat.

Aceasta capteaza imagini de pe dispozitivul utilizatorului, identifica punctele cheie de pe fata folosind ML Kit din Firebase si aplică peste acestea diferite elemente grafice.

Aplicația oferă 3 filtre si posibilitatea de a comuta între camera spate si cea față.

2 dintre filtre sunt compuse din elemente pentru ochi și pentru buze, in timp ce ultimul ocupă toată fața. Aceste 3 filtre folosesc diferite puncte faciale care pot sta la baza multor variații ale imaginilor suprapuse folosite.

Filtrele sunt formate din următoarele:

* + Filtrul 1:
    - Ochelari 8 bit
    - Tigara



Fig. 4 Fitru ochelari + țigară

* + Filtrul 2:
    - Ochi falși
    - Buze false



Fig. 5 Filtru ochi + buză

* + 3.Filtrul 3:
    - Mască Batman
  + 

Fig. 6 Filtru Mască Batman

Cod aplicație

Activity\_main.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"**>  
  
 <**com.otaliastudios.cameraview.CameraView  
 android:id="@+id/camera\_view"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:adjustViewBounds="true"  
 android:keepScreenOn="true"  
 app:cameraFacing="front"** />  
  
  
 <**com.zqc.ml.snapchat.OverlayView  
 android:id="@+id/overlay\_view"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"** />  
  
 <**Button  
 android:id="@+id/flipcamerabutton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Flip Camera"** />  
  
 <**Button  
 android:id="@+id/changeFilter"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentBottom="true"  
 android:text="Filter"** />  
</**RelativeLayout**>

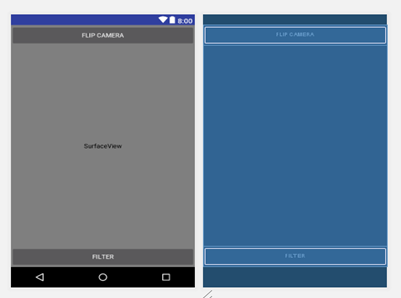


Fig. 6 Design-ul aplicației

MainActivity.kt

**package** com.zqc.ml.snapchat  
  
**import** android.content.Intent  
**import** android.content.pm.PackageManager  
**import** android.os.Bundle  
**import** android.provider.MediaStore  
**import** android.support.v4.app.ActivityCompat  
**import** android.support.v7.app.AppCompatActivity  
**import** android.view.WindowManager  
**import** com.otaliastudios.cameraview.CameraView  
**import** kotlinx.android.synthetic.main.activity\_main.\*  
**import** kotlinx.android.synthetic.main.activity\_main.view.\*  
  
  
**class** MainActivity : AppCompatActivity(), ActivityCompat.OnRequestPermissionsResultCallback {  
  
 */\*\*  
 \* The constant permission request code  
 \*/* **private val PERMISION\_REQUEST\_CODE** = 3  
  
 **override fun** onCreate(savedInstanceState: Bundle?) { *//initializeaza aplicatia* **super**.onCreate(savedInstanceState)  
 *// Make this activity full screen  
 window*.setFlags(WindowManager.LayoutParams.*FLAG\_FULLSCREEN*, WindowManager.LayoutParams.*FLAG\_FULLSCREEN*)  
  
 setContentView(R.layout.*activity\_main*)  
 checkAndRequestCameraPermission()  
  
  
  
 flipcamerabutton.setOnClickListener **{** *//schimba camera* **val** intentToActivity2 = Intent (**this**,Main2Activity::**class**.*java* )  
 startActivity(intentToActivity2)  
 finish() **}** changeFilter.setOnClickListener **{** *//schimba filtrul* **val** intentToActivity2 = Intent (**this**,Main3Activity::**class**.*java* )  
 startActivity(intentToActivity2)  
 finish() **}** }  
  
  
 */\*\*  
 \* Request for camera permission if it is needed  
 \*/* **private fun** checkAndRequestCameraPermission() {  
 **if** (ActivityCompat.checkSelfPermission(**this**, android.Manifest.permission.*CAMERA*)  
 != PackageManager.*PERMISSION\_GRANTED*) {  
 ActivityCompat.requestPermissions(**this**, *arrayOf*(android.Manifest.permission.*CAMERA*),  
 **PERMISION\_REQUEST\_CODE**)  
 } **else** {  
 startFaceProcessor()  
 }  
 }  
  
 */\*\*  
 \* Start the face processor  
 \*/* **private fun** startFaceProcessor() {  
 *// Observe activity lifecycle to start, stop and destroy camera view based on lifecycle events  
 lifecycle*.addObserver(MainActivityLifecycleObserver(camera\_view))  
  
  
  
 *// Start the face processing* **val** faceProcessor = FaceProcessor(camera\_view, overlay\_view)  
 faceProcessor.startProcessing()  
 }  
  
  
  
 */\*\*  
 \* Handle the request permission result here  
 \*/* **override fun** onRequestPermissionsResult(requestCode: Int, permissions: Array<**out** String>, grantResults: IntArray) {  
 **if** (requestCode == **PERMISION\_REQUEST\_CODE**) {  
 **if** (android.Manifest.permission.*CAMERA* == permissions[0] &&  
 grantResults[0] == PackageManager.*PERMISSION\_GRANTED*) {  
 startFaceProcessor()  
 }  
 }  
 **super**.onRequestPermissionsResult(requestCode, permissions, grantResults)  
 }  
}

OverlayView

**package** com.zqc.ml.snapchat  
  
**import** android.content.Context  
**import** android.graphics.\*  
**import** android.util.AttributeSet  
**import** android.view.View  
**import** com.google.firebase.ml.vision.face.FirebaseVisionFace  
**import** com.google.firebase.ml.vision.face.FirebaseVisionFaceLandmark  
  
  
**class** OverlayView(context: Context?, attrs: AttributeSet?) : View(context, attrs) {  
  
 *// The detected face* **var face**: FirebaseVisionFace? = **null  
 set**(value) {  
 **field** = value  
  
 *// Trigger redraw when a new detected face object is passed in* postInvalidate()  
 }  
  
 *// The preview width* **var previewWidth**: Int? = **null** *// The preview height* **var** previewHeight: Int? = **null  
  
 private var widthScaleFactor** = 1.0f  
 **private var heightScaleFactor** = 1.0f  
  
 *// The glasses bitmap* **private val glassesBitmap**: Bitmap = BitmapFactory.decodeResource(*resources*, R.drawable.*glasses*)  
 **private val ochelariBitmap**: Bitmap = BitmapFactory.decodeResource(*resources*, R.drawable.*ochelaridif*)  
  
 *// The cigarette bitmap* **private val cigaretteBitmap**: Bitmap = BitmapFactory.decodeResource(*resources*, R.drawable.*cigarette*)  
 **private val tigara2Bitmap**: Bitmap = BitmapFactory.decodeResource(*resources*, R.drawable.*tigara2*)  
 **private val candyBitmap**: Bitmap = BitmapFactory.decodeResource(*resources*, R.drawable.*candy*)  
  
 **override fun** onDraw(canvas: Canvas?) {  
 **super**.onDraw(canvas)  
  
 *// Create local variables here so they canot not be changed anywhere else* **val** face = **face  
 val** previewWidth = **previewWidth  
 val** previewHeight = previewHeight  
  
 **if** (face != **null** && canvas != **null** && previewWidth != **null** && previewHeight != **null**) {  
  
 *// Calculate the scale factor* **widthScaleFactor** = canvas.*width*.toFloat() / previewWidth.toFloat()  
 **heightScaleFactor** = canvas.*height*.toFloat() / previewHeight.toFloat()  
  
 drawGlasses(canvas, face)  
 drawCigarette(canvas, face)  
 }  
 }  
  
 */\*\*\*  
 \* Draw glasses on top of eyes  
 \*/* **private fun** drawGlasses(canvas: Canvas, face: FirebaseVisionFace) {  
 **val** leftEye = face.getLandmark(FirebaseVisionFaceLandmark.*LEFT\_EYE*)  
 **val** rightEye = face.getLandmark(FirebaseVisionFaceLandmark.*RIGHT\_EYE*)  
  
 **if** (leftEye != **null** && rightEye != **null**) {  
 **val** eyeDistance = leftEye.*position*.*x* - rightEye.*position*.*x* **val** delta = (**widthScaleFactor** \* eyeDistance / 1.5).toInt()  
 **val** glassesRect = Rect(  
 translateX(leftEye.*position*.*x*).toInt() - delta,  
 translateY(leftEye.*position*.*y*).toInt() - delta,  
 translateX(rightEye.*position*.*x*).toInt() + delta,  
 translateY(rightEye.*position*.*y*).toInt() + delta)  
 canvas.drawBitmap(**glassesBitmap**, **null**, glassesRect, **null**)  
 *//canvas.drawBitmap(ochelariBitmap, null, glassesRect, null)* }  
 }  
  
 **private fun** drawGlasses2(canvas: Canvas, face: FirebaseVisionFace) {  
 **val** leftEye = face.getLandmark(FirebaseVisionFaceLandmark.*LEFT\_EYE*)  
 **val** rightEye = face.getLandmark(FirebaseVisionFaceLandmark.*RIGHT\_EYE*)  
  
 **if** (leftEye != **null** && rightEye != **null**) {  
 **val** eyeDistance = leftEye.*position*.*x* - rightEye.*position*.*x* **val** delta = (**widthScaleFactor** \* eyeDistance / 2).toInt()  
 **val** glassesRect = Rect(  
 translateX(leftEye.*position*.*x*).toInt() - delta,  
 translateY(leftEye.*position*.*y*).toInt() - delta,  
 translateX(rightEye.*position*.*x*).toInt() + delta,  
 translateY(rightEye.*position*.*y*).toInt() + delta)  
 *//canvas.drawBitmap(glassesBitmap, null, glassesRect, null)* canvas.drawBitmap(**ochelariBitmap**, **null**, glassesRect, **null**)  
 }  
 }  
  
 */\*\*  
 \* Draw cigarette at the left mouth  
 \*/* **private fun** drawCigarette(canvas: Canvas, face: FirebaseVisionFace) {  
 **val** rightMouth = face.getLandmark(FirebaseVisionFaceLandmark.*RIGHT\_MOUTH*)  
 **val** leftMouth = face.getLandmark(FirebaseVisionFaceLandmark.*LEFT\_MOUTH*)  
  
 **if** (leftMouth != **null** && rightMouth != **null**) {  
 **val** mouthLength = ((leftMouth.*position*.*x* - rightMouth.*position*.*x*) \* **widthScaleFactor**).toInt()  
 **val** cigaretteRect = Rect(  
 translateX(leftMouth.*position*.*x*).toInt() - mouthLength,  
 translateY(leftMouth.*position*.*y*).toInt(),  
 translateX(leftMouth.*position*.*x*).toInt(),  
 translateY(leftMouth.*position*.*y*).toInt() + mouthLength  
 )  
  
 canvas.drawBitmap(**cigaretteBitmap**, **null**, cigaretteRect, **null**)  
 *//canvas.drawBitmap(candyBitmap, null, cigaretteRect, null)* }  
 }  
  
 **private fun** drawCigarette2(canvas: Canvas, face: FirebaseVisionFace) {  
 **val** rightMouth = face.getLandmark(FirebaseVisionFaceLandmark.*RIGHT\_MOUTH*)  
 **val** leftMouth = face.getLandmark(FirebaseVisionFaceLandmark.*LEFT\_MOUTH*)  
  
 **if** (leftMouth != **null** && rightMouth != **null**) {  
 **val** mouthLength = ((leftMouth.*position*.*x* - rightMouth.*position*.*x*) \* **widthScaleFactor**).toInt()  
 **val** cigaretteRect = Rect(  
 translateX(leftMouth.*position*.*x*).toInt() - mouthLength,  
 translateY(leftMouth.*position*.*y*).toInt(),  
 translateX(leftMouth.*position*.*x*).toInt(),  
 translateY(leftMouth.*position*.*y*).toInt() + mouthLength  
 )  
  
 *//canvas.drawBitmap(cigaretteBitmap, null, cigaretteRect, null)* canvas.drawBitmap(**tigara2Bitmap**, **null**, cigaretteRect, **null**)  
 }  
 }  
  
 */\*\*  
 \* Adjusts the x coordinate from the preview's coordinate system to the view coordinate system.  
 \*/* **private fun** translateX(x: Float): Float {  
 **return** *width* - scaleX(x)  
 }  
  
 */\*\*  
 \* Adjusts the y coordinate from the preview's coordinate system to the view coordinate system.  
 \*/* **private fun** translateY(y: Float): Float {  
 **return** scaleY(y)  
 }  
  
 */\*\* Adjusts a vertical value of the supplied value from the preview scale to the view scale. \*/* **private fun** scaleX(x: Float): Float {  
 **return** x \* **widthScaleFactor** }  
  
  
 */\*\* Adjusts a vertical value of the supplied value from the preview scale to the view scale. \*/* **private fun** scaleY(y: Float): Float {  
 **return** y \* **heightScaleFactor** }  
}

FaceProcessor

**package** com.zqc.ml.snapchat  
  
**import** com.google.firebase.ml.vision.FirebaseVision  
**import** com.google.firebase.ml.vision.common.FirebaseVisionImage  
**import** com.google.firebase.ml.vision.common.FirebaseVisionImageMetadata  
**import** com.google.firebase.ml.vision.face.FirebaseVisionFaceDetectorOptions  
**import** com.otaliastudios.cameraview.CameraView  
  
**class** FaceProcessor(**private val cameraView**: CameraView, **private val overlayView**: OverlayView) {  
  
 *// Initialize the face detection option, and we need all the face landmarks* **private val options** = FirebaseVisionFaceDetectorOptions.Builder()  
 .setLandmarkType(FirebaseVisionFaceDetectorOptions.*ALL\_LANDMARKS*)  
 .build()  
  
 *// Obtain the FirebaseVisionFaceDetector instance* **private val detector** = FirebaseVision.getInstance().getVisionFaceDetector(**options**)  
  
 **fun** startProcessing() {  
  
 *// Getting frames from camera view* **cameraView**.addFrameProcessor **{** frame **->  
  
 if** (frame.*size* != **null**) {  
 **val** rotation = frame.*rotation* / 90  
 **if** (rotation / 2 == 0) {  
 **overlayView**.**previewWidth** = **cameraView**.*previewSize*?.*width* **overlayView**.**previewHeight** = **cameraView**.*previewSize*?.*height* } **else** {  
 **overlayView**.**previewWidth** = **cameraView**.*previewSize*?.*height* **overlayView**.**previewHeight** = **cameraView**.*previewSize*?.*width* }  
 *// Build a image meta data object* **val** metadata = FirebaseVisionImageMetadata.Builder()  
 .setFormat(FirebaseVisionImageMetadata.*IMAGE\_FORMAT\_NV21*)  
 .setWidth(frame.*size*.*width*)  
 .setHeight(frame.*size*.*height*)  
 .setRotation(rotation)  
 .build()  
 *// Create vision image object, and it will be consumed by FirebaseVisionFaceDetector  
 // for face detection* **val** firebaseVisionImage = FirebaseVisionImage.fromByteArray(frame.*data*, metadata)  
  
 *// Perform face detection* **detector**.detectInImage(firebaseVisionImage).addOnSuccessListener **{** faceList **->  
 if** (faceList.**size** > 0) {  
 *// We just need the first face* **val** face = faceList[0]  
  
 *// Pass the face to OverlayView* **overlayView**.**face** = face  
 }  
 **}** }  
 **}** }  
}

Bibliografie

1. https://natario1.github.io/CameraView/

2. https://github.com/natario1/CameraView

3. https://firebase.google.com/docs/ml-kit/

4. https://kotlinlang.org/