Opioid Counter-Diversion project

Stage Année 4

Année universitaire 2019-2020

Nom et prénom de l'élève-ingénieur	Enzo CHOUISNARD	**Spécialité**	Robotique
Nom de l'entreprise / organisation	New York University Lab		
Ville / Pays	New York, USA		
Dates de stage	17 juin 2020 - 28 août 2020		

Remerciements

Je tiens à remercier tout particulièrement le professeur Dennis Shasha pour avoir cru en nos capacités, pour mener à bien ce projet dans le but d'une parution publique et de support de conférence. Son expérience et son enthousiasme ont servi de moteur dynamique à toute l'équipe.

Merci également à mes camarades, Caspar Lant, Simon Lebeaud et Julien Leclerc sans qui le projet n'aurait pas pu aboutir dans les délais requis. Mention toute particulière à Caspar qui, de part sa présence à New York, a pu réaliser toutes les tâches nécessitant du présentiel en cette période de COVID-19.

Merci aussi aux laboratoires, notamment le MechLab, de New York University qui nous ont autorisé à utiliser leurs imprimantes 3D afin de réaliser notre prototype.

Enfin, merci à l'ensemble de la direction de Sorbonne Université, de Polytech Sorbonne et de la Spécialité Robotique pour m'avoir permis d'effectuer ce stage, même en temps de crise sanitaire, où tout stage à l'extérieur de la zone Euro devaient être annulés.

Résumé du Projet

Le projet du professeur Shasha qui nous a été confié est la conception d'un distributeur de pilules intelligent permettant d'empêcher la revente illégale de médicament. Pour ce faire, nous avons créé un distributeur portatif décidé inviolable à moins de le casser (ce qui empêcherait le réassort en pharmacie) qui distribue une pilule lorsque l'application reconnaît le visage du patient. En effet, le dispositif est relié à une application de détection faciale. L'utilisateur doit se soumettre au test pour récupérer sa pilule. Une fois ce test réalisé, la détection continue. Nous avons ajouté plusieurs étapes, et le patient doit ingérer la pilule devant la caméra tout en montrant, à la fin, qu'il ne l'a pas caché sous sa langue ou dans sa bouche.

Pour réaliser l'application, Simon et Julien étaient les leader de cette partie. Nous avons utilisé une application sous Android Studio afin de coder la reconnaissance faciale, en nous aidant des travaux de l'année précédente. L'application est codée en Java et possède une base de donnée de visages, notamment celui du patient, permettant de valider ou non le test de reconnaissance faciale.

Pour réaliser la partie hardware, Caspar et moi même étions en charge de cette partie. Nous avons réalisé sous Autodesk Fusion 360 plusieurs CAO du système de distribution. Ayant trouvé un compromis entre frixion et facilité de codage, nous avons imprimé le dispositif grâce aux laboratoires de NYU. Nous avons imposé au système qu'il soit portatif et potentiellement rechargeable en pharmacie tout en étant inviolable sauf destruction. Le but étant de dissuader un délinquant de faire les démarches dans le but de revendre illégalement les médicaments. Pour le code, nous avons utilisé un micro-controleur Bluetooth piloté par Arduino.

Au terme de notre projet, les deux parties fonctionnent indépendamment l'une de l'autre, ce qui correspond au projet principal qui était une preuve de faisabilité. Cependant, nous n'avons pas eu le temps de relier grâce au bluetooth les deux parties et totalement supprimer l'interaction patient-distributeur pour plutôt piloter la distribution via la reconnaissance ou non de l'individu. Néanmoins, ce projet sera présenté par NYU lors de conférences données sur la place de l'automatique dans le monde de la santé dans le but d'inspirer des entreprises à poursuivre le projet en vue d'une commercialisation.

C'est ainsi que je vous transmet notre rapport scientifique et technique approuvé par le professeur Shasha. Ma fiche d'évaluation a été, quant à elle, directement transmise par le professeur à l'adresse mail présente sur le papier. Je n'y ai donc pas accès.

Countering Opioid Diversion using Machine Vision – 2020 version

ENZO CHOUISNARD - CASPAR LANT - SIMON LEBEAUD - JULIEN LECLERC - DENNIS SHASHA, New York University, USA

Opioids are substances that act on opioid receptors to relieve pain and produce a feeling of euphoria. Medically they are primarily used for pain relief, but euphoria is the primary reason they are addictive. To avoid redistribution of legally sold opioids to third parties (either addicts or criminal agents for addicts), we seek to ensure that the correct person takes each pill that is sold.

Our approach is to develop a minimally privacy-intrusive surveillance mechanism called OApp that films the taking of a pill to prevent pill diversion. The surveillance takes place only during the time starting from when the pill is removed from the dispenser to the time it has been placed in the mouth of the target patient for at least 10 seconds. At all other times, the dispenser-camera setup can be covered. The dispenser has no need to monitor sound. Generally, OApp analyzes a video each time a patient takes the pill. Specifically, the person in the video needs to be recognized, numbers on the pill need to be detected, and the pill itself needs to be tracked until it remains inside the patient's mouth for ten seconds. In addition, we need to do hand detection to ensure the patient doesn't take out the pill during the dispenser-to-mouth process.

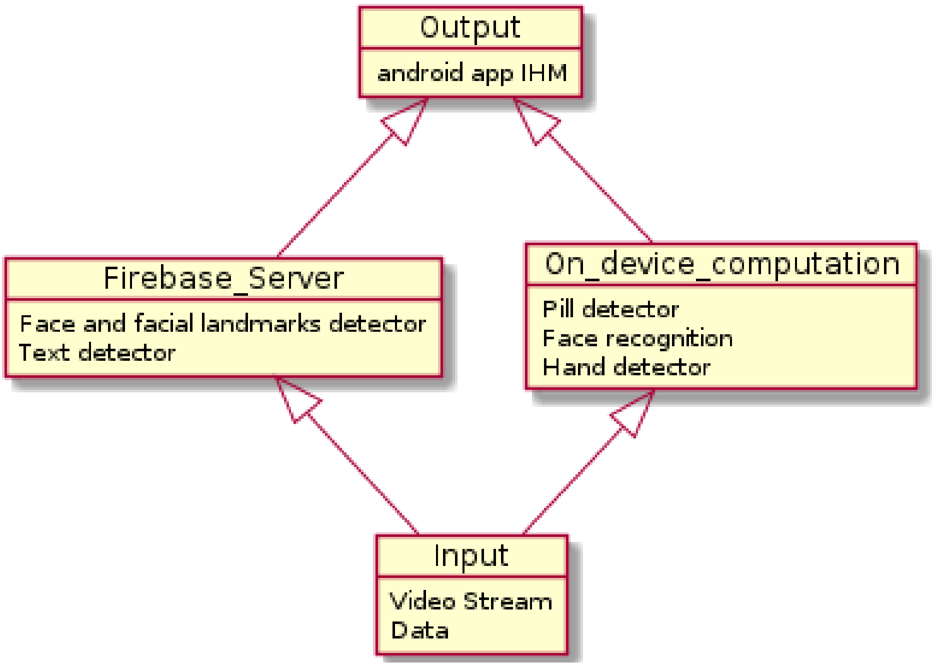
This report builds on previous work by Xiyuan Zhao, Hashim Hayat, Handi Zhang, Tairi Zheng, Shrey Jain, and advisor Dennis Shasha, which will be referred to here as Oapp 2019.

Additional Key Words and Phrases: opioid, computer vision, machine learning, face recognition, scene text recognition

Author's address: Enzo Chouisnard - Caspar Lant - Simon Lebeaud - Julien Leclerc - Dennis Shasha , New York University, USA, caspar@nyu.edu, enzo.chouisnard@gmail.com, simon.lebeaud@insa-rouen.fr, julien.leclerc@insa-rouen.fr, shasha@cims.nyu.edu.

2020. XXXX-XXXX/2020/8-ART1 $15.00
https://doi.org/

1 ARCHITECTURE



The architecture is similar to that of OAPP 2019.
There are seven major components

1. Video stream: The video stream comes in from the camera and is broken down into frames and then each frame is broken down to the pixel level for detailed analysis. In order to optimize the speed of the analysis in all processes, the application skips some of the video frames. We have seen from our phones that the video is 30 fps. Here to be able to manage processing in real-time, we process 1 frame every 20 frames.

2. Pill recognition process: This process is responsible for the recognition of the pill at various locations such as in the user's hands and or on his or her tongue. The output is the result of the detection.

3. Face recognition process: The face recognition process is responsible for detecting the patient's face and then compares the face with the profile image of the patient stored in the Firebase Storage in order to determine whether that person is authorized to take the pill (i.e. "authentication").

4. Face and landmark detector process: The face and landmark recognition process is responsible for recognizing and tracking patient landmarks such as the mouth. We used the Firebase API for face and landmark detection. This is done using cloud computing so it doesn't impact time computation very much. In output we obtain face position and/or mouth position.

5. Text detection process: At the beginning of the detection we take some frames in which we can see the pill text. We process them at the end of the detection. Each frame is sent to the Firebase Server and analysed. There is text detection followed by text recognition. The output is the result of the comparison of text between the text detected on the pill and the expected text.

6. Hand detection process: We need to detect hands in the video, to know whether the patient might be removing the pill. Hand detection is run on the mobile device. We don't control where the hand is but simply detect whether there are hands in the frame.

7. Client interface: The client interface is the IHM android app. It shows the patient face and all detection instructions. Based on the different detection results, the instructions change.

2 CORE DETECTION STEPS

-Simon Section Done/Dennis reviewed on Aug 28, 2020. If you do further modifications, please let me know.

This section gives an overview of the implementation of the recognition steps:

(1) Face verification: Analyzing how many people are in the video and recognizing the user. Throughout the process the person should be the patient and the patient only.
(2) Letter recognition: Recognizing the letters on the pill. The letters on the pill should be the same as the letters that were emitted by the dispenser. For now and for testing purposes the user inputs the three letters that are expected before any recognition is done
(3) Hand detection: Detecting whether there are hands in the frames and what they do.
(4) Pill detection: Detecting whether the pill is inside the mouth.

When running the detection, the person should satisfy some constraints:
- The user should have decent illumination.
- The background should be plain to avoid any possible false positives or false negatives,
- The sun should not be shining on the user's face, if the skin of the person is glowing it could ruin the detection. We recommend normal indoor lighting with a plain (monocolor) background.

2.1 Part One : face verification

Implementation

For the prototype, when a user connects for the first time to the application the user needs to take a picture of himself/herself. Eventually, this picture will be done by health authorities.

This image is saved on Firebase's cloud storage and is used to run face verification. The procedure is to compare the saved image to the images retrieved from the frames taken during the first phase of detection, after the pill is dispensed.

In order to run face verification we use a deep neural network model well known called FaceNet. We had to convert the keras FaceNet Model to a Tensorflow Lite model. TensorFlow Lite models (.tflite) are made to be used on smartphones and smaller devices in order to reduce computation time.

First, to verify the user's identity in a given frame, we need to detect his/her face on both the referenced image stored on the cloud and in the frame(s). Face detection is done by using the Firebase's ML kit API. If more then one face is detected on the frame, a warning is shown on screen.

Second, the model takes color images of size 160×160 in input, so once the system has detected the face, it crops the image to just the face and then reshapes the image to the right input size if needed.

Third, the image goes through the network. We than obtain the image's embedding vector of size 128.
This vector is what enables us to compare the two faces. In fact once the software FaceRecognitionDetector class obtains the embedding vector of the reference face and the face to be verified, it can compare those two vectors and determine whether the image shot from the dispenser is the intended recipient.

For now this embedding computation is done every time for the reference image, but it would be more efficient to store the embedding on the cloud.

The final step is to compare the two vectors (A and B) using cosine similarity:

$$Similarity = \frac{A \cdot B}{||A|| \cdot ||B||}$$

Once computed, the similarity gives us a result that ranges from -1 if the two vector are very dissimilar to 1 if the two vectors are the same.
Testing the face verification, depending on the hardware used, we determine that a threshold of 0.85 gives us a very accurate result.

2.2 Part Two : Face, Mouth and pill Detection

Implementation
As noted above, Oapp 2020 use cloud computing from Firebase to detect faces. This face detection API is also able to detect features such as mouth and eyes, reducing the needed computation on the phone itself.

```
**** face detection using Firebase ***
FirebaseVisionFaceDetectorOptions options =
    new FirebaseVisionFaceDetectorOptions.Builder()
            .setContourMode(FirebaseVisionFaceDetectorOptions.ALL_CONTOURS)
            .build();
FirebaseVisionFaceDetector detector = FirebaseVision.getInstance()
    .getVisionFaceDetector(options);
FirebaseVisionImage image = FirebaseVisionImage.fromBitmap(bitmap);
Task<List<FirebaseVisionFace>> result =
    detector.detectInImage(image)
            .addOnSuccessListener(
                    new OnSuccessListener<List<FirebaseVisionFace>>() \{
                        Override
                             public void onSuccess(List<FirebaseVisionFace> faces) {
                             for(FirebaseVisionFace face : faces) {
                             face.getBoudingBox();
                             }
                             }
                             });
```

As shown here, Oapp 2020 uses the option to detect All_CONTOURS. Doing so enables Oapp 2020 to get the mouth, eyes, nose, eyebrows when running the detection. This leads to the bounding box of the mouth needed for pill detection

Oapp 2020 also uses OpenCv, so for the pill detected we have kept the python implementation from Oapp 2019 and have adapted it to Android Java.

```
**** pill detection Java ***
public Mat StartPillDetection(Mat frame, int[] mouth) {

    Mat hsv_image = new Mat();
    Mat black_White_image = new Mat();

    // We transform the RGB image to the HSV image format
    Imgproc.cvtColor(frame, hsv_image, Imgproc.COLOR_BGR2HSV);

    // We create our range of white
```

```
        Scalar light_white = new Scalar(0, 0, 200);
        Scalar dark_white = new Scalar(145, 30, 255);

        // We apply the filter to our HSV image, we get a filter image (black and white)
        Core.inRange(hsv_image, light_white, dark_white, black_White_image);


        // We create a rectangle with the mouth position
        Rect roi = new Rect(mouth[0], mouth[1], mouth[2], mouth[3]);

        // We crop the image, we just want to analyse mouth area
        Mat reshape_frame = new Mat(black_White_image, roi);

        // We count number of white pixels into the mouth area
        int result = Core.countNonZero(reshape_frame);

        // Debugging message
        Log.e("Pill detection", String.valueOf(result));


        if ((result > 50)&&(result < 900)) {
            setPill_detected(true);
        } else setPill_detected(false);


        return black_White_image;
    }
```

2.3 Part Three : Hand Detection

In order to determine whether the consumption of the pill is suspicious, Oapp 2020 has to determine whether the patient has removed or swapped the pill. To do that Oapp2020 has to determine where the user's hands are. So, after the person puts the pill on his/her tongue, Oapp 2020 shouldn't detect any hand on the frame. If the person does so, Oapp 2020 tags the process as suspicious and the step is not validated.

In order to run hand detection, Oapp 2020 used a framework called MediaPipe. MediaPipe is a open-source cross-platform framework maintained and shared by Google. MediaPipe shares ML tools to facilitate the deployment of machine learning technologies into demos and applications. One of the tools is a hand and finger tracking system that can infer 21 three dimensional landmarks of a hand with only one frame.

Oapp 2020 needs to know only if a hand was visible in the frame, so in the hand tracking pipeline from MediaPipe, Oapp 2020 uses only one of the first steps which is called hand_palm detection. So Oapp 2020 used a DNN called a hand_palm model from MediaPipe in order to detect hand.

With this solution, Oapp 2020 can accurately know whether there is a hand in a given frame. But sometimes Oapp 2020 gets a false positive when the background isn't plain or if the user is moving the camera too much.

Implementation

Oapp 2020 hand detection runs on the phone itself, so the Tensorflow Lite model is stored in the assets folder of the application.

First Oapp 2020 loads the model. Than Oapp 2020 runs the input through the interpreter and gets the best result for the detection. If the output value is greater than a specified threshold, than we assume that a hand is detected

2.4 Part Four : Text Recognition

Text detection is done using Firebase's API which performs very accurately if the text on the pill is printed, as one would expect from a manufactured pill. During the dispensing and consumption process, the patient is asked to place the pill in front of his/her mouth with the letters clearly visible and hold that position for approximately 5 seconds. During those 5 seconds Oapp2020 runs text detection on five frames, one per second. This step is very dependant on the size of the letters. During testing we took very standard white pills of size 0 or 00, the text on those should be printed to be as large as the pill itself.

3 INSTALLATION

To install this code, clone the project's code from github:
https://github.com/Countering-Opioid-Diversion-using-Machine-Vision-

The application requires Android Studio. During the installation of Android Studio you should select that you want to install java android 21 SDK Version (API 21 Android 5.0 Lollipop)

Once Android studio is installed you can open the project. (File -> Open -> Select the project folder from the git you cloned)
The Open CV library is already loaded into the project.

You are not done yet, if you try running the app on you phone it won't work right now.

You will need to set up a Firebase account. Once created, try creating a Firebase android.
After this process, Firebase will give you a google-services.son file. This file needs to be placed in the *app* folder of the android application.

The project is now installed and can be built and installed on your phone.

4 RUNNING THE PROJECT

(1) You will need to put your phone into developer mode. To do so follow the steps of the following link, which also explains how to run the app:
https://javatutorial.net/connect-android-device-android-studio

(2) You can also generate an .apk file and install it on your phone. On Android Studio, go to Build -> Build Bundle(s)/ APK -> Build APK(s) After the build is done you can find the .apk file in the folder app/build/outputs/apk/debug.

5 USER INTERACTION

The patient follows a specific protocol in response to the requests from the system:

(1) Step 1: Please put the pill in front of your mouth with the letters clearly visible to the camera.
(2) Step 2: Please put the pill on your tongue, then remove your hands.
(3) Step 3: Please keep the pill on your tongue for 10 seconds with your mouth closed.
(4) Step 4: Starting the 10 seconds countdown...

(5) Step 5: Please open your mouth and show that the pill is still on your tongue.
(6) Step 6: Thank you. You have accomplished all the steps.

6 HARDWARE

6.1 Design

We designed a physical dispensing device to act in conjunction with our software platform. The hardware device pairs with a smartphone via Bluetooth Low Energy, and gives the software app confirmation that the pill has been successfully dispensed. To create the best physical dispenser we tried three different mechanisms that could dispense a pill.

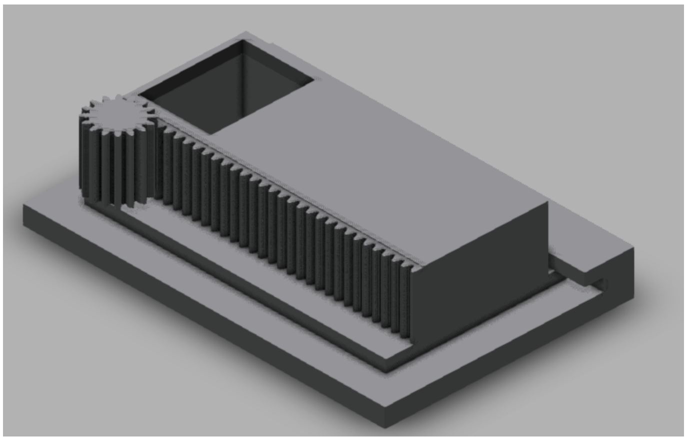


Fig. 1. First mechanism : Gear and rack mechanism

First we thought about creating a sliding box that is small enough to contain only one pill. Causing this box to go back and forth would dispense a pill while blocking the others from falling. But due to COVID-19 our printing possibilities were limited and the tolerance of the printing machines was too low to produce reliably accurate racks and gears. So this led us to another idea :

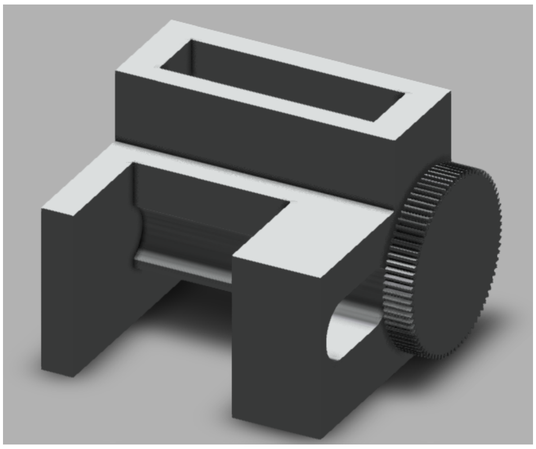


Fig. 2. Second mechanism : Barrel mechanism

The barrel mechanism was simple. A rotary barrel with a hole that fits the pill was placed inside the device. With a motor swinging the gear on the right-hand side of the figure, we can create the barrel rotation. At the back of the device there was a trap to dispense the pill. This device and the first one used the same basic idea but with the rack replaced by a gear. After printing, we realised that the rotation created a lot of friction and we decided to create another sliding mechanism but without any friction.

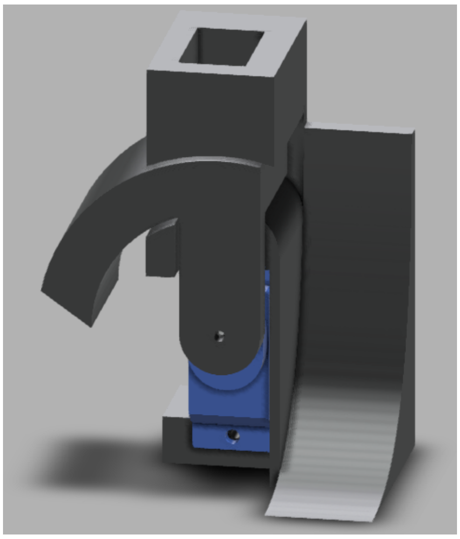


Fig. 3. Third mechanism : Sliding mechanism

With this final design, we can create a rotation thanks to a servo motor without any friction because there is a small gap between the different parts of the mechanism. The pills are stacked into the upper hole. When the device receives the order to dispense a pill, the motor rotates to the left to align the rotary hole with the upper hole. A pill falls into the rotary part and the motor rotates to the right to dispense the pill into the ramp.

6.2 Manufacturing

The manufacturing of the 3D printed parts was made by Caspar thanks to his relations with multiple labs that agreed to print our device.

6.3 Equipment used

In terms of equipment we used for the hardware part of the project we have :

(1) A breadboard and cables
(2) An external power supply (USB cable connected to computer in our case)
(3) Sub-micro Servo motor (model SG51R)
(4) Microcontroller: The Adafruit nRF52840 Father (link)
(5) Arduino IDE and a PC to send code to the card
(6) A Bluetooth-capable smartphone

(demo videos)

7 APP TO DISPENSER COMMUNICATION

We did not manage to implement the Bluetooth communication between the app and the dispense. We tried making a lite implementation of the the LE_Connect app from Adafruit (link) but the app was to complex. In the application we did implement a button for dispensing the app. This button doesn't have a purpose for now but we thought it could help for future work on the app.
The Bluetooth communication should append when this button is pressed.

8 ANDROID STUDIO

For the software we decided to design and develop an Android application using Android Studio Java.

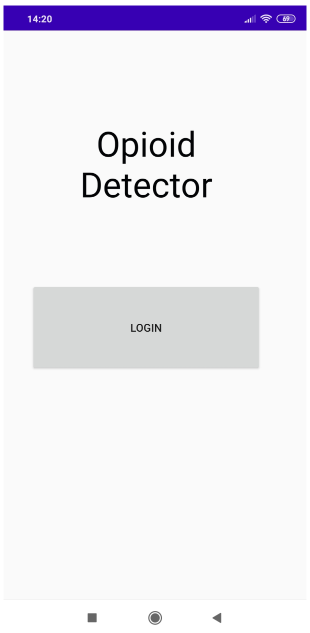
Since Oapp2019 was in Python we couldn't keep all the algorithms, but since OpenCv is available in Java we kept the algorithm to detect the pill. We also kept a similar design for the steps to follow during detection.

The authentication model for the app is based on Firebase Auth (created by google).

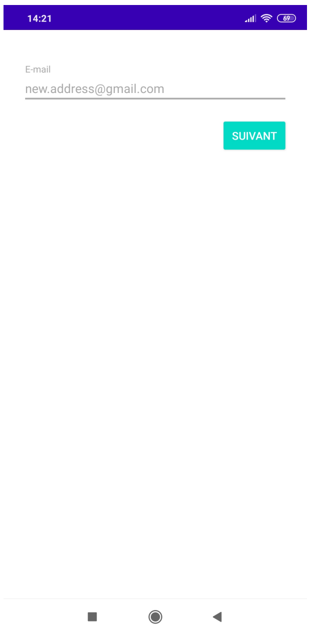
TO run the project you will have to add Firebase to your project. To do that follow this tutorial : https://firebase.google.com/docs/android/setup. Don't forget to allow the storage and the email authentication in your Firebase Account.

9 FIRST USE OF THE APPLICATION

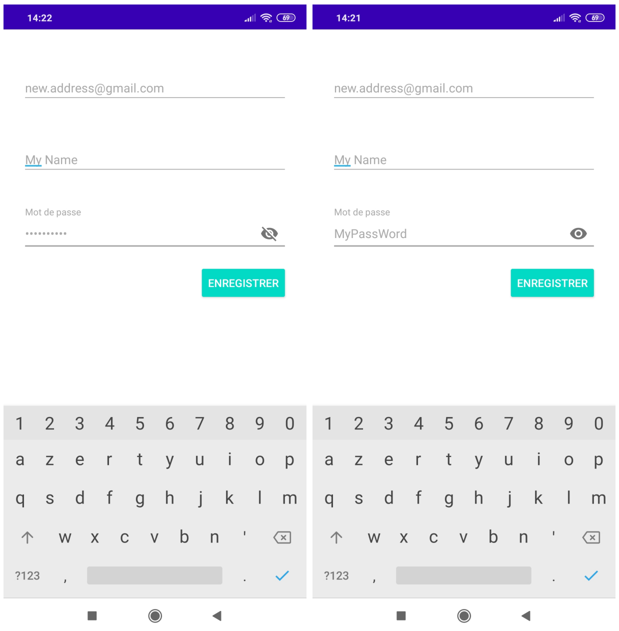
To create a new account, please tap on the Login button.



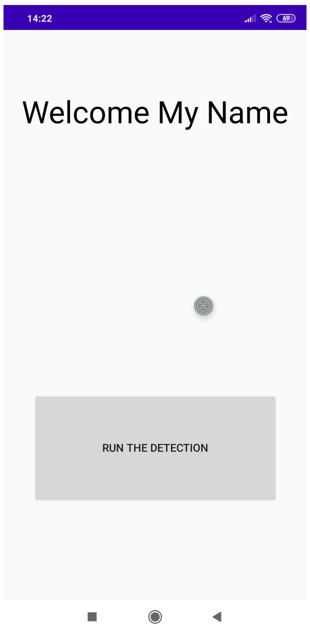
Then put your email address.

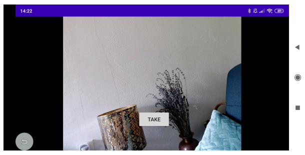


On a new page you will put in your name and your password.



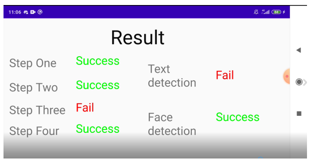
Then you will have to take your first profile image by clicking on the "plus" button.





9.1 Using the application

(1) Click on the "Run the Detection" button
(2) Put the text written on the pill on the app
(3) Then the detection activity will begin
(4) Step 1. Face verification, we want to detect that it's the right person on five frames.
Our counter is incremented when the Face Detector detects that's there is only one person and that's the right person (Account User), else the person can't pass this step.
(5) Step 2. Please put the pill in front of your mouth with the text clearly visible.
First Oapp 2020 checks that there is only one face on the screen, then Oapp 2020 gets the mouth position and checks that there is a pill in front of the mouth. Moreover, Oapp 2020 checks the presence of one hand. If there is a pill and one hand the counter is incremented (we need 5 good frames) by one.
(6) Step 3. Please put the pill on your tongue and remove your hands.
We just want to see the pill, so when a hand is detected the counter restarts.
(7) Step 4. Please close your mouth during 10 seconds.
During this step if a hand or the pill appear a Boolean named "Pill Removed" is switched to True and this step will be marked as Failed. But the patient can continue the detection, the counter restarts to 10 when there is an error.
(8) Step 5. Open your mouth and show that the pill is still on your tongue.
We have a tolerance counter that allows the patient to open his/her mouth and show the pill, if that tolerance is respected the patient followed that step well. If Oapp 2020 detects a hand during that step Oapp 2020 assumes that the pill could have been removed by the patient so this step is marked as Failed.
(9) Step 6. The app will process the frame in order to detect the text written on the pill.
(10) Result Activity. We show the patient the result of the detection.


Result

Step One	Success	Text detection	Fail
Step Two	Success		
Step Three	Fail	Face detection	Success
Step Four	Success		

subsectionApplication Operation

(1) Face Verification : FaceNetModel
(2) Face Detection : Firebase (also allows to detect mouth position)
(3) Hand Detection : HandModel
(4) Pill Detector : white color detection (on the mouth)
First we transform the RGB image to HSV.
Then we apply a filter to that new image.
Finally we count the number of white pixel on the mouth.
(5) Text detector : Firebase

subsectionRecommendations

(1) Only one person (one face) should be on the screen during the Detection
(2) Put your phone at approximately 20 cm (8 inches) (one forearm's length) from your face.
(3) Please do the detection indoors.
(4) Pay attention to the brightness. There should be no reflection.
(5) Background : monocolor (not mandatory but advisable)
(6) Text : Arial 13, bold

REFERENCES

[1] Documentation: Android developpers
https://developer.android.com/docs
[2] Documentation Firebase
https://firebase.google.com/docs
[3] Documentation: OpenCV
https://docs.opencv.org/3.4/javadoc/index.html
[4] Florian Schroff, Dmitry Kalenichenko, and James Philbin.
FaceNet: A Unified Embedding for Face Recognition and Clustering. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
[5] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang
EAST: An Efficient and Accurate Scene Text Detector. Megvii Technology Inc., Beijing, China, 2015
[6] Juan A. Figueroa, Heidy Sierra, Emmanuel Arzuaga
Real-Time Hand Detection with the use of YOLOv3. LARSIP