**0building the face surveillance system for management and recognition using opencv ON Raspberry PI**

xây dựng hệ thống giám sát, quản lý, nhận dạng khuôn mặt người sử dụng mã nguồn mở opencv trên Raspberry PI

***Trần Hồ Thủy Tiên, Nguyễn Thị Minh Hỷ, Lê Văn Cường, Nguyễn Thanh Liêm***

*Trường Đại học Bách Khoa, Đại học Đà Nẵng*

***ABSTRACT:*** *Nowadays, Internet of Things (IoT) is one of the most important elements of any future technology forecast. By the end of 2010, it is estimated that 25 billion devices will be connected to Internet, and all will be able to become “smart” when connected to the Internet. Implementing IoT will also make the management of human beings more effective, the society becomes safer and people are better protected. It is not out of the trend of using IoT technology to improve the ability to monitor and manage people effectively in the management as well as ensure security. In management, human supervision, security control, the most important thing is to identify people and identify each person’s personal information. In human identity, there is face recognition. This paper presents the direction of research, development, and building a system that allows a person to be identified face-to-face and identity information to track their appearance in the system. The application uses the open source OpenCV built-in Raspberry Pi device to collect face data, transmit to Web Service management, processing and storing data.*

***Key words:*** *IoT, OpenCV, Raspberry Pi, MongoDB, Nodej.*

***TÓM TẮT.*** *Hiện nay, Internet of Things (IoT) là một trong những yếu tố quan trọng của mọi dự báo về công nghệ tương lai. Dự báo đến cuối 2020 sẽ có 25 tỉ vật dụng có kết nối Internet, mọi vật đều có khả năng trở nên “thông minh” khi có kết nối Internet. Ứng dụng triển khai IoT cũng sẽ làm cho việc quản lý, giám sát con người hiệu quả hơn, xã hội trở nên an toàn hơn và con người ngày càng được bảo vệ tốt hơn. Không nằm ngoài xu hướng ứng dụng công nghệ IoT nhằm nâng cao khả năng giám sát, quản lý con người tạo hiệu quả trong công tác quản lý cũng như đảm bảo khả năng an ninh. Trong quản lý, giám sát con người, kiểm soát an ninh, điều quan trọng nhất là nhận dạng được con người và xác định thông tin cá nhân của mỗi người. Trong nhận dạng con người có nhận dạng khuôn mặt. Bài báo này trình bày hướng nghiên cứu triển khai và xây dựng hệ thống cho phép một người được nhận dạng khuôn mặt, ghi lại thông tin thời điểm nhận dạng và người được nhận dạng để theo dõi sự xuất hiện của họ trong hệ thống nhằm quản lý, giám sát. Ứng dụng sử dụng mã nguồn mở OpenCV tích hợp trên thiết bị Raspberry Pi để thu thập dữ liệu khuôn mặt, truyền đến Web Service quản lý, lưu trữ và xử lý dữ liệu.*

***Từ khóa:*** *IoT, OpenCV, Raspberry Pi, MongoDB, Nodejs.*



Gửi hình ảnh

Kết quả

Truy vấn

Yêu cầu

Phản hồi

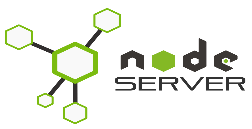


Phân tích,

Nhận dạng,

Phân loại, Thu thập.

Client



# GIỚI THIỆU

Công nghệ Internet of Things (IoT) đang trở nên phổ biến, hàng tỷ thiết bị được kết nối chung với nhau bằng Internet. Với IoT mỗi đồ vật, thiết bị, con người được cung cấp một định danh của riêng mình và tất cả có khả năng truyền tải, trao đổi thông tin, dữ liệu qua một mạng duy nhất mà không cần đến sự tương tác trực tiếp giữa người với người, hay người với máy tính [4]. Bên cạnh đó, IoT có thể triển khai một mạng lưới các thực thể thông minh, có khả năng tự tổ chức và hoạt động tùy theo tình huống, môi trường, đồng thời chúng cũng có thể liên lạc với nhau để trao đổi thông tin, dữ liệu. Với khả năng định danh cao, số lượng các thực thể trong hệ thống được định danh chính xác, duy nhất, đảm bảo tốt khả năng quản lý, điều khiển của hệ thống.

Hiện nay, việc điều khiển, giám sát các hình ảnh còn thực hiện rất thủ công nên gặp nhiều khó khăn như: tốn nhiều thời gian, chi phí, dữ liệu lưu trữ lớn. Chính vì vậy, nhu cầu cấp thiết là cần một hệ thống giám sát khuôn mặt người thông minh, tự động thu nhập dữ liệu, nhận dạng và lưu trữ. Hệ thống này nhóm tác giả tạm đặt tên hệ thống giám sát khuôn mặt người-FIDS (Face Identify System).

# THIẾT KẾ HỆ THỐNG FIDS

## Kiến trúc hệ thống

**Hình 1.** Kiến trúc hệ thống FIDS.

## Thiết bị:

### Raspberry Pi Camera Board v1.3 (5MB, 1080p):

* Cấu hình:
  + Độ phân giải 5 megapixel.
  + Hỗ trợ video 1080p30, 720p60, 640x480p90
  + Kích thước: 20 x 25 x 9 mm.
* Nhiệm vụ: Chụp ảnh gửi cho Raspberry Pi.

### Raspberry Pi 3:

* Cấu hình:
  + Broadcom BCM2837 chipset running at 1.2 GHz.
  + 64-bit quad-core ARM Cortex-A53.
  + 802.11 b/g/n Wireless LAN; 1 x 10/100 Ethernet port; Bluetooth 4.1 (Classic & Low Energy).
  + Dual core Videocore IV® Multimedia co-processor.
  + 1 GB LPDDR2 memory.
* Sử dụng hệ điều hành Raspian. [8]
* Nhiệm vụ: Xử lý ảnh nhận dạng khuôn mặt, gửi kết quả về Server.

### Express API server:

* Nhiệm vụ: Web service tiếp nhận xử lý thông tin từ Raspberry, React Client, truy vấn dữ liệu Mongodb.

### React Client

* Nhiệm vụ: Hiển thị giao diện tương tác phía người sử dụng hệ thống.

## Các thành phần chính của hệ thống

Hệ thống FIDS bao gồm các thành phần chính: Module thu nhận hình ảnh; Server xử lý hình ảnh; Client quản trị user.

Thành phần phát hiện khuôn mặt

Thành phần nhận dạng khuôn mặt

Kết nối, Gửi dữ liệu

Thành phần cập nhật khuôn mặt

Thành phần kiểm tra user

Thành phần tạo mẫu

Lưu trữ khuôn mặt

Module thu nhận hình ảnh

Server xử lý hình ảnh

Client cập nhật user

Thành phần cập nhật khuôn mặt

Thành phần kiểm tra user

Server xử lý hình ảnh

Thành phần tạo mẫu

Lưu trữ khuôn mặt

Module thu nhận hình ảnh

Thành phần phát hiện khuôn mặt

Kết nối, Gửi dữ liệu ảnh đã nhận dạng được

Thành phần thu thập khuôn mặt

Thành phần nhận dạng khuôn mặt

Client quản trị, giám sát user

**Hình 2.** Các thành phần chính của hệ thống FIDS.

### Module thu nhận hình ảnh:

Pi modul camera sẽ thu hình ảnh cho Raspberry Pi.

Raspberry Pi phát hiện khuôn mặt và tiến hành chụp ảnh để nhận dạng, nếu không nhận dạng được sẽ tiến hành tạo dữ liệu mẫu và huấn luyện cho khuôn mặt mới. Gửi hình ảnh nhận dạng về Server.

### Server xử lý hình ảnh:

Server kiểm tra hình ảnh nếu là một user chưa tồn tại thì tiến hành tạo mới user với dữ liệu mẫu hoặc là của một user đã tồn tại. Sau đó, tiến hành lưu trữ trong bộ nhớ hệ thống và trên Mongodb.

### Client quản trị user:

Client cập nhật lại user trong hệ thống, render hình ảnh của các user trong hệ thống, xoá user khỏi hệ thống cùng các hình ảnh liên quan nếu cần.

# XÂY DỰNG HỆ THỐNG

Từ kiến trúc hệ thống FIDS được thiết kế trên, chúng tôi xây dựng mô hình hệ thống FIDS như sau:

## Module thu nhận hình ảnh

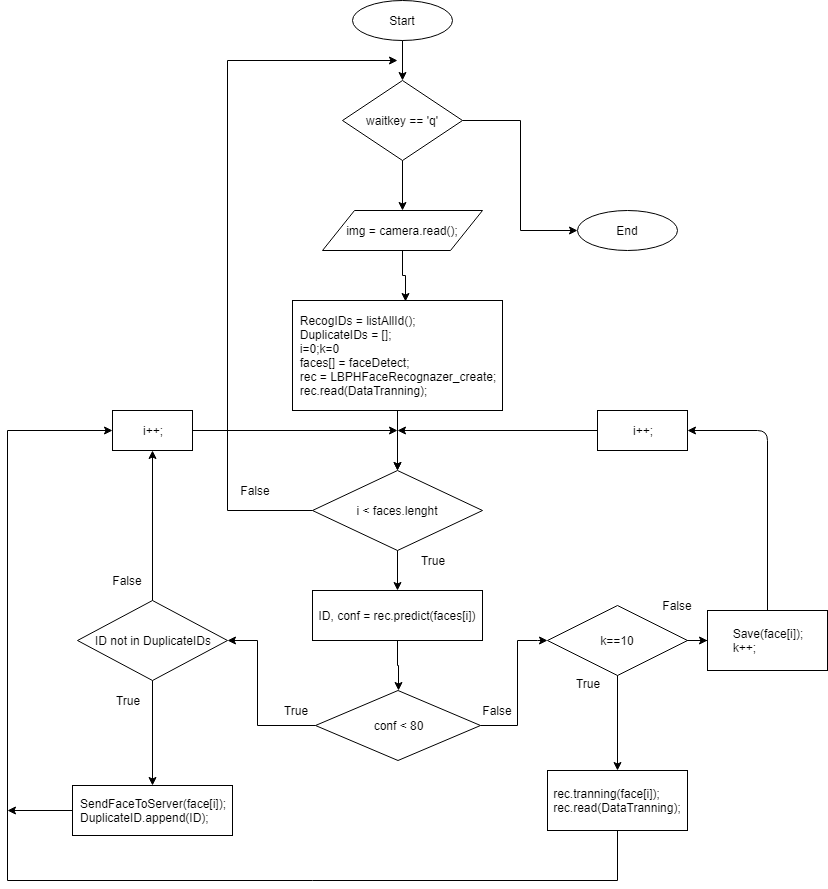
**Đầu vào**: Hình ảnh chụp được từ Raspberry Pi Camera.

**Xử lý**: Áp dụng mã nguồn mở OpenCV thực hiện các chức năng:

* + - Thu thập hình ảnh.
    - Raspberry Pi phát hiện khuôn mặt.
    - Nhận dạng khuôn mặt sử dụng thuật toán Local Binary Patterns Histograms trong OpenCV.
    - Nếu không nhận dạng được sẽ tiến hành tạo dữ liệu mẫu và huấn luyện cho khuôn mặt mới.

**Đầu ra**: Gửi hình ảnh nhận dạng được hoặc hình ảnh mới đến Server.

Sơ đồ thuật toán xử lý nhận dạng được nhúng trên thiết bị Raspberry Pi được mô tả như Hình 3.

****

**Hình 3.** Sơ đồ thuật toán xử lý nhận dạng được nhúng trên thiết bị Raspberry Pi

## Server xử lý hình ảnh

**Đầu vào**: Hình ảnh nhận được từ Raspberry Pi 3.

**Xử lý**: Kiểm tra dữ liệu hình ảnh vừa nhận được từ Raspberry:

* + - Nếu là một user chưa tồn tại thì tạo user mới với dữ liệu mẫu.

Ngược lại cập nhật thông tin khuôn mặt.

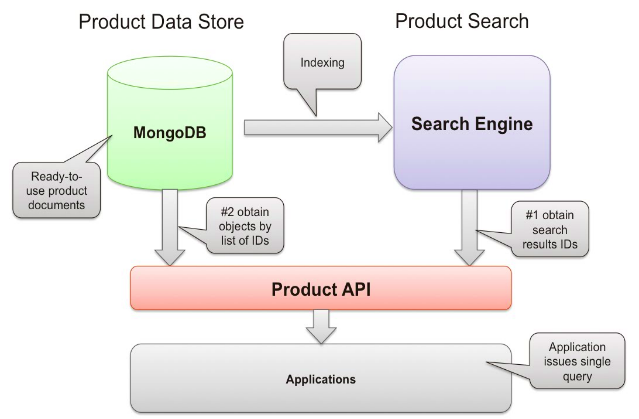
**Đầu ra**: Lưu trữ hình ảnh trong bộ nhớ hệ thống và trên MongoDB.

## Client quản trị user

Hệ thống ***Product API*** cho phép truy tìm dữ liệu hình ảnh của user trên ***MongoDB*** thông qua ***Search Engine*** trả về ID của user đã nhận dạng được cho ***Applications*** của Client như mô tả trong Hình 4.

Client hiển thị thông tin user, cập nhật lại user trong hệ thống. Tiếp tục Render hình ảnh của các user.

Cuối cùng, có thể xoá user khỏi hệ thống cùng các hình ảnh liên quan nếu cần.

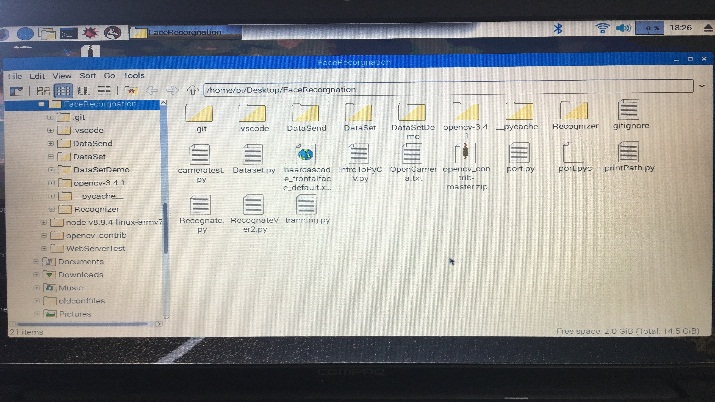


**Hình 4.** Hệ thống Product API.

# KẾT QUẢ TRIỂN KHAI THỰC NGHIỆM HỆ THỐNG FIDS

## Module thu nhận hình ảnh

Chúng tôi xây dựng hệ thống FaceRecognition gồm 3 module Dataset; Trainning; Recognition sử dụng thư viện OpenCV. Nhúng hệ thống FaceRecognition vào thiết bị Raspberry Pi 3. Chạy thử nghiệm từ thiết bị Raspberry đã nhận dạng được thành công khuôn mặt với thời gian đo được khoản trên dưới 30 mili giây cho một khuôn mặt nhận dạng.



**Hình 5.** Module FaceRecognition nhúng trên RaspberryPi.

### Module Dataset

Một số hàm OpenCV được sử dụng để tạo tập các image cho mỗi userid được chỉ ra như Hình 6. [7]

*import cv2*

*import numpy as np*

*faceDetect=cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml');*

*cap=cv2.VideoCapture(0);*

*gray=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY);*

*faces=faceDetect.detectMultiScale(gray,1.3,5);*

*cv2.imwrite('DataSet/'+str(id)+"\_"+str(sampleNum)+".jpg",gray[y:y+h,x:x+w]);*

*cv2.imshow("Face",img);*

*cap.release();*

*cv2.destroyAllWindows();*

**Hình 6.** Module Dataset.

### Module Trainning

Mã hóa khuôn mặt bằng OpenCV được sử dụng để huấn luyện khuôn mặt. [7]

*import os*

*import cv2*

*import numpy as np*

*from PIL import Image*

*rec = cv2.face.LBPHFaceRecognizer\_create();*

*path = 'DataSet';*

*def image load\_image\_file (“img.jpg”);*

*def face\_landmarks\_list face\_landmarks (image);*

*def getImagesWithID(path):*

*IDs, faces = getImagesWithID(path);*

*rec.train(faces,IDs);*

*recognizer.save('Recognizer/TrainningData.yml');*

*cv2.destroyAllWindows();*

**Hình 7: Module Trainning.**

### Module Recognition

Nhận dạng khuôn mặt sử dụng thuật toán Local Binary Patterns Histograms (LBPH) trong OpenCV. [7]

*import os*

*import cv2*

*import requests*

*import numpy as np*

*from PIL import Image*

*faceDetect=cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml');*

*cap=cv2.VideoCapture(0);*

*recognizer=cv2.face.LBPHFaceRecognizer\_create();*

*recognizer.read('Recognizer/TrainningData.yml');*

*def myface face\_endcoding (image.);*

*def bool compare\_face (myface, unknowface);*

*def checkRecog(id,conf,face);*

*def getImagesWithID(path);*

*def listAllId();*

*def sendfile();*

*def sendImageToServer(path);*

*cap.release();*

*cv2.destroyAllWindows();*

**Hình 8.** Module Recognition.

Thiết lập thiết bị kết nối với Server, chuyển dữ liệu hình ảnh đến Server thành công với thời gian truyền nhanh, đảm bảo tính an toàn dữ liệu.

## Server xử lý hình ảnh:

Hệ thống FaceRecognitionSys-Server sử dụng:

MongoDB triển khai trên mLab của Amazon Web Service (AWS) lưu cơ sở dữ liệu (CSDL) user được nhận dạng và các thông tin liên quan đến user được mô tả như Bảng 1 và hình ảnh của user ở Bảng 2.

**Bảng 1.** Bảng user được nhận dạng.

|  |  |  |
| --- | --- | --- |
| **Tên** | **Kiểu** | **Mô tả** |
| userid | String | Id của người được nhận dạng |
| fullname | String | Tên người được nhận dạng |
| email | String | Email người được nhận dạng |
| address | String | Địa chỉ người được nhận dạng |
| class | String | Lớp của người được nhận dạng, thông tin thêm |
| createAt | Datetime | Thời gian khởi tạo |
| updateAt | Datetime | Thời gian cập nhật |

**Bảng 2.** Bảng image của user được nhận dạng.

|  |  |  |
| --- | --- | --- |
| **Tên** | **Kiểu** | **Mô tả** |
| imageid | String | Id của hình đã nhận dạng |
| userid | String | Id của người được nhận dạng |
| imagepath | String | Đường dẫn đến tệp tin ảnh trên server |
| createAt | Datetime | Thời gian khởi tạo |
| updateAt | Datetime | Thời gian cập nhật |

Thiết lập Express API Server sử dụng Nodejs Server chạy trên hệ điều hành Ubuntu 16.04. Chúng tôi thiết lập Server FaceRecognitionSys trên Amazon Web Service (AWS) đồng thời cài đặt MongoDB 3.6 và kết nối đến MongoDB service để khởi tạo CSDL trên Server.

Xây dựng module image.controller trên Server gồm các hàm như Hình 9.

*const fs = require('fs');*

*const Image = require('../models/image.model');*

*const User = require('../models/user.model');*

*const imageFile = req.files.img;*

*const userId = imageFile.name.split('\_')[0];*

*const listResult = [];*

*let imageUpdate = [];*

*function getNumber(); // Lay number de tao imageid*

*function base64Encode(file); // Chuyen image thanh chuoi base64*

*function arrayDiff(a1, a2); // Lay khac nhau giua 2 array*

*exports.create(req, res); // Tao va Luu mot image moi*

*exports.listAll(req, res);*

*exports.findOne(req, res);*

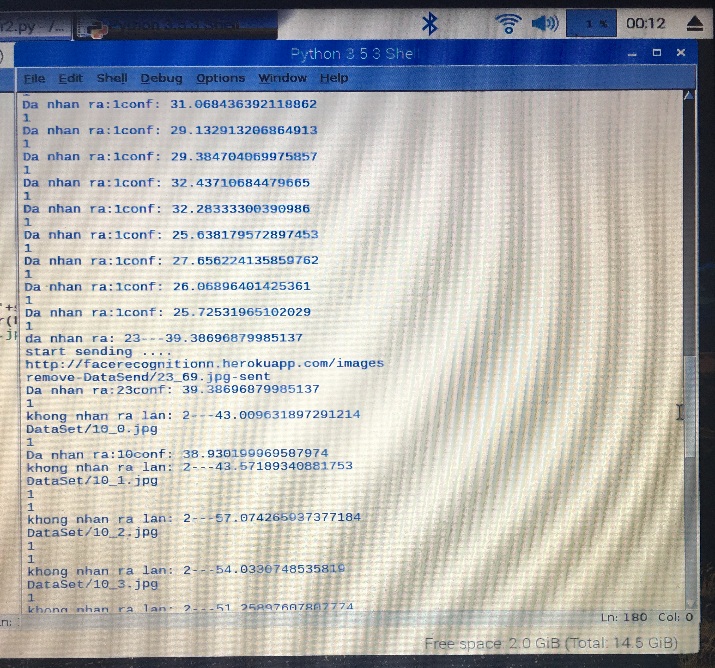
*exports.delete(req, res); // Xoa mot Image*

*exports.getDates(req, res); // Lay ngay*

*module.exports.listResult = listResult;*

**Hình 9: Module image.controller**

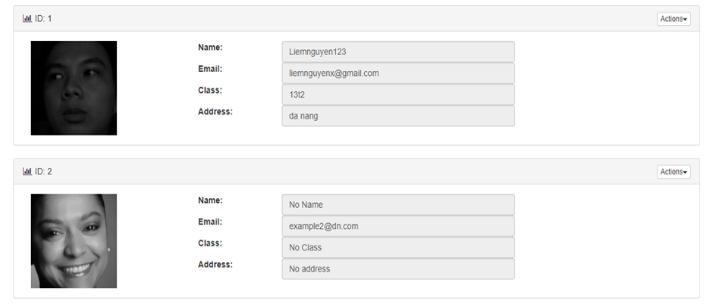
Module thu nhận hình ảnh đã nhận dạng được các ảnh ở các góc nghiêng khác nhau của user, gửi hình ảnh thành công đến FaceRecognitionSys-Server như Hình 10.



**Hình 10.** Ảnh nhận dạng Raspberry gửi đến Server thành công.

## Client quản trị user

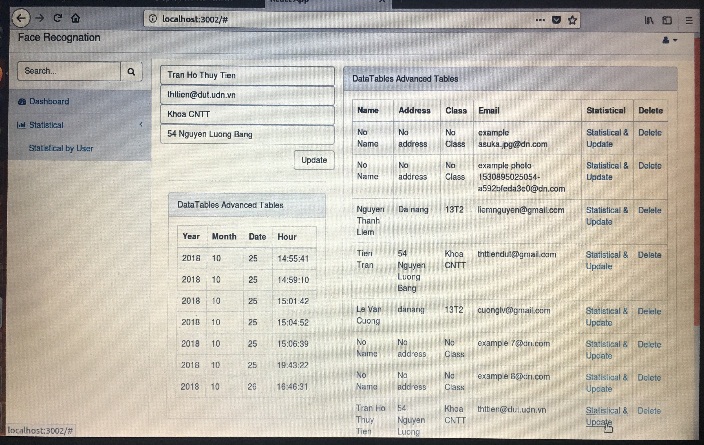
Xây dựng được ứng dụng Client có tên React-Recognition cho phép kết nối đến FaceRecognitionSys-Server, truy xuất CSDL MongoDB [3]. Sau đó, cập nhật lại user trong hệ thống, render hình ảnh của các user trong hệ thống, xoá user khỏi hệ thống cùng các hình ảnh liên quan nếu cần.



**Hình 12.** Ứng dụng Web Client cập nhật thông tin user.

Ngoài ra, hệ thống React-Recognition có thể hiển thị được thời gian hình ảnh của user gửi đến Server và lưu vào CSDL cũng như thời gian cập nhật sau cùng để phục vụ cho công việc giám sát và thống kê user.

Ứng dụng Web Client có thể quản lý, giám sát toàn bộ user được nhận dạng. Ghi lại các thời điểm user được nhận dạng gồm: giờ:phút:giây, ngày, tháng, năm như Hình 11.



**Hình 11.** Web Client quản lý, giám sát các user nhận dạng.

# NHẬN XÉT ĐÁNH GIÁ

Hệ thống giám sát, quản lý và nhận dạng khuôn mặt người FIDS có thể phát hiện, nhận dạng được khuôn mặt người từ thiết bị Raspberry Pi. Huấn luyện được một tập các dữ liệu mẫu, cứ mỗi khuôn mặt người, chúng tôi thử nghiệm với các kịch bản khác nhau:

Kịch bản 1: Chụp liên tiếp 16 ảnh mẫu với các góc chụp khác nhau để nhận dạng.

Kịch bản 2: Chụp liên tiếp 10 ảnh mẫu với các góc chụp khác nhau để nhận dạng.

Kịch bản 3: Chụp liên tiếp 8 ảnh mẫu với các góc chụp khác nhau để nhận dạng.

Kịch bản 4: Chụp liên tiếp 4 ảnh mẫu với các góc chụp khác nhau để nhận dạng.

Quá trình chạy thử với tập dữ liệu ảnh mẫu trên, đo thời gian nhận dạng khuôn mặt với nhiều lần khác nhau, chúng tôi thu được thời gian nhận dạng trung bình như mô tả ở Bảng 3.

Bảng 3. Số ảnh mẫu, tổng kích thước, thời gian trung bình.

|  |  |  |
| --- | --- | --- |
| **Số ảnh mẫu** | **Tổng kích thước (KB)** | **Thời gian nhận dạng trung bình (mili giây)** |
| 16 | 224 | 25.12931793 |
| 10 | 140 | 25.69881770 |
| 8 | 112 | 25.78834884 |
| 4 | 56 | 25.84126558 |

Với kết quả trên được mô tả bằng biểu đồ như Hình 13. Hiệu quả tối ưu của Module thu nhận ảnh là tổng kích thước lưu trữ ảnh càng nhỏ càng tốt, thời gian nhận dạng trung bình càng nhanh càng tốt. Vì vậy, chúng tôi nhận thấy như mô tả ở biểu đồ lựa chọn 8 ảnh mẫu là hợp lý.

**Hình 13.** Tối ưu số ảnh mẫu.

Cuối cùng, hệ thống sẽ chọn ra một ảnh chính rõ nhất để gửi dữ liệu đến AWS Server quản lý, lưu trữ, đồng thời ghi lại được các thời điểm user được nhận dạng.

Hệ thống Web Client truy cập đến cơ sở dữ liệu lưu trên Server để giám sát, thống kê, thu thập được toàn bộ các user đã quét khuôn mặt qua camera của thiết bị, được nhận dạng ở các thời điểm khác nhau.

# KẾT LUẬN

Bài báo trình bày kết quả nghiên cứu xây dựng một hệ thống giám sát, quản lý, nhận dạng khuôn mặt người sử dụng mã nguồn mở OpenCV. Hệ thống đã triển khai chức năng Server trên nền tảng của AWS. Việc sử dụng tài nguyên từ AWS trong hệ thống thừa hưởng được các ưu điểm và tiện ích của dịch vụ AWS như: khả năng mở rộng, khả năng chịu lỗi, khả năng lưu trữ dữ liệu, không tốn chi phí đầu tư server…đồng thời phù hợp với kiến trúc cơ sở dữ liệu MongoDB và có thể mở rộng trên hệ thống nhiều ứng dụng Client chạy song song.

Chúng tôi sẽ tiếp tục phát triển hệ thống theo hướng mô hình song song nhiều máy nhận dạng Client đặt phân tán khắp nơi. Ứng dụng này có thể mở rộng giám sát, chấm công, điều khiển các hoạt động liên quan đến nhận dạng khuôn mặt người trong khuôn viên trường học, công sở với độ chính xác cao.

TÀI LIỆU THAM KHẢO

Dr. Adrian Rosebrock, *“Computer vision and deep learning”*, <http://pyimg.co/guide>.

Armab Kumar Saha, Sachet Sircar, Priyasha Chatterjee, Souvik Dutta, Anwesha Mitra*, “A raspberry Pi controlled cloud based air and sound pollution monitoring system with temperature and humidity sensing”*, Computing and Communication Workshop and Conference (CCWC), 2018 IEEE 8th Annual, 607-611, 2018.

Dominique Guinard, Vlad Trifa, *“Building the web of things: with examples in node.js and raspberry pi”*, ManningPublications Co., 2016.

R Kumar, M Pallikonda Rajasekasan, *“An IoT based patient* and Intelligent Data Engineering (ICCTIDE), International Conference on 1-4, 2016.

Soman Kumar, Ashish Jasuja, *“Air quality monitoring system based on IoT using Raspberry Pi”*, Computing, Communication and Automation (ICCCA), 2017 International Conference on 1341-1346, 2017.

<https://www.mlab.com>

<https://www.opencv.org/>

<https://www.raspberrypi.org/downloads/raspbian>

<https://www.tutorialspoint.com/nodejs/index.htm>

Liên hệ:

ThS. Trần Hồ Thủy Tiên,

Trường Đại học Bách Khoa, Đại học Đà Nẵng.

Địa chỉ: K48/23 Ngô Sĩ Liên, Tp Đà Nẵng.

Email: [thttien@dut.udn.vn](mailto:thttien@dut.udn.vn)