1/1/2022

Darius Razvan Radu

Universitatea tehnica cluj-napoca

Proiectarea unui sistem de supraveghere cu camere Web

Contents

[1. Introducere 2](#_Toc120153390)

[1.1. Context 2](#_Toc120153391)

[1.2. Specificatii 2](#_Toc120153392)

[1.3. Obiective 2](#_Toc120153393)

[2. Studiu bibliografic 3](#_Toc120153394)

[1.1. Tehnologii folosite 3](#_Toc120153395)

[1.2. Algoritmi folositi 4](#_Toc120153396)

[2. Design 5](#_Toc120153397)

[3. Implementare 7](#_Toc120153398)

[4. Testare 8](#_Toc120153399)

[5. Concluzii 9](#_Toc120153400)

# Introducere

## Context

Scopul acestui proiect este implementarea unui sistem de supraveghere cu camere web. Sistemul va permite vizualizarea pe Internet a obiectivului urmarit si alertarea sonora in cazul in care apare o situatie neasteptata.

Scenariul ales va fi unul foarte usual, vom alerta sonor in momentul in care detectam miscare in cadru. Utilizatorul va putea seta alarma, iar in cazul in care aceasta a fost setata va alerta printr-un semnal sonor, la un interval de timp ales.

Acest proiect va putea fi folosit de orice utilizator, pentru a urmarii activitatea dintr-o anumita incapere.

## Specificatii

Proiectul va consta intr-o aplicatie Web, deci va putea fi accesata cu usurinta de oricine. Aplicatia se va conecta la camera web conectata la sistemul de calcul. In viitor, ca si o extindere, am putea adauga conexiunea la multiple camere. Aplicatia va fi dezvoltata in IDE-urile de la JetBrains si ca si limbaj de programare se va folosi Pytohn pentru detectie si HTML/CSS pentru partea de interfata Web. De asemenea vom folosi framework-ul Django pentru partea de web.

## Obiective

Proiectarea si implementarea unui sistem usor de utilizat si intuitiv pentru supervizarea unei incaperi. Pentru acest lucru, este necesarea primirii imaginii de la camere web, aplicarea logicii de detectie pe aceste cadre, iar apoi in functie de rezultat (daca s-a detectat ceva sau nu in cadru), alarmarea proprietarului printr-un semnal sonor.

De asemenea, un alt obiectiv este implementarea unei aplicatii cat mai usor si intuitiv de folosit. Acest lucru va fi realizat prin crearea unei interfete cat mai simple si cu meniuri cat mai intuitive.

# Studiu bibliografic

## Tehnologii folosite

Pentru acest proiect vom folosi cateva librarii speciale, ce ne vor ajuta sa facem detectia efectiva a obiectelor ce se afla in scena.

Una dintre aceste librarii este OpenCV.

OpenCV este o biblioteca de functii specializata pentru aplicatii de Computer Vision. Computer Vision este o ramura a Machine Learning-ului, specializata pe manipulare de imagini. Principalele caracteristici ale Computer Vision-ului sunt intelegerea imaginilor si a video-urilor, modul in care sunt stocate si manipularea acestora pentru a optine informatii din acestea. OpenCV este principala librarie folosita pentru acest domeniu. Aceasta biblioteca ofera suport pentru operatii pe imagini/video-uri in timp real, aspect foarte important pentru aplicatiile de actualitate. Pentru proiectul nostru, in care trebuie sa primim imagini/video-uri in timp real si sa le procesam pentru a optine informatii despre ce se intampla in scena, aceasta biblioteca este foarte utila.

In acelasi timp, pentru a face manipulari pe diferite date, intr-un numar atat de mare, OpenCV-ul va fi combinat cu NumPy. NumPy este, la fel, o librarie specializata pe manipulari de date, pentru limbajul Python (limbaj in care se va dezvolta aplicatia). Numpy pune la dispozitie multe structuri de date, pentru calcule complexe pe un numar mare de date, stocate in array-uri sau matrici, cat si functii pentru calcule matematice avansate.

OpenCV va fi folosit pentru manipularea imaginilor video. In principiu, vom accesa camera/camerele video conectate in aplicatie, iar cu ajutorul acestei librarii vom putea sa accesam frame-urile captate de camera video. Odata captate aceste frame-uri, ne vom indrepta atentia spre manipularea imaginilor, pentru a detecta daca exista miscare in aceste cadre.

Libraria functioneaza destul de simplu. Vom avea nevoie de modele pentru obiectele ce le dorim a fi detectate. Sper exemplu, daca dorim sa detectam un om ce se afla in scena, vom descarcat un model ce semnifica omul, iar libraria se va ocupa cu patternmatching pentru a detecta daca acel model se afla in scena sau nu.

Ca si un feature suplimentar, se va adauga in aplicatia optiunea de a selecta ce anume se doreste a se detecta in scena. Daca utilizatorul doreste sa detecteze strict oameni ce se misca in scena, acest lucur va fi posibil, dar in cazul in care dorim sa detectam si alte obiecte, utilizatorul va putea selecta acest lucru (ex. Plecam de acasa si dorim sa vedem daca animalul de companie a ajuns intr-o anumita zona nedorita a casei.)

Odata ce un obiect este detectat, vom verifica daca in frame-ul viitor acesta si-a schimbat pozitia. Acesta este cel mai simplu mod prin care putem sa verificam daca un obiect se afla in miscare sau nu. Comparand coordonatele obiectelor detectate, ne vom da seama daca ceva s-a miscat, iar daca acest lucru se intampla, vom alarma utilizatorul printr-un mesaj sonor.

Un alt pas al proiectului va fi alertarea sonora a utilizatorului in cazul in care ceva s-a miscat in frame. Pentru acest lucru vom folosi PyAudio.

PyAudio este un modul dezvoltat pentru limbajul python, ce ofera acces la porturile audio ale device-ului folosit. Cu acest modul, vom putea sa accesam porturile audio si sa trimitem orice sunet spre difuzoarele dispozitivului utilizat.

Pentru partea de interactiune web se va folosi framework-ul Django. Acest framework este special pentru programarea web in Python. Cateva informatii teoretice ar fi modul in care acesta opereaza. In principiu, pentru a lucra cu acest framework, trebuie sa ne creem in interiorul proiectului aplicatii (pentru proiectul nostru va fi doar una). Aceste aplicatii comunica direct prin protocoale HTTP/HTTPS cu serverul web.

Principalele module ale unei aplicatii in django sunt:

* Modulul de urls: acest modul contine toate url-urile din aplicatie. Acestea marcheaza fiecare locatie in care putem ajunge cu aplicatia noastra.
* Modulul de views: aceasta denumire poate partea ciudata la prima vedere. Nu este acelasi lucru cu view-urile clasice, adica interfata pe care o vede un utilizator, ci view in django se refera mai degraba la niste handles pentru aceste interfete. (un fel de controllers pentru aplicatile windows sau in alte framework-uri).
* Modulul de models: acest modul contine modelele pentru o interactiune cu baza de date, sau daca avem nevoie de anumite entitati. Dar in cazul nostru, pentru aceasta aplicatie nu vom folosi acest modul.
* Modulul de forms: acest modul este o alternativa pentru partea de interfata utilizator. In django, putem sa cream fisiere html in care sa scriem interfata utilizator clasic, in limbajul structural HTML, dar putem sa ne folosim de anumite tooluri ce ni se pun la dispozitie de django, si anume de forms. Acestea sunt niste clase in python, ce reprezinta anumite elemente basic de UI. Acestea apoi pot fi apelate/folosite in fisierele HTML, printr-un limbaj specific djago.
* Alte module importante, dar pe care nu le vom folosi sunt: admin, apps, tests, etc.

Django este un framework interesant si relativ usor de folosit pentru a face aplicatii web. Pentru ceea ce avem nevoie in proiectul acesta este suficient si mai usor de folosit django, decat alte tehnologi. Din acest considerent a fost ales.

## Algoritmi folositi

Pentru detectia obiectelor, se vor folosi o serie de metode din openCV care folosesc retele neuronale pentru detectia obiectelor in functie de niste modele prestabilite. Am descarcat un set relativ bun de modele pentru detectie. Acest model contine o serie de clase de obiecte ce pot fi detectate. Noi vom fi interesati pentru moment doar de clasa persoana. Apoi putem sa extindem aplicatia pentru detectia mai multor tipuri, in functie de ce dorim.

Pentru detectie am creat o clasa speciala Camera, ce contine o metoda dedicata pentru detectia obiectelor. Aceasta clasa de baza, va putea fi apoi folosita pentru crearea anumitor camere specifice, cu handling specific.

Detectia functioneaza, prin incarcarea acestui model cu ajutorul openCV. Apoi in urma detectiei realizate de aceasta biblioteca, ni se va returna o serie de date. Aceste date vor fi folosite pentru determinarea prezentei sau nu a unui obiect in scena. Ni se vor returna coordonate pentru obiectele detectate. In cazul in care aceste coordonate se vor modifica (deci obiectul este in miscare), se va apela o metoda din modulul de audio ce va face ca difuzorul sa faca beep.

Difuzorul va fi activat cu ajutorul bibliotecii PyAudio. Mecanismul este simplu si in cazul acestui modul. In momentul in care se va apela functia de play, modulul va incarca un fisier audio (un beep). Acest fisier va fi parcurs si fiecare stream va fi trimis catre difuzor, care va stii sa scoata sunetul din fisier.

# Design

## UML diagram

Aplicatia va fi implementata conform urmatoarei diagrame UML:

Diagram, schematic

Description automatically generated

## Interpretare diagrama

* Tamplates:

Modulul de tamplates reprezinta interfata grafica a aplicatiei. Aceasa va contine un fisier html, cu urmatoarele componente include in el: un checkbox, ce ne va spune daca alarma este setata sau nu, un buton pentru setarea alarmei si 2 campuri in care se poate vizualiza imaginea video de la 2 camere.

* Views:

Acest modul este pentru handling-ul request-urilor ce vin de la server-ul web. Contine metoda de handling pentru pagina de homepage (homepage\_view), doua metode pentru handling-ul celor 2 campuri video. Deoarece aceste imagini video, pentru a putea sa se schimbe imaginile statice, vor trebui create la un alt url, iar noi in homepage vom afisa imaginea de la acel url.

Si o functie ce se va ocupa cu trimiterea imaginilor in timp real spre url-urile camerelor video.

* Camera:

Modulul de camera contine 3 clase speciale ce se ocupa de camerele video. Aceste clase sunt: Camera, care era o clasa de baza, deci orice logica generica aplicabila oricarei camere, isi va gasi aici locul. Aici avem functii pentru desenarea de rect si detectia de obiecte.

Clasa VideoCam, ce se refera de modelarea webcam-ului de la laptop. Aceasta va avea functii pentru activarea camerei, returnarea frame-ului de la camera si o metoda speciala de update, care intr-un loop infinit va tot apela functia ce returneaza frame-ul pentru a avea imagini in timp real.

Clasa MobileCam, ce se va conecta printr-un url special la o aplicatie instalata pe telefonul mobil, iar aceasta camera ii va pune la dispozitie imaginile de la camera. La fel, avem functii specifice pentru returnarea frame-ului de la camera.

Clasa VideoCam si MobileCam, mostenesc clasa Camera.

* Audio

Acest ultim modul, se ocupa de activarea difuzorului pentru a genera un sunet de beep. Acest modul va fi activat doar in momentul in care detectia este pornita, si doar in momentul in care s-a detectat miscare.

## Diagrama use case



## Interpretare diagrama use case

Principalul actor al acestei aplicatii va fi utilizatorul aplicatiei. Acesta va putea realiza 3 actiuni simple:

Vizualizarea imaginilor venite de la camera video.

Acest lucru va putea fi facut cu usurinta pe interfata grafica. Utilizatorul va fi capabil sa vada imaginea video in timp real pe pagina web a aplicatiei.

Activarea alarmei.

Alarma va fi activata prin apasarea unui buton si dezactivata printr-o a doua apasare a butonului.

## UI design

A screenshot of a computer

Description automatically generated with medium confidence

# Implementare

* Modulul de view:

Modulul de view se ocupa cu tratarea request-urilor venite de la server. Acest modul contine 4 functii importante:

def Homepage\_view(request):

Aceasta functie se ocupa in principal cu tratarea request-urilor provenite de la pagina principala a site-ului. Pe aceasta pagina se pot vedea video-uri de la cele 2 camere (web si mobile) si un checkbox cu ajutorul caruia vom seta alarma.

Pentru a putea sa afisam video-urile online, vom crea 2 noi url-uri, la care se vor afisa imaginile provenite de la camere. Pentru crearea acestor pagini noi, si accesarea lor prin url, se vor folosii functiile:

def webcam\_feed(request) si def mobilecam\_feed(request)

Aceste 2 functii se folosesc de un helper function def gen(camera) care se ocupa de generarea video ului din multitudinea de frame uri primite de la camera. Acest video va fi trimis prin url la server ul web.

* Modulul de camera:

In acest modul avem 3 clase:

class Camera(object):

Aceasta clasa contine 4 functii:

def set\_alarmOn(self, alarmOn): un setter pentru flag-ul care va arata daca detectia e pornita sau nu.

def imencode\_frame(frame): se ocupa de resize si codarea frame ului care va fi trimis spre view pentru afisare.

def drawRect(frame, rect): se ocupa de desenarea unui dreptunghi in jurul obiectului detectat.

def detect(self, frame): functia ce se va ocupa de detectia obiectului.

Acest lucru va fi realizat folosind biblioteca OpenCV, ce ne pune la dispozitie o multitudine de metode prin care putem sa detectam obiecte cu ajutorul unor modele. Modelul folosit a fost coffeemodel.

Functia detect functioneaza astfel: in cazul in care variabila alarmOn este true, va incepe continuu detectia obiectelor din cadru. Pentru inceput, trebuie sa incarcam modelul. Vom realiza detectia cu ajutorul unor CNN-uri. Vom crea un blob, ce va fi trimis spre modelul de detectie.

Apoi folosind functia „forward”, modelul ne va detectia imaginile ce se potrivesc cu acesta. Structura returnata este una complexa, pe care o vom parsa si vom obtine coordonatele obiectelor detectate. Dupa ce avem aceste coordonate, vom desena un dreptunghi pe ecran si vom anunta cu un semnal sonor.

Tind sa mentionez ca detectia se va face pe ambele camere, deci imaginea de la ambele camere va fi trimisa spre modulul de detectie.

class VideoCamera(Camera):

Aceasta clasa reprezinta camera video a laptopului. Aceasta clasa se va ocupa de orice are nevoie camera web: de conectarea acesteia, la preluarea frame-urilor de la ea, iar detectia va fi facuta de clasa parinte.

class MobileCamera(Camera):

Aceasta clasa reprezinta camera de la telefonul mobil. Aceasta clasa se va ocupa de orice are nevoie camera telefonului: de conectarea prin URL, preluarea frame-urilor, iar detectia va fi facuta de clasa parinte.

* Modulul de audio:

Pentru a anunta sonor utilizatorul ca un obiect a fost detectat, am folosit biblioteca pyAudio. Aceasta biblioteca ne ajuta sa incarcam fisiere audio, care apoi pot fi procesate de difuzorul device-ului pe care ruleaza aplicatia.

* Modulul de templates:

Acest modul contine fisiere HTML care sunt folosite pentru a realiza site ul web. Acest site web este compus din 2 campuri pentru afisarea imaginilor (imagini primite in timp real de la camere sub forma de frame uri), un checkbox care va zice daca alarma este pornita si un buton pentru activarea/dezactivarea acesteia. Checkbox ul este folosit cu ajutorul unor clase din framework-ul django. Am creat o clasa frame, care va contine ca element un booleanField. Acest field va fi reprezentat pe view ca si un checkbox.

# Testare

Avand in vedere ca proiectul este orientat spre ramura de computer vision, am incercat sa folosesc diferite modele pentru a detecta obiectele in miscare. Incercand mai multe modele am observat ca performanta si precizia detectiei depinde cel mai mult de complexitatea modulului de invatare.

Pentru acest proiect am folosit algoritmi implementati in librari din OpenCV, deci nu am avut acces foarte mare la parametrii de learning ai retelelor neuronale. Astfel precizia detectiei prototipului nostru este data strict de detalierea modelului si de lumina ambientala la care este supus.

Testarea modelului a decurs prin punerea sub stres a prototipului, cu multe miscari repetate in medii cu lumina puternica sau proasta. Modelul nu se descurca bine in medii cu lumina puternica, acest lucru fiind cel mai mare impediment in timpul testarii.

Detectia a fost impiedicata de multe ori si de viteza de procesare a aplicatiei. Avand in vedere ca am implementat prototipul ca o aplicatie web, serverul limiteaza foarte mult puterea de procesare a retelei.

Testarea a fost facuta pe 2 dispozitive (camera web si camera telefonului mobil). Ambele camere reusesc sa perceapa imaginea si sa detecteze anumite obiecte, dar uneori daca sunt foarte multe miscari in cadru, dureaza o perioada pana la auzirea semnalului sonor.

In ceea ce priveste aplicatia web, imaginea se vede bine, cu un delay suportabil, iar timpul de raspuns a serverului este relativ acceptabil. La fel ca in cazul modulului de detectie, am testat afisarea camerei punand aplicatia sub stres continuu pentru a testa delay-ul camerelor si a server-ului.

# Concluzii

In concluzie, aplicatia are ca scop supravegherea unui obiectiv si detectarea, daca se doreste, a anumitor miscari din cadru, lucru realizat cu ajutorul unui modul de procesare de imagini implementat cu biblioteca OpenCV si limbajul Python.

Aplicatia este una simpla ce ofera functionalitatea dorita. Modulul de detectie al obiectelor functioneaza in majoritatea cazurilor si sub conditii favorabile de lumina. Scopul educativ al proiectului a fost de a introduce cunostinte de baza de computer vision si de a dezvolta folosirea unui framework de web.

Consider ca obiectivele proiectului au fost atinse, iar aplicatia este un prototip bun, ce poate fi impunatatit prin folosirea unor modele mai exacte sau chiar implementarea unui mecanism mult mai performant de detectie de obiecte.