|  |  |  |
| --- | --- | --- |
| O imagine care conține Grafică, simbol, design grafic, artă  Descriere generată automat | **Departamentul Automatică şi Informatică Industrială**  **Facultatea Automatică şi Calculatoare**  **Universitatea Nationala de Stiinta si Tehnologie**  **POLITEHNICA din Bucureşti**  Splaiul Independenţei 313, 060042, Bucureşti, România  Sala ED 412, Tel. 021/402.92.69  [www.aii.pub.ro](http://www.aii.pub.ro), email: secretariat.aii@upb.ro |  |

|  |  |
| --- | --- |
|  |  |

**Sesiunea de Comunicări Şiinţifice Studenţeşti**

**Ediţia 2024**

**Detectie de obiecte in fluxuri video**

**Autor: Mihai Violeta, ACS, 2024, 333AA**

**Adresa e-mail : violeta.mihai@stud.acs.upb.ro**

**Îndrumător ştiinţific: Conf. Dr. ing. Stefan Alexandru Mocanu**

**Cuprins:**

[1. Introducere ………………………………………….](#_Toc284253286) 3

2. Suport tehnic ………………………………………. 3

[3. Etapa de implementare …………………………](#_Toc284253301) 5

[4. Mod de utilizare …………………………………..](#_Toc284253305) 13

[5. Concluzii ……………………………………………](#_Toc284253313) 15

[6. Bibliografie …………………………………………](#_Toc284253320) 16

1. Introducere

* Prezentare a temei

Proiectul porneste de la ideea detectiei de obiecte intr-un flux video si incearca sa ajunga la o aplicatie utila a acestui concept foarte popular in computer vision: o camera de supraveghere care trimite notificari proprietarului atunci cand identifica anumite obiecte in incapere. Intr-o asemenea aplicatie este util ca si utilizatorul sa poata avea acces la stream.

Ca si componente, sunt folosite: ESP32-Cam pentru achizitia de imagini, Arduino UNO R3, un motor de tip stepper si un driver (pentru controlul pozitiei camerei).

* Obiective:

Lucrarea de fata isi propune sa ofere o solutie fezabila (mai ales din perspectiva software) pentru tema de mai sus. Acest lucru presupune o interfata grafica minimala + un mod de utilizare usor de inteles, chiar si in lipsa instructiunilor.

1. Suport tehnic

* Algoritmul YOLOv3

Desi initial luam in considerare sa incerc o solutie proprie pentru un algoritm de detectie de obiecte (bazata probabil pe clasificatori), antrenat pe un set de date personalizat, am renuntat la idee, considerand ca nu face subiectul acestui proiect. In schimb, am explorat cativa dintre algoritmii poluari de object detection disponibili. Urmeaza a fi prezentate cele mai relevante idei legate de alegerea algoritmului.

O clasificare populara a algoritmilor de computer vision este bazata pe numarul de parcurgeri ale unei imagini pentru a identifica obiectele din cadrul ei. Astfel, exista algoritmi One-Stage si Two-Stage, in prima categorie aflandu-se cei care proceseaza o imagine o singura data inainte sa faca predictii legate de aceasta, sacrificand acuratetea in ceea ce priveste obiectele mici, in schimbul vitezei si a complexitatii computationale reduse [[1](https://www.v7labs.com/blog/yolo-object-detection)]. In a doua categorie se afla algoritmii mai lenti, care parcurg imaginea o data pentru a incadra diferitele obiecte, iar apoi inca o data pentru a le analiza pe fiecare in parte. Prima categorie este mai folosita in realitate, deoarece majoritatea aplicatiilor care necesita acesti algoritmi sunt in timp real. You Only Look Once (YOLO) face, evident, parte din aceasta categorie.

Lansat in 2015, YOLO este pana in prezent cel mai performant algoritm de acest tip. Spre deosebire de solutiile anterioare care foloseau clasificatori [[2](https://pjreddie.com/darknet/yolo/)] , YOLO incearca o abordare noua, folosind o singura retea neurala. Aceasta imparte imaginea in regiuni si prezice probabilitatea ca acele regiuni sa contina un obiect.

Ca mod de functionare, acesta imparte imaginea in mai multe celule (grid) si calculeaza probabilitatea ca acestea sa faca parte dintr-un obiect. 2 celule pot contine parti din mai multe obiecte, deci pot face parte din mai multe bounding-box-uri.

Ca output al algoritmului vom avea pentru fiecare obiect detectat, un bounding box, clasa si siguranta predictiei.

YOLO este capabil sa proceseze un video cu 25 milisecunde latenta [[3](https://arxiv.org/pdf/1506.02640.pdf)]. In prezent, algoritmul a ajuns la a 8-a versiune, sprijinit de comunitate, fiind un algoritm open source. In acest proiect a fost folosita versiunea 3 deoarece am gasit mai mult suport pentru ea online.

Am folosit ca si set de date COCO (Common Objects in Context), un set de date ce contine 80 de clase (vezi fisierul COCO.names) si are la baza peste 300 000 de imagini. [[4](https://www.v7labs.com/blog/coco-dataset-guide#:~:text=The%20COCO%20(Common%20Objects%20in,5%20captions%20describing%20the%20scene)]

Cateva din tutorialele folosite sunt: [[2](https://pjreddie.com/darknet/yolo/)], [[5](https://www.youtube.com/watch?v=npJsmbFZiMg&t=525s)].

* ESP32-CAM

Pentru portabilitate am ales aceasta placuta ce contine o camera de 2Mpx, are o raza de 65 de grade si se poate conecta la WiFi sau Bluetooth. De asemenea, are si un consum mic de energie. Este utilizata impreuna cu un programator pentru comunicarea seriala cu calculatorul. Numarul default de fps este de 10, insa poate ajunge pana la 60 (cu racire recomandata). Calitatea videoclipului depinde si de reteaua folosita. Aceasta poate fi schimbata modificand credentialele din fisierul camera.ino.

* Suport hardware

Suportul hardware este alcatuit din 2 componente ce actioneaza independent: camera si o platforma improvizata care ii permite sa se roteasca. Este folosit un Arduino UNO pentru a controla prin intermediul unui driver un stepper care se roteste aproximativ 100 de grade in ambele directii. Camera este legata de axul motorului. Toate componentele sunt fixate pe un suport de carton. Alimentarea se realizeaza cu ajutorul a 2 baterii externe: una pentru camera si alta pentru Arduino. Programele care ruleaza pe cele doua placute sunt independente. Schema partii care asigura mobilitatea este:

Schema a suportului rotativ


Figura 1: Schema electrica suport rotativ

1. Etapa de implementare

* Biblioteci folosite
  + OpenCV

Este folosita cu mai multe scopuri: preluarea de imagini de la un URL si transformarea lor intr-un format numeric cu care calculatorul poate lucra (cv.imdecode()), desenarea de dreptunghiuri in jurul obiectelor identificate si afisarea stream-ului de la camera.

Modulul dnn permite incarcarea modelului YOLO din calculator (continut in fisierele yolov3.cfg si yolov3.weights).

* + PushBullet

Este o biblioteca ce permite interactiunea cu platforma cu acelasi nume pentru a putea trimite notificari pe telefon.

* + Urllib

Este folosita pentru a prelua imaginile transmise continuu de camera la adresa IP locala a camerei.

* + Biblioteci Arduino IDE

*WiFi.h* permite placutei sa se conecteze la reteaua de internet.

*WebServer.h* permite crearea unui server web unde camera va transmite imagini. ESP 32 va actiona ca un server http.

*Esp32cam.h* include functii specifice, spre exemplu setarea rezolutiei camerei.

* API

Pentru a trimite notificari, se foloseste serviciul PushBullet, o platforma ce faciliteaza trimiterea de imagini, documente, mesaje intre mai multe dispozitive, pe baza unui cont gratuit. Este usor de integrat in cod, fiind nevoie doar de un access key (preluat din contul meu) si cateva functii specifice.

* Cod
  + Structura fisierelor si interactiunea dintre ele

O imagine care conține diagramă, linie, Plan, Paralel

Descriere generată automat

Figura 2: Structura Programe

* + Cod Python

In main.py se utilizeaza modelul YOLO. Cateva linii semnificative de cod insotite de comentarii sunt cele referitoare la setarea parametrilor si incarcarea modelului :

# dimensiunea patratului maxim in care intra un obiect de interes

whT = 320

# thresholdul peste care se afiseaza apartenenta la o clasa

# daca algoritmul este peste 50% sigur de obiect se afiseaza

confThreshold = 0.5

# un coeficient care elimina redundanta: daca sunt 2 sau mai multe obiecte se suprapun

# ele sunt retinute si sortate in functie de scorul de incredere

# daca setam acest coeficient ca fiind mai mic, se elimina mai puternic obiectele suprapuse

# insa este posibil sa se elimine si detectii valide. Daca este mai mare, este posibil

# sa priveasca acelasi obiect ca pe 2 obiecte diferite

nmsThreshold = 0.3

# incarcam modelul YOLO

modelConfig = "C:/Users/Violeta/Desktop/AMp/yolov3.cfg"

modelWeights = "C:/Users/Violeta/Desktop/AMp/yolov3.weights"

# obiectul net reprezinta modelul nostru. Este incarcat din cele 2 fisiere

net = cv2.dnn.readNetFromDarknet(modelConfig, modelWeights)

# seteaza backend-ul preferat- cine ruleaza reteaua neurala (o alternativa ar fi CUDA)

net.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_OPENCV)

# seteaza componenta hardware care se ocupa de partea computationala.

# se alege CPU-ul pentru portabilitate (alternativa: GPU)

net.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CPU)

De asemnea, de interes este si modul in care imaginile sunt preluate de la serverul web de pe ESP32 si introduse in algoritm. Acestea trebuiesc convertite intr-un format numeric, apoi intr-un format pe care OpenCV il poate folosi. Ca input direct pentru reteaua neurala este folosit un obiect de tip BLOB (binary large object).

Afisam apoi imaginea cu obiectele detectate. Aceste actiuni au loc in interiorul unei bucle while True, imaginile afisate formand un flux continuu:

 # se ia o imagine de la url

    img\_resp = urllib.request.urlopen(url)

    # se converteste intr-un format numeric

    imgnp = np.array(bytearray(img\_resp.read()), dtype=np.uint8)

    # converteste imaginea din format numeric intr-un format pentru opencv

    im = cv2.imdecode(imgnp, -1)

    # se intoarce imaginea cu fundul in jos pentru o asezare mai usoara a hardware-ului

    im = cv2.flip(im, 0)

    # un obiect de tip blob (binary large object) e creat din imaginea preluata de la url

    blob = cv2.dnn.blobFromImage(im, 1 / 255, (whT, whT), [0, 0, 0], 1, crop=False)

    # acest blob e setat ca input pentru reteaua neurala cu numele "net"

    net.setInput(blob)

    # in outputNames se pun numele straturilor de output ale modelului

    outputNames = net.getUnconnectedOutLayersNames()

    # se retin in outputs datele ce ies din straturile de iesire

    outputs = net.forward(outputNames)

    # se apeleaza functia

    findObject(outputs, im)

    # se afiseaza stream-ul pana cand

    cv2.imshow('Camera rec', im)

In supraveghere.py, codul este identic, insa nu se mai afiseaza streamul, ci se adauga cateva linii de cod ce fac posibila trimiterea de notificari.

In primul rand, pentru a face legatura cu contul meu de PushBullet este nevoie de un access\_token:

access\_token = "Nu\_degeaba\_facem\_SSC"

Avem nevoie de un titlu pentru mesaj si mesajul in sine:

data = "Camera de supraveghere"

text\_persoana = "Alerta persoana."

Este introdusa o variabila care numara frame-urile in care apare obiectul de interes. Cand aceasta variabila ajunge la valoarea 20 (aprox 2 sec de aparitie a obiectului), se trimite o notificare.

global time\_person

if time\_person == 20:

                pb = PushBullet(access\_token)

                push = pb.push\_note(data, text\_persoana)

In User.py este creata o interfata grafica simpla pentru utlizator. Exista 3 butoane: unul pentru pornirea programului main.py, unul pentru pornirea programului supraveghere.py si altul pentru oprirea oricarui program inceput.

Pentru a lansa acest program intr-o mainera mai prietenoasa (fara sa se intre pe program si sa se apese butonul RUN), am folosit comanda <<python -m PyInstaller --onefile "C:\Users\Violeta\Desktop\AMp\User.py">> pentru a crea un executabil. Acesta poate fi gasit in folderul “dist”.

* + Cod Arduino UNO

Pentru controlul motorului, se seteaza pinii:

Stepper myStepper(stepsPerRevolution, 8, 10, 9, 11);

Se seteaza numarul de pasi si viteza:

const int stepsPerRevolution = 300;

 myStepper.setSpeed(2);

Se porneste mai intai in sensul acelor de ceasornic, se opreste 0.5s si apoi porneste invers:

myStepper.step(stepsPerRevolution);

  delay(500);

  myStepper.step(-stepsPerRevolution);

  delay(500);

* + Cod ESP32 Cam

Rolul acestui program este de a configura camera, a realiza conexiunea la WiFi si de a afisa adresa la care vom gasi fluxul video.

Se creaza un server web care comunica pe portul 80 (http) prin comanda:

WebServer server(80);

Se seteaza dimensiunea imaginilor capturate:

static auto hiRes = esp32cam::Resolution::find(800, 600);

Se realizeaza o functie care face o poza (daca nu reuseste trimite un mesaj de eroare la server), afiseaza pe ecran faptul ca a fost facuta poza si o trimite catre server. Mai intai se seteaza cateva campuri din headerul protocolului – 200 (cod ok) si tipul datei – “image/jpeg”

void serveJpeg()

{

  auto frame = esp32cam::capture();

  if (frame == nullptr) {

    Serial.println("CAPTURE FAIL");

    server.send(503, "", "");

    return;

  }

  Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame->getHeight(),

                static\_cast<int>(frame->size()));

  server.setContentLength(frame->size());

  server.send(200, "image/jpeg");

  WiFiClient client = server.client();

  frame->writeTo(client);

}

In sectiunea de setup se configureaza aspecte legate de camera, precum rezolutia sau dimensiunea buffer-ului de imagini:

{

    using namespace esp32cam;

    Config cfg;

    cfg.setPins(pins::AiThinker);

    cfg.setResolution(hiRes);

    cfg.setBufferCount(2);

    cfg.setJpeg(80);

    bool ok = Camera.begin(cfg);

    Serial.println(ok ? "CAMERA OK" : "CAMERA FAIL");

  }

Mai departe, se configureaza reteaua- se seteaza modul “WIFI\_STA” ce semnifica faptul ca ESP32-ul doar se va conecta la retea, nu va actiona la randul lui ca un router. Intre resetari ale placutei nu se vor retine credentiale.

WiFi.persistent(false);

  WiFi.mode(WIFI\_STA);

  WiFi.begin(WIFI\_SSID, WIFI\_PASS);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

  }

Se afiseaza adresa IP locala a placutei si se porneste serverul

Serial.println(WiFi.localIP());

  Serial.println("  /cam-hi.jpg");

  server.on("/cam-hi.jpg", handleJpgHi);

Loop-ul se ocupa de cererile clientilor prin functia: server.handleClient();

1. Mod de utilizare

Pentru inceput, camera trebuie conectata la o retea de internet locala, acest lucru realizandu-se modificand credentialele din fisierul camera.ino.

Camera porneste daca este alimentata (fie de la priza, fie de la o baterie externa sau laptop). La fel si suportul rotativ.

Se deschide fisierul User.exe ce lanseaza o fereastra. Aceasta are 3 butoane: primul porneste streamul video, al doilea nu afiseaza streamul insa daca, camera vede un obiect de interes trimite o notificare. Notificarea este trimisa o singura data intre utilizari. Al treilea buton opreste actiunea curenta. Fereastra se inchide apasand x.

O imagine care conține text, calculator, captură de ecran, Sistem de operare

Descriere generată automat

Figura 3: Interfata utilizator

O imagine care conține electronice, cablu, Inginerie electronică, Cabluri electrice

Descriere generată automat

Figura 5: Proiectul

O imagine care conține text, captură de ecran, Font, număr

Descriere generată automat

Figura 6: Notificarea

* Personalizare

In prezent, se trimite notificarea daca se decteaza cel putin 20 de frame-uri in care apare o persoana. Acest numar poate fi modificat din software in functie de preferinte (in fiecare secunda se inregistreaza in jur de 10 cadre). La fel, se poate personaliza mesajul trimis sau obiectul de interes.

In modul de vizualizare camera, sunt identificate doar cateva din cele 80 de obiecte posibile, acest lucru putand fi modificat doar din cod. Am pastrat cateva relevante pentru demonstratie. O aplicatie completa ar presupune si posibilitatea de a customiza aceste cerinte din interfata grafica (ce anume sa detecteze, dupa cat timp de la aparitie etc).

* Cerinte

Pentru un utilizator obisnuit, folosirea aplicatiei este destul de clara din interfata grafica. Nu am reusit sa scap de terminalul din spate, ceea ce este putin intimidant pentru cineva care doar vrea sa foloseasca aplicatia. Din partea utilizatorului este nevoie doar de o conexiune stabila in locul unde vrea sa amplaseze camera. Ar fi utila o modalitate mai simpla de a modifica reteaua de WiFi la care se conecteaza aparatul, spre exemplu din interfata grafica, nu din cod, insa aceasta functionalitate nu a fost implementata aici. Mai este nevoie si de o priza aflata in proximitate sau de niste baterii. Pentru usurinta in folosire s-ar putea folosi un HUB USB si, eventual, un circuit cu un intrerupator care sa permita pornirea sistemului doar prin apasarea unui buton.

De asemenea, atunci cand se schimba reteaua, trebuie sa luam adresa IP locala afisata de camera.ino si sa o punem in cele 2 programe python.

Pentru primirea notificarilor este nevoie de un cont pe platforma PushBullet. Acesta trebuie legat de programul supraveghere.py printr-un token.

1. Concluzii

Algoritmul folosit are rezultate satisfacatoare, desi are suficiente limitari. In primul rand, nu face distinctia dintre o poza cu obiectul si un obiect. Are probleme in a identifca unele obiecte- spre exemplu carti (nu stiu daca este din cauza pozitiei sau altceva) sau mouse-uri (poate pentru ca mouse-ul meu arata putin mai atipic). Are rezultate bune atunci cand exista obiecte unele langa altele. De asemenea, e foarte bun la detectia persoanelor – este de ajuns sa vada o mana uneori sau o parte din persoana ca sa afiseze acest lucru.

Din punct de vedere al rapiditatii suntem mai rapid constransi de hardware si de viteza internetului decat de algoritm in sine.

Placuta se incinge dupa 20-30 de secunde de utilizare continua (nu suficient de mult cat sa o afecteze) la fps-ul default. Daca ar fi sa crestem numarul de cadre, cu siguranta ar fi nevoie de un mini- ventilator.

Algoritmul poate fi antrenat si pe propriul set de date, deci s-ar putea adauga mai multe obiecte de interes (multe din lista de 80 de clase din setul de date sunt inutile pentru o detectie intr-o incapere- spre exemplu elefant, hidrant, avion etc).

O problema pentru un utilizator ar fi schimbarea locatiei camerei (reteaua), deoarece trebuie sa modifice 3 fisiere de cod.

Controlul stepperului este si el unul primitiv deoarece nu retine o pozitie anume de inceput, pornind din locul unde a fost lasat ultima data. Cablul de la camera ingreuneaza aceasta miscare. Ideal ar fi fost ca stepper-ul sa roteasca si sursa de alimentare a camerei pentru a nu exista probleme precum incurcarea firelor.

Cred ca aceasta camera de supravegere poate fi intr-adevar folosita intr-un mod viabil, inclusiv de cineva care nu a citit aceasta documentatie, avand o utilizare intuitiva.

Acest proiect poate fi integrat intr-unul mai complex, spre exemplu intr-un robot.

1. Referinte bibliografice
2. V7labs.com, <https://www.v7labs.com/blog/yolo-object-detection>
3. Pjreddie.com, <https://pjreddie.com/darknet/yolo/>
4. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali FahradiYou Only Look ONce: Unified, Real-Time Object Detection. Univeristy of Washington. <https://arxiv.org/pdf/1506.02640.pdf>
5. V7labs,<https://www.v7labs.com/blog/coco-dataset-guide#:~:text=The%20COCO%20(Common%20Objects%20in,5%20captions%20describing%20the%20scene>
6. Youtube, <https://www.youtube.com/watch?v=npJsmbFZiMg&t=525s>