УчИМИ Ученически институт по математика и информатика

ДВАДЕСЕТА УЧЕНИЧЕСКА КОНФЕРЕНЦИЯ

УК’21

ТЕМА НА ПРОЕКТА

HumanRadar

Автор:

Владимир Валентинов Вълков

ППМГ „Добри Чинтулов”, гр.Сливен, 10в клас

Email: **vladimirvylkov07@gmail.com** Телефон: **0877 478497**

Научен ръководител (консултант):

Петър Веселинов Стоянов

Управител, Астра Пейджинг ЕООД

Email: **peter@astrapaging.com** Телефон: **0878 624434**

Резюме на български език

HumanRadar

Автоматичното откриване и преброяване на хора е проблем, който се налага да бъде решаван в най-разнообразни области от живота – посетители в магазин, пешеходен трафик, оценка на интерес на посетители към определен вид стоки, охранителни системи и др.

В съвремените системи за видеонаблюдение са реализирани алгоритми за откриване на движение, но разпознаването и преоброяването на хора е реализирано само в най-скъпите системи.

Чрез HumanRadar е възможно вече изградена система за видеонаблюдение да бъде разширена с нова функционалност без да се налага подмяна на оборудване и закупуване скъпи хардуерни или софтуерни модули.
По този начин всеки голям магазин може да получи ценен инструмент за анализ на клиентския трафик, като информира собственика/мениджъра кога и в какви часове магазинът е най-посещаван, пред кои стоки клиентите се задържат най-много и др.

В ситуацията на пандемия от **COVID-19**, **HumanRadar** може да се използва и като инструмент за откриване на струпвания на хора на опасна дистанция и да генерира съответните предупреждения.

Резюме на английски език

HumanRadar

Automatic detection and counting of people is a problem that needs to be solved in various areas of life - shoppers, pedestrian traffic, assessment of visitors interest in a particular type of goods, security systems and more.

In modern video surveillance systems are implemented algorithms for motion detection, but the recognition and counting of people is realized only in the most expensive ones.

Through HumanRadar it is possible to expand an already built video surveillance system with new functionality without having to replace equipment and purchase expensive hardware or software modules.
In this way, each large store can get a valuable tool for analyzing customer traffic, informing the owner / manager when and at what hours the store is most visited, in front of which goods customers stay the most and more.

With the notorious **COVID-19** situation, **HumanRadar** can also be used as a tool to detect crowds at dangerous distances and generate appropriate warnings.

УВОД

Автоматичното откриване и преброяване на хора е проблем, който се налага да бъде решаван в най-разнообразни области от живота – посетители в магазин, пешеходен трафик, оценка на интерес на посетители към определен вид стоки, охранителни системи и др.

Целта на проекта е да се създаде софтуер, който успешно открива хора, следи ги, преброява ги и в следствие анализира резултатите. HumanRadar успява успешно да постигне тези цели чрез **YOLO**. Данните се записват в база данни и последствие се създава графика на резултатите.

Завършеният проект има за цел да помогне на собственици на магазини, ресторанти или всякакви закрити заведения като им предоставя данни, чрез които всеки собственик може да направи някакъв анализ.

Цели на проекта:

1.Откриване на хора в кадрите от видеокамера в реално време

2. Проследяване на движението на откритите обекти

3.Преброяване на текущите и преминалите обекти в обхвата на камерата

4. Запаметяване на резултатите в база данни

5. Анализиране на резултатите

Етапи на разработка:

1. Определяне на цялостната идея на проекта.
2. Разделяне на проекта на отделни етапи и планиране на срокове за завършване.
3. Запознаване с начина на работа на YOLO, както и останалите библиотеки на проекта.
4. Създаване на цялостен сорс код.
5. Извършване на различни тестове
6. Анализиране на положителните страни и недостатъците на проекта.
7. Създаване на краен продукт, работещ надежно и безотказно.
8. Оформяне на документация и цялостен завършек на проекта.

Документацията е разделена на следните глави:

Глава 1 - Проектиране и реализация на софтуера

В тази глава са описани използваните технологии и защо са избрани точно те.

Глава 2 - Софтуер

Тук е представен и разгледан подробно целия сорс код, който управлява проекта.

Глава 3 - Основни моменти от разработката на проекта

В тази глава са описани основните проблеми и моменти, през които преминах в процеса на създаване на проекта.

Глава 4 - Снимки и начин на работа

Представени са снимки на крайния продукт и начина му на използване в ежедневието.

Глава 5 - Заключение

Описание на постигнатите резултати и бъдещи идеи за развитие на проекта.

Глава 1- Проектиране и реализация на софтуера

1.1 Език за програмиране – Python

Проекта е написан изцяло на Python. Използвани са последните технологии за Python разработки.

1.2 Обработка на изображения – OpenCV

OpenCV е библиотека за езиците Python, C/C++ създадена от Intel, която е безплатна за използване под лицензa Apache 2 с отворен код. Библиотеката свърши чудесна работа свързана с обработка на изображения в реално време. Има и много други подобни билиотеки, но избрах OpenCV поради това, че е много лесна за работа и се справя много бързо в real-time обработката на изображения.

1.3 Framework за машинно обучение – TensorFlow

Tensorflow е библиотека за езиците Python, C++, CUDA създадена от Google Brain team, която е безплатна за използване под лицензa Apache 2 с отворен код. Тя може да се използва за редица задачи, но има особен фокус върху обучението и извода на дълбоки невронни мрежи. Основното и предимство е, че може да засича обекти с GPU, а не с CPU като повечето подобни библиотеки. Работата с GPU значително ускорява работата на проекта, защото GPU е създадено точно за обработка на снимки.

1.4 Масиви и математически функции – NumPy

NumPy е библиотека за езика за програмиране на Python, C създадена като Community project. Библиотеката е безплатна за използване. Тя добавя подръжка за големи многоизмерни масиви и матрици, заедно с голяма колекция от математически функции на високо ниво за работа с тези масиви.

1.5 Графики – MatPlotLib

MatplotLib е билиотека за езика Python и неговото числово математическо разширение NumPy, която е безплатна за използване. Тя представя библиотека, която се използва за създаване и изобразяване на графики, които могат да бъдат персонализирани по собствен вкус.

1.6 Засичане на обекти – YOLO

YOLO е безплатен за използване алгоритъм създаден от Darknet за езика Python. YOLO се занимава с намирането до 80 вече обучени обекти (хора, коли, животни, сгради и др.). Реших да използвам YOLO, поради високата скорост с която може да засича обекти и лесната му употреба.

Глава 2 – Софтуер

Проекта се оправлява от функционален сорс код, написан на средата за разработка (**Integrated Development Environment - IDE**) на PyCharm, която може да се изтегли безплатно от официалния сайт. PyCharm използва адаптираната версия на езика Python.

Целият сорс код на проекта е добавен към документацията в **ПРИЛОЖЕНИЕ 1.**

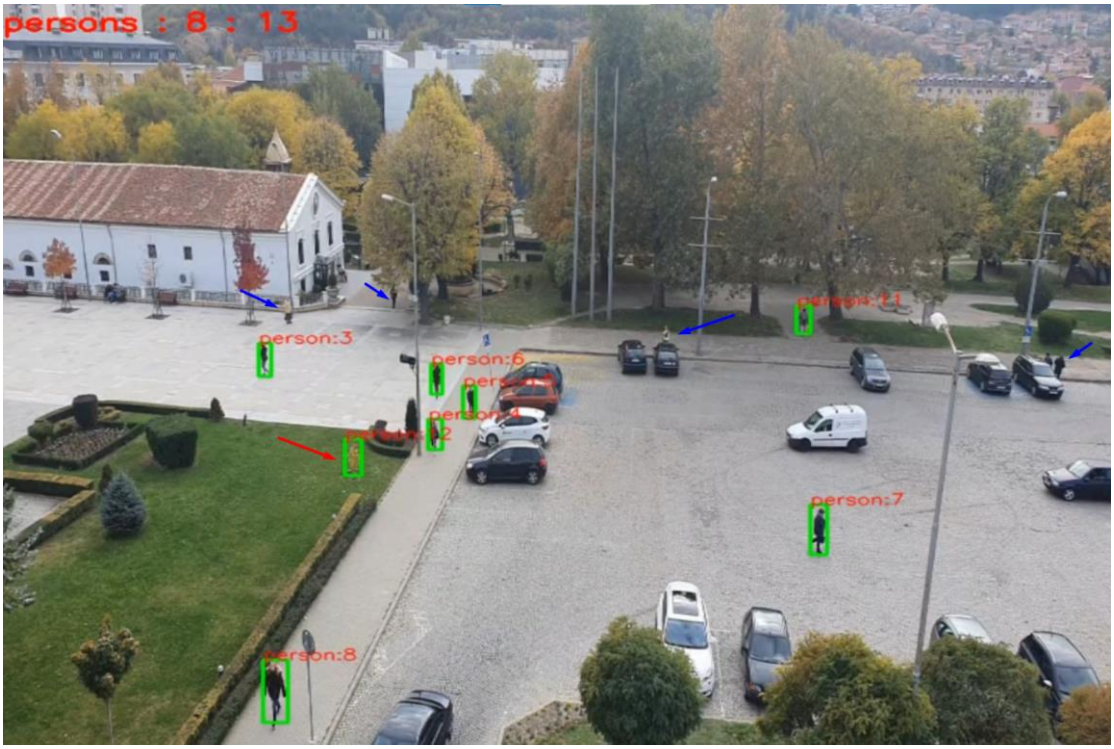
Глава 3 - Основни моменти от разработката на проекта

За да се получи добре работещ продукт трябваше да се преодолеят няколко основни проблема:

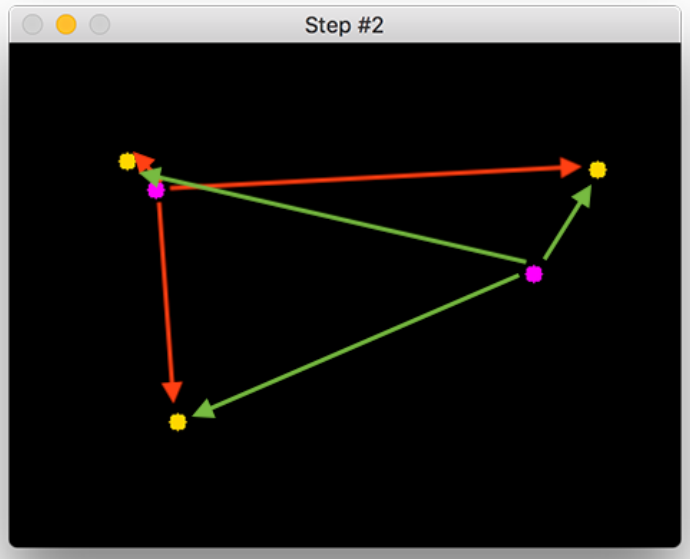
1. Имплементация на YOLO – проблеми с бъгове в библиотеката, които трябваше да бъдат заобиколени
2. Използване на GPU, а не CPU за обработката на снимки в реално време – проблеми със скоростa, която е много важна, когато се работи в реално време
3. Правилно структуриране на SQLite базата данни.
4. Object tracking – проблеми със следенето на намерените обекти.

Глава 4 - Снимки и начин на работа

4.1 Снимка на завършения продукт



4.2Tracking



4.3 Config file

```
path = E:\IT състезания\PeopleDetection\test_input\0001-1547.mp4
radiusThreshHold = 150.0
framesThreshHold = 25
logInterval = 1000
```

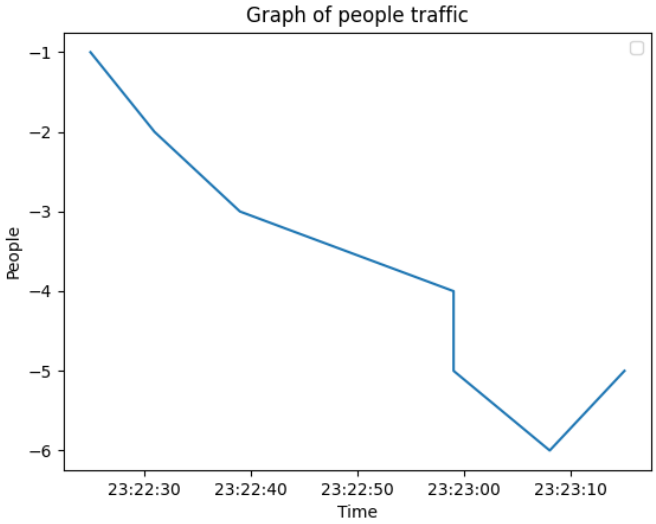
4.4 Counter log

	id	log date time	human counter
1	3354	2020-12-10 18:36:34	0
2	3355	2020-12-10 18:36:36	0
3	3356	2020-12-10 18:36:38	0
4	3357	2020-12-10 18:36:39	0
5	3358	2020-12-10 18:36:40	0
6	3359	2020-12-10 18:36:41	2
7	3360	2020-12-10 18:36:42	0
8	3361	2020-12-10 18:36:43	0
9	3362	2020-12-10 18:36:44	0
10	3363	2020-12-10 18:36:45	0
11	3364	2020-12-10 18:36:46	0
12	3365	2020-12-10 18:36:47	0
13	3366	2020-12-10 18:36:48	0

4.5 Person log

	id	log date time	direction
1	105	2020-12-10 18:37:30	0
2	106	2020-12-10 18:38:44	0
3	107	2020-12-10 18:38:45	0
4	108	2020-12-10 18:39:04	0
5	109	2020-12-10 18:39:22	1
6	110	2020-12-10 18:39:45	0
7	111	2020-12-10 18:40:39	0
8	112	2020-12-10 18:40:51	0
9	113	2020-12-10 18:41:09	0
10	114	2020-12-10 18:42:04	0
11	115	2020-12-10 18:42:53	0
12	116	2020-12-10 18:42:56	0
13	117	2020-12-10 18:45:02	1

4.6 Графика



4.7 Начин на работа

Начинът на работа е много прост. Всяка секунда с помоща на YOLO се засичат обектите на снимката. Стойностите, които се получават за всеки обект са – координати на обекта, коефициент на точност и други, но ние използваме само тези две стойности. За да се избегнат грешки се проверява дали коефициента на точност на всеки един обект е повече от 0.5 ако да се приема за човек, ако не значи го приемаме за обект. За да можем да следим хората всяка секунда се проверяват миналите и сегашните координати и най-близките координати се считат за едни и същи(*фиг. 4.2*). Когато някой обект се изгуби се чака определени frame-ове, които могат да се персонализират в config файла и когато минат тез frame-ове определения обект с id и координати се губи и се запаметява в базата данни. В config файла могат да се персонализират няколко неща (*фиг.4.3*)

- path – мястото на клипа, който ще се анализира (ако ще се използва камерата се изписва една нула)
- radiusThreshHold – радиуса, който се гледа за object tracking (колкото по-близко са обектите, толкова по-голям е thresh hold-а, колкото по-надалече, толкова по-малък thresh hold)
- framesThreshHold – колко frame-а да минат, когато се изгуби обект за да го забравим.
- logInterval – през колко време в милисекунди да се запаметява в база данни

На всеки logInterval милисекунди се запаметява в базата данни, която е съставена от две таблици – една, която запаметява кога колко души е имало в кадър (*фиг. 4.4*) и друга, която запаметява за всеки един човек кога е напуснал кадър и в каква посока е ходил(1 за нагоре и 0 за надолу) (*фиг. 4.5*). Когато всички данни са запаметени се създава графика, която чете от базата данни тези стойности(*фиг. 4.6*). На *фиг.4.1* може да се види и как изглежда изображение от клипа.

Глава 5 – Заключение

С направените експерименти може да се направи заключение, че HumanRadar с помощта на YOLO успешно открива и преброява трафик на хора в определена зона при:

- използване на камера с добра резолюция
- добра осветеност на зоната
- избягване на зони с голяма отдалеченост от камерата
- използване на добра видеокарта с по-мощен GPU

Плановете за развитие на проекта включват:

- Експериментална реализация в голям търговски обект.
- Изчисляване на скорост на движение на обектите.
- Откриване на струпвания на хора за продължително време и генериране на предупреждения
- Визуализиране на откритите обекти върху схематичен план на търговския обект

ИЗТОЧНИЦИ НА ИНФОРМАЦИЯ

YOLO - https://pjreddie.com/darknet/yolo/
Tensorflow - https://www.tensorflow.org/
Tensornets - https://github.com/taehoonlee/tensornets
Matplotlib - https://matplotlib.org/

ПРИЛОЖЕНИЕ 1 – СОРС КОД

```
import tensorflow.compat.v1 as tf
import tensornets as nets
import cv2
import numpy as np
from trackedObject import trackedObject
import sqlite3
from sqlite3 import Error
import time
import matplotlib.pyplot as plt
from configparser import ConfigParser


tf.disable_v2_behavior()
# make the program run with GPU and not with CPU
gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
    try:
        # Currently, memory growth needs to be the same across GPUs
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu, True)
        logical_gpus = tf.config.experimental.list_logical_devices('GPU')
        print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
    except RuntimeError as e:
        # Memory growth must be set before GPUs have been initialized
        print(e)

# setup DB
conn = sqlite3.connect(r"E:\IT състезания\HumanRadar-Not git\humanradar.db")
cur = conn.cursor()


# setup YOLOV3
inputs = tf.placeholder(tf.float32, [None, 416, 416, 3])
model = nets.YOLOv3COCO(inputs, nets.Darknet19)
output = None
classes = {'0': 'person'}
list_of_classes = [0]
trackedObjects = []

config = ConfigParser()
config.read('E:\IT състезания\PeopleDetection\config.ini',"utf-8")
path = config.get('section_a', 'path')
radiusThreshHold = config.getfloat('section_a', 'radiusThreshHold')
framesThreshHold = config.getint('section_a', 'framesThreshHold')
logInterval = config.getint('section_a', 'logInterval')
masterID = 0
```

```
frames = 0
skipFrames = 0
counter = 0
m = int(round(time.time() * 1000))
startDate = time.strftime('%Y-%m-%d %H:%M:%S')


def logDB(counter):
    try:
        date = time.strftime('%Y-%m-%d %H:%M:%S')


        conn.execute(f"Insert into humanlog (log_date_time, human_counter) values ('{date}',
{counter})")



    except Error as e:
        print(e)

def logDB_people(dir='null'):
    try:
        date = time.strftime('%Y-%m-%d %H:%M:%S')

        conn.execute(f"Insert into personlog (log_date_time, direction) values ('{date}', '{dir}')")
    except Error as e:
        print(e)
def graph():
    try:
        cur.execute(f"Select * from personlog where log_date_time >= '{startDate}' and
log_date_time <= '{time.strftime('%Y-%m-%d %H:%M:%S')}'")
        rows = cur.fetchall()

        xGraph = list()
        yGraph = list()
        value = 0

        for row in rows:
            if row[2] == 1:
                value += 1
            elif row[2] == 0:
                value -= 1

            xGraph.append(np.datetime64(row[1]))
            yGraph.append(value)



        plt.plot(xGraph, yGraph)
        plt.xlabel("Time")
        plt.ylabel("People")
```

```
        plt.title("Graph of people traffic")
        plt.legend()
        plt.show()
        plt.savefig(r'E:\IT състезания\PeopleDetection\test_output\Graph.png', dpi=None,
facecolor='w', edgecolor='w',
                    orientation='portrait', papertype=None, format='png',
                    transparent=False, bbox_inches=None, pad_inches=0.1)
    except Error as e:
        print(e)


with tf.Session() as sess:
    sess.run(model.pretrained())
    cap = cv2.VideoCapture(path)  # setup the video
    #cap = cv2.VideoCapture(r"E:\Download\0001-1547.mp4")  # setup the video
    fps = int(cap.get(cv2.CAP_PROP_FPS))
    skipFrames = fps
    print(fps)
    while (cap.isOpened()):  # start looping through the video frame by frame
        millis = int(round(time.time() * 1000))
        if millis - m >= logInterval:
            logDB(counter)
            counter = 0
            m = millis
        # resizing the frame
        ret, frame = cap.read()
        frames += 1
        for tObj in trackedObjects:
            tObj.count()
            tObj.zone(time.time() * 1000)
            if tObj.frames >= framesThreshHold:
                if tObj.zone1 != 0 and tObj.zone2 != 0:
                    if tObj.zone1 - tObj.zone2 > 0:
                        direction = 1 #up
                    else:
                        direction = 0 #down
                    logDB_people(direction)
                trackedObjects.remove(tObj)

        if frame is None:
            break
        img = cv2.resize(frame, (800, 800))
        imge = np.array(img).reshape(-1, 800, 800, 3)
        preds = sess.run(model.preds, {inputs: model.preprocess(imge)})

        # take bounding boxes coordinates
        boxes = model.get_boxes(preds, imge.shape[1:3])
        cv2.namedWindow('image', cv2.WINDOW_NORMAL)

        cv2.resizeWindow('image', 1024, 768)
```

```
        boxes = np.array(boxes)
        # draw bounding boxes and count how many people are on the image
        for j in list_of_classes:
            count = 0
            if str(j) in classes:
                lab = classes[str(j)]
            if len(boxes[j]) != 0 and lab == "person":

                for i in range(len(boxes[j])):
                    box = boxes[j][i]
                    x = box[0]
                    y = box[1]
                    confidence = box[4]

                    if confidence >= 0.5:
                        count += 1
                        # print(str(len(boxes[j])) + ":" + str(len(trackedObjects)))
                        tempTracked = None
                        for tObj in trackedObjects:
                            # print(f"{tObj.x} : {tObj.y} : {tObj.id}")
                            if tObj.insideRadius(x, y, radiusThreshHold):
                                tObj.clear()
                                tObj.update(x, y)
                                tempTracked = tObj

                        if tempTracked is None and frames % skipFrames == 0:
                            masterID += 1
                            counter += 1
                            tempTracked = trackedObject(x, y, masterID)
                            trackedObjects.append(tempTracked)

                        if tempTracked is not None:
                            cv2.rectangle(img, (box[0], box[1]), (box[2], box[3]), (0, 255, 0), 2)
                            cv2.putText(img, lab + ":" + str(tempTracked.id), (box[0], box[1]),
cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                                        (0, 0, 255), lineType=cv2.LINE_AA)

        cv2.putText(img, "persons : " + str(count) + " : " + str(masterID), (1, 25),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 2)

        cv2.imshow("image", img)
        # saving output every frame
        if output is None:
            fourcc = cv2.VideoWriter_fourcc(*'XVID')
            output = cv2.VideoWriter(r"E:\IT състезания\PeopleDetection\output.avi", fourcc,
fps, (1024, 768), True)
        else:
            output.write(cv2.resize(img, (1024, 768)))
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
```

```
        elif cv2.waitKey(1) & 0xFF == ord('r'):
            conn.commit()
            graph()

cap.release()
if conn:
    conn.commit()
    graph()
    conn.close()
if output != None:
    output.release()
cv2.destroyAllWindows()
```