1. Viktor ZANOZIN[1], 2. Andrzej ZANKIEWICZ[2]

Politechnika Białostocka, Wydział Elektryczny (2)
ORCID: 2. 0000-0003-3840-5356

doi:10.15199/48.2023.02.11

Obrazowe monitorowanie zachowania dystansu społecznego z wykorzystaniem algorytmów sztucznej inteligencji

Streszczenie. *W artykule przedstawiony został system obrazowej analizy zachowania dystansu społecznego za pomocą współczesnych algorytmów detekcyjnych opartych na konwolucyjnych sieciach neuronowych. Algorytm wykonywany jest na procesorze graficznym (GPU), dzięki czemu wykonany system może zostać zaimplementowany na komputerze PC średniej klasy. Wynik detekcji obrazowany jest graficznie poprzez objęcie wykrytych w analizowanej scenie osób ramkami w kolorze zależnym od wyznaczonego dystansu.*

Abstract*. The article presents a system of visual analysis of social distancing behavior using modern detection algorithms based on convolutional neural networks. The algorithm is executed on a graphics processor (GPU), so that the system made can be implemented on a mid-range PC. The detection result is graphically illustrated by covering the people detected in the analyzed scene with frames in a color depending on the determined distance. (**Visual monitoring of social distancing behavior using artificial intelligence algorithms**).*

Słowa kluczowe: analiza obrazu, widzenie komputerowe, sztuczna inteligencja, dystans społeczny.
Keywords: image analysis, computer vision, artificial intelligence, social distance.

Wstęp

Pojawienie się i ogólnoświatowe rozprzestrzenienie wirusa SARS-CoV-2 w istotny sposób zmieniło wiele zasad zachowań społecznych. W szczególności jednym z najbardziej odczuwalnych skutków była konieczność znacznego ograniczenia bezpośrednich kontaktów międzyludzkich. Wymusiło to masową ekspansję wykorzystania różnych środków technicznych m.in. do zdalnej komunikacji, zdalnej edukacji, a także do weryfikacji stosowania nałożonych ograniczeń. Jednym z wprowadzonych ograniczeń był wymóg zachowania dystansu społecznego, w szczególności w miejscach publicznych. Według Światowej Organizacji Zdrowia, odległość, wynosząca co najmniej 1 metr w znaczący sposób zmniejsza ryzyko zachorowania [1]. Wymóg taki spowodował pojawienie się różnego rodzaju rozwiązań mających na celu monitorowanie stosowania nałożonych ograniczeń. Systemy takie zaczęły być powszechnie wprowadzane w dużych metropoliach oraz zakładach pracy. Charakteryzują się one przede wszystkim wykorzystaniem systemów wizyjnych oraz metod uczenia maszynowego i sztucznej inteligencji.

W artykule przedstawiony został proponowany system obrazowego monitorowania zachowania dystansu społecznego wykorzystujący algorytmy sztucznej inteligencji. System ten został praktycznie uruchomiony i zweryfikowany pod kątem skuteczności działania.

Zagadnienie automatycznej analizy obrazu

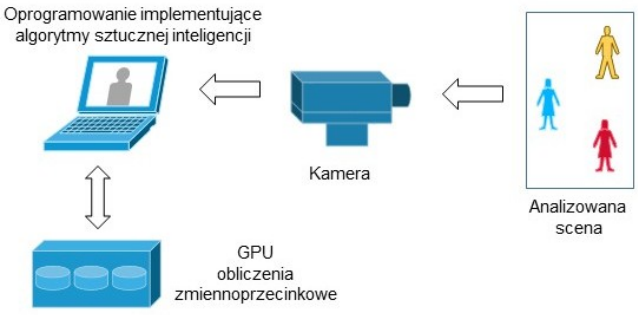
Widzenie komputerowe (ang. computer vision) jest interdyscyplinarną dziedziną naukową, która zajmuje się zagadnieniami wspomagającymi urządzenia techniczne w identyfikacji obiektów lub przetwarzaniu danych uzyskiwanych z obrazów, w podobny sposób, jak robią to ludzie. Najczęściej etapem końcowym takiego przetwarzania jest ilościowa ocena obserwowanego na obrazie zjawiska lub jego pewna interpretacja symboliczna. W ogólnym przypadku operacje wykonywane przez system rozpoznawania obrazów mogą być podzielone na następujące poziomy przetwarzania [2]:

- przetwarzanie niskiego poziomu. Jest to wykonanie czynności wstępnych, takich jak akwizycja obrazu, poprawa jakości odwzorowania (filtracja, eliminacja zakłóceń),
- przetwarzanie średniego poziomu. Obejmuje proces segmentacji obrazu oraz wyodrębniania cech charakterystycznych (krawędzie obrazów),
- przetwarzanie wysokiego poziomu. Dokonywany jest proces klasyfikacji oraz lokalizacji obrazu, następuje interpretacja analizowanej sceny.

W obecnych czasach praktyczna realizacja widzenia komputerowego może być znacznie ułatwiona poprzez zastosowanie wyspecjalizowanych bibliotek, wśród których jedną z najbardziej znanych i powszechnie stosowanych jest OpenCV [3, 4]. Zapewnia ona cały szereg algorytmów widzenia komputerowego oraz odpowiednich modułów, obejmujących takie zagadnienia jak możliwość pracy z GPU (ang. Graphical Processing Unit) do szybkiego wykonywania złożonych obliczeń, algorytmy sztucznej inteligencji, kalibracja kamery, filtracja i przetwarzanie obrazów oraz inne [3]. Możliwa jest również współpraca OpenCV z zewnętrznymi bibliotekami uczenia maszynowego, takimi jak Tensorflow oraz PyTorch. Dzięki zastosowaniu sztucznej inteligencji, system rozpoznawania obrazów uzyskuje cechy zachowania inteligentnego takie jak możliwość wnioskowania na podstawie zbioru różnych, nieskojarzonych ze sobą danych, uczenie się na przykładach, samodoskonalenie się systemu oraz interpretacja informacji (np. rozpoznawanie obiektów) na podstawie niekompletnych danych [5].

Sprzętowa i programowa struktura systemu

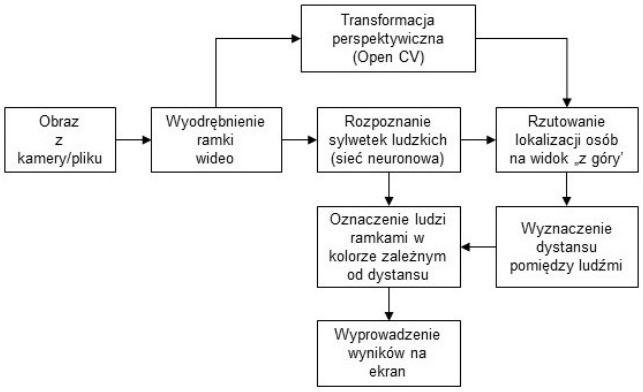
Struktura sprzętowa zaprojektowanego systemu została przedstawiona na rysunku 1.



Rys.1. Schemat blokowy opracowanego systemu

System składa się z kamery skierowanej na analizowaną scenę oraz komputera PC z procesorem graficznym (GPU). Działanie systemu zdefiniowane jest w oprogramowaniu wykonywanym na komputerze PC, które realizuje następujące kroki przetwarzania (rys. 2):

1. Obraz z kamery jest dzielony na odpowiednie ramki.
2. Do rozpoznawania obecności osób, znajdujących się na obrazie, jest wykorzystywana sieć neuronowa oraz odpowiednie algorytmy detekcyjne (YOLO lub SSD).
3. Otrzymane lokalizacje sylwetek ludzkich są mapowane za pomocą macierzy transformacji perspektywicznej (z wykorzystaniem biblioteki OpenCV), dzięki czemu uzyskiwany jest widok „Bird's eye view". Celem tego etapu jest uzyskanie nowego układu współrzędnych pozwalającego na analizę metryczną sceny.
4. W oparciu o widok „Bird's eye view" wyznaczany jest dystans pomiędzy zlokalizowanymi osobami (odległość w pikselach jest przeliczana z wykorzystaniem danych kalibracyjnych).
5. W zależności od wyznaczonej odległości, wykryte osoby oznaczane są zielonymi lub czerwonymi ramkami.



Rys.2. Schemat działania oprogramowania systemu

Z uwagi na fakt, że nawet najszybsze algorytmy, wykorzystujące konwolucyjne sieci neuronowe, takie jak YOLO oraz SSD, wymagają stosunkowo dużych mocy obliczeniowych, do wykonywania obliczeń wykorzystany został procesor graficzny NVIDIA GeForce 940 MX, który wraz z biblioteką CUDA tworzy równoległą uniwersalną platformę obliczeniową. W celach porównawczych sprawdzono także wydajność obliczeń wykonywanych przez podstawowy procesor w komputerze PC.

Jako rozwiązania alternatywne rozpatrywane było wykorzystanie procesora neuronowego Intel Movidius, który może być dołączony do mikrokomputerów jednopłytkowych typu Raspberry PI oraz wykorzystanie mocy obliczeniowej kart graficznych w chmurze (np. na platformie Amazon Web Services). W pierwszym przypadku dokładniejsza analiza wykazała, że procesor neuronowy Intel Movidius nie obsługuje przewidywanych do wykorzystania wersji bibliotek, natomiast wykorzystanie chmurowych zasobów obliczeniowych związane jest ze stosunkowo wysokimi kosztami. Dlatego ostatecznie wybrany został lokalny procesor GPU na platformie PC.

Jednym z zagadnień, związanym z mierzeniem dystansu, jest odpowiednia kalibracja programowa, która umożliwia prawidłowe oszacowanie odległości pomiędzy ludźmi, na podstawie liczby pikseli. Jedną z najprostszych metod, którą można zastosować w tym przypadku, jest wiedza o tym ile wynosi średni wzrost człowieka. Zaznaczając w procesie kalibracji na obrazie odcinek, który odpowiadałby wzrostowi człowieka, za pomocą funkcji przeskalowania w OpenCV można określić ile pikseli zajmuje zaznaczona odległość.

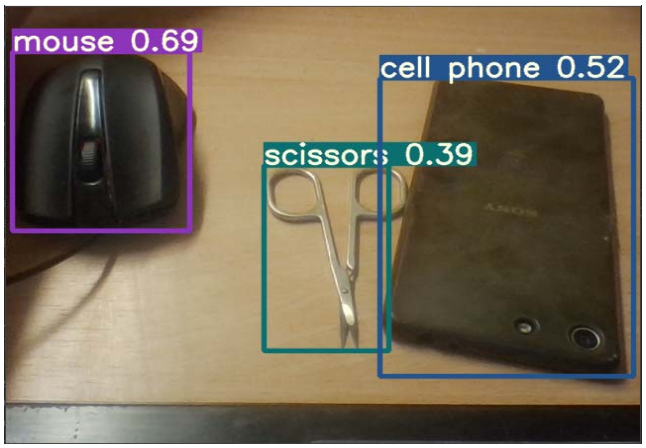
Biorąc pod uwagę pewną „standardowość" modeli sylwetki ludzkiej, do rozpoznawania ludzi może nadawać się gotowy wytrenowany model, np. COCO [6], co pozwoli wyeliminować czasochłonny proces uczenia sieci na zbiorze treningowym. Zbiór COCO zawiera w sobie setki zdefiniowanych klas, co znacząco upraszcza proces klasyfikacji obiektów.

W utworzonym oprogramowaniu zdefiniowany został także interfejs graficzny, który w postaci okienka aplikacji desktopowej (realizowanego poprze bibliotekę CV) wyświetla takie informacje jak:

- liczba osób, znajdujących się na analizowanym obrazie,
- liczba wystąpień niebezpiecznych odległości pomiędzy ludźmi,
- zmiana koloru ramki lokalizacyjnej wokół osoby znajdującej się w odległości uznanej za niebezpieczną,
- prezentacja widoku „Bird's eye view" wraz z zaznaczonymi lokalizacjami ludzi.

Podstawowym algorytmem użytym w projekcie do rozpoznawania osób jest YOLOv5 [7]. Stanowi on rodzinę architektur obrazowej detekcji obiektów oraz modeli wstępnie wyuczonych z wykorzystaniem zbioru COCO [6]. YOLOv5 zawiera pięć zestawów wag, różniących się pomiędzy sobą wydajnością, szybkością oraz rozmiarem. Jest to obecnie jeden z najszybszych modeli, wykorzystywany do przetwarzania obrazów.

Projekt z oficjalnego repozytorium YOLOv5 zawiera cały zestaw gotowych narzędzi, dzięki którym możliwa jest analiza obrazów. Punktem wejściowym projektu jest funkcja `detect.py`. Wywołując tę funkcję z dostępnymi argumentami można skonfigurować m.in. źródło obrazu (kamera, plik), typ modelu (1 z 5), próg rozpoznawalności obiektu, wybór dostępnego urządzenia do przetwarzania (CPU lub GPU). Przykładowy wynik działania detektora YOLOv5, uzyskany narzędziem `detect.py` został przedstawiony na rysunku 3. Do każdego rozpoznanego obiektu została przypisana etykieta, odpowiadająca klasie tego obiektu ze zbioru COCO oraz prawdopodobieństwo wystąpienia tego obiektu. Są to dane, uzyskiwane po destrukturyzacji obiektu, który jest zwracany przez biblioteczną metodę `apply_classifier`.

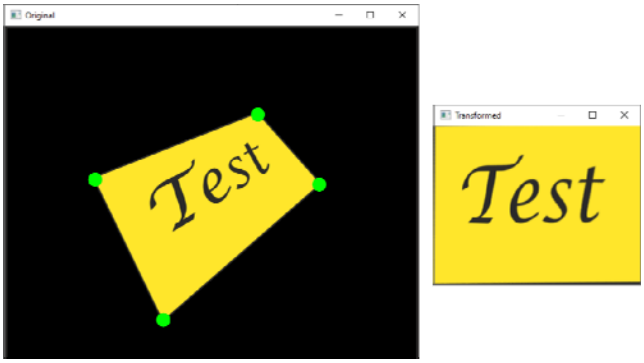


Rys.3. Przykład klasyfikacji wykonanej detektorem YOLOv5"

Kalibracja przetwarzanej sceny

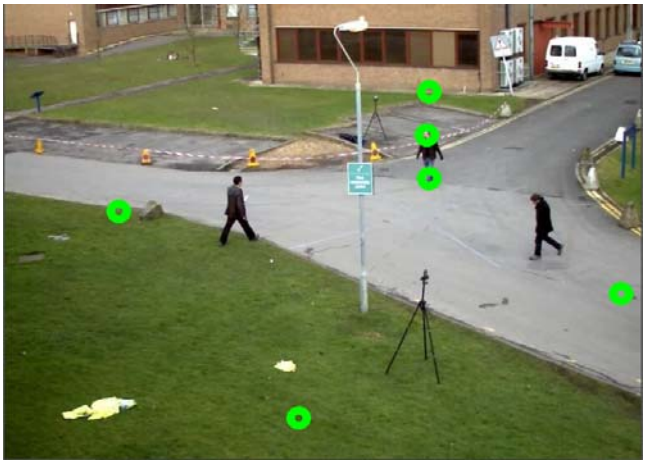
Jednym z podstawowych problemów oszacowania dystansu pomiędzy obiektami jest zniekształcenie perspektywiczne, które wynika z kąta umiejscowienia kamery w stosunku do analizowanej sceny. W artykule [8] zostało przedstawione rozwiązanie tego problemu poprzez transformację perspektywiczną za pomocą bibliotek OpenCV oraz NumPy w języku Python. Na rysunku 4

została zobrazowana idea takiej transformacji zastosowanej do obrazu kartki z napisem widzianej pod kątem około 45 stopni w stosunku do osi prostopadłej do płaszczyzny kartki. Transformacja taka wymaga podania współrzędnych czterech wierzchołków przekształcanego obszaru (na rysunku 4 zaznaczono je zielonymi punktami). Efektem końcowym przekształcenia jest widok „z góry" (ang. „Bird's eye view") przedstawiony na rysunku 4 po prawej stronie.



Rys.4. Przykład działania transformacji perspektywicznej

W przypadku projektowanego systemu przedstawione powyżej przekształcenie perspektywiczne ma na celu kalibrację rozmiaru obserwowanej sceny. Dlatego oprócz czterech punktów definiujących obszar przekształcenia perspektywicznego określa się jeszcze dwa punkty o znanej odległości, które posłużą do wyznaczenia wskaźnika metrycznego analizowanej sceny. W tym celu utworzono osobny skrypt w języku Python umożliwiający wykonanie niezbędnej kalibracji. W zależności od podanego źródła obrazu program wychwytuje pierwszą ramkę ze źródła video (plik lub kamera) lub wczytuje podany źródłowy plik graficzny oraz za pomocą myszki daje możliwość zaznaczenia sześciu punktów: na podstawie pierwszych czterech zostają wyznaczone wierzchołki obszaru RoI do transformacji perspektywicznej na widok „z góry", natomiast dwa pozostałe punkty służą do ustalenia pewnego wskaźnika metrycznego (np. wzrost dorosłej osoby albo inny obiekt o znanych wymiarach), będącego przelicznikiem pozwalającym na oszacowanie dystansu na podstawie liczby pikseli. Na rysunku 5 przedstawiono zaznaczone punkty wybrane do kalibracji przykładowej sceny. Wykorzystany do tego został jeden z plików wideo zawartych w zestawie „PETS 2009 datasets" udostępnionym publicznie do celów badawczych [9].

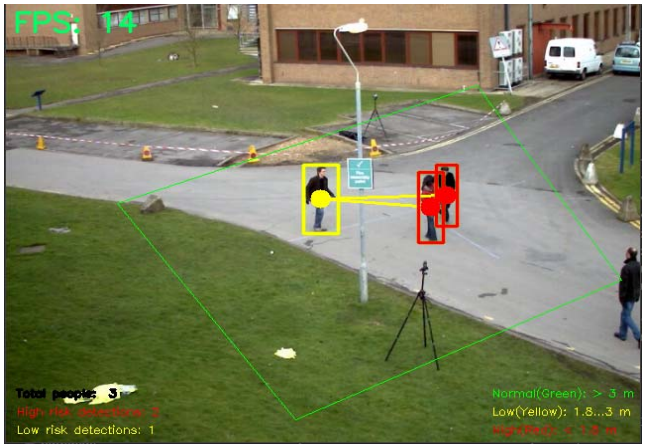


Rys.5. Przykład rysunku wstawionego jako plik „fig.tif"

Mając obliczone odległości w nowym układzie współrzędnych (po transformacji perspektywicznej na widok „z góry"), można rzutować te wartości na widok sceny. Ponadto na ekranie wyświetlane są takie informacje jak:

- liczba przetwarzanych klatek na sekundę (FPS),
- liczba osób znajdujących się wewnątrz obszaru RoI („Total people"),
- liczba osób objętych wysokim ryzykiem („High risk"),
- liczba osób objętych niskim ryzykiem („Low risk"),
- ustalony bezpieczny dystans („Normal (Green): >3 m"),
- dystans niskiego ryzyka („Low (Yellow): 1.8 - 3 m"),
- dystans wysokiego ryzyka („High (Red): <1.8 m").

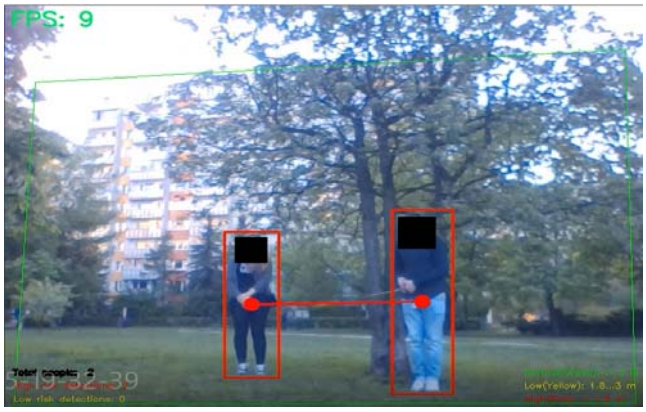
Do wyświetlania tych informacji została użyta biblioteka OpenCV. FPS został zmierzony dla każdej ramki za pomocą biblioteki imutils. Na rysunku 6 przedstawiono widok uzyskiwany w trakcie działania programu. Cienka zielona linia przedstawia granice wybranego obszaru RoI.



Rys.6. Główne okno programu

Testy z kamerą w czasie rzeczywistym

Poniżej zostały przedstawione wyniki uzyskane w trakcie testów z kamerą w czasie rzeczywistym. W odróżnieniu od plików wideo, które w poprzednim punkcie zostały wykorzystane m.in. do zobrazowania procesu kalibracji, tutaj sprawdzona została zdolność systemu do prawidłowego obliczenia dystansu w warunkach gdy obszar RoI nie stanowił odwzorowania poziomej płaszczyzny. Kalibracja wyglądała w identyczny sposób jak zostało to opisane wcześniej.



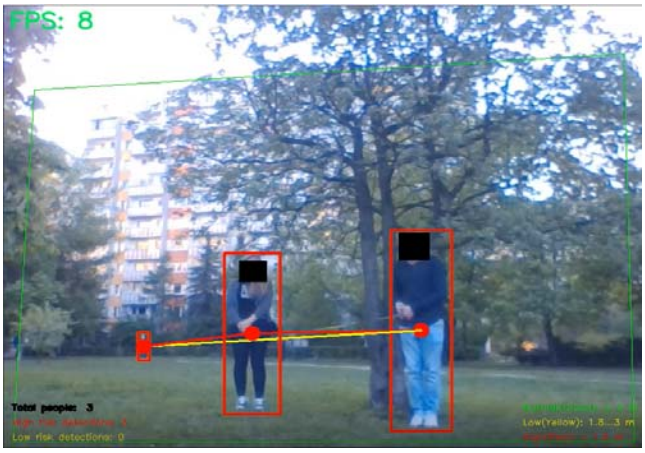
Rys.7. Przykład detekcji niebezpiecznego dystansu

W pierwszym przypadku zmierzono maksymalny dystans, który jest traktowany jako niebezpieczna odległość. Wynosił on 175-188 cm w zależności od zniekształcenia perspektywicznego (rys. 7). Następnie

zmierzono przypadek, w którym dystans był równy 3 metry (rys. 8). Wartości, znajdujące się pomiędzy tymi dwoma dystansami były traktowane jako „niskie ryzyko” i oznaczane w programie żółtą ramką.

Rys.8. Przykład detekcji bezpiecznego dystansu

Biorąc pod uwagę charakter zaznaczonego RoI, warto również dodać, że odwzorowanie nie obejmujące w pełni poziomej płaszczyzny może spowodować nieprawidłowe określenie dystansu dla obiektów znajdujących się w tle obrazu. Zostało to pokazane na rysunku 9.



Rys.9. Nieprawidłowe określenie dystansu dla obiektów „w tle”

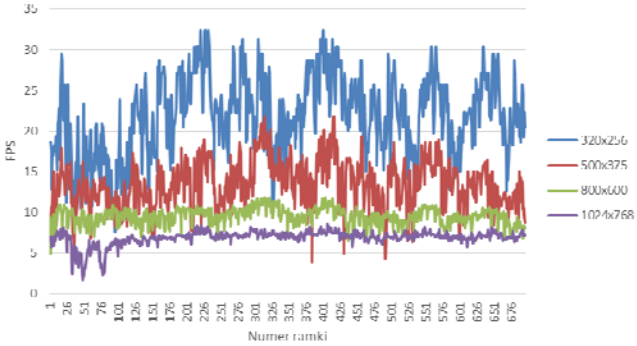
Mając informacje z powyższych wyników, można wywnioskować, że podczas instalacji kamery należy zwrócić uwagę na zakres obszaru analizowanej sceny. Zaznaczony w trakcie kalibracji obszar RoI powinien stanowić odwzorowanie poziomej płaszczyzny w przypadku obiektów pomiędzy którymi dystans może podlegać zniekształceniu perspektywicznemu. Natomiast, w przypadku gdy zniekształcenie to nie jest istotne, jako RoI można zaznaczyć nawet prawie całą ramkę obrazu.

Analiza wydajnościowa i jakościowa

Metodologia testów wydajnościowych w niniejszym projekcie sprowadza się przede wszystkim do porównania liczby klatek na sekundę (FPS) pod wpływem następujących czynników:
- rozmiar ramki,
- liczba obiektów analizowanej sceny,
- rodzaj i wariant algorytmu,
- rodzaju procesora (GPU lub CPU).

Drugim czynnikiem podczas oceny jakości jest precyzja detektorów, która jest podawana w dokumentacji oficjalnej jako mAP (ang. mean Average Precision), zazwyczaj mierzona na podstawie zbioru COCO. Odpowiednie charakterystyki mAP dla użytego detektora YOLOv5 można znaleźć na oficjalnej stronie projektu YOLOv5 [7]. Wartości te mogą służyć do ogólnego oszacowania skuteczności pracy danej wersji detektora. W przypadku analizy obrazów ruchomych mAP ma mniejszy wpływ na ocenę wizualną ze względu na możliwość korekty błędu powstałego w danej ramce podczas analizy kolejnych ramek.

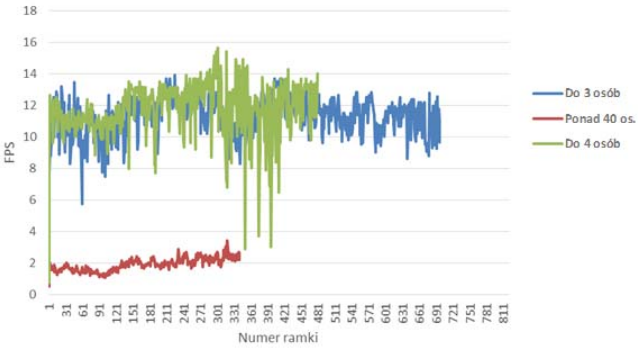
Na rysunku 10 przedstawiono wynik badania wpływu rozmiaru ramki obrazu na szybkość przetwarzania (FPS).



Rys.10. Wpływ rozmiaru ramki na szybkość przetwarzania (FPS)

Z uzyskanych danych wynika, że rozmiar ramki ma bardzo istotne znaczenie w trakcie detekcji obrazów. W przypadku małego rozmiaru tj. 320x256 pikseli, uzyskano na pewnych odcinkach analizę pełnego strumienia wideo w czasie rzeczywistym. W przypadku większych rozdzielczości obrazu szybkość przetwarzania odpowiednio maleje. Warto podkreślić, że w przypadku mniejszych rozdzielczości jakość detekcji się pogarsza w stosunku do małych obiektów analizowanej sceny (ich obraz składa się z niewielkiej liczby pikseli i nie zawsze może być poprawnie rozpoznany). Biorąc pod uwagę, że analiza odbywa się tylko w zdefiniowanym obszarze RoI oraz to, że wszystkie wyniki detekcji są filtrowane w ramach jednej klasy, która ma dość ściśle określone typowe wymiary (sylwetka człowieka), przy odpowiednim dobraniu obszaru RoI zmniejszenie rozdzielczości nie powinno doprowadzić do istotnego pogorszenia jakości detekcji.

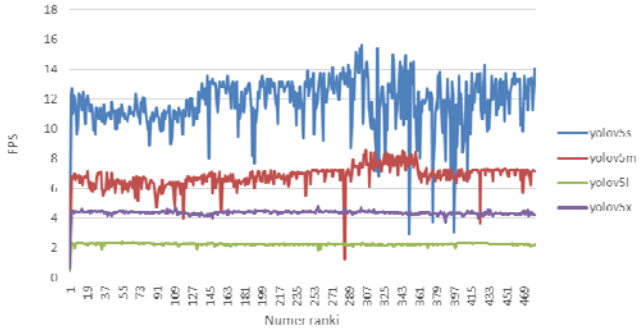
Na rysunku 11 przedstawiono wynik badania wpływu liczby wykrywanych osób na szybkość przetwarzania (określanej poprzez liczbę przetwarzanych klatek obrazu na sekundę – FPS). Jak widać, znaczna liczba osób obecnych w analizowanej scenie w istotny sposób zmniejsza szybkość przetwarzania.



Rys.11. Wpływ liczby osób w scenie na szybkość przetwarzania

Oznacza to, że w przypadku takich sytuacji konieczne jest zastosowanie sprzętu o większej mocy obliczeniowej. W praktyce oznacza to szybszy procesor GPU (lub np. skorzystanie z chmurowych zasobów obliczeniowych), natomiast jako podstawowy procesor CPU wystarczy jednostka o przeciętnej szybkości.

W trakcie analizy przetestowano kilka rodzin algorytmów w oparciu o plik wideo z maksymalnie 4 osobami. W zależności od rodzaju algorytmu detektora oraz wybranych wag otrzymano stosunkowo duży zakres wyników (rys. 12).



Rys.12. Porównanie wartości FPS dla różnych wersji YOLOv5

Każdy kolejny zestaw wag zaoferowany dla YOLOv5 wiąże się ze wzrostem liczby warstw detektora, a przez to dwukrotnym spadkiem prędkości przetwarzania, jak to jest widoczne na rysunku 12. Dla yolov5s są to 224 warstwy, yolov5m – 308 warstw, yolov5l – 392 warstwy, yolov5x – 476 warstw. Ma to również odzwierciedlenie w dokładności, nawet z punktu widzenia oceny wizualnej (znikanie obwiedni detekcyjnych, pomyłki algorytmu dla „lżejszych" wag).

Mówiąc o różnorodności algorytmów detektorów, należy również wspomnieć o SSD, który jest głównym „konkurentem" YOLO. Biorąc pod uwagę, że nie ma oficjalnych repozytoriów z realizacją SSD na PyTorch, do porównania wykorzystano projekt z artykułu [10] z realizacją na Tensorflow. Otrzymane wyniki świadczą o tym, że algorytm YOLOv5 jest dwa razy szybszy niż SSD MobileNet v1 (średnio 11 FPS vs 5 FPS).

Ostatnim elementem wykonywanych analiz wydajnościowych było porównanie szybkości przetwarzania GPU (NVIDIA GeForce 940 MX) z szybkością uzyskiwaną na standardowym CPU (Intel Core i7-7500U 2.7GHz with Turbo Boost up to 3.5 GHz). Otrzymane wyniki wykazały, że stosowanie GPU nawet o bardzo przeciętnej obecnie wydajności pozwala prawie pięciokrotnie zwiększyć szybkość przetwarzania w stosunku do tego, co zostało osiągnięte za pomocą CPU (średnio 13 FPS vs 3 FPS dla ramki 800x600 pikseli).

Z punktu widzenia części sprzętowej szybkość przetwarzania zależy w bezpośredni sposób od liczby rdzeni CUDA. Staje się oczywiste, że rozwiązania dla systemów wbudowanych np. typu NVIDIA Jetson w znacznym stopniu przegrywają z nowszymi dedykowanymi kartami graficznymi. Wykorzystana w opisanym rozwiązaniu karta graficzna NVIDIA GeForce 940MX posiada 384 rdzeni CUDA z rodziny Maxwell, co jest analogiczne do procesora NVIDIA Jetson Xavier NX [11]. Biorąc pod uwagę, że współczesne przeciętne karty graficzne mają po kilka tysięcy rdzeni CUDA oraz że wiążący się z tym pobór mocy może sięgać kilkaset watów, stosowanie rozwiązań typu embedded w projektach ze sztuczną inteligencją (zwłaszcza związanych z zaawansowaną analizą obrazów) jest stosunkowo ograniczone. Tym niemniej istnieją biblioteki przeznaczone dla mikrokontrolerów realizujące dość zaawansowaną analizę sygnałów akustycznych oraz podstawową analizę obrazów wideo [12, 13].

Podsumowanie

W systemach wizyjnych, wykorzystujących algorytmy sztucznej inteligencji, szybkość przetwarzania zależy bezpośrednio od takich parametrów jak rozmiar ramki, rodzaj algorytmu, wybrane wagi oraz liczba obiektów występujących na obrazie. Najszybszym algorytmem w trakcie badań był yolov3-tiny, chociaż jakość detekcji była stosunkowo niska z powodu mniejszej liczby warstw. Z punktu widzenia rzeczywistego systemu do monitorowania dystansu społecznego najbardziej optymalnym rozwiązaniem byłby YOLOv5 (z wagami yolov5s) dlatego, że wykazał on stosunkowo dużą dokładność przy dość wysokiej liczbie FPS (np. w porównaniu z yolov3, który jest szybszy, ale jakość detekcji jest dość niska). Dla karty graficznej GeForce 940MX czas rzeczywisty był ciągle utrzymywany w przypadku ramki z rozmiarem 320x256 pikseli, co nie powodowało spadku jakości detekcji.

Podczas analizy obrazu szczególną uwagę należy zwrócić na umiejscowienie kamery. Na otwartych przestrzeniach analizowana scena powinna w pewien sposób stanowić odwzorowanie poziomej płaszczyzny, zidentyfikowane obwiednie powinny być znacząco mniejsze od zaznaczonego obszaru RoI.

Testy z kamerą pokazały, że tolerancja podczas pomiaru dystansu na podstawie liczby pikseli jest rzędu 8-10% (wartość określona na podstawie liczby przypadków niepoprawnie wyznaczonego dystansu i związanego z tym oznaczenia osób niewłaściwym kolorem ramki). Uzyskane wyniki mogą być stosunkowo wiarygodne, żeby stanowić rozwiązanie systemu do analizy zachowania dystansu społecznego.

Badania zostały zrealizowane w ramach pracy nr WZ/WE-IA/1/2020 w Politechnice Białostockiej i sfinansowane z subwencji badawczej przekazanej przez Ministra Edukacji i Nauki. Artykuł wykorzystuje koncepcję stanowiącą część pracy dyplomowej magisterskiej mgr inż. Viktora Zanozina napisanej pod kierunkiem dr inż. Andrzeja Zankiewicza.
Na rysunku 1 wykorzystano grafiki (ikony) dostępne w ramach Cisco Free Icons Library.

***Autorzy**: mgr inż. Viktor Zanozin, E-mail: wiktor.zanozin@gmail.com; dr inż. Andrzej Zankiewicz, Politechnika Białostocka, Wydział Elektryczny, ul. Wiejska 45D, 15-351 Białystok, E-mail: a.zankiewicz@pb.edu.pl.*

LITERATURA

[1] World Health Organization, *Considerations for implementing and adjusting public health and social measures in the context of COVID-19, Interim guidance*, WHO 2021, WHO reference number: WHO/2019-nCoV/Adjusting_PH_measures/2021.1
[2] Gonzalez R.C., Woods R.E., *Digital Image Processing*, Fourth Edition, Pearson Education Limited, 2018
[3] https://opencv.org/, dostęp 23.06.2022
[4] Villan A.F., *Mastering OpenCV 4 with Python*, Packt Publishing, Birmingham, 2019
[5] http://www.eletel.p.lodz.pl/pstrumil/po/rozpoznawanie.pdf, dostęp 23.06.2022
[6] https://cocodataset.org/, dostęp 23.06.2022
[7] https://github.com/ultralytics/yolov5, dostęp 23.06.2022
[8] Rosebrock A., *4 Point OpenCV getPerspective Transform Example*, https://www.pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/, dostęp 23.06.2022
[9] Ferryman J., Ellis A., PETS2010: Dataset and Challenge, *7th IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2010, 143-150
[10] Anwar A., *Using Python to Monitor Social Distancing in a Public Area*, https://towardsdatascience.com/monitoring-social-distancing-using-ai-c5b81da44c9f, dostęp 23.06.2022
[11] https://www.nvidia.com/en-us/autonomous-machines/jetson-store/, dostęp 23.06.2022
[12] Suder J., Możliwości przetwarzania sekwencji wizyjnych w systemach wbudowanych, *Przegląd Elektrotechniczny*, 98 (2022), nr 1, 188-191
[13] Sakr, F.; Bellotti, F.; Berta, R.; De Gloria, A. Machine Learning on Mainstream Microcontrollers. *Sensors* 2020, 20, 2638