

A G H

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

WYDZIAŁ ELEKTRONIKI, INFORMATYKI I TELEKOMUNIKACJI

Praca magisterska

Analiza ruchu pieszego i dostaw towarów w strefach ograniczonego ruchu

Analysis of pedestrian traffic and goods deliveries in Limited Traffic Zones

Autorka:

Kierunek studiów:

Typ studiów:

Opiekun pracy:

Klaudia Katarzyna Bednarz

Teleinformatyka

Stacjonarne

dr hab. inż. Mikołaj Leszczuk

Kraków, 2021

*Serdecznie dziękuję mojemu promotorowi
dr hab. inż. Mikołajowi Leszczukowi za
pomoc i czas poświęcony w realizacji tej
pracy.*

Spis treści

1. Wprowadzenie	7
1.1. Opis problemu	7
1.2. Cele pracy	8
1.3. Zawartość pracy	9
2. Detekcja obiektów - stan wiedzy	11
2.1. Popularne metody detekcji	11
2.1.1. Sliding window	11
2.1.2. Selective Search	12
2.1.3. R-CNN	13
2.1.4. YOLO	14
2.2. YOLOv5 - opis	15
2.2.1. Modele YOLOv5	16
2.2.2. YOLOv5 API	17
2.2.3. Metryki modelu	18
3. Analiza danych wejściowych	21
3.1. Wybór lokalizacji poddanych badaniom	21
3.2. Obróbka danych	22
3.2.1. Zbieranie danych	22
3.2.2. Przetwarzanie danych	23
3.3. Trenowanie modelu	24
3.4. Testy modelu	25
4. Implementacja licznika	27
4.1. Norfair tracker	27
4.2. Działanie licznika	28
5. Uzyskane wyniki	31
5.1. Środowisko obliczeniowe	31
5.2. Opis analizowanych scenariuszy	31
5.2.1. Podsumowanie detekcji wykonanej w zwykły dzień	32
5.2.2. Największa jednocześnie ilość obiektów w interwale 15 min	36
5.2.3. Czasy postoju pojazdów	38
5.2.4. Porównanie dni świątecznych ze zwykłymi	39
6. Podsumowanie	43

1. Wprowadzenie

We współczesnym świecie nieustannie rośnie potrzeba opracowywania coraz to nowszych i bardziej zaawansowanych systemów, dzięki którym gromadzenie i analiza danych stają się szybsze. Każdego dnia, używając aplikacji mobilnych, robiąc zakupy online, czyli w szerokim ujęciu korzystając z dobrodziejstw internetu, przyczyniamy się do powiększania ilości przetwarzanych danych. Z tego powodu wprowadzane są rozwiązania, które pozwalają na automatyzację związań z tym procesów i tworzenie modeli analitycznych, eliminując przy tym „manualną” pracę wykonywaną przez człowieka. Ostatnimi czasy coraz większe zainteresowanie budzi *Data Science* - dziedzina, która ciągle się rozwija i daje szerokie pole do usprawnienia żmudnych procesów związanych z przetwarzaniem i interpretacją danych oraz wykorzystująca techniki uczenia maszynowego.

Uczenie maszynowe i sztuczna inteligencja prawdziwie zawładnęły światem eksploracji danych i znajdują zastosowanie w wielu sektorach życia publicznego. Współczesne systemy bankowe pozwalają na szybkie przetwarzanie i interpretację danych klientów oraz umożliwiają identyfikowanie potencjalnych oszustów czy wyłudzeń. Dzięki wykorzystaniu tej technologii w dziedzinie ochrony zdrowia obserwujemy szeroki rozwój telemedycyny. Uczenie maszynowe wykorzystywane jest także w administracji publicznej, gdzie znajduje zastosowanie np. w instytucjach zajmujących się zapewnianiem bezpieczeństwa obywateli oraz w transporcie - np. do wyznaczania efektywnych tras dla dostawców[6].

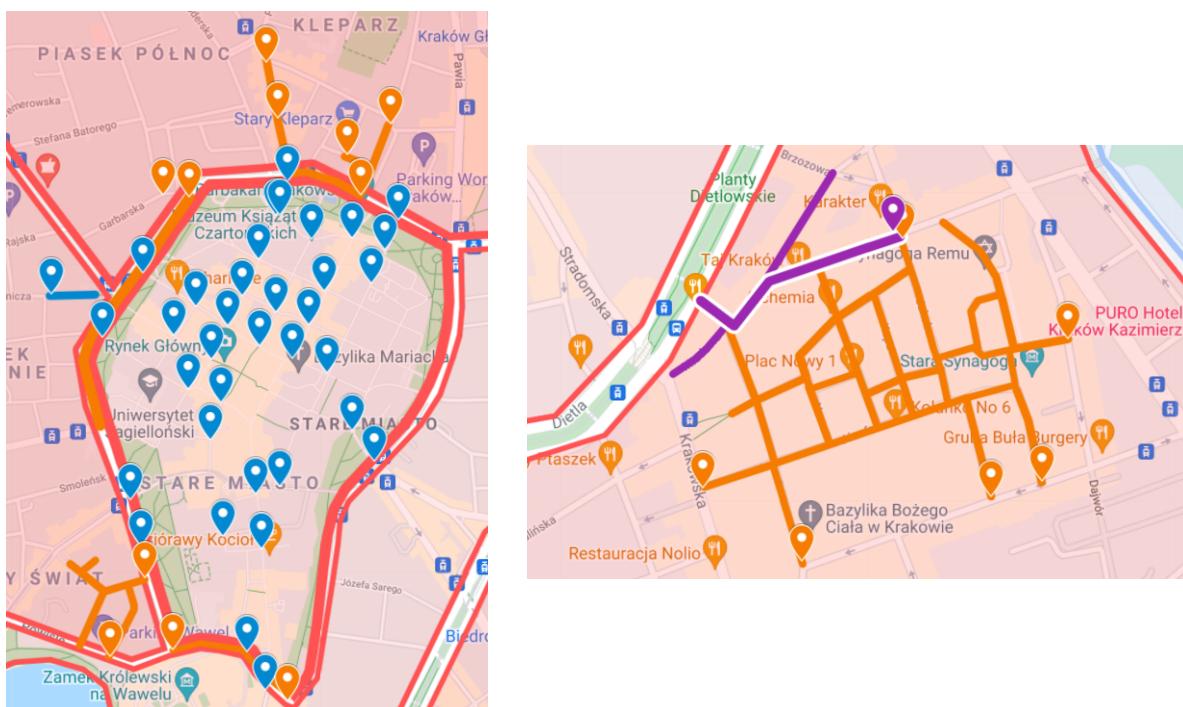
Techniki uczenia maszynowego wykorzystywane są także w zagadnieniu, z którym bezpośrednio związana jest niniejsza praca - widzeniu komputerowym (ang. *Computer vision*). Zadaniem tej dziedziny jest symulowanie procesu ludzkiego widzenia. Dzięki algorytmom uczącym się, możliwe jest odwzorowanie odkrywania i interpretacji zjawisk, a także klasyfikacji obiektów, które zachodzą podczas analizy otaczającego nas świata, poprzez zmysł wzroku. Gromadzenie informacji o rzeczywistości w tym wypadku odbywa się na podstawie analizy obrazu lub sekwencji obrazów. Dane pozyskiwane ze sceny poddanej interpretacji mogą być w tym procesie dzielone na mniejsze podzbiory, modyfikowane i wielokrotnie filtrowane. Wszystko po to, aby bez udziału człowieka udało się poprawnie „zrozumieć” kontekst sytuacji i np. rozpoznać sytuację zagrożenia albo zebrać dane do statystyki[2].

1.1. Opis problemu

Tematyka niniejszej pracy magisterskiej bezpośrednio nawiązuje do ostatniego z wymienionych przykładów na zastosowanie widzenia komputerowego w życiu społecznym. Problem, któremu niniejszy projekt wychodzi naprzeciw to potrzeba kontroli i gromadzenia wiedzy na temat ruchu na drogach po części wyłączonych z użytkowania (tzw. strefy ograniczonego ruchu). Dzięki analizie publicznie dostępnych nagrani z kamer zainstalowanych w tych miejscowościach,

możliwe jest zebranie danych i statystyk, które mogą posłużyć jako materiał do podejmowania przez władze miasta dalszych decyzji w zakresie organizacji ruchu drogowego.

Strefy ograniczonego ruchu w Krakowie obejmują teren Starego Miasta oraz Kazimierz (wyznaczona przez ulice zaznaczone na pomarańczowo i fioletowo na Rys.1.1.). W strefach obowiązują przepisy dotyczące dozwolonego ruchu pojazdów, godzin dostaw oraz opłat postojowych. Restrykcje te mają na celu ograniczenie ruchu samochodowego w ścisłym centrum oraz promowanie ekologicznych środków transportu jako sposób przemieszczania się na terenie miasta.



Rys. 1.1. Obowiązujące strefy ograniczonego ruchu w Krakowie[19].

Analiza zachowania pieszych i ruchu pojazdów na terenie strefy daje możliwość zapoznania się z faktycznym stanem rzeczy w kontekście przestrzegania przepisów drogowych, a także planowania usprawnień. Posiadanie takiej wiedzy niesie za sobą szereg potencjalnych korzyści, takich jak np. optymalizacja częstotliwości planowanych dostaw, wprowadzenie dodatkowych środków zapewniających bezpieczeństwo pieszym, czy dodatkowego oznakowania.

1.2. Cele pracy

Niniejsza praca stanowi rozwinięcie istniejącego projektu, którego przedmiotem było opracowanie algorytmu wykrywającego i zliczającego rowerzystów, pojawiających się na udostępnionych filmach z krakowskich kamer miejskich.

Celem niniejszej pracy magisterskiej było stworzenie systemu graficznego rozpoznawania obiektów ruchomych, który pozwoliłby na analizę ruchu pieszych oraz dostaw towarów w strefach ograniczonego ruchu. Podczas jej realizacji korzystano z danych pochodzących z publicznie dostępnych kamer internetowych. Do jej implementacji posłużył popularny algorytm detekcji obrazów - *YOLO*.

Praca została wykonana w konsultacji i z pozyskaniem danych z Urzędu Miasta Krakowa oraz miejskich jednostek. Projekt ma być pomocą w ocenie inwestycji prowadzonych przez władze miasta lub planowaniu przyszłych projektów - kod źródłowy, wyniki badań oraz część opisowa pracy zostaną przekazane do biura Miejskiego Inżyniera Ruchu (MIR).

1.3. Zawartość pracy

W rozdziale drugim zostało umieszczone wprowadzenie do zagadnienia detekcji obiektów oraz zgromadzono tam informacje na temat różnych znanych metod detekcji. Opisano schemat działania każdej z nich, w szczególności skupiając się na *YOLOv5*, czyli metodzie, która posłużyła do wykonania części praktycznej niniejszej pracy magisterskiej.

Rozdział trzeci został poświęcony procesowi kompletowania i przetwarzania danych. Opisano w nim chronologicznie postępy w pracy nad projektem, napotkane problemy oraz sposób radzenia sobie z nimi, a także wykorzystane narzędzia. Dodatkowo, zawarto w nim informacje na temat procesu uczenia modelu, a także przeprowadzonych testów i zbieranych metryk.

W rozdziale czwartym opisano działanie stworzonego licznika obiektów, który opiera się o proste zliczanie rezultatów zwracanych przez mechanizm trackowania obiektów. Opisane zostały tam mechanizmy zaimplementowane w celu wykrywania nieruchomych obiektów oraz sposób korekcji błędnie wykrytych obiektów.

Rozdział piąty poświęcono analizie i interpretacji wyników detekcji uzyskanych dzięki wykorzystaniu opracowanego licznika poprzez uruchomienie go na filmach z kamer zainstalowanych w różnych lokalizacjach w centrum Krakowa. Opisano w nim wybrane scenariusze badawcze oraz przeprowadzono przegląd otrzymanych wyników.

W ostatnim rozdziale zamieszczono podsumowanie pracy oraz wnioski z przeprowadzonych badań. Zawarto w nim również propozycje usprawnień dla stworzonego licznika.

2. Detekcja obiektów - stan wiedzy

Detekcja obiektów to zagadnienie związane z rozpoznawaniem wzorców. Znajduje zastosowanie w analizie i interpretacji obrazów (rozumieniu scen i rozpoznawaniu działań)[8]. Polega na lokalizowaniu i klasyfikowaniu obiektów na obrazach i zdjęciach, gotowych plikach wideo oraz poprzez analizę wideo w czasie rzeczywistym. Każdy z tych sposobów wymusza inne podejście do wykrywania obiektów:

- obraz - wyszukiwanie obiektów na wyodrębnionej scenie jest najprostszym z wymienionych ujęć. Przebieg procesu detekcji polega w uproszczeniu na właściwym odczycie obrazu i przekazaniu go do algorytmu. Wynikiem przetwarzania jest obraz, na którym prawidłowo rozpoznane zostały przedmioty należące do odpowiadających im kategorii.
- wideo - detekcja obiektów w tym wypadku wymaga podzielenia filmu na klatki (ang. *frames*) oraz przekazania ich kolejno do algorytmu wykrywania aż do zakończenia pliku. Pojedyncze klatki wynikowe są zbierane i zapisywane w celu otrzymania nowego wideo z prawidłowo wychwyconymi przedmiotami.
- *real-time by camera* - podejście to wymaga w zasadzie tego samego zestawu czynności jak w przypadku detekcji wideo. W tym wypadku jednak pojedyncze klatki nie są odczytywane z pliku, ale bezpośrednio z kamery. Przetworzone klatki są „odkładane” w osobnym oknie, tak aby możliwa była obserwacja zmian w czasie rzeczywistym.

Zwykle wizualizacja wykrytych obiektów odbywa się poprzez otoczenie ich prostokątnymi ramkami *bounding box*. Obiekty, które posiadają te same cechy zostają przypisane do jednej kategorii. Prawidłowe przypisanie kategorii danemu obiekowi może być utrudnione, w przypadku kiedy mamy do czynienia z np. zakłóceniami tła lub zmianami perspektywy[8].

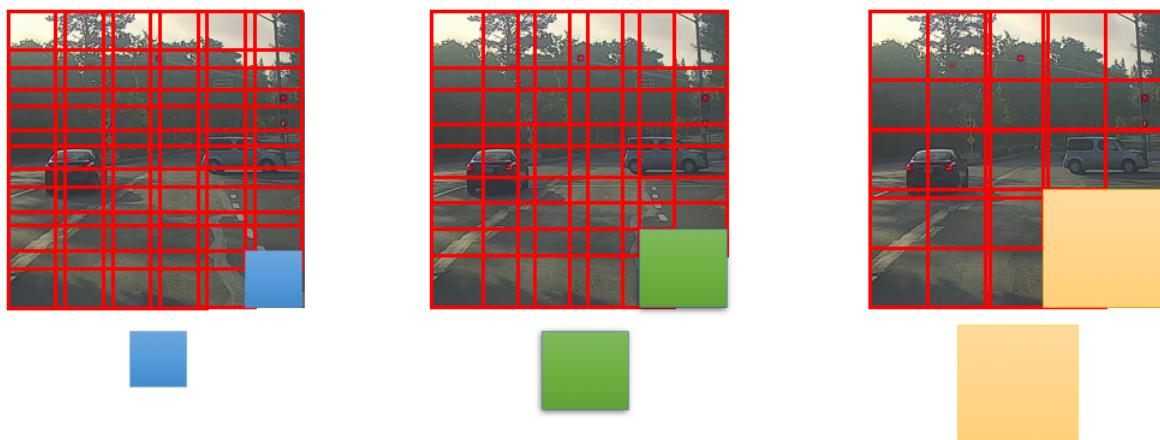
2.1. Popularne metody detekcji

Sekcja ta została poświęcona opisowi popularnych algorytmów stosowanych w procesie detekcji obiektów. Szczególną uwagę poświęcono metodzie wykorzystanej w implementacji licznika opracowanego w ramach niniejszego projektu - *YOLO*.

2.1.1. Sliding window

Detekcja obiektów może odbywać się poprzez porównywanie punktów charakterystycznych obrazu będącego przedmiotem badań i obrazu referencyjnego. Niekiedy, stosując się do takigo

podejścia, możemy w wyniku otrzymać zbyt dużo podobnych do siebie obiektów (tzw. problem koncentracji cech), co utrudnia wykrycie konkretnego przedmiotu zainteresowania. Metoda okna przesuwnego (ang. *sliding window*) pozwala na wyeliminowanie tego problemu poprzez ograniczenie cech charakterystycznych, branych pod uwagę przy pojedynczym porównaniu wycinka obrazu z obrazem docelowym. Okno o określonym, a najlepiej - odpowiednio dla danego problemu zoptymalizowanym rozmiarze, jest przesuwane w obrębie przeszukiwanego obrazu w każdej iteracji. Im większe okno, tym mniejsza złożoność detekcji, ale i też więcej objętych nim cech charakterystycznych - występuje problem z dopasowaniem. Zbyt małe okno nie zawierają wystarczającej liczby cech charakterystycznych i wiążą się z większym nakładem obliczeniowym. Wybór optymalnego rozmiaru okna warto podejmować bazując na rozmiarze obrazu będącego przedmiotem zainteresowania detekcji. Można manipulować także odległością przesuwu, modyfikując w ten sposób liczbę iteracji, tak aby nie ryzykować utraty informacji i jednocześnie minimalizować złożoność obliczeniową[9].



Rys. 2.1. Ilustracja działania metody okna przesuwnego z wykorzystaniem różnych wielkości okna[3].

Wyszukiwanie metodą okna przesuwnego następuje po wytrenowaniu klasyfikatora na zbiorze próbek pochodzących z obrazu referencyjnego oraz losowych obiektach. Próbki pozyskane z porównywanych w ten sposób obrazów należą do osobnych klas - przedmiot detekcji zawiera tzw. przykłady pozytywne, natomiast przeszukiwane obiekty losowe, które nie są dla nas interesujące - negatywne. Po zakończeniu trenowania oraz wyborze rozmiaru okna i odległości przesuwu, wyuczony klasyfikator może posłużyć do wyszukiwania pożądanego przedmiotu detekcji na obrazie docelowym[15].

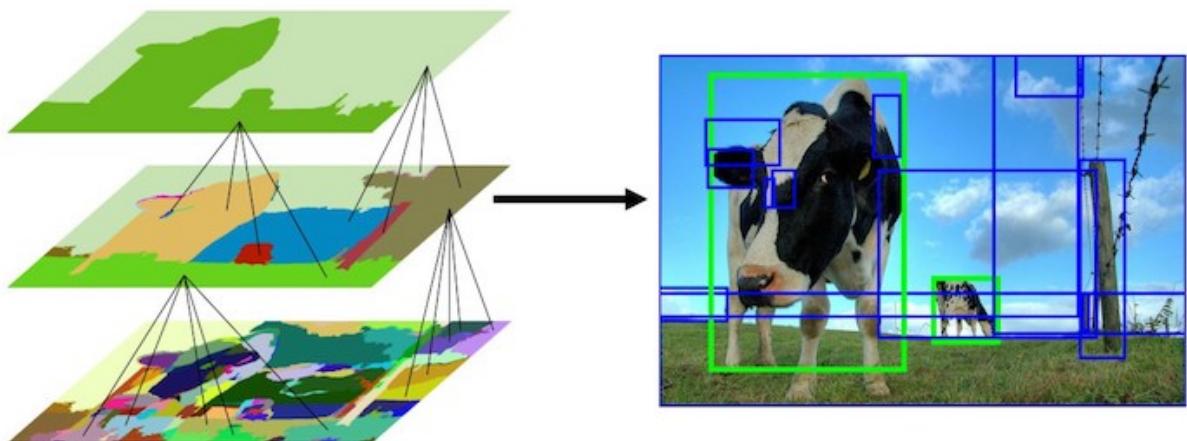
2.1.2. Selective Search

Metoda *Selective Search* jest podejściem należącym do grupy algorytmów polegających na wyznaczaniu podobnych do siebie regionów przeszukiwanego obrazu (ang. *Region proposal algorithms*) w procesie tzw. segmentacji. Miara podobieństwa regionów ustalana jest na podstawie kombinacji kryteriów:

- podobieństwa kolorów,

- podobieństwa tekstur,
- rozmiaru i miary wypełnienia.

Algorytm *Selective Search* polega na hierarchicznym generowaniu propozycji regionów rozpoczynając od nadsegmentacji obrazu, stopniowo zwiększając rozmiar proponowanych regionów. Uruchamiany w różnych warunkach, np. w komplementarnych przestrzeniach barw, pozwala wygenerować propozycje regionów wartych przeszukania, o zróżnicowanych rozmiarach i proporcjach[13]. Metoda ta wiąże się z ryzykiem wyznaczenia licznych „fałszywych” segmentów, które w późniejszych krokach zostaną odrzucone przez algorytm rozpoznawania obiektów. Wraz z rosnącą liczbą nieprawidłowo wyznaczonych segmentów rośnie czas wykonywania algorytmu. Jest to jednak spodziewany i akceptowalny rezultat - tak długo, dopóki *Selective Search* zwraca regiony, w których istnieją poszukiwane obiekty.



Rys. 2.2. Przykład przebiegu procesu segmentacji hierarchicznej obrazu[14].

2.1.3. R-CNN

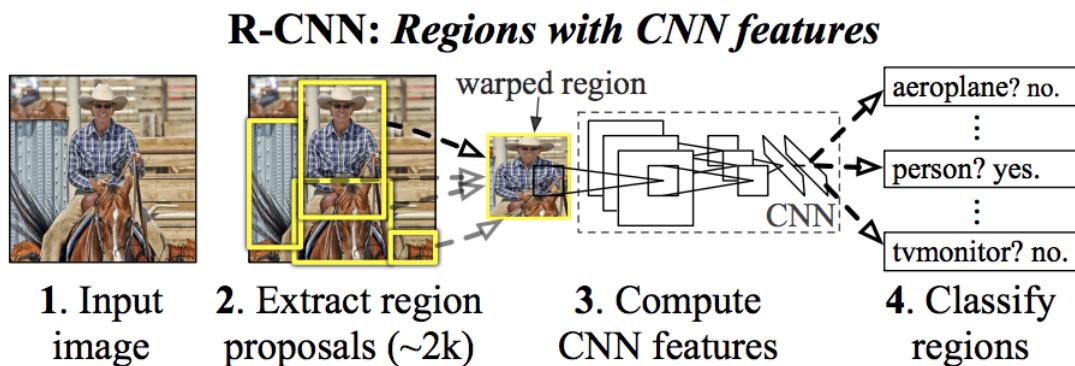
Metoda R-CNN (ang. *Region Based Convolutional Neural Networks*) podczas rozwiązywania problemu detekcji może wykorzystywać dowolne podejście do wyodrębnienia z obrazu 2000 regionów - zarówno *sliding windows* jak i wyszukiwanie selektywne. Istotą metody jest przedstawienie regionu za pomocą wektora cech, stworzonego dzięki konwolucyjnej sieci neuronowej (*CNN*), który jest uproszczoną reprezentacją analizowanego obrazu[10].

Sieci konwolucyjne zapewniają szybsze tempo uczenia w porównaniu ze zwykłymi sieciami neuronowymi. Ich działanie można opisać jako filtrację danych uczących i eksponowanie rozpoznanych cech charakterystycznych. Sieci konwolucyjne są szczególnie dedykowane do zadań wizyjnych, takich jak rozpoznawanie, klasyfikacja czy rekonstrukcja.

Proces wyodrębniania wektora cech z obrazu może przebiegać następująco:

1. Przekazanie obrazu wejściowego do sieci CNN.
2. Przepuszczenie obrazu przez kilka warstw konwolucyjnych oraz warstw łączących - w każdym kroku następuje filtrowanie danych w celu wyodrębnienia map cech wspólnych dla różnych regionów, które posłużą do rozpoznawania wzorców.

3. Redukcja rozmiaru - łączenie map cech pochodzących z różnych propozycji regionów w wektory dotyczące większych obszarów.
4. Przekazanie wektora cech do „w pełni połączonej warstwy” (*fully connected layer*), która zwraca wynik klasyfikacji[1].



Rys. 2.3. Schemat działania R-CNN[11].

Po klasyfikacji następuje postprocessing służący wyznaczeniu *bounding box*-ów otaczających obiekty, pozbyciu się duplikatów oraz ponownej oceny pól na podstawie innych obiektów obecnych na obrazie.[18]

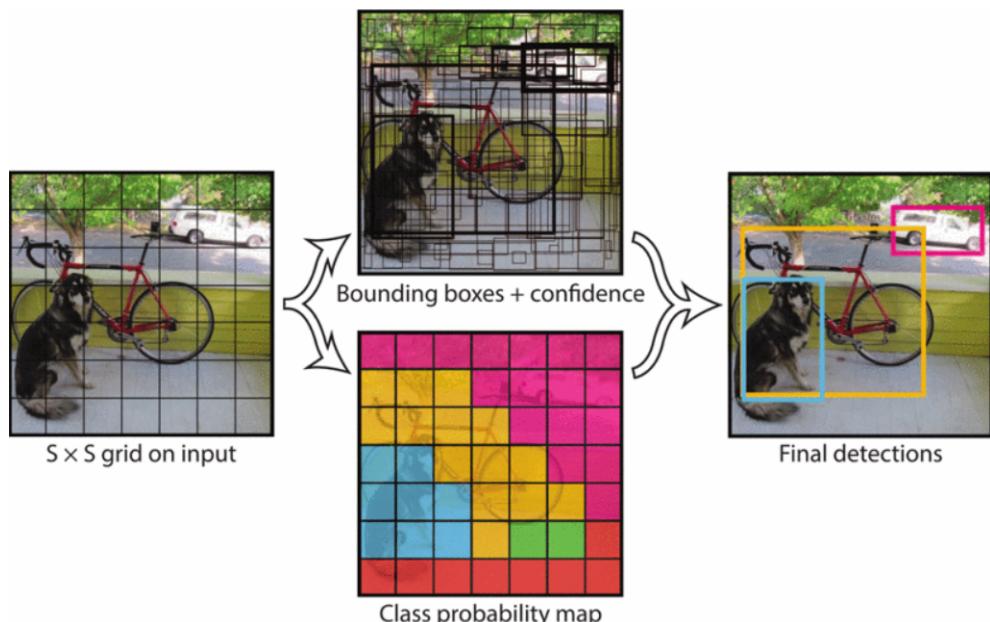
- Fast R-CNN - szybsza metoda detekcji bazująca na algorytmie R-CNN, wykluczająca konieczność wyodrębniania z obrazu 2000 regionów. W pierwszym kroku algorytmu obraz wejściowy jest przekazywany do CNN. W wyniku tej operacji generowana jest konwolucyjna mapa cech, na podstawie której za pomocą wyszukiwania selektywnego można wyodrębnić propozycje regionów. W takim podejściu operacja konwolucji jest wykonywana tylko raz.
- Faster R-CNN - algorytm całkowicie eliminuje wykorzystanie czasochłonnego wyszukiwania selektywnego. Działa podobnie jak Fast R-CNN, ale po wygenerowaniu konwolucyjnej mapy cech propozycje regionów są wyodrębniane za pomocą oddzielnej sieci neuronowej.

2.1.4. YOLO

Metoda *YOLO* (ang. *You Only Look Once*) wykorzystuje konwolucyjne sieci neuronowe i polega na lokalizowaniu obiektów oraz etykietowaniu ich w tym samym czasie. Umożliwia wykrywanie wielu obiektów różnych kategorii na jednym obrazie. *YOLO* traktuje problem rozpoznawania obiektów jako pojedynczy problem regresyjny. Do detekcji wykorzystywana jest sieć konwolucyjna, która odpowiada za równoczesną predykcję *bounding box*-ów oraz prawdopodobieństwa, że wyznaczony obiekt należy do danej kategorii.

YOLO działa w następujący sposób:

1. Na początku działania algorytmu następuje podział obrazu wejściowego za pomocą „siatki” na wiele małych komórek.
2. W każdej z komórek w tym samym czasie (zgodnie z nazwą metody *You Only Look Once*) rozpoznawane są fragmenty obwiedni (*bounding box-ów*), otaczających potencjalne obiekty. Jednocześnie określone jest prawdopodobieństwo, z jakim wyznaczony wycinek obwiedni otacza faktyczny obiekt danej klasy. Proces ten „zwraca” rozkład prawdopodobieństwa po wszystkich klasach, na których trenowana była sieć.
3. Wynikiem działania algorytmu jest obraz, na którym wyznaczono *bounding box-y* o różnej grubości linii - im grubsza linia tym większe prawdopodobieństwo, że wewnątrz znajduje się obiekt.
4. Na koniec odrzucone w procesie *non-maximum suppression* zostają najcieńsze *bounding box-y* (np. te z prawdopodobieństwem wystąpienia tam obiektu mniejszym niż 30%)[18].



Rys. 2.4. Model systemu detekcji oparty o YOLO[18].

YOLO jest najszybszą znaną do tej pory metodą detekcji obiektów. Istnieje kilka wersji algorytmów z rodziny YOLO, które różnią się między sobą pod względem architektury i wydajności. Najnowsza oficjalnie opublikowana wersja, wykorzystana podczas implementacji części technicznej niniejszej pracy, to YOLOv5.

2.2. YOLOv5 - opis

YOLOv5 to model detekcji obiektów stworzony przez firmę *Ultralytics* - autorów m.in. YOLOv3, opracowanego z wykorzystaniem *PyTorch* - popularnego pythonowego framework'a znajdującego zastosowanie w rozwoju zagadnień uczenia maszynowego. Deweloperzy *Ultralytics* zajmują się prowadzeniem badań nad sztuczną inteligencją i nowoczesnymi metodami

detekcji oraz udostępnianiem rozwiązań w zakresie uczenia maszynowego, które nie wymagają od użytkowników wdrażania się w złożone architektury sieci neuronowych. Repozytorium *YOLOv5* jest open-source'owe, a udostępnione modele mogą być wykorzystywane i rozbierane przez programistów w ramach własnych projektów.

2.2.1. Modele YOLOv5

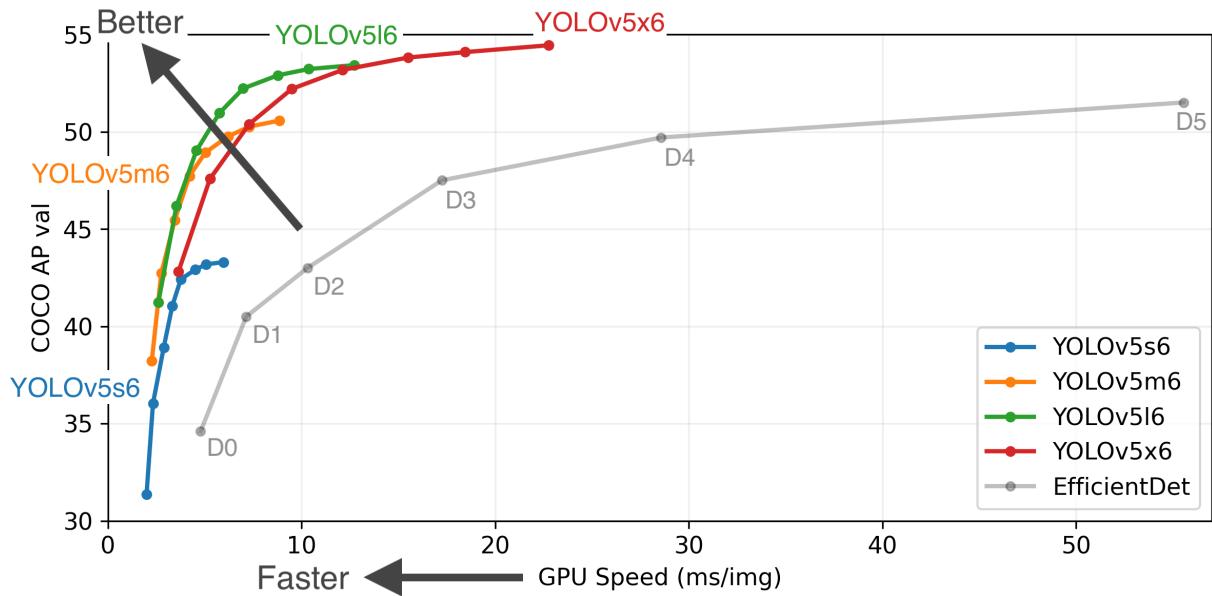
YOLOv5 udostępnia 4 modele wytrenowane na zbiorze danych *COCO - Common Objects in Context*. Jest to obszerna baza specjalnie wyselekcyjowanych zdjęć przygotowanych do uczenia modeli mających na celu przetwarzanie obrazów. Na zebranych zdjęciach zawarte są różne obiekty otoczone *bounding box*-ami, które oznaczone są odpowiadającymi im etykietami. Podstawowa struktura bazy należącej do zbioru *COCO* opiera się na formacie *JSON* i składa się z 5-ciu sekcji.

```
{
  "info": {...},
  "licenses": [...],
  "images": [...],
  "categories": [...],
  "annotations": [...]
}
```

Rys. 2.5. Opis struktury bazy należącej do zbioru *COCO* w postaci *JSON*-a[4].

- *info* - zawiera podstawowe informacje na temat zbioru - jak wersja, rok powstania, adres *URL*, pod którym jest dostępny,
- *licenses* - adres *URL*, identyfikator oraz nazwa licencji,
- *images* - zawiera takie dane jak nazwa obrazu, jego wymiary, numer licencji, *URL*,
- *categories* - czyli klasy, do jakich zakwalifikowane są obiekty istniejące na obrazie, w sekcji zawarte są dane: id, nazwa kategorii oraz nad-kategoria,
- *annotations* - najważniejsza sekcja opisująca zbiór - zawiera właściwe informacje potrzebne do przekazania do modelu, takie jak m.in. koordynaty *bbox*-ów, powierzchnia właściwa zajmowana przez rozpoznawany obiekt (ang. *segmentation mask*), identyfikator obrazu oraz klasy[4].

Wytrenowane na przetworzonych w ten sposób zbiorach danych modele *YOLOv5* występują w czterech wariantach: *s* (*small*), *m* (*medium*), *l* (*large*) oraz *x* (*extra large*). Nazwy modeli odpowiadają ilości danych przekazanych do trenowania modelu. Różnią się one między sobą m.in: dokładnością i wydajnością[17].



Rys. 2.6. Porównanie wariantów YOLOv5[17].

2.2.2. YOLOv5 API

Opracowana przez *Ultralytics* implementacja YOLOv5 jest publicznie dostępna w serwisie *GitHub* oraz jest ciągle rozwijana. W repozytorium projektu znajdują się takie pliki jak:

- tutorial udostępniony w formie notatnika *Google Colaboratory*,
- plik *requirements.txt*, zawierający listę bibliotek, które należy zainstalować - niezbędnych do uruchomienia projektu,
- przykładowe skrypty *train.py*, *test.py* oraz *detect.py*, w których zaprezentowano w jaki sposób można korzystać z API.

API YOLOv5 pozwala na wytrenowanie własnego modelu, poprzez rozszerzenie jednego dostępnych modeli, które zostały opisane w rozdziale 2.2.1. Aby to zrealizować należy zebrać kolekcję zdjęć, przedstawiających obiekty, które chcemy rozpoznawać za pomocą utworzonego modelu. Dla każdego z przygotowanych zdjęć musi istnieć odpowiadający mu plik w formacie *YOLO*. To plik *.txt* o takiej samej nazwie jak plik ze zdjęciem, w którym każda linia jest reprezentacją innego obiektu rozpoznanego na obrazie. Takie pliki można wygenerować wykorzystując specjalne narzędzia służące do „etykietowania” obiektów widocznych na zdjęciach, takie jak np. *makesense.ai*[7].

W pliku w formacie *YOLO* następujące po sobie kolumny oznaczają kolejno:

- klasę obiektu,
- współrzędne x i y punktu przecięcia przekątnych *bounding-box*'a otaczającego obiekt,

- szerokość *bounding-box'a*,
- wysokość *bounding-box'a*.

0	0.784120	0.398271	0.113345	0.46782
0	0.767861	0.5718020	0.016309	0.061992
1	0.744438	0.396429	0.017167	0.036800
0	0.0480258	0.698827	0.016309	0.077111
2	0.643348	0.581647	0.038628	0.072576
2	0.745923	0.800886	0.061803	0.185975

Rys. 2.7. Wygenerowane etykiety w formacie YOLO.

Kluczowym elementem niezbędnym do wytrenowania modelu na swoich danych jest utworzenie pliku o rozszerzeniu *.yaml*, który następnie jest przekazywany jako parametr do skryptu *train.py*. Plik powinien zawierać ścieżki do zestawu treningowego i testowego zabranych danych oraz listę klas obiektów, które chcemy wykrywać. Chcąc polegać na modelu udostępnionym w projekcie *YOLOv5* należy wykorzystać lub rozszerzyć istniejącą już listę klas znajdującą się w pliku *coco.yaml* (Rys. 2.8).

```
train: /content/gdrive/MyDrive/Praca magisterska/collab/train_final
val: /content/gdrive/MyDrive/Praca magisterska/collab/test_final

nc: 81 # number of classes
names: [ 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat', 'traffic light',
    'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',
    'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee',
    'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard',
    'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple',
    'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch',
    'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone',
    'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear',
    'hair drier', 'toothbrush', 'cargovelo' ]
```

Rys. 2.8. Plik *dataset.yaml* wykorzystany w projekcie.

2.2.3. Metryki modelu

YOLOv5 pozwala na równoczesne przeprowadzanie treningu modelu oraz zapisywanie wyników uczenia w narzędziu chmurowym *wandb.ai - Weights & Biases (W&B)*. Jest to oprogramowanie służące rejestraniu i wizualizacji przebiegu uczenia, zintegrowane z *YOLOv5*. Trafiają tam metryki zebrane podczas uczenia modelu.

Liczba przetrzymywanych wyników dla jednego użytkownika jest ograniczona - najstarsze wyniki są usuwane rotacyjnie. Po pomyślnie przeprowadzonym treningu modelu, na zarejestrowanym przez użytkownika koncie w narzędziu gromadzone są wykresy metryk, takie jak:

- *precision* - określa „wrażliwość” modelu, jej wartość mówi o tym ile obiektów wśród wszystkich rozpoznanych na obrazie zostało zaklasyfikowane do właściwych im kategorii, przyjmuje wartości od 0 do 1,

- *recall* - wartość metryki mówi o tym ile obiektów zostało poprawnie rozpoznanych względem tego ile obiektów powinno rzeczywiście zostać rozpoznane, przyjmuje wartości od 0 do 1,
- *mAP* (ang. *mean Average Precision*) - wartość wyznaczana graficznie jako pole pod wykresem *precision - recall*, przyjmuje wartości od 0 do 1,
- *box loss* - wartość wyrażająca jak dokładnie wyznaczona jest obwiednia wokół wykrytego obiektu,
- *cls loss* - wyraża poprawność klasyfikacji danego *bboxa*,
- *object loss* - metryka, na którą składają się błąd klasyfikacji, błąd lokalizacji (czyli jak wyznaczona została obwiednia względem rzeczywistego położenia obiektu) oraz błąd pewności detekcji,
- oraz przykładowe zdjęcia, na których dokonano predykcji podczas testów.

Wykorzystanie *wandb.ai* w projekcie rozszerzającym *YOLOv5* daje możliwość porównywania przebiegów poszczególnych metryk w procesie trenowania modelu. Aby skorzystać z tego narzędzia należy zainstalować pakiet *wandb*, a podczas pierwszego uruchomienia skryptu *train.py* skonfigurować konto użytkownika, w którym gromadzone będą dane sporządzanego modelu.

3. Analiza danych wejściowych

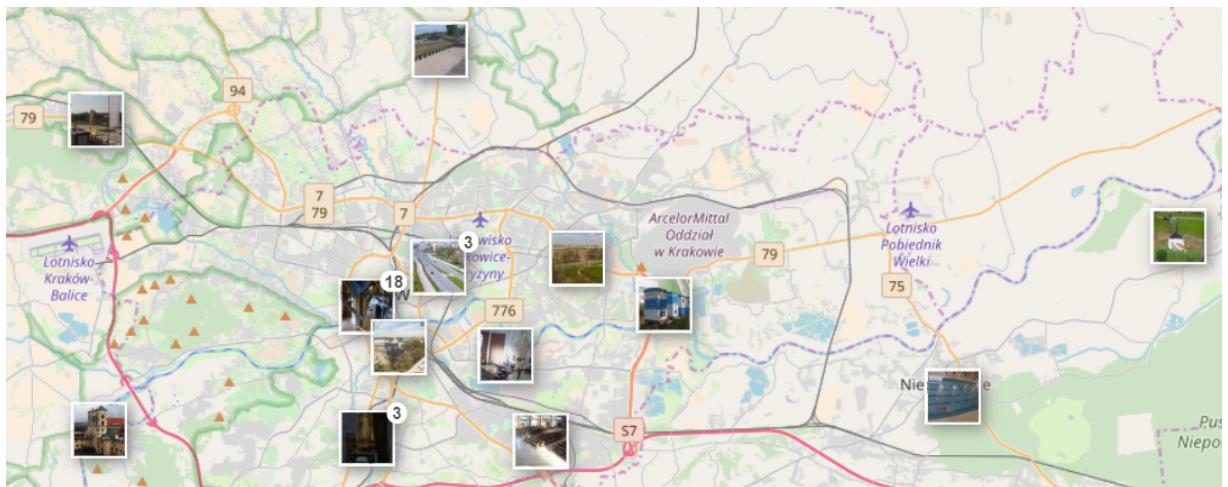
Rozdział poświęcono procesowi kompletowania danych (zdjęć oraz odpowiadającym im plików z etykietami w formacie *YOLO*), które posłużyły do późniejszej nauki i testów modelu. Opisano w nim sposób przygotowania danych oraz wykorzystane do tego narzędzia, a także sam proces trenowania oraz testowania powstałego modelu.

3.1. Wybór lokalizacji poddanych badaniom

Od 2014 roku, kiedy Rada Miasta Krakowa podjęła temat stworzenia monitoringu wizyjnego w referendum zorganizowanym celem poprawy bezpieczeństwa obywateli, na terenie Krakowa obserwujemy zwiększenie się liczby zainstalowanych kamer w przestrzeni publicznej. Łącznie, biorąc pod uwagę różne podmioty sfery komunalnej, takie jak Policja i Straż Miejska, czy placówki oświatowe, jest ich 12698 (stan na rok 2018)[12].

Kamery monitoringu nie są ogólnodostępne, a wykorzystanie ich do pracy magisterskiej wymagałoby uzyskania pozwolenia Straży Miejskiej/ZDMK. Jednakże, niniejszy projekt ma na celu zaprezentowanie działania modelu i algorytmu, które w zastosowaniu będą uniwersalne - umożliwi wykorzystanie ich na obrazach pochodzących z różnych źródeł. W związku z tym do realizacji tego projektu posłużono się obrazami publicznie dostępnych kamer internetowych, z których obraz jest transmitowany w czasie rzeczywistym.

W Krakowie zainstalowane są 34 publicznie dostępne kamery internetowe[16].



Rys. 3.1. Rozmieszczenie kamer internetowych w Krakowie[16].

Przedmiotem badań tej pracy jest analiza ruchu pieszych i dostaw towarów w strefach ograniczonego ruchu. Wybrane kamery internetowe znajdują się w strefach przedstawionych w rozdziale 1. Lokalizacje poddane analizie to:

- Rynek Główny - widok z okna (kamera nieruchoma),
- Ulica Grodzka (z Hotelu Senackiego),
- Ulica Grodzka - Rynek Główny.

W celu zgromadzenia zróżnicowanych zdjęć przedstawiających obiekty z różnych perspektyw, wykorzystano również obraz z kamery ruchomej, zainstalowanej w Rynku Głównym, przedstawiającej widok od strony ul. Brackiej.

3.2. Obróbka danych

Elementem niezbędnym do rozpoczęcia pracy nad modelem było zgromadzenie odpowiedniej bazy filmów, w której istotne było przechwytcenie kontrastujących warunków pogodowych, zacienienia danego miejsca, a także pór dnia. Dzięki temu obiekty wybierane z poszczególnych klatek i przekazywane dalej do treningu modelu, były zróżnicowane. Zbieranie danych przeprowadzono na przełomie marca i kwietnia 2021 roku. Skompletowano godzinne filmy przechwytywane bez przerwy ze wszystkich czterech kamer z kilkunastu dni. Wybór takiego przedziału czasu umożliwił zebranie danych z tygodnia zwykłego oraz wielkanocnego.

Dodatkową wartością płynącą z eksperymentu bazującego na danych z tamtego okresu jest trwająca pandemia i osiągane wtedy rekordy w liczbie zachorowań na *covid-19*, a także obojętne w Polsce obostrzenia.

3.2.1. Zbieranie danych

W pierwszej fazie kompletowania danych zrealizowano przechwytywanie transmisji z wybranych kamer oraz zapisywanie ich lokalnie na dysku w formacie *.mp4*. Narzędzie, które zostało do tego wykorzystane to *youtube-dl - command-line*'owy program, który pozwala na pobieranie filmów z m.in. z *YouTube*, ale także z innych źródeł.

```
#!/bin/bash
timeout 3599 /usr/local/bin/youtube-dl -o "/media/klaudia/8fcc6fb9-4004-493e-b4b6-b7efdc1df57b/
klaudiamgr/cameras/grodzka_rynek/%(title)s_%(date '+%Y%m%dT%H%M%S%z').%(ext)s" http://
185.70.181.30:8080/hls/live.stream.m3u8
```

Rys. 3.2. Przykład wykorzystania narzędzia *youtube-dl*.

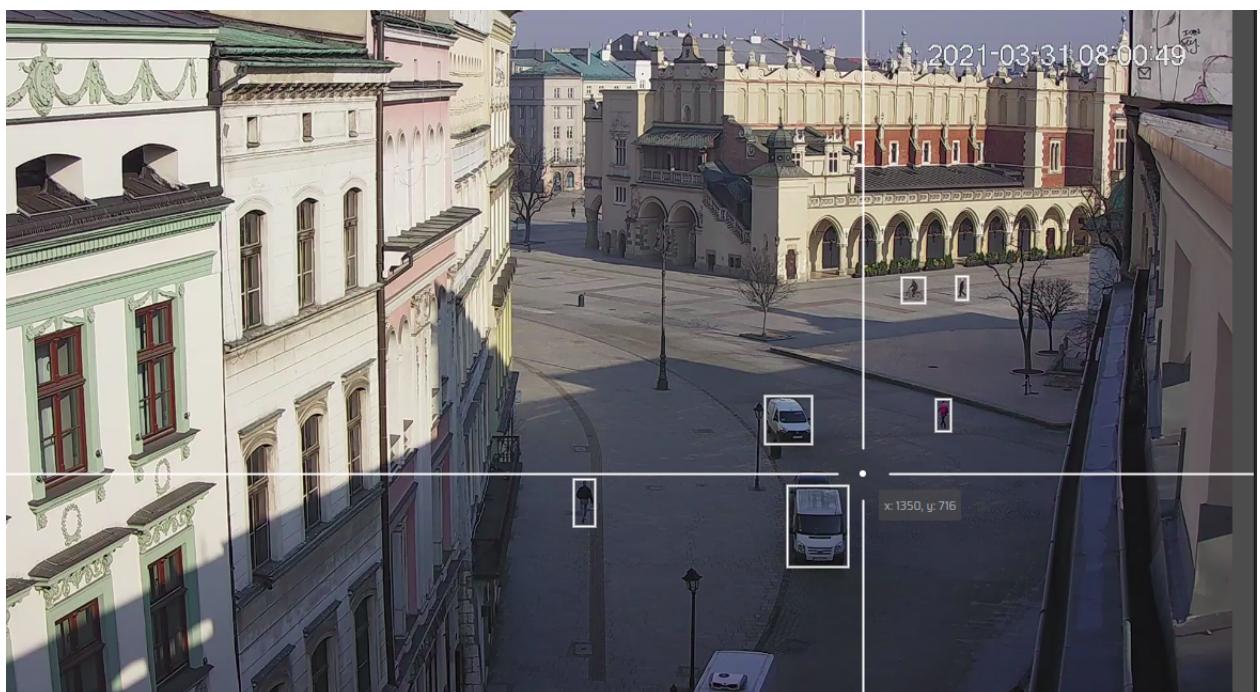
Aby uniknąć manualnego uruchamiania skryptów realizujących pobieranie zapisów kamer, wprowadzono automatyzację poprzez wykorzystanie *crontaba*. Opracowano także skrypt, umożliwiający zapisywanie filmów ze wszystkich kamer równolegle. W początkowej fazie kompletowania danych podjęto próbę zapisu filmów 24-godzinnych, jednak takie podejście wiązało się z wysokim ryzykiem utraty danych. Przerwanie połączenia internetowego skutkowało natychmiastowym zakończeniem zapisu. W związku z tym, w ostatecznym rozwiążaniu

zadanie uruchamiane było co godzinę, a utworzona w ten sposób baza filmów składa się z kilkuset plików .mp4.

Kolejnym etapem pracy nad zbiorem danych przekazanych później do modelu było wyodrębnienie i zapis klatek z poszczególnych filmów przy pomocy narzędzia *VLC*. Aby uzyskane w ten sposób zdjęcia nie były zbyt podobne do siebie, wybrano do tego filmy pochodzące z różnych dni i godzin, przedstawiające odmienne warunki pogodowe.

Po zgromadzeniu odpowiedniej ilości zdjęć, przystąpiono do procesu etykietowania widocznych na nich obiektów z wykorzystaniem narzędzia *makesense.ai*. Przed rozpoczęciem pracy z narzędziem, do nowo utworzonego projektu typu „Object detection” wprowadzono klasy „person”, „bicycle”, „car” oraz „cargovelo”, wykorzystane później do trenowania modelu.

Późniejsze postępowanie w programie *makesense.ai* wiązało się ze żmudnym procesem otaczania obiektów poszczególnych klas *bounding-box*-ami oraz oznaczania ich odpowiadającymi etykietami (pomimo, że narzędzie posiada mechanizm auto-detekcji i etykietowania obrazów). Po zakończeniu każdego etapu prac należało wyeksportować przetworzone dane - wykorzystane narzędzie umożliwia eksport w formacie *YOLO*.



Rys. 3.3. Proces etykietowania obiektów.

3.2.2. Przetwarzanie danych

Pierwszym i najważniejszym krokiem w przygotowaniu danych dla problemu detekcji jest podział zgromadzonych danych na zbiory: treningowy i testowy. Aby opracowywany model poprawnie rozpoznawał i klasyfikował obiekty na obrazie, należy dostarczyć mu opisanych wzorców do nauki oraz walidacji. Zwykle proporcja, którą wybieramy to 80% danych treningowych do 20% testowych - taką też obrano w realizacji niniejszego projektu.

Po przeprowadzeniu losowego podziału zdjęć na grupę treningową i testową, zmodyfikowano wymiary przetwarzanych obrazów tak, aby pasowały do algorytmu *YOLOv5*. W tym wypadku każde ze zdjęć otrzymało rozmiar 640x480.

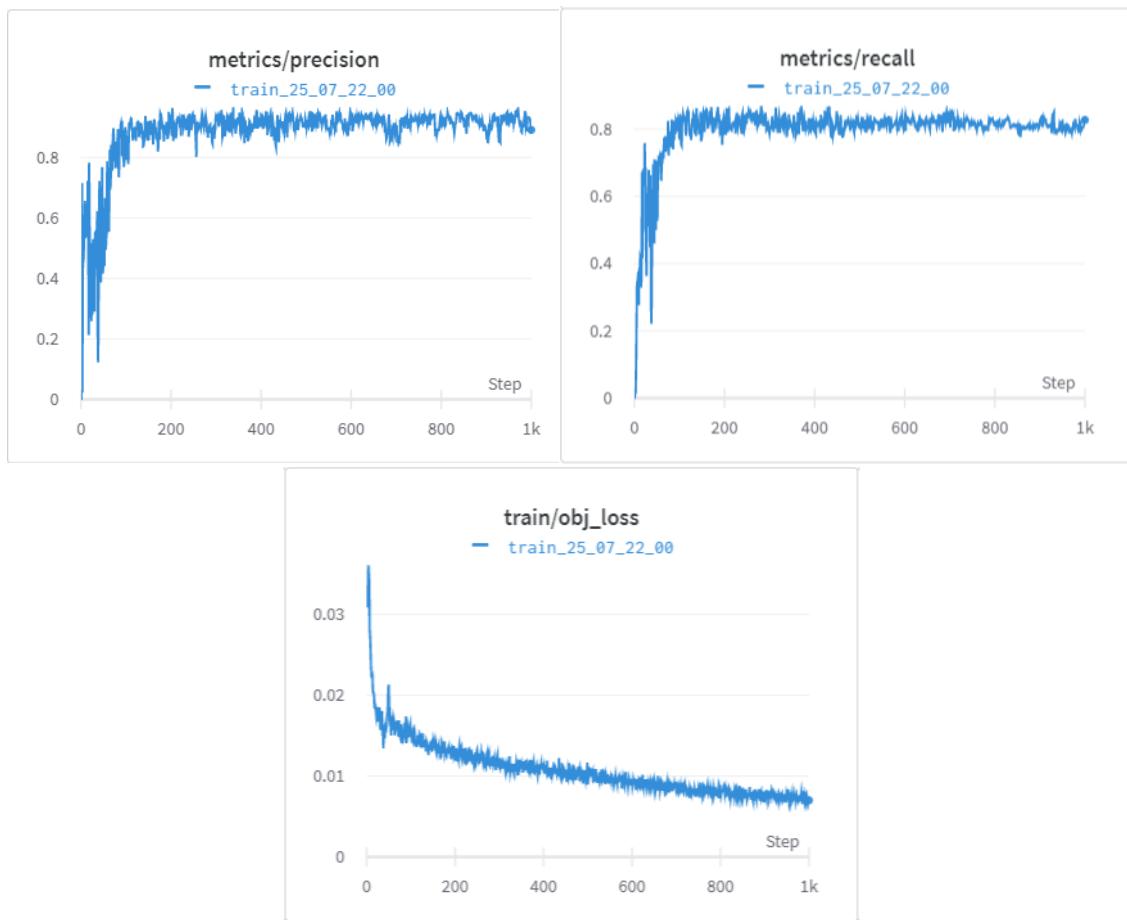
Przetworzone w ten sposób zdjęcia wraz z odpowiadającymi im plikami formatu *YOLO*, zawierającymi informacje o widocznych na nich obiektach, zgromadzono w folderach *train* i *test* oraz przeniesiono w docelowe miejsce składowania danych. W części praktycznej tego projektu wykorzystano narzędzie *Google Colaboratory*, dlatego oczywisty wybór padł na *Google Drive*.

3.3. Trenowanie modelu

Faza trenowania modelu została poprzedzona przygotowaniem elementów niezbędnych do uruchomienia *YOLOv5 API* oraz konfiguracją środowiska. Aby jak najbardziej efektywnie wykorzystać dostępne zasoby sprzętowe oraz czasowe, w utworzonym notatniku *Google Colaboratory* obrano akcelerator sprzętowy typu *GPU*. Notatnik powiązano również z zasobami zgromadzonymi na dysku *Google* oraz pobrano projekt *YOLOv5*. Przed rozpoczęciem trenowania skonstruowano plik z konfiguracją w formacie *.yaml*, przygotowany specjalnie pod wykonanie niniejszego projektu (Rys. 2.8). Celem przeprowadzenia efektywnego treningu obrano szereg argumentów przekazanych do skryptu *train.py*:

- *img* - większy z wymiarów zdjęć wykorzystanych w procesie uczenia,
- *batch* - liczba próbek szkoleniowych wykorzystywanych w jednej iteracji,
- *epochs* - liczba cykli treningowych,
- *data* - ścieżka do pliku *.yaml*,
- *save-period* - długość okresu w cyklach (*epochs*), po jakim zapisywany jest stan modelu,
- *weights* - wykorzystane wagi, (do treningu modelu przygotowanego w ramach niniejszej pracy magisterskiej wykorzystany został model *YOLOv5l6*. Wybór podyktowany został dobrymi metrykami modelu oraz kwestiami wydajnościowymi - trenowanie z zastosowaniem *YOLOv5x6* trwało zbyt długo, nawet przy wykorzystaniu układu *GPU* w *Google Colab* (Rys. 2.6)),
- *name* - nazwa modelu.

Opracowanie modelu, który możliwie najlepiej spełniał założenia projektu wymagało wielokrotnego rozszerzania kompletowanego zestawu danych, a co za tym idzie wielokrotnego uruchamiania skryptu trenującego. Pojedyncze uruchomienie w notatniku *Google Colaboratory*, zwłaszcza dla większej liczby kroków, trwało nawet do kilkunastu godzin, co zaburzało ciągłość uczenia. *Google Colaboratory* posiada ograniczenia czasu wykorzystania maszyny wirtualnej przypisanej do sesji użytkownika, a także czasu bezczynności, które wahają się w zależności od priorytetu nadanego aktywności użytkownika (zwykle użytkownicy, którzy nie wykonują długotrwałych sesji obliczeniowych są traktowani z wyższym priorytetem).



Rys. 3.4. Wizualizacja wyników uczenia.

3.4. Testy modelu

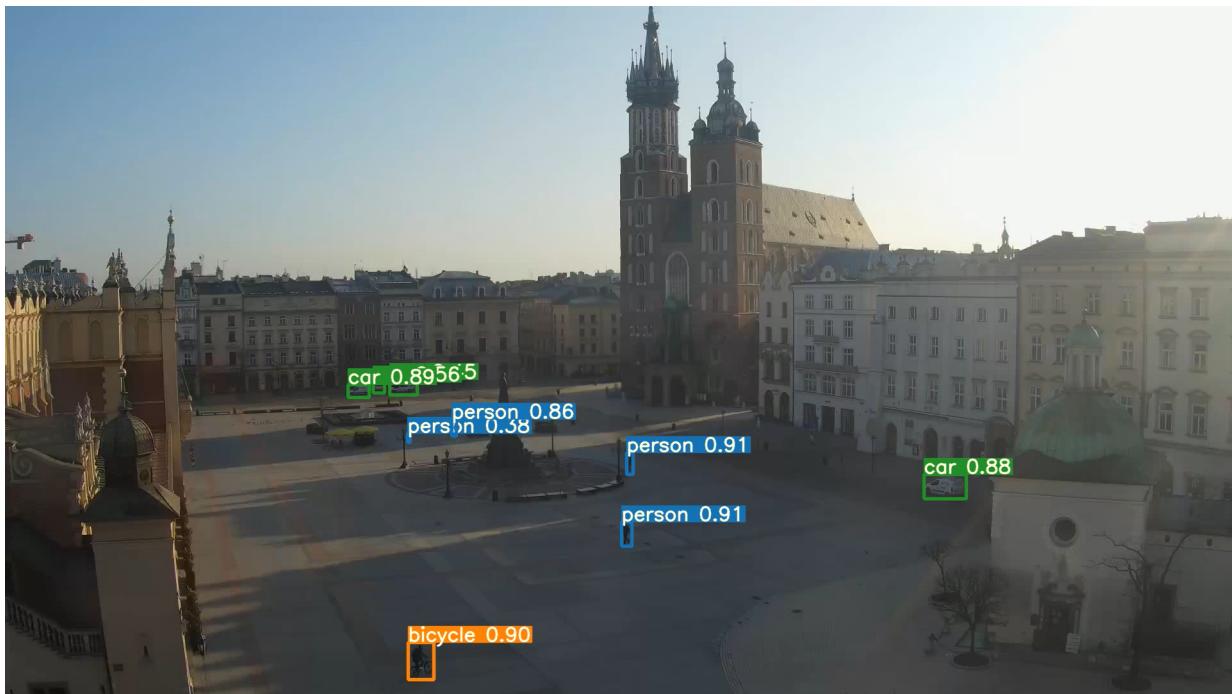
W procesie nauki modelu (po każdej iteracji uczenia) - poza przeglądem metryk gromadzonych w *wandb.ai* weryfikowano jego działanie poprzez testy przeprowadzane na najkrótszych z zapisanych filmów z wybranych kamer. Wykorzystano do tego skrypt *detect.py* uruchamiany z następującymi parametrami:

- *img-size* - większy z wymiarów klatek przetwarzanego filmu,
- *source* - ścieżka do pliku .mp4,
- *weights* - ścieżka do wytrenowanego modelu,
- *classes* - numery wykrywanych klas (wybrane zgodnie z utworzonym plikiem .yaml),
- *conf-thres* - próg „pewności” wystąpienia obiektu danej kategorii (np. pokaż tylko obiekty, dla których daną kategorię określono z prawdopodobieństwem większym niż 30%).

Wynikiem działania skryptu jest wygenerowany plik .mp4 bazujący na pliku wyjściowym, ale z rozpoznanymi obiekttami wybranych kategorii. Obiekt danej klasy otoczony jest obwiednią

określonego koloru, towarzyszy mu etykieta oraz wartość prawdopodobieństwa, z jakim został zaklasyfikowany do swojej kategorii.

Po zebraniu odpowiedniej ilości metryk oraz przeprowadzeniu empirycznych badań na różnych modelach, wybrany został ten, dla którego skuteczność detekcji na różnych nagraniach była najlepsza.



Rys. 3.5. Wizualna weryfikacja działania opracowanego modelu.

4. Implementacja licznika

Kluczowym elementem praktycznej części projektu, wykonanej na potrzeby przeprowadzenia analizy ruchu pieszych i dostaw towarów w strefach ograniczonego ruchu, było opracowanie licznika obiektów. Licznik przygotowano wykorzystując implementację *open-source*-owej biblioteki do trackowania obiektów 2D, dostępnej w serwisie *GitHub - Norfair*[5].

4.1. Norfair tracker

Norfair jest narzędziem rozwijanym i utrzymywanym przez firmę *Tryoabs*, która zajmuje się doradztwem projektowym w optymalizacji procesów z wykorzystaniem uczenia maszynowego. *Norfair* umożliwia pracę z różnymi metodami detekcji, w tym również *YOLOv5*. Tracker jest w pełni konfigurowalny i prosty w dostosowaniu do potrzeb użytkownika.

Działanie trackera obiektów *Norfair* polega na wyznaczaniu estymowanej przyszłej pozycji obiektu na podstawie wiedzy o jego wcześniejszym ruchu. Następnie szacowane położenie jest dopasowywane za pomocą funkcji odległości do rzeczywistego, które jest zwracane przez wykorzystywany detektor[5]. Funkcja odległości zastosowana w niniejszym projekcie generuje oczekiwany dystans obliczając wartość metryki euklidesowej, a więc długości odcinka łączącego te punkty.

$$d(A,B) = \sqrt{(x_{1A} - x_{1B})^2 + (x_{2A} - x_{2B})^2 + \dots + (x_{nA} - x_{nB})^2} = \sqrt{\sum_{i=1}^n ((x_{iA} - x_{iB})^2)}$$

Rys. 4.1. Wzór na odległość pomiędzy dwoma punktami w przestrzeni n - wymiarowej.

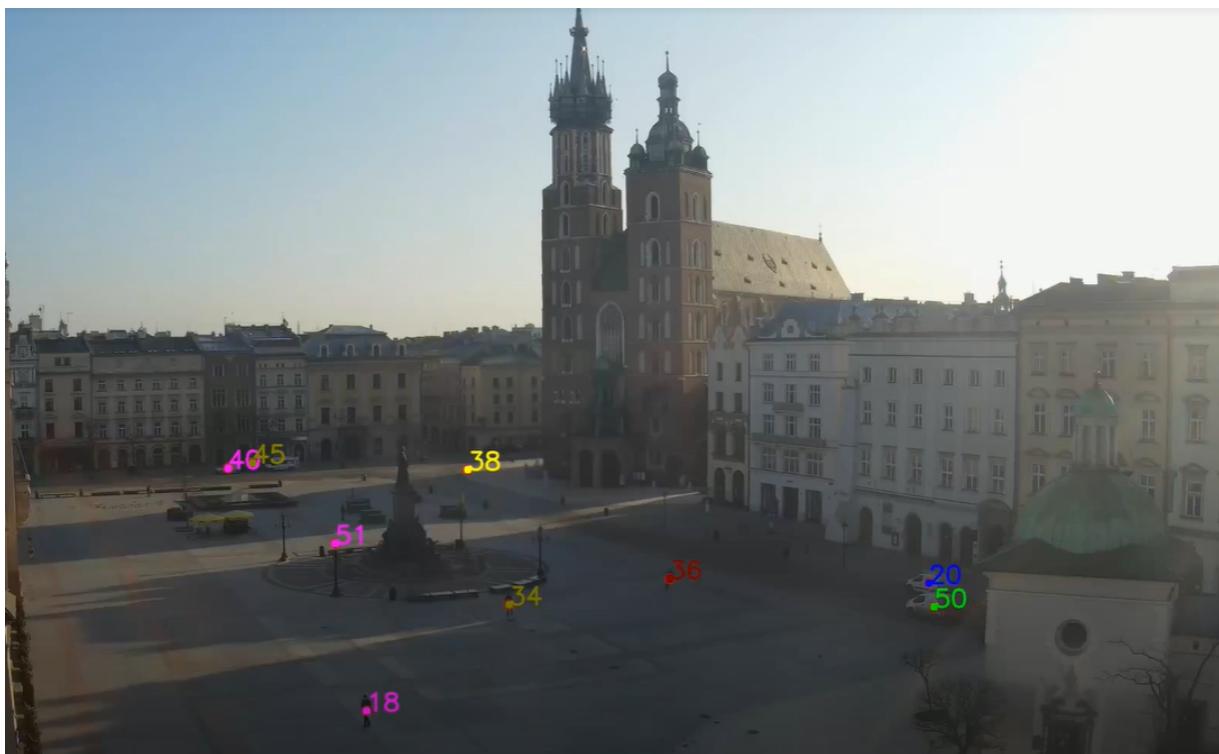
Tracker daje możliwość wizualizacji procesu „śledzenia” obiektów na przetwarzanym video na dwa sposoby:

- *centroid* - trackowany obiekt jest oznaczany w każdej klatce za pomocą punktu umieszczonego w centrum *bounding-box*-a wyznaczonego przez detektor,
- *bbox* - trackowany obiekt jest otaczany *bounding box*-em.

Norfair przypisuje namierzonym obiektem unikalne identyfikatory, dzięki czemu zapewniona jest ciągłość „śledzenia”. Właściwości konkretnego obiektu są zapisywane do momentu aż zniknie on z analizowanego obszaru. Cechy obiektu aktualizowane w każdej iteracji to:

- numer ostatniej klatki, w której został namierzony,

- czas istnienia obiektu w danym obszarze,
- zajmowana pozycja.

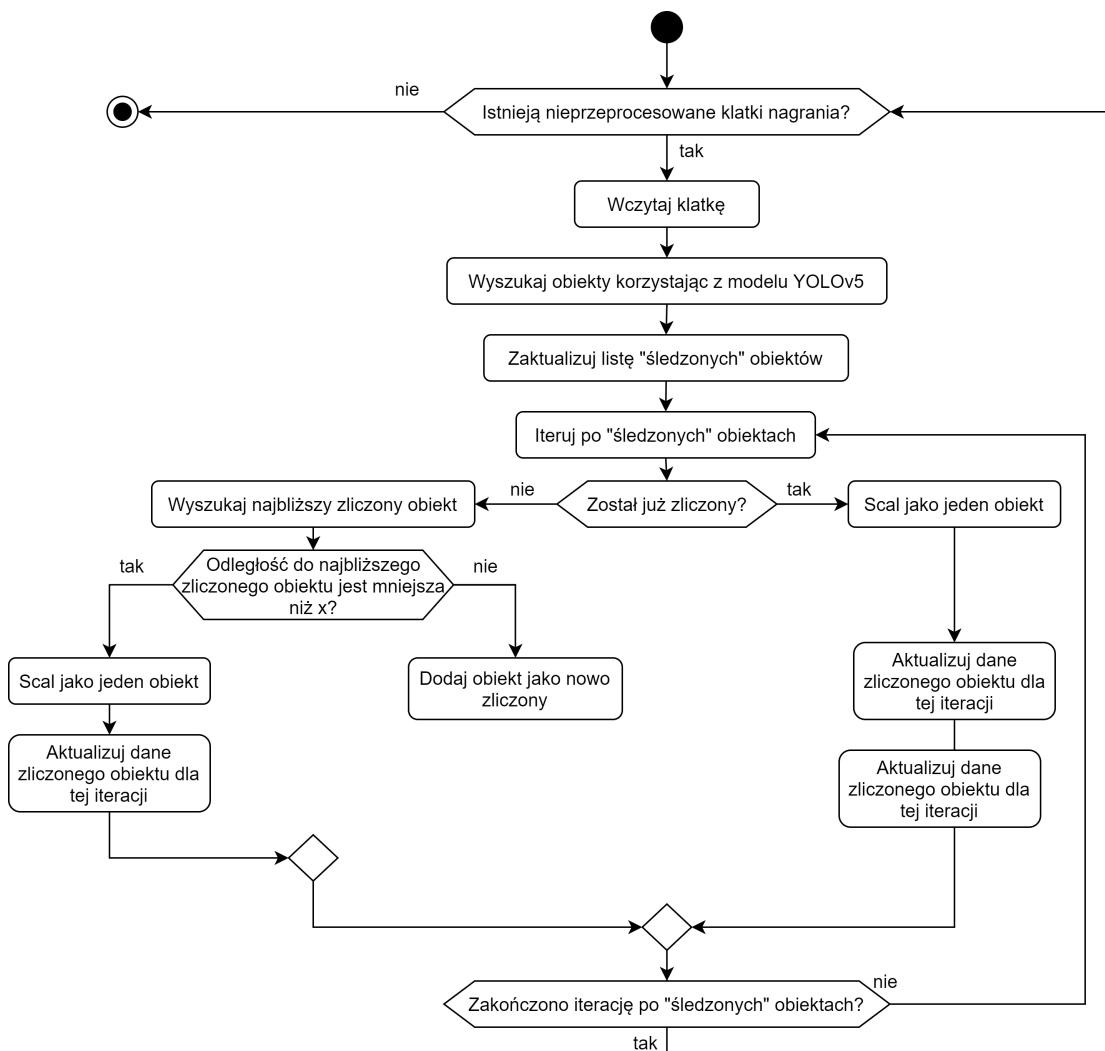


Rys. 4.2. Klatka wideo przetworzonego za pomocą trackera *Norfair*.

4.2. Działanie licznika

Implementacja trackera została rozszerzona o następujące elementy:

- Obsłużenie „nieciągłości” ruchu wykrywanego obiektu - w przypadku, kiedy obraz kamery jest niewyraźny, zacieniony czy rozmazany zdarza się, że tracker nie wychwytuje każdej klatki ruchu danego obiektu. W takim przypadku przy zastosowaniu wyjściowej implementacji trackera zaburzona była ciągłość detekcji i obiekt był wykrywany ponownie z nowym identyfikatorem - jako zupełnie inny obiekt. Licznik opracowany na potrzeby projektu pozwala na zaburzenia ciągłości, których poziom nie przekracza określonego w konfiguracji progu, np. 20 pikseli. Wówczas, kiedy inny obiekt o tej samej klasie zostaje wykryty w promieniu nie większym niż 20 pikseli od miejsca „zgubienia” wcześniej wykrywanego obiektu, to algorytm uznaje ten nowy obiekt za równoznaczny temu „zgubieniem”.



Rys. 4.3. Graf opisujący działanie algorytmu licznika.

- Pomiar czasu postoju pojazdu - jeżeli dla obiektu kategorii „car” różnica odległości pomiędzy środkami *bbox*-ów, które go otaczają, w kolejnych klatkach jest mniejsza niż umowna wartość ustalona przez użytkownika, uznajemy, że obiekt się nie porusza i naliczony zostaje czas nieaktywności.
- Eksport zebranych informacji - w wyniku uruchomienia licznika na konkretnym pliku video generowane są dla niego 2 pliki (jeden w formacie *.txt*, drugi *.csv*), zawierające podsumowanie detekcji. Pierwszy z pików gromadzi ogólne informacje o tym ile obiektów danej kategorii wystąpiło na analizowanym video. Drugi zawiera bardziej szczegółowe dane. W kolejnych kolumnach zapisywane są:
 - unikalny identyfikator obiektu,
 - identyfikator klasy, do której należy,
 - numery pierwszej i ostatniej klatki istnienia obiektu w zasięgu kamery,
 - czas postoju w przypadku, jeśli analizowanym obiektem jest pojazd (czas mierzony w klatkach).

Object ID	Class	Object First Frame	Object Last Frame	Object Max Idle
2	0	8	313	0
3	0	8	387	0
19	1	280	371	0
12	2	8	414	237
23	0	69	416	0
18	0	390	1445	0
36	0	744	1175	0

Tabela 4.1. Fragment wygenerowanego pliku .csv z podsumowaniem detekcji.

5. Uzyskane wyniki

Rozdział poświęcony przeglądowi wyników otrzymanych dzięki uruchomieniu opracowanego licznika na zgromadzonych filmach. Opisano w nim analizowane scenariusze badawcze oraz umieszczono interpretację uzyskanych rezultatów.

5.1. Środowisko obliczeniowe

Ze względu na ograniczone zasoby sprzętowe udostępniane przez *Google Colaboratory*, analizę zebranych filmów z wykorzystaniem licznika przeprowadzono na *Prometheus*-ie - superkomputerze zainstalowanym w Akademickim Centrum Komputerowym Cyfronet AGH. Klaster obliczeniowy *Prometheus* składa się z 1728 klastrów HP Apollo 8000, które korzystają z procesorów Xeon E52680v3. Każdy z nich ma 24 rdzenie fizyczne, które składają się na 55,728 rdzeni obliczeniowych. Do zarządzania zadaniami *Prometheus* wykorzystuje system kolejkowy *SLURM*, który udostępnia użytkownikom zasoby obliczeniowe na zadany okres czasu, pozwala na równoległe uruchamianie zadań oraz umożliwia monitorowanie pracy wykonywanej na poszczególnych węzłach.

5.2. Opis analizowanych scenariuszy

Sposób działania opracowanego licznika narzucał pewne ograniczenie dla lokalizacji poddanych badaniu. Do analizy liczby pieszych i dostaw towarów wykorzystano tylko filmy z trzech nieruchomych kamer. Przesuwające się „oko” kamery zamontowanej w Rynku Głównym uniemożliwiało zebranie wiarygodnych statystyk. Wraz z ruchem kamery nieruchome pojazdy znikłyby z jej zasięgu, a więc nieprawdziwy był wyznaczony czas ich postoju. Po powrocie kamery do pozycji wyjściowej takie obiekty zostałyby zliczone ponownie.

Po wstępny przeglądzie uzyskanych wyników stwierdzono, że w badanych lokalizacjach nie wykryto żadnych obiektów klasy *cargovelo* - być może tego typu pojazdy nie poruszają się po terenie ścisłego centrum Krakowa. W wizualizacji wyników, zamieszczonej w dalszej części rozdziału nie uwzględniono zatem obiektów tej kategorii.

Do przeprowadzenia analizy ruchu w strefach ograniczonego ruchu obrano następujące scenariusze badawcze:

1. Wyznaczenie liczby pieszych, rowerzystów oraz pojazdów, wykrytych w ciągu zwykłego dnia roboczego - statystyki zbierane dla wszystkich trzech kamer, wyznaczone dla każdej godziny.

2. Wyznaczenie maksymalnej liczby pieszych, rowerzystów oraz pojazdów, znajdujących się równocześnie w obszarze kamery w interwale 15 min - statystyki zbierane dla wszystkich trzech kamer, wyznaczone dla każdej godziny w przedziale czasu od 6:00 do 12:00.
3. Analiza czasów postoju pojazdów w okresie dostaw, tj. w przedziale czasu od 7:00 do 11:00 w interwale co godzinę - średnia liczba dostaw przechwyconych w każdej kamerze, średni czas postoju (przybliżony średni czas dostawy).
4. Porównanie statystyk zebranych z trzech kamer w ciągu okresu zwykłego oraz świątecznego - porównanie zwykłej soboty, niedzieli i poniedziałku z weekendem Wielkanocnym (Wielka Sobota, Wielkanoc i Poniedziałek Wielkanocny) pod kątem liczby wykrytych pieszych, rowerzystów i pojazdów.

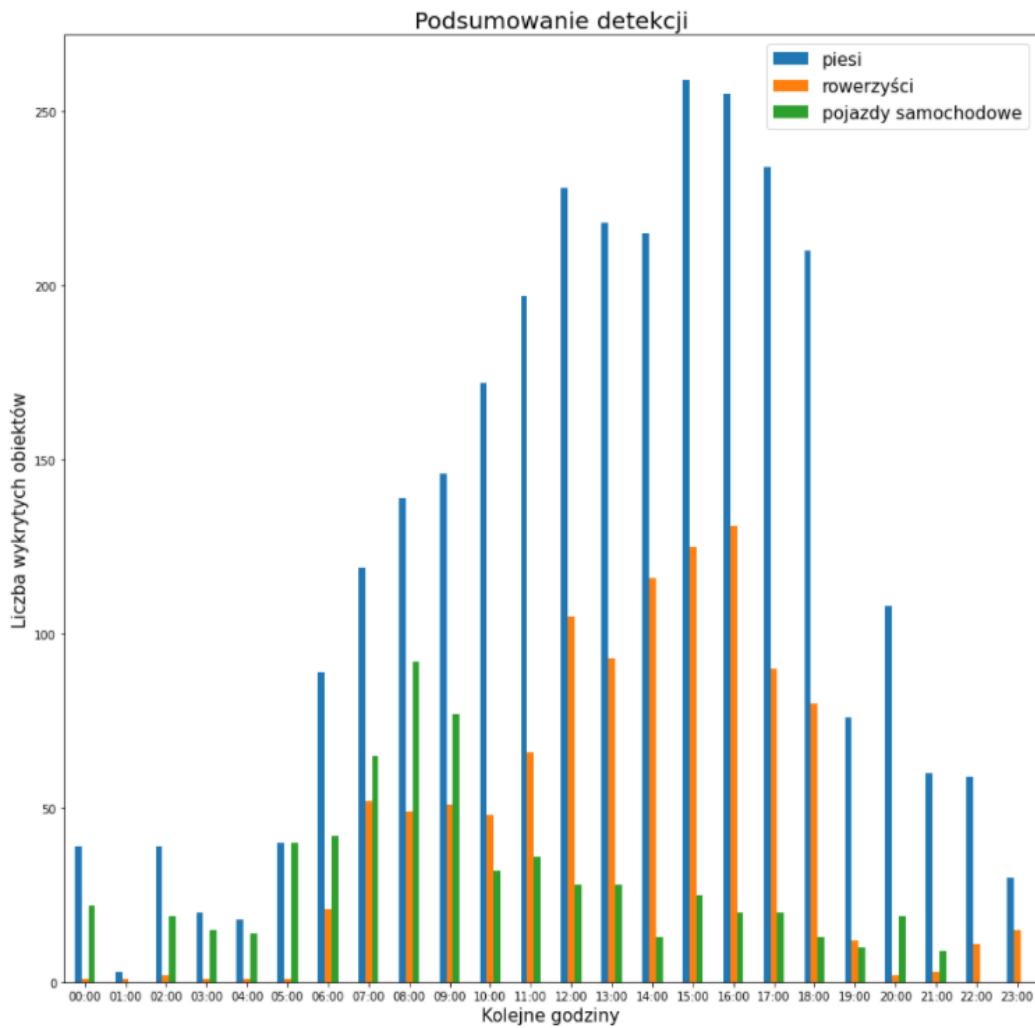
5.2.1. Podsumowanie detekcji wykonanej w zwykły dzień

W opisywanym scenariuszu badano liczbę pieszych, rowerzystów oraz pojazdów wykrytych przez zaimplementowany licznik. Analizie poddano filmy zebrane w poniedziałek 29 marca 2021r. W dalszej części podrozdziału przedstawiono wyniki detekcji otrzymane dla każdej z analizowanych lokalizacji.

5.2.1.1. Rynek Główny

Na otrzymanym wykresie można zaobserwować następujące zależności:

- największe zagęszczenie pieszych w badanym obszarze wystąpiło w godzinach między 15:00 a 18:00,
- najwięcej pojazdów pojawiło się w badanym obszarze w godzinach przypadających na czas dostaw, tj. od 7:00 do 10:00,
- najwięcej rowerzystów zostało wykrytych w godzinach 15:00-17:00, a więc w czasie, kiedy najwięcej osób wraca z pracy lub decyduje się na aktywny relaks na świeżym powietrzu po pracy.



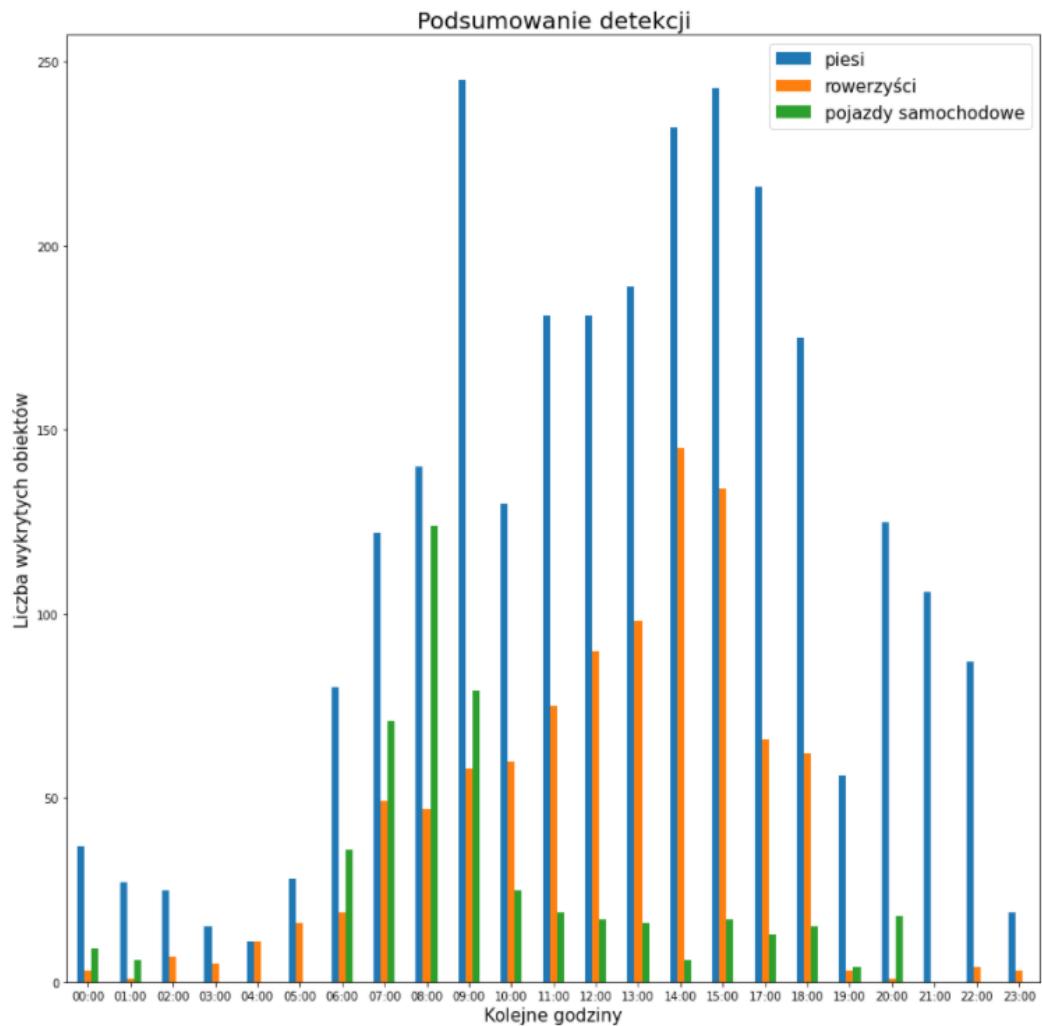
Rys. 5.1. Podsumowanie detekcji Rynek Główny, 29.03.2021r.

Zestawienie największych wartości liczby uczestników ruchu otrzymanych w badanym obszarze wraz z odpowiadającymi im godzinami:

	Wartość	Godzina
Maksymalna liczba wykrytych pieszych	259	15:00
Maksymalna liczba wykrytych rowerzystów	131	16:00
Maksymalna liczba wykrytych pojazdów	92	8:00

5.2.1.2. Ul. Grodzka - widok na Rynek Główny

Kamera z widokiem na Rynek Główny zainstalowana na ulicy Grodzkiej obejmuje swoim zasięgiem część placu przed Kościółem Mariackim oraz Sukiennice.



Rys. 5.2. Podsumowanie detekcji ul. Grodzka (rynek) 29.03.2021r.

Zaobserwowane prawidłowości wynikające z wykresu zamieszczonego powyżej (Rys. 5.2):

- najwięcej pieszych zostało wykrytych o godzinie 9:00, kiedy otwierają się Sukiennice oraz o 15:00 - po zakończeniu pierwszej zmiany pracujących tam sprzedawców,
- najwięcej pojazdów wykryto o godzinie 8:00, a więc znowu w czasie przypadającym na dostawy towarów oraz dojazd do pracy,
- najwięcej rowerzystów znalazło się w badanym obszarze w godzinach 14:00 - 15:00 - czas przypadający na powroty do pracy.

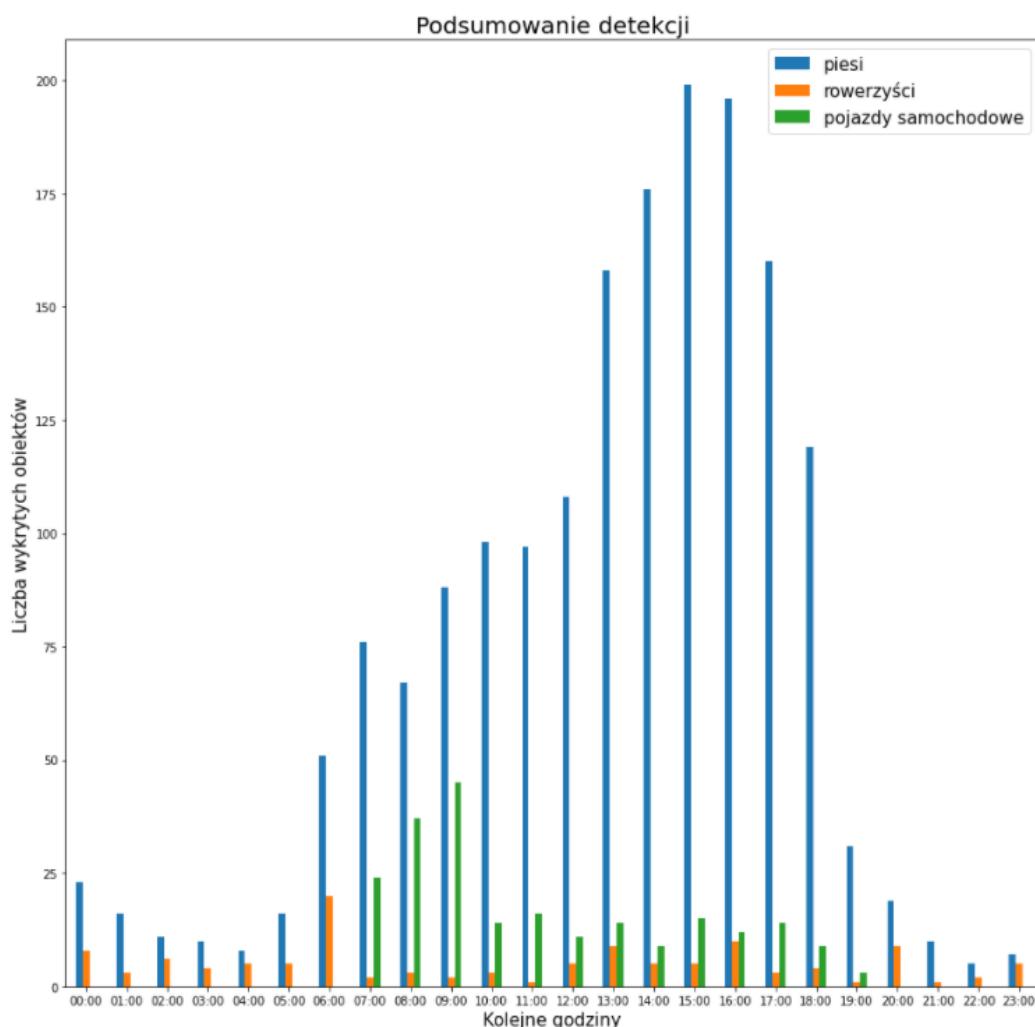
Tabela przedstawiająca maksymalne wyniki detekcji oraz czasy ich wystąpienia dla każdej z kategorii:

	Wartość	Godzina
Maksymalna liczba wykrytych pieszych	245	9:00
Maksymalna liczba wykrytych rowerzystów	145	14:00
Maksymalna liczba wykrytych pojazdów	124	8:00

5.2.1.3. Ul. Grodzka - widok z Hotelu Senackiego

Trzecia z lokalizacji poddanych analizie to widok z Hotelu Senacki na ulicę Grodzką. Zasięg kamery w porównaniu z poprzednimi jest niewielki - obejmuje dziedziniec Kościoła św. Apostołów Piotra i Pawła oraz mały fragment ulicy.

Jak widać na wykresie zamieszczonym poniżej (Rys. 5.3), otrzymane wyniki różnią się od zebranych dla pozostałych lokalizacji pod kątem liczby wykrytych pojazdów oraz rowerzystów. Może to być spowodowane znacznie mniejszym zasięgiem kamery oraz jakością zebranych filmów. Obraz z tej kamery często jest prześwietlony, szczególnie w godzinach porannych, w słoneczne dni.



Rys. 5.3. Podsumowanie detekcji ul. Grodzka (widok z Hotelu Senackiego)
29.03.2021r.

Zależności zaobserwowane na wykresie:

- najwięcej pieszych pojawiło się w analizowanym obszarze w godzinach od 14:00 do 18:00, czyli czas na powroty z pracy do domu oraz rekreacyjne wieczorne spacery w kierunku Wisły,

- w badanym obszarze przez cały dzień pojawiło się stosunkowo mało rowerzystów, a ich największą liczbę wykryto o godzinie 6:00 oraz 20:00 - jest to czas często przeznaczany na treningi
- najwięcej pojazdów wykryto w tym obszarze w godzinach od 7:00 do 10:00 - okres przypadający na poranne dostawy towarów.

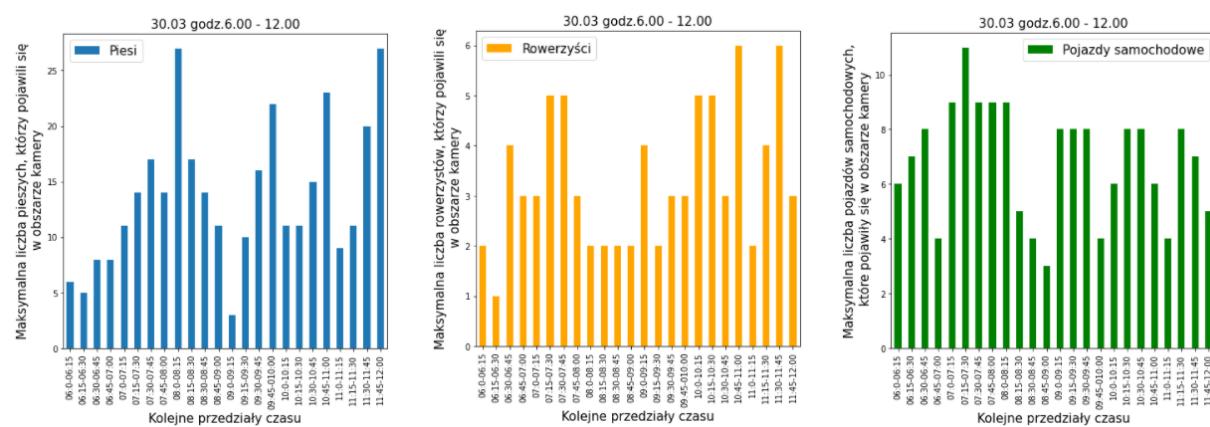
Poniżej zamieszczono zestawienie maksymalnych wartości wyznaczonych przez licznik obiektów dla każdej z kategorii:

	Wartość	Godzina
Maksymalna liczba wykrytych pieszych	199	15:00
Maksymalna liczba wykrytych rowerzystów	23	6:00
Maksymalna liczba wykrytych pojazdów	45	9:00

5.2.2. Największa jednoczesna ilość obiektów w interwale 15 min

W badanym scenariuszu wyznaczono największą liczbę obiektów każdej kategorii, znajdujących się w kadrze kamery w danym przedziale czasu. Do analizy wykorzystano filmy zebrane w ciągu „zwykłego” dnia (wtorek 30. marca w godzinach od 6:00 do 12:00).

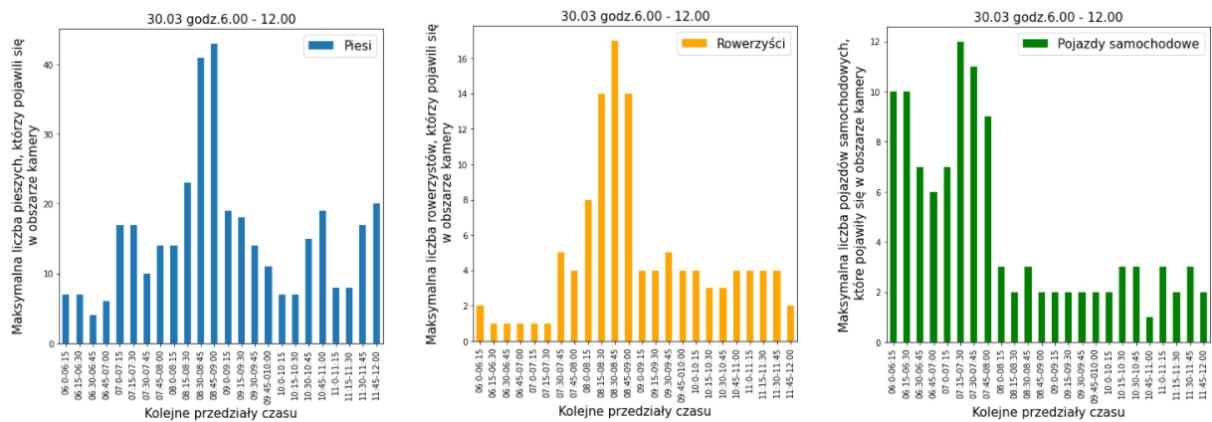
Dane dotyczące maksymalnej liczby pieszych, uzyskane dla kamery zamontowanej w lokalizacji Rynek Główny (Rys. 5.4) nie posiadają łatwo zauważalnej korelacji z danymi dotyczącymi sumarycznej liczby osób pojawiających się w obszarze kamery w kolejnych godzinach. Największe zagęszczenie rowerzystów wystąpiło w godzinach od 6:00 do 8:00 oraz o 11:00 i 12:00, co może być związane z podróżami do pracy oraz dostawami jedzenia w porze lunchu. Najwięcej pojazdów samochodowych znajdujących się równocześnie w obszarze kamery zarejestrowano pomiędzy godziną 7:00 a 8:00.



Rys. 5.4. Maksymalna ilość obiektów zarejestrowana przez kamerę w Rynku Głównym 30.03.2021r.

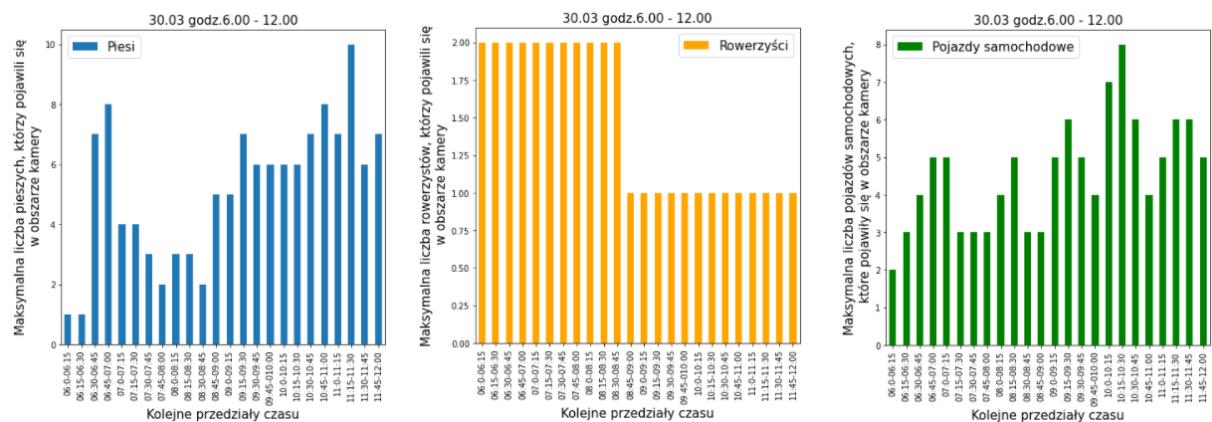
Dla kamery zamontowanej na ul. Grodzkiej, skierowanej w stronę Sukiennic, analogiczne dane zaprezentowano na wykresach 5.5. Ruch rejestrowany w obszarze kamery rozkłada się

bardziej regularnie niż w przypadku Rynku Głównego oraz dane tworzą podobny schemat dla każdej z badanych kategorii obiektów. Najwięcej pieszych oraz rowerzystów znajdujących się równocześnie w oku kamery zarejestrowano w okolicach godziny 9.30, natomiast najwięcej pojazdów samochodowych zliczono wcześnie rano - od godziny 6:00, z maksymalnym wynikiem o godzinie 7:00.



Rys. 5.5. Maksymalna ilość obiektów zarejestrowana przez kamerę z ul. Grodzkiej (widok na Rynek Główny) 30.03.2021r.

Wyniki zebrane dla kamery z ulicy Grodzkiej (widok hotelu Senackiego) zebrano na wykresach 5.6. Kamera obejmuje swoim zasięgiem mały fragment ulicy, w związku z czym odnotowane maksymalne wartości są niższe niż w przypadku pozostałych lokalizacji. Różnice w wartościach pomiędzy porównywanymi przedziałami czasu są niewielkie, więc trudno je interpretować. Najwięcej pieszych znalazło się jednocześnie w tym obszarze pomiędzy godziną 9:00 a 12:00, natomiast więcej rowerzystów wystąpiło przed godziną 9:00.

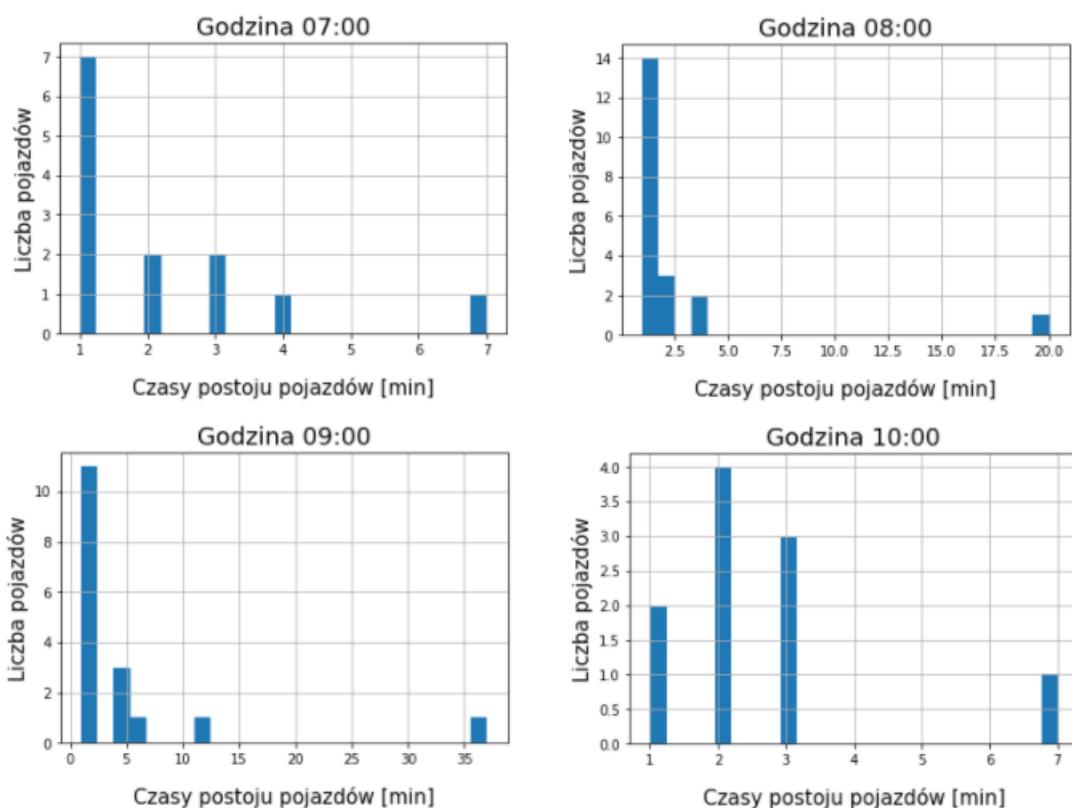


Rys. 5.6. Maksymalna ilość obiektów zarejestrowana przez kamerę z ul. Grodzkiej (widok z Hotelu Senackiego) 30.03.2021r.

5.2.3. Czasy postoju pojazdów

W opisywanym scenariuszu badano czasy postoju pojazdów mierzone w zwykły dzień - poniedziałek 29 marca, w godzinach od 7:00 do 11:00. W dalszej części podrozdziału przedstawiono wyniki pomiarów ze wszystkich lokalizacji poddanych badaniu, zebrane dla poszczególnych godzin. Zamieszczone w podrozdziale wykresy opisują zgromadzone dane w formie histogramu. Na wykresach przedstawiono ile pojazdów znajdujących się w obszarze kamery stało nieruchomo podczas danego przedziału czasu. Przed wygenerowaniem wykresów odfiltrowano wyniki, które wskazywały czas postoju jako nie mniejszy niż 1 min.

Wyniki pomiarów czasu postoju pojazdów zebrane dla kamery zainstalowanej w Rynku Głównym przedstawiono poniżej (Rys. 5.7).

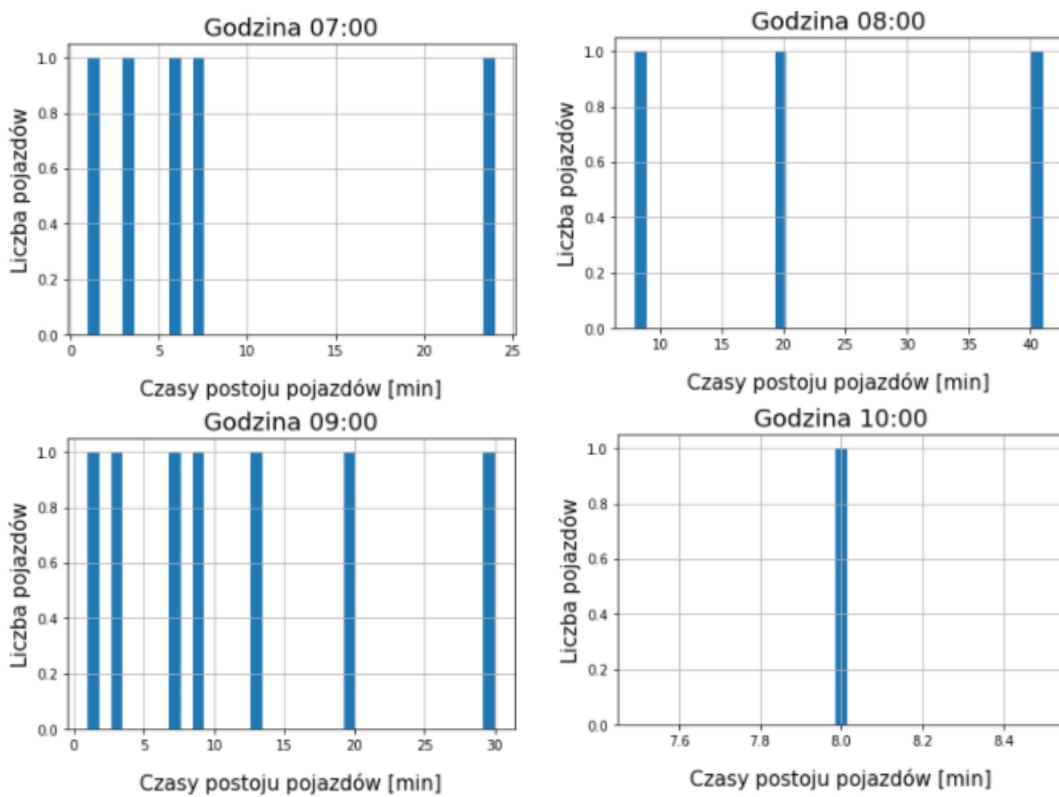


Rys. 5.7. Czasy postoju pojazdów - Rynek Główny.

W Rynku Głównym pomiędzy godziną 7:00, a 11:00 zarejestrowano w sumie 60 postojów, z których ponad połowa - 57% to postoje trwające nie dłużej niż 1 min. Z analizy danych przedstawionych na wykresach (Rys. 5.7) wynika, że najczęściej pojazdów zatrzymało się w badanej lokalizacji w czasie pomiędzy 8:00 a 9:00. Najdłuższy zarejestrowany postój odnotowano pomiędzy 9:00 a 10:00 i trwał on 37 min. Zakładając, że wszystkie wyniki większe niż 5 min są rzeczywistymi czasami przeprowadzonych dostaw - w godzinach od 7:00 do 11:00 w obszarze kamery zainstalowanej w Rynku Głównym zrealizowano 6 dostaw, gdzie średni czas dostawy wynosił ok. 15 min.

W kolejnym kroku poddano analizie czasy postoju pojazdów zebrane dla ul. Grodzkiej (widok z kamery skierowanej w stronę Rynku Głównego) przedstawiono na Rys. 5.8. W badanym

okresie czasu zarejestrowano 16 postojów, co oznacza, że odnotowano jedynie 1/4 z liczby postojów z Rynku Głównego. Kamera zainstalowana jest w taki sposób, że rejestruje dużo przejazdów, ale mało postojów - obejmuje małą przestrzeń parkingową, ale też duży plac, po którym mogą poruszać się pojazdy. Najwięcej pojazdów zatrzymało się na ulicy Grodzkiej pomiędzy 9:00 a 10:00. Najdłuższy postój odnotowano w przedziale pomiędzy 8:00 a 9:00 i trwał on ok. 40 minut. Średni czas dostawy w badanej lokalizacji (przy założeniu, że dostawa nie trwa krócej niż 5 minut) jest równy ok. 16 minut.



Rys. 5.8. Czasy postoju pojazdów - ul. Grodzka (Rynek).

Kamera zainstalowana na ul. Grodzkiej (widok z hotelu Senackiego) obejmuje swoim zasięgiem krótki fragment ulicy i skupia się głównie na kościele św. Apostołów Piotra i Pawła. W godzinach od 7:00 do 11:00 zarejestrowano tam tylko 4 postoje. W okolicy nie ma wielu sklepów i restauracji, więc odnotowane postoje najprawdopodobniej nie dotyczą dostaw, ale wizyt w rzecznym kościele. Pomiędzy godziną 7:00 a 9:00 w badanej lokalizacji nie wykryto żadnego postoju - dostawy na ulicy Grodzkiej realizowane są w innych miejscach, znajdujących się poza zasięgiem kamery. Opierając się wyłącznie na analizie filmów z tej kamery, liczbę odbywających się tam dostaw można by estymować na podstawie wykrywanych przejeżdżających samochodów, co nie byłoby wystarczająco miarodajne.

5.2.4. Porównanie dni świątecznych ze zwykłymi

W tabeli 5.1 przedstawiono wyniki detekcji (ilość wykrytych obiektów każdej kategorii) przeprowadzonej w Rynku Głównym na przestrzeni trzech dni tygodnia „zwykłego” (od soboty do poniedziałka) oraz w analogicznym okresie w Święta Wielkanocne. Analizę przeprowadzono

dla godzin porannych (od 6:00 do 12:00). Analizując wyniki otrzymane w każdej z par można jednoznacznie stwierdzić, że w okresie świątecznym liczba wykrytych obiektów poszczególnych kategorii w godzinach od 6:00 do 10:00 była widocznie mniejsza. Zaskakująco mały ruch zarejestrowany w Rynku Głównym w tamtym okresie mógł być spowodowany trwającą pandemią oraz możliwością zdalnego uczestniczenia w nabożeństwach świątecznych. Nieoczekiwana obserwacją dotyczącą danych zebranych w Poniedziałek Wielkanocny jest fakt, że po godzinie 10:00 na Rynku zaczęło się gromadzić więcej pieszych i rowerzystów. Przyczyną tego zjawiska mogła być słoneczna pogoda oraz chęć aktywnego wykorzystania „dodatkowego” dnia wolnego od pracy.

	niedziela zwykła			Wielka Niedziela		
	piesi	rowerzyści	pojazdy samochodowe	piesi	rowerzyści	pojazdy samochodowe
06:00:00	67	5	11	57	7	10
07:00:00	61	8	28	52	8	17
08:00:00	95	19	28	78	21	14
09:00:00	111	41	26	57	8	10
10:00:00	190	71	22	145	43	20
11:00:00	232	91	34	216	52	17

	sobota zwykła			Wielka Sobota		
	piesi	rowerzyści	pojazdy samochodowe	piesi	rowerzyści	pojazdy samochodowe
06:00:00	78	10	21	50	3	22
07:00:00	79	14	32	66	11	25
08:00:00	124	24	50	83	22	30
09:00:00	190	64	28	105	16	31
10:00:00	162	44	32	132	21	17
11:00:00	178	48	35	177	47	14

	poniedziałek zwykły			Poniedziałek Wielkanocny		
	piesi	rowerzyści	pojazdy samochodowe	piesi	rowerzyści	pojazdy samochodowe
06:00:00	89	21	42	45	11	0
07:00:00	119	52	65	47	3	16
08:00:00	139	49	92	68	16	21
09:00:00	146	51	77	107	21	23
10:00:00	172	48	32	206	65	23
11:00:00	197	66	36	257	101	25

Tabela 5.1. Porównanie wyników detekcji z dni zwykłych i świątecznych (godziny poranne).

W tabeli 5.2 zamieszczono porównanie wyników detekcji z całej doby, przeprowadzonej w poniedziałek zwykły oraz w Poniedziałek Wielkanocny w Rynku Głównym. Do szczegółowej analizy dobowej wybrano właśnie te dni ze względu na duże zróżnicowanie wyników. Porównanie dnia roboczego ze świątecznym pod kątem obserwowanego ruchu daje większe pole do analizy i interpretacji niż porównanie ze sobą dwóch dni wolnych.

Otrzymane wyniki bardzo dobrze odzwierciedlają nastawienie społeczne do szeroko rozumianych „wyjść” w ciągu dnia, w zależności od tego czy dany dzień jest dniem roboczym czy

wolnym od pracy. Analizując zebrane dane pod kątem odnotowanej liczby pieszych oraz rowerzystów można zaobserwować, że w Poniedziałek Wielkanocny w godzinach od 10:00 do 20:00 w Rynku Głównym zarejestrowano znacznie większy ruch niż w poniedziałek roboczy (wiersze wyróżnione szarym kolorem). Biorąc pod uwagę zsumowane wyniki detekcji przechodniów z każdej godziny z tego okresu dla obydwu dni, w Poniedziałek Wielkanocny wykryto w tym czasie o ok. 32% więcej pieszych niż w zwykły dzień. Posługując się analogicznym tokiem rozumowania, liczba rowerzystów „przechwyconych” w kamerze w Poniedziałek Wielkanocny w godzinach od 10:00 do 20:00 jest większa o ok. 43%.

Wyniki detekcji pojazdów samochodowych, przeprowadzonej w tych dniach w godzinach porannych (od 4:00 do 10:00), są skrajnie różne. Ruch samochodowy zarejestrowany w tym przedziale czasu w Poniedziałek Wielkanocny był bardzo mały - był to dzień wolny od pracy, również dla dostawców oraz pracowników większości restauracji. Porównanie liczby samochodów zarejestrowanych w pozostałych godzinach jest trudne, ponieważ otrzymane wyniki nie przedstawiają żadnej prawidłowości i rozkładają się raczej losowo.

	poniedziałek zwykły			Poniedziałek Wielkanocny		
	piesi	rowerzyści	pojazdy samochodowe	piesi	rowerzyści	pojazdy samochodowe
00:00:00	39	1	22	65	7	23
01:00:00	3	1	0	41	7	20
02:00:00	39	2	19	34	3	14
03:00:00	20	1	15	34	3	18
04:00:00	18	1	14	4	8	0
05:00:00	40	1	40	32	15	0
06:00:00	89	21	42	45	11	0
07:00:00	119	52	65	47	3	16
08:00:00	139	49	92	68	16	21
09:00:00	146	51	77	107	21	23
10:00:00	172	48	32	206	65	23
11:00:00	197	66	36	257	101	25
12:00:00	228	105	28	316	141	16
13:00:00	218	93	28	304	194	17
14:00:00	215	116	13	322	168	13
15:00:00	259	125	25	313	175	20
16:00:00	255	131	20	256	119	18
17:00:00	234	90	20	247	112	8
18:00:00	210	80	13	246	87	10
19:00:00	76	12	10	241	57	26
20:00:00	108	2	19	163	22	28
21:00:00	60	3	9	46	1	15
22:00:00	59	11	0	37	1	11
23:00:00	30	15	0	22	10	0

Tabela 5.2. Porównanie wyników detekcji w poniedziałek zwykły oraz świąteczny z całej doby.

6. Podsumowanie

Wynikiem niniejszej pracy magisterskiej jest licznik obiektów opracowany w oparciu o istniejącą implementację trackera *Norfair*, wykorzystujący algorytm detekcji *YOLO*. Praca została wykonana z wykorzystaniem Infrastruktury PL-Grid.

Badania opisane w rozdziale 5 niniejszej pracy przeprowadzono z wykorzystaniem stworzonego licznika, poprzez uruchamianie go na uprzednio zgromadzonych materiałach z kamer, z których transmisja online jest publicznie dostępna w internecie. Wyniki przeprowadzonych badań są wartościowe, ponieważ odzwierciedlają rzeczywiste zachowania pieszych oraz natężenie ruchu pojazdów w okresie obowiązywania obostrzeń pandemicznych. Z drugiej strony, z powodu trwającej pandemii, okres poświęcony gromadzeniu materiałów do analizy był dość specyficzny, przez co uzyskane wyniki mogą się znacznie różnić od wyników, które można było by uzyskać przeprowadzając analogiczne badanie np. w okresie wakacyjnym lub w czasie przed pandemią. Badania wykonano opierając się na zapisanych wcześniej plikach wideo, dzięki czemu możliwa była manualna weryfikacja działania licznika. Opracowany licznik jest w stanie również działać na danych odczytywanych ze strumienia wideo w czasie rzeczywistym. Zastosowanie takiego trybu działania licznika wymaga jedynie wykorzystania odpowiedniego parametru startowego - adres *URL*, pod którym udostępniono transmisję obrazu z kamery.

Opracowane rozwiązanie sprawdziło się wystarczająco dobrze w procesie analizy filmów z kamer zainstalowanych w wybranych lokalizacjach, ale nie jest idealne. Poniżej przedstawiono propozycje usprawnień dla powstałego licznika obiektów:

- Rozszerzenie zbioru danych w celu poprawy ciągłości detekcji. Podczas weryfikacji działania licznika zaobserwowano następującą zależność - podczas ruchu samochodu wjeżdżającego w zacienione miejsce zmienia się kontrast względem miejsca dobrze oświetlonego i obiekt często jest gubiony. Aby wyeliminować ten problem, należałoby rozszerzyć dane treningowe o obiekty bardziej zróżnicowane pod względem padającego na nie światła.
- Rozszerzenie funkcjonalności licznika o wyliczanie czasu postoju na zmieniającym się obszarze. Algorytm wyliczający czas postoju pojazdu działa poprawnie wyłącznie dla kamer statycznych, ale nie sprawdzi się na kamerach ruchomych (do stwierdzenia o tym czy obiekt stoi nieruchomo, czy nie, wykorzystano pomiar zmiany jego położenia w pikselach).
- Usprawnienie algorytmu poprawiającego ciąłość detekcji przez nadpisywanie właściwości zgubionych obiektów przez nowo wykryte w pobliżu. Może zdarzyć się przypadek, że w tym samym miejscu zostaną zgubione dwa obiekty tej samej kategorii - wówczas licznik może pomylić je ze sobą.

- Dodatkowo, wyniki byłyby jeszcze bardziej dokładne, gdyby kamery zamontowano niżej i gdyby obejmowały one węższy obszar. Na filmach poddanych analizie ludzie znajdujący się na najdalszym planie są tak mali, że detektor ich nie rozpoznaje.

Bibliografia

- [1] P. Dong i W. Wang. „Better region proposals for pedestrian detection with R-CNN”. *2016 Visual Communications and Image Processing (VCIP)*. 2016, s. 1–4. DOI: [10.1109/VCIP.2016.7805452](https://doi.org/10.1109/VCIP.2016.7805452).
- [2] X. Li i Y. Shi. „Computer Vision Imaging Based on Artificial Intelligence”. *2018 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)*. 2018, s. 22–25. DOI: [10.1109/ICVRIS.2018.00014](https://doi.org/10.1109/ICVRIS.2018.00014).
- [3] R. Arora. *Convolutional implementation of the sliding window algorithm*. <https://medium.com/ai-quest/convolutional-implementation-of-the-sliding-window-algorithm-db93a49f99a0>. [Online; accessed 28-February-2021]. Sty. 2020.
- [4] *Getting started with COCO dataset*. <https://towardsdatascience.com/getting-started-with-coco-dataset-82def99fa0b8>. [Online; accessed 24-July-2021].
- [5] *Getting started with COCO dataset*. <https://github.com/tryolabs/norfair/tree/master/demos/yolov5>. [Online; accessed 26-July-2021].
- [6] *Machine Learning Teoria i praktyka*. https://www.sas.com/pl_pl/insights/analytics/machine-learning.html. [Online; accessed 06-March-2021].
- [7] Piotr Skalski. *Make Sense*. <https://github.com/SkalskiP/make-sense/>. 2019.
- [8] Z. Zhao i in. „Object Detection With Deep Learning: A Review”. *IEEE Transactions on Neural Networks and Learning Systems* 30.11 (2019), s. 3212–3232. DOI: [10.1109/TNNLS.2018.2876865](https://doi.org/10.1109/TNNLS.2018.2876865).
- [9] J. Lee, J. Bang i S. Yang. „Object detection with sliding window in images including multiple similar objects”. *2017 International Conference on Information and Communication Technology Convergence (ICTC)*. 2017, s. 803–806. DOI: [10.1109/ICTC.2017.8190786](https://doi.org/10.1109/ICTC.2017.8190786).
- [10] Sharif Elfouly. *R-CNN (Object Detection)*. <https://medium.com/@selfouly/r-cnn-3a9beddf55a>. [Online; accessed 04-March-2021]. Lip. 2019.
- [11] Ross Girshick i in. „Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Czer. 2014.
- [12] ROZWÓJ MONITORINGU WIZYJNEGO W KRAKOWIE. https://www.bip.krakow.pl/?dok_id=87015. [Online; accessed 11-July-2021].
- [13] K. E. A. van de Sande i in. „Segmentation as selective search for object recognition”. *2011 International Conference on Computer Vision*. 2011, s. 1879–1886. DOI: [10.1109/ICCV.2011.6126456](https://doi.org/10.1109/ICCV.2011.6126456).

- [14] J. R. R. Uijlings i in. „Selective Search for Object Recognition”. *International Journal of Computer Vision* 104.2 (2013), s. 154–171.
- [15] A. Helwan i D. U. Ozsahin. „Sliding Window Based Machine Learning System for the Left Ventricle Localization in MR Cardiac Images”. 9 (2017). DOI: <https://doi.org/10.1155/2017/3048181>.
- [16] *WorldCam cały świat w jednym miejscu, Kraków - Kamery internetowe.* <https://www.worldcam.pl/kamery/polska/krakow/lista/25>. [Online; accessed 11-July-2021].
- [17] *YOLOv5.* <https://github.com/ultralytics/yolov5>. [Online; accessed 11-July-2021].
- [18] J. Redmon i in. „You Only Look Once: Unified, Real-Time Object Detection”. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, s. 779–788. DOI: [10.1109/CVPR.2016.91](https://doi.org/10.1109/CVPR.2016.91).
- [19] *Zarządzanie Dróżami Miasta Krakowa (<https://zdmk.krakow.pl/>).* <https://www.google.com/maps/d/viewer?mid=1NIH-VbeBqdyei7IrSb4KFEwg-Sg&ll=50.058324615317105%2C19.974245575037344&z=13>. [Online; accessed 13-March-2021].