POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

KIERUNEK INFORMATYKA

Projekt inżynierski

Stworzenie środowiska graficznego (w Unity) symulującego poruszające się obiekty

Autor: Mateusz Mazurek

Kierujący pracą: dr inż. Michał Staniszewski

Gliwice, Luty 2018 rok

Prace dedykuję żonie
w podziękowaniu
za wsparcie i cierpliwość.

Załącznik Nr 2 do Zarz. Nr 97/08/09

O Ś W I A D C Z E N I E

Wyrażam zgodę/nie wyrażam* zgody na udostępnienie mojej pracy dyplomowej/rozprawy doktorskiej*

....., dnia .....

.....
(podpis)

.....
(poświadczenie wiarygodności podpisu przez Dziekanat)

* właściwe podkreślić

Spis treści

1. Wstęp

Organizm człowieka posiada 5 podstawowych zmysłów: wzrok, słuch, smak, węch oraz zmysły somatyczne czyli te związane z receptorami na skórze jak dotyk czy odczuwanie ciepła. Prawidłowe działanie wszystkich zmysłów było sprawą kluczową dla przetrwania rodzaju ludzkiego i rozwoju gatunku w prehistorii. Jest także niezwykle istotne dla funkcjonowania człowieka w dzisiejszym świecie, w społeczeństwie. Zmysły umożliwiają nam komunikację z otaczającym światem, ostrzegają przed czyhającym niebezpieczeństwem, pomagają w zdobyciu pożywienia, odnajdywaniu drogi do celu czy choćby komunikowaniu się z innymi ludźmi. Zmysły są również narzędziem do dostarczenia rozrywki człowiekowi, w formie na przykład muzyki czy filmu. Przy tym ostatnim należy zatrzymać się i pochylić nad nim na dłużej, bowiem korzysta on z dobrodziejstwa zmysłu wzroku, który stoi na piedestale wśród pozostałych zmysłów. To właśnie zmysł wzroku dostarcza organizmowi największej ilości informacji i to prawdopodobnie bez niego najciężej funkcjonować człowiekowi samodzielnie w społeczeństwie.

Wraz z rozwojem cywilizacji oraz postępującym razem z nim rozwojem techniki człowiek w pewnym momencie historii zaczął tworzyć urządzenia, których sposób działania zaczerpnął właśnie (poprzez badanie własnego ciała) z mechanizmów, które ono zawiera. I tak człowiek na podstawie analizy własnego organizmu wynalazł na przykład mikrofon i głośnik bazujące na budowie ucha oraz krtani, ale stworzył także kamerę, której działanie oparte jest na zasadach optyki wykorzystywanych właśnie przez ludzkie oko

Kamera jako urządzenie rejestrujące obraz umożliwia gromadzenie olbrzymich ilości informacji zarówno do rozrywki jak i analizy rzeczywistości. W dzisiejszym świecie Kamery w formie monitoringu obecne są w coraz większej ilości miejsc. Wykorzystywane są do różnych celów takich jak: ochrona określonych miejsc, budynków czy ludzi. Naukowcy wykorzystują je do badania zachowań ludzi i zwierząt, analizy zjawisk pogodowych czy badania kosmosu.

Kamery nieustannie zbierające dane obecne są również przy drogach, którymi codziennie poruszają się ludzie. Monitorowanie dróg i analiza ruchu to obecnie standard nowoczesnego transportu. ze względu na możliwość jego wykorzystania do wielu rzeczy, np.: monitorowania natężenia ruchu i tworzących się korków w celu polepszenia przepustowości dróg lub

wyznaczenia alternatywnych tras, wykrywania wykroczeń drogowych (takich jak przejazd na czerwonym świetle) czy detekcji miejsc szczególnie narażonych na wypadki drogowe.

Organizm ludzki w sposób naturalny przystosowany jest do analizy i przetwarzania obrazu dostarczanego przez narządy wzroku. Inaczej ma się sprawa w przypadku nagrania z kamery w formie strumienia pozornie niezwiązanych ze sobą danych. Bez odpowiedniego oprogramowania komputer nie jest w stanie takich danych zanalizować ani ich nawet wyświetlić w formie przystępnej dla człowieka. Nie wspomniano tu o głębszej analizie tych danych takiej jak, analiza poszczególnych obiektów na obrazie, czy detekcja przesunięcia obiektu pomiędzy następującymi po sobie klatkami obrazu, co wymaga znacznie bardziej złożonych algorytmów i stanowi jeden z przedmiotów niniejszej pracy. Ma to tym większe znaczenie, że w dzisiejszym świecie coraz większy nacisk stawia się na automatyzację procesów. Oczywiście dążenie do większej automatyzacji i autonomiczności dotyczy również systemów monitoringu, które, jak opisano w tej pracy, coraz lepiej i przy coraz mniejszym udziale człowieka są w stanie wykrywać, analizować oraz samodzielnie podejmować decyzję dotyczące zaobserwowanego ruchu. To właśnie dążenie jest inspiracją do tworzenia doskonalszych tzn. obarczonych mniejszym ryzykiem błędu oraz większą precyzją programów służących do wykrywania ruchu i jego interpretowania na czym skupia się niniejsza praca.

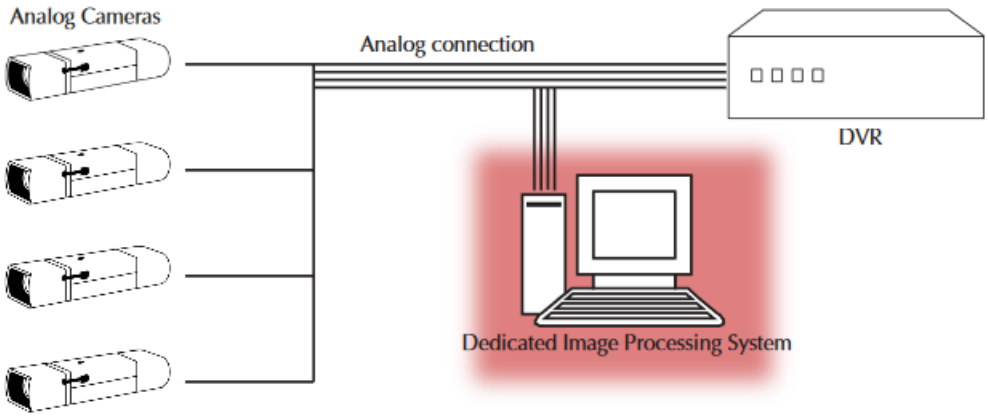
Celem projektu jest stworzenie środowiska bazującego na silniku graficznym Unity zawierającego elementy poruszające się generowanym losowo ruchem w różnych sceneriach z możliwością zmiany kamery (widok 1 i 3 osoby). Środowisko powinno umożliwiać wygenerowanie pliku wynikowego. Dostępna powinna być także funkcjonalność wykrycia ruchu i zaznaczenia położenia obiektu. Środowisko będzie wykorzystywane do oceny przydatności metod śledzenia obiektów w obrazach monitoringu miejskiego.

2. Analiza istniejących rozwiązań rynkowych

Na rynku funkcjonuje obecnie kilka, konkurujących ze sobą rozwiązań, systemów monitoringu wykorzystujących zaawansowane algorytmy analizy obrazu w celu detekcji obiektów i ich przemieszczenia oraz możliwości zaawansowanej symulacji ruchu drogowego. Pokazują one stopień zaawansowania i dopracowania technologii przetwarzania obrazu, już nie w zamkniętych, syntetycznych środowiskach takich jak ośrodki badawcze, ale w rzeczywistych warunkach i komercyjnych zastosowaniach. Wszystkie wymienione systemy są podobne w kontekście analizy obrazu jednak różnią się podejściem do architektury systemu, a część z nich wprost jest wykorzystywana do monitorowania lub symulacji ruchu drogowego. Stanowią one inspirację i wzór dla niniejszego projektu. W niniejszej pracy przedstawiono opis kilku takich systemów oraz obszary możliwych ich zastosowań.

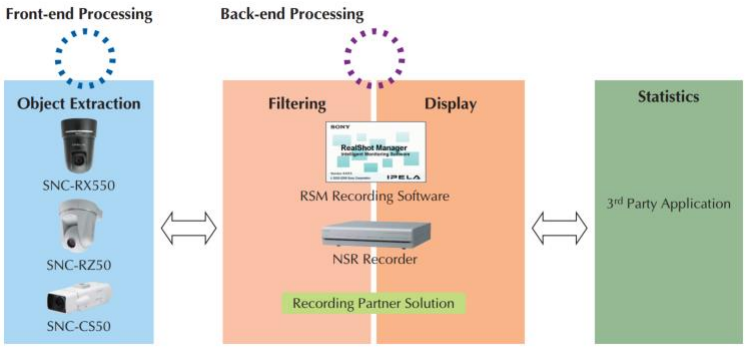
2.1. Sony Distributed Enhanced Processing Architecture (DEPA)

Starsze systemy monitoringu CCTV (rys. 2.1-1) obarczone są licznymi ograniczeniami. Największym z nich jest człowiek. Zdecydowana większość ludzi nie jest skupić się przez dłuższy okres czasu na jednej rzeczy, zwłaszcza jeśli byłoby nim analizowanie, na żywo, obrazu z kamery monitoringu. Kolejnym ograniczeniem człowieka jest równoczesna analiza obrazu przesyłanego z wielu źródeł. Organizm człowieka jest w stanie poświęcić całą swoją uwagę jednej rzeczy przez krótki czas, ale nie jest w stanie skupiać się na wielu rzeczach przez dłuższy czas. Dodatkowo czas analizy jak i wyszukiwania konkretnych materiałów starszych, analogowych systemów monitoringu, gdzie obraz z kamer nagrywany był na magnetycznych taśmach kaset VHS, jest w większości przypadków zdecydowanie za długi. Nawet przejście na czysto cyfrowe metody zapisu nie wyeliminowały wielu z wad starych systemów CCTV takich jak czasochłonna analiza zebranego materiału.



Rysunek 2.1-1 Klasyczne podejście [1]

Z tą świadomością, naprzeciw tym problemom wychodzi firma Sony wprowadzając na rynek w 2006 roku system DEPA. Jest to rozproszona platforma monitoringu przemysłowego. Zbudowana jest ona w architekturze rozproszonej co oznacza, że znaczna część obliczeń związanych z detekcją obiektów i analizą ruchu wykonywanych jest po stronie kamer (rys. 2.1-2) celu odciążenia głównego procesora co, według deklaracji producenta [1], przyczynia się również do zmniejszenia kosztów całej platformy oraz stosunkowo łatwym rozbudowaniu jej o kolejne kamery.



Rysunek 2.1-2 Architektura platformy DEPA [1]

W warstwę front-endu kosztowo-efektywne procesory w kamerach wykonują obliczenia związane z wykrywanie obiektów i oddzieleniem ich od tła, wykrywaniem ich ruchu lub jego braku, a następnie informacje te przesyłane są w formie metadanych do jednostki centralnej, która wychwytuje obiekty spełniające założone kryteria, archiwizuje dane i wyświetla informacje oraz zarządza komunikatami, w tym alarmami. Według deklaracji producenta

metadane nie wpływają w sposób istotny na ilość przesyłanych informacji. Wpływają natomiast pozytywnie na czas wyszukiwania konkretnych zdarzeń na materiale wideo, gdyż odnoszą się do konkretnego na nim momentu, eliminując konieczność przeszukiwania całego nagrania[1].

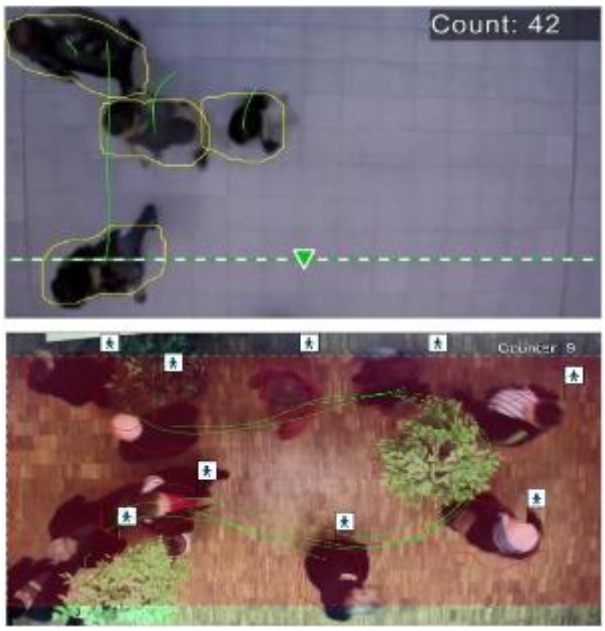
Platforma DEPA umożliwia rozpoznawanie obiektów zarówno statycznych jak i dynamicznych. W fazie pre-processingu kamery poprzez rozpoznanie cieni, roślinności oraz odfiltrowanie szumów spowodowanych niską jakością nagrania (np. przy wysokiej czułości matrycy wymaganej przy niskim poziomie oświetlenia), w celu uniknięcia fałszywych alarmów. Są w stanie poprawnie rozpoznać przenikające się i czasowo zasłonięte obiekty nawet przez obiekty o podobnym kolorze. Aby uzyskać takie możliwości kamery porównują, zamiast dwóch kolejnych, piętnaście następujących po sobie klatek. Rozpoznaje również pojawienie się lub zniknięcie statycznych obiektów będących wtedy częścią tła. Istnieje możliwość modyfikacji parametru czasu po którym obiekt jest uznawany za stacjonarny. System w celu minimalizacji fałszywych alarmów na statycznym obrazie, posiada również specjalne algorytmy zapobiegające takim zdarzeniom w wyniku na przykład zmieniającego się oświetlenia.

Jednostka centralna w fazie post-processingu oprócz filtrowania obiektów według odgórnych kryteriów jest w stanie rozpoznać i zakomunikować wystąpienie określonych zdarzeń takich jak: pojawienie się obiektu na zastrzeżonym terenie, zniknięcie obiektu z danego miejsca, obecność obiektu w danym miejscu przez określoną ilość czasu, przekroczenie wirtualnej linii (np. określającej ogrodzenie), przekroczenie dopuszczalnej ilości obiektów w danym obszarze, pojawienie się lub zniknięcie niepilnowanych obiektów (np. pojawienie się niepilnowanego bagażu na lotnisku). Umożliwia również pobranie parametrów obiektu takich jak ID, prędkość czy kierunek.

2.2. Bosch Intelligent Video Analytics

Kolejnym systemem, oferującym spore możliwości, obecnym na rynku jest system firmy Bosch. Omawiany system jest kolejną iteracją, rozwijanej rodziny systemów monitoringu tej firmy, w wersji 6.30. Przynosi ona usprawnienia względem poprzedniej wersji takie jak na przykład podwojenie odległości z której możliwa jest detekcja obiektów [6].

System ten oferuje szeroki zakres możliwości, m.in. funkcję zliczania osób, naruszenia strefy chronionej, klasyfikację i śledzenie obiektów, tryb muzealny i inne. Zasada działania jest podobna jak w przypadku systemu firmy Sony: analizowany jest obraz przesyłany z kamer przy użyciu specjalnych algorytmów bazujących między innymi na przepływie optycznym [2].

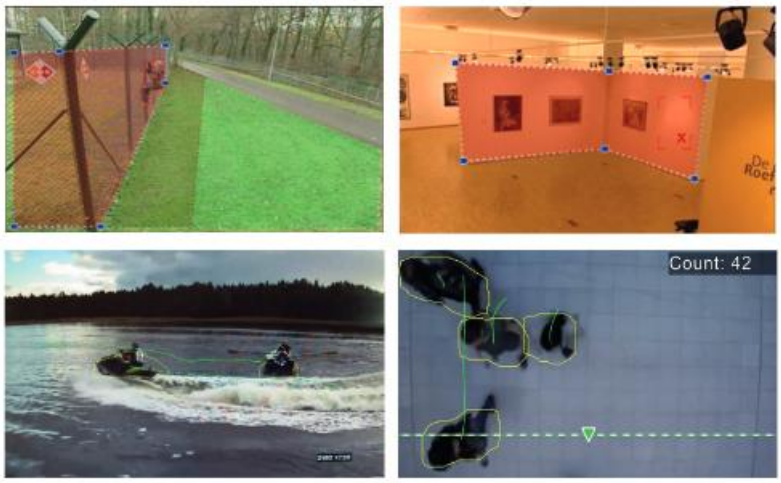


Rysunek 2.2-1 Liczenie osób [4]

System ten przy użyciu ruchomych kamer umożliwia śledzenie jednego obiektu. Producent wskazuje jednak na ograniczenia systemu związane z możliwością zmylenia systemu, gdy na scenie pojawi się więcej obiektów lub pojawi się ruch tła w postaci np. poruszanych wiatrem drzew. System jest także w stanie wykryć jedynie poruszające się obiekty[2]. Intelligent Video Analytics 6.30 posiada funkcję liczenia osób przebywających w polu widzenia kamery przy ustawieniu jej w odpowiednim położeniu co przedstawia rysunek 2.2-1. Dla zwiększenia precyzji liczenia firma Bosch zastosowała analizę 3D. Kolejną funkcją jest możliwość ochrony zadanej strefy. System zareaguje w momencie pojawienia się w chronionej strefie obiektu nieuprawnionego. Wskazano tutaj także na ograniczenie w postaci możliwego zmylenia analityki przez poruszające się zwierzęta [3]. Obecny jest również „tryb muzealny" umożliwiający na przykład, wykrycie dotknięcia eksponatu (na przykład obrazu) czy przekroczenie ustawionych barier.

Platforma firmy Bosch jest w stanie klasyfikować wykryte obiekty do kategorii: osoba wyprostowana, samochód, ciężarówka, motocykl. Tutaj również podane przez producenta ograniczenia mówią na przykład o małych zwierzętach i czołgających się osobach, które nie zostaną przez system sklasyfikowane.

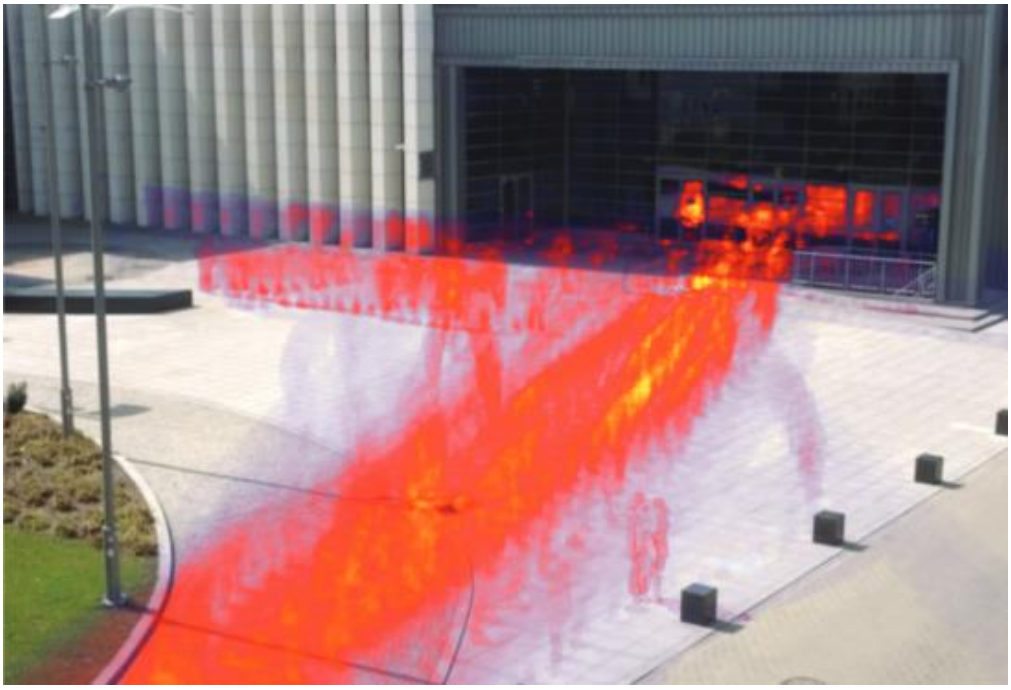
W związku z licznymi ograniczeniami systemu i dużym ryzykiem fałszywych alarmów związanych ze środowiskiem zewnętrznym można wysnuć wniosek, że omawiana platforma bezpieczeństwa jest lepiej przystosowana do ochrony wnętrz budynków lub terenu przemysłowego gdzie występuje niskie ryzyko zakłóceń ze strony poruszającej się roślinności i zwierząt.



Rysunek 2.2-2 Wizualizacja pracy systemu [6]

2.3. COMARCH Security Platform

Platforma bezpieczeństwa firmy Comarch to system o szerokich możliwościach z zakresu analityki bezpieczeństwa miasta oraz ruchu drogowego. System umożliwia analizę w czasie rzeczywistym obrazu z kamer monitoringu. Analiza może wykrywać szereg zróżnicowanych zdarzeń takich jak wykrywanie kolizji i jazda pod prąd oraz śledzenie jednego i wielu obiektów. Dostępne są również rzadziej spotykane możliwości w rodzaju wykrywania osób szwendających się czy przewracających, porzucania dużych odpadów, zliczanie osób na danym terenie czy dewastacje znaków lub budynków (graffiti).



Rysunek 2.3-1 Wizualizacja natężenia ruchu pieszych na terenie głównego kompleksu COMARCH w Krakowie [8]

Oprogramowanie analityczne platformy udostępnia funkcjonalności dotyczące analizy i skracania wideo. Umożliwia ona analizę natężenia ruchu drogowego jak i pieszych (rys 2.3-1). Widoczne jest tu także nieco inne podejście niż w przypadku systemu firmy Sony omawianego powyżej. Rozwiązanie firmy Comarch jest rozwiązaniem scentralizowanym. Podczas gdy system Sony wysyłał metadane umożliwiające szybkie odniesienie do zdarzenia na nagraniu, firma Comarch udostępniła funkcjonalność skracania nagrania wideo. Odpowiednie algorytmy analizują nagranie pod kątem wykrywania zdarzeń i obiektów usuwając fragmenty z pustą sceną oraz nakładając na siebie fragmenty w sposób nie zakłócający odbioru całości

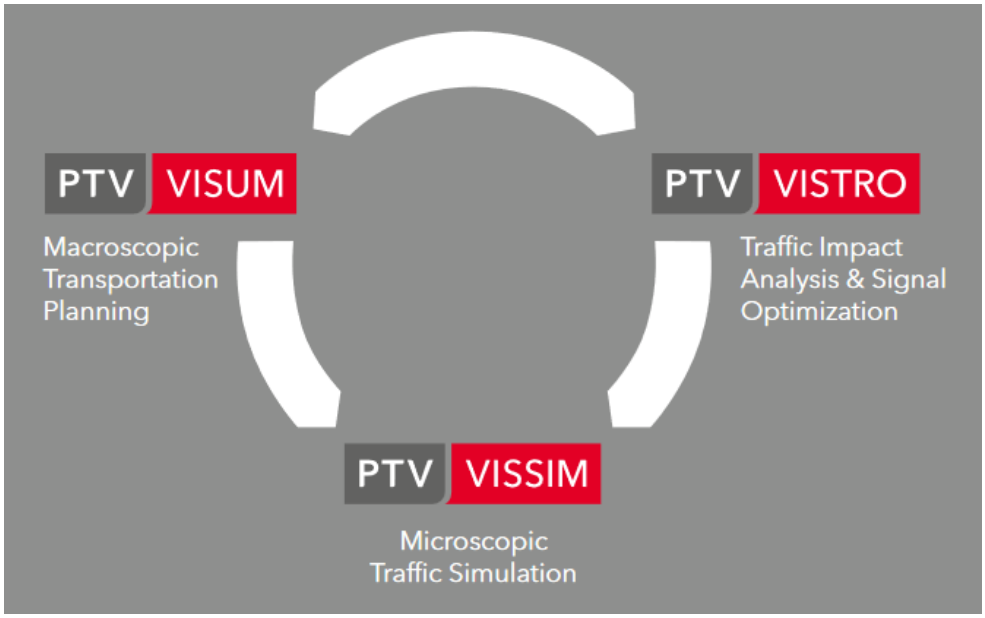
nagrania(rys. 2.3-2). Producent podaje, że wielogodzinne nagranie można skrócić nawet do kilku minut.

Rysunek 2.3-2 Skracanie wideo [8]

System Comarch Security Platform przeszedł pomyślnie testy i jest obecnie używany przez Policję np. w Krakowie.

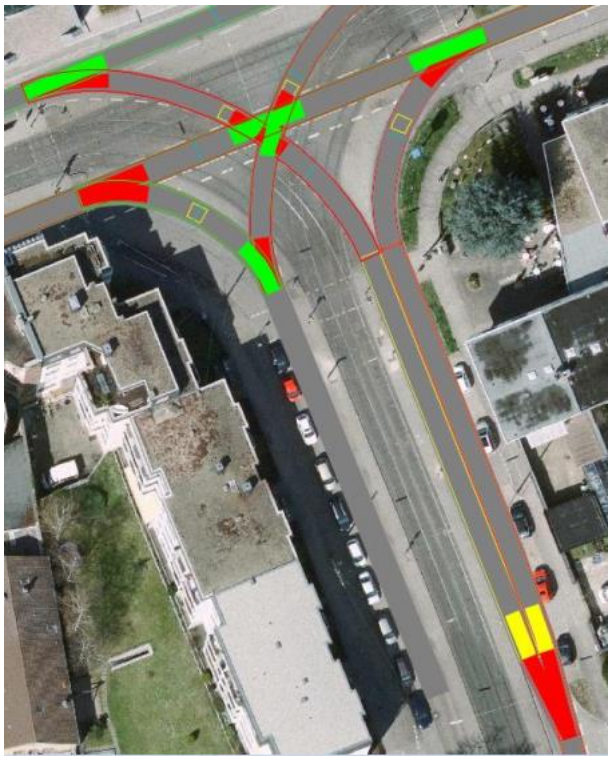
2.4. PTV VISSIM

Oprogramowanie niemieckiej firmy PTV Vissim jest kompleksowym narzędziem do mikro symulacji ruchu drogowego, czyli takiej, w której każdy uczestnik ruchu jest traktowany jako osobny obiekt. Wchodzi on w skład większej i jeszcze bardziej kompleksowej platformy do zarządzania ruchem w skład której wchodzą trzy moduły (rysunek 2.4-1): VISUM służący do symulacji makroskopowej (tzn. w skali miasta, województwa), VISTRO wyspecjalizowany pod kątem analizy korków i ich przyczyn oraz sygnalizacji świetlnej, oraz VISSIM umożliwiający symulację mikroskopową.



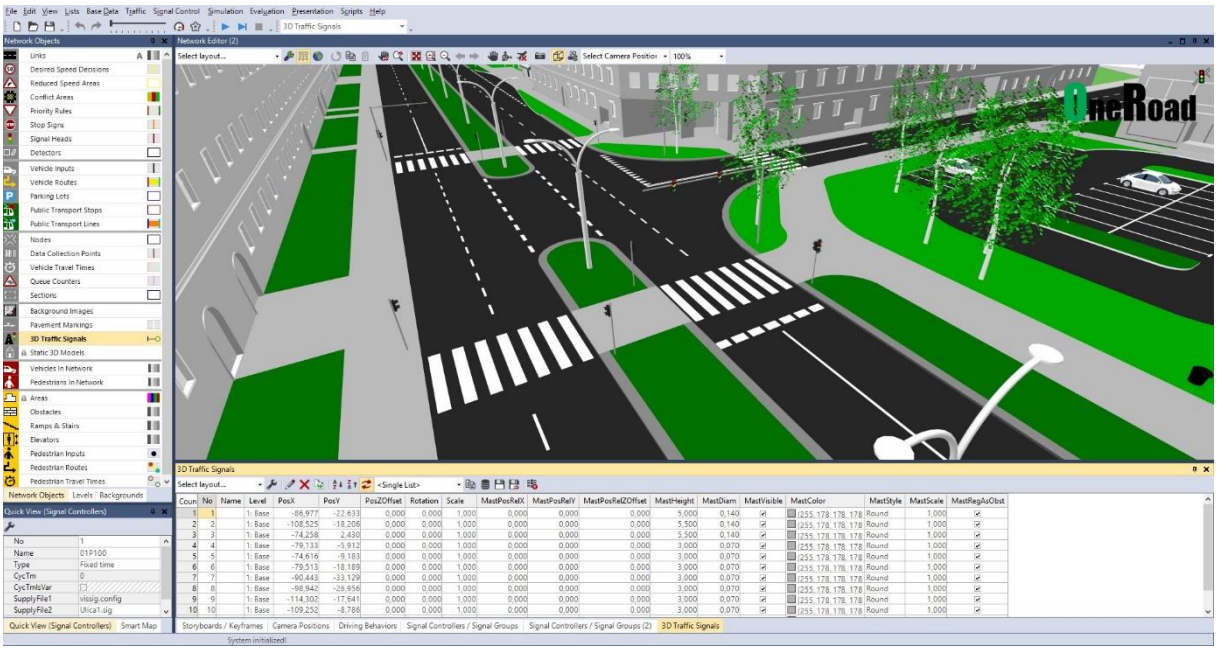
Rysunek 2.4-1 Platforma firmy PTV [20]

Oprogramowanie umożliwia realistyczną symulację poruszania się różnych typów obiektów, od samochodów osobowych i ciężarówki, przez pojazdy transportu publicznego takie jak tramwaje i autobusy, aż po jednoślady takiej jak rower czy motocykl oraz pieszych, którzy w celu zwiększenia realizmu oparci się na odrębnym modelu poruszania.



Rysunek 2.4-2 Modelowanie skrzyżowań [20]

Program umożliwia precyzyjne modelowanie pasów ruchu przy wykorzystaniu systemu *non-lane based behaviour* umożliwiającym symulację wyprzedzania w ramach jednego pasa ruchu (co ma umożliwiać na przykład symulację wyprzedzania roweru). Do tego posiada zaimplementowaną obsługę sygnalizacji świetlnej którą można w dowolny sposób konfigurować, system bowiem obsługuje zarówno sygnalizację stało czasową jak i akomodacyjną czy acykliczną. Podobnie symulacja ruchu pieszych odbywa się z uwzględnieniem sygnalizacji świetlnej. Umożliwia to optymalny dobór działania świateł dla potrzeb konkretnego skrzyżowania. System może mieć także zastosowanie przy uprzywilejowaniu komunikacji miejskiej.



Rysunek 2.4-3 Interfejs programu [19]

Pomimo ze system został stworzony w celu symulacji pojedynczych węzłów komunikacyjnych, jest on na tyle elastyczny że umożliwia symulacje całej sieci połączeń składających się z nawet kilkudziesięciu skrzyżowań zarówno na syntetycznych, trójwymiarowych makietach jak i na rzeczywistych zdjęciach czy mapach (rysunki 2.4-2 i 2.4-3).

Ponadto Platforma firmy PTV udostępnia własne API (Interfejs programistyczny) umożliwiające integrację własnych aplikacji, na przykład do sterowania sygnalizacją świetlną lub modyfikującą model jazdy czy modele zachowań symulowanych pojazdów.

3. Metody detekcji ruchu

3.1. Istota ruchu w obrazie

Przed przystąpieniem do analizy metod rozpoznawania ruchu w sekwencji obrazów konieczną do rozważenia kwestią jest sam ruch i jego istota. Słownik określa ruch jako zmianę pozycji która może być zauważalna, słyszalna lub wyczuwalna [9] lub jako zmianę położenia w stosunku do innych punktów dokonującą się w czasie [10]. Na potrzeby zagadnienia wykrywania ruchu w sekwencji obrazów dwuwymiarowych można uznać za ruch wszelkie przemieszczenie, obrót bądź skalowanie (zmianę rozmiaru) obiektu.

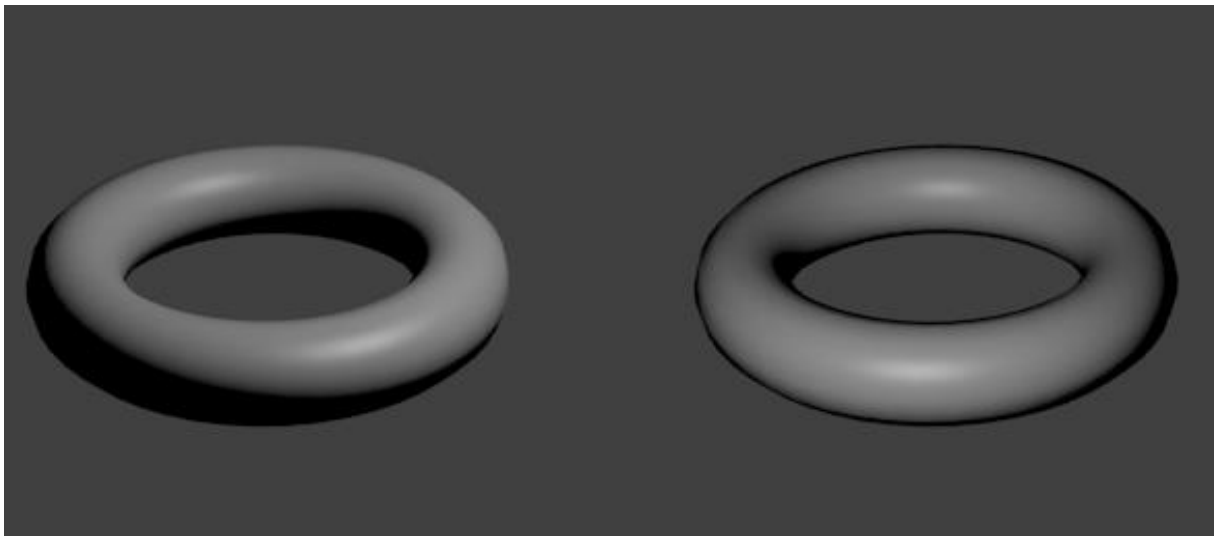
Rzutowanie obiektów trójwymiarowych, na dwuwymiarową płaszczyznę nagrania wideo, wiąże się jednak, z nieuchronną utratą części informacji dotyczących ruchu obiektów na scenie, co ma ścisły związek z pojęciami pola ruchu i przepływu optycznego.

Pole ruchu – jest polem powstałym w skutek zrzutowania wektorów rzeczywistego ruchu obiektu trójwymiarowego na płaszczyznę dwuwymiarową dając informacje o jednoznacznym przekształceniu. Proces rzutowania wektorów wiąże się jednak z utratą informacji o składowej prostopadłej do obrazu przez co nie jest możliwe odtworzenie pierwotnych wektorów ruchu [11].

Przepływ optyczny – Jest to pole wektorów umożliwiających przekształcenie jednego obrazu w drugi. Przepływ optyczny nie jest operacją jednoznaczną w sensie takim, że można uzyskać kilka takich pól, które umożliwią uzyskanie takiego samego obrazu wynikowego [7][12].

W zagadnieniu wykrywania ruchu w sekwencji obrazów można w pewnych warunkach zastosować przepływ optyczny. Różnica między polem ruchu, a przepływem optycznym polega jednak na tym, że to pierwsze pokazuje faktyczny ruch obiektu natomiast przepływ optyczny jego obraz. To znaczy że istnieją przypadki które uniemożliwiają uzyskanie pola ruchu z przepływu optycznego. Przepływ optyczny wskazując jedynie zmiany obrazu podatny jest, przy wykrywaniu ruchu, na zakłócenia w postaci na przykład zmiany oświetlenia czy nieodpowiednie teksturowanie[13]. Możliwość wystąpienia takich zaburzeń ukazana jest na przykładzie torusa (rys. 3.1-1).

Przedstawiony obiekt nie wykonał żadnego ruchu natomiast została dokonana zmiana w jego oświetleniu w związku czym uzyskano niezerowy przepływ optyczny tj. wykrycie ruchu którego w istocie nie było. W odwrotnym przypadku, biorąc po uwagę jednolity kolor i brak jakiejkolwiek niejednolitej tekstury torusa, brak zmiany oświetlenia przy jednoczesnym ruchu



Rysunek 3.1-3-1 Torus

obrotowym wzdłuż osi pionowej obiektu spowodowałby uzyskanie zerowego przepływu optycznego przy niezerowym polu ruchu i w konsekwencji brak wykrycia ruchu mającego miejsce.

Innymi trudnościami z którymi muszą zmierzyć się algorytmy bazujące na analizie obrazu to na przykład problem szczelinowy, występujący, gdy niecały obiekt znajduje się na scenie bądź jest on za duży oraz gdy występuje przesłanianie się obiektów częściowe lub całościowe, które powoduje że obiekt lub jego część chwilowo znika ze sceny (rys.3.1-2).

Rysunek 3.1-2 Przesłanianie się obiektów

3.2. Analiza metod detekcji ruchu

Metody detekcji ruchu można podzielić na:

- Metody różnicowe – analizujące różnice wartości składowych poszczególnych pikseli
- Metody bazujące na przepływie optycznym:
    - Gradientowe – analizujące pochodne intensywności obrazu
    - Częstotliwościowe – operujące na filtrach w dziedzinie częstotliwości
    - Korelacyjne – dopasowujące odpowiadające sobie obszary

3.2.1. Metoda różnicowa

Metoda różnicowa jest bardzo prostym sposobem na sprawdzenie zmian występujących w obrazie. Polega ona na prostym odjęciu od siebie wartości odpowiadających sobie pikseli w dwóch następujących po sobie klatkach obrazu. Jednak w rzeczywistych warunkach bardzo trudno o niezmienność wartości pikseli w kolejnych klatkach nawet tych należących do tła obrazu. Spowodowane jest to przez wiele czynników takich jak szumy wynikające z niedokładności matryc w kamerach czy naturalne zmiany oświetlania związane chociażby z cyklem dobowym czy przesłonięciem słońca przez chmury. Dla obejścia tego problemu można wziąć pod uwagę dodatkowy parametr jakim będzie czułość, dzięki któremu zignorowane zostaną zmiany mniejsze niż zadany parametr.

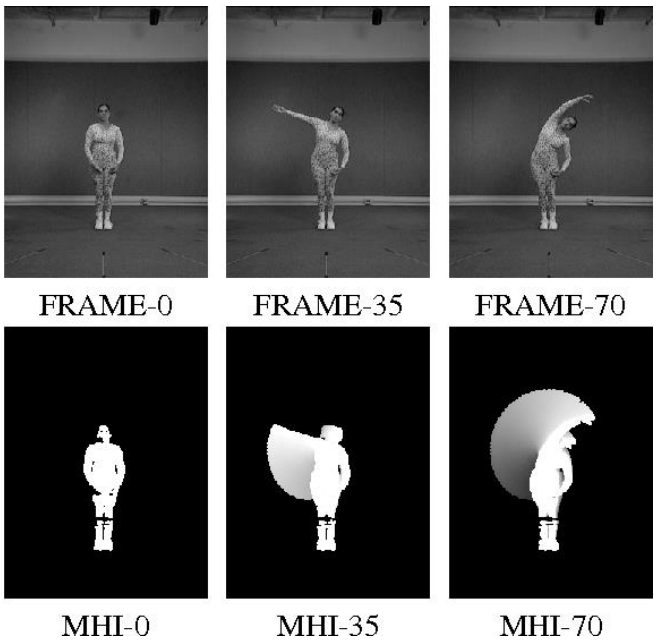
Zapis matematyczny będzie więc wyglądał następująco [11]:

$$d(i,j) = \begin{cases} 0 \; gdy \; |f(i,j) - g(i,j)| < \varepsilon \\ 1 \; gdy \; |f(i,j) - g(i,j)| > \varepsilon \end{cases}$$

Gdzie f(i,j) i g(i,j) to odpowiadające sobie piksele dwóch kolejnych klatek, a ε oznacza czułość. Wartość czułości może być dobrana w sposób doświadczalny.

Równanie można interpretować w sposób taki, że odpowiedni piksel obrazu wynikowego d(i,j) ma wartość równą 1, gdy nastąpiła zmiana wartości składowych piksela spowodowana ruchem obiektu przez ten piksel (gdy przynajmniej odjemna lub odjemnik są częścią poruszającego się obiektu) lub wystąpiły pewne zakłócenia, na przykład poruszenie kamery.

Metody różnicowe używane są również do tworzenia obrazów historii ruch (ang. Motion history image – MHI) co przedstawiono na rysunku 4.2.1-1.



Rysunek 3.2.1-1 Przykład tworzenia obrazu historii ruchu [18]

3.2.2. Metoda Gradientowa

Metoda gradientowa jest techniką przepływu optycznego wykorzystującego pochodne przestrzenne i czasowe. Jednak aby ją stosować wymagane są pewne założenia. Pierwszym z nich jest niezmienność natężenia światła na obrazie [14], co można zapisać jako:

$$\frac{dE}{dt} = 0$$

Kolejnym koniecznym założeniem jest niewielki ruch obiektu oraz przestrzenna spójność sąsiadujących punktów czy położenie na tej samej powierzchni i wykonywanie podobnych ruchów.

Metody bazujące na gradientach opierają się na lokalnych przybliżeniach szeregów Taylora dlatego wychodząc od równanie definiującego przepływ optyczny:

$I(\vec{x}, t) = I\left(\vec{x} + \vec{\delta}x, t + \delta t\right) \quad gdzie\ \delta\vec{x}\ jest\ przemieszczeniem\ punktu\ w\ czasie\ \delta t$[11]I wyprowadzając jego prawą stronę we wspomniany szereg Taylora:

$$I(\vec{x} + \delta\vec{x}, t + \delta t) = I(\vec{x}, t) + (\nabla I)^T\vec{\delta}x + \delta t I_t + O^2$$

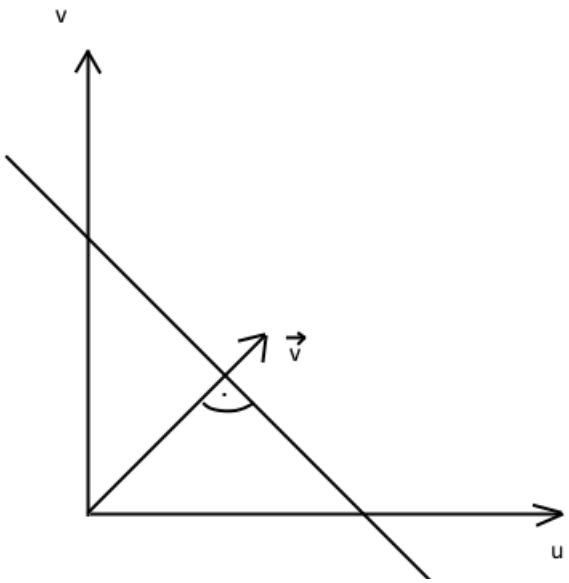
$gdzie\ O^2\ to\ pochodne\ wyższych\ rzędów\ które\ można\ pominąć$

Następnie po skróceniu równania o $I(\vec{x}, t)$ otrzymujemy:

$$(\nabla I)^T\vec{v} + I_t = 0$$

Gdzie $\nabla I = (I_x(x,t), I_y(x,t))^T$ stanowi przestrzenny gradient funkcji intensywności, natomiast $\vec{v} = (u, v)^T$ jest wektorem przepływu optycznego. W związku z tym, że jest to równanie dwóch niewiadomych [7] nie jest możliwe jego jednoznaczne wyliczenie wprost. Potrzebne są do tego dodatkowy zestaw równań i ograniczenia.

Równanie więc definiuje więc prostą ograniczającą z prostopadłym do niej wektorem składowej normalnej $\vec{v}$:



Rysunek 3.2.2-1 Ograniczenie prędkości optycznej [7]

Uzyskany w ten sposób wektor:

$$\vec{v}_{\perp} = \frac{-I_t \nabla I}{||\nabla I||_2^2}$$

pozwala na wyznaczenie składowych normalnych prędkości optycznej.

W metodzie gradientowej stosuje się podejście globalne i lokalne. Pierwsze umożliwia uzyskanie gęstszego pola przepływowego, natomiast podejście lokalne opierając się na wartościach prędkości normalnej (z sąsiedztwa rozpatrywanego punktu) umożliwia ograniczenie ilości wyliczanych punktów, a tym samym zmniejszenie złożoności obliczeniowej i czasu obliczeń.

3.2.3. Metoda częstotliwościowa

Metoda częstotliwościowa to grupa algorytmów wykorzystująca czułe na kierunek ruchu filtry częstotliwościowe, bazując na czasowo-przestrzennym rozkładzie energii w przestrzeni Fouriera, umożliwiając tym samym wykrycie ruchu, który nie został wykryty przez metody opierające się na dopasowaniu [16]. Dla metody gradientowej lub korelacyjnej trudnym przypadkiem może być losowy układ punktów, natomiast korzystając z metody częstotliwościowej i transformaty Fouriera, kierunek może być łatwo wyznaczony przy użyciu ukierunkowanej energii jako wyjściowego parametru.

Dla dwuwymiarowego, przesuwającego się sygnału intensywności obrazu, transformata Fouriera ma postać:

$$\hat{a}(\vec{k}, \omega) = \check{I}_0(\vec{k})\delta(\vec{v}^T\vec{k} + \omega)$$

Gdzie $\check{I}_0(\vec{k})$ oznacza transformatę Fouriera intensywności obrazu $I(\vec{x}, 0)$, a δ oznacza deltę Diraca, $\vec{k} = (k_x, k_y)^T$ oznacza częstość przestrzenną, a ω częstotliwość czasową.

Idąc dalej można otrzymać równanie ograniczenia przepływu optycznego w dziedzinie częstotliwościowej:

$$\vec{v}^T\vec{k} + \omega = 0$$

Wynika z tego że prędkość obiektu na obrazie dwuwymiarowym jest funkcją czasowo-przestrzennej częstotliwości tworząc płaszczyznę dla przestrzeni Fouriera na obrazie źródłowym [15][16].

Metody częstotliwościowe dzielą się na metody analizujące rozkład energii w przestrzeni Fouriera, np. metoda Adelsona i Bergena, którzy zaproponowali algorytmy wykorzystujące fakt równoznaczności rozpoznania ruchu z wykorzystaniem czasowo-przestrzennej orientacji czy wykorzystanie filtru Gabora, który jest funkcją Gaussa pomnożoną przez funkcję sinusoidalną, oraz na metody analizujące fazy sygnału w tej przestrzeni np. metoda Fleeta i Jepsona (również oparte na filtrach fazowych Gabora).

3.2.4. Metoda Korelacyjna

Metoda korelacyjna opiera się na dopasowywaniu do siebie fragmentów obrazów w sekwencji. Przy takim dopasowaniu można zdefiniować współczynnik korelacji jako:

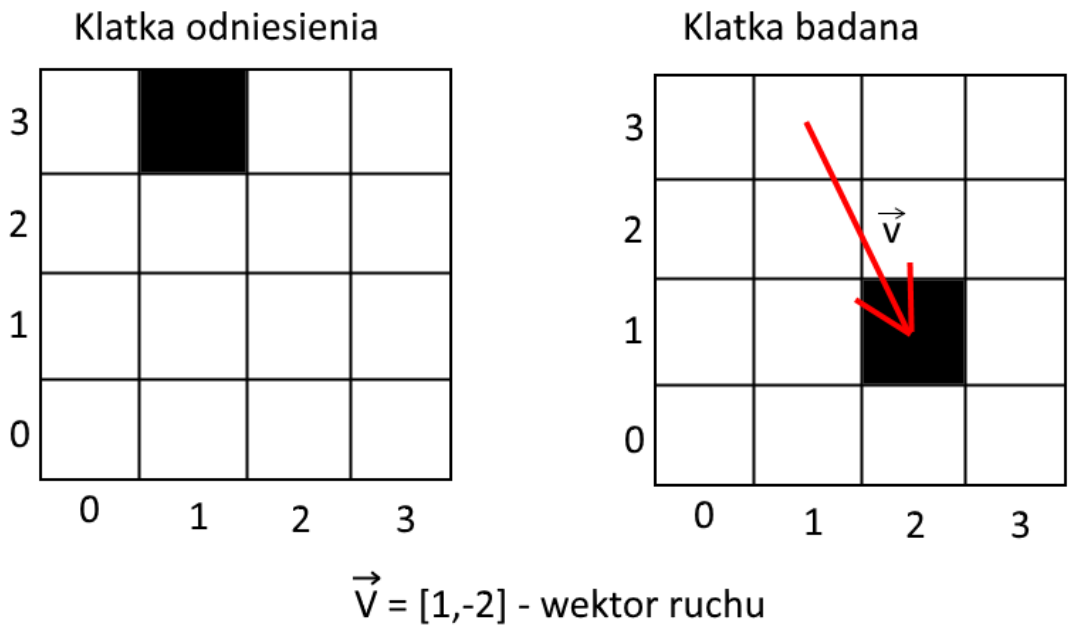
$$\int_D f(\vec{x} + \overrightarrow{\delta x}) g(\vec{x}) d\vec{x}$$

Znalezienie δx maksymalizującego funkcję prawdopodobieństwa w powyższym wzorze umożliwia znalezienie przemieszczenia między f i g. Kryterium korelacji można przedstawiać w różny sposób. Można przyjąć korelację bezpośrednią z zsumowanymi iloczynami wartości odpowiadających sobie punktów w obrazie, korelacją znormalizowanej średniej z różnicą punktów ze średnią wartością intensywności dla danego obrazu, która jest następnie mnożona i sumowana, a także sumy kwadratów różnic z sumą kwadratów różnic intensywności odpowiednich pikseli.

Metody te najczęściej stosuje się przy obrazie niskiej jakości lub z niewielką ilością klatek na sekundę gdzie brak odpowiedniej precyzji obliczeń uniemożliwia stosowanie metod gradientowych i częstotliwościowych[15].

W przypadku stosowania metod korelacyjnych konieczne jest przyjęcie założenia, że sąsiednie punkty należą do tego samego obiektu. Jest to możliwe również w przypadku obrotu lub skalowania.

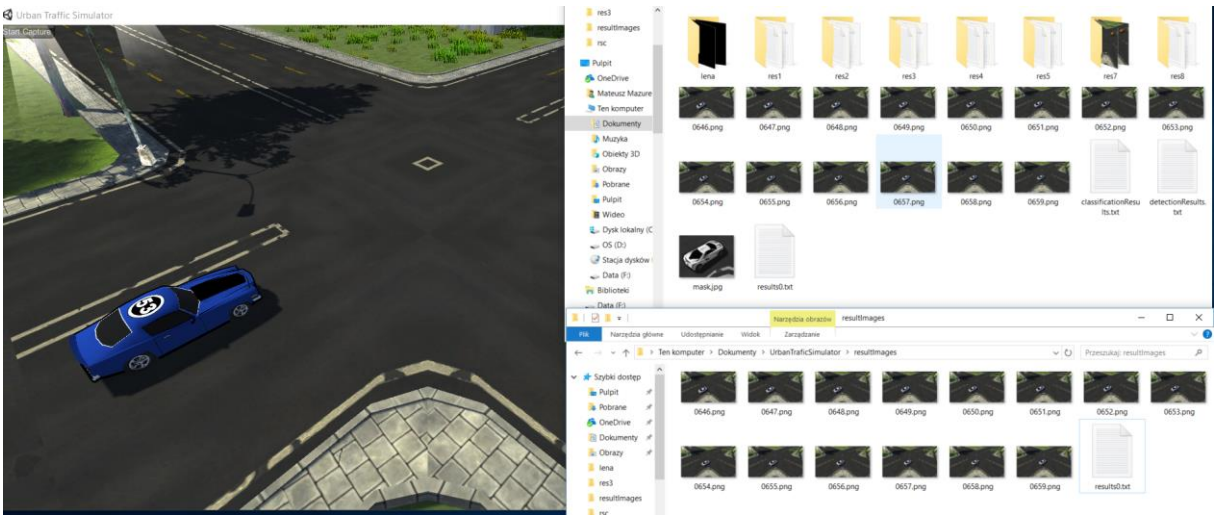
Najpopularniejszą metodą bazującą na korelacji jest metoda dopasowania bloków, którą można wykorzystać na przykład podczas kompresji wideo. Polega ona na podzieleniu obrazu na bloki, a następnie wyszukanie odpowiadających sobie bloków w kolejnych obrazach w sekwencji co umożliwi wyznaczenie wektora ruchu (rys 4.2.4-1).



Rysunek 3.2.4-1 Wyznaczanie wektora ruchu w dopasowaniu blokowym

4. Implementacja

Na potrzeby niniejszego projektu przygotowane zostały dwie aplikacje. Aplikacja UrbanTrafficSimulator mająca na celu symulację ruchu miejskiego i działania kamer monitoringu miejskiego (rys. 4-1) oraz pracującą w konsoli aplikację umożliwiającą analizę sekwencji obrazu w poszukiwaniu poruszających się obiektów. Obie aplikacje oraz zaimplementowana w nich funkcjonalność zostały omówione poniżej.



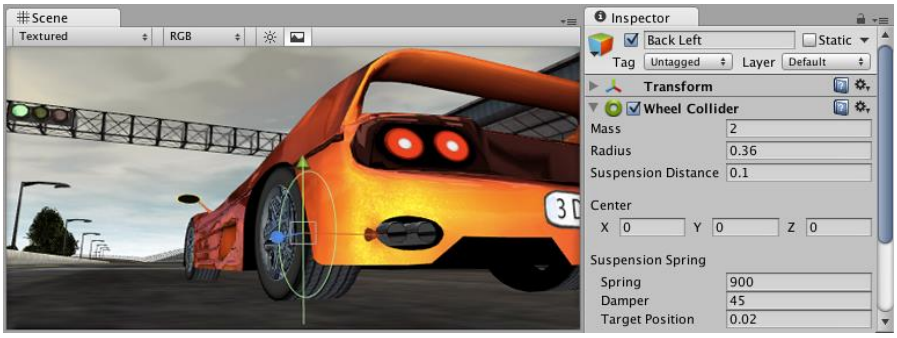
Rysunek 4-1 UrbanTrafficSimulator

4.1. Środowisko symulujące ruch obiektów

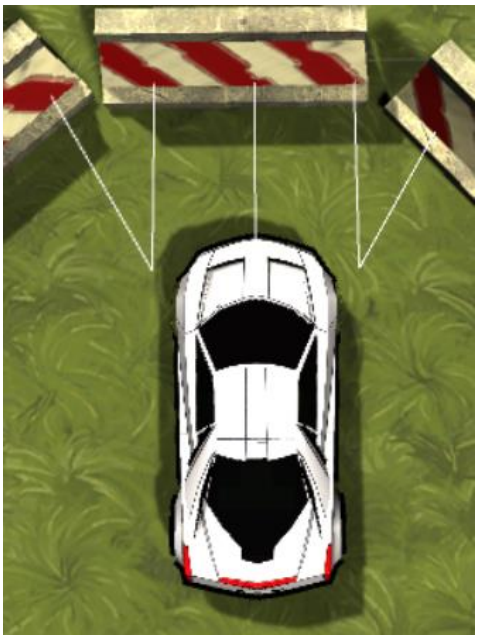
Środowisko symulujące poruszające się obiekty zostało zaimplementowane z użyciem silnika Unity umożliwiającego tworzenie zarówno dwuwymiarowych jak i trójwymiarowych gier umożliwiającego programowanie w językach takich jak C# czy UnityScript. Jest to obecnie jeden z najpopularniejszych silników do gier, zwłaszcza w segmencie tzw. gier niezależnych, ze względu na względną prostotę użytkowania i (w podstawowej wersji) darmowy model dystrybucji. Po raz pierwszy został zaprezentowany w roku 2005 na konferencji Apple's Worldwide Developers Conference. W niniejszej pracy wykorzystano wersję 2017.1.1f1 tego silnika.

W środowisku umieszczono trójwymiarowe obiekty reprezentujące poruszające się po drogach samochody. Do stworzenia modelu jazdy wykorzystano, wbudowaną w silnik Unity klasę, WheelCollider (rys.4.1-1). Jest to klasa umożliwiająca symulowanie zjawisk fizycznych zachodzących w prawdziwym świecie pomiędzy kołem samochodu, a nawierzchnią po której

porusza się samochód oraz jego zawieszeniem. Oferuje ona szerokie możliwości konfiguracji od parametrów takich jak wielkość czy masa koła, poprzez sztywność zawieszenia, moment obrotowy silnika aż po tarcia gumy o powierzchnię. Klasa ta używana jest do napędzania i sterowania pojazdami.



Rysunek 4.1-1 Unity WheelCollider [17]



Dla modeli aut przygotowano również system wykrywania kolizji, oparty na raycastingu (rys. 4.1-2), w celu umożliwienia reakcji na zbliżanie się do innego obiektu. Z pięciu punktów rozmieszczonych w modelu auta wysyłane są promienie (trzy na wprost, dwa skośnie), a następnie dokonywana jest ewentualne detekcja obiektu w zasięgu.

Rysunek 4.1-2 Detekcja obiektów z użyciem raycastingu

Samochody napędzane przez klasę WheelCollider poruszają się po trasie zbudowanej w formie grafu skierowanego (rys 4.1-3), w którym każdy wierzchołek wskazuje na wierzchołek następny i nie jest możliwy ruch pod prąd. W przypadku napotkania rozgałęzienia, czyli odniesienia wierzchołka do kilku kolejnych, wierzchołek następny wybierany jest w sposób pseudolosowy, co ma powodować równomierny rozpływ aut po makiecie i zapobiec przypadkom, w których jakaś droga nie byłaby uczęszczana.

Rysunek 4.1-3 Trasa w formie grafu skierowanego

Środowisko udostępnia funkcjonalność zapisania sekwencji zrzutów ekranu na dysku komputera wraz z informacją o położeniu widocznego samochodu na dwuwymiarowej płaszczyźnie obrazu. Środowisko umożliwia także przełączanie widoku pomiędzy kamerami symulującymi kamery monitoringu jak i poprzez kliknięcie na pojazd przejście w tryb kamery pierwszo – lub trzecioosobowej (rys. 4.1-4) Jednak ze względu na charakter algorytmu wykrywania ruchu jedynie obraz uzyskany z kamer monitoringu może być użyty do analizy.

Rysunek 4.1-4 Widok trzecioosobowy

Poniżej zamieszczono tabelę z krótką charakterystyką najważniejszych klas utworzonych w ramach programu:

Klasa	**Opis klasy**	**Główne metody**
CarEngine	Główna klasa sterująca pojazdami w programie. Zawiera metody sterujące, wyznaczające następny punkt docelowy oraz zarządzające przełączaniem kamer w ramach obiektu.	FixedUpdate – standardowa metoda UNITY uruchamiana w stałych odstępach czasu. Sprawdza stan obiektu i zarządza wywołaniem metod sterujących. Update – standardowa metoda UNITY wywoływana co klatkę obrazu. Zarządza wywołaniem metod sterujących zmianą kamery. SwitchCamera – metoda przełączająca kamerę pomiędzy widokiem pierwszo i trzecio-osobowym Sensors – metoda zarządzająca czujnikami zbliżeniowymi

		samochodów i wykrywająca zbliżającą się kolizję Drive – metoda sterująca prędkością jazdy ApplySteer – metoda sterująca kierunkiem jazdy
PathScript	Metoda stworzona na użytek edytora UNITY ułatwiająca projektowanie trasy przemieszczania się samochodów	OnDrawGizmosSelected – standardowa metoda UNITY umożliwiająca rysowanie obiektów w oknie podglądu sceny w edytorze UNITY. Użyta to narysowania podglądu połączeń na trasie przejazdu obiektów
CameraControllerScript	Klasa zarządzająca przełączaniem kamer monitoringu oraz pobraniem i zapisaniem danych wynikowych	Start – standardowa metoda UNITY wywoływana przy tworzeniu obiektu. Użyta do inicjalizacji stanu początkowego. OnGui- standardowa metoda UNITY zarządzająca interfejsem użytkownika. Użyta do obsługi przycisku rozpoczynającego pobieranie danych SaveData – metoda zapisująca tablicę prawdy do pliku tekstowego Update – użyta do pobierania zrzutów ekranu i aktualnej pozycji obiektów w celu umożliwienia zapisu wyników

		TakeCarsPosition – metoda zbierająca aktualną pozycję wszystkich obiektów na scenie setDefaultCamera – metoda przywracająca widok do domyślnej kamery na scenie FixetUpdate – użyta do zarządzania przełączaniem kamer
CarFactoryScript	Klasa służąca do zapełnienia makiety odpowiednią ilością obiektów	GenerateCars – metoda tworząca nowe obiekty samochodów w stałych odstępach czasu

Po uruchomieniu programu i wybraniu odpowiedniej rozdzielczości, pobranie danych wynikowych umożliwia przycisk znajdujący się w lewym górnym rogu ekranu. Jego wciśnięcie powodują rozpoczęcie pobrania danych, a ponowne wciśnięcie przerywa pobranie danych i zapisuje wyniki w folderze z dokumentami aktualnego użytkownika systemu operacyjnego.

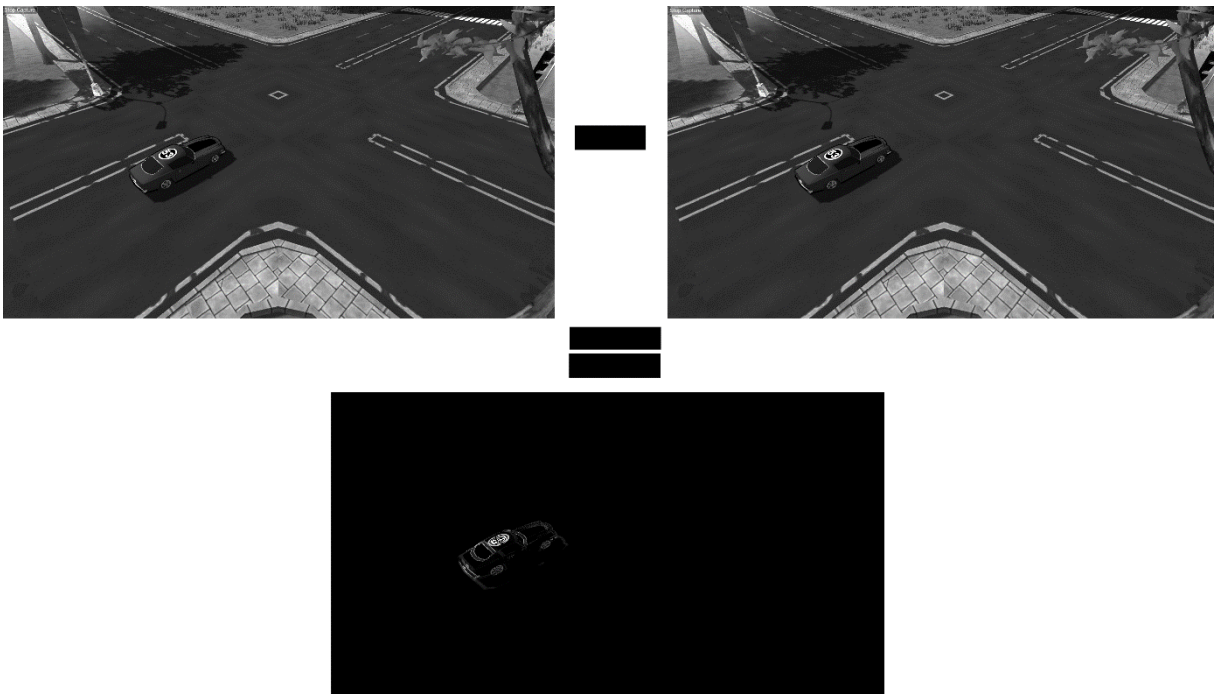
4.2. Analiza Ruchu

Do wykrywania ruchu została napisana osobna aplikacja wykorzystująca biblioteki OpenCV. Są to biblioteki udostępniające możliwości operowania zarówno na pojedynczych obrazach jak i również plikach wideo. Umożliwiają one wykonywanie od bardzo prostych czynności po operacje bardzo złożone, takie jak na przykład -wykonywane w tej pracy- wykrywanie ruchu. Biblioteki OpenCV są tworzone przez ogromną społeczność i udostępniane na zasadach OpenSource. Pierwotnie były opracowane przez firmę Intel i wydane w roku 2000. Napisane są w języku C++ jednak oprócz tego języka umożliwiono ich użycie również w Javie czy Pythonie na różnych systemach operacyjnych takich jak Windows, Linux, MacOS, IOS czy Android.

W przypadku niniejszego projektu do analizy ruchu wykorzystano omawianą wcześniej metodę różnicową porównującą dwa następujące po sobie zrzuty ekranu.

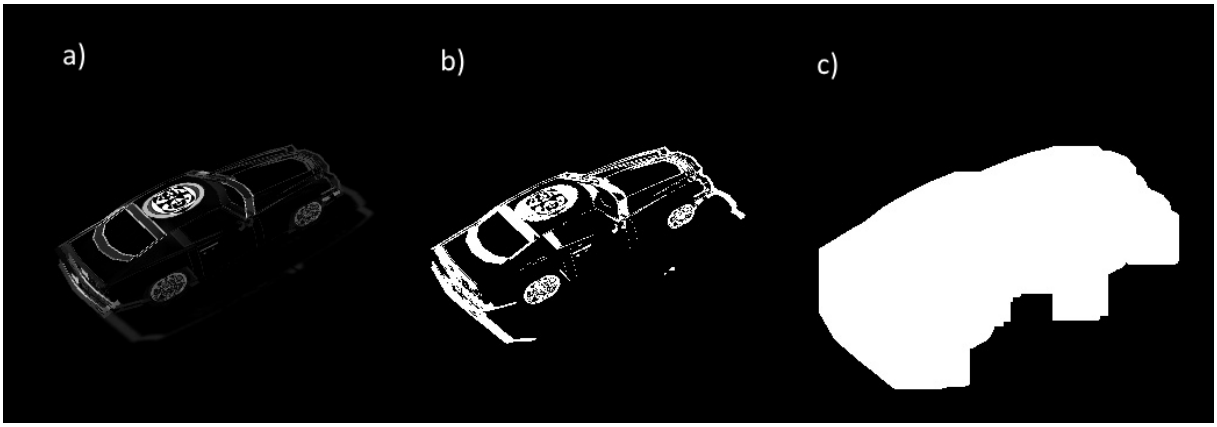
Do analizy obrazu wykorzystano następujące elementy bibliotek OpenCV:

- Metoda absdiff, która wykonuje „odjęcie” od siebie podanych dwóch obrazów (rys. 5.2-1)



Rysunek 4.2-4-1 Tworzenie obrazu różnicowego

- Metody Erosion i Dilation – metody te powodują odpowiednio zanikanie i ekspansję elementów tła obrazu (interpretowane jako jasne obszary). Metoda Erosion może być użyta w celu odfiltrowania ewentualnych szumów i innych drobnych zakłóceń. Następnie w celu wyeksponowania właściwego obiektu stosuje się metodę Dilation (rys. 5.2-2).



Rysunek 5.2-2 a) obraz różnicowy b) wynik metody Erosion c) wynik metody Dilation

Tak przygotowany obraz może być użyty do analizy obiektów (jasne fragmenty obrazu), do czego wykorzystano metodę findContours służącą do wykrywania krawędzi w obrazie. Znaleziony zbiór konturów obiektów można następnie przefiltrować ustalając minimalną oczekiwaną wartość powierzchni szukanego obiektu. Wyliczenie takiej powierzchni umożliwiają na przykład Momenty Geometryczne (również zaimplementowane w klasie Moments w bibliotece OpenCV).

Aby umożliwić poprawną pracę programu należy przed jego uruchomieniem skopiować zrzuty ekranu wraz z plikiem tekstowym zawierającym tablicę prawdy do folderu *rsc* znajdującego się w katalogu z plikiem *MotionDetectorV3.jar,* a następnie z okna poleceń systemu windows (ustawionym na folder z plikiem .jar) uruchomić program następującym poleceniem:

java -Djava.library.path=opencvdir -jar MotionDetectorV3.jar

gdzie *opencvdir* to katalog zawierający bibliotekę opencv_java2413.dll. Wyniki zostaną zapisane w folderze *rsc*.

5. Testy

Poniżej opisano wyniki testów przeprowadzonych w celu oceny skuteczności i możliwości praktycznego zastosowania stworzonego oprogramowania.

5.1. Metodologia testów i badane parametry

Testy prowadzono na wygenerowanych w programie UrbanTrafficSimulator próbkach danych (testy na makiecie oraz specjalnie przygotowanych scenach z 5, 10, 25 i 50 samochodami na każdej ze 100 wygenerowanych klatek) na które składa się sekwencja obrazów oraz plik tekstowy z tablicą prawdy zawierającą odpowiednio numer klatki, odpowiadający nazwie pliku z obrazem, oraz współrzędnie poziome i pionowe dwóch naprzeciwległych wierzchołków prostokąta opisanego na obiekcie.

Przykładowa zawartość pliku:

646 320 628 679 348

647 339 616 687 344

648 351 608 691 341

649 369 597 699 337

650 380 589 704 334

651 397 578 711 330

652 408 571 715 327

653 424 560 722 323

Gdzie pierwsza kolumna określa numer klatki obrazu, dwie następne współrzędne x i y jednego wierzchołka, dwie ostatnie, odpowiednio współrzędne x i y, wierzchołka naprzeciwległego względem pierwszego. Wielokrotne wystąpienie tego samego numeru klatki oznacza obecność na scenie kilku obiektów.

Podczas testowania dokładności analizy obrazu wykorzystano 4 parametry[21]:

- True Positive (TP) – prawidłowe wykrycie miejsca ruchu
- True Negative (TN) – prawidłowe wykrycie braku ruchu
- False Positive (FP) – błędne wykrycie ruchu tj. wykrycie ruchu, który nie miał miejsca, zbyt niedokładne wskazanie jego miejsca
- False Negative (FN) – błędne wykrycie braku ruchu tj. nie wykrycie ruchu, który miał miejsce

Na podstawie tych czterech parametrów obliczono następujące wartości:

- True Positive Rate (TPR) – czułość

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN}$$

- Positive Predictive Value (PPV) – precyzja

$$PPV = \frac{TP}{TP + FP}$$

Gdzie P oznacza ilość pozytywnych próbek.

5.2. Wyniki testów

Test dla dużej próbki danych ze zmienną ilością obiektów na makiecie:

P = 843	TP = 518	TN = 64
	FP = 966	FN = 325
	TPR = 0,614	PPV = 0,349

Wyniki podawane są do trzech cyfr znaczących.

Poniżej zamieszczono wyniki testu porównawczego, dla małej próbki z jednym poruszającym się samochodem, mającego sprawdzić wpływ szumów na skuteczność detekcji.

Wyniki w próbce bez dodanego szumu:

P = 14	TP = 14	TN = 0
	FP = 0	FN = 0
	TPR = 1	PPV = 1

Wyniki w próbce z dodanym ręcznie szumem:

P = 14	TP = 7	TN = 0
	FP = 7	FN = 7
	TPR = 0,5	PPV = 0,5

Wyniki ze sceny testowej z 5 samochodami (100 klatek):

P = 500	TP = 498	TN = 0
	FP = 0	FN = 2
	TPR = 0,996	PPV = 1

Wyniki ze sceny testowej z 10 samochodami (100 klatek):

P = 1000	TP = 699	TN = 0
	FP = 111	FN = 301
	TPR = 0,699	PPV = 0,863

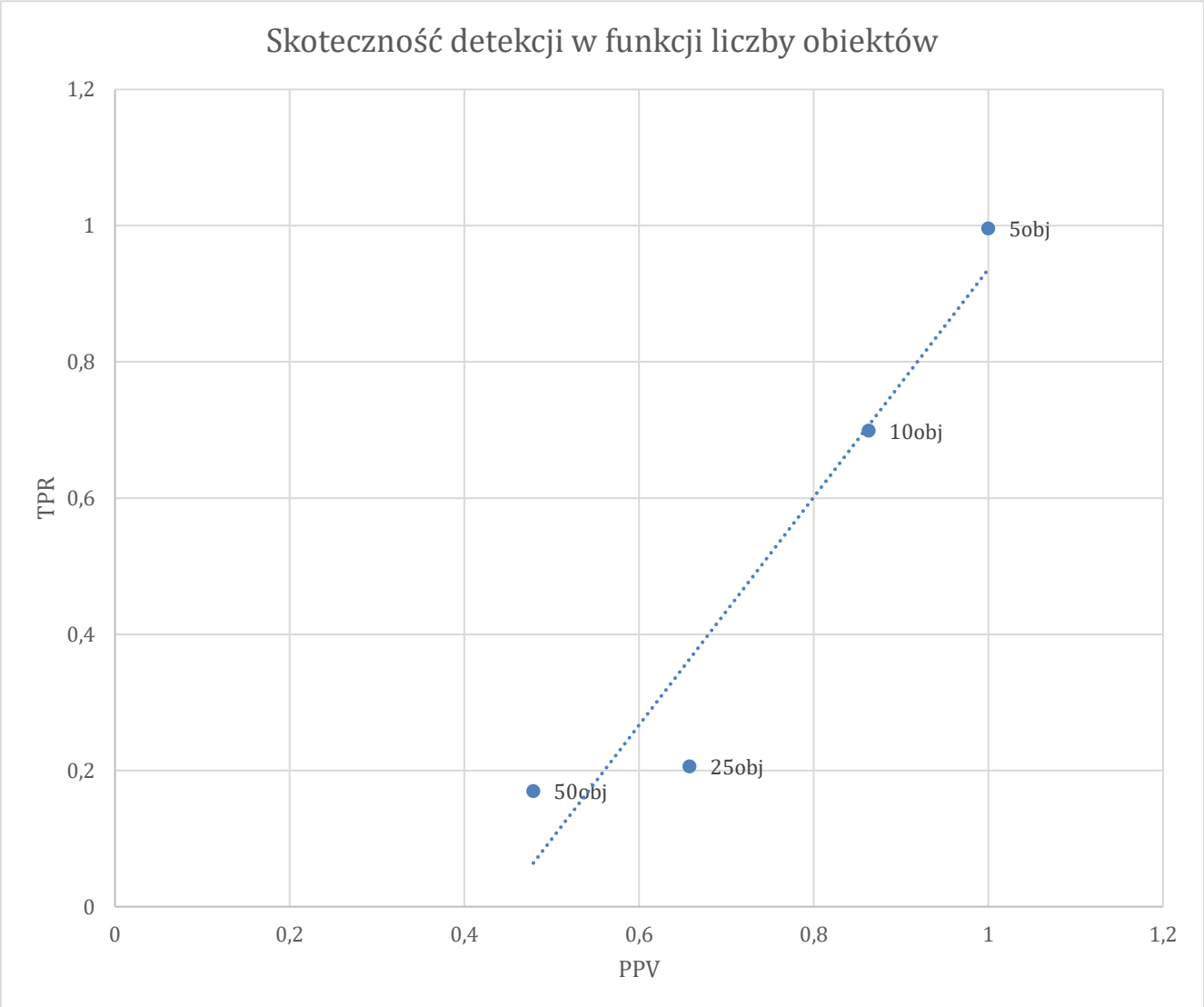
Wyniki ze sceny testowej z 25 samochodami (100 klatek):

P = 2500	TP = 516	TN = 0
	FP = 268	FN = 1984
	TPR = 0,206	PPV = 0,658

Wyniki ze sceny testowej z 50 samochodami (100 klatek):

P = 5000	TP = 848	TN = 0
	FP = 924	FN = 4152
	TPR = 0,1696	PPV = 0,479

Poniżej zamieszczono wykres obrazujący wyniki detekcji z próbek dla różnej ilości obiektów na scenie:



6. Analiza wyników

Test dla dużej próbki, wykazał przeciętną czułość algorytmu wynoszącą w przybliżeniu około 60%, przy bardzo niskiej precyzji wynoszącej około 35%, dla sekwencji pobranej z makiety miasta. Przyczyną takiego wyniku może być niska odporność programu na, wspomniany na początku tej pracy, problem szczelinowy pokazujący się w momencie wjeżdżania bądź opuszczanie sceny przez samochód.

Przy zlikwidowaniu problemu szczelinowego, poprzez ustalenie stałej liczby obiektów na scenie, uzyskana czułość wahała się od około 16% do nawet powyżej 99% w zależności od ilości obiektów znajdujących się na scenie. Zauważona została tendencja wykładniczego wzrostu czułości wraz ze zmniejszeniem liczby obiektów na scenie. Dodatkowo, na wyniki pomiarów może mieć również wpływ wielkość obiektów i odległości pomiędzy obiektami.

W związku z tym, że wykonane w tym projekcie środowisko, jest symulatorem czysto syntetycznym, co oznacza, że nie generuje żadnych fałszywych sygnałów i zakłóceń takich jak: szumy matrycy czy drgania kamery pod wpływem wiatru. Ze względu na to, zdecydowano się wykonać jeszcze jeden test. W celu sprawdzenia zdolności odfiltrowywania szumów przez algorytm, przygotowano krótką sekwencję obrazów z jednym pojazdem poruszającym się w obrębie sceny, w taki sposób, aby wyeliminować inne czynniki mogące mieć wpływ na rezultat (na przykład wspomniany problem szczelinowy). Podczas testu przeprowadzono dwa przejścia, pierwsze, bez modyfikacji, jako punkt odniesienia oraz drugie, gdzie do każdego obrazu w sekwencji (ręcznie w programie graficznym) dodano szum. W pierwszym przypadku, skuteczność detekcji wyniosła 100% jednak po dodaniu szumu wartość ta spadła o połowę, co pokazuje niską odporność algorytmu na tego typu zakłócenia.

7. Podsumowanie

Po głębszej analizie tematu, koniecznej do wykonania niniejszego projektu, stwierdzono dążenie w systemach monitoringu do jak największej automatyzacji. Firmy, tworząc swoje systemy monitoringu i analizy obrazu, dążą do jak największej autonomiczności takiego systemu i maksymalnego zmniejszenia koniecznego nadzoru czy ingerencji przez człowieka. W ten trend wpisuje się również wykonane w ramach tej pracy oprogramowanie. Poprzez symulacje i badanie zdolności wykrywania, umożliwia ono przetestowanie oprogramowania pod kątem -tak wymaganej obecnie- autonomiczności.

W tym projekcie nakreślono jedynie zalążek ogromu tematyki jaką jest analiza obrazu pod kątem wykrywania ruchu. Od lat pracuje się na poprawą jakości i wydajności algorytmów. W przypadku tego projektu również zbadano skuteczność detekcji i wykazano trudności związane z użyciem metod różnicowych i kierunek dalszego rozwoju zarówno tego algorytmu jak i w symulatorach ruchu drogowego.

Zaprezentowane w niniejszej pracy oprogramowanie, może znaleźć szerokie zastosowanie w systemach monitoringu wizyjnego oraz do ich kontroli i sprawdzenia dokładności wykonywanych przez nie pomiarów. W zaprezentowanym przypadku algorytm jest używany do wykrywania poruszających się samochodów, podobnie (choć oczywiście w mniejszym zakresie) jak umożliwia to wspomniana na początku platforma Comarch Security Platform, ale nic nie stoi na przeszkodzie, aby program wykrywający ruch został wykorzystany do detekcji innych typów obiektów, na przykład pieszych czy intruzów -jak ma miejsce w innych- w tym też w omówionych w niniejszej pracy systemach.

Program UrbanTrafficSimulator, ma na celu uniezależnienie osoby testującej systemy analizy obrazu z monitoringu od dostarczanego jej materiału i umożliwia samodzielne jego wygenerowanie z różnych kamer i w różnych sytuacjach. Nie jest jednak rozwiązaniem kompleksowym, tak jak ma to miejsce w przypadku omówionych systemów firm Sony, Bosch, Comarch czy PTV, dlatego wymaga dalszej pracy.

Podczas analizy i testów oprogramowania zauważono kilka możliwych kierunków, w które może zmierzać dalszy rozwój tego oprogramowania. Podczas testów zwrócono uwagę na to, że ze względu na charakter symulacji, program UrbanTrafficSimulator dostarcza obraz czysty,

pozbawiony jakichkolwiek zakłóceń czy zmian oświetlenia. W dalszym etapie rozwoju (w celu zbliżenia uzyskanych wyników bliższych wynikom rzeczywistym) można na poziomie programu zaimplementować dodawanie różnych czynników wprowadzających zakłócenia, takich jak różne warunki pogodowe, (w tym deszcz) jak również dodanie postprocesów symulujących niedokładności matryc stosowanych w kamerach, czy też niedokładności, spowodowane wysokim stopniem kompresji materiału. i wywołanych przez te czynniki szumów.

Elementem wymagającym dopracowania jest również kwestia problemu szczelinowego. Podobnie jest w przypadku aplikacji wykrywającej ruch, gdyż dalszy jej rozwój powinien skupić się na odporności algorytmu na takie właśnie zakłócenia. Dalszy rozwój oprogramowania może zakładać również system identyfikacji poszczególnych obiektów i możliwość badania ich toru ruchu czy też badanie intensywności natężenia ruchu (czego przykład można zaobserwować w systemie firmy Comarch). Można również zaimplementować kolejne typy symulowanych obiektów, na przykład pieszych.

Bibliografia

- [1] IP video monitoring systems – technical documentation. Sony Intelligence Video Analytics – Distributed Enhanced Processing Architecture (DEPA), Sony Corporation Grudzień 2006 ver.1.0
- [2] Bosch Security Systems Technical Note FW 6.30 Intelligent Tracking – 18 Sierpień, 2016
- [3] Bosch Security Systems Technical Note FW 6.30 Intrusion Detection – 21 Kwiecień, 2016
- [4] Bosch Security Systems Technical Note FW 6.30 Counting – 21 Kwiecień, 2016
- [5] Bosch Security Systems Technical Note FW 6.30 Object Classification – 21 Maj, 2016
- [5] Bosch Security Systems Technical Note FW 6.30 Museum Mode – 21 Kwiecień, 2016
- [6] Bosch Intelligent Video Analytics 6.30 – 2016, http:// www.boschsecurity.com
- [7] Determining Optical Flow – Berthold K.P. Horn, Brian G. Schnuck – Massachusetts Institute of Technology
- [8] Platforma Comarch Security – 2016, https://securityplatform.comarch.pl
- [9] Oxford Advanced Learner's Dictionary – Oxford University Press wydanie 8
- [10] Słownik Języka Polskiego – Państwowe Wydawnictwo Naukowe - https://sjp.pwn.pl/
- [11] Porównanie skuteczności algorytmów detekcji ruchu dla systemów wizyjnych ruchu ulicznego w wykrywaniu pojazdów – Marcin Nazimek, 2014, strony 13-35
- [12] Recovering Motion Fields: An Evaluation of Eight Optical Flow Algorithms - B. Galvin, B. McCane, K. Novins, D. Mason and S. Mills Computer Science Department University of Otago, New Zealand, 1998, strony 195-196
- [13] Motion Field and Optical Flow – Prof. Yuan-Fang Wang, Prof. Octavia Camps, Maj 2014, strony 15-18
- [14] Real-Time Optical Flow - Ted Camus, Maj 1995, strony 6-13
- [15] Wykrywanie i parametryzacja ruchu obiektów na podstawie sekwencji obrazów - Krzysztof Charusta, 2007, strony 7-9
- [16] The Computation of Optical Flow - S.S. Beauchemin and J.L. Barron, 1995, strony 5-17 i 24-27

- [17] Unity Manual – WheelCollider, https://docs.unity3d.com/Manual/class-WheelCollider.html
- [18] Hierarchical Motion History Images for Recognizing Human Motion – James W. Davis, 2001
- [19] OneRoad -VISSIM - http://www.oneroad.pl/vissim/
- [20] PTV VISSIM – http://www.vision-traffic.ptvgroup.com
- [21] Tablica pomyłek - https://pl.wikipedia.org/