WYDZIAŁ PODSTAWOWYCH PROBLEMÓW TECHNIKI
POLITECHNIKI WROCŁAWSKIEJ

Kontrola i statystyka obiektów

Projekt zespołowy

Autorzy:
Jakub Duda
Mikołaj Dukiel
Piotr Klepczyk
Mateusz Laskowski

Prowadzący:
dr inż. Łukasz Krzywiecki

WROCŁAW 2018

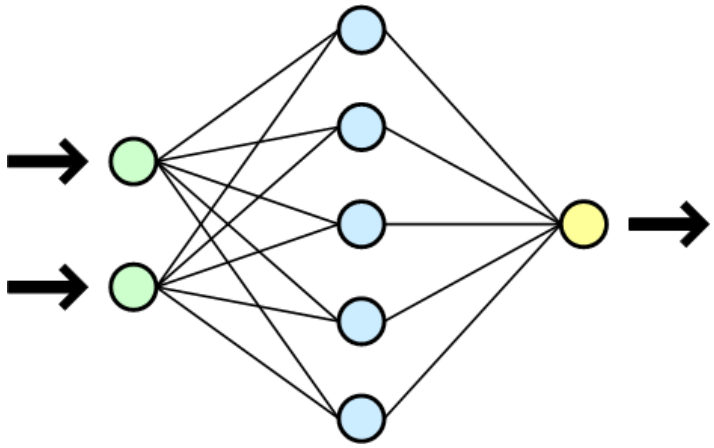
Spis treści

Wstęp

Celem projektu jest utworzenie programu do odczytu wideo, gdzie wideo wejściowe będzie analizowane i przetwarzane na dane statystyczne oraz wideo poglądowe, jako dane wyjściowe. Projekt będzie realizowany w małych krokach. Wpierw do danych wejściowych będziemy udostępniać obrazy, następnie nagrane wideo, a ostatnim, oraz najtrudniejszym celem będzie użytkowanie programu w czasie rzeczywistym. Oczywiście wideo będzie ograniczone jakościowo, jednak samo działanie powinno pozytywnie zaskoczyć. Celem oczywiście będzie również odpowiedni trening sieci neuronowych, które następnie pozwolą na rozpoznawanie cech wielu obiektów. Sieci neuronowe udostępnią nam dane, które my, twórcy będziemy chcieli odpowiednio opisać i wykorzystać do statystyki. Ostatnim celem będzie odtworzenie wideo po nałożonym filtrze, gdzie będzie można rozpoznać przypisane obiekty.

Technologie i zakres projektu. Naszym wspólnym pomysłem jest stworzenie programu, który będzie wykorzystywał jedną z nowszych technologii w świecie informatyki. Chcemy przedstawić oraz użyć w swojej pracy technologię z wykorzystaniem sieci głębokiego uczenia się przy przetwarzaniu dużych ilości danych. Technika ta opiera się na specjalnych sieciach neuronowych. Jednym z zastosowań tej techniki to możliwość rozpoznawania różnych obiektów w naturalnych scenach naszego życia. Wygodnym rozwiązaniem w tej technologii jest to, że nauczanie sieci rozpoznawania obiektów nie polega na wpajaniu mu szczególnych cech obiektów, lecz przedstawiamy mu przykładowe obiekty oraz jego nazwy, a w efekcie otrzymujemy sieć, która jest w stanie rozpoznawać dość szybko wiele różnych obiektów. Ten proces nazywa się treningiem sieci, gdzie wygląda to jak zwykła nauka dziecka. Przedstawia mu się przykłady, sieć je przetwarza i sama wychwytuje ważne cechy obiektu. Trening jest czasochłonny, ale bardzo opłacalny w porównaniu z manualną nauką poszczególnych cech. Technologia głębokiej sieci, czyta cały obraz jaki jej się podaje do odczytu jako zbiór cech, które później w sieci neuronowej decyduje o istotnych cechach i odnajduje je w nim obiekty. Dzięki czemu można znaleźć wiele obiektów za jedną analizą obrazu. Znając już technologię, na której będziemy opierać cały swój projekt, chciałbym przedstawić jego zakres. Projekt będzie obejmował trzy wyróżniające się segmenty. Pierwszy to obróbka obrazu, a nawet udoskonalenie do możliwości odtwarzania obrazu live. Wideo live lub po produkcyjne wideo będzie danymi wejściowymi do programu. Problemów w tym segmencie jest naprawdę sporo. Jednym z nich to rozdzielenie wideo na poszczególne klatki. Jednak nie celujemy w wideo najlepszej jakości, ponieważ wcześniej wspomniana technologia radzi sobie naprawdę dobrze w średniej jakości wideo. Następny segment to sieć głębokiego uczenia, która umożliwi nam z danych wejściowych odczytać obiekty, które sieć neuronowa rozpoznaje. Ostatnim segmentem będzie analiza i statystyka otrzymanych wyników z segmentu drugiego. Chcemy również, aby danymi wychodzącymi było wideo, na którym będą zaznaczone rozpoznane obiekty przez sieć neuronową.

Sieć neuronowa to połączenie elementów zwanych sztucznymi neuronami, które tworzą co najmniej trzy warstwy: warstwa wejściowa, warstwa ukryta oraz warstwa wyjściowa. Warto zaznaczyć, że warstw ukrytych może być wiele, w przeciwieństwie do warstw wejściowych i wyjściowych. Neurony sieci przetwarzają informacje dzięki temu, że ich połączeniom nadaje się parametry, zwane wagami, które modyfikuje się podczas działania całej sieci. Modyfikowanie wag w języku informatycznym zwane jest jako „uczeniem się sieci”.



Ryc.0.1 – przykładowy schemat warstw sieci neuronowych (kolor: zielony – warstwa wejściowa, niebieski – warstwa ukryta, żółty – warstwa wyjściowa)

W procesie uczenia się sieci tkwi istota i rzeczywista wartość sieci neuronowej. W procesie uczenia się nie projektuje się algorytmu kolejnego przetwarzania danych wejściowych, lecz stawia się sieci przykładowe zadania (w naszym przypadku rozpoznawania kształtów/obiektów), a następnie zgodnie z założoną strategią uczenia modyfikuje się połączenia elementów sieci, a dokładniej jej współczynniki wagowe poszczególnych połączeń. Mamy dwie wyróżniające się strategie uczenia sieci neuronowej. Pierwsza to uczenie nadzorowane, zwane „uczeniem z nauczycielem", które polega na porównaniu sygnału wyjściowego ze znanymi prawidłowymi odpowiedziami. Drugą metodą nauki to uczenie bez nadzoru, zwane inaczej „uczeniem bez nauczyciela", które polega na pozwoleniu sieci samej określenia czy dane elementy są ważne pomijalne, a następnie sama uogólnia wyniki i modernizuje wagi. Jest to dobry sposób, aby uogólniać wyniki, dzięki czemu sieć będzie w stanie rozpoznawać dane zbliżone do wzorca. Niestety ta metoda nie nadaje się na sam start nauki sieci neuronowej ponieważ, taka sieć nie będzie w stanie określić dobrego wzorca przez co będzie popełniać mnóstwo błędów. Najlepszym sposobem nauki jest zmieszanie obu strategii, lecz pierwszą z nich powinna być mimo wszystko strategia „uczenia z nauczycielem". Problemem sieci neuronowych jest to, że nie potrafi wykonać obliczeń, w których wymagane są duże dokładności.

Rozwój projektu. Program jak najbardziej może wyliczać natężenie ruchu na skrzyżowaniach. Dzięki algorytmowi rozpoznawania obiektów będziemy w stanie określać różne obiekty na pliku wideo, a nawet wideo prosto z transmisji kamery. Projekt może być rozwijany na wiele sposobów. Jednym z nich, które my widzimy to dynamiczne zwiększanie i zmniejszanie przepustowości skrzyżowań w zależności od poziomu obiektów oczekujących na zjazd. Mając ten system na wielu skrzyżowaniach można przewidzieć, gdzie fala wielu samochodów następnie się pojawi i umożliwić tak zwaną „zieloną falę", gdzie naprawdę pozwoli to na błyskawiczne usunięcie korków z głównych tras. Kolejnym rozwinięciem to wykrywanie pojazdów MPK, które miałyby priorytet. System odczytuje z kamery autobus MPK, przesyła dane o zwiększeniu priorytetu danego pasa do zaświecenia zielonego światła. Byłby to na pewno bardzo złożone działanie, ale w informatyce nie ma rzeczy niemożliwych. Zapewne byłoby jeszcze wiele rozwiązań, których w tym momencie nie widzimy, ale to pokazuje jak dużą przyszłość ma ten projekt pod względem marketingowym. Warto zaznaczyć, że również można rozwinąć naszą główną technologię, czyli sieci neuronowe. Nasze sieci można nauczyć rozpoznawać również i w nocy. Niestety cechy, które nauczyły się w dzień, nie będą w stanie dobrze funkcjonować w nocy. Wideo traci na jakości oraz w grę wchodzi mniejsza rozróżnialność obiektów (człowiek na

czarno ubrany nie będzie się wiele różnił od cienia człowieka). Będzie można nauczyć naszą sieć neuronową chociaż najważniejszych dla nas obiektów (człowiek, samochód, rower, autobus itd.), co umożliwi działanie nawet w nocy wcześniej wspomnianego przykładowego opisu możliwości wykorzystania programu (kontrolowany ruch w celu zmniejszenia zakorkowania na skrzyżowaniach). Jeżeli już jesteśmy przy rozwijaniu sieci, warto zauważyć, że można również, dzięki rozpoznawaniu szczególnych obiektów takich jak broń biała, niebezpieczne paczki, broń palna itd., być w stanie zwiększyć bezpieczeństwo, bądź zapobiec tragedią. A to wszystko mógłby robić dla nas program.

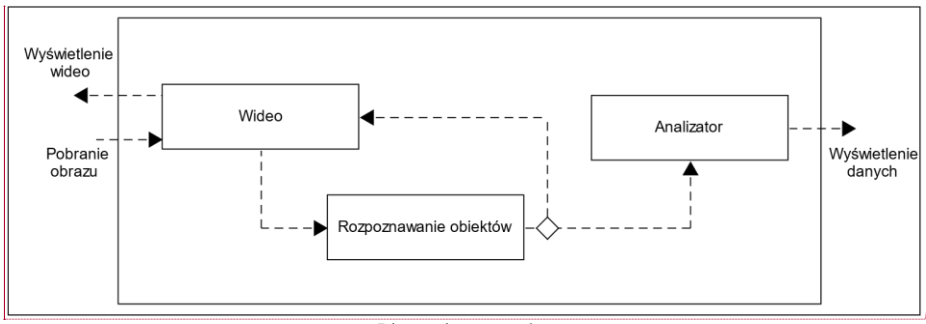
1. Analiza problemu

Opis zagadnień technicznych związanych z projektem:

Uczenie maszynowe (ang. machine learning) jest głównym celem samouczenia się maszyn. Ma to główne, praktyczne zastosowanie w dziedzinie sztucznej inteligencji do stworzenia automatycznego systemu potrafiącego doskonalić się przy pomocy zgromadzonego doświadczenia oraz nabywania na tej podstawie nowej wiedzy. Można zatem powiedzieć, że sprawiamy, że maszyny poznają świat w sposób empiryczny. Machine learning jest konsekwencją rozwoju idei sztucznej inteligencji i metod jej wdrażania praktycznego. Dotyczy rozwoju oprogramowania stosowanego zwłaszcza w innowacyjnych technologiach i przemyśle. Odpowiednie algorytmy mają pozwolić oprogramowaniu na zautomatyzowanie procesu pozyskiwania i analizy danych do ulepszania, a także rozwoju własnego systemu, sieć neuronowa, sieć nauczania, dostępne biblioteki i algorytmy spełniające funkcjonalność projektu, konwersja i obsługa wideo(kodowanie i konterner?, format)

1.1. Komponenty programu

Projekt składa się z trzech głównych komponentów. Nazwy poszczególnych komponentów: `Wideo`, `Rozpoznanie`, `Analizator`. Poniżej przedstawiono graficzny opis zależności pomiędzy komponentami.



Ryc1.1.1 Diagram komponentów systemu

Komponent `Wideo` - Program będzie przechwytywał obraz z kamery w stałych odstępach czasowych. Przechwycony obraz zostanie przetworzony, poprawnie sformatowany do dalszego przetwarzania wideo, a następnie zostanie rozbity na pojedyncze klatki, na których będzie funkcjonował następny komponent. Po tych operacjach, wideo jest gotowe, aby komponent `Rozpoznanie`, mógł poprawnie funkcjonować. Ogólną funkcjonalnością tego komponentu jest przygotowanie obrazu w taki sposób by spełniał wymagania kolejnego etapu działania programu. Komponent ten będzie odpowiadał również za wyświetlanie obrazu po wykryciu obiektów w taki sposób, że wykryty obiekt zostanie zaznaczony prostokątem z opisem co za typ obiektu został rozpoznany w tym miejscu. Dane, które nasz komponent otrzyma od komponentu `Rozpoznanie`, zostaną przetworzone, tak aby móc nanieść oczekiwane przez nas elementy na wideo wychodzącym z całego programu. Ten komponent odpowiada za dane wejściowe jak i dane wyjściowe, które użytkownik będzie w stanie zobaczyć jako wyniki pracy programu.

Z komentarzem [ML1]: Popraw Ryc -> dokładniejszy wygląd, stylistyka

Z komentarzem [PK2]: Poprawione, czeka na weryfikację 😉

Komponent `Rozpoznawanie` – Komponent ten, wykorzystuje algorytm do identyfikacji obiektów z plików wideo lub z plików graficznych. Identyfikacja obiektów odbywa się na obrazie otrzymanym na wejściu. Po wykonaniu pracy zostanie zwrócone, ile obiektów zostało rozpoznanych, jakiego były one typu i z jakim procentem dopasowania zostały one rozpoznane, te dane zostaną przekazane do `Analizatora`. Będą dane przekazywane również z powrotem do komponentu odpowiedzialnego za obsługę wideo, dane te będą zawierały wielkość i umiejscowienie rozpoznanego obiektu oraz jego typ. Algorytm dzieli obraz na siatkę 13x13 małych komórek. Każda komórka jest odpowiedzialna za 5 obszarów. Obszar opisuje prostokąt, który otacza obiekt. YOLO zwraca wynik prawdopodobieństwa, że obszar coś zawiera. Wynik nie mówi, jaki obiekt się znajduje tylko czy może w nim coś być. Dla każdego obszaru komórka również przewiduje klasę. Działa to jak klasyfikator (daje rozkład prawdopodobieństwa na wszystkich możliwych klasach. Ilość klas zależy od wersji YOLO. Yolov3 (najnowsza wersja) ma ich najwięcej my jednak używamy yolov3-tiny, aby przyśpieszyć proces rozpoznawania. Prawdopodobieństwo przewidzenia przez obszar i przewidywania klas są połączone w jeden końcowy wynik, który zwraca nam to, czy obszar zawiera jakiś typ obiektu. Na przykład żółty kontener zwraca nam wynik z prawdopodobieństwem 85%, że znajduje się w nim "dog". Ponieważ istnieje 13 x 13 = 169 komórek siatki i każda komórka przewiduje 5 kontenerów, mamy w sumie 845 ramek ograniczających. Wiele z tych kontenerów będzie miała niskie wyniki prawdopodobieństwa, więc zostawiamy tylko te, gdzie prawdopodobieństw jest większe od 30% (co można zmienić dodając flagę -thresh <val> podczas uruchamiania programu). W przykładzie z 845 wyników otrzymaliśmy tylko 3, ponieważ dały najlepsze wynik. Należy pamiętać że chociaż było 845 osobnych prognoz, wszystkie zostały wykonane w tym samym czasie - sieć neutronowa została uruchomiona tylko raz. I dlatego YOLO jest taki szybki.

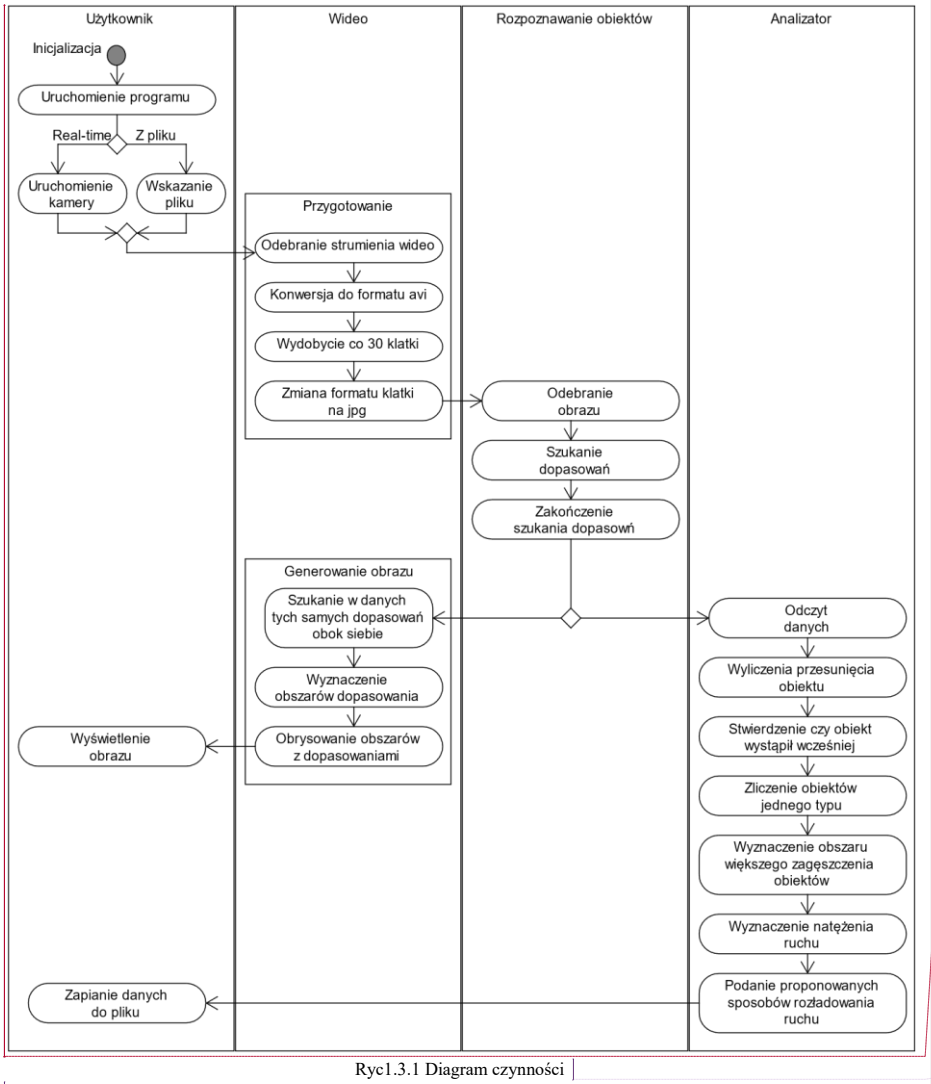
Komponent `Analizator` - Komponent ma rozpoznać, czy na sąsiednich klatkach widzimy ten sam obiekt. Wykorzystuje on do tego dane, które otrzyma z poprzedniego komponentu - `Rozpoznanie`. Jeżeli na dwóch sąsiednich klatkach widzimy ten sam pojazd, czyli na następnej klatce w niewielkim od siebie odstępie, w stosunku do poprzedniej klatki, lub, z powodu perspektywy, jest minimalnie większy lub mniejszy obiekt spełniający dokładnie te same cechy, np. czerwony sportowy samochód, to mamy bardzo duże prawdopodobieństwo, że na obu klatkach mamy do czynienia z dokładnie tym samym pojazdem. Gdy już stwierdzimy, że na kolejnych dwóch klatkach jest ten sam obiekt, to liczmy jego prędkość, czyli o ile się on przesunął w czasie między dwoma klatkami, który jest stały i wynosi 0,5 sekundy. Wykorzystamy do tego współrzędne prostokąta, który zaznacza rozpoznany z poprzedniego komponentu obiekt. Jego przesunięcie, odpowiednio przeliczone w skali, będzie wynikiem tej funkcjonalności. Komponent również zliczy natężenie ruchu, czyli sprawdzi, jak wiele pojazdów znajduje się w tym samym momencie na drodze oraz przepustowość, czyli jak dużo opuściło skrzyżowanie w ciągu jednego cyklu. Weźmie on również pod uwagę typ pojazdów, gdyż przykładowo jedna ciężarówka zajmuje zdecydowanie więcej miejsca od kilku motocykli. Na podstawie danych oraz ustawień własnych, określi on również, jakie działanie dla sygnalizacji świetlnej będzie najkorzystniejsze, czyli proporcjonalnie do natężenia danej strefy ruchu będzie dobierał długość trwania świecenie światła w cyklu świetlnym dla tych stref. Odpowiada on również za generowanie wniosków na podstawie wcześniej opisanych i wyuczonych danych. Do ich sformułowania będą potrzebne takie informacje, jak długość cyklu świateł, liczba pasów w danym kierunku i tym podobne. Ostatnią funkcjonalnością tego komponentu będzie prowadzenie statystyk na podstawie gromadzonych danych, które przyczynią się do rozwoju algorytmu i jeszcze większemu usprawnieniu ruchu drogowego. Sprawdzi on w jakim okresie natężenie jest największe, a w jakim najmniejsze. Porówna jakie ustawienie dało największą przepustowość, a także sprawdzi, który z samochodów najdłużej czasu spędził na skrzyżowaniu. To wszystko zostanie wypisane do pliku.

NOWY 1.2 Scenariusz

Z komentarzem [ML3]: Dodać opis użytkownika (scenariusz jego działania), WYMAGANE CO MUSI ZROBIC, JAK URUCHIMI, CO MUSI BYĆ SPEŁNIONE. CZEGO MOŻE OCZEKIWAC UZYTKOWNIK PODCZAS DZIALANIA. PRZERWANIE PROGRAMU

1.3. Diagram czynności

Poniżej przedstawiono diagram obejmujący funkcjonalność programu z podziałem na komponenty oraz z operacjami przeprowadzanymi w obrębie komponentu.



Ryc1.3.1 Diagram czynności

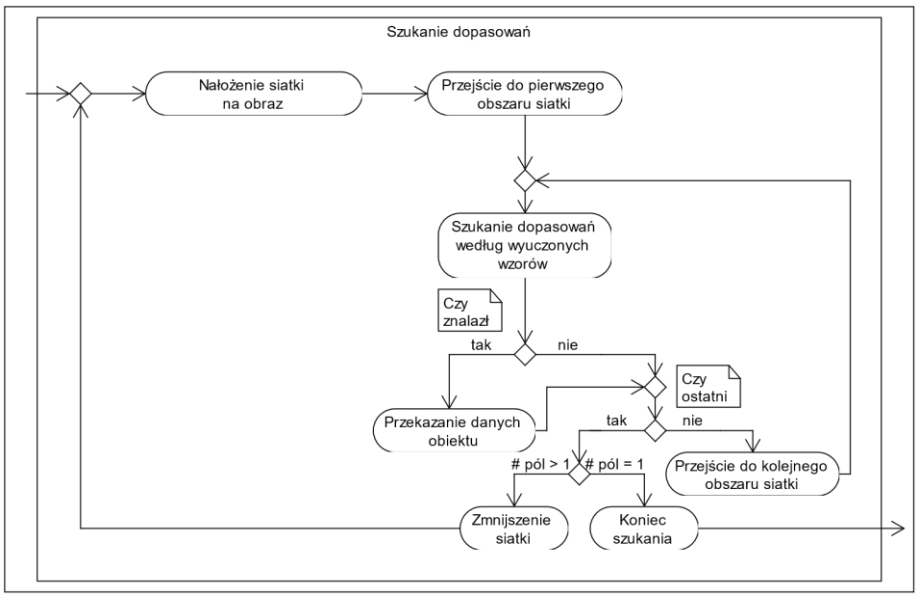
Dokładne działanie `Szukania dopasowań` z komponentu `Rozpoznawanie obiektów` zostało zaprezentowane na osobnym diagramie (Ryc1.3.2).

Z komentarzem [ML4]: Popraw Ryc -> fatalna jakość, zmiana nazw oraz lepszy opis, niezgodność z opisanymi komponentami (przestarzałe) / MEGA ROZPISAĆ, POŁĄCZYĆ Z PONIŻSZYM DIAGRAMEM. DOKŁADNY OPIS CZYNNOŚCI! POTEM DODAĆ OPISY KAŻDEGO KROK PO KROKU, OSOBA Z ZEWNĄTRZ MA WIDZIEĆ JAKIE SĄ MECHANIZMY, TO NIE JEST DLA NAS TYLKO OSOBY KTÓRA NIE ZOBACZY NASZEGO KODU, TAK ABY MOGLA SAMA SE TO NAPISAĆ

Z komentarzem [PK5R4]: Zrobione

Z komentarzem [PK6R4]:

Z komentarzem [ML7]:



Ryc1.3.2 Diagram szukania dopasowań

1.4. Biblioteki

Z komentarzem [ML8]: Dodać główne biblioteki wchodzące w skład programu, krótki opis. GŁÓWNA BIBLIOTEKA SIECI NEURONOWYCH I PROGRAMU YOLO, NA KTÓRYM BAZUJE NASZ PROGRAM. DODAĆ TE KTÓRE UŻYTKUJEMY RÓWNIEŻ W KOMPONENCIE WIDEO I ANALIZATOR.

2. Dokumentacja kodu

Z komentarzem [ML9]: Podstawowe metody działające w programie przynajmniej po 3 na komponent, na początek / ZOSTAWIC NA RAZIE, NIKOGO TO NIE INTERESUJE