AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Informatyki, Elektroniki i Telekomunikacji

Instytut Elektroniki

Projekt z przedmiotu Programowanie w Języku Python

Object follower

Autorzy: Michał Stankiewicz, Filip Kapłunow, Radosław Feiglewicz
Kierunek studiów: Elektronika, III rok
Opiekun pracy: dr inż. Maciej Wielgosz

Kraków, 2022

1. Opis algorytmu

Podstawą działania algorytmu jest analizowanie obrazu z kamery za pomocą sieci neuronowej, w architekturze AlexNet. Obraz jest dzielony w pionie na segmenty, które mogą na siebie nachodzić. Następnie segmenty są skalowane do rozmiaru odpowiedniego dla sieci i kolejno podawane do niej. Na podstawie jej odpowiedzi, wyrażonej w prawdopodobieństwie, podejmowana jest decyzja o tym, czy śledzony obiekt znajduje się w segmencie. Jeśli tak, podawanie segmentów jest przerywane, wokół segmentu rysowana jest obwódka i robot podejmuje decyzje o ruchu. Wykonywany jest skręt w prawo jeśli aktywował się segment po prawej, w lewo jeśli po lewej, lub jedzie do przodu jeśli aktywował się środkowy segment. Robot zatrzymuje się, jeśli obiektu nie wykryto.

2. Opis kodu

vision_module.py

W tym pliku znajduje się klasa *vision_mod* w której zaimplementowano określanie pozycji śledzonego obiektu. Składa się ona z następujących metod:

- *get_pos()* – pobiera obraz z kamery i zwraca numer segmentu w którym wykryto obiekt
- *get_pos_with_feed()* – pobiera obraz z kamery i zwraca numer segmentu w którym wykryto obiekt wraz z ramką z kamery z naniesioną obwódką
- get_draw_list(frame) – w niej przeprowadzana jest analiza ramki i wyliczany numer segmentu z obiektem
- *draw_bounding_boxes(frame, draw_list, sub_coords)* – rysuje obwódkę wokół wybranego segmentu
- *create_segments(frame, dims, overlap)* – dzieli obraz z kamery na segmenty o zadanych wymiarach i zadanym nakładaniu się
- *preprocess(camera_value)* – przeprowadza wstępną obróbkę ramki, zgodnie z wymaganiami sieci neuronowej
- *__init__(network_path, out_features_num, main_feature)* – konstruktor, tworzy wszystkie niezbędne obiekty, a także pobiera parametry sieci neuronowej do wczytania
- *deinit()* – zatrzymuje pobieranie danych z kamery

network_train_laptop.py

W tym pliku znajduje się kod niezbędny do przeszkolenia sieci neuronowej. Na początku importowane są dane, na podstawie których szkolona będzie sieć, a także określane parametry obróbki obrazów zgodnie z wymaganiami sieci. Dalej dane dzielone są na treningowe i testowe, po czym przekazuje się je do obiektów zarządzających ładowaniem danych do treningu i testowania. Następnie definiowane są parametry szkolonej sieci (w tym liczba wyjść), po czym sieć ładowana jest do GPU. Na końcu w cyklach sieć jest szkolona i optymalizowana do wykrywania obiektu, a także testowana pod tym kątem. Gotowa sieć zapisywana jest do pliku o zadanej nazwie.

main.ipynb

Główny plik, w którym znajduje się kod odpowiadający za sterowanie robotem. Podczas inicjalizacji tworzone są obiekty odpowiedzialne za sterowaniem silnikami robota (zmienna robot) oraz za określanie pozycji śledzonego obiektu (zmienna mod). Oprócz tego tworzony jest podgląd na żywo z kamery i wypisanie wykrytego segmentu obrazu. W pętli głównej program sprawdza czy obiekt, na którym została wyszkolona sieć (tutaj plik *network_ball_model.pth*), został wykryty. W zależności od tego, w którym miejscu obiekt został rozpoznany, obraz jest odpowiednio modyfikowany, a silniki wysterowywane, tak aby robot jechał w pożądanym kierunku (funckja *robot_update(draw_list)*). Program można wyłączyć wywołując funkcję *mod.deinit()*.