Projekt

Analiza obrazu w symulatorze Carla

1. Ogólny opis realizacji i wyników

   Podczas projektu zostało zrealizowane napisane 3 skrypty w języku Python:

   - yolov1.py
   - yolo_odleglosc.py
   - yolo_sterowanie.py

   Pierwszy z nich tworzy samochód w środowisku Carla sterowany autopilotem wraz z kamerą przyczepioną do przodu samochodu, a następnie przetwarza obraz poprzez zastosowanie algorytmu „yolo” z użyciem zdefiniowanych przez użytkowania parametrów sieci.

   Następny skrypt służy do weryfikacji algorytmu wyznaczania odległości. Tworzy on dwa samochody których środki są oddalone od siebie o 10 metrów. Algorytm zwraca wartość 6,37m jest ona poprawna ponieważ jest to odległość pomiędzy przodem a tyłem samochodu.

   Ostatnim skrypt odpowiada za próbę podążania za samochodem sterowanym autopilotem. Algorytm opiera się na wyznaczeniu odległości od wykrytego samochodu i dostosowaniu w zależności od niej prędkości. Natomiast sterowanie skręcaniem polega na ustawieniu środku ramki wykrytego samochodu w środku obrazu kamery. W przypadku braku wykrycia samochodu sterowanie przedstawionym algorytm zamienia się na sterowanie autopilotem. Niestety w tym przypadku algorytm wykonuje się za wolno i samochód albo się nie porusza albo porusza się na autopilocie. Poniższe zdjęcie pokazuje ile razy wywołuje się funkcja w

przypadku wywołania skryptu yolov1.py(917 razy) i yolo_sterowanie.py(47 razy). Może to być problem z wielowątkowością Pythona.

```
F:\WindowsNoEditor\PythonAPI\examples>python yolo_sterowanie.py
ActorBlueprint(id=vehicle.tesla.model3,tags=[vehicle, tesla, model3])
Transform(Location(x=66.961761, y=-4.278575, z=0.275307), Rotation(pitch=0.000000, yaw=-179.144165, roll=0.000000))
destroying actors
Function process_img has been used  47  times.

F:\WindowsNoEditor\PythonAPI\examples>python yolov1.py^S
python: can't open file 'yolov1.py‼': [Errno 22] Invalid argument

F:\WindowsNoEditor\PythonAPI\examples>python yolov1.py
ActorBlueprint(id=vehicle.tesla.model3,tags=[vehicle, tesla, model3])
Transform(Location(x=56.961758, y=-4.278575, z=0.275307), Rotation(pitch=0.000000, yaw=-179.144165, roll=0.000000))
Location(x=56.961758, y=-4.278575, z=0.001624)
destroying actors
Function process_img has been used  917  times.
```

Warto wspomnieć o próbie stworzenia inteligentnego agenta którego zadaniem miało być nauczenie się poruszania się w symulacji. Jego zadaniem było wykorzystanie wyników sieci yolo do sterowania prędkością. Jednak ze względu na nierozwiązany problem z bibliotekami nie udało się zrealizować tego zadania, efektem tego jest skrypt reinforcment_projekt.py.

2. Szczegółowa realizacja

   Aby wszystkie skrypty działały muszą one się znajdować w folderze PythonAPI\examples. Aby je uruchomić trzeba wcześniej uruchomić Carlę, a następnie uruchomić konsolę w folderze examples i wywołać je w konsoli np. python yolov1.py. W skryptach są komentarze ułatwiające zrozumienie działania programu.

   Skrypt yolov1.py:

   Na początku skryptu znajdują się biblioteki i zmienne globalne wykorzystywane w reszcie programu. Później są zdefiniowane dwie funkcje process_img i process_img2. Służą one do przetwarzania obrazu druga z nich wyświetla obraz z kamery a druga wyświetla obraz po przepuszczeniu przez sieć yolo. Później jest właściwa część programu gdzie skrypt łaczy się z serwerem Carli i stworzenie samochodu i kamery. Następnie jest uruchamiany autopilot i samochód zaczyna się poruszać. Bardzo ważna jest instrukcja: *sensor.listen(lambda data: process_img(data))*, ponieważ pozwala ona na wywołanie funkcji proces_img za każdym razem kiedy odbiera sygnał od sensora. Właśnie w dzięki tej instrukcji możliwe jest pozyskanie informacji z obrazu. Na końcu programu wszystkie elementy stworzone są usuwane i połączenie z serwerem kończy się.

   Skrypt yolo_odleglosc.py:

   W tym skrypcie zostały przeprowadzone bardzo podobne procesy co we wcześniejszym algorytmie z tym że została wyznaczona ogniskowa kamery i została wykorzystana wysokość ramki(otrzymanej dzięki yolo). Ogniskowa została wyznaczona przy pomocy następującego wzoru:

$$\frac{\frac{w}{2}}{f} = tg\frac{\alpha}{2}$$

   Gdzie α to HFOV(Horizontal Field of View), a *w* to szerokość obrazu w pikselach. Następnie przy pomocy wzoru:

$$d = rzeczywista\_wysokość\_obiektu * \frac{ogniskowa}{wysokość\_obiektu\_w\_pikselach}$$

W skrypcie jest podana rzeczywista wysokość pojazdu Tesla Model 3- 1,443m. We wzorze można używać zamiennie szerokości i wysokości. Niestety nie udało się znaleźć położenia punktu orientacyjnego dla modelu Tesli Model 3 w Carli. Dlatego nie da się zweryfikować dokładnie czy odległość 6,37 m jest poprawna. Szacunkowo da można stwierdzić iż wynik jest względnie poprawny. Zakładając, iż punkty lokalizacyjne są w połowie pojazdu, a pojazd ma długość 4,694m(źródłó) wówczas oszacowana rzeczywista odległość wynosi 10m-4,694m = 5,306m. Wówczas różnica pomiędzy odległością wyznaczoną przez program, a rzeczywistą oszacowaną wynosi około 1m. Należy jednak pamiętać, że nie mamy dokładnych informacji na temat modelu Tesli w Carli i wcale nie oznacza to błędnego wyniku.

Skrypt yolo_sterowanie.py:
Skrypt yolo_sterowanie.py jest rozwinięciem skryptu yolo_odleglosc.py o dodanie drzewa decyzyjnego sterującym pojazdem i włączenie autopilota w pojeździe znajdującym się przed kamerą. Jest to pojazd, za którym ma podążać pojazd, którym sterujemy. W tym celu funkcja proces_img został wzbogacona o dodatkowy argument, którym jest pojazd, którym skrypt ma sterować. Dodatkowo została zdefiniowana funkcja sterowanie, która liniowo steruje prędkością w zależności od odległości od wykrytego pojazdu. A także skręcaniem w zależności od jego położenia w osi poziomej względem środka obrazu wychwytywanego kamery. Jest ona właśnie wywoływana w funkcji process_img w momencie wykrycia samochodu. Jak było wcześniej już wspominane funkcja wykonuje się za mało razy aby móc stwierdzić jakość dobranego algorytmu sterowania. Bardzo możliwe, że problem jest z wielowątkowością Pythona lub trzeba przetestować to na sprzęcie o wyższej mocy obliczeniowej lub napisać to inaczej.

Skrypt reinforcment_projekt.py:
Jest to plik zupełnie inny, w którym wykorzystywany jest koncept reinforcment learningu. Jest to praca pochodząca ze strony: https://pythonprogramming.net/reinforcement-learning-self-driving-autonomous-cars-carla-python/ wzbogacona o funkcję process_img wraz z obliczanie odległości od najbliższego obiektu. Zostaje to wykorzystane w procesie przyznawania nagrody.

3. Podsumowanie i możliwość dalszego rozwoju
   Wyniki z pomiarami odległości i zastosowaniem sieci yolo można uznać za satysfakcjonujące. Natomiast porażką okazały się próby stworzenia agentów poruszających się w świecie Carli jak i prostego sterowania w celu podążania za samochodem.
   Projekt można rozwinąć poprzez zastosowanie sieci yolo w pliku domyślnym Carli manual_control.py. Ponadto można spróbować zastosować plik reinforcment_projekt.py na komputerze z dobrej jakości podzespołami i sprawnie działającymi bibliotekami.