System rozpoznawania pachołków w chmurze punktów za pomocą LIDARa

Adam Paleczny i Michał Stasiak

Założenia projektu

- Stworzenie systemu rozpoznającego pachołki w chmurze punktów
- Etapy projektu:
  - Usunięcie punktów stanowiących podłoże
  - Klasteryzacja euklidesowa i otrzymanie potencjalnych chmur punktów
  - Wytrenowania Modelu PointNet z datasetu MedelNet40 z pachołkami drogowymi
  - Implementacji modelu dla danych z LIDARa

Wykorzystane urządzenia

LIDAR Ouster os1 128

Nvidia Jetson

Wykorzystane technologie

- ROS - Robot Operating System
- PCL Point Cloud Library C++
- Tensorflow

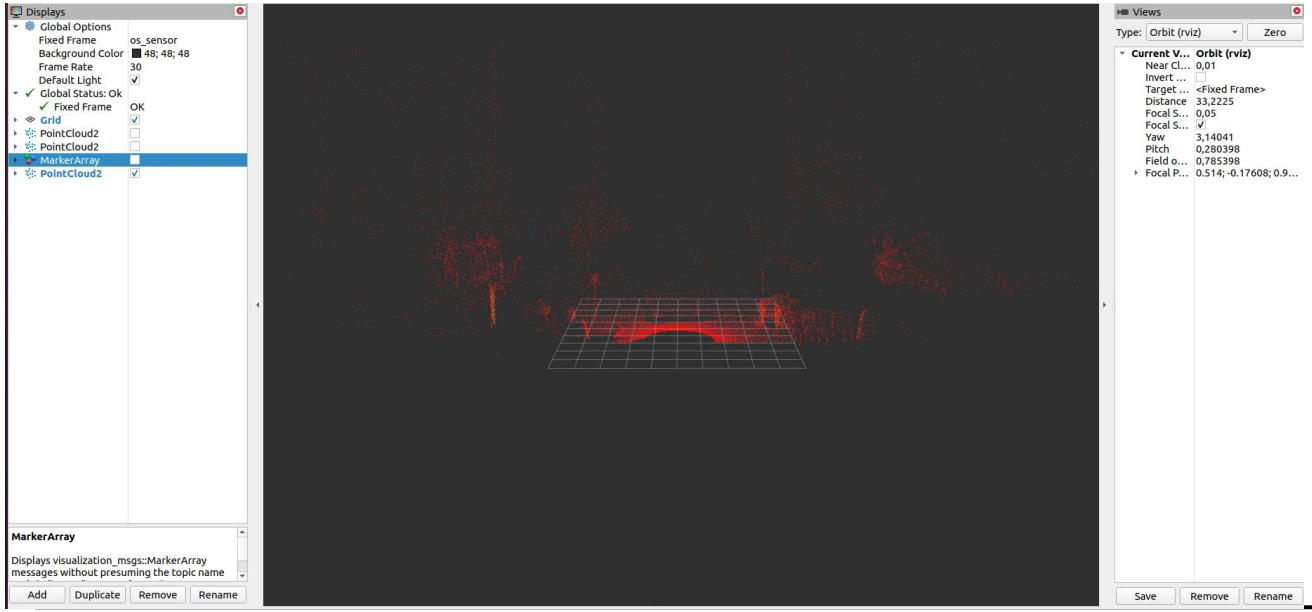
Dataset - ModelNet40

ModelNet40 - dataset zawierający 40 różnych obiektów chmur punktów.

Na potrzeby projekty wykorzystaliśmy tylko jedną klasę “cone”. Zawiera ona

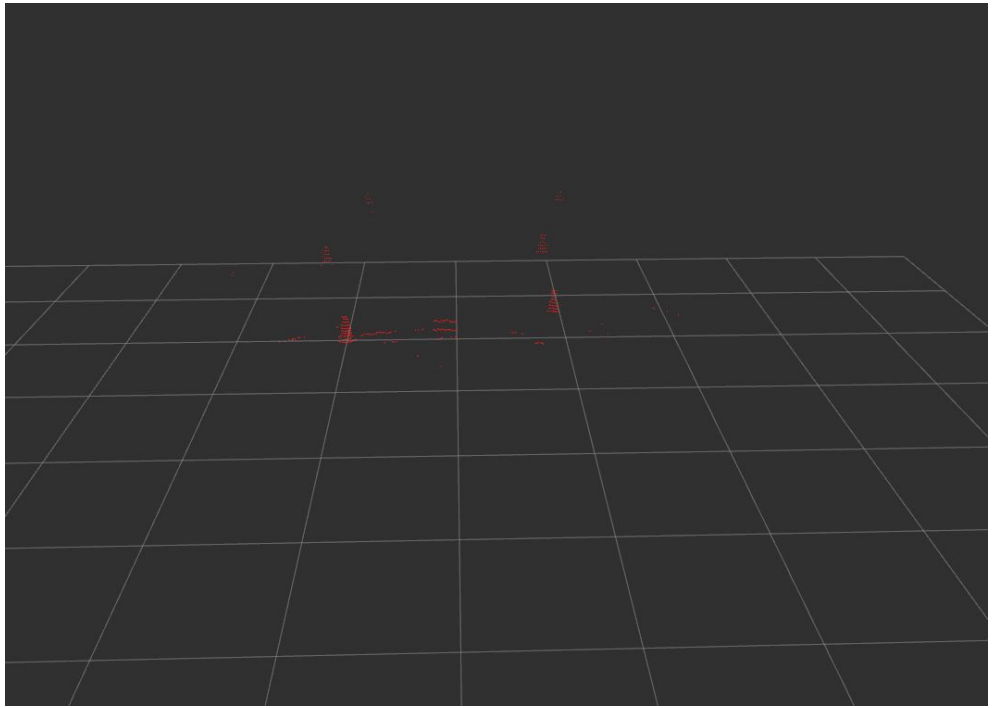
Chmura punktów

Około 270 tyś punktów

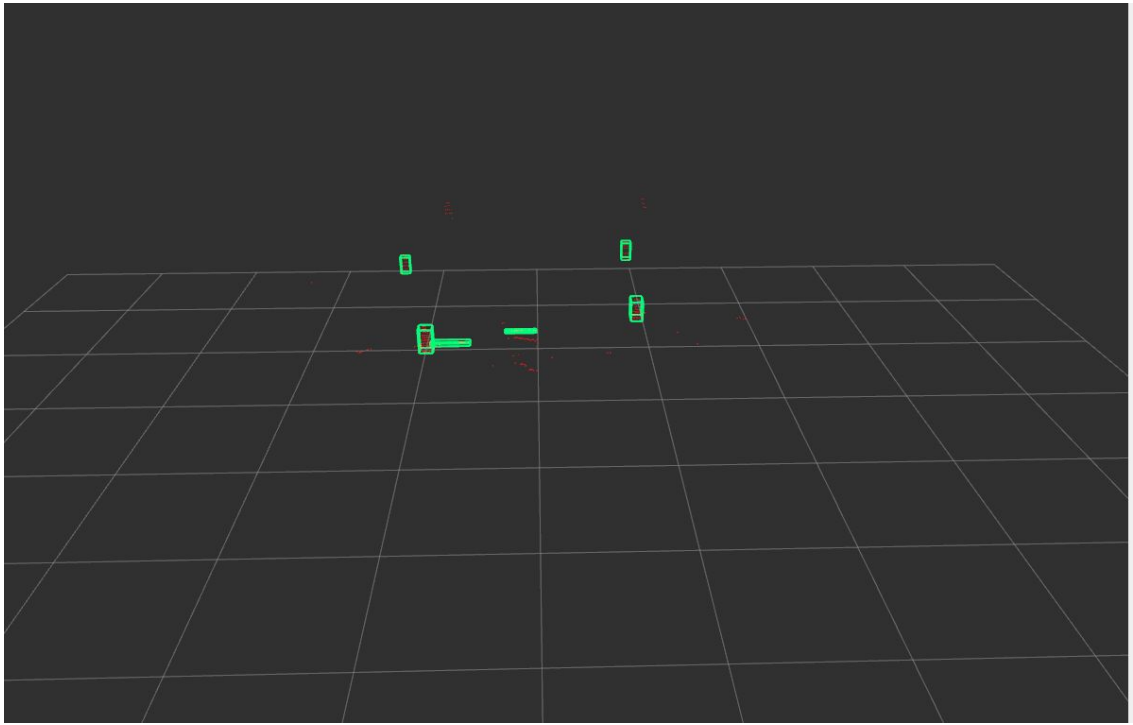


Usuwanie ziemi - Patchwork++

Redukcja chmury punktów



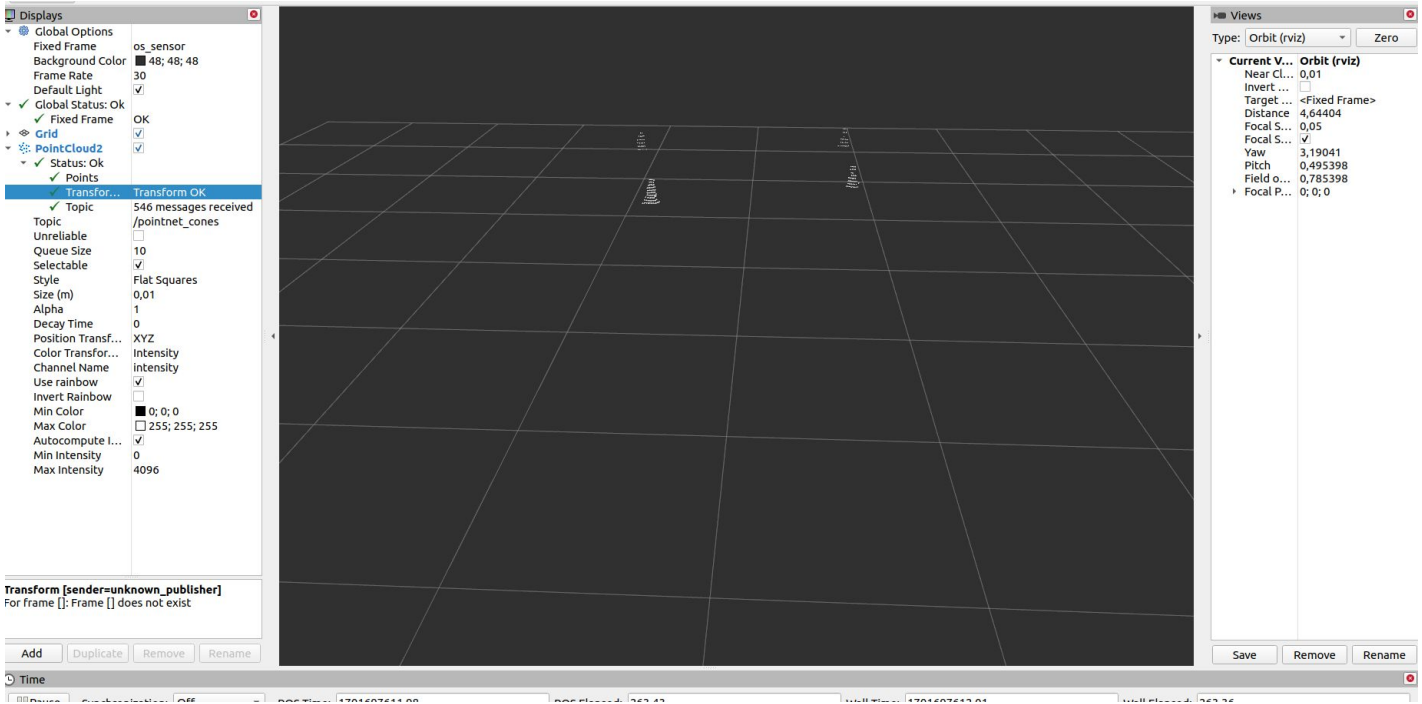
Klasteryzacja euklidesowa



Trenowanie Modelu

```
Epoch 12/15
522/522 [==============================] - 11s 22ms/step - loss: 1.5263 - accuracy: 0.7752 - val_loss: 43.3635 - va
l_accuracy: 0.7750
Epoch 13/15
522/522 [==============================] - 11s 21ms/step - loss: 1.5039 - accuracy: 0.7957 - val_loss: 21857.2188 -
val_accuracy: 0.8500
Epoch 14/15
522/522 [==============================] - 11s 20ms/step - loss: 1.4760 - accuracy: 0.8102 - val_loss: 11.0448 - va
l_accuracy: 0.8500
Epoch 15/15
522/522 [==============================] - 11s 20ms/step - loss: 1.5692 - accuracy: 0.8096 - val_loss: 140104.6562
- val_accuracy: 0.7250
<keras.callbacks.History at 0x7fb242c65150>
```

Osiągnięte rezultaty w chmurze punktów



Trenowanie nowego modelu z mniejszą ilością punktów potrzebnych do detekcji

- Zmniejszenie punktów wejściowych do 17
- Zmniejszenie dokładności kosztem dalszych detekcji

```
Epoch 13/15
1044/1044 [==============================] - 30s 29ms/step - loss: 1.0152 - accuracy: 0.7617 - val_loss: 3628.3979 - val_accuracy: 0.7000
Epoch 14/15
1044/1044 [==============================] - 31s 29ms/step - loss: 1.0107 - accuracy: 0.7723 - val_loss: 1.1127 - val_accuracy: 0.5750
Epoch 15/15
1044/1044 [==============================] - 30s 29ms/step - loss: 0.9984 - accuracy: 0.7842 - val_loss: 1.1828 - val_accuracy: 0.7250
<keras.src.callbacks.History at 0x7fbcf8ca5510>
```

Co dalej?

- Test z nowym modelem
- Code refactor
- Implementacja całego systemu na Dockerze przy użyciu Docker Compose?