PZP2

Celem projektu jest opracowanie algorytmu detekcji obiektów pozwalającego wykrywać przeszkody takie jak drzewa, krzaki, czy też słupy elektryczne znajdujące się na polach rolniczych na podstawie informacji pozyskanych z sensorów głębi. Wynikiem projektu będzie algorytm umożliwiający automatyczną detekcję przeszkód w chmurze punktów pozyskanych z lidaru lub kamer stereowizyjnych.

W ramach projektu zostanie wykorzystana baza RELLIS-3D. Jest to baza wykorzystywana do zaawansowanych automatycznych nawigacji terenowych. RELLIS-3D bazuje na informacjach z LiDARu oraz kamer RGB, obrazów stereowizyjnych, IMU oraz z GPS. LiDAR zapewnia bardzo dokładne informacje 3D, a kamery zapewniają kolor i informację do uzyskania dokładnych informacji o semantyce.

RELLIS-3D udostępnia dane w postaci 5 sekwencji zsynchronizowanych danych przechwyconych z jazdy terenowej w postaci ROS bag format. Dane te zawierają obrazy RGB, chmurę punktów z LiDARu, obrazy stereo, pomiary GPS oraz IMU (Inercyjna jednostka pomiarowa).

Name	Sensors	# Annotations	# Classs	Annotation Type	Modality
RUGD [14]	camera	7546	24	pixel-wise	RGB
DeepScene [15]	camera	366	6	pixel-wise	RGB, Depth, NIR, NRG, NDVI, EVI
Pezzementi et al [16]	camera	95000	1	bounding box	RGB
YCOR [1]	camera	1076	8	pixel-wise	RGB
Dabbiru et al [17]	simLiDAR	2743	6	point-wise	Point Cloud
Ours	camera, LiDAR	6235/13556	20	pixel/point-wise	RGB, Point Cloud

PCL biblioteka chmur punktów- biblioteka zawierająca algorytmy do zadań przetwarzania Point Cloud (chmur punktów) oraz do przetwarzania geometrii 3D.

Biblioteka zawiera algorytmy filtrowania, szacowania cech, rekonstrukcji powierzchni, rejestracji 3D, dopasowywania modelu, rozpoznawania obiektów i segmentacji.

Algorytmy te są używane do percepcji w robotyce do filtrowania wartości odstających z zaszumionych danych, łączenia chmur punktów 3D, segmentowania odpowiednich części sceny, wyodrębniania kluczowych punktów i obliczania deskryptorów w celu rozpoznawania obiektów w świecie na podstawie ich geometrycznego wyglądu, do tworzenia powierzchni z chmur punktów i ich wizualizacji.

Biblioteka ta jest w pełni zintegrowana z ROS (Robot Operating System).

ROSbag – narzędzie ROS do przechwytywania i odtwarzania danych ROS. Rosbag używa formatu pliku o nazwie bags, który rejestruje wiadomości ROS poprzez słuchanie tematów i nagrywanie wiadomości.

Dane:

Dane pobierane są z Datasetu RELLIS-3D w formie pliku ROS Bag.

Nasz algorytm ma na celu wykrycie przeszkód na drodze na podstawie danych z Lidaru, dlatego wykorzystujemy zbiór danych zawierających tylko poniższe 3 informację:

```
path:        example_synced.bag
version:     2.0
duration:    59.9s
start:       Feb 15 2020 10:34:38.41 (1581791678.41)
end:         Feb 15 2020 10:35:38.33 (1581791738.33)
size:        4.8 GB
messages:    1800
compression: none [1200/1200 chunks]
types:       sensor_msgs/CameraInfo  [c9a58c1b0b154e0e6da7578cb991d214]
             sensor_msgs/Image       [060021388200f6f0f447d0fcd9c64743]
             sensor_msgs/PointCloud2 [1158d486dd51d683ce2f1be655c3c181]
topics:      /os1_cloud_node/points           600 msgs    : sensor_msgs/PointClo
ud2
             /pylon_camera_node/camera_info   600 msgs    : sensor_msgs/CameraIn
fo
             /pylon_camera_node/image_raw     600 msgs    : sensor_msgs/Image

real    0m1.599s
user    0m0.244s
sys     0m0.500s
```

W projekcie będą wykorzystywane tylko 2 informację : dane chmury punktów z Lidaru oraz przechwycony obraz.

Początkowo dane miały być przetwarzane za pomocą biblioteki bagpy. Narzędzie bagreader z biblioteki bagpy służy do dekodowania i odczytu plików rosbag. Niestety obliczenia te były bardzo kosztowne czasowo oraz brakowało narzędzi do edytowania danych z plików .bag

W aktualnej wersji do odczytu danych wykorzystujemy system ROS. Umożliwia on za pomocą udostępnianych bibliotek prosty oraz szybki sposób odczytywania danych z pliku .bag

Wykorzystanie systemu ROS łączy się jednak z ograniczeniem działania programu do systemu Linux z zainstalowanym systemem ROS.

Pozyskiwanie danych:

Dane pozyskiwane są z przykładowego pliku example_synced.bag.

Program korzysta z systemu ROS, dlatego należy uruchomić roscore, czyli zbiór węzłów i programów, które są niezbędne dla systemu opartego na ROS.

Program do pozyskiwania danych napisany jest w Pythonie. Polega on na uruchamianiu procesów ROS pobierających dane. Jako argumenty przyjmuje rosbag oraz topics, czyli tematy danych do odczytu.

Główną funkcją *to_pcd()*

Funkcja ta polega na odczycie wiadomości sensor_msgs/PointCloud2 z pliku .bag do pliku pcd (Point Cloud Data). Jest to format wspierający chmurę punktów 3D.

```
@ubuntu:~$ rosrun pcl_ros bag_to_pcd example_synced.bag /os1_cloud_node/points  .
/pointclouds
Creating directory ./pointclouds
Saving recorded sensor_msgs::PointCloud2 messages on topic /os1_cloud_node/points to .
/pointclouds
Got 131072 data points in frame ouster1/os1_lidar on topic /os1_cloud_node/points with
 the following fields: x y z intensity t reflectivity ring noise range
Data saved to ./pointclouds/1581791678.433744128.pcd
```

Narzędzie bag_to_pcd służy do odczytu wszystkich wiadomości z pliku .bag z zadanym topic (tematem wiadomości) do folderu w postaci pliku .pcd

Wszystkie odczytane wiadomości znajdują się w folderze ./pointclouds

Uzyskanie plików .jpg - *get_images()*

Do odczytania zdjęć aktualnego stanu drogi zostanie wykorzystany plik launch. Plik launch jest powszechnym narzędziem ROS. Zapewnia wygodny sposób uruchamiania wielu węzłów oraz używane są do inicjalizacji wymagań np. ustawienia parametrów.

```
<launch>
<arg name="bagfile"/>
<arg name="topic"/>
<node pkg="rosbag" type="play" name="rosbag" required="true" args="$(arg bagfile)"/>
<node name="extract" pkg="image_view" type="extract_images" respawn="false"
required="true" output="screen" cwd="ROS_HOME">
<remap from="image" to="$(arg topic)"/>
</node>
</launch>
```

Plik export_img.launch uruchamia 2 węzły – odczytywanie danych z pliku .bag oraz wydobycie danych sensor_msgs/Image z pliku .bag o zadanym topicu do pliku .jpg

Do uruchomienia wykorzystujemy narzędzie roslaunch (narzędzie umożliwiające uruchomienie wielu węzłów ROS). Jako argumenty podajemy plik export_img.launch, ścieżkę do pliku .bag oraz topic, czyli wiadomości sensor_msgs/Image które chcemy zamienić na pliki .jpg

Domyślnie pliki zapisywane są do folderu .ros, dlatego należy przenieść wszystkie pliki .jpg do naszego folderu ./images, z którego dane będą dalej pobierane.