**Program do generacji trójwymiarowych modeli na podstawie danych głębi- Instrukcja programisty**

W programie można wyróżnić cztery główne klasy o nazwach GUI, LinearScanner, StructeredLight oraz MyDelaunay zawarte odpowiednio w plikach gui.py, linearScan.py, structuredLight.py, MyDelaunay.pyx. Poniżej przedstawiono omówienie poszczególnych klas wraz z ich modułami oraz funkcjami.

**Klasa GUI:**

Klasa GUI odpowiada za interfejs graficzny programu. To dzięki tej warstwie użytkownik jest w stanie generować trójwymiarowe modele na podstawie danych z kamery głębi. Interfejs utworzono za pomocą biblioteki PyQt5, która posiada wiele elementów kluczowych do poprawnej obsługi programu. W skład powyższej klasy wchodzą dwie kolejne, CreateFromPCL oraz CreateFromLinearScanner. Obie stanowią osobne funkcje programu. Pierwsza zajmuje się obróbką danych pochodzących z metody światła strukturalnego. Druga zaś akwizycją danych z nagrania kamery głębi oraz ich konwersją na chmury punktów. Obie klasy zawierają przekształcenia umożliwiające utworzenie siatki na chmurze punktów przy wykorzystaniu algorytmu BPA oraz triangulacji Delaunay’a.

**Klasa LinearScanner**

Klasa LinearScanner odpowiada za przetworzenie danych pochodzących z nagrania kamery głębi do postaci chmury punktów. Następnie wykonuje algorytm triangulacji Delaunay’a oraz BPA w celu nałożenia na nie siatki. Poniżej opisane zostały funkcje występujące w danej klasie:

* updateVals- Funkcja służąca do komunikacji obiektu z interfejsem graficznym. Poprzez użycie tej funkcji możliwa jest aktualizacja zmiennych wewnętrznych w klasie takich jak ścieżka do nagrania, współrzędne wybranych przez użytkownika punktów oraz wiele innych.
* createFromVideo – Funkcja będąca kluczowym elementem klasy. Zawiera ona szereg czynności, prowadzących do generacji chmury punktów oraz siatek triangulacyjnych. Funkcje w niej występujące służą do załadowania kolumn głębi oraz koloru z nagrania, przetworzenia danych chmurę punktów, wyświetlenie rezultatów użytkownikowi. Funkcja wykorzystuje zewnętrzny moduł Read3D.
* calcHeightFromPixel – Funkcja służy do wyznaczania rzeczywistej wysokości obiektu na podstawie jego wysokości w pikselach oraz odległości od kamery. Zasada działania funkcji została dokładniej opisana w pracy.
* meanPointDistance – Funkcja służy do wyznaczania średniej odległości punktów od środka układu współrzędnych. Jest to pomocne przy filtracji przekłamanych punktów w skutek błędnych odczytów z kamery głębi.
* filterAndLinearize – Funkcja odpowiada za oflagowanie przekłamanych punktów, a następnie ich linearyzację w celu otrzymania poprawnych wartości. Korzysta ona przy tym z zewnętrznego modułu LinearizePoints.
* clearWrongPointsAndVectorize – Do zadań funkcji należy usunięcie błędnych punktów nieusuniętych podczas linearyzacji oraz przekształcenie macierzy punktów. Podane na wejściu zbioru punktów są wielowymiarowe, jednak w celu ułatwienia operacji należy je przekształcić do postaci macierzy jednowymiarowej.
* Create3DPoints – Główna funkcja klasy zawierająca odpowiednie przekształcenia mające na celu konwersję dwuwymiarowych punktów w ich trójwymiarowe odpowiedniki. Poprzez zastosowanie wzorów trygonometrycznych oraz danych o głębi, na wyjściu funkcji otrzymywane są trójwymiarowe punkty wraz z kolorem zapisanym w osobnej macierzy.
* createBpa – Funkcja służy do generacji siatki przy wykorzystaniu algorytmu BPA. W celu realizacji zadania korzysta z biblioteki Open3D do wyznaczenia siatki oraz obliczenia normalnych do powierzchni chmury punktów. Na wyjściu otrzymywany jest gotowy rezultat w postaci siatki triangulacyjnej.
* CreateDelaunay – Funkcja przetwarza punkty w sposób możliwy do obsłużenia przez zewnętrzną klasę MyDelaunay. Po otrzymaniu wyników triangulacji przetwarza je i podaje na wyjście gotowy obiekt trójwymiarowy.
* createPcd – Zadaniem funkcji jest zamiana macierzy punktów oraz pikseli do chmury punktów obsługiwanej przez program Open3D.
* showPcd – Funkcja odpowiada za wyświetlanie chmury punktów oraz siatki triangulacyjnej na ekranie użytkownika.

Klasa wykorzystuje również dodatkowe moduły takie jak Read3D oraz linearizePoints, posiadające funkcje potrzebne do odpowiedniego funkcjonowania programu.

Moduł Read3D służy do obsługi nagrań z kamery trójwymiarowej. Zawiera funkcje takie jak:

* loadColorDepth – Funkcja służy do uruchomienia nagrania z kamery głębi wskazanego przez użytkownika. Obsługuje kolejkę, służącą do buforowania wejściowych klatek nagrania, w celu ich dalszej obróbki. Wyświetla również aktualny postęp ładowania na interfejsie graficznym.
* slowProc – Ze względu na złożoność obliczeniową użyto kolejki do przetwarzania danych z jednej klatki nagrania kamery głębi. Funkcja nakłada na siebie współrzędne głębi oraz koloru dla poszczególnych klatek nagrania. Wykorzystując biblioteki deweloperskie firmy Intel o nazwie PyRealSense2 aplikuje filtry mające za zadanie zmniejszyć ilość luk w nagraniu. Po ułożeniu zdjęć, funkcja zbiera dane o jednej kolumnie obrazu oraz przekazuje je z powrotem do procesu nadrzędnego jakim jest funkcja loadColorDepth. Po przetworzeniu danych następuje wyświetlenie aktualnego ujęcia głębi oraz koloru na ekranie użytkownika.

Moduł linearizePoints służy do linearyzacji przekłamanych punktów. Zawiera funkcje takie jak:

* myCubic – Funkcja odpowiadająca za odpowiednie przekształcenie współrzędnych punktów, tak by można je było zlinearyzować. Wykorzystując funkcję z biblioteki scipy dokonuje aproksymacji dostępnych wartości wielomianem trzeciego stopnia. Po wykonaniu odpowiednich operacji wyznacza ostateczne wartości punktów oraz zwraca je na wyjściu.
* Func- Zwraca wartość wielomianu trzeciego stopnia w zależności od współczynników oraz wartości X.
* createLinearizedValues – Funkcja służy do utworzenia aproksymowanych wartości punktów na podstawie wielomianu trzeciego stopnia przechodzącego przez wszystkie dostępne punkty.

**Klasa CreateFromPCL**

Klasa CreateFromPCL odpowiada za przetworzenie danych pochodzących z utworzonych przez użytkownika chmur punktów. Wykorzystując algorytm RANSAC łączy ona ze sobą sąsiednie chmury w spójną całość. Następnie wykonuje algorytm triangulacji Delaunay’a oraz BPA w celu nałożenia na nie siatki. Poniżej opisane zostały funkcje występujące w danej klasie:

* createAllFromPly – Główna funkcja klasy, uruchamiająca proces wyrównania i dopasowania chmur punktów. Uruchamia wszystkie mniejsze funkcje takie jak ładowanie chmur punktów do pamięci programu, odpowiednim ich obrotem, zespoleniem sąsiadujących chmur ze sobą oraz wyświetleniem wyników użytkownikowi.
* loadPlys – Funkcja zajmuje się ładowaniem utworzonych przez użytkownika chmur punktów do pamięci programu. Po wpisaniu początkowej nazwy pliku wyszukuje wszystkie odpowiadające kryteriom pliki. Głównym kryterium jest akwizycja klatek z odstępem kątowym zadanym przez użytkownika. Funkcja konwertuje odstęp kątowy do liczby klatek odstępu.
* clearAllPcds – Funkcja odpowiada za filtrację punktów. Dla każdej z załadowanych chmur sprawdzana jest odległość punktów od kamery. Jeśli mieści się ona w zakresie podanym przez użytkownika, to punkty zostają podane na wyjście funkcji. Poprzez zastosowanie danego algorytmu możliwe jest dokładne usunięcie tła oraz pozostałości po obracanej tacce.
* translatePcds – Po uruchomieniu danej funkcji chmury zostają przesunięte do początku układu współrzędnych. Dzięki temu można je obrócić we współrzędnych globalnego układu, a nie we współrzędnych kamery.
* rotatePcds – Funkcja wyznacza macierz obrotu dla poszczególnych chmur punktów. Kąt zależy od indeksu danej klatki. Mając macierz obrotu, dokonuje obrotu wszystkich punktów w chmurze wokół początku układu współrzędnych.
* createAngleList – Funkcja pomocnicza wyznaczająca przesunięcie kątowe pomiędzy kolejnymi klatkami. Poprzez podział pełnego obrotu przez całkowitą liczbę klatek, uzyskiwany jest kąt obrotu dla poszczególnej chmury punktów.
* createGlobalRegistrationForClouds – Funkcja przeprowadzająca rejestrację oraz sklejanie wszystkich sąsiednich chmur punktów. Iteruje poprzez wszystkie pary chmur, aż do momentu ich wyczerpania.
* globalSourceTarget – Główna funkcja zajmująca się odpowiednim ułożeniem względem siebie chmur punktów. Rejestracja oraz sklejanie przeprowadzane są poprzez zastosowanie kilku mniejszych funkcji. Na początku dokonuje decymacji punktów w chmurze oraz wyznacza ich histogram cech. Złączenie chmur odbywa się przy wykorzystaniu funkcji z biblioteki Open3D przeprowadzającej algorytm RANSAC. Poprzez zastosowanie danego algorytmu, możliwe jest uzyskanie macierzy transformacji pozwalającej na gładkie połączenie chmury źródłowej oraz docelowej. Przesunięte w ten sposób chmury podawane są na wyjście.
* flatenMultipleClouds – Funkcja pomocniczą konwertująca wiele chmur w macierzy do postaci pojedynczej chmury. Jest bardzo pomocna przy wyświetlaniu chmur oraz ich zapisywaniu.
* showPcd – Funkcja odpowiada za wyświetlanie chmury punktów oraz siatki triangulacyjnej na ekranie użytkownika.
* createDelaunay – W danej funkcji przeprowadzana jest autorska triangulacja Delaunay’a w celu utworzenia siatki na chmurze punktów. Przygotowany w ten sposób model podawany jest na wyjście funkcji.
* createBpa – Funkcja generuje siatkę triangulacyjną za pomocą metody BPA. Oblicza normalne do powierzchni chmury oraz promień kuli. Wszystkie te parametry są wymagane do poprawnego funkcjonowania algorytmu z biblioteki Open3D.
* createAndShowResults – Funkcja odpowiada za wyświetlenie użytkownikowi wybranych przez niego obiektów. Odpowiada za generację oraz prezentację chmury punktów oraz siatek triangulacyjnych.

**Klasa MyDelaunay**

Klasa MyDelaunay odpowiada za przeprowadzenie trójwymiarowej triangulacji Delaunay’a na zadanym zbiorze punktów. W celu optymalizacji powyższa klasa została przekształcona do języka cython. Poniżej opisane zostały funkcje występujące w danej klasie:

* removeSharedFace – Funkcja odpowiada za odnalezienie współdzielonych ścian pomiędzy ostrosłupem aktualnym, a wszystkimi pozostałymi. Wszystkie współdzielone ściany zostają następnie usunięte.
* removeTouchingTetra – Funkcja odpowiada za podział zbyt dużych ostrosłupów na mniejsze. Odbywa się to poprzez utworzenie nowych ostrosłupów zawierających trzy wierzchołki poprzedniego oraz dodatkowy wierzchołek będący punktem poddanym triangulacji.
* createSuperTetra – Funkcja odpowiada za utworzenie super ostrosłupa. Jest to ostrosłup początkowy zawierający wszystkie punkty triangulacyjne wewnątrz siebie.
* liesOnSuper – Funkcja potrzebna do poprawnego usunięcia ostrosłupów mających wspólne wierzchołki z super ostrosłupem. Poprzez wykorzystanie iloczynu zbioru, jeśli liczba wierzchołków w wynikowym zbiorze jest większa od 2, to ostrosłupy mają współdzieloną ścianę i należy je usunąć.
* computeTrianglePoints – Główna funkcja odpowiadająca za przeprowadzenie trójwymiarowej triangulacji Delaunay’a. Zasada działania algorytmu została dokładnie przedstawiona w pracy. Dla każdego punktu w zbiorze, wyznaczana jest jego przynależność do sfery opisanej na ostrosłupie. Jeżeli punkt leży wewnątrz ostrosłupa, to dany ostrosłup należy podzielić wykorzystując funkcję removeTouchingTetra. Na koniec wszystkie ostrosłupy mające wspólne wierzchołki z początkowym super ostrosłupem są usuwane.
* calcDistance – Funkcja odpowiada za wyznaczenie odległości pomiędzy dwoma punktami.
* compareTetraFaces – Funkcja zajmująca się wyznaczaniem wierzchołków wspólnych w obu ostrosłupach. Wykorzystuje przy tym zbiory, ponieważ kolejność numeracji wierzchołków może być różna.
* transformToIndexes – Funkcja zamienia współrzędne wierzchołków ostrosłupa do postaci ich indeksów w początkowym zbiorze. Jest to pomocne przy późniejszym wyświetlaniu utworzonych siatek triangulacyjnych.
* computeVertexCoordsFacesColors – Funkcja zajmuje się wyświetlaniem utworzonych siatek triangulacyjnych. Jest to funkcja służąca do sprawdzania działania programu i nie jest używana wewnątrz klasy.
* computeVerticies – Funkcja startowa, uruchamiająca wyznaczanie siatki triangulacyjnej, a następnie konwersję gotowych wierzchołków do postaci akceptowalnej przez zewnętrzną bibliotekę.

Klasa wykorzystuje również dodatkowy moduł o nazwie Tetrahedron. Jest to klasa opisująca ostrosłup, zawierająca pomocne funkcje wykorzystywane podczas przeprowadzania triangulacji. Ze względu na optymalizację, ten moduł również został napisany w języku cython. Funkcjami występującymi w powyższej klasie są:

* calculateSphereCenter – Funkcja uruchamiana przy inicjalizacji ostrosłupa. Przekształca otrzymane współrzędne środka sfery opisanej na czworościanie oraz jej promień. Gotowe współrzędne podawane są na wyjście funkcji.
* calcCenter – Główna funkcja zajmująca się wyznaczaniem środka sfery opisanej na ostrosłupie. Poprzez zastosowanie odpowiednich wzorów oraz obliczenie wyznaczników uzyskiwany jest środek sfery. Na koniec jej promień jest liczony jako odległość środka od dowolnego wierzchołka ostrosłupa.
* calcDet – Funkcja pomocnicza pomocna przy obliczaniu wyznacznika macierzy potrzebnego w funkcji calcCenter. Zajmuje się ona utworzeniem odpowiedniej macierzy, a następnie obliczeniem jej wyznacznika.
* pointInSphere – Funkcja dokonuje sprawdzenia, czy dany punkt wejściowy znajduje się wewnątrz sfery opisanej na ostrosłupie. Dokonywane jest to poprzez wyznaczenie odległości pomiędzy środkiem sfery,a punktem.
* dist – Funkcja wyznacza odległość pomiędzy dwoma punktami.