

URB3D



Metoda trójwymiarowego modelowania obszarów urbanistycznych z wykorzystaniem metod fotogrametrii



Autorzy: Daniel Borkowski · Julia Farganus[✉] · Rafał Mielniczuk · Katarzyna Wochal

Opiekun: Marek Krótkiewicz

Streszczenie

Celem pracy jest wykonanie aplikacji, która wykorzystuje metody fotogrametrii do modelowania miejskich scen 3D. Dane wejściowe stanowią zdjęcia obszarów miejskich, które są przetwarzane w celu stworzenia modelu 3D, a następnie segmentowane na obiekty przestrzeni miejskiej, takie jak budynki, tereny zielone, itp.. Aplikacja będzie wizualizować model oraz wyniki segmentacji semantycznej.

Innowacyjność tego projektu polega na połączeniu, adaptacji i udoskonaleniu najlepszych dostępnych rozwiązań, takich jak Gaussian Splatting i PointNet, aby stworzyć nowy, kompleksowy produkt.

Przetwarzanie dużych scen miejskich jest wyzwaniem dla obecnie istniejących rozwiązań, które skupiają się głównie na pojedynczych obiektach lub zamkniętych scenach. Typowa scena miejska natomiast może obejmować setki zdjęć, a powstała chmura może zawierać miliony punktów. Dodatkowym wyzwaniem jest niebalansowana reprezentacja kategorii semantycznych. Nasze rozwiązanie ma na celu efektywne przetwarzanie dużych zbiorów danych przy rozsądnym zużyciu zasobów czasowych i pamięciowych.

Zastosowania biznesowe otrzymywanych w ten sposób modeli 3D są szerokie: od gier wideo, przez architekturę, robotykę, pojazdy autonomiczne, po modelowanie urbanistyczne.

1 WPROWADZENIE

Nasz projekt skupia się na problemie rekonstrukcji trójwymiarowej scen urbanistycznych, jej klasyfikacji oraz wizualizacji. Warto podkreślić, że obszar naszej pracy jest relatywnie nowy i stawia wyzwania związane z efektywnością przetwarzania dużego zbioru danych - w naszym przypadku chmury punktów, która może składać się nawet z paru milionów punktów. Na rynku dostępne są rozwiązania które możemy wykorzystać, więc naszym głównym celem jest zbadanie ich użyteczności w naszym problemie i ich ewentualna adaptacja.

Nasze rozwiązanie będzie umożliwiało przeprowadzenie rekonstrukcji do modelu trójwymiarowego na podstawie odpowiednio przygotowanego zbioru zdjęć, klasyfikację otrzymanej sceny na zbiór predefiniowanych klas istotnych w kontekście scen urbanistycznych, oraz wizualizację wykonanych obliczeń.

Jako zespół stawiamy następujące cele, które chcemy zrealizować:

1. Skomponowanie własnego zbioru danych
2. Wykorzystanie algorytmu Gaussian Splatting do rekonstrukcji sceny 3D
3. Filtracja chmury punktów przy użyciu różnych technik
4. Zastosowanie architektur sieci neuronowych takich jak PointNet do klasyfikacji chmury punktów
5. Adaptacja istniejących bibliotek do wizualizacji wyników
6. Implementacja własnego algorytmu do renderowania gaussianów

2 STAN WIEDZY

Unikalność naszego projektu wynika z połączenia wielu rozwiązań które istnieją samodzielnie na rynku. Algorytm Structure-from-Motion jest popularną fotogrametryczną techniką uzyskiwania chmury punktów ze zbioru zdjęć i jego implementacja oferowana jest m. in. przez oprogramowanie COLMAP.

W przypadku modelu 3D często stosowaną techniką są siatki, ale ich wadą jest niekompatybilność z algorytmami sztucznej inteligencji. Popularne są też rozwiązania wykorzystujące sieci neuronowe jak np. NeRF [3], jednak długi czas trenowania, osiągający nawet parę dni, jest nieefektywny. Z tego powodu

zdecydowaliśmy się na wykorzystanie algorytmu Gaussian Splatting [1], który buduje model sceny z tzw. gaussianów, które można interpretować jako punkty rozmyte. Można natomiast napotkać różne adaptacje tego algorytmu, jak np. CityGaussian [2].

Ważnym krokiem jest również filtracja chmury punktów w celu usunięcia odstających punktów lub tych nieistotnych dla wyników klasyfikacji. W tym obszarze znajdują się np. techniki statystyczne czy oparte na sąsiedztwie.

Istniejące architektury sieci neuronowych dla zadania segmentacji są głównie przeznaczone dla scen zamkniętych lub pojedynczych obiektów. Popularnym rozwiązaniem jest PointNet [4] oraz jego następnik PointNet++ [5] oparte na wielowarstwowy perceptronie, jak i również bardziej skomplikowane rozwiązania jak KPConv [6] wykorzystujące konwolucje. Wyzwaniem dla naszego projektu będzie dostosowanie takich architektur do chmury punktów o wielkości rzędu milionów punktów.

W przypadku renderowania istnieją rozwiązania przeznaczone zarówno do chmur punktów jak i do splatów, zaimplementowane często przy pomocy WebGL. Wyzwanie stanowi jednak wydajne i efektywne przedstawienie milionów elementów, co wymaga skorzystania z GPU i niskopoziomowego pisania kodu.

3 WYNIKI

3.1 Funkcjonalność oprogramowania

Modularność naszego projektu sprawia, że wygodniej jest omówić niezależnie każdą z części. Dla jasności, przebieg całego procesu jest następujący:

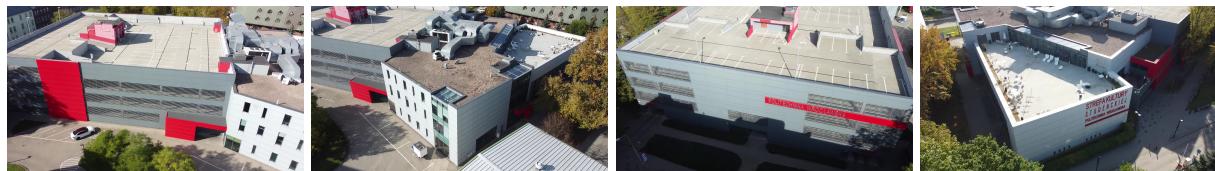
1. Wczytanie plików zdjęć.
2. Rekonstrukcja SFM do chmury punktów, która jest podstawą do kolejnego etapu.
3. Wytrenowanie modelu sceny przy pomocy algorytmu Gaussian Splatting.
4. Filtracja, następnie segmentacja chmury punktów z poprzedniego kroku.

Każda funkcjonalność, jak i wizualizacje wyników poszczególnych zadań są dostępne poprzez interfejs użytkownika.

3.2 Akwizycja danych

W projekcie założono wykorzystanie metod fotogrametrycznych do tworzenia trójwymiarowych modeli obszarów urbanistycznych. Za część projektu przyjęto z tego względu również pozyskanie własnych zestawów danych fotograficznych (fotogramów), które spełniałyby wymogi techniczne, umożliwiające późniejszą rekonstrukcję 3D. Niezbędne było wykonanie dużej liczby ujęć, obejmujących wiele kątów i perspektyw oraz zapewnienie odpowiedniego nakładania się zdjęć dla poprawnego działania oprogramowania fotogrametrycznego, które identyfikuje i dopasowuje wspólne punkty widoczne na wielu zdjęciach.

Akwizycję zrealizowano na kampusie **Politechniki Wrocławskiej**, koncentrując się na budynkach **C5**, **C7** oraz Strefie Kultury Studenckiej (**SKS**) i pozyskując zdjęcia zarówno z lotów bezzałogowym statkiem powietrznym, jak i z poziomu gruntu. Stanowią one kompletne zbiory danych, które spełniły wymogi jakościowe i posłużyły do budowy testowych modeli.



Rysunek 1: Przykładowe zdjęcia z akwizycji danych przedstawiające SKS

3.3 Structure from motion

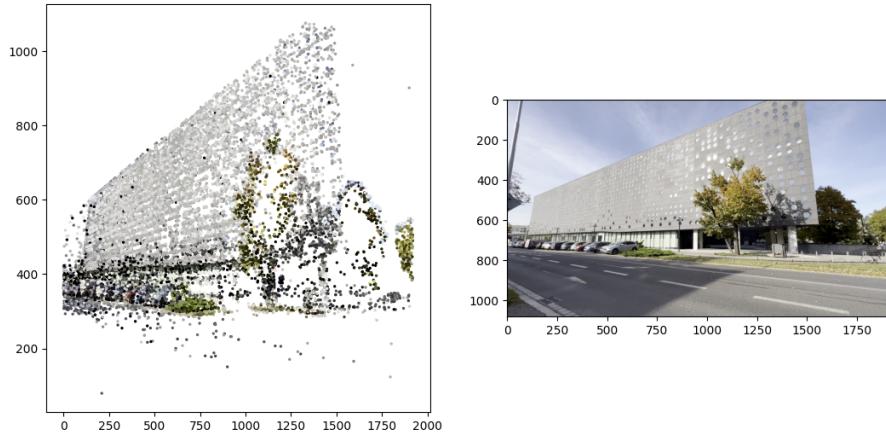
Kolejnym etapem projektu było wykorzystanie techniki *Structure from Motion* (SfM) do wyznaczania struktur przestrzennych scen na podstawie dobranych zestawów zdjęć dwuwymiarowych.

Algorytmy SfM, identyfikując i łącząc punkty wspólne między zdjęciami, ustalają zarówno rozmieszczenie tych punktów w przestrzeni, jak i pozycje i orientacje kamer, z których wykonano zdjęcia. Proces

ten pozwala na oszacowanie struktury trójwymiarowej sfotografowanego obszaru, czyli wygenerowanie chmury punktów odwzorowującej scenę w postaci zbioru punktów 3D o przypisanych kolorach.

Do realizacji tego zadania użyto popularnego narzędzia COLMAP, a konkretne jego wersji w formie biblioteki *pycolmap*, oferującej funkcjonalności m.in. do wykrywania charakterystycznych cech na obrazach, łączenia punktów wspólnych na zdjęciach czy prowadzenia rekonstrukcji sceny 3D na podstawie dopasowań między nimi.

Uzyskana chmura punktów jest dodatkowo poddawana filtracji z wykorzystaniem metod opartych na analizie sąsiedztwa każdego punktu. Proces ten pozwala na eliminację szumów poprzez usunięcie punktów, które nie wpisują się w lokalne wzorce przestrzenne. Tak przygotowana trójwymiarowa reprezentacja sceny służy za podstawę do modelowania z zastosowaniem algorytmu Gaussian Splatting.



Rysunek 2: Projekcja przykładowej chmury punktów na płaszczyznę porównana do zdjęcia

3.4 Gaussian Splatting

Przy pomocy biblioteki *gsplat* [7] zawierającej implementację *Gaussian Splatting* w Pythonie wykonaliśmy eksperymenty polegające na uruchomieniu algorytmu dla różnych wartości hiperparametrów w celu znalezienia wartości które prowadzą do jak najbardziej optymalnego procesu trenowania w kontekście czasu trwania i wykorzystania pamięci.

Na wejściu algorytmu podawana jest chmura punktów, która jest bazą do dalszego dzielenia i powstawania gaussianów, a ich parametry: pozycja, kolor, skala i rotacja są optymalizowane przy pomocy metody spadku wzdłuż gradientu. Metryki przyjęte do oceny jakości to **SSIM** (Structural Similarity Index Measure), **PSNR** (Peak Signal-to-Noise Ratio) oraz **LPIPS** (Learned Perceptual Image Patch Similarity).

W wyniku przeprowadzenia eksperymentów okazało się, że najważniejszymi sterującymi procesem parametrami są

1. Liczba Gaussianów: w przypadku scen urbanistycznych w celu oddania odpowiedniej szczegółowości potrzebne jest parę milionów Gaussianów, dla naszych scen było to zwykle 3 mln.
2. Strategia i częstość adaptacji: określają w jaki sposób oraz jak często dodawane i usuwane są Gaussiany.
3. Liczba iteracji: zwykle im dłużej trenowana jest scena tym lepsze wyniki otrzymujemy, jednak zależy to również od przyjętej strategii. Liczba ta wpływa bezpośrednio na czas trenowania, powinna wynieść nie mniej niż parę tysięcy.
4. Stopień zmiennych harmonicznych: wyrażają one kolor, im większy stopień tym lepsza jakość sceny, ale też zwiększone zużycie pamięci i wydłużony czas trenowania.

Poniżej przedstawione są przykładowe wizualizacje oraz całościowe wyniki dla naszych scen testowych. Renderowania zostały wykonane przy pomocy biblioteki *nerfview* która również służy do wizualizacji splatów. Na poniższych rysunkach są od lewej do prawej: prawdziwe zdjęcie i widok modelu.



Rysunek 3: Scena SKS



Rysunek 4: Scena C5



Rysunek 5: Scena C7

scena	PSNR	SSIM	LPIPS	liczba gaussianów	czas trenowania	pamięć pliku (MB)
SKS	22.03	0.71	0.25	2,937,549	2h53m	661
C5	21.98	0.71	0.26	4,564,464	10h40m	675
C7	22.63	0.72	0.29	3,000,000	15h15m	675

Tabela 1: Całościowe metryki dla testowych scen. Otrzymane wartości PSNR, SSIM oraz LPIPS zwykle świadczą o dobrej jakości scenie, która oddaje wystarczające szczegóły i wygładzone artefakty.

3.5 Segmentacja

3.6 Wizualizacja

Wizualizacja modeli 3D stanowi wyzwanie dla użytkownika końcowego, głównie z powodu braku spójnych platform umożliwiających realizację całego procesu – od wgrania plików wejściowych po interakcję z modelem – w ramach jednej aplikacji. Dostępne na rynku rozwiązania wymagają korzystania z aplikacji trzecich i pewnej wiedzy technicznej, co prowadzi do problemów z integracją i spójnością działania.

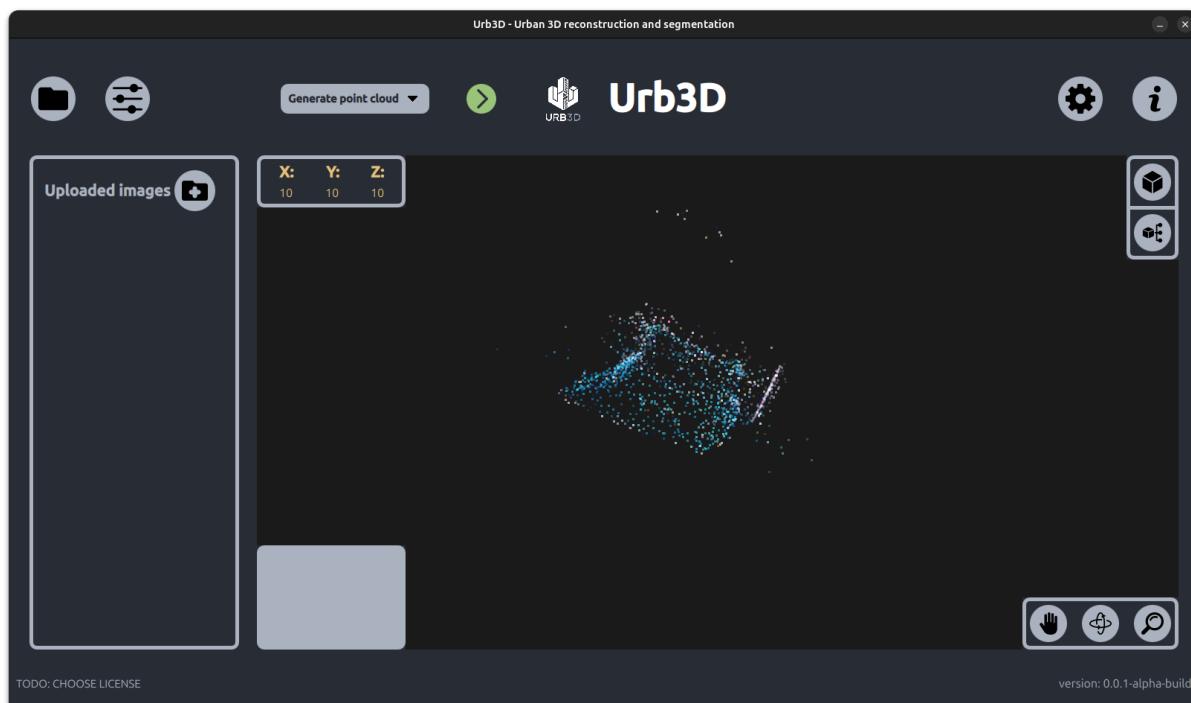
Celem projektu było stworzenie intuicyjnego, dynamicznego i responsywnego **interfejsu** zintegrowanego z wydajnym **renderingiem** GPU. Aplikacja umożliwia użytkownikowi końcowemu realizację pełnego procesu wizualizacji – od tworzenia i modyfikacji modelu po jego segmentację i wyświetlanie – w jednej aplikacji. Projekt rozwiązuje problem fragmentarycznej funkcjonalności dostępnych aplikacji, oferując spójne środowisko do obsługi modeli 3D.

Korzyści z realizacji własnej wizualizacji

- zwiększoną wydajność dzięki GPU,
- eliminacją konieczności korzystania z wielu narzędzi,
- uproszczony proces użytkowania, co zwiększa dostępność aplikacji dla mniej zaawansowanych użytkowników.

Rendering wykorzystuje plik .ply jako dane wejściowe do wczytania splatów. Są one renderowane jako sześciany, w których wnętrzu generowane są shadery, bazujące na skalowaniu i rotacji splatów. Takie podejście umożliwia abstrakcyjne przedstawienie splatów przy jednoczesnym zachowaniu wysokiej dokładności wizualnej.

Interfejs został zaimplementowany w QML, PyQt, natomiast rendering w technologiiach C, OpenGL oraz OpenCL.



Rysunek 6: Zrzut ekranu przedstawiający główny widok aplikacji

4 PODSUMOWANIE

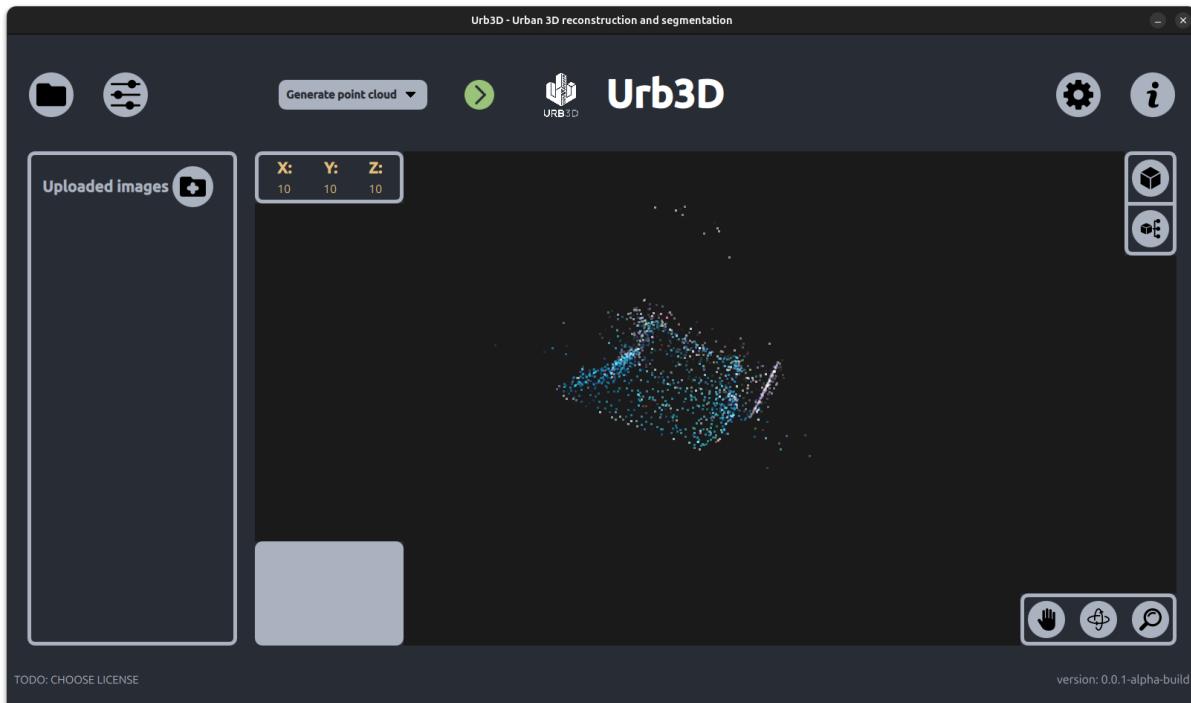
Rekonstrukcja i klasyfikacja krajobrazów urbanistycznych ma wiele potencjalnych zastosowań w dziedzinach takich jak *Smart City* czy też *Virtual Reality*. Nasz projekt pokazał, że wykonanie takiego oprogramowania jest możliwe przy pomocy integracji istniejących rozwiązań i ich adaptacji.

4.1 Wnioski

Kluczowymi czynnikami decydującymi o wykorzystaniu danego rozwiązania są: wykorzystanie zasobów, łatwość użycia oraz oferowane funkcje. Nasz projekt dzięki przejrzystemu interfejsowi użytkownika zapewnia intuicyjne korzystanie z funkcjonalności takich jak dostosowywanie parametrów, wczytywanie zdjęć, uruchamianie poszczególnych etapów oraz przeglądanie rezultatów. Dzięki trafnemu wyborowi algorytmów zapewniamy odpowiednią jakość wyników, które mogą być praktycznie zastosowane razem lub oddzielnie.

4.2 Kierunki rozwoju

Większe obszary / Rekonstrukcji? Algorytm Gaussian Splatting - wiecej opcji / wariantacji Segmentacji? Rendering?



Rysunek 7: Zrzut ekranu przedstawiający własny renderer

4.3 Podziękowania

?

LITERATURA

- [1] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.
- [2] Yang Liu, He Guan, Chuanchen Luo, Lue Fan, Junran Peng, and Zhaoxiang Zhang. Citygaussian: Real-time high-quality large-scale scene rendering with gaussians, 2024.
- [3] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.
- [4] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2016.
- [5] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017.
- [6] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds, 2019.
- [7] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for Gaussian splatting. *arXiv preprint arXiv:2409.06765*, 2024.