

Politechnika Wrocławskiego
Wydział Informatyki i Telekomunikacji

ZESPOŁOWE PRZEDSIĘWZIĘCIE INŻYNIERSKIE

**Metoda trójwymiarowego modelowania
obszarów urbanistycznych
z wykorzystaniem metod fotogrametrii**

Julia Farganus
Katarzyna Wochal
Daniel Borkowski
Rafał Mielniczuk

Opiekun pracy
dr hab. inż. Marek Krótkiewicz, prof. PWr

Spis treści

1 Wprowadzenie	1
1.1 Cel i zakres prac	1
2 Stan wiedzy w obszarze przedsięwzięcia	2
2.1 Rekonstrukcja	2
2.1.1 SFM	2
2.1.2 Gaussian Splatting	2
2.2 Segmentacja	3
3 Projekt	3
3.1 Specyfikacja wymagań na produkt programowy	3
3.2 Architektura	3
3.3 Implementacja	3
3.3.1 Struktura plikowa projektu	5
3.3.2 Gaussian Splatting	5
3.4 Testy	7
3.4.1 Testy jednostkowe	7
4 Wyniki i analiza badań	7
4.1 Rekonstrukcja - Gaussian Splatting	7
4.1.1 Wstęp	7
4.1.2 Zależność wyników od wartości hiperparametrów	8
4.1.3 Porównanie zdjęć między eksperymentami	11
4.1.4 Wyniki dla pozostałych scen	11
4.1.5 Podsumowanie	12
5 Podsumowanie	12

1 Wprowadzenie

Przedmiotem projektu było wykonanie aplikacji wykorzystującej metody fotogrametrii do modelowania trójwymiarowych scen miejskich. Zaimplementowany program umożliwia użytkownikowi zrekonstruowanie chmury punktów i modelu 3D na podstawie podanych na wejściu zdjęć fotogrametrycznych za pomocą dostosowanych do specyfiki problemu w toku eksperymentów rozwiązań *structure from motion* i *gaussian splatting*. Otrzymana w ten sposób chmura może być przez użytkownika poddana segmentacji semantycznej, przeprowadzanej przez wytrenowany do tego celu model sztucznej inteligencji w postaci sieci neuronowej. Aplikacja oferuje również wizualizację wykonanych obliczeń, która możliwa jest dzięki użyciu przystosowanego dla większej wydajności mechanizmu renderowania.

Wytwarzony produkt informatyczny ze względu na integrację wielu rozwiązań i efektywną implementację procesu *end-to-end* przejawia potencjał w zastosowaniach biznesowych począwszy od branż takich jak gry wideo, przez architekturę, robotykę, pojazdy autonomiczne, skończywszy na modelowaniu urbanistycznym.

1.1 Cel i zakres prac

Rekonstrukcja, klasyfikacja i wizualizacja scen urbanistycznych to dynamicznie rozwijające się zagadnienie, które zyskało na znaczeniu dzięki rosnącej dostępności nowoczesnych technologii, takich jak LiDAR, oraz postępowi w dziedzinie sztucznej inteligencji. Kluczowym wyzwaniem pozostaje jednak efektywne przetwarzanie ogromnych zbiorów danych – chmur punktów, które nierazadko obejmują miliony elementów. Choć na rynku istnieją liczne programy i algorytmy wspierające tego typu analizy, ich skuteczne wykorzystanie w praktyce bywa problematyczne, głównie z uwagi na skalę i złożoność danych urbanistycznych.

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

Poniżej znajdują się cele, które zostaną zrealizowane w przedsięwzięciu:

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 w obszarze przedsięwzięcia

2.1 Rekonstrukcja

Wirtualne reprezentacje krajobrazów miejskich są coraz bardziej wykorzystywane w różnego rodzaju zadaniach. Mimo rozwoju technik, począwszy od pozyskiwania danych (np. LiDAR), skończywszy na parametrycznych modelach (np. sieci neuronowe), zadanie modelowania pozostawia nierozerwiane problemy wpływające na jakość modelu wynikające z np. trudności akwizycji danych. Istnieje wiele sposobów rekonstrukcji które można podzielić ze względu na dane wejściowe, poziom szczegółowości, automatyzację czy też dane wyjściowe. W naszym projekcie rekonstrukcja zachodzi na dwóch etapach - najpierw jako chmura punktów, następnie jako zbiór splatów.

2.1.1 SFM

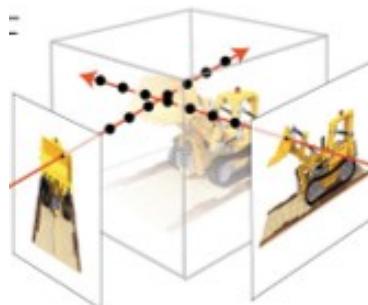
Opis 'sparse reconstruction'

2.1.2 Gaussian Splatting

Popularną techniką rekonstrukcji z chmury punktów jest siatka (ang. mesh), wykorzystana np. w pracy City3D[city3D]. Często jednak uzyskanie dobrej jakości siatki wymaga kombinacji wielu różnych geometrycznych algorytmów. Wraz z rozwojem sztucznej inteligencji zaczęły pojawiać się i w dziedzinie rekonstrukcji rozwiązań wykorzystujących sieci neuronowe - takie jak np. NeRF[nerf], gdzie informacje o scenie zawarte są w wagach modelu. Niedoskonałością tego rozwiązania jest jednak długi czas trenowania nawet dla sceny jednego obiektu, wahający się od parunastu godzin do paru dni. Z pomocą przychodzą inne metody, jak np. Gaussian Splatting [gaussiansplatting], który rezygnuje w całości z sieci neuronowej i wykorzystuje zbiór tzw. "splatów" do zbudowania sceny. Jako że w naszym projekcie stawiamy na optymalne wykorzystanie zasobów, to zdecydowaliśmy się na wybór właśnie ostatniej z wymienionych metod.



Rysunek 1: City3D - siatka



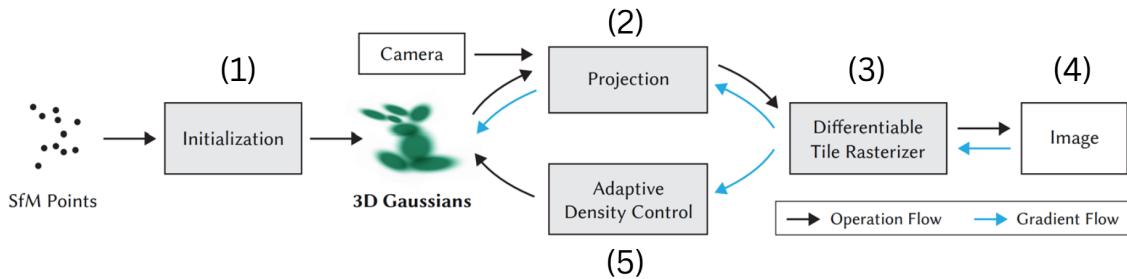
Rysunek 2: Nerf - sieć neuronowa



Rysunek 3: Gaussian Splatting - splat'y

Splat (tłum. punkt rozmyty) jest rozszerzeniem punktu i posiada atrybuty

- środek (x, y, z)
- kolor
- przeźroczystość
- macierz kowariancji (koduje rotację i skalę)



Rysunek 4: Optymalizacja zbioru splatów

Najlepszy zbiór splatów opisujący scenę jest znajdowany w procesie optymalizacji opisanym poniżej schematem

1. Wykorzystanie chmury punktów z rekonstrukcji doinicjalizacji zbioru splatów
2. Rzutowanie perspektywiczne 3-wymiarowych gaussianów na obraz widziany z danej kamery, czego wynikiem jest zbiór dysków
3. Renderowanie polegające na akumulowaniu dla każdego piksela wkładu od różnych nachodzących dysków
4. Obliczanie funkcji straty - porównanie z prawdziwym obrazkiem z danej kamery
5. Adaptacja gęstości gaussianów - klonowanie, dzielenie lub usuwanie wg. kryteriów zależnych od strategii

Niestety metoda ta nie pozostaje bez wad. W przypadku dużych zbiorów danych zużycie pamięci może wynieść nawet parę GB, co wymaga dużej jakości karty graficznej. Inną kwestią jest duża liczba hiperparametrów, które często wymagają ręcznego dostosowania do danej sceny.

2.2 Segmentacja

Warto nadmienić, że nie jest to segmentacja [**pointnet**].

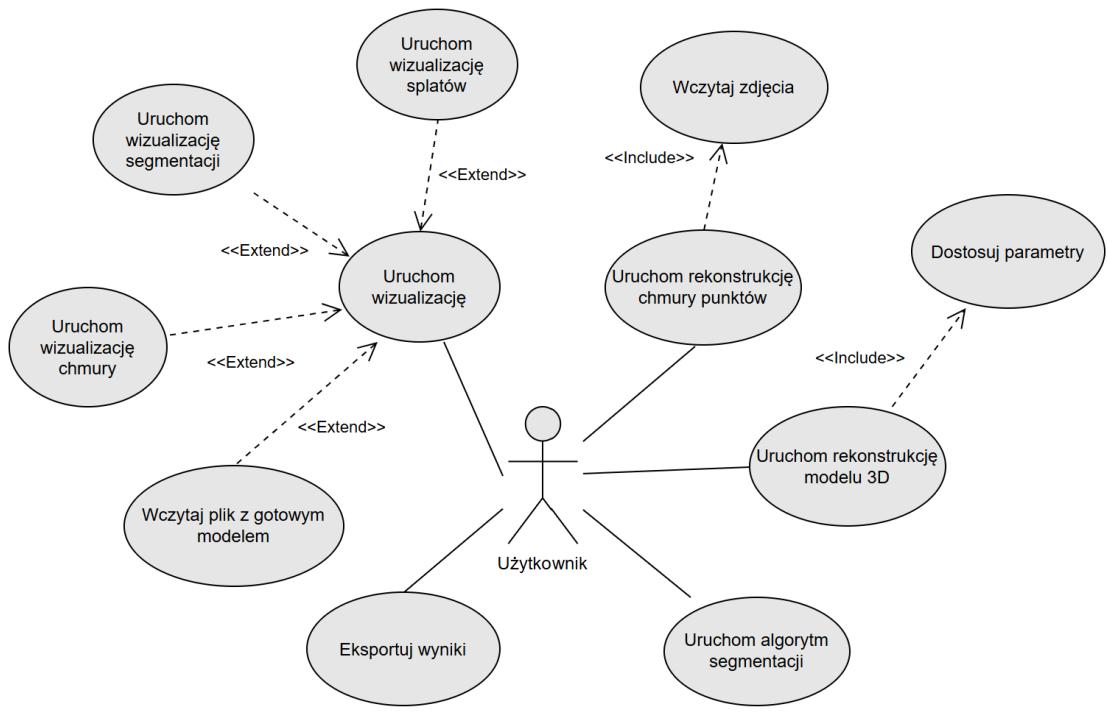
3 Projekt

3.1 Specyfikacja wymagań na produkt programowy

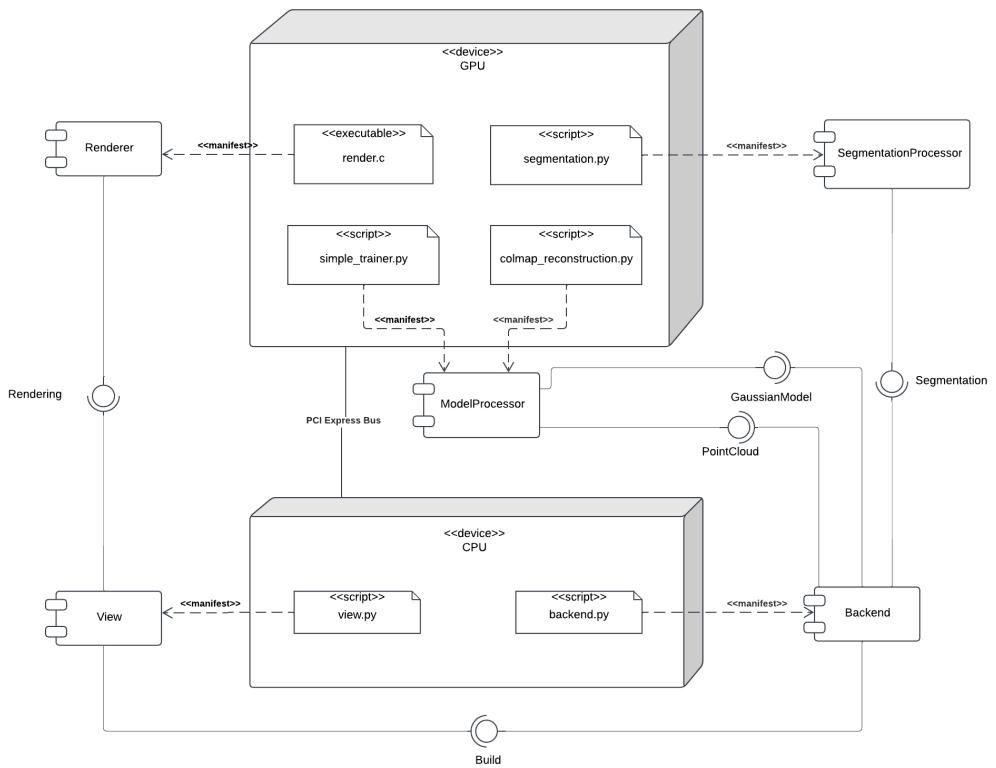
3.2 Architektura

3.3 Implementacja

Opis zastosowanych technologii i dokumentów

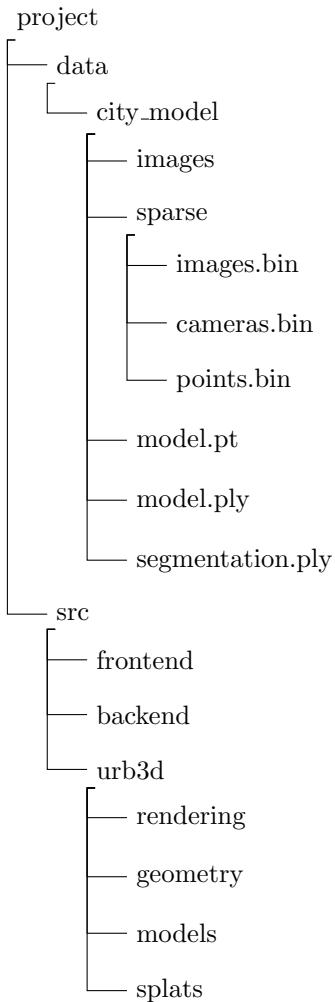


Rysunek 5: Diagram przypadków użycia



Rysunek 6: Diagram wdrożenia komponentów

3.3.1 Struktura plikowa projektu



3.3.2 Gaussian Splatting

W celu uruchomienia algorytmu Gaussian Splatting korzystamy z gotowej implementacji którą zapewnia biblioteka *gsplat* [[ye2024gsplatosource library gaussian](#)]. Znajdują się w niej metody które odzwierciedlają wykonywane kroki w teoretycznym opisie algorytmu (np. *rasterization*) a także dodatkowe elementy oparte na nowszych badaniach które usprawniają proces trenowania i renderowania (np. *sparse gradient*, *absgrad*, itp.). Dodatkowym plusem biblioteki jest wsparcie do renderowania dużych scen. Jej użycie wymaga posiadania karty graficznej CUDA.

Jedną z naszych kontrybucji jest dostosowanie istniejącego skryptu trenującego *simple_trainer* tak aby działał on poprawnie i zgodnie z naszymi wymaganiami. Przyjmuje on następujące argumenty:

```
1 options:
2 -h, --help                  show this help message and exit
3 --data_dir DATA_DIR        Path to the data directory.
4 --result_dir RESULT_DIR    Path to the results directory.
5 --data_factor DATA_FACTOR  Data factor.
6 --init_type {sfm,random}    Initialization type.
7 --strategy {default,mcmc}  Strategy type.
8 --max_steps MAX_STEPS     Maximum number of steps.
9 --init_num_pts INIT_NUM_PTS Initial number of points (only for random).
10 --delta_steps DELTA_STEPS Delta steps for evaluation and saving.
11 --scale_reg SCALE_REG     Scale regularization value.
```

```

20  --opacity_reg OPACITY_REG
21      Opacity regularization value.
22  --min_opacity MIN_OPACITY
23      Minimum opacity.
24  --refine_stop_iter REFINE_STOP_ITER
25      Refinement stop iteration.
26  --refine_every REFINE_EVERY
27      Refine frequency (iterations).
28  --refine_start_iter REFINE_START_ITER
29      Refinement start iteration.
30  --reset_every RESET_EVERY
31      Reset opacities every this steps. [strategy=default]
32  --pause_refine_after_reset PAUSE_REFINE_AFTER_RESET
33      Pause refining GSS until this number of steps after reset.
34          [strategy=default]
35  --cap_max CAP_MAX
36      Maximum cap for MCMC gaussians. [strategy=mcmc]
37  --sh_degree_interval SH_DEGREE_INTERVAL
38      Add spherical harmonics degree interval.
39  --sh_degree {1,2,3}
40      Degree of spherical harmonics.
41  --init_scale INIT_SCALE
42      Initial scale.
43  --init_opa INIT_OPA
44      Initial opacity.
45  --packed PACKED
46      Use packed mode for rasterization.
47  --sparse_grad SPARSE_GRAD
48      Use sparse gradients for optimization.

```

Z których obowiązkowe to *data_dir* - ścieżka do folderu z rekonstrukcją i zdjęciami i *result_dir* - ścieżka gdzie zostaną zapisane wyniki.

Opis parametrów

- *data_dir* - obowiązkowy; ścieżka do folderu z rekonstrukcją i zdjęciami
- *result_dir* - obowiązkowy; ścieżka gdzie zostaną zapisane wyniki
- *init_type* - sposób inicjalizacji chmury punktów, losowy (*random*) lub z rekonstrukcji sfm (*sfm*)
- *strategy* - strategia zagęszczania gaussianów
- *max_steps* - maksymalna liczba iteracji
- *init_num_pts* - początkowa liczba punktów dla sposobu inicjalizacji *random*
- *delta_steps* - liczba iteracji po których dokonywać ewaluacji oraz zapisu checkpointa
- *scale_reg* - współczynnik regularyzacji skali gaussianów
- *opacity_reg* - współczynnik regularyzacji przeźroczystości gaussianów
- *min_opacity* - prog przeźroczystości poniżej którego splaty są odrzucane
- *refine_every* - liczba iteracji po których następuje ulepszanie gaussianów wg. wybranej strategii
- *refine_start_iter* - liczba iteracji od początku trenowania po której zachodzi pierwszy raz ulepszanie
- *reset_every* - liczba iteracji po której zachodzi wyzerowanie przeźroczystości jeśli strategią jest *default*
- *pause_refine_after_reset* - liczba iteracji po wyzerowaniu które należy odczekać aby zaszedł krok doskonalenia gaussianów jeśli strategią jest *default*
- *cap_max* - maksymalna liczba gaussianów jeśli strategią jest *mcmc*
- *sh_degree_interval* - liczba iteracji po której dodawany jest kolejny stopień zmiennych harmonicznych
- *sh_degree* - stopień współczynników zmiennych harmonicznych
- *init_scale* - początkowa skala gaussianów
- *init_opa* - początkowa przeźroczystość gaussianów

3.4 Testy

3.4.1 Testy jednostkowe

W ramach przetestowania poprawności kodu zostały wykonane dla części skryptów testy jednostkowe przy pomocy biblioteki *pytest*.

```
1 scripts\test_colmap_reconstruction.py::test_colmap_reconstruction_files_exist
2 scripts\test_distance_pcd_filtering.py::test_distance_pcd_filtering_files_exist
3 scripts\test_distance_pcd_filtering.py::test_distance_pcd_filtering_works
4 scripts\test_distance_pcd_filtering.py::test_error_on_wrong_type
5 scripts\test_save_model.py::test_save_model
6 scripts\test_save_model.py::test_fail_on_wrong_path_name
7 scripts\test_segmentation.py::test_segmentation_files_exist
8 scripts\test_segmentation.py::test_segmentation_class_is_added
9 scripts\test_segmentation.py::test_segmentation_classes_are_in_range
10 scripts\test_segmentation.py::test_segmentation_fails_on_corrupted_input
11 scripts\test_simple_trainer.py::test_simple_trainer_files_exist
12 scripts\test_simple_trainer.py::test_simple_trainer_pt_file_is_correct
13 scripts\test_simple_trainer.py::test_training_fails_on_incomplete_input
14 scripts\test_statistical_pcd_filtering.py::test_error_on_wrong_type
```

Testy obejmowały m. in. sprawdzenie poprawności powstałych plików oraz występowania błędów.

4 Wyniki i analiza badań

To co wyszło i jak

4.1 Rekonstrukcja - Gaussian Splatting

4.1.1 Wstęp

Jak już wcześniej zostało wspomniane, algorytm posiada wiele hiperparametrów które mogą mieć różne optymalne wartości w zależności od sceny. Ogólnym więc zaleceniem jest uruchomienie algorytmu dla różnych wartości, a za pomoc mogą posłużyć poniższe wyniki badań.

Uwaga: do renderowania została użyta funkcjonalność biblioteki gsplat, więc wyniki mogą się różnić dla naszego własnego renderowania.

Przyjęte metryki

1. SSIM (ang. Structural Similarity Index Measure) - miara podobieństwa zdjęć, która bierze pod uwagę percepcyjne zjawiska np. związane z jasnością. Zakres wartości wynosi [-1, 1] wyższa wartość oznacza lepszą jakość,
2. PSNR (ang. Peak Signal-to-Noise Ratio) - stosunek maksymalnej mocy sygnału do mocy szumu zakłócającego ten sygnał, wyrażany w decybelach. Zakres wartości wynosi od 0 do ∞ , wyższa wartość oznacza lepszą jakość,
3. LPIPS (ang. Learned Perceptual Image Patch Similarity) - metryka oparta na badaniach związanych z sieciami neuronowymi. Niższa wartość oznacza lepszą jakość.

Eksperymenty dla sceny SKS

Poniżej zostaną opisane wyniki różnych trenowań dla sceny SKS. W tabeli 1 uwzględnione są najważniejsze hiperparametry od których zależy jakość trenowania i końcowych wyników, natomiast tabela 2 zawiera wyniki dla odpowiadających przebiegów.

i	init_opa	init_scale	init_type	scale_reg	sh_degree	sh_interval	strategy	cap_max	refine_every
0	0.10	1.00	sfm	0.00	3	5000	default	-	100
1	0.10	1.00	sfm	0.01	3	5000	default	-	100
2	0.10	1.00	sfm	0.01	3	5000	default	-	500
3	0.10	1.00	sfm	0.01	3	5000	default	-	1000
4	0.50	0.10	sfm	0.01	3	5000	default	-	1000
5	0.10	1.00	sfm	0.01	3	5000	mcmc	3000000	100
6	0.10	1.00	sfm	0.01	3	5000	mcmc	3000000	500
7	0.10	1.00	sfm	0.01	3	5000	mcmc	3000000	1000
8	0.50	0.10	sfm	0.01	3	5000	mcmc	3000000	100
9	0.50	0.10	sfm	0.01	1	5000	mcmc	3000000	100
10	0.50	0.10	sfm	0.01	2	5000	mcmc	3000000	100
11	0.50	0.10	sfm	0.01	2	5000	default	-	500
12	0.50	0.10	sfm	0.01	1	5000	default	-	100
13	0.50	0.10	sfm	0.01	2	5000	mcmc	3000000	500
14	0.50	0.10	sfm	0.01	1	5000	mcmc	3000000	500
15	0.50	0.10	sfm	0.01	2	5000	default	-	1000
16	0.50	0.10	sfm	0.01	2	5000	mcmc	3000000	1000

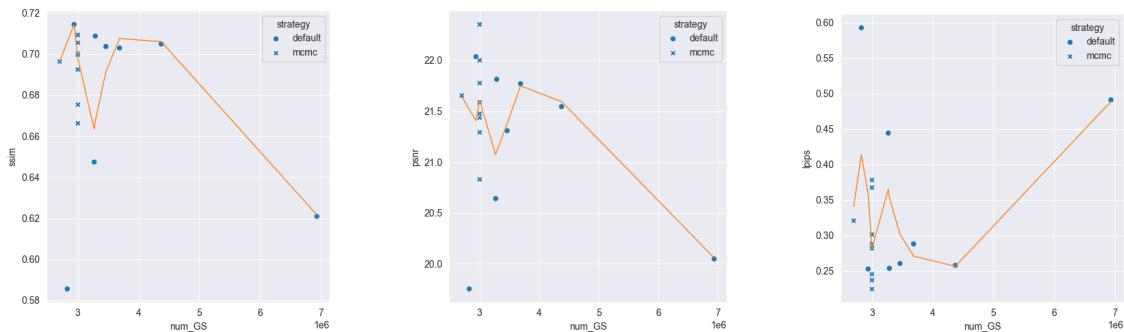
Tabela 1: Konfiguracja trenowania dla sceny SKS

i	iteration	psnr	ssim	lpips	num_GS	time (h)	size (MB)
0	19900	20.64	0.65	0.44	3,268,771	4.85	735.00
1	11500	19.75	0.59	0.59	2,825,402	2.01	635.00
2	46500	21.77	0.70	0.29	3,685,999	4.72	829.00
3	71999	21.81	0.71	0.25	3,286,360	5.78	739.00
4	63200	22.04	0.71	0.25	2,937,549	2.88	661.00
5	56999	22.35	0.71	0.24	3,000,000	7.96	675.00
6	41999	21.65	0.70	0.32	2,699,815	3.98	607.00
7	89499	20.83	0.67	0.38	3,000,000	10.46	675.00
8	31999	21.59	0.68	0.37	3,000,000	9.24	675.00
9	44499	21.29	0.69	0.29	3,000,000	4.94	263.00
10	56999	21.44	0.70	0.25	3,000,000	6.09	434.00
11	51999	21.54	0.70	0.26	4,373,358	3.97	633.00
12	14499	20.05	0.62	0.49	6,934,038	0.66	608.00
13	46999	22.00	0.71	0.28	3,000,000	4.55	434.00
14	44499	21.78	0.70	0.30	3,000,000	3.91	263.00
15	79499	21.31	0.70	0.26	3,461,759	4.25	501.00
16	109499	21.47	0.71	0.22	3,000,000	9.58	434.00

Tabela 2: Wyniki eksperymentów dla sceny SKS

4.1.2 Zależność wyników od wartości hiperparametrów

Liczba Gaussianów

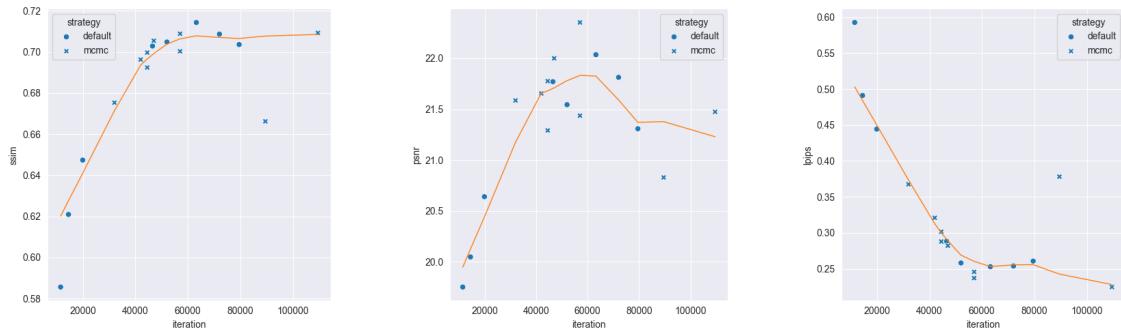


Rysunek 7: Zależność wartości metryk od liczby Gaussianów

Wyniki na wykresie 7 mogą zaprzeczyć oczekiwaniu, że wraz z większą liczbą Gaussianów jakość sceny się polepsza. Zależy to również od strategii:

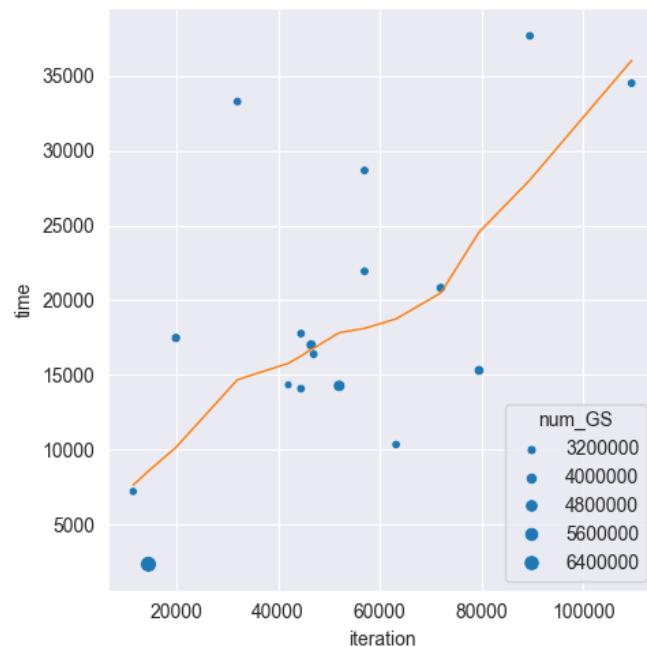
1. default - strategia ta nie ma górnego ograniczenia na liczbę gaussianów, co może komplikować proces uczenia, gdyż nie można kontrolować końcowej ilości punktów,
2. mcmc - strategia ta ma ograniczenie na liczbę gaussianów, co może jednak sprawiać trudność, gdyż zwykle trudno oszacować z góry jaka liczba punktów będzie odpowiednia.

Liczba iteracji



Rysunek 8: Zależność wartości metryk od liczby iteracji

W przypadku liczby iteracji jak widać na wykresach 8 trend polepszania się metryk wraz z większą ilością powtórzeń trenowania. Należy jednak wziąć pod uwagę to, że dla różnych ustawień minimalny czas uczenia może być różny, dlatego najlepiej jest na bieżąco obserwować metryki w trakcie trenowania i przerwać w momencie, gdy już one się nie poprawiają.

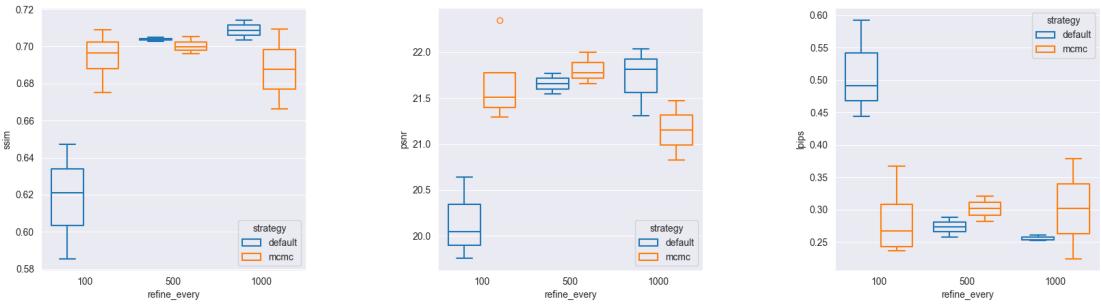


Rysunek 9: Czas trenowania w zależności od liczby iteracji

Istotne jest jednak podkreślenie tego, że dla tej samej liczby iteracji czas może być różny, tak jak to pokazuje wykres 9, co też wynika od np. przyjętej strategii czy też maszyny. Zwykle przyrost liczby Gaussianów w czasie jest wykładniczy, co też wykładniczo wydłuża czas jednej iteracji.

Częstość adaptacji

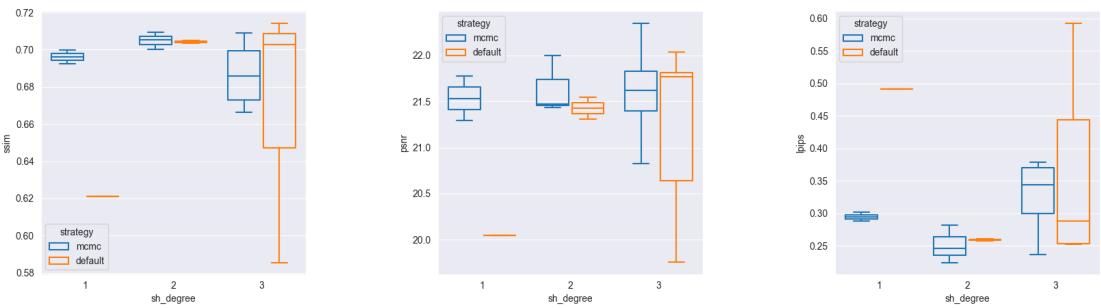
Powysze dwie zmienne - liczba Gaussianów oraz liczba iteracji - najbardziej są uzależnione od częstości adaptacji Gaussianów, ponieważ wtedy zachodzi dodawanie lub usuwanie Gaussianów. Testowane były wartości: 100, 500 i 1000, tak jak to pokazują wykresu 10. Ich zachowanie zależy od wybranej strategii:



Rysunek 10: Zależność wartości metryk od częstości adaptacji

1. default - jak już wspomniano strategia ta nie ma górnego ograniczenia na liczbę Gaussianów, co powoduje szybką ich proliferację bez pozytywnego wpływu na jakość sceny, więc zalecana jest wartość powyżej 500.
2. mcmc - zalecana jest wartość poniżej 500, gdyż wtedy algorytm szybko osiągnie maksymalną liczbę Gaussianów i przez resztę iteracji będzie poprawiał ich parametry.

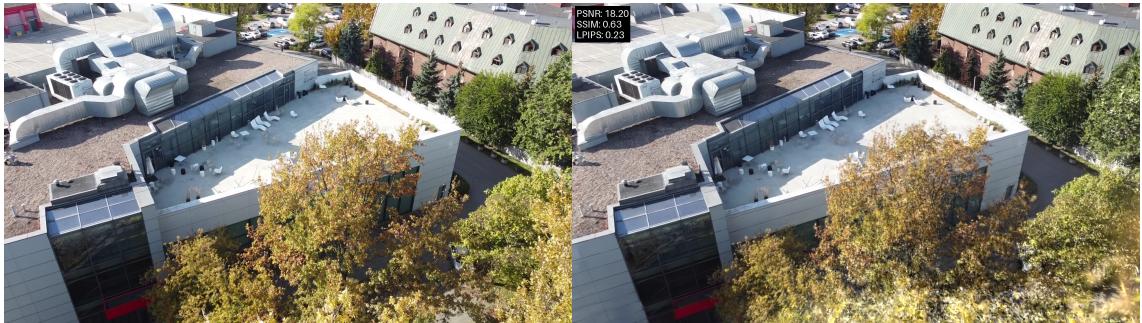
Stopień zmiennych harmonicznych



Rysunek 11: Zależność wartości metryk od stopnia zmiennych harmonicznych

Prawie wszystkie eksperymenty zakładały wartość 3, co daje najlepszą jakość zdjęć, jednak za cenę zwiększonego zużycia pamięci i wydłużonego czasu trenowania. Jednak jak pokazuje rozkład na ilustracjach 11 wystarczająco dobre metryki można uzyskać z $sh_degree = 2$, jednocześnie zyskując na oszczędzonych zasobach.

4.1.3 Porównanie zdjęć między eksperymentami



Rysunek 12: Eksperyment i = 5



Rysunek 13: Eksperyment i = 5

Zdjęcie 12 z dobrą jakością pokazuje szczegóły jak np. krzesła, jednak widoczne są w dolnym prawym boku artefakty związane z niewystarczającą ilością zdjęć ujmujących ten obszar sceny. Podobnie na zdjęciu 13, dlatego odpowiednio wykonana akwizycja danych jest kluczowa dla jakości rekonstrukcji.

4.1.4 Wyniki dla pozostałych scen



Rysunek 14: Budynek C5

scena	PSNR	SSIM	LPIPS	liczba gaussianów	czas trenowania	pamięć pliku (MB)
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 3: Wyniki dla scen C5 i C7

Ilustracje 14 i 15 pokazują ujęcia dla pozostałych scen testowych, a tabela 3 całkowite metryki.



Rysunek 15: Budynek C7

4.1.5 Podsumowanie

Eksperymenty zostały przeprowadzone na jednej scenie ze względu na ograniczenia czasowe, jednak powyższe wnioski powinny się tyczyć innych scen. *gsplat* proponuje również inne parametry które mogą polepszyć jakość sceny, ale ze względu na ograniczenia nie zostały przetestowane.

5 Podsumowanie

koniec i bomba