

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

blablabla zamierzamy zrobimy misja

1.1 Cel i zakres prac

To z opisu

1.2 tutaj opis z fiszki

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 niezbalansowana reprezentacja kategorii semantycznych. Nasze rozwiązanie ma na celu efektywne przetwarzanie dużych zbiorów danych przy rozsądny 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.

2 Stan wiedzy w obszarze przedsięwzięcia

2.1 Rekonstrukcja

Można wyodrębnić gaussian splatting od sfm.

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

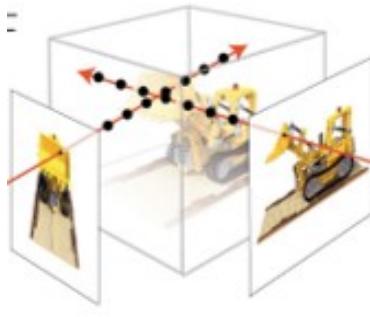
Popularną techniką rekonstrukcji z chmury punktów jest siatka (ang. mesh), wykorzystana np. w pracy City3D[3]. Często jednak uzyskanie dobrej jakości siatki wymaga kombinacji wielu różnych geometrycznych algorytmów. Wraz z rozwojem sztucznej inteligencji zaczęły pojawiać się i w dziedzinie rekonstrukcji rozwiązań wykorzystujące sieci neuronowe - takie jak np. NeRF[2], gdzie informacje o scenie zawarte są w wagach modelu. Niedoskonałość 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 [1], 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.

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

- środek (x, y, z)
- kolor



Rysunek 1: City3D - siatka



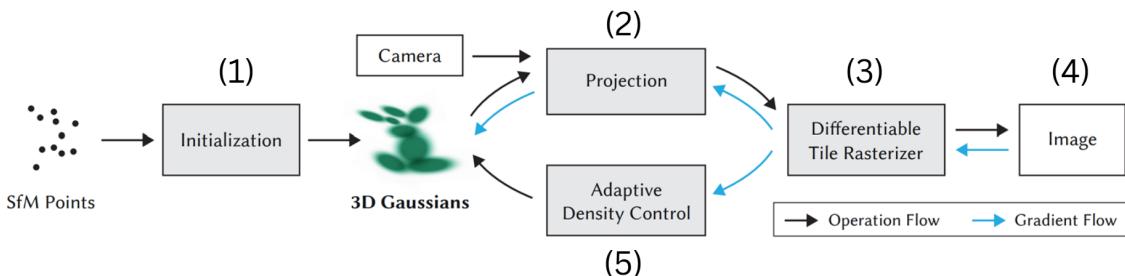
Rysunek 2: Nerf - sieć neuronowa



Rysunek 3: Gaussian Splatting - splat'y

- przeźroczystość
- macierz kowariancji (koduje rotację i skalę)

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



Rysunek 4: Optymalizacja zbioru splatów

1. Wykorzystanie chmury punktów z rekonstrukcji do inicjalizacji 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 dobrej jakości karty graficznej. Inną kwestią jest duża liczba hiperparametrów, które często wymagają ręcznego dostosowania do danej sceny.

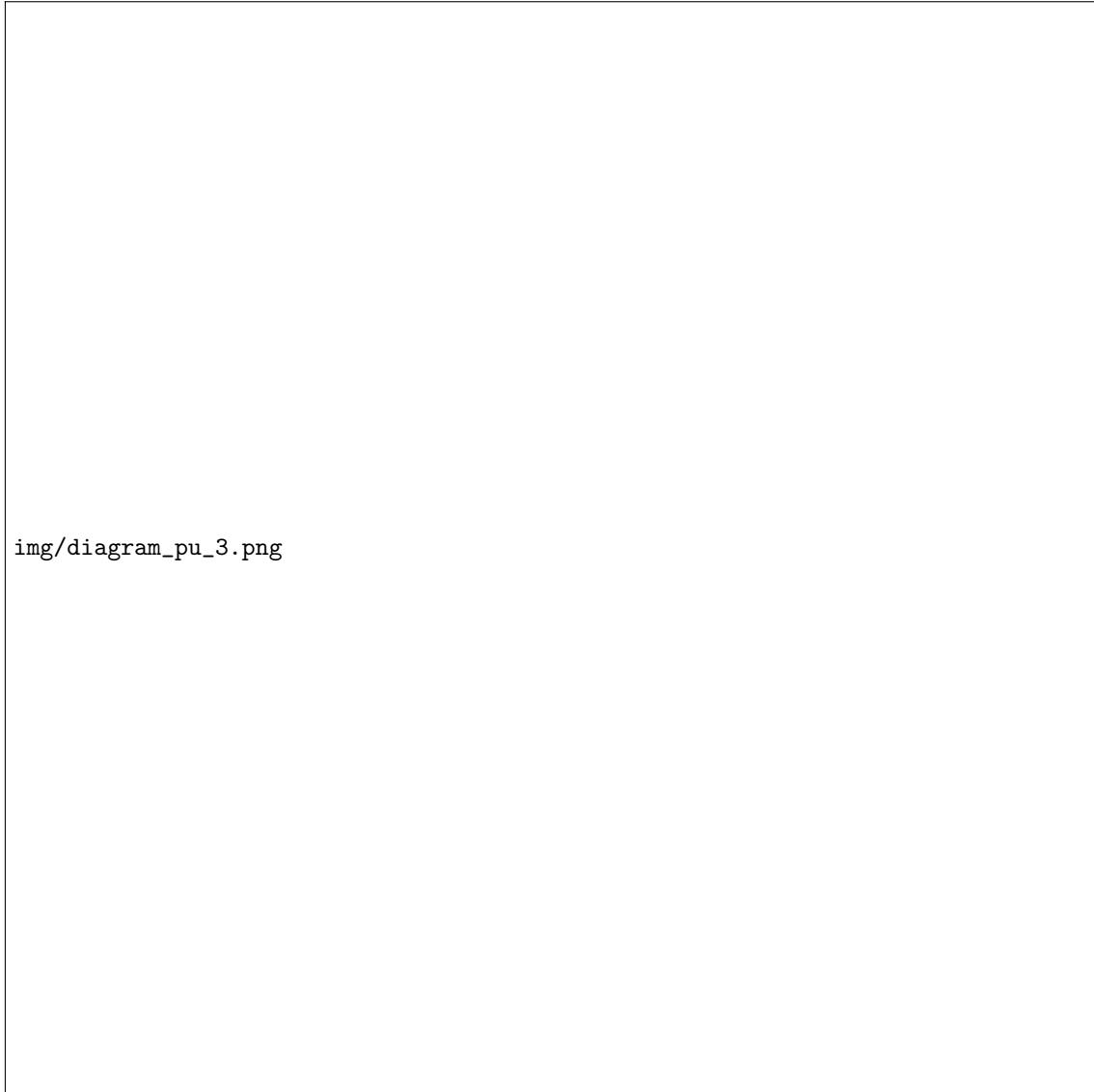
2.2 Segmentacja

Warto nadmienić, że nie jest to segmentacja [4].

3 Projekt

3.1 Specyfikacja wymagań na produkt programowy

Czy my tu musimy dać diagramy wymagań? :O Diagram przypadków użycia



Rysunek 5: Diagram przypadków użycia

3.2 Architektura

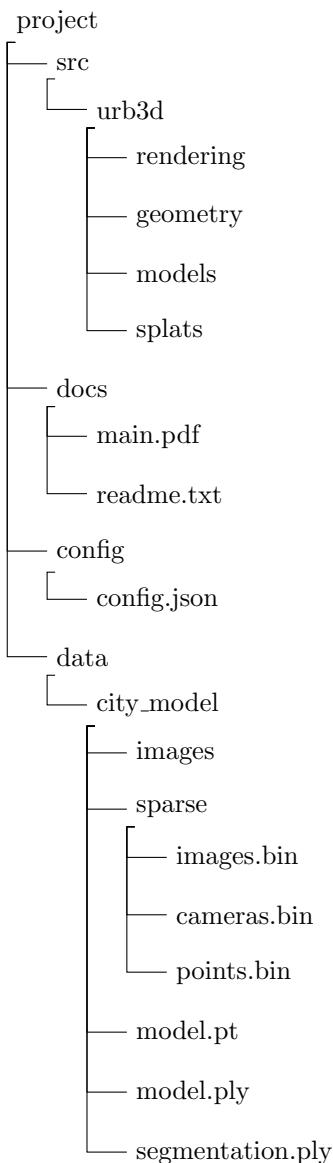
Projekt architektury, zastosowane wzorce projektowe??? Jakiś język graficzny tutaj

3.3 Implementacja

Opis zastosowanych technologii i dokumentów

3.3.1 Struktura plikowa projektu

to wrzucić pod implementację



3.3.2 Gaussian Splatting

W celu uruchomienia algorytmu Gaussian Splatting korzystamy z gotowej implementacji która zapewnia biblioteka *gsplat* [5]. 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
11
12
13
14
15

```

```

16  --delta_steps DELTA_STEPS
17          Delta steps for evaluation and saving.
18  --scale_reg SCALE_REG
19          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      Maximum cap for MCMC gaussians. [strategy=mcmc]
36  --sh_degree_interval SH_DEGREE_INTERVAL
37          Add spherical harmonics degree interval.
38  --sh_degree {1,2,3}    Degree of spherical harmonics.
39  --init_scale INIT_SCALE
40          Initial scale.
41  --init_opa INIT_OPA    Initial opacity.
42  --packed PACKED       Use packed mode for rasterization.
43  --sparse_grad SPARSE_GRAD
44          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

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 naszych 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.

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

Tabela 1: Konfiguracja trenowania dla sceny SKS

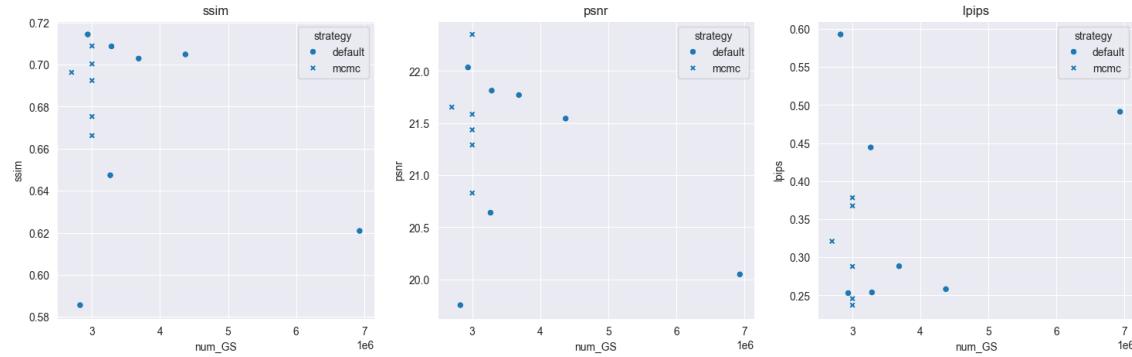
W powyższej tabeli ?? uwzględnione są najważniejsze hiperparametry od których zależy jakość trenowania i końcowych wyników.

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

Tabela 2: Wyniki eksperymentów dla sceny SKS

4.1.2 Zależność wyników od wartości hiperparametrów

Liczba Gaussianów

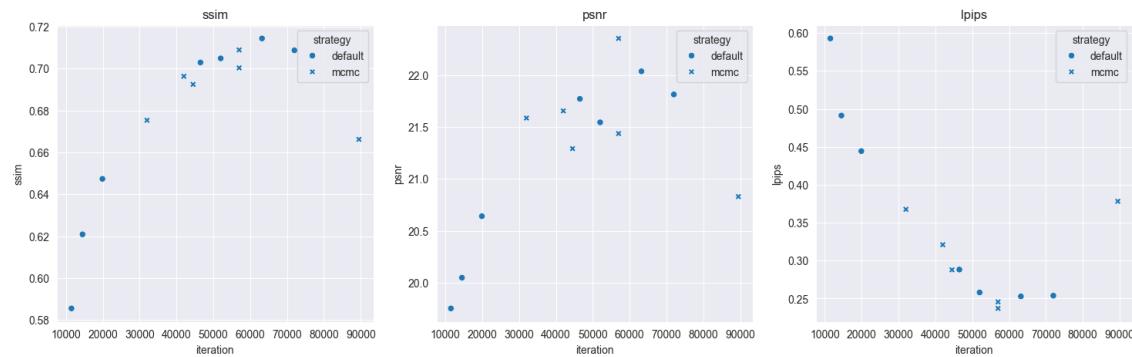


Rysunek 6: Zależność wartości metryk od liczby Gaussianów

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

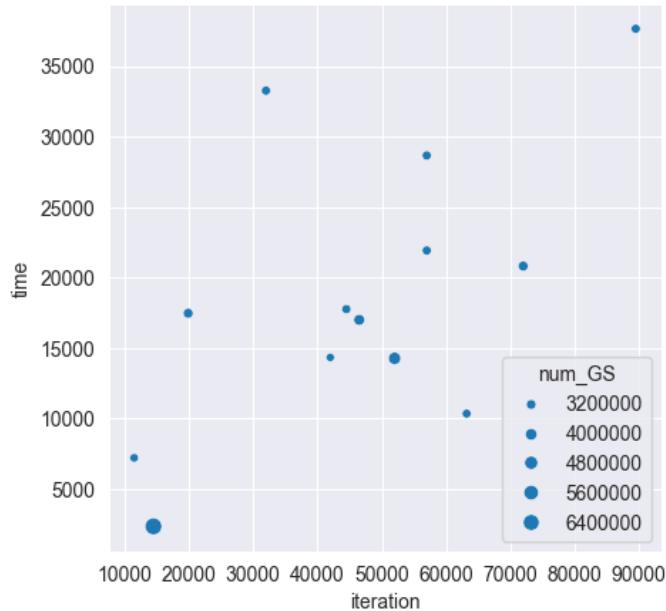
1. default - strategia ta nie ma górnego ograniczenia na liczbę Gaussianów, co czasem może komplikować proces uczenia
2. mcmc - strategia ta ma ograniczenie na liczbę Gaussianów, co może jednak sprawiać trudność dla użytkownika

Liczba iteracji



Rysunek 7: Zależność wartości metryk od liczby iteracji

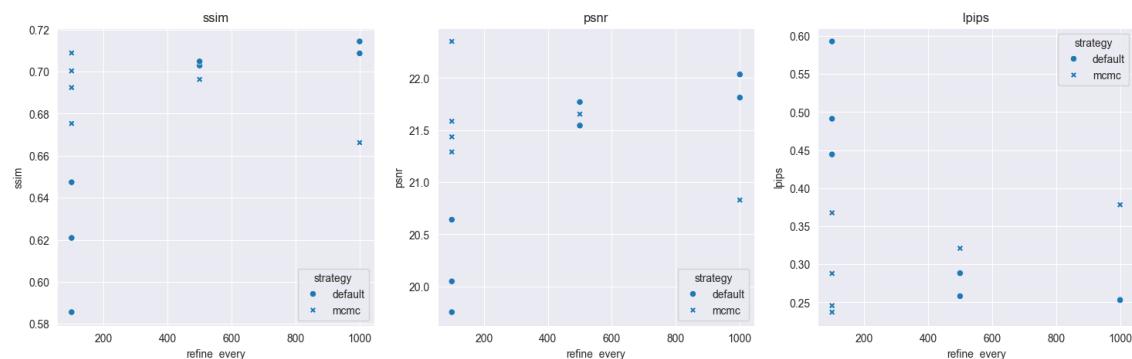
W przypadku liczby iteracji widać trend polepszania się metryk wraz z większą ilością powtórzeń trenowania.



Rysunek 8: Czas trenowania w zależności od liczby iteracji

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

Częstość adaptacji



Rysunek 9: Zależność wartości metryk od częstości adaptacji

Powyzsze 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. Ich zachowanie zależy od wybranej strategii:

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 Prawie wszystkie eksperymenty zakładały wartość 3, co daje najlepszą jakość zdjęć, jednak za cenę zwiększonego zużycia pamięci i wydłużonego czasu trenowania.

4.1.3 Porównanie zdjęć między eksperymentami



Rysunek 10: Eksperyment $i = 5$ z najlepszymi metrykami



Rysunek 11: Eksperyment $i = 1$ z najgorszymi metrykami

Pierwsze zdjęcie z dobrą jakością pokazuje szczegóły jak np. krzesła, na drugim zdjęciu widoczne są artefakty spowodowane niewystarczającym dotrenowaniem gaussianów.

4.1.4 Wyniki dla pozostałych scen



img/res_imgs/rendered_c5_0001.png

Rysunek 12: Wyrenderowany C5

4.1.5 Podsumowanie

Eksperymenty zostały przeprowadzone na jednej scenie ze względu na ograniczenia czasowe, jednak powyższe wnioski powinny się tyczyć również 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

References

- [1] Bernhard Kerbl et al. “3D Gaussian Splatting for Real-Time Radiance Field Rendering”. In: *ACM Transactions on Graphics* 42.4 (July 2023). URL: <https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/>.
- [2] Ben Mildenhall et al. “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis”. In: (2020). arXiv: 2003.08934 [cs.CV]. URL: <https://arxiv.org/abs/2003.08934>.
- [3] Charalambos Poullis and Suya You. “3D Reconstruction of Urban Areas”. In: (2011), pp. 33–40. doi: [10.1109/3DIMPVT.2011.14](https://doi.org/10.1109/3DIMPVT.2011.14).
- [4] Charles R. Qi et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: (2017). arXiv: 1612.00593 [cs.CV]. URL: <https://arxiv.org/abs/1612.00593>.
- [5] Vickie Ye et al. “gsplat: An Open-Source Library for Gaussian Splatting”. In: *arXiv preprint arXiv:2409.06765* (2024). arXiv: 2409.06765 [cs.CV]. URL: <https://arxiv.org/abs/2409.06765>.



img/res_imgs/img_0001.png

Rysunek 13: Budynek C5



img/res_imgs/rendered_c7_12.png

Rysunek 14: Wyrenderowany C7



Rysunek 15: Budynek C7