SIGK - Projekt 4
Rendering

Autor: Łukasz Dąbała

1 Wymagania projektu

W ramach projektu należy stworzyć program, który będzie realizował opisane w temacie funkcje. Projekt jest zadaniem zespołowym, gdzie każdy zespół składa się z 2 osób.

Głównym językiem programowania powinien być język Python wraz z frameworkiem przeznaczonym do sieci neuronowych: Pytorch.

Za projekt można uzyskać maksymalnie $x \times 10p.$, gdzie x to liczba osób w zespole. Każdy z członków zespołu może dostać maksymalnie 10 punktów.

Ocenie w ramach projektu podlegają:

1. Działanie programu - realizacja funkcji (7 p.)
2. Dokumentacja dokonanych eksperymentów oraz wizualizacja wyników (3 p.)

Projekt uznaje się za oddany w momencie prezentacji go prowadzącemu.

2 Rendering neuralny

W ramach projektu trzeba stworzyć rozwiązanie oparte o sieć typu GAN lub model dyfuzyjny, którego zadaniem będzie realizacja modelu oświetlenia Phonga dla zadanej sceny. Zadana sieć powinna być spięta z dostarczonym rendererem.

Należy również porównać generowane wyniki z referencją korzystając z metryki FLIP[1] (https://github.com/NVlabs/flip).

2.1 Model oświetlenia Phonga

Model oświetlenia Phonga składa się z 3 składowych:

1. otoczenia - odpowiedzialnego za czynnik związany ze światłem pośrednim (kompensacja globalnego oświetlenia)
2. rozproszenia - odpowiedzialnego za właściwości matowe obiektu
3. odbicia - odpowiedzialnego za rozbłyski na powierzchni.

.

Czynnik otoczenia Realizuje równanie $I = k_a I_a$, gdzie:

1. k_a to współczynnik materiału obiektu odpowiedzialny za kolor pochodzący od otoczenia
2. I_a to kolor światła otoczenia.

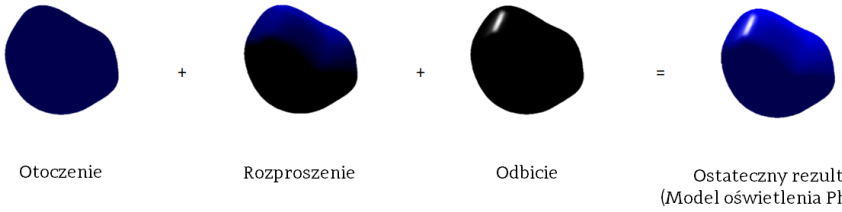
Czynnik rozproszenia Realizuje równanie $I = k_d(\vec{L} \cdot N)\vec{I}_d$, gdzie:

1. k_d to współczynnik materiału odpowiedzialny za kolor pochodzący od właściwości rozproszenia,
2. $\vec{L}$ to wektor w kierunku źródła światła,
3. $\vec{N}$ to wektor normalny do powierzchni,
4. I_d to kolor światła dla rozproszenia.

Czynnik odbicia Realizuje równanie $I = k_s(\vec{V} \cdot \vec{R})^n I_s$, gdzie:

1. k_s to współczynnik materiału odpowiedzialny za kolor odbicia
2. $\vec{V}$ to wektor od powierzchni w kierunku obserwatora
3. $\vec{R}$ to wektor światła odbitego od powierzchni
4. n - współczynnik połyskliwości
5. I_s - to kolor światła dla odbicia

Przykładowe złożenie czynników oraz ostateczny rezultat można zobaczyć na rysunku 1.



Rysunek 1: Przykładowe złożenie dla modelu oświetlenia Phonga

2.2 Parametry sceny

Żeby zadanie było możliwe do zrealizowania w ograniczonym czasie, ograniczamy też scenę i jej parametry. W związku z tym:

1. Zakładamy, że na scenie znajduje się jeden obiekt. Dla naszej sceny będzie to siatka kuli.
2. Na scenie znajduje się jedno punktowe źródło światła.
3. Kamera umieszczona jest w stałym miejscu.

2.3 Generacja zbioru danych

W celu realizacji zadania, należy wcześniej wygenerować zbiór danych, dzięki którym będzie możliwe wytrenowanie modelu.

Korzystając z dostarczonego kodu renderera, chcemy wygenerować serię obrazów ze sceny z konkretnymi parametrami:

1. Właściwości obiektu
   (a) każda ze składowych położenia obiektu powinna być losowana z przedziału $< -20, 20 >$.
   (b) każda z 3 składowych koloru rozproszenia powinna być losowana z przedziału $< 0, 255 >$.
   (c) współczynnik połyskliwości powinien być losowany z przedziału $[3, 20]$.
   (d) współczynnikiem dla otoczenia jest stały kolor $[76, 76, 76]$
   (e) współczynnikiem dla odbicia jest stały kolor $[255, 255, 255]$.
   (f) zakładamy, że nie skalujemy oraz nie obracamy obiektem
2. Właściwości kamery - wykorzystujemy dane kamery zapisane w programie. W celu chęci utrudnienia sobie zadania, można również modyfikować właściwości kamery tzn. położenie, kierunek patrzenia oraz field of view.
3. Właściwości światła

(a) każda ze składowych położenia światła powinna być losowana z przedziału $< -20, 20 >$.

(b) światło ma stały kolor dla wszystkich 3 składowych (otoczenia, rozproszenia i odbicia) odpowiednio: $[25, 25, 25]$, $[255, 255, 255]$, $[255, 255, 255]$

Generowane obrazki powinny być wielkości 128×128.

Przed przystąpieniem do trenowania rozwiązania należy zadać sobie pytanie, czy nie warto zapisać pewnych parametrów w inny sposób niż bezpośredni (być może np. relatywne zapisanie wartości położenia ułatwi modelowi zadanie)?

2.4 Kod

Parę wskazówek do kodu:

1. W kodzie należy dodać wylosować wartości w funkcji **on_render** w klasie **PhongWindow**.
2. Jeżeli chcemy zmienić ścieżkę wyjściową dla plików, można to zrobić to modyfikując typ wyliczeniowy w pliku **main.py** i parametr –**output_path**.
3. Jeżeli chcemy zmienić wielkość okna i jednocześnie wielkość zapisywanych obrazów, najwygodniej jest to zrobić zmieniając parametr **WINDOW_SIZE**.

Literatura

[1] Pontus Andersson, Jim Nilsson, and Tomas Akenine-Möller. Visualizing and Communicating Errors in Rendered Images. In Adam Marrs, Peter Shirley, and Ingo Wald, editors, *Ray Tracing Gems II*, chapter 19, pages 301–320. 2021.