Logo

Description automatically generated

**Damian Andrysiak**

Kierunek: informatyka

Specjalność: informatyka stosowana

Numer albumu: 374341

**Badanie efektywności zwiększania rozdzielczości obrazów za pomocą sieci neuronowych**

Praca magisterska

wykonana pod kierunkiem

dr. hab. prof. nadzw. UŁ.

Pawła Kowalczyka

w Katedrze Fizyki Ciała Stałego

WFiIS UŁ

**Łódź 2023**

# Spis treści

[Spis treści 2](#_Toc134127540)

[1. Wstęp 3](#_Toc134127541)

[2. Zbiór danych 5](#_Toc134127542)

[2.1. Proces generowania danych 5](#_Toc134127543)

[2.2. Przygotowanie silnika do generowania danych 9](#_Toc134127544)

[2.3. Przygotowanie projektu do generowania danych 10](#_Toc134127545)

[2.4. Uruchomienie procesu generowania danych 11](#_Toc134127546)

[2.5. Normalizacja danych 12](#_Toc134127547)

[3. Podsumowanie i bibliografia 12](#_Toc134127548)

[3.1. Podsumowanie 12](#_Toc134127549)

[3.2. Bibliografia 12](#_Toc134127550)

# Wstęp

Podstawowym celem pracy pt. „Badanie efektywności zwiększania rozdzielczości obrazów z wykorzystaniem sieci neuronowych” będzie zbadanie możliwości powiększania obrazów HDR z rozdzielczości 1080p do rozdzielczości 4k (tj. 3840x2160 pikseli) z wykorzystaniem sieci neuronowych.

Na samym wstępie warto wspomnieć o technice, dzięki której można uzyskać zwiększenie rozdzielczości zdjęć, jest nią SISR (eng. Single Image Super Resolution - Uzyskanie Super Rozdzielczości z Pojedynczego Obrazu). Metoda ta wykorzystuje głębokie sieci konwolucyjne do uzyskania lepszej jakości oraz rozdzielczości z obrazu o niższej rozdzielczości w odróżnieniu od metod czysto analitycznych takich jak interpolacja najbliższych sąsiadów (eng. Nearest-neighbors), bilionowa (eng. Bilinear) lub bikubiczna (eng. Bicubic). Sieci neuronowe są w stanie nauczyć się bardziej złożonych zależności miedzy pikselami przez co generowane obrazy są wyższej jakości i zawierają więcej szczegółów. Ich popularność rośnie z roku na rok, miedzy innymi dzięki ich możliwościom adaptacji do wielu problemów, dostępności dużej ilości wytrenowanych ówcześnie modeli, które są gotowe do użycia oraz coraz większej ilości API (eng. Application Programming Interface). Umożliwiają one tworzenie własnych modeli w oparciu o gotowe dokumentacje, przystępne nawet dla osób nieskorelowanych ze sztuczna inteligencja.

W opisywanej pracy magisterskiej została użyta zmodyfikowana architektura głębokiej sieci neuronowej typu UNET[1]. Składa się z ona trzech głównych bloków, jakimi są kolejno blok zmniejszający rozdzielczość (eng. Downsample Block/Encoder), który jest odpowiedzialny za ekstrakcję cech oraz zmniejszanie rozdzielczości, tudzież kompresję obrazu, pozwalająca na wydobywanie szczegółów, następnie występuje wąskie gardło (eng. Bottleneck), który odpowiada za ostateczną ekstrakcję cech, a ostatnim blokiem jest blok zwiększający rozdzielczość (eng. Upsample Block/Decoder), przynajmniej do rozdzielczości wejściowej obrazu. Cześć pracy, związana bezpośrednio z siecią neuronowa została napisana z użyciem biblioteki PyTorch.

Kolejnym ważnym aspektem w pracy jest proces generowania danych. Do tego celu został wykorzystany silnik graficzny o nazwie Unreal Engine w wersji 4.26.2, który został poddany modyfikacjom umożliwiającym zapisywanie na dysk pojedynczych klatek generowanych podczas używania sekwencji w danym projekcie.

Unreal Engine jest to jeden z najbardziej, jak nie najbardziej popularny silnik graficzny służący do generowania wysokiej jakości danych oraz tworzenia gier komputerowych.

Opisywana praca magisterska została podzielona na dwa podstawowe pod problemy: Silnik graficzny umożliwiający generowanie zbioru danych oraz sieć neuronowa wraz z całokształtem procesu uczenia oraz walidacji wyników. Oba te procesy zostały opisane ze szczegółami w rozdziale nr 2.

# Zbiór danych

## Proces generowania danych

Proces generowania danych jest ścisłe związany z wspomnianym w rozdziale 1 silnikiem o nazwie Unreal Engine w wersji 4.26.2. Silnik ten posiada bardzo rozbudowany proces generowania obrazów, który z wersji na wersje zaczyna być coraz trudniej rozróżnialny, kiedy to porównamy obrazy wygenerowane przez silnik a rzeczywiste zdjęcia. Na przykładzie [Rysunek 2.1] możemy wyróżnić sześć podstawowych potoków graficznych (eng. Graphic pipelines/passes), które maja miejsce podczas generowania pojedynczej klatki (wykluczając dwa ostatnie potoki tj. eng. User Interface oraz eng. Backbuffer Resolution):

1. **Geometria widoku (eng. View Geometry)** jest to proces w potoku graficznych, który służy do przygotowania macierzy projekcji, translacji itp. Dla aktualnej ramki/klatki oraz:
   1. **Tworzenia geometrii/bufora głębokości (eng. Depth Buffer)** odpowiedzialnego za informacje o odległości położenia obiektu od kamery w wartości składowych X,Y oraz Z sceny trójwymiarowej.
   2. **Tworzenia geometrii/bufora prędkości/ruchu (eng. Velocity Buffer)** odpowiedzialnego za informacje o poruszaniu się obiektu w wartościach X,Y oraz Z sceny trójwymiarowej.
2. **Głębokość pola widoku (eng. Depth of Field)** jest odpowiedzialny za efekt rozmycia obiektów na scenie bazując na wartościach z bufora głębokości. Obiekty, na których obecnie gracz nie skupia swojej uwagi, staja się rozmyte, jak pokazano na [Rysunek 2.2].
3. **Metoda anty-aliasingowa (eng. Anti-Aliasing Method)** jest odpowiedzialna za wygładzanie krawędzi, jak pokazano na [Rysunek 2.3] przed wygładzeniem krawędzi oraz po ich wygładzeniu [Rysunek 2.4]. Najbardziej popularne metody Anty-Aliasingowe to FXAA (eng. Fast Aproximate Anti-Aliasing), MSAA (eng. Multi-Sampling Anti-Aliasing), TAA[2] (eng. Temporal Anti-Aliasing) oraz SSAA (Super-Sampling Anti-Aliasing).
4. **Metoda rozmywania ruchu kamery (eng. Motion Blur)** polega na dodaniu do sceny efektu rozmycia podczas ruchu kamery [Rysunek 2.5].
5. **Efekt rozmycia światła (eng. Bloom)** jest stosowany do uzyskania efektu rozmycia/poświaty ze źródła światła [Rysunek 2.6].
6. **Mapowanie tonów (eng. Tonemapping/Tonemapper)** służy do mapowania wysokiego zakresu kolorow (eng. HDR – High Dynamic Range) do niskiego zakresu kolorów (eng. LDR – Low Dynamic Range).

Table, timeline

Description automatically generatedRysunek 2.1 Przykładowy wygląd procesów graficznych potrzebnych do wygenerowania pojedynczej klatki w silniku **Unreal Engine**.

A picture containing indoor, engine

Description automatically generatedRysunek 2.2 Przykładowy wygląd sceny po aplikacji procesu głębokości pola widoku/sceny (eng. **Depth of Field**), aktualny obiekt (A), na którym skupia się gracz pozostaje w wysokiej ostrości, reszta obiektów (np. B), staje się rozmyta.Chart, line chart

Description automatically generated

Rysunek 2.3 Wygląd krawędzi przed ich wygładzeniem metodą **Anty-Aliasingowa**.

Chart, line chart

Description automatically generated

Rysunek 2.4 Wygląd krawędzi po ich wygładzeniu metodą **Anty-Aliasingowa**.

A picture containing indoor, music

Description automatically generated

Rysunek 2.5 Ramka po aplikacji efektu rozmycia zależnego od ruchu kamery.





Rysunek 2.6 **Obrazek A** przedstawia scenę przed efektem rozmycia światła, a **Obrazek B** tuz po jego zastosowaniu.

## Przygotowanie silnika do generowania danych

Pierwszym krokiem w procesie przygotowania (modyfikacji) silnika Unreal Engine był wybór właściwego miejsca do pobierania wygenerowanych klatek. W celu uzyskania zbioru danych o jak najwyższej możliwej jakości, użytego do trenowania sieci neuronowej w opisywanej pracy, dane zostały pobrane z wejścia oraz wyjścia Temporalnego Anty-Aliasingu (eng. Temporal Anti-Aliasing)[Rysunek 2.7].

Następnym krokiem było dopisanie instrukcji w języku C++, które umożliwiły pobieranie danych z określonego miejsca i zapisywanie ich na dysk podczas pracy silnika.

Diagram

Description automatically generated

Rysunek 2.7 **Scene Color TAA Input** są to dane wejściowe do sieci neuronowej, a **Scene Color TAA Output** jest celem sieci.

## Przygotowanie projektu do generowania danych

W celu poprawnego pobierania danych z silnika, ważne jest przygotowanie projektu z którego będą pobierane dane (klatki). Odbywa się to za pomocą pobrania projektu ze sklepu Epic (firma, która stworzyła Unreal Engine) oraz dodaniu do Blueprint projektu (język wysoko poziomy stosowany w Unreal Editor) kolejno kilku elementów:

1. **Event BeginPlay** służy do uruchomienia dalszych elementów zaraz po uruchomieniu sceny.
2. **Execute Console Command** służy do wykonania danej komendy. Komenda “**r.BufferVisualizationDumpFramesAsHDR 1”** określa, ze scena będzie generowana w szerokim zakresie barw tj. HDR (eng. High Dynamic Range) **.**
3. **Execute Console Command** służy do wykonania danej komendy. Komenda “**r.BufferVisualizationDumpFramesAsHDR 1”** określa, ze scena będzie generowana w szerokim zakresie barw tj. HDR (eng. High Dynamic Range) **.**
4. **Delay** jest odpowiedzialny za pauzę sekwencji w celu wyeliminowania np. niepoprawnie załadowanych tekstur.
5. **Play** uruchamia sekwencję (ówcześnie nagrany film, animacje itp.).
6. **Execute Console Command** posiadający komendę **“r.DumpPass.Allow 1”**, która uruchamia proces pobierania danych z silnika i zapisywania ich na dysk.

A screenshot of a video game

Description automatically generated

Rysunek 2.8 Przykładowy wygląd poprawnej sekwencji elementów w Blueprints.

## Uruchomienie procesu generowania danych

Proces generowania danych rozpoczyna się w momencie wywołania pliku wsadowego w formacie .bat oraz ścieżki do projektu z którego chcemy pobierać dane.

**> plik\_wsadowy.bat NazwaProjektu1/NazwaProjektu2.uproject**

Przykładowe wywołanie procesu generacji danych zostało przedstawione na [Rysunek 2.9].



Rysunek 2.9 Przykładowe wywołanie procesu generowania danych w wierszu poleceń.

## Wygenerowane dane

Wygenerowane dane są zapisane na dysku w formacie OpenEXR [3]

# Podsumowanie i bibliografia

## Podsumowanie

ADDDASD

## Bibliografia

[1] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015. doi: 10.1007/978-3-319-24574-4\_28.

[2] L. Yang, S. Liu, and M. Salvi, “Number 2 STAR-State of The Art Report,” 2020. [Online]. Available: http://onlinelibrary.wiley.com.Thisarticlemaybeusedfornon-commercialpurposesinaccordancewiththeWileySelf-ArchivingPolicy[http://olabout.wiley.com/WileyCDA/Section/id-820227.html].

[3] Academy Software Foundation, “OpenEXR,” 2003. https://openexr.com/en/latest/ (accessed May 05, 2023).