

Lucas da Silva dos Santos
Matheus Zanivan Andrade
Rafael Nascimento Lourenço

Geração Procedural de Mapas para Jogos através da Segmentação de Imagens por Rede Neural Convolucional

São Paulo - Brasil

2023

Lucas da Silva dos Santos
Matheus Zanivan Andrade
Rafael Nascimento Lourenço

Geração Procedural de Mapas para Jogos através da Segmentação de Imagens por Rede Neural Convolucional

Monografia apresentada na disciplina Trabalho de Conclusão de Curso, como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação.

Centro Universitário Senac - Santo Amaro
Bacharelado em Ciência da Computação

Orientador: Thyago Conchado Quintas

São Paulo - Brasil
2023

Lucas da Silva dos Santos
Matheus Zanivan Andrade
Rafael Nascimento Lourenço

Geração Procedural de Mapas para Jogos através da Segmentação de Imagens por Rede Neural Convolucional

Monografia apresentada na disciplina Trabalho de Conclusão de Curso, como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação.

Thyago Conchado Quintas
Orientador

Professor
Convidado 1

Professor
Convidado 2

São Paulo - Brasil
2023

*Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.*

Resumo

Esta monografia descreve o desenvolvimento de uma ferramenta para jogos que oferece uma nova funcionalidade. A ferramenta começa com a seleção de uma foto, que é processada por um modelo de rede neural convolucional especializado em segmentação panóptica. Isso permite a segmentação da imagem, incluindo a separação de objetos da mesma classe, como pessoas e carros. Após o modelo gerar a imagem de saída, será possível selecionar um contorno detectado e, a partir disso, gerar um mapa de forma procedural, combinado com o diagrama de Voronoi para criar os biomas do mapa. Além disso, será implementada uma automação no motor gráfico, incorporando um mapa jogável tridimensional e um minimapa bidimensional para geolocalização.

Palavras-chaves: segmentação panóptica, geração procedural, diagrama de Voronoi, mapas, jogos.

Abstract

This monograph describes the development of a tool for games that offers new functionality. The tool starts with the selection of a photo, which is processed by a convolutional neural network model specialized in panoptic segmentation. This allows for the segmentation of the image, including the separation of objects of the same class, such as people and cars. After the model generates the output image, it will be possible to select a detected outline and, from that, generate a procedural shape map, combined with the Voronoi diagram to create the biomes of the map. Furthermore, an automation will be implemented in the graphics engine, incorporating a playable three-dimensional map and a two-dimensional minimap for geolocation.

Key-words: panoptic segmentation, procedural generation, Voronoi diagram, maps, games.

Listas de ilustrações

Figura 1 – número de jogos publicados na Steam.	17
Figura 2 – Diagrama de Voronoi.	22
Figura 3 – Diagrama de Voronoi separado em solo e mar	23
Figura 4 – Diagrama de Voronoi separado em solo e mar com os cantos dos polígonos indicando a direção para o litoral	24
Figura 5 – Diagrama de Whittaker	25
Figura 6 – Resultado final da geração do mapa	25
Figura 7 – Algoritmos de detecção facial e de roupas/cabelos por cor localizam e reconhecem pessoas nesta imagem	26
Figura 8 – Segmentação de instâncias de objetos pode-se delinear cada pessoa e objeto em uma cena complexa	26
Figura 9 – Diagrama de dados de píxeis. À esquerda, uma imagem de Lincoln; no centro, os píxeis rotulados com números de 0 a 255, representando sua luminosidade; e à direita, apenas esses números.	27
Figura 10 – Um robô futurista com design elegante e moderno, sentado em uma cadeira enquanto lê um livro sobre inteligência artificial. O robô tem um olhar pensativo e curioso enquanto aprende sobre o assunto	29
Figura 11 – Ilustração da relação entre os principais tópicos de aprendizado de máquina	30
Figura 12 – Modelo de um neurônio não-linear.	30
Figura 13 – Gráfico da função <i>Sigmoid</i> .	31
Figura 14 – Gráfico da função <i>ReLU</i> .	32
Figura 15 – Gráfico da função <i>ReLU</i> .	32
Figura 16 – Gráfico da função <i>Softmax</i> .	33
Figura 17 – Gráficos mostrando subajuste, balanceado e sobreajuste respectivamente	34
Figura 18 – Comparação de uma rede neural convencional com uma rede neural profunda.	35
Figura 19 – Camadas principais de uma rede neural convolucional	35
Figura 20 – Tipos de segmentação em redes neurais convolucionais	40
Figura 21 – Exemplo de arquitetura de rede totalmente convolucional	40
Figura 22 – Arquitetura codificador-decodificador UNet	41
Figura 23 – Exemplo da classificação dos conjuntos usados nas métricas de segmentação	42
Figura 24 – Arquitetura geral do EfficientPS	46
Figura 25 – Etapas da proposta	52
Figura 26 – Resultado inicial do EfficientPS.	53
Figura 27 – Expectativa do resultado do EfficientPS.	53

Figura 28 – Resultado do EfficientPS obtido seguindo os passos da publicação no repositório.	54
Figura 29 – Passos da seleção da saída da inteligência artificial.	55
Figura 30 – Passos da seleção da saída da inteligência artificial mais performático. .	56
Figura 31 – Ilustração do cálculo para definir a localização dos vértices.	57
Figura 32 – Ilustração do diagrama de Voronoi.	57
Figura 33 – Exemplo de mapa de altura.	60
Figura 34 – Resultado no Unity sem usar filtro de desfoque no mapa de altura. . .	62
Figura 35 – Resultado no Unity usando filtro de desfoque no mapa de altura. . .	62
Figura 36 – Ilustração da classificação de conjuntos entre a imagem de entrada e o mapa de altura	63
Figura 37 – Resultados da última execução de cada combinação de imagens. . . .	66
Figura 38 – Passos do programa final.	67

Lista de tabelas

Tabela 1 – Top 15 modelos que melhor classificam pessoas com métrica P em segmentação panóptica	45
Tabela 2 – Relação entre umidade e elevação para biomas	59
Tabela 3 – Resultados dos testes entre imagem de entrada e mapa de altura	62
Tabela 4 – Resultados dos testes entre imagem de entrada e output3d	64
Tabela 5 – Resultados dos testes entre mapa 2d e mapa de altura	64
Tabela 6 – Resultados dos testes entre imagem de entrada e mapa 2d	65
Tabela 7 – Resultados dos testes entre imagem de entrada e mapa de altura	65
Tabela 8 – Resultados dos testes entre mapa 2d e mapa de altura	65

Lista de abreviaturas e siglas

IA	Inteligência Artificial
RGB	Red, Green and Blue ou Vermelho, Verde e azul
RNC	Rede Neural Convolucional
CNN	Convolutional Neural Network
RTC	Rede Totalmente Convolucional
RoI	Region of Interest ou Região de interesse
IoU	Intersection over Union ou União sobre intersecção
RPC	Pirâmide de Características
ECLE	Extrator de Características em Larga Escala
RPR	Rede de Proposta de Região

Sumário

1	INTRODUÇÃO	17
1.1	Objetivos	19
2	FUNDAMENTAÇÃO TEÓRICA	21
2.1	Geração procedural de conteúdo	21
2.1.1	Diagrama de Voronoi	21
2.1.2	Geração de biomas no diagrama de Voronoi	22
2.1.3	Representação	24
2.2	Visão computacional	26
2.3	Inteligência Artificial	28
2.3.1	Aprendizado de Máquina	29
2.3.1.1	Rede neural artificial	29
2.3.1.1.1	Redes neurais convolucionais	34
2.4	Trabalhos relacionados	48
3	DESENVOLVIMENTO	51
3.1	Proposta	51
3.2	Implementação	52
3.2.1	Segmentar imagens	52
3.2.2	Selecionar contorno	54
3.2.3	Geração procedural do mapa	55
3.2.3.1	Ilha gerada no contorno	55
3.2.3.2	Unity - Mapa 3d	58
3.2.3.2.1	Terreno	60
3.2.3.2.2	Minimap	60
3.2.3.2.3	Jogabilidade	61
3.2.4	Testes	61
3.2.5	Pós processamento	61
3.2.6	Interface gráfica	63
4	RESULTADOS	65
4.1	Apresentação	65
4.2	Analise dos resultados	66
5	CONSIDERAÇÕES FINAIS	69
5.1	Conclusão	69

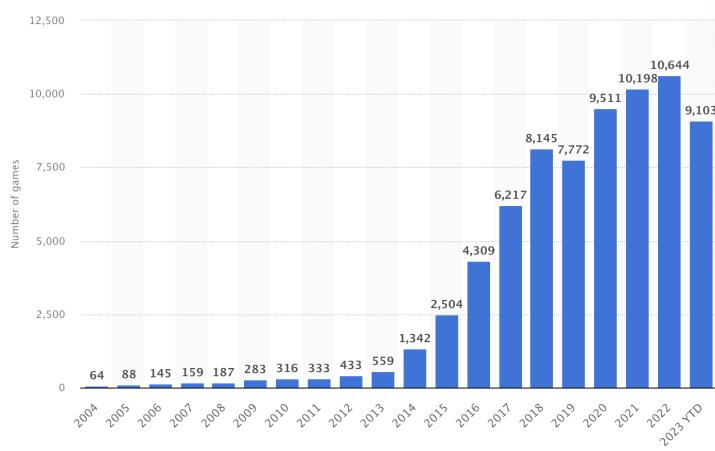
5.2	Trabalhos futuros	69
-----	--------------------------	----

REFERÊNCIAS	71
--------------------	----

1 Introdução

A indústria de jogos digitais cresce cada vez mais. De acordo com Santana (2022), essa indústria tende a ultrapassar em 2023, os US\$ 200 bilhões (aproximadamente, R\$ 1 trilhão). Novos jogos são produzidos e publicados diariamente, e somente na plataforma digital Steam, foram 10.644 novos títulos em 2022 como podemos ver na Figura 1

Figura 1 – número de jogos publicados na Steam.



Fonte: Clement (2023)

No cenário de jogos, os mapas desempenham um papel fundamental, fornecendo orientação aos jogadores e criando a sensação de escala em uma área. Por exemplo o jogo de aventura pirata chamado Sea of Thieves, os mapas revelam locais de interesse, como tesouros escondidos, missões e áreas perigosas, além de ajudar os jogadores a planejar suas estratégias, explorar o mundo virtual e tomar decisões com base em informações espaciais. Portanto os mapas enriquecem a experiência geral do jogo, mas cria-los pode ser um desafio, especialmente levando em consideração o orçamento disponível. Pois demandaria muitos recursos criar vários mapas diferentes com intuito de entretenimento do jogador. Em jogos como Minecraft, um elemento importante é a geração procedural, que consiste em um conjunto de algoritmos e ferramentas para geração de conteúdo, no qual se cria os mundos, com ilhas contendo biomas, cavernas, vilas, dentre outros recursos. Com essa diversidade de características pode-se evitar o tédio de sempre jogar no mesmo mapa (W!N, 2023; FOFFANO, 2020).

Contextualizando, a área de Geometria Computacional é um ramo da ciência da computação que estuda algoritmos e estruturas de dados, servindo para resolução computacional de problemas geométricos. O diagrama de Voronoi é um dos tópicos mais discutidos dessa área e possui uma gama de utilizações, dentre elas pode ser utilizado

para resolver alguns problemas relacionados a jogos como por exemplo marcar pontos no mapa, desses pontos criar regiões, e a partir dessas regiões criar biomas gerando um mapa ([RODRIGUES, 2019](#)).

De acordo com [Lisboa \(2022\)](#) é muito comum usar técnicas procedurais combinado com Inteligência Artificial (IA) para melhorar ou personalizar a experiência do jogador. Por exemplo, o jogo RimWorld é um simulador de colônia que gera um planeta de forma procedural e utiliza uma IA para narrar a história, abrangendo psicologia, ecologia, combate e diplomacia, dentre outros. Logo, essa combinação entre IA e a geração procedural cria uma jogabilidade única ao jogador.

Em IA, um ramo que está em ascensão é o de segmentação de imagem com redes neurais convolucionais, que constitui-se em classificar os pixels de uma imagem ou criar áreas na imagem para destacar cada objeto (todas classes que são contáveis como pessoas, carros, etc) detectado ou até mesmo mesclar essas duas técnicas. Neste ramo existem diversas aplicações, como por exemplo carros autônomos e sistemas de vigilância. Na aplicação de carros autônomos é necessário identificar humanos para tomar decisões de freio, em sistemas de vigilância é necessário identificar para alertar e automatizar o processo de segurança. Portanto, nessas aplicações reais observa-se a importância em identificar seres humanos para a tomada de decisões ([ULKU; AKAGÜNDÜZ, 2022](#)).

Com base na contextualização é possível perceber que o mercado de jogos está em ascensão, o mapa é um recurso importante e pode ser usado a técnica de geração procedural para diversificar, o diagrama de Voronoi pode ser usado para gerar biomas em mapas no processo de geração procedural, a técnica de geração procedural de conteúdo unido a inteligência artificial é muito utilizado em jogos para criar personalizações, no ramo de inteligência artificial a segmentação com redes neurais convolucionais está em destaque. Logo pode-se perceber a relevância desses temas no curso de ciência da computação e na atualidade, propõe-se então, uma solução para personalizar mapas de jogos utilizando um modelo de IA da área de segmentação usando redes neurais convolucionais.

Com o objetivo de gerar mapas com biomas de forma procedural com personalização de IA, decidiu-se utilizar o resultado da segmentação de imagem por rede neural convolucional para o usuário selecionar uma área e assim delimitar o contorno da ilha, adicionando, portanto, uma personalização. Essa aplicação possibilita um desenvolvedor de jogos criar um protótipo de mapa rapidamente ou aprimorar e usar esse recurso no jogo, possibilitando o jogador tirar ou selecionar uma foto e escolher um contorno para gerar um mapa com aquele formato.

Por fim, para contribuição científica tem-se a hipótese de que quanto mais pontos o diagrama de Voronoi tiver maior será a precisão da compatibilidade entre o mapa gerado e o contorno escolhido. Para chegar a essa conclusão, comprometeu-se definir alguns testes com métricas em prol de mensurar a qualidade da geração procedural com o contorno

selecionado.

1.1 Objetivos

O objetivo principal deste trabalho é desenvolver uma ferramenta que ofereça uma alternativa para a geração procedural de mapas de ilhas, utilizando o diagrama de Voronoi para a criar biomas. Além disso, pretende-se combinar segmentação com redes neurais convolucionais para permitir a personalização desses mapas. Essa ferramenta terá a capacidade de reconhecer os contornos reconhecidos (classificados no conjunto de dados, logo o resultado terá uma detecção abrangente dentro do escopo de classes obtidas) de uma imagem, e gerar um mapa com um mapa baseado nos limites do contorno escolhido.

Adicionalmente, os seguintes objetivos específicos serão abordados:

- Selecionar e analisar conjuntos de dados contendo classes relevantes, como pessoas, carros, entre outros, para treinar um modelo de rede neural convolucional específico para segmentação de imagens.
- Utilizar algoritmos para criar diagramas de Voronoi.
- Aplicar um algoritmo para reconhecer a imagem com o contorno selecionado e gerar como resultado a imagem do mapa gerado.
- Utilizar o resultado da segmentação para selecionar indicar o que é terreno em cima do diagrama de Voronoi.
- Gerar os biomas no diagrama de Voronoi.
- Criar testes em prol de mensurar a semelhança entre o contorno do mapa gerado com o contorno escolhido.

2 Fundamentação teórica

Este capítulo apresenta os conceitos fundamentais necessários para a realização dos objetivos propostos na monografia. Os tópicos foram organizados na ordem em que são utilizados na ferramenta final. Primeiro, será apresentado o tópico de geração procedural para gerar o mapa requerido, o diagrama de Voronoi para ser aplicado como um filtro na imagem e aplicar os biomas. Em seguida, o conceito geral de visão computacional e por fim, serão apresentados os conceitos de inteligência artificial, tanto em um contexto amplo quanto em relação ao conteúdo proposto, que é a segmentação panóptica.

2.1 Geração procedural de conteúdo

Segundo [Yannakakis e Togelius \(2018\)](#) a geração procedural de conteúdo constitui métodos e automações utilizados para gerar conteúdo. A geração procedural de conteúdo também é uma parte importante da inteligência artificial de um jogo e já vem sendo utilizada desde 1980. Essa técnica pode ser utilizada para gerar níveis, mapas, textura, regras de jogo, história, entre outras coisas.

É difícil dizer qual algoritmo foi utilizado para geração de conteúdo dos jogos modernos e os códigos fontes não são facilmente acessíveis. Já nos jogos antigos os códigos fontes e as estratégias utilizadas são acessíveis e muito bem documentadas na internet. São geralmente utilizados algoritmos de geração aleatória que podem ser classificados como sendo de força bruta, e são usados para criar estruturas ou mapas dependendo do tipo de jogo ([DORMANS, 2010](#)).

2.1.1 Diagrama de Voronoi

Segundo [Rodrigues \(2019\)](#) diagrama de Voronoi é o particionamento do espaço onde cada região é associada a um ponto do conjunto.

O diagrama de Voronoi é gerado a partir das distâncias euclidianas entre os vizinhos de um conjunto de pontos do plano ([SANTOS, 2016](#)). Esse diagrama possui uma gama de utilizações, por exemplo, estudar epidemias, encontrar o ponto mais próximo, calcular a precipitação de uma área, estudar os padrões de crescimento das florestas, etc, ([POLÍGONOS..., 2018](#)).

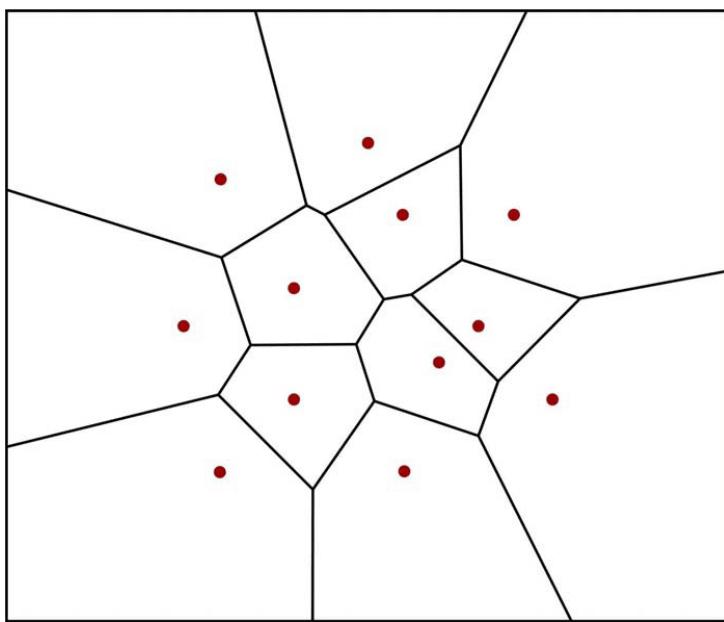
Seja um conjunto de índices $I_n = \{1, 2, 3, \dots, n\}$ e $A = \{p_1, p_2, \dots, p_n\} \subset \mathbb{R}^2$ um conjunto de pontos onde $2 \leq n < \infty$, define-se então como região de Voronoi o conjunto de pontos associado a p_i , onde d é a distância euclidiana

$$V(p_i) = \{p | d(p_i, p) \leq d(p_j, p); i \neq j, i, j \in I_n\}, \quad (2.1)$$

Tem-se um conjunto formado por essas regiões sendo $V(A) = V(1), V(2), \dots, V(n)$ ([RODRIGUES, 2019](#)).

Na figura Figura 2 pode-se ver a relação dos conjuntos de pontos com o diagrama de Voronoi. Contendo os pontos em vermelhos e retas que são perpendiculares a distância dos pontos vermelhos vizinhos.

Figura 2 – Diagrama de Voronoi.



Fonte: [Medeiros \(2013\)](#)

2.1.2 Geração de biomas no diagrama de Voronoi

Biomas são regiões ecológicas que possuem uma fauna e flora com atributos estruturais semelhantes ([BIOMAS..., s. d.](#)). Segundo [Patel \(2010\)](#) gera-se um mapa e seus biomas definindo-se o litoral, os litorais são as bordas que irão separar a água e o solo. Na Figura 3 mostra-se essa separação, sendo os polígonos marrom o solo e os polígonos azuis o mar.

É possível utilizar qualquer formato para gerar as ilhas, como formas geométricas e funções de ruídos como o Perlin Noise ([PATEL, 2010](#)).

A elevação do mapa é definida pelos cantos dos polígonos e é calculada através da distância de todos polígonos indicado como solo até o litoral ([PATEL, 2010](#)). A Figura 4 representa essa elevação, sendo os polígonos azuis o mar, os níveis da elevação são os polígonos de verde(mais baixo) a branco(mais alto) e as setas são as direções das encostas.

Figura 3 – Diagrama de Voronoi separado em solo e mar



Fonte: [Patel \(2010\)](#)

A elevação é utilizada para gerar os biomas. Um exemplo seria elevações altas significativa que é uma montanha, logo ela deve possuir neve. Adicionando mais uma camada, além da elevação, como a de umidade, podemos gerar uma variedade maior de biomas. A umidade é calculada de quão longe o polígono está de um corpo d'água ([PATEL, 2010](#)).

Diagrama de Whittaker

O diagrama de Whittaker é uma forma de dividir os terrenos gerados a partir da técnica de geração procedural. Esse diagrama inclui valores de temperatura e umidade para separar os biomas, exemplificado na Figura 5 ([WHITTAKER..., 2018](#)).

Usando a elevação como representante da temperatura de um bioma, é possível utilizar o diagrama de Whittaker. Fazendo alterações nesse diagrama, é possível adicionar ou remover biomas. Com essa nova camada possibilita a adição de rios ao mapa ([PATEL, 2010](#)) e assim obtendo o resultado apresentado na figura 6.

Figura 4 – Diagrama de Voronoi separado em solo e mar com os cantos dos polígonos indicando a direção para o litoral



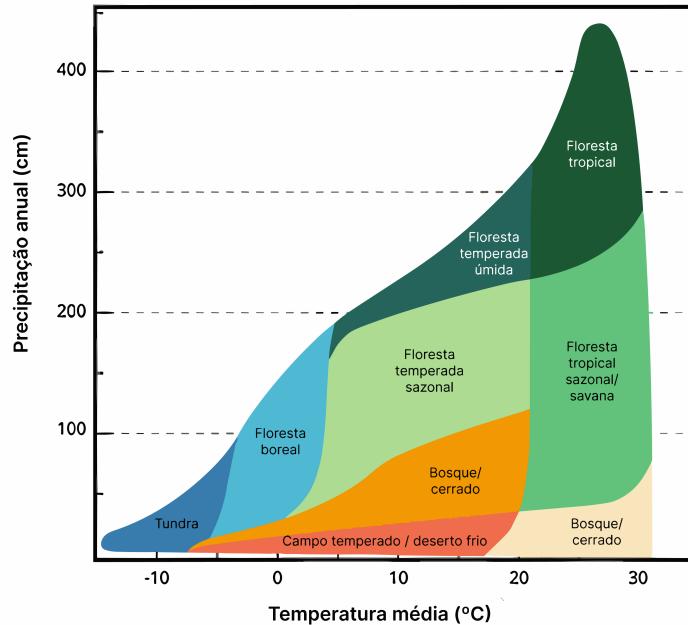
Fonte: [Patel \(2010\)](#)

2.1.3 Representação

Existem duas formas de representar os mapas gerados pela geração procedural, uma delas é a representação 2D que possuem duas dimensões de espaço, não possuem perspectiva e são criadas a partir de figuras geométricas — como por exemplo os polígonos gerados pelo diagrama de Voronoi — e imagens ([TECHNOLOGIES, s. d.](#)). Pode-se utilizar os dados da elevação para representar o mapa gerado, onde cada pixel possui um valor de 0 a 255 definidos pela altura, sendo 0 (preto) a menor altura possível e 255 (branco) a maior altura possível, assim tendo um mapa nomeado de mapa de altura. Com o mapa de altura é possível criar uma representação 3D, segundo [Technologies \(s. d.\)](#) os jogos 3D contem três dimensões de espaço, utiliza-se perspectiva e objetos sólidos com geometria tridimensional.

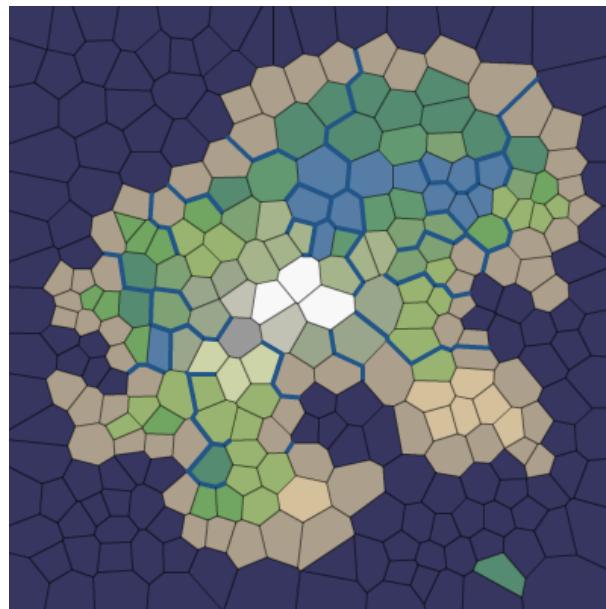
Com a integração de dois mapas distintos, torna-se viável proporcionar uma perspectiva tridimensional em um deles e uma visão superior — conhecida como minimapa — no outro. Esses mapas atuam em conjunto, complementando-se para oferecer ao jogador uma percepção de sua localização. Conforme destacado por [Criando... \(2018\)](#), o Unity se destaca como uma ferramenta versátil para o desenvolvimento de jogos tanto em 2D quanto

Figura 5 – Diagrama de Whittaker



Fonte: Mendes (2019)

Figura 6 – Resultado final da geração do mapa



Fonte: Patel (2010)

em 3D, permitindo a representação conjunta de mapas tridimensionais e bidimensionais.

Para utilizar essa técnica de geração procedural de mapas, é necessário começar com uma imagem que sirva como base para a geração do mapa. É empregada a visão computacional para extrair dados da imagem.

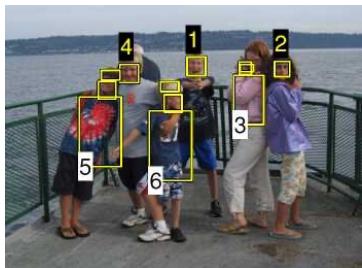
2.2 Visão computacional

A visão computacional está em constante avanço, aproximando cada vez mais os computadores da capacidade visual humana. De acordo com Horst Haußecker e Bernd Jähne, no livro "Computer Vision and Applications" (HAUßECKER BERND JÄHNE, 1999), a visão computacional é uma área da computação que se dedica à interpretação de imagens por meio de algoritmos e técnicas de processamento de imagens. Essa área abrange a aquisição, processamento e análise de imagens, com o objetivo de extrair informações úteis para resolver problemas específicos.

Segundo Richard Szeliski no livro "Computer Vision: Algorithms and Applications" (SZELISKI, 2022), nas últimas décadas ocorreram avanços significativos na busca de aproximar a visão computacional da visão humana, porém não obteve total êxito. Isso ocorre porque, enquanto o olho humano enxerga com aparente facilidade as estruturas tridimensionais e suas nuances, a visão computacional depende de técnicas matemáticas altamente precisas para recuperar a forma tridimensional e a aparência dos objetos.

Nas figuras Figura 7 e Figura 8, evidencia-se a notável capacidade de um computador em distinguir, classificar e até mesmo compreender os elementos presentes em uma fotografia.

Figura 7 – Algoritmos de detecção facial e de roupas/cabelos por cor localizam e reconhecem pessoas nesta imagem



Fonte: [Szeliski \(2022\)](#)

Figura 8 – Segmentação de instâncias de objetos pode-se delinejar cada pessoa e objeto em uma cena complexa



Fonte: [GmbH \(2023\)](#)

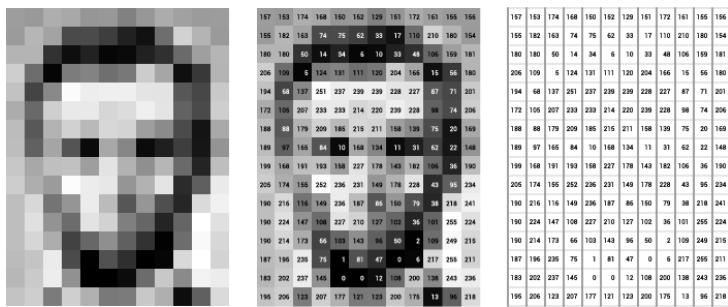
No entanto, apesar do sucesso no uso dessas técnicas, o computador ainda não consegue oferecer a mesma quantidade de detalhes na explicação de uma imagem como o olho humano. Isso se deve à maior facilidade do computador em compreender linguagem em comparação à visualização. A tarefa de ensinar um computador a ver e descrever com precisão e riqueza de detalhes o que está sendo observado é extremamente complexa (SZELISKI, 2022).

A visão é um elemento crucial para capacitar a inteligência artificial a realizar diversas tarefas. A fim de replicar a visão humana, é necessário que as máquinas sejam

capazes de adquirir, processar, analisar e compreender imagens. ([MARR, 2019](#))

No processamento de computação visual, as imagens são adquiridas e representadas como uma matriz 2D de pixels. Cada pixel corresponde a um ponto na imagem e é representado por um valor numérico que varia de 0 a 255. Esses valores de pixel descrevem a intensidade da cor em uma escala de cinza caso a imagem de entrada esteja em preto e branco, pois se a imagem ter cores do espectro RGB o computador identificará três matrizes de canais referente as cores correspondentes. Dessa forma, um computador interpreta uma imagem como uma ou mais matrizes de números, permitindo que seja analisado e compreendido os detalhes visuais presentes na imagem. Um exemplo dessa matriz exemplificado na Figura 9 do presidente dos Estados Unidos, Abraham Lincoln([AMINI, 2023](#)).

Figura 9 – Diagrama de dados de pixels. À esquerda, uma imagem de Lincoln; no centro, os pixels rotulados com números de 0 a 255, representando sua luminosidade; e à direita, apenas esses números.



Fonte: [Babich \(2020\)](#)

Os algoritmos de visão computacional utilizados atualmente são fundamentados em reconhecimento de padrões. O procedimento consiste em treinar computadores por meio de uma vasta quantidade de dados visuais. Os computadores processam imagens, rotulam os objetos nelas contidos e identificam padrões entre esses objetos ([BABICH, 2020](#)).

Esse processo de treinamento e reconhecimento de padrões permite que os computadores identifiquem objetos e compreendam seu contexto visual. Com essa capacidade, o computador consegue realizar tarefas como, por exemplo, reconhecimento facial Figura 7.

Em visão computacional é comum usar algumas técnicas para separar objetos de interesse pois a partir disso é possível aplicar alterações em objetos específicos, e para isso utiliza-se um técnica chamada de máscara binária ([NVIDIA, 2023](#); [EMBARCADOS, 2017](#)).

A máscara binária — ou imagem binária — contém apenas duas cores, geralmente preto e branco, ou valores 0 e 1. Sendo o branco (ou 1) o objeto em destaque e o preto (ou 0) o fundo ([EMBARCADOS, 2017](#)).

Será abordado dois algoritmos com propostas parecidas para selecionar um objeto e destaca-lo em uma imagem binária, sendo eles selecionar imagem por cor e por inundação, ambos utilizando o biblioteca em python OpenCV, uma biblioteca multiplataforma para visão computacional, tendo métodos para auxiliar na manipulação por exemplo de imagens ([OPENCV, 2023a](#)).

O método de selecionar por cor se baseia em pegar a cor específica do clique na imagem e percorrer a imagem comparando a cor alvo com a cor da imagem, caso seja a mesma pinte o mesmo pixel da nova imagem como branco e caso não seja pinte como preto. Uma maneira performática de selecionar por cor é definir um espectro de cores e aplicar uma mascara usando o método *inRange* da biblioteca OpenCV, pode-se escolher apenas uma cor ([OPENCV, 2023c](#)).

O método de selecionar por preenchimento por inundação é um algoritmo de expansão a partir de um pixel, validando se contém a mesma cor. A implementação inicia uma matriz de zeros com tamanho 2 pixels maior do que a imagem original. O clique na imagem será a semente — ou em inglês seed — e a partir disso o algoritmo começa uma expansão para os pixels vizinhos — de cima, baixo, esquerda e direita — caso contenha o mesmo valor de cor pinta de cor branca, e refaz com os pixels marcados anteriormente até não ter mais o pontos brancos para marcar ([OPENCV, 2023b](#)).

Ademais, existem diversas opções de ferramentas que auxiliam na criação de interfaces gráficas para visualização e manipulação de imagens, fornecendo funções e modelos para criar telas, sendo uma das mais utilizadas o PyQt5 ([MCFARLAND, 2023](#)).

O PyQt5 incorpora as bibliotecas Qt em C++, proporcionando recursos para o desenvolvimento multiplataforma. Isso simplifica a criação de interfaces em Python em diversos ambientes, como Windows, Mac, Linux e dispositivos móveis ([LIMITED, 2023](#)).

É bastante comum na área de visão computacional contarmos com o auxílio de modelos de inteligência artificial que capacitam o computador a reconhecer padrões e características nas imagens processadas.

2.3 Inteligência Artificial

Inteligência artificial é uma técnica científica que simula o pensamento humano de forma que possa ser executado em uma máquina, podendo ser utilizada para criar soluções com uma linha de progressão parecida ao raciocínio lógico. Isto permite ao computador reconhecer e interpretar o mundo ao redor com imagens e textos, criando-se uma ampla área de atuação que otimiza tarefas antes só realizadas por seres humanos ([SILVA; MAIRINK, 2019](#)).

A ideia geral de inteligência artificial foi apresentado primordialmente no artigo

de Alan Turing — conhecido como pai da computação — denominado de *Computing Machinery and Intelligence* em 1950, outro conceito apresentado também foi o Teste de Turing, uma série de questionamentos que visa provar se a máquina pode executar um comportamento inteligente semelhante ao ser humano ([BRASIL, 2023](#)).

As aplicações de IA são várias, sendo algumas: controlar estoques de produtos nas empresas tanto na logística interna como externa, dirigir carros de forma autônoma, reconhecimento facial com base em vídeos ou fotos, criar imagens com base em um texto como na Figura 10, uma imagem criada usando plataforma DALL · E e até mesmo classificar em imagens, objetos e/ou píxeis na área de segmentação ([STEFANINI, 2020](#); [RAMESH et al., 2021](#)).

Figura 10 – Um robô futurista com design elegante e moderno, sentado em uma cadeira enquanto lê um livro sobre inteligência artificial. O robô tem um olhar pensativo e curioso enquanto aprende sobre o assunto



Fonte: DALL · E [Ramesh et al. \(2021\)](#)

2.3.1 Aprendizado de Máquina

Segundo [Woschank, Rauch e Zsifkovits \(2020\)](#), aprendizado de máquina é uma subcategoria de inteligência artificial que se refere à detecção de padrões importantes de uma base de dados. As ferramentas utilizadas aumentam a eficiência dos algoritmos para lidar com bases de dados grandes.

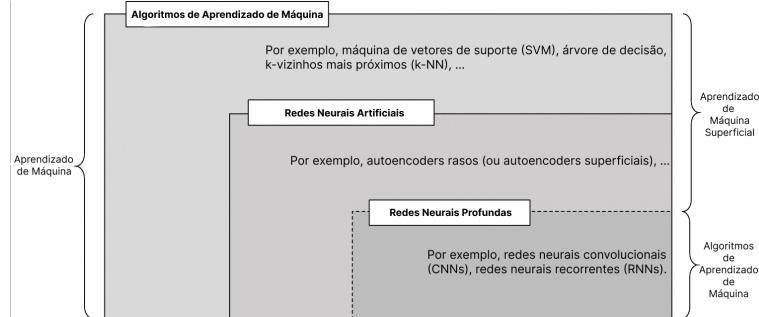
Portanto, essa técnica permite ao computador melhorar os resultados com base na experiência, isso indica uma relação direta entre o quanto o programa consumiu de dados e qualidade da solução do problema ([BROWN, 2021](#)).

Dentro desse nicho existem outros como: redes neurais, algoritmos evolucionários, algoritmos de busca, aprendizado por reforço, dentre outros. ([SIRCAR et al., 2021](#)).

É possível observar uma hierarquia entre aprendizado de máquina e os principais

termos, sendo eles: redes neurais artificiais e aprendizado profundo, ilustrado na Figura 11 (JANIESCH; ZSCHECH; HEINRICH, 2021).

Figura 11 – Ilustração da relação entre os principais tópicos de aprendizado de máquina

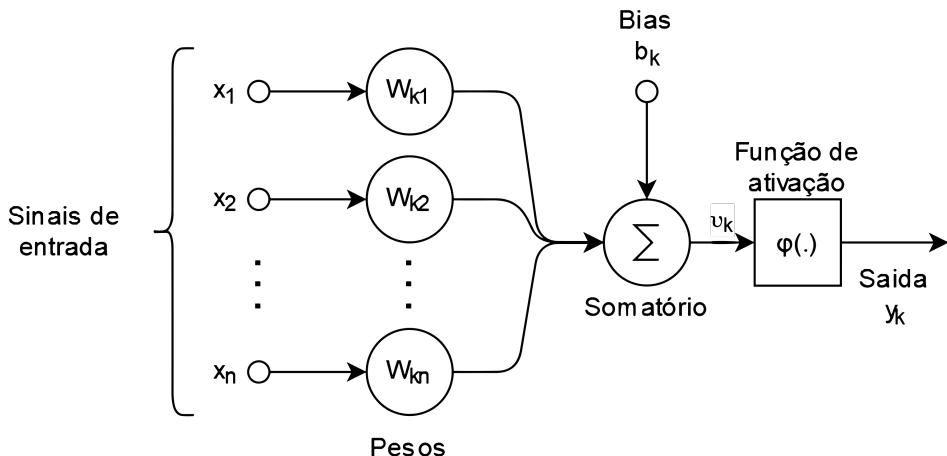


Fonte: Janiesch, Zschech e Heinrich (2021)

2.3.1.1 Rede neural artificial

Uma rede neural artificial é uma representação matemática de unidades de processamento conectadas chamadas de neurônios artificiais. Essa arquitetura simula sinapses, cada sinal trocado entre os neurônios pode aumentar ou atenuar os sinais de outros durante o aprendizado (JANIESCH; ZSCHECH; HEINRICH, 2021).

Figura 12 – Modelo de um neurônio não-linear.



Fonte: Haykin (1999)

Observa-se na Figura 12 o funcionamento de um neurônio k . Os sinais de entradas são partes de um vetor x de tamanho n , sendo o vetor composto por $x_1, x_2 \dots x_n$. Essas componentes são combinadas em uma soma ponderada utilizando seus respectivos pesos, $w_{k1}, w_{k2} \dots w_{kn}$, formando assim a seguinte equação (MARTI; BARROS, 2017 apud HAYKIN, 1999):

$$v_k = \sum_{i=1}^n (x_i * w_{ki}) \quad (2.2)$$

O resultado dessa equação produz o potencial de ativação v_k , esse resultado é somado com o *bias* ou viés b_k para manipular a saída y_k do neurônio, essa soma éposta em uma função não-linear nomeada de função de ativação $\varphi(.)$. Essas funções mapeiam a saída em um intervalo $[0, 1]$ ou $[1, -1]$. A função de saída pode ser representada com a seguinte equação (MARTI; BARROS, 2017 apud HAYKIN, 1999):

$$y_k = \varphi(v_k + b_k) \quad (2.3)$$

O aprendizado ocorre na fase de treinamento, onde é ajustando os pesos w_k e o viés b_k de cada neurônio k . Os pesos w_k são utilizados para calcular a taxa de crescimento da função e o viés b_k é necessário para descolar a saída da função. Com isso é possível modelar uma função linear $y = w^T * x + b$ (MARTI; BARROS, 2017).

Para cada amostra o modelo compara os resultados dos valores atuais dos pesos w_k e viés b_k com o resultado esperado (alvo). Uma função de perda é utilizada para gerar um vetor de gradientes e para quantificar o erro encontrado para a configuração atual do modelo. O modelo atualiza os pesos w_k e os viés b_k no sentido contrário do vetor de gradientes, buscando minimizar a função de perda de acordo com uma taxa de aprendizado (*learning rate*), esse processo é chamado de retropropagação ou *backpropagation* (MARTI; BARROS, 2017).

Ao combinar diversos neurônios artificiais forma-se uma rede neural artificial. Essas redes buscam simular o processamento de informação do cérebro humano (FERNEDA, 2006).

Nas redes neurais, os neurônios são organizados em grupos de unidade de processamento chamados camadas. A primeira e a última camada são nomeadas de camada de entrada e camada de saída, e as demais de camadas ocultas. As camadas mais próximas da entrada são responsáveis por identificar características mais primitivas e as seguintes combinam essas informações para identificar padrões mais complexos (MARTI; BARROS, 2017).

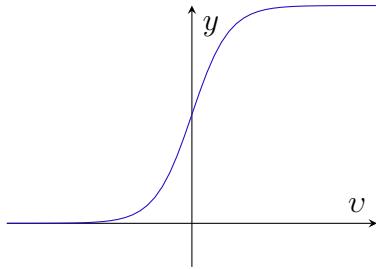
Função de ativação

A função de ativação retorna a saída de um neurônio (HAYKIN, 1999), aqui pode-se ver quatro tipos de funções de ativação:

1. Função *Sigmoid*, uma função não-linear que produz uma curva com a forma de "S". Usada para mapear valores previstos em probabilidades. Tem o valor de saída

Figura 13 – Gráfico da função *Sigmoid*.

$$\varphi(v) = \frac{1}{1 + e^{-v}} \quad (2.4)$$



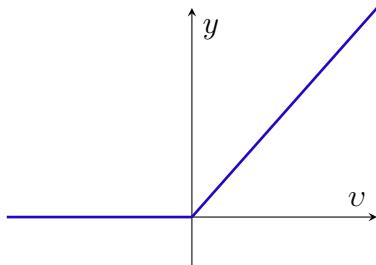
Fonte: Criação própria

entre 0 e 1 (GHARAT, 2019). Segundo Gharat (2019), a função *Sigmoid* tem uma convergência lenta, é computacionalmente cara e para valores muito extremos causa problemas na previsão.

- Função *ReLU* (Unidade Linear Retificada), função não-linear inspirada nos neurônios do cérebro que retorna um valor positivo ou 0 (RIZZO; CANATO, 2020). A função

Figura 14 – Gráfico da função *ReLU*.

$$\varphi(v) = \max(0, v) \quad (2.5)$$



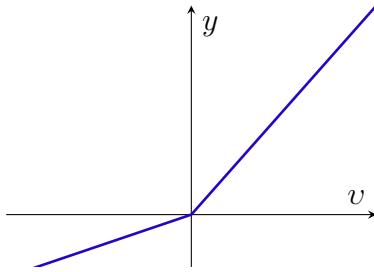
Fonte: Criação própria

ReLU é computacionalmente eficiente e converge rapidamente, porém quando a entrada da função se aproxima de zero a rede neural não consegue executar o retropropagação, sendo assim não há aprendizado (GHARAT, 2019).

- Função *Leaky ReLU* (Unidade Linear Retificada com Vazamento), função não-linear variante da *ReLU* que retorna um valor positivo ou v/a_i , sendo a_i um valor na faixa $(1, \infty)$ (XU et al., 2015). Possui as mesmas características da função *ReLU*, mas sem o problema da retropropagação. (GHARAT, 2019).
- Função *Softmax*, calcula a distribuição de probabilidades de um evento em "n" eventos e fornece a probabilidade do valor de entrada pertencer a uma classe específica, geralmente usada na camada de saída (GHARAT, 2019).

Figura 15 – Gráfico da função *ReLU*.

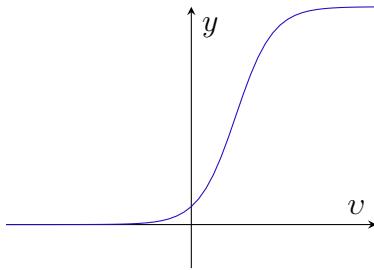
$$\varphi(v) = \begin{cases} v & \text{if } v_k \geq 0 \\ \frac{v}{a_k} & \text{if } v_k < 0 \end{cases} \quad (2.6)$$



Fonte: Criação própria

Figura 16 – Gráfico da função *Softmax*.

$$\varphi(v) = \frac{e^{v_i}}{\sum_{j=0} e^{v_j}} \quad (2.7)$$



Fonte: Criação própria

Com a função *Softmax* é possível normalizar a saída para valores entre 0 e 1, bem como calcular a probabilidade da entrada, e por causa dessas características é utilizada na camada de saída da rede neural ([GHARAT, 2019](#)).

Função de perda

A função de perda é calculada na camada de saída, e serve para mensurar o sucesso obtido comparando com fórmulas o resultado predito com o resultado real do conjunto de dados. O resultado dessa função irá ajudar na retropropagação, *i.e.*, servirá para ajustar os pesos e vieses da conexão entre os neurônios para minimizar o erro. A seguir algumas funções de perda, pontuando que todo esse subtópico é baseado em [Alzubaidi et al. \(2021\)](#).

Softmax ou entropia cruzada ou logarítmica

Muito utilizada para medir a performance de uma rede neural convolucional principalmente quando o resultado tem várias classes. Antes dessa função de perda é necessário usar a função de ativação softmax descrita na Figura 16, pois precisa de uma saída dentro de uma distribuição de probabilidade. Sendo N o número de classes ou o número de neurônios na camada de saída.

$$H(p, y) = - \sum_{i=1}^N y_i \log(p_i) \quad (2.8)$$

Euclidiana ou erro quadrático médio

Muito utilizada para problemas de regressão.

$$H(p, y) = \frac{1}{2N} \sum_{i=1}^N (p_i - y_i)^2 \quad (2.9)$$

Hinge

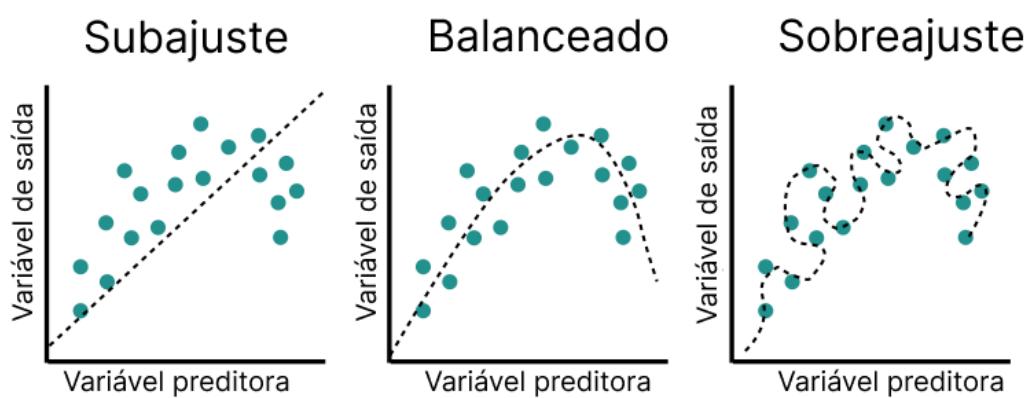
Muito utilizado para classificação binária.

$$H(p, y) = \sum_{i=1}^N \max(0, m - (2y_i - 1)p_i) \quad (2.10)$$

Regularização

Quando se modela uma arquitetura de redes neurais pode se chegar em três casos, sendo eles: subajuste (*underfit*), balanceado (*optimal*) e sobreajuste (*overfit*). O sobreajuste é quando, no treinamento, o modelo acerta as classes porém nos testes não, isso mostra uma dificuldade em generalizar as características. Já o subajuste não consegue pontuar bem em nenhum caso mostrando que o conjunto de dados de treinamento está pequeno para detectar padrões. Por outro lado, o balanceado é quando produz resultados bons tanto no conjunto de dados de treinamento quanto no de testes (ALZUBAIDI et al., 2021; TAYE, 2023).

Figura 17 – Gráficos mostrando subajuste, balanceado e sobreajuste respectivamente

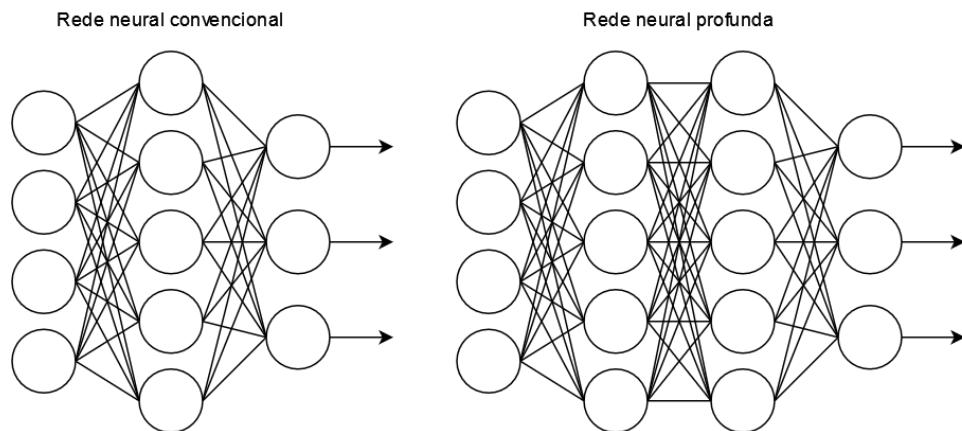


Fonte: [Educative \(2022\)](#)

Rede neural profunda

A principal diferença entre uma rede neural artificial e uma rede neural profunda é a quantidade de camadas, já que uma rede neural profunda possui várias camadas de processamento (MARTI; BARROS, 2017 apud HAYKIN, 1999).

Figura 18 – Comparação de uma rede neural convencional com uma rede neural profunda.



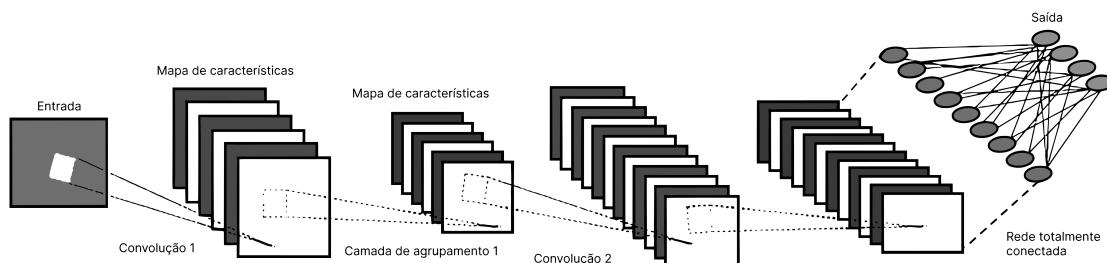
Fonte: Criação própria

2.3.1.1.1 Redes neurais convolucionais

Uma rede neural convolucional é análoga à rede neural artificial, i.e., feita de neurônios que otimizam o aprendizado através dele mesmo. A principal diferença é que a rede neural convolucional é amplamente utilizada em soluções que detectam padrões em imagens, logo existem funcionalidades específicas da própria arquitetura para essa tarefa (O'SHEA; NASH, 2015).

Uma arquitetura básica de uma rede neural convolucional tem as seguintes camadas: convolucional, agrupamento e totalmente conectada. Ilustrada na Figura 19 (SARKER, 2021).

Figura 19 – Camadas principais de uma rede neural convolucional



Fonte: Sarker (2021)

Camada convolucional

Segundo [Taye \(2023\)](#) camada convolucional é essencial para esse tipo de arquitetura e usa um filtro — ou kernel — para aplicar na imagem e direcionar para o próximo neurônio. Esse filtro é uma matriz de números que terá uma operação aplicada em todos os píxeis da imagem — que também é representado por matriz(es) — as informações cruciais para esse filtro são: tamanho, largura e pesos. Isto é utilizado para extrair características com uma base matemática, criando uma relação direta entre um píxel e os píxeis ao redor. Os pesos começam de forma pseudoaleatórias e são ajustados no decorrer do aprendizado. O resultado dessa camada é chamado de mapa de características. O tamanho da saída será baseado na fórmula abaixo sendo os tamanhos I da imagem, F do filtro e a S da saída ([TAYE, 2023](#)).

$$\begin{aligned} \mathbf{I}_x - \mathbf{F}_x + 1 &= \mathbf{S}_x \\ \mathbf{I}_y - \mathbf{F}_y + 1 &= \mathbf{S}_y \end{aligned} \tag{2.11}$$

A seguir um exemplo dos passos para construir a matriz resultante baseado em [Alzubaidi et al. \(2021\)](#).

$$\begin{array}{ccc}
 \text{Matriz 2x4} & \text{Filtro 2x2} & \text{Resultado} \\
 \left[\begin{array}{cc|cc} 0 & 2 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array} \right] & \otimes & \left[\begin{array}{cc} 0 & 1 \\ -1 & 2 \end{array} \right] = \left[\begin{array}{cc|c} 1 & - & - \end{array} \right] \\
 \\
 \left[\begin{array}{cc|cc} 0 & 2 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array} \right] & \otimes & \left[\begin{array}{cc} 0 & 1 \\ -1 & 2 \end{array} \right] = \left[\begin{array}{cc|c} 1 & 1 & - \end{array} \right] \\
 \\
 \left[\begin{array}{cc|cc} 0 & 2 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array} \right] & \otimes & \left[\begin{array}{cc} 0 & 1 \\ -1 & 2 \end{array} \right] = \left[\begin{array}{cc|c} 1 & 1 & 2 \end{array} \right]
 \end{array}$$

Tamanho do passo e preenchimento

O tamanho do passo — ou *stride* — serve para especificar a distância de píxeis entre os passos da camada. No exemplo acima esse parâmetro é definido como 1, por isso a matriz selecionada pula 1 pixel para direita entre os passos. Esse valor altera o tamanho da matriz resultante ([SARKER, 2021](#)).

O preenchimento — ou *padding* — é uma técnica utilizada para manter o mesmo tamanho da entrada, adicionando bordas com zeros antes das operações da camada para ter como saída uma matriz da mesma dimensão da matriz original. Isso é usado devido a desvantagem em perder os detalhes nas bordas das imagens no processamento de uma camada ([SARKER, 2021](#)).

Camada de agrupamento

A camada de agrupamento — ou *pooling* — tem como tarefa primordial uma técnica para reduzir o tamanho do mapa de características, porém preservando os padrões mais relevantes. Dentre os recursos essenciais dessa camada estão o tamanho do agrupamento e a operação que será realizada. O maior problema dessa camada é pelo fato dela apenas identificar onde essas características estão e se existem ou não, *i.e.*, dependendo de qual operação e a quantidade de camadas pode não ser possível guardar as principais características de forma íntegra causando uma redução no desempenho final da predição ([SARKER, 2021](#)).

Existem vários tipos de agrupamento, os mais utilizados são: agrupamento máximo, agrupamento médio e agrupamento global médio que estão explicados abaixo em exemplos baseados em [Alzubaidi et al. \(2021\)](#).

Agrupamento máximo

É definido o resultado com base no máximo encontrado pelo tamanho do agrupamento, exemplo a seguir usando um mapa de características com tamanho 4x4 e agrupamento de tamanho 2x2.

$$\left[\begin{array}{cc|cc} 4 & 25 & 44 & 10 \\ 8 & 14 & 8 & 33 \\ \hline 17 & 2 & 16 & 34 \\ 5 & 13 & 24 & 7 \end{array} \right] = \begin{bmatrix} 25 & 44 \\ 17 & 34 \end{bmatrix}$$

Agrupamento médio

É definido o resultado com base na média encontrada pelo tamanho do agrupamento, exemplo a seguir usando um mapa de características com tamanho 4x4 e agrupamento de tamanho 2x2.

$$\left[\begin{array}{cc|cc} 4 & 25 & 44 & 10 \\ 8 & 14 & 8 & 33 \\ \hline 17 & 2 & 16 & 34 \\ 5 & 13 & 24 & 7 \end{array} \right] = \begin{bmatrix} 12 & 23 \\ 9 & 20 \end{bmatrix}$$

Agrupamento global médio

É definido o resultado com base na média geral do mapa o que sempre tem como saída uma matrix 1x1, exemplo a seguir usando um mapa de características com tamanho 4x4.

$$\left[\begin{array}{cccc} 4 & 25 & 44 & 10 \\ 8 & 14 & 8 & 33 \\ 17 & 2 & 16 & 34 \\ 5 & 13 & 24 & 7 \end{array} \right] = [16]$$

Camada totalmente conectada

A camada totalmente conectada geralmente é utilizada no final da arquitetura e cria a partir de cada neurônio uma ligação direta para cada etiqueta final. Isso torna essa camada extremamente pesada computacionalmente. O número de neurônios dessa camada é equivalente ao número de classes propostas. Além disso é quando chega nessa camada que a função de perda é calculada e se inicia a retropropagação (ALZUBAIDI et al., 2021; TAYE, 2023).

Aperfeiçoamento

Segundo Alzubaidi et al. (2021), Taye (2023) existem algumas técnicas para aperfeiçoar os resultados do modelo, sendo elas:

- Dropout: Muito utilizada para evitar sobreajuste pois está técnica irá desligar um neurônio aleatoriamente colocando a saída dele como zero no processo de treinamento e portanto fará o modelo a aprender a identificar características diferentes em outros neurônios possibilitando a generalização do modelo.
- Aumentar o tamanho do conjunto de dados: caso não seja possível criar ou encontrar um maior existem técnicas para aumentar artificialmente acrescentando pequenas mudanças nas imagens existentes, algumas são rotacionar, recortar e inverter horizontalmente ou verticalmente.
- Normalização em lote: normaliza as saídas para treinar a rede mais rápido
- Aumentar o tempo de treinamento
- Aumentar a profundidade ou largura da arquitetura
- Ajustar os hiperparâmetros

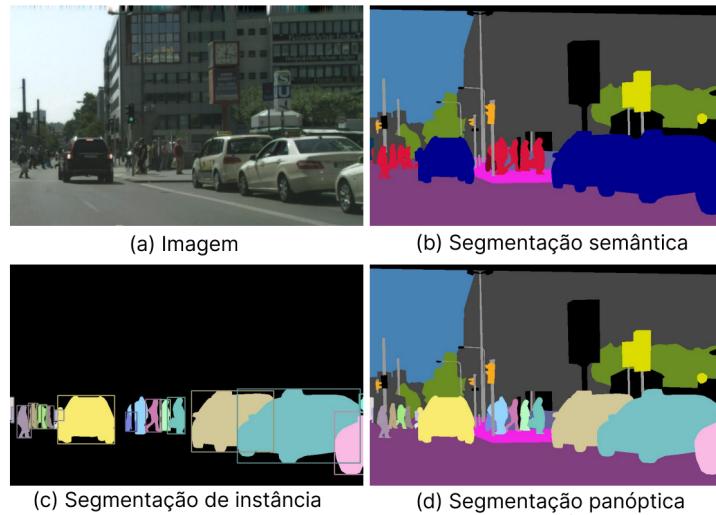
Segmentação

O estudo de segmentação semântica dentro da área de redes neurais convolucionais têm três principais nichos, sendo eles: segmentação semântica que é a classificação por pixel, a segmentação de instância que atribui um id para cada objeto encontrado de uma classe, e a segmentação panóptica que junta as duas anteriores para criar uma imagem semelhante a saída de segmentação semântica porém separando objetos de mesma classe sendo essa a mais recente e completa, a diferença entre esses três tipos está ilustrado na Figura 20 (ULKU; AKAGÜNDÜZ, 2022; WANGENHEIM, 2021).

Segmentação semântica

A segmentação semântica começou a ter resultados satisfatórios a partir de redes totalmente convolucionais, com o objetivo de segmentar imagens classificando pixels, esse modelo descarta a camada totalmente conectada pois a saída deverá ser uma imagem e não uma classificação — isso a torna mais rápida para treinar do que as redes neurais convolucionais —, logo usa camadas deconvolucionais para transformar a matriz de características em uma imagem de qualquer dimensão na saída. A RTC criou a arquitetura chamada de salto (ou conexões) que serve para evitar perdas em camadas de agrupamento criando conexões entre camadas não consecutivas — geralmente entre camadas convolucionais e

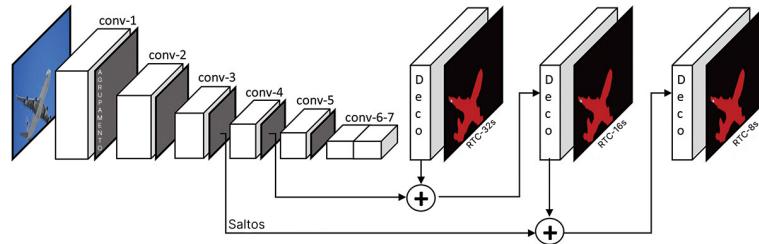
Figura 20 – Tipos de segmentação em redes neurais convolucionais



Fonte: Kirillov et al. (2019)

deconvolucionais — como apresentado na Figura 21, a arquitetura de salto evoluiu para arquitetura codificador-decodificador.

Figura 21 – Exemplo de arquitetura de rede totalmente convolucional



Fonte: Ulku e Akagündüz (2022)

A arquitetura codificador-decodificador — ou Encoder-Decoder —, é separada em dois passos: o primeiro para convergir no mapa de características — chamado de codificador — e o segundo para reverter — chamado de decodificador — as camadas de agrupamento para aumentar a dimensão da saída, usando camadas deconvolucionais e de desagrupamento. Outra característica importante é a conexão entre camadas de mesmo nível, como por exemplo a arquitetura UNet, que foi a primeira a implementar o padrão Codificador-Decodificador. Na Figura 22 pode-se observar que tem formato da letra U, sendo a descida a parte de codificação e subida decodificação (ULKU; AKAGÜNDÜZ, 2022; WANGENHEIM, 2021; RONNEBERGER; FISCHER; BROX, 2015).

Figura 22 – Arquitetura codificador-decodificador UNet



Fonte: Ronneberger, Fischer e Brox (2015)

Segmentação de instância

Outro problema dentro da área de visão computacional é a detecção de objetos, a primeira solução foi com Características de regiões com RNC — Regions with CNN features (R-CNN) — que se resume em dividir a imagem de entrada em regiões de interesse e nessas regiões aplicar uma RNC. A arquitetura que seleciona essas regiões é chamada de Rede de Proposta de Região (RPR) — ou Region Proposal Network (RPN) — o que auxilia na detecção por caixas delimitadoras. Essa ideia inicial foi estendida para segmentação de instância criando também máscara nos objetos, como por exemplo a arquitetura Mask R-CNN ([ULKU; AKAGÜNDÜZ, 2022](#); [WANGENHEIM, 2021](#)).

A arquitetura Mask R-CNN é derivado do Fast R-CNN — aprimoramento do R-CNN aplicando conceito RoIPool para classificar — onde há uma segmentação de máscara em cada ROI — ou Região de interesse — paralela com a classificação da caixa delimitadora. A máscara é classificada com uma pequena RTC em cada ROI. Além de ter uma pequena melhoria na RoIPool, pois havia um problema de alinhamento nas localizações espaciais exatas, essa camada é chamada de RoIAlign ([HE et al., 2017](#)).

Segmentação panóptica

Um problema encontrado na segmentação semântica é que objetos de mesma classe não são separados como na segmentação de instância, logo surgiu uma ideia para criar uma solução usando as duas técnicas. Esse conceito surgiu do trabalho Kirillov et al. (2019) e consiste na definição geral da ideia, uma métrica — que será explicada posteriormente — unificada para classificar os resultados do modelo além de fazer a distinção entre coisas — ou stuff — que não são contáveis, como o céu e os objetos — ou things — que são

contáveis como carros, pessoas, etc. Os principais conjunto de dados para tarefa panóptica são COCO-Panoptic, Cityscapes, Mapillary Vistas, ADE20K, e Indian Driving Dataset. Cada conjunto de dados possui diversas classes para o modelo aprender e todos são de contexto urbano que apesar de ser amplo ainda tem suas limitações ([BARLA, 2022](#)).

Métricas e técnicas

No ramo de segmentação existem diversas métricas que podem ser usadas, dentre elas algumas serão destacaadas nesta monografia a seguir:

Classificação de conjuntos

A classificação de conjuntos é uma técnica para criar relações entre a imagem de predição e a imagem do conjunto de dados. Ela se divide em quatro conjuntos sendo eles: Positivo Verdadeiro — ou True Positive(TP) — , Negativo Verdadeiro — ou True Negative(TN) —, Falso Positivo — ou False Positive(FP) — quando um objeto não é correspondido na imagem de predição e por fim o Falso Negativo — ou False Negatives(FN) — quando um objeto não é correspondido na imagem do conjunto de dados. Essa técnica pode ser usada em modelos de classificação binária usando uma matriz de confusão mas na área de segmentação costuma-se contar os pixeis e sua representação visual pode-se observar na Figura 23 ([KIRILLOV et al., 2019](#); [WANG; WANG; ZHU, 2020](#)).

Figura 23 – Exemplo da classificação dos conjuntos usados nas métricas de segmentação



Fonte: [Kirillov \(2019\)](#) editado

F1 Score

O F1 Score é uma métrica que pode ser utilizada para calcular a eficiência de modelos de segmentação de instância, baseando-se em encontrar uma relação entre a área das classes da imagem de saída com as classes na imagem do conjunto de dados porém diminuindo o peso dos erros. Varia de 0 a 1 sendo 1 com maior eficiência. Segue a exemplificação da Equação (2.12) (CHICCO; JURMAN, 2020):

$$F1\ Score = \frac{TP}{(TP + \frac{FP}{2} + \frac{FN}{2})} \quad (2.12)$$

Coeficiente de correlação de Matthews

O coeficiente de correlação de Matthews — ou Matthews Correlation Coefficient (Mcc) — é uma métrica que pode ser utilizada para mensurar a eficiência de modelos de segmentação, baseando-se no coeficiente de correlação de Pearson. Varia de -1 a 1 sendo 1 com maior correlação, 0 inconclusivo e -1 sem correlação. Segue a exemplificação da Equação (2.13) (CHICCO; JURMAN, 2020; NEDEA, 2020):

$$Mcc = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \quad (2.13)$$

Taxa de descoberta falsa

A taxa de descoberta falsa — ou False Discovery Rate (FDR) — é uma métrica utilizada para calcular o erro obtido na imagem de predição, baseando-se em encontrar uma relação entre ao erro obtido, divido pelo erro mais acerto esperado. Varia de 0 a 1 sendo 1 o pior caso e 0 o melhor. Segue a exemplificação da Equação (2.14) (NEDEA, 2020):

$$Fdr = \frac{FP}{FP + TP} \quad (2.14)$$

Taxa de Falso Negativo

A taxa de falso negativo — ou False Negative Rate (FNR) — é uma métrica utilizada para calcular o erro obtido na imagem verdade, baseando-se em encontrar uma relação entre ao erro obtido, divido pelo erro mais acerto esperado. Varia de 0 a 1 sendo 1 o pior caso e 0 o melhor. Segue a exemplificação da Equação (2.15) (NEDEA, 2020):

$$Fnr = \frac{FN}{FN + TP} \quad (2.15)$$

Acurácia

A acurácia — ou Accuracy (Acc) — é uma métrica utilizada para calcular a eficiência de modelos de segmentação semântica, baseando-se em encontrar uma relação entre a área das classes da imagem de saída com as classes na imagem do conjunto de dados para todos conjuntos. Varia de 0 a 1 sendo 1 com maior eficiência. Segue a exemplificação da Equação (2.16) (CHICCO; JURMAN, 2020):

$$Acc = \frac{TP + TN}{(TP + TN + FP + FN)} \quad (2.16)$$

União sobre intersecção

A união sobre intersecção — ou Intersection over Union (IoU) — ou índice de Jaccard é uma métrica muito utilizada para calcular a eficiência de modelos de segmentação, ela se baseia em encontrar uma relação entre a área das classes da imagem de saída com as classes na imagem do conjunto de dados para qualquer valor positivo, isto é, desconsiderando o conjunto positivo falso. Varia de 0 a 1 sendo 1 com maior eficiência. Segue a exemplificação da Equação (2.17) (BORA, 2022):

$$IoU = \frac{TP}{(TP + FP + FN)} \quad (2.17)$$

Qualidade panóptica

A qualidade panóptica — ou Panoptic Quality (PQ) — foi definido pela primeira vez no artigo Kirillov et al. (2019), e se resume na fórmula:

$$PQ = \frac{\sum_{(p,g) \in TP} IoU(p, g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} \quad (2.18)$$

Multiplicando a Equação (2.18) por $\frac{|TP|}{|TP|}$ tem-se:

$$PQ = \underbrace{\frac{\sum_{(p,g) \in TP} IoU(p, g)}{|TP|}}_{\text{Segmentation Quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{Recognition Quality (RQ)}} \quad (2.19)$$

Portanto pode-se concluir que PQ é apenas uma simplificação para uma fórmula que contém uma relação entre métricas de segmentação semântica e de instância.

Com base no subtópico Segmentação percebe-se que a segmentação panóptica é a mais completa e por efeito de estudos será utilizado o mesmo para concluir o trabalho. Como nesse nicho existem várias alternativas será utilizado a métrica demonstrada na Equação (2.18) para analisar resultados e selecionar o modelo.

Resultados de modelos do nicho de segmentação panóptica

Os resultados são de uma competição em aberto criada pela Cityscapes Dataset, essa competição tem várias modalidades e esses são referentes ao nicho de segmentação panóptica utilizando a métrica PQ na classe de pessoas. A exibição dos resultados contendo a métrica PQ que classifica pessoas com os quinze melhores modelo encontra-se na Tabela 1 ([DATASET, 2023](#)).

Tabela 1 – Top 15 modelos que melhor classificam pessoas com métrica P em segmentação panóptica

Nome do modelo	PQ (%)
EfficientPS [Mapillary Vistas]	61,6
EfficientPS [Cityscapes-fine]	60,9
Panoptic-DeepLab w/ SWideRNet [Mapillary Vistas + Pseudo-labels]	60,6
hri_panoptic	60,6
Naive-Student (iterative semi-supervised learning with Panoptic-DeepLab)	60,2
Panoptic-DeepLab w/ SWideRNet [Mapillary Vistas]	59,8
iFLYTEK-CV	59,2
Panoptic-DeepLab [Mapillary Vistas]	58,5
Panoptic-DeepLab w/ SWideRNet [Cityscapes-fine]	58,4
Seamless Scene Segmentation	57,7
Axial-DeepLab-XL [Mapillary Vistas]	57,2
Unifying Training and Inference for Panoptic Segmentation [COCO]	56,5
kMaX-DeepLab [Cityscapes-fine]	56
Axial-DeepLab-L [Mapillary Vistas]	55,9
TASCNet-enhanced	55,2

EfficientPS

EfficientPS é uma solução eficiente para a segmentação panóptica proposta no artigo [Mohan e Valada \(2021\)](#). E em seu repositório na internet de [Mohan e Valada \(2020\)](#) possui algumas recomendações de tecnologias, que são as que foram usadas para fazer e testar o modelo, logo poderá funcionar em outros cenários porém sem suporte dos criadores. Essas tecnologias são: uma distribuição de sistema operacional ¹ com núcleo Linux de acordo com [Hat \(2023\)](#), a linguagem de programação Python na versão 3.7 que de acordo com [Magazine \(2023\)](#) é amplamente utilizada para o ramo de IA devido a simplicidade e diversas bibliotecas que facilitam a criação de modelos complexos, o PyTorch na versão 1.7 que de acordo com [PyTorch \(2023\)](#) é uma estrutura baseada na biblioteca Torch que visa facilitar a criação de modelos de aprendizado profundo, CUDA Toolkit na versão 10.2 que de acordo com [Developer \(2023\)](#) é uma biblioteca para proporcionar aceleração do

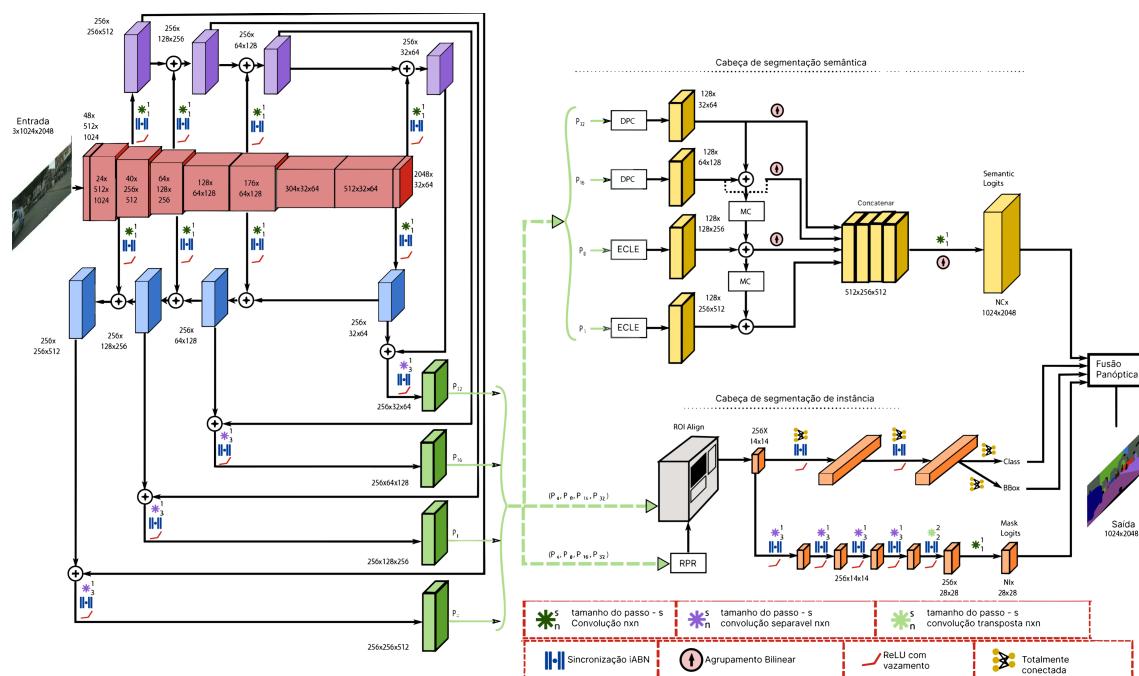
¹ Programa que gerencia a parte física de um computador; Meio termo entre o usuário e a parte física do computador

processamento com placas de vídeo da marca Nvidia e por fim GCC na versão 7 ou 8 que de acordo com [Project \(2023\)](#) é um compilador da linguagem C e C++ que geralmente são utilizados em bibliotecas Python como por exemplo PyTorch.

Arquitetura

O artigo apresenta uma arquitetura que se inicia com um *backbone* — parte para identificar características — usando uma Rede de Pirâmide de Características(RPC)² de 2 caminhos seguido de dois cabeçotes paralelos um para uma arquitetura de segmentação semântica que é autoria deles e outra de instância com modificações baseadas na topologia Mask R-CNN e finalmente a saída dos dois cabeçotes são combinadas no módulo de fusão panóptica para gerar a saída final com a imagem de segmentação panóptica, esta arquitetura é ilustrada na Figura 24.

Figura 24 – Arquitetura geral do EfficientPS



Fonte: [Mohan e Valada \(2021\)](#)

Backbone da rede

A espinha dorsal — ou backbone — se consiste em uma codificação combinado a uma bifurcação paralela usando RPC. O codificador é essencial para arquiteturas de segmentação e para melhorar a capacidade de representação é necessário aumentar o número de parâmetros e a complexidade, porém nesse artigo os autores chegaram em

² Estrutura de pirâmide para extrair características em várias escalas de uma imagem ([LIN et al., 2016](#))

uma solução balanceada nesse quesito. O codificador contém nove blocos (em vermelho), mostrado na Figura 24 e a 2^o, 3^o, 5^o e 9^o saídas — da esquerda para direita — correspondem aos fatores de redução de amostragem x4,x8,x16 e x32 respectivamente. Essas saídas vão conectar com a bifurcação paralela que são de sentidos opostos para gerar mais detecções de características. Após isso será feita uma combinação entre camadas de mesma dimensão utilizando camadas de convolução separável em profundidade — divide em etapa espacial e de canal, aplicada a cada canal e cada pixel de saída respectivamente — resultando nas saídas $P_4 + P_8 + P_{16} + P_{32}$ (MOHAN; VALADA, 2021; LIMA, 2021).

Cabeçote de Segmentação Semântica

O cabeçote de segmentação semântica é autoria dos autores e é dividido em três módulos sendo eles: Extrator de Características em Larga Escala (ECLE) — ou Large Scale Feature Extractor (LSFE) — para capturar recursos finos em larga escala de forma eficiente, módulo DPC deve ser capaz de capturar contexto de longo alcance, porém em pequena escala e o módulo MC deve ser capaz de mitigar a incompatibilidade entre recursos de grande e pequena escala nas camadas de agregação (MOHAN; VALADA, 2021).

As quatro entradas do cabeçote $P_4 + P_8 + P_{16} + P_{32}$ são separadas, sendo $P_{16} + P_{32}$ — pequena escala — alimentam dois módulos DPC paralelos e $P_4 + P_8$ — larga escala — alimentam dois módulos ECLE paralelos (MOHAN; VALADA, 2021).

Cabeçote de segmentação de instância

Este cabeçote é derivada da arquitetura Mask R-CNN e as modificações foram três, sendo: trocar a convolução padrão por convolução separável em profundidade — para reduzir o número de parâmetros consumidos pela rede —, camada de normalização em lote foi substituída por iABN Sync³ e a função ReLU definida em Equação (2.5) por Leaky ReLU definida em Equação (2.6) (MOHAN; VALADA, 2021; LIMA, 2021; SCHUMACHER, s. d.).

Módulo de fusão panóptica

O módulo da fusão panóptica é necessário para construir a imagem com segmentação panóptica. Nessa parte os resultados são unidos aos dois cabeçotes anteriormente explicados. Esta tarefa não é simples pois é necessário criar uma lógica para obter o melhor resultado diante das sobreposições encontradas. O módulo foi criado no intuito de ser adaptativo e usar as duas entradas de forma equivalente (MOHAN; VALADA, 2021).

Resumindo o módulo aplica algumas técnicas para reduzir o número de instâncias baseando-se na métrica logist — valor numérico que pontua confiança — aplica algumas

³ normalização em lotes entre cores de GPU para aumentar o desempenho

agregações entre os resultados dos dois cabeçotes e desenha com fundo preto as instâncias com melhor classificação de confiança, logo depois preenche com a parte de *stuff* — classes semânticas sem importância — da entrada semântica ([MOHAN; VALADA, 2021](#)).

2.4 Trabalhos relacionados

Esta seção destina-se a análise e discussão da metodologia e dos resultados propostos por [Kirillov et al. \(2019\)](#), [Mohan e Valada \(2021\)](#), [Patel \(2010\)](#).

Panoptic Segmentation

No trabalho [Kirillov et al. \(2019\)](#) é definido a ideia geral de segmentação panóptica além de definir conceitos importantes como coisas e objetos e a métrica unificada para medir o desempenho de modelos dessa área. Também é feito alguns testes comparando resultados humanos com um modelo simples proposto com eles combinando PSPNet e Mask R-CNN usando a métrica de qualidade panóptica definida por eles. Os resultados mostraram a superioridade humana na segmentação panóptica em três conjuntos de dados diferentes, sendo eles: Cityscapes, ADE20k e Vistas. As métricas usadas foram qualidade panóptica, qualidade semântica, qualidade de reconhecimento, qualidade panóptica de coisas e qualidade panóptica de objetos. O melhor resultado para a máquina em comparação com o humano foi no conjunto de dados Cityscapes avaliando a qualidade semântica, sendo 84,1 para o humano e 80,9 para máquina. O pior resultado para a máquina em relação ao humano foi no conjunto de dados ADE20k na qualidade panóptica de coisas, sendo 71,0 para os humanos e 24,5 para a máquina.

EfficientPS: Efficient Panoptic Segmentation

No trabalho [Mohan e Valada \(2021\)](#) é introduzido o EfficientPS, uma arquitetura concebida para abordar a complexa tarefa de segmentação panóptica, uma área crucial em aplicações como direção autônoma, robótica e realidade aumentada. A metodologia proposta engloba um backbone leve baseado na arquitetura EfficientNet, um módulo de fusão inovador e uma nova função de perda que visa equilibrar as coisas e objetos na segmentação. Os resultados obtidos, em comparação com métodos anteriores, utilizando conjuntos de dados reconhecidos, como Cityscapes, COCO e KITTI, indicam que o EfficientPS supera implementações anteriores, alcançando pontuações da métrica de qualidade panóptica notáveis. Além disso, destaca-se pela eficiência computacional, evidenciada por tempos de inferência significativamente menores em um único GPU. Este trabalho representa uma contribuição relevante para a pesquisa em segmentação panóptica, fornecendo uma abordagem eficaz e eficiente para aplicações que demandam precisão e agilidade nesta tarefa complexa.

Polygonal Map Generation for Games

No artigo de Patel (2010) é apresentado toda uma jornada de desenvolvimento de um algoritmo de geração procedural de conteúdo, é mostrando as técnicas para gerar o mapa com o diagrama de Voronoi, gerar os rios e biomas utilizando as camadas de elevação e umidade do polígono e a aplicando isso no diagrama de Whittaker para definir o bioma do polígono. Também é apresentado uma técnica para adicionar ruídos nas arestas dos polígonos fazendo com que o mapa se torne mais orgânico e realista.

3 Desenvolvimento

Neste capítulo é apresentada a descrição da proposta e relatado o processo de desenvolvimento e avaliação da implementação.

3.1 Proposta

Este trabalho propõe a criação de um protótipo destinado a gerar um mapa de forma procedural, cujo contorno é selecionado a partir de uma imagem processada por um modelo de segmentação panóptica¹. O mapa resultante replicará o contorno da seleção em duas dimensões e três dimensões.

A Figura 25 apresenta os passos dessa abordagem, onde a imagem a representa a entrada para o processo de segmentação panóptica, e a imagem b é o resultado desse processo. Utilizou-se o modelo de segmentação panóptica EfficientPS criado por [Mohan e Valada \(2021\)](#), uma solução eficiente para segmentação panóptica e de acordo com [Dataset \(2023\)](#) possui a melhor classificação da métrica de qualidade panóptica definida por [Kirillov et al. \(2019\)](#) para classificar pessoas, o que é importante em um contexto atual segundo [Ulku e Akagündüz \(2022\)](#), e pode ser encontrado no repositório dos próprios autores [Mohan e Valada \(2021\)](#) disponível na internet. Após a segmentação da imagem, o usuário terá a capacidade de selecionar um contorno com as técnicas de selecionar por cor e por preenchimento de inundação de acordo com [OpenCV \(2023c\)](#) e [OpenCV \(2023b\)](#), respectivamente. Além disso decidiu-se criar mais duas opções para segmentação devido a limitação dos conjuntos de dados para segmentação panóptica, de acordo com [Barla \(2022\)](#) os conjuntos são de contexto urbano o que limita o número de classes de objetos, logo adicionar a seleção por cor e por preenchimento de inundação aumentará a gama de utilização, possibilitando por exemplo selecionar um desenho feito em uma folha sulfite. O resultado da seleção será uma máscara binária de acordo com [Embarcados \(2017\)](#) é uma maneira de isolar um objeto na imagem, conforme ilustrado na imagem c, que representa a escolha do segmento, nesse exemplo o carro amarelo da imagem b. Subsequentemente, um mapa procedural, que segundo [Yannakakis e Togelius \(2018\)](#) são métodos e automações que geram mapas, será gerado com o contorno escolhido, conforme evidenciado na imagem d. Essa representação incluirá uma sobreposição do contorno do veículo, meramente para ilustrar que a ilha gerada terá um contorno semelhante. As cores na representação indicam o oceano (azul), a floresta (verde) e as montanhas (cinza). Além disso, será automatizado para atualizar o relevo de um mapa em três dimensões.

¹ No resultado apenas será identificado os pixels de classes contidas nos conjuntos de dados escolhidos, e todos citados no capítulo 2 são de contexto urbano

Figura 25 – Etapas da proposta



Fonte: Kirillov et al. (2019) e autoria própria

3.2 Implementação

Pode-se dividir a implementação em cinco partes, sendo elas: segmentar imagens, selecionar contorno, gerar proceduralmente o mapa com biomias, analisar casos de pós processamento e interligar as ferramentas através de uma interface gráfica.

3.2.1 Segmentar imagens

A segmentação da imagem é usada para classificar os píxeis da imagem a partir de padrões reconhecidos por uma inteligência artificial. A partir disso é possível o usuário selecionar o contorno para gerar o mapa.

A linguagem de programação utilizada para desenvolvimento do projeto foi Python pois existem muitas bibliotecas que auxiliam na criação de arquiteturas complexas de Inteligência Artificial como o [EfficientPS](#).

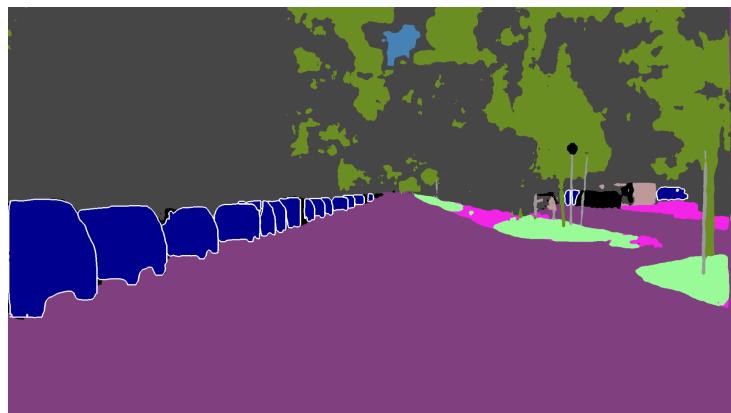
Utilizou-se o código aberto oficial do trabalho científico postado no repositório de [Mohan e Valada \(2020\)](#) que foi baseado no tópico [EfficientPS](#).

Utilizou-se um arquivo pré treinado desse modelo contendo o aprendizado do conjunto de imagens do *Cityscapes* para segmentação panóptica, a ideia inicial era treinar com pelo menos mais um conjunto de dados porém surgiu-se diversos desafios para serem solucionados, dentre eles: conseguir autorização para baixar conjuntos de dados específicos

para segmentação panóptica, outro problema foi preparar esse conjunto de dados para processar o aprendizado, além de problemas de limitação de processamento bruto. Portante decidiu-se utilizar o próprio modelo salvo baixado do repositório.

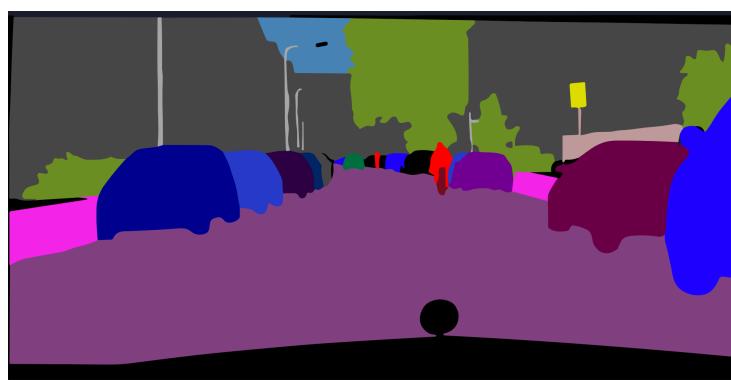
Percebeu-se que o resultado do modelo proposto não foi o esperado pois objetos de mesma classe como carros tem a mesma cor com uma borda branca exemplificados na Figura 26. Logo tentou-se mudar o código para gerar uma saída com objetos de mesma classe com cores diferentes como na figura Figura 27 que está no artigo do EfficientPS por [Mohan e Valada \(2021\)](#), no qual cada objeto da classe carro tem uma cor diferente e não tem borda entre os objetos.

Figura 26 – Resultado inicial do EfficientPS.



Fonte: Criação própria

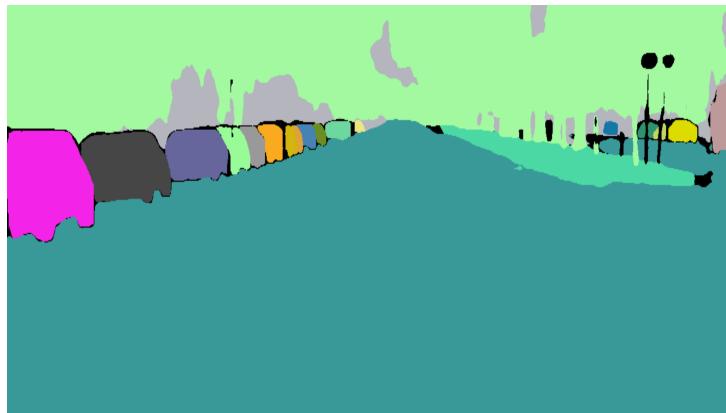
Figura 27 – Expectativa do resultado do EfficientPS.



Fonte: [Mohan e Valada \(2021\)](#)

Seguiu-se os passos citados em uma publicação de [Mohan e Gabriele \(2021\)](#) no repositório de [Mohan e Valada \(2020\)](#) porém o resultado obtido não foi satisfatório pois não possível discernir quais segmentos pertenciam as respectivas classes, o resultado é ilustrado na Figura 28.

Figura 28 – Resultado do EfficientPS obtido seguindo os passos da publicação no repositório.



Fonte: Criação própria

3.2.2 Selecionar contorno

Para representar a seleção do contorno utilizou-se uma técnica chamada de imagem binária, segundo [Embarcados \(2017\)](#) consiste em isolar um objeto da imagem. Pode-se ser utilizada como uma máscara para auxiliar o processamento do mapa no contorno desejado ([AZNAG et al., 2020](#)).

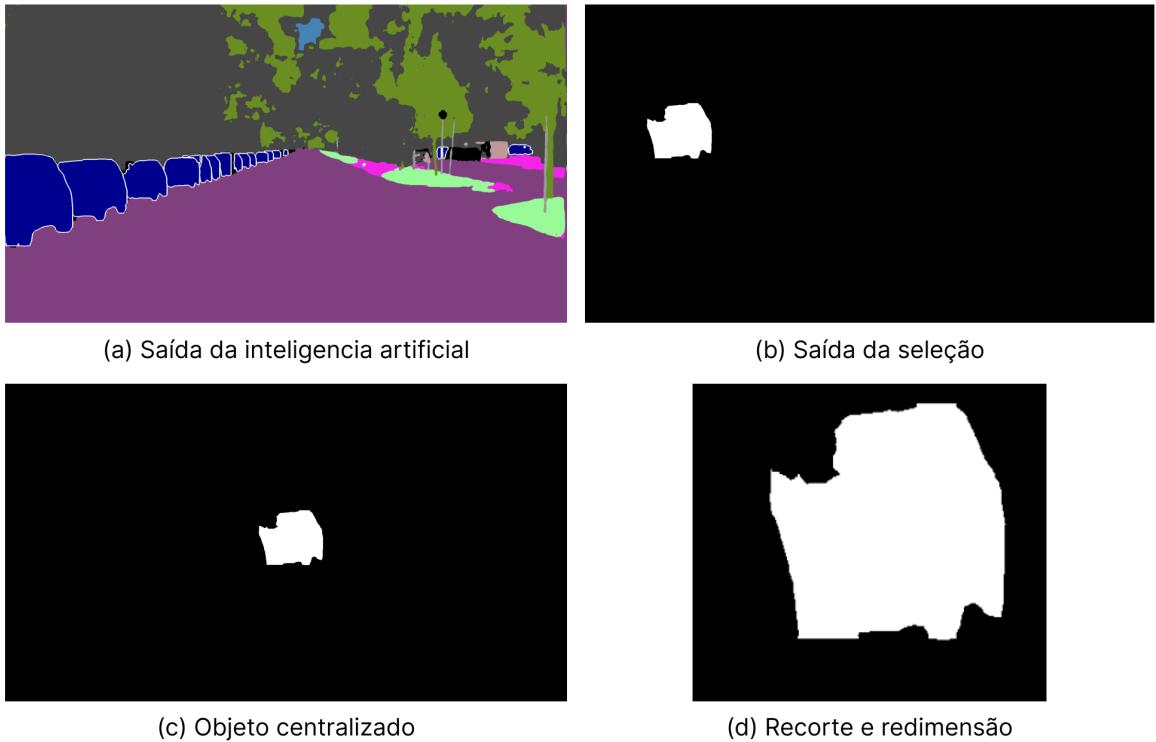
Utilizou-se duas maneiras para selecionar o contorno, sendo elas: selecionar por cor e selecionar por preenchimento de inundação. Selecionar por cor segundo [OpenCV \(2023c\)](#) consiste em isolar tudo na imagem com aquela cor, já selecionar por preenchimento de inundação segundo [OpenCV \(2023b\)](#) consiste em um algoritmo de expansão comparando com uma faixa delimitada de cores. Ambos tem-se como saída uma imagem binária.

Tratamento da imagem binária

Após a saída dos algoritmos de seleção, o objeto selecionado é detectado e centralizado em uma nova imagem, depois recortamos e redimensionamos a partir do centro de forma que fique quadrada. Todos os passos são observáveis na Figura 29.

Porém há uma maneira mais performática para fazer isso, inclusive melhorando os resultados devido ao redimensionamento e a borda não chegarem a um bom cenário. Essa outra maneira usa um método para localizar o objeto na imagem binária e corta a imagem com as coordenadas e dimensões oferecidas. Após isso é acrescentado uma borda na altura ou largura com a folga entre as duas, tornando assim a imagem quadricular, depois é redimensionado para manter um padrão e acrescentado mais uma vez uma borda para representar o mar. É possível observar na Figura 30 que o objeto não foi distorcido como na método anterior e manteve-se centralizada na imagem final.

Figura 29 – Passos da seleção da saída da inteligência artificial.



Fonte: Autoria própria

3.2.3 Geração procedural do mapa

Usou-se como base para a geração procedural do mapa o artigo [Polygonal Map Generation for Games](#) com uma implementação feita em Python por [Markiewicz et al. \(2021\)](#).

3.2.3.1 Ilha gerada no contorno

Para gerar o mapa da ilha é preciso gerar um diagrama de Voronoi. No diagrama primeiro é definido os pontos de quantidade preestabelecida e de localização pseudo-aleatória, esses pontos serão os centroides dos polígonos. Os vértices dos polígonos são gerados a partir da intersecção entre retas perpendiculares aos pontos médios entre os nós vizinhos, logo é criado outro grafo com esses pontos. A definição dos vértices é ilustrada na Figura 31, sendo os pontos vermelhos os centroides que são ligados por linhas pretas para gerar pontos médios, representados por pontos amarelos para traçar uma reta perpendicular, depois é calculado a intersecção representado pelo ponto azul que se torna o vértice, e a cor rosa representa a aresta do polígono.

A Figura 32 ilustra o modelo do diagrama de Voronoi do algoritmo, sendo os pontos vermelhos os centroides que são ligados por linhas pretas, os pontos azuis são os vértices

Figura 30 – Passos da seleção da saída da inteligência artificial mais performático.



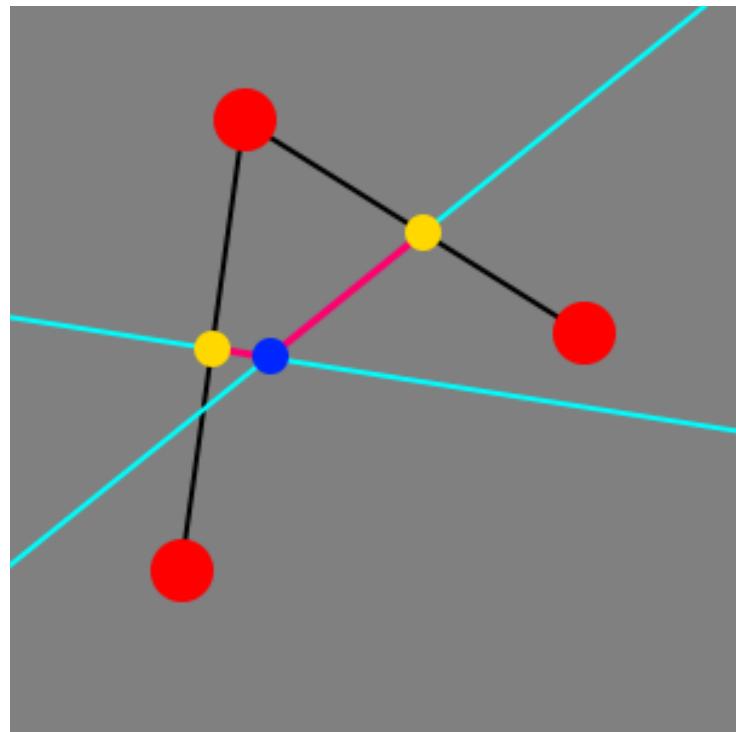
Fonte: Autoria própria

que se ligam com linha branca para tornar as arestas formando assim o polígono.

Cada polígono criado será nomeado de região e basta definir o tipo do terreno, elevação e umidade. Na lógica, marca-se todos os polígonos com o terreno de tipo oceano, depois percorre-se todos os pontos da imagem de entrada e verifica-se se cada pixel que está contido no objeto da imagem binária encontra-se nos polígonos gerados. Se os pontos participarem, marca-se o terreno como tipo terra, e em seguida checa-se em quais polígonos do tipo terra são vizinhos do tipo oceano, se sim, define-se como do tipo litoral, e também assina-se os pontos desses polígonos que encontram-se no polígono do tipo oceano.

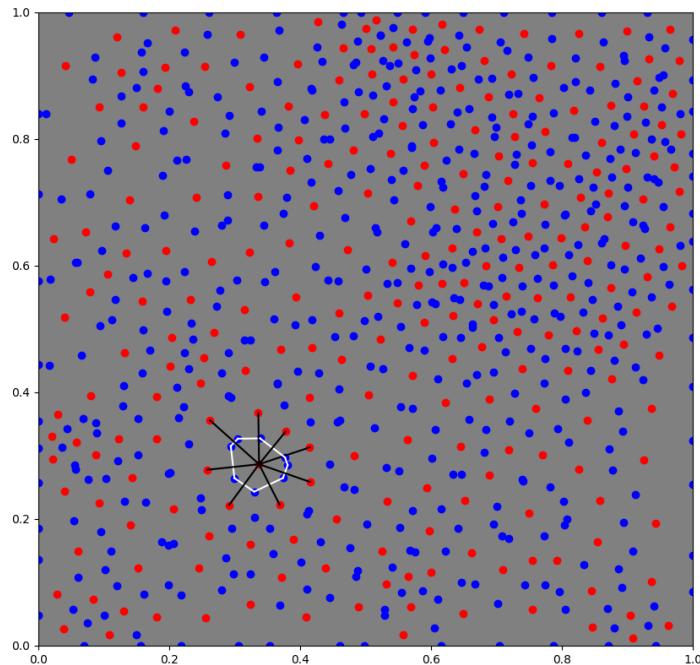
Para se calcular a elevação dos polígonos primeiramente é calculado a elevação das arestas de todos os polígonos, isso é feito a partir de uma busca em profundidade que possui o inicio em todas as arestas que tocam a borda do gráfico. A toda nova iteração, as arestas que tocam os cantos são atribuídas com o valor 0, e assim defini-se uma evolução da altura conforme chega-se ao centro do mapa. Após isso é feito o calculo de redistribuição de elevação, que com base na elevação calculada anteriormente é feito uma lista ordenadas pela elevação, começando da menor para a maior. Cada item da lista é enumerado a partir de um índice i e com essa enumeração é calculado usando a Equação (3.1). Após isso é calculado a Equação (3.2) sendo X a elevação do vértice. E por final para calcular a elevação do polígono é feito a média dos vértices que pertencem ao polígonos.

Figura 31 – Ilustração do cálculo para definir a localização dos vértices.



Fonte: Autoria própria

Figura 32 – Ilustração do diagrama de Voronoi.



Fonte: Autoria própria

$$Y = i / \text{total de indices} \quad (3.1)$$

$$X = \text{sqrt}(\text{fator}) - \text{sqrt}(\text{sqrt} * (1 - Y)) \quad (3.2)$$

Para criação dos rios busca-se os vértices do tipo terra e litoral que possuem uma elevação mínima definida, ou se os vértices se localizam próximos de um lago, então armazena-se em uma lista. Seleciona-se de forma pseudoaleatória uma quantidade de vértices e verifica-se se o tipo de terreno dos vizinhos dos vértices e das arestas que foram tocados pelos vértices. Para cada um desses é verificado se o tipo de terreno é terra ou litoral. Após isso, busca-se em todas as arestas a dona do vértice, ao encontrar é marcado como sendo um rio.

Na geração de umidade é atribuído um valor para cada vértice que possui uma aresta do tipo rio, ou se é vizinho de um lago, e é colocado esse item em uma fila. Logo em seguida, busca-se em profundidade, em todos os vértices adjacentes dos itens da fila, e é calculado uma nova umidade a partir de uma multiplicação de um fator com o valor da umidade anteriormente atribuída, e verifica-se se os vértices adjacentes possuem uma umidade maior que a nova umidade calculada, se forem é colocado o vértice adjacente na fila e é atribuído o valor da nova umidade. Para terrenos do tipo oceano é atribuído o valor máximo de umidade. Por fim, é atribuído o valor da umidade para os polígonos e é redistribuído a umidade a partir de uma lista polígonos com umidades ordenadas, para cada item i executa-se a Equação (3.3) e assim é calculado o valor final da umidade.

$$i/\text{tamanho}(lista) - 1 \quad (3.3)$$

Para assinalar os biomas verifica-se o tipo do terreno, altura e a umidade, cada um desses parâmetros liga-se um determinado bioma. E para atribuição final aplica-se a função descrita na Figura 5 porém edita-se para chegar nos resultados da Tabela 2 (PATEL, 2010).

Para gerar o mapa 3d é gerado uma imagem denominada de mapa de altura, construído com base nos dados de elevações do grafo. Gera-se uma imagem com tonalidades de cinza sendo o pixel branco com valor 255 o ponto mais alto do mapa e o pixel preto com valor 0 o ponto mais baixo. A Figura 33 ilustra uma saída de mapa de altura.

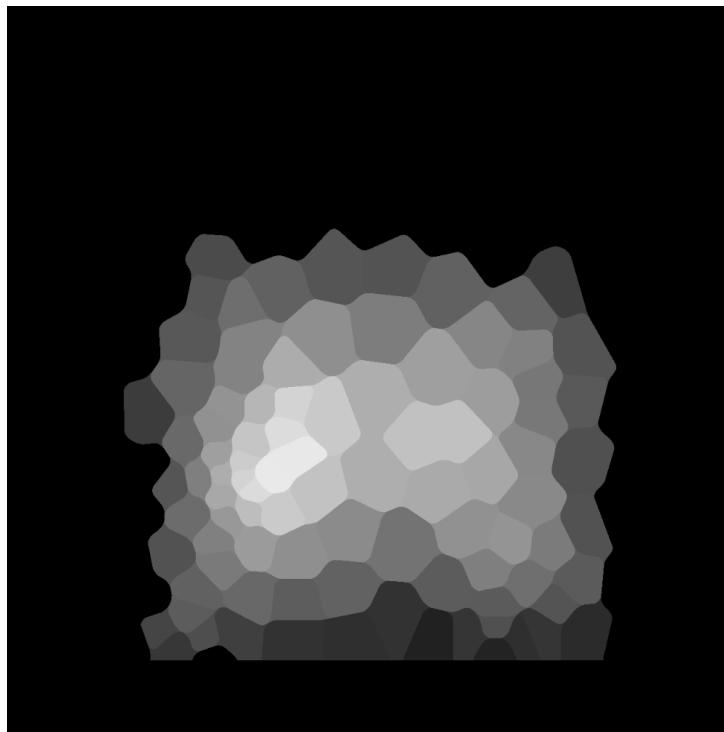
3.2.3.2 Unity - Mapa 3d

Para demonstrar uma aplicação das imagens utilizou-se o motor gráfico Unity. Criou-se uma automatização entre a geração do mapa e a atualização no projeto Unity. Este processo pode ser dividido em três partes sendo elas o terreno, minimapa em 3d e a jogabilidade.

Tabela 2 – Relação entre umidade e elevação para biomias

Zona de Elevação	Zona de Umidade				
	6 (úmido)	5	4	3	2
4 (alto)		NEVE		TUNDRA	DESNUDO
3		TAIGA		ARBUSTIVO	DESERTO TEMPERADO
2	FLORESTA TROPICAL TEMPERADA	FLORESTA DECÍDUA TEMPERADA		GRAMADO	DESERTO TEMPERADO
1 (baixo)	FLORESTA CHUVOSA TROPICAL	FLORESTA SAZONAL TROPICAL	GRAMADO		DESERTO SUB-TROPICAL

Figura 33 – Exemplo de mapa de altura.



Fonte: Autoria própria

3.2.3.2.1 Terreno

Para atualizar o terreno usou-se primeiramente o pacote *Terrain Tools* que facilita pois é possível usar um mapa de altura em png ou raw para criar um terreno, ao executar cria-se um terreno no qual o pixel mais branco (255) corresponde a altura mais alta.

Porém esse processo era manual e a ideia era que fosse automatizado, logo utilizou-se um algoritmo ligado a câmera do personagem que toda vez em que inicia-se a cena atualiza-se o terreno com a nova imagem de mapa 3d em raw. O algoritmo abre a imagem e percorre anotando o relevo proporcional a uma matriz e depois aplica no terreno.

Além da funcionalidade de relevo adicionou-se um pacote para alterar a textura de acordo com a altura para diferenciar o oceano da ilha e o bioma de floresta. Utilizou-se o pacote denominado de *Terrain Toolkit 2017*, no script é carregado as texturas e definido alguns parâmetros para atualizar com o terreno.

3.2.3.2.2 Minimap

A saída da geração procedural visa dois principais resultados, sendo eles: o mapa de altura para definir o terreno e o mapa 2d que pode ser usado como minimapa.

Para usar o mapa 2d como minimapa de forma automatizada foi necessário adicionar nos *scripts* uma conversão de uma imagem em formato PNG para um Sprite (2D e UI)

e atualizar nas propriedades do objeto de jogo. Possibilitando a localização de um ícone representando o jogador nesse minimapa por meio da projeção das coordenadas 3D do mundo do jogo em um mapa 2D.

3.2.3.2.3 Jogabilidade

Criou-se um personagem com movimentação para andar pelo mapa baseado no vídeo no Youtube de [BloxyDev \(2022\)](#). Adicionando os objetos e o script principal além de criar uma tag de chão para detectar a colisão é possível jogar no mapa andando em todas as direções e pulando.

3.2.4 Testes

Será feito um teste baseado no subtópico [Métricas e técnicas](#) no qual irá comparar uma combinação de imagens podendo conter a imagem de entrada do algoritmo de geração procedural ou alguma saída, mapa de altura ou mapa 2d. Em maioria, será utilizado a métrica IoU pois esta é a que pior classifica nosso modelo e também facilita a percepção de melhora, metrificando a semelhança obtida do contorno desejado. Portanto quanto maior essa métrica maior a compatibilidade com o contorno inicialmente proposto.

Dentre essas métricas, será utilizado também uma métrica para o desfoque encontrado em uma imagem, aplica um filtro para detectar bordas em x e y, calcula o gradiente e resulta o desvio padrão. Outra métrica utilizada nos testes foi a duração do tempo no código, medida em segundos e apenas contabilizando a parte de geração procedura dos mapas.

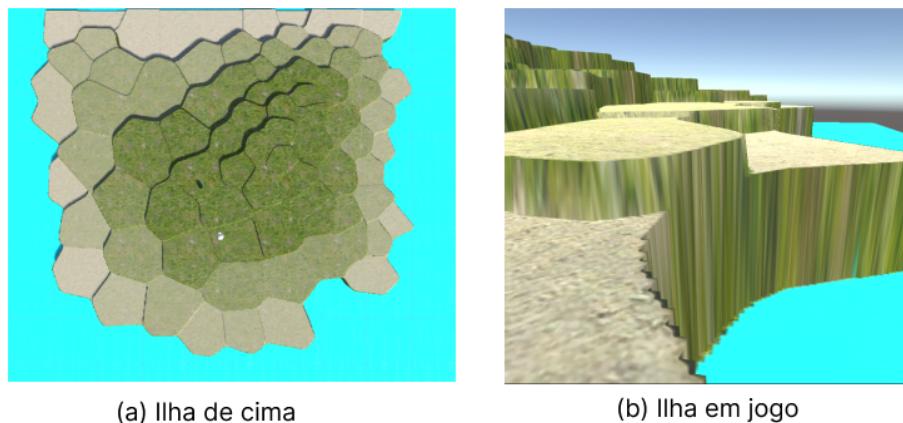
Criou-se um teste genérico para reutilizar em diversos testes e facilitar a reprodução caso algo mude. É definido na chamada do teste quais imagens serão feito as métricas, com isso é possível definir todas combinações possíveis. Para cada combinação será executado 5 imagens de entrada pré-definidas e em cada imagem irá percorrer um laço baseado em algum modificador sendo eles, tamanho do filtro de desfoque, quantidade de pontos ou tamanho da borda no mapa 2d, em cada iteração do parâmetro modificador será executado três vezes.

Em cada iteração de todos esses laços aninhados será gerado proceduralmente algum mapa ou os dois, processado as métricas e armazenado, após isso será calculado a média das métricas e criado uma tabela em [L^AT_EX](#) como resultado.

3.2.5 Pós processamento

Analisando-se o resultado gerado no Unity do mapa 3d na Figura 34 percebe-se que não existe uma harmonia entre os biomas e por isso gerou-se uma solução aplicando um desfoque na imagem de mapa de altura para suavizar a mudança de biomas.

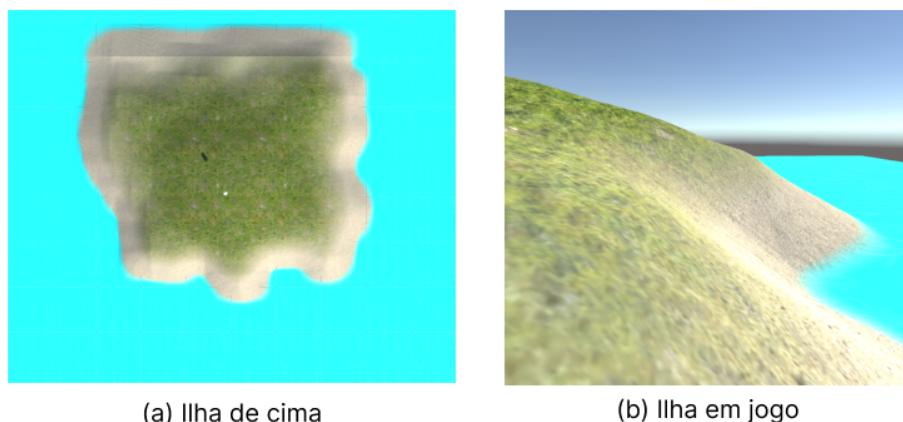
Figura 34 – Resultado no Unity sem usar filtro de desfoque no mapa de altura.



Fonte: Autoria própria

Aplicando um filtro — ou kernel — de desfoque com tamanho 100x100, obteve-se o resultado ilustrado na Figura 35. Porém se analisar a Tabela 3 pode-se observar que o resultado piorou as métricas, conclui-se que o desfoque piorou a qualidade de equivalência de contornos, aumentando um problema de escala, visto que apenas o erro do mapa de altura foi encontrado. Na Figura 36 observa-se uma representação visual dos conjuntos definidos no subtópico [Classificação de conjuntos](#).

Figura 35 – Resultado no Unity usando filtro de desfoque no mapa de altura.

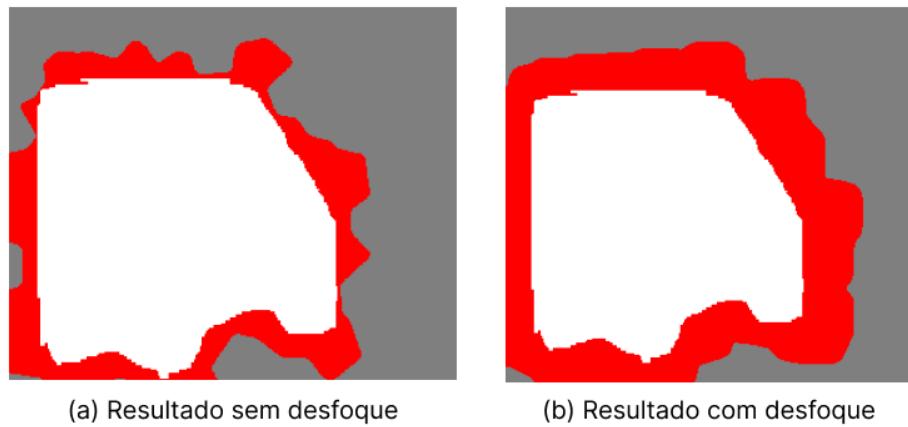


Fonte: Autoria própria

Tabela 3 – Resultados dos testes entre imagem de entrada e mapa de altura

Tamanho filtro	Desfoque	IoU	FDR	FNR
0	56.05645	0.71056	0.28944	0.0
100	10.2862	0.55579	0.44421	0.0

Figura 36 – Ilustração da classificação de conjuntos entre a imagem de entrada e o mapa de altura



Fonte: Autoria própria

Logo para tratar esse problema adotou-se uma solução baseada em redimensionar a imagem e adicionar uma borda para que diminui-se assim o tamanho do mapa de altura. Portanto foi redimensionado uma imagem 1000x1000 para 800x800, reduzindo em 20% e depois adicionado uma borda de cor preta em todos os lados de tamanho 100, logo retorna ao tamanho original. E para encontrar o melhor tamanho do filtro de desfoque para essas condições executou-se os testes alterando o tamanho do filtro após a aplicação da técnica de diminuir o tamanho do mapa. Os resultados referentes aos testes encontram-se na Tabela 4. Para selecionar o melhor resultado definiu-se a questão de ter a menor métrica de desfoque — que significa ter mais desfoque na imagem — e com a métrica IoU que varie no máximo 1% da métrica sem o filtro de desfoque. Logo seleciona-se o resultado de filtro de tamanho 80 pois a métrica IoU tem 79% assim como o tamanho de filtro 0. Percebe-se também que com a diminuição da borda resultou no compartilhamento entre os erros da imagem de entrada com o mapa de altura.

Após melhorar o resultado do mapa de entrada é necessário ajustar o mapa 2d visto que esse deve ser bem parecido com o mapa de altura, para informar por exemplo a localização exata do personagem em jogo com visualização do mapa inteiro. Observa-se na Tabela 5 os resultados obtidos dos testes com o tamanho da borda adicionada ao mapa 2d e o maior valor de IoU obtido foi 60.

3.2.6 Interface gráfica

Utilizou-se a biblioteca PyQt5 para criar uma interface gráfica na qual o usuário poderá interagir e criar um mapa a partir da seleção do contorno detectado pelo modelo de IA.

Tabela 4 – Resultados dos testes entre imagem de entrada e output3d

Tamanho filtro	Desfoco	IoU	FDR	FNR
100	11.35778	0.76178	0.22752	0.01707
90	11.88265	0.76887	0.2162	0.02281
80	12.37601	0.79299	0.18761	0.02824
70	13.59314	0.79101	0.18298	0.03676
60	14.75522	0.80283	0.16196	0.04787
50	16.29726	0.81851	0.13458	0.06073
40	18.35148	0.81608	0.12325	0.07635
30	21.52558	0.81866	0.10467	0.0935
20	27.79682	0.8183	0.08627	0.11246
10	38.13084	0.80744	0.07181	0.13757
0	48.89338	0.79895	0.05978	0.15749

Tabela 5 – Resultados dos testes entre mapa 2d e mapa de altura

Tamanho borda	IoU	FDR	FNR
0	0.82075	0.02435	0.16192
10	0.84148	0.02679	0.13833
20	0.86438	0.03616	0.10622
30	0.87938	0.04263	0.0846
40	0.89456	0.05307	0.0578
50	0.90031	0.06612	0.03827
60	0.90771	0.07929	0.0152
70	0.89229	0.10441	0.00408
80	0.86867	0.13132	1e-05
90	0.84037	0.15963	0.0
100	0.81396	0.18604	0.0

Esse módulo é responsável para conectar todas as partes e obter o mapa. Portanto é necessário abrir uma imagem do diretório local, carregar e disparar a execução do processo de segmentação de imagem. Após o resultado da IA, permitir a seleção do contorno, criar uma imagem binária a partir do contorno e envia-lá como argumento na geração procedural de mapas.

Além disso para promover a usabilidade utilizou-se um *loading* específico para PyQt5 e para isso teve-se que usar threads, criando classes para rodar as tarefas de forma separada e síncrona.

4 Resultados

Neste capítulo é apresentado os resultados obtidos da implementação e discutido o impacto e possíveis causas.

4.1 Apresentação

Para apresentar os resultados será abordado tabelas contendo testes com todas combinações definidos nas Tabelas 6 a 8. Em adição com a ilustração da última execução de cada combinação representada na Figura 37.

Tabela 6 – Resultados dos testes entre imagem de entrada e mapa 2d

Pontos	IoU	Acc	F1	MCC	FDR	FNR	Duração
50	0.71361	0.86085	0.8323	0.74079	0.2762	0.01923	5.33293
100	0.80033	0.9151	0.88857	0.82809	0.17938	0.0292	10.0857
150	0.83825	0.93506	0.91168	0.86398	0.13792	0.03125	15.32477
200	0.85695	0.94434	0.92268	0.88059	0.10822	0.04269	20.22667
250	0.868	0.94946	0.92908	0.89029	0.09454	0.04525	27.26459
300	0.87405	0.95241	0.93263	0.89612	0.08326	0.04984	33.46705

Tabela 7 – Resultados dos testes entre imagem de entrada e mapa de altura

Pontos	IoU	Acc	F1	MCC	FDR	FNR	Duração
50	0.67955	0.83753	0.80827	0.70471	0.31036	0.02058	5.29547
100	0.74929	0.88637	0.85581	0.77953	0.23518	0.02501	9.822
150	0.79092	0.91133	0.88236	0.82054	0.18669	0.0319	15.43131
200	0.80368	0.91818	0.89056	0.83234	0.17252	0.03302	21.06372
250	0.82316	0.92831	0.90248	0.85016	0.14821	0.03784	25.92615
300	0.82598	0.93022	0.90415	0.85286	0.14182	0.04223	32.75651

Tabela 8 – Resultados dos testes entre mapa 2d e mapa de altura

Pontos	IoU	Acc	F1	MCC	FDR	FNR	Duração
50	0.91515	0.9589	0.95559	0.91721	0.06528	0.0222	5.0949
100	0.90352	0.95708	0.94924	0.91312	0.08082	0.01838	10.08713
150	0.90342	0.9592	0.94915	0.91636	0.08265	0.01631	14.82656
200	0.90421	0.96087	0.94955	0.91901	0.08312	0.01487	20.51247
250	0.90505	0.96207	0.95004	0.92088	0.08284	0.0142	26.63407
300	0.90357	0.96252	0.94921	0.92112	0.08565	0.01276	33.59293

Na Figura 38 é demonstrado todos os passos contídos na execução do programa gerado em Python.

Figura 37 – Resultados da última execução de cada combinação de imagens.



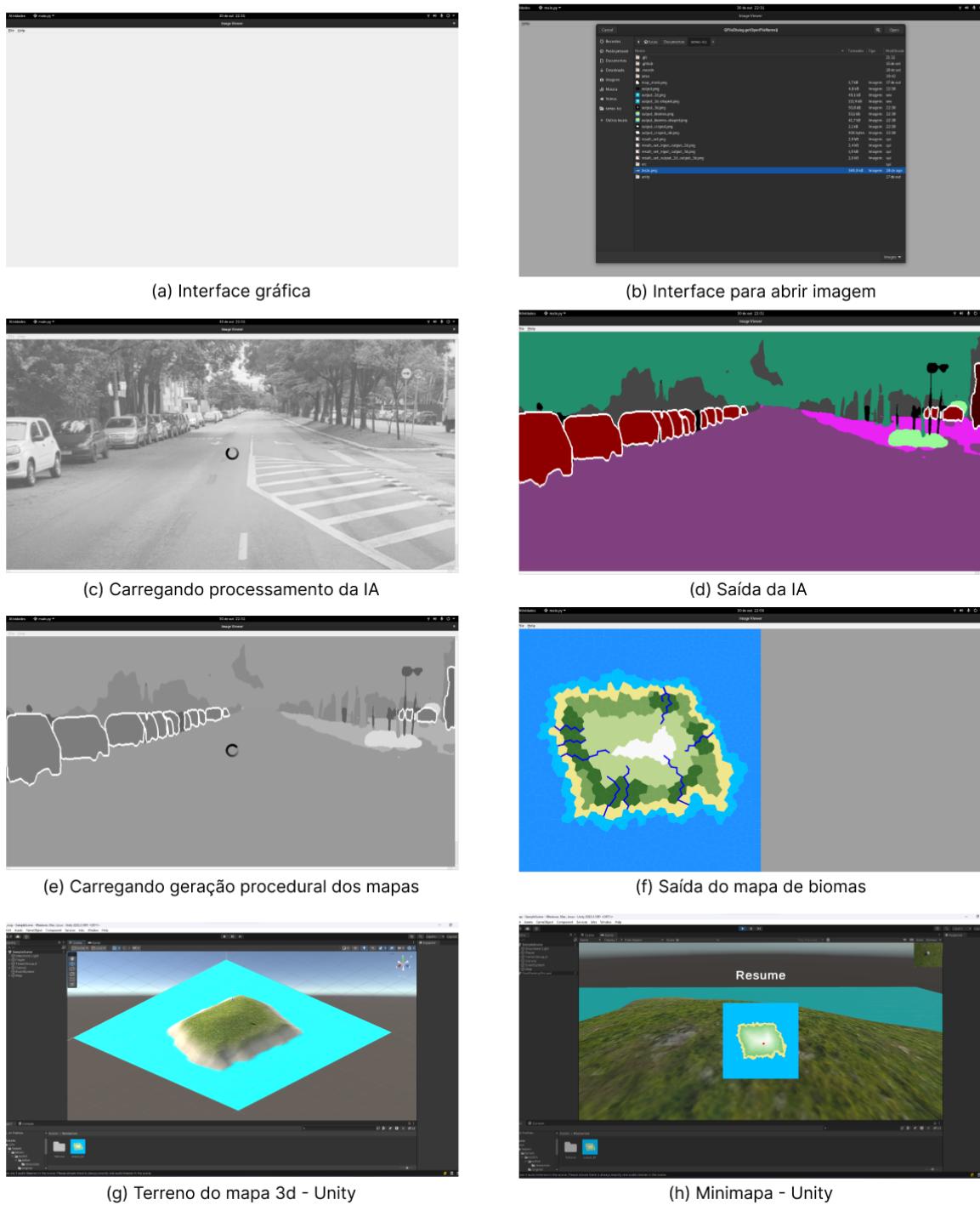
Fonte: Autoria própria

4.2 Analise dos resultados

Percebe-se com os dados das Tabelas 6 a 8 que os resultados são bem parecidos e que quanto mais pontos forem usados melhor a precisão em todas as métricas, porém demora mais segundos em média para executar. Portanto deve-se analisar o cenário para o processamento e qual o melhor custo benefício.

Isso se comprova nas ilustrações da última execução — com 300 pontos — de cada combinação de imagens nas Figura 38, percebe-se que os erros ilustrados com cores vermelha e verde são poucos e os acertos em branco e cinza prevalecem na grande maioria da imagem.

Figura 38 – Passos do programa final.



Fonte: Autoria própria

5 Considerações finais

Neste capítulo é desenvolvido as conclusões do trabalho e trabalhos futuros para evoluir com a ciência na tentativa de conseguir resultados melhores.

5.1 Conclusão

Esta monografia apresentou uma solução capaz de cria mapas 2d e 3d a partir de um contorno de uma imagem utilizando segmentação panóptica. Pode-se dividir em 4 fases sendo elas: segmentação da imagem, seleção de um contorno, geração procedural dos mapas e a automação para o terreno no Unity.

Portanto pode-se afirmar que o objetivo do trabalho foi concluído com um resultado satisfatório. Porém pelo fato de existir uma proporção entre adicionar mais pontos e melhorar os resultados junto com o tempo de execução, pode ser um problema em algum cenário com um equipamento com processamento inferior. Além do fato do fato da inteligência artificial ter sido desenvolvimento usando a biblioteca CUDA Toolkit que pode limitar ainda mais o devido as condições do cenário ideal.

Com esse contexto, a proposta apresentada pode ser usada para resultados melhores, provavelmente utilizando algoritmos mais eficientes e multiplataforma para permitir essa funcionalidade para mais usuários. Com a automação para Unity mostra-se as duas funcionalidades podendo ser utilizado por desenvolvedores para criarem um esboço rápido de um mapa 2d/3d ou como funcionalidade para o consumidor de jogos, abrindo a possibilidade de gerar um mapa a partir de um contorno selecionado de uma foto tirada e segmentada.

5.2 Trabalhos futuros

As seguintes propostas podem ser estudadas e aplicadas para aprimorar os resultados obtidos:

- Utilizar um modelo de segmentação panóptica com suporte multiplataforma
- Utilizar paralelismo para melhorar o tempo de execução
- Utilizar rasterização para melhorar o tempo de execução
- Testar tempo de execução com outros métodos de geração procedural visando um mapa baseado em um contorno

Referências

- ALZUBAIDI, L. et al. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data*, v. 8, n. 1, p. 53, Mar 2021. ISSN 2196-1115. Disponível em: <<https://doi.org/10.1186/s40537-021-00444-8>>. Citado 6 vezes nas páginas 33, 34, 36, 37, 38 e 39.
- AMINI, A. *MIT 6.S191: Convolutional Neural Networks*. 2023. Disponível em: <https://www.youtube.com/watch?v=NmLK_WQBXB4&t=1337s>. Acesso em: 15 mai. 2023. Citado na página 27.
- AZNAG, K. et al. Binary image description using frequent itemsets. *Journal of Big Data*, v. 7, n. 1, p. 32, May 2020. ISSN 2196-1115. Disponível em: <<https://doi.org/10.1186/s40537-020-00307-8>>. Citado na página 54.
- BABICH, N. *What Is Computer Vision? How Does It Work?* 2020. <<https://xd.adobe.com/ideas/principles/emerging-technology/what-is-computer-vision-how-does-it-work/>>. Acesso em: 18 mai. 2023. Citado na página 27.
- BARLA, N. *Panoptic Segmentation: Definition, Datasets & Tutorial 2023*. 2022. V7 Labs. Acesso em: 25 jun. 2023. Disponível em: <<https://www.v7labs.com/blog/panoptic-segmentation-guide>>. Citado 2 vezes nas páginas 41 e 51.
- BIOMAS: características e tipos. [S.l.]: Maestrovirtuale.com, s. d. <<https://maestrovirtuale.com/biomas-caracteristicas-e-tipos/>>. Acesso em: 05 jun. 2023. Citado na página 22.
- BLOXYDEV. *First Person Movement Unity 2022*. 2022. Vídeo no YouTube. Acesso em: 27 nov. 2023. Disponível em: <<https://www.youtube.com/watch?v=yL2Tv72tV7U>>. Citado na página 61.
- BORA, K. *Intersection Over Union (IoU) in Object Detection and Segmentation*. 2022. url`https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/`. Acesso em: 21 out. 2023. Citado na página 44.
- BRASIL, R. N. G. *Quem inventou a inteligência artificial? Veja como nasceu uma das sensações da ciência*. 2023. Disponível em: <<https://www.nationalgeographicbrasil.com/ciencia/2023/03/quem-inventou-a-inteligencia-artificial-veja-como-nasceu-uma-das-sensacoes-da-ciencia>>. Citado na página 28.
- BROWN, S. *Machine learning, explained*. 2021. <<https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained>>. Acesso em: 11 mai. 2023. Citado na página 29.
- CHICCO, D.; JURMAN, G. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC Genomics*, v. 21, n. 1, p. 6, Jan 2020. ISSN 1471-2164. Disponível em: <<https://doi.org/10.1186/s12864-019-6413-7>>. Citado 2 vezes nas páginas 42 e 43.

CLEMENT, J. *Number of games released on Steam worldwide from 2004 to 2022*. 2023. <<https://www.statista.com/statistics/552623/number-games-released-steam/>>. Acesso em: 14 mar. 2023. Citado na página 17.

CRIANDO jogos com Unity e Azure. 2018. Acesso em: 03 Dez. 2023. Disponível em: <<https://learn.microsoft.com/pt-br/shows/on-net/building-games-with-unity-and-azure#:~:text=O%20Unity%20%C3%A9%20uma%20plataforma%20de%20desenvolvimento%20de,consoles%20de%20jogos%20ou%20at%C3%A9%20mesmo%20na%20Web>>. Citado na página 24.

DATASET, C. *Panoptic Semantic Labeling Task - PQ on class-level*. 2023. <<https://www.cityscapes-dataset.com/benchmarks>>. Acesso em: 25 mai. 2023. Citado 2 vezes nas páginas 45 e 51.

DEVELOPER, N. *CUDA Toolkit*. 2023. Acesso em: 3 Dez. 2023. Disponível em: <<https://developer.nvidia.com/cuda-toolkit>>. Citado na página 45.

DORMANS, J. Adventures in level design: Generating missions and spaces for action adventure games. Weesperzijde 190, 1097DZ Amsterdam, The Netherlands, 2010. Citado na página 21.

EDUCATIVE. *Overfitting and underfitting*. [S.l.]: Educative, 2022. <<https://www.educative.io/>>. Acesso em: 01 jun. 2023. Citado na página 34.

EMBARCADOS. *Aplicação de visão computacional com OpenCV*. 2017. Acesso em: 20 out. 2023. Disponível em: <<https://embarcados.com.br/aplicacao-de-visao-computacional-com-opencv/>>. Citado 3 vezes nas páginas 27, 51 e 54.

FERNEDA, E. Redes neurais e sua aplicação em sistemas de recuperação de informação. *Ciência da Informação*, SciELO Brasil, v. 35, n. 1, p. 41–53, 2006. Disponível em: <<https://www.scielo.br/j/ci/a/SQ9myjZWlxnyXfstXMgCdcH/>>. Citado na página 31.

FOFFANO, G. Sea of thieves: Branle-bas de combat bande de forbans. Le Café du Geek, 2020. Disponível em: <<https://lecafedugeek.fr/sea-of-thieves-branle-bas-de-combat-bande-de-forban/>>. Citado na página 17.

GHARAT, S. *What, Why and Which?? Activation Functions*. 2019. Medium. Acesso em: 24 mai. 2023. Disponível em: <<https://medium.com/@snaily16/what-why-and-which-activation-functions-b2bf748c0441>>. Citado 2 vezes nas páginas 31 e 32.

GMBH, H. *Instance Segmentation*. 2023. <<https://hasty.ai/docs/mp-wiki/model-families/instance-segmentor>>. Acesso em: 27 mar. 2023. Citado na página 26.

HAUÆCKER BERND JÄHNE, B. J. H. *Handbook of computer vision and applications*. [S.l.]: ACADEMIC PRESS, 1999. ISBN ISBN 0-12-379770-5 (set). — ISBN 0-12-379771-3 (v. 1). Citado na página 26.

- HAYKIN, S. S. *Neural Networks: A Comprehensive Foundation*. [S.l.]: Prentice Hall, 1999. Citado 3 vezes nas páginas 30, 31 e 34.
- HE, K. et al. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. Disponível em: <<http://arxiv.org/abs/1703.06870>>. Citado na página 41.
- JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. Machine learning and deep learning. *Electronic Markets*, v. 31, n. 3, p. 685–695, Sep 2021. ISSN 1422-8890. Disponível em: <<https://doi.org/10.1007/s12525-021-00475-2>>. Citado 2 vezes nas páginas 29 e 30.
- KIRILLOV, A. *Panoptic Segmentation: Task and Approaches*. [S.l.]: CVPR, 2019. <<http://feichtenhofer.github.io/cvpr2019-recognition-tutorial/>>. Acesso em: 05 jun. 2023. Citado na página 42.
- KIRILLOV, A. et al. *Panoptic Segmentation*. 2019. Acesso em: 05 jun. 2023. Citado 7 vezes nas páginas 40, 41, 42, 44, 48, 51 e 52.
- LIMA, A. Redes neurais convolucionais separáveis em profundidade. *Acervo Lima*, 11 2021. Disponível em: <<https://acervolima.com/redes-neurais-convolucionais-separaveis-em-profundidade/>>. Citado na página 47.
- LIMITED, R. C. *PyQt5: Python Bindings for Qt v5*. 2023. Acesso em: 03 Dez. 2023. Disponível em: <<https://www.riverbankcomputing.com/software/pyqt/>>. Citado na página 28.
- LIN, T. et al. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. Disponível em: <<http://arxiv.org/abs/1612.03144>>. Citado na página 46.
- LISBOA, A. O que é um jogo procedural? *Canaltech*, 2022. Disponível em: <<https://canaltech.com.br/games/o-que-e-um-jogo-procedural-228162/>>. Citado na página 18.
- MAGAZINE, A. I. *Top 7 Python Neural Network Libraries for Developers*. 2023. Acesso em: 3 Dez. 2023. Disponível em: <<https://analyticsindiamag.com/top-7-python-neural-network-libraries-for-developers/>>. Citado na página 45.
- MARKIEWICZ, M. et al. *Polygonal Map Generation for Games*. 2021. Acesso em: 05 jun. 2023. Disponível em: <<https://github.com/TheFebrin/Polygonal-Map-Generation-for-Games>>. Citado na página 55.
- MARR, B. *7 Amazing Examples Of Computer And Machine Vision In Practice*. 2019. <<https://www.forbes.com/sites/bernardmarr/2019/04/08/7-amazing-examples-of-computer-and-machine-vision-in-practice/?sh=4ee6506b1018>>. Acesso em: 18 mai. 2023. Citado na página 26.
- MARTI, L.; BARROS, T. Aprendizado profundo: Fundamentos, histórico e aplicações. In: SBC. *Anais do XIV Simpósio Brasileiro de Sistemas Colaborativos*. [S.l.], 2017. Citado 3 vezes nas páginas 30, 31 e 34.
- MCFARLAND, A. *10 melhores bibliotecas Python para GUI*. 2023. Acesso em: 3 Dez. 2023. Disponível em: <<https://www.unite.ai/pt/10-melhores-bibliotecas-python-para-gui/>>. Citado na página 28.

MEDEIROS, A. *Diagrama de Voronoi e suas aplicações em SIG*. 2013. Disponível em: <<https://clickgeo.com.br/diagrama-de-voronoi-aplicacoes-sig/>>. Acesso em: 27 nov. 2023. Citado na página 22.

MENDES, M. *Ecologia 02 - Os grandes biomas terrestres*. 2019. <<http://maxaug.blogspot.com/2019/07/ecologia-02-os-grandes-biomas-terrestres.html>>. Acesso em: 05 jun. 2023. Citado na página 25.

MOHAN, R.; GABRIELE, G. *EfficientPS GitHub Issue #23*. 2021. <<https://github.com/DeepSceneSeg/EfficientPS/issues/23>>. Acesso em: 27 nov. 2023. Citado na página 53.

MOHAN, R.; VALADA, A. *Efficientps*. 2020. Repositório do GitHub. Acesso em: 27 nov. 2023. Disponível em: <<https://github.com/deepsceneseg/efficientps>>. Citado 3 vezes nas páginas 45, 52 e 53.

MOHAN, R.; VALADA, A. Efficientps: Efficient panoptic segmentation. *International Journal of Computer Vision (IJCV)*, 2021. Disponível em: <<https://arxiv.org/abs/2004.02307>>. Citado 6 vezes nas páginas 45, 46, 47, 48, 51 e 53.

NEDEA, D. *Confusion Matrix Calculator*. 2020.
url<https://www.mdapp.co/confusion-matrix-calculator-406/>. Acesso em: 21 out. 2023. Citado na página 43.

NVIDIA. *Computer Vision*. 2023. Acesso em: 20 out. 2023. Disponível em: <<https://www.nvidia.com/pt-br/glossary/data-science/computer-vision/>>. Citado na página 27.

OPENCV. *About OpenCV*. 2023. Acesso em: 20 out. 2023. Disponível em: <<https://opencv.org/about/>>. Citado na página 28.

OPENCV. *OpenCV modules*. 2023. Acesso em: 20 out. 2023. Disponível em: <https://docs.opencv.org/3.4/d7/d1b/group__imgproc__misc.html#gaf1f55a048f8a45bc3383586e80b1f0d0>. Citado 3 vezes nas páginas 28, 51 e 54.

OPENCV. *Thresholding Operations using inRange*. 2023. Acesso em: 20 out. 2023. Disponível em: <https://docs.opencv.org/3.4/da/d97/tutorial_threshold_inRange.html>. Citado 3 vezes nas páginas 28, 51 e 54.

O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015. Disponível em: <<https://arxiv.org/abs/1511.08458>>. Citado na página 34.

PATEL, A. *Polygonal Map Generation for Games*. 2010. <<http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/>>. Acesso em: 05 jun. 2023. Citado 6 vezes nas páginas 22, 23, 24, 25, 48 e 58.

POLÍGONOS de Thiessen ou Voronoi- Como gerar e para que utilizá-los. 2018. <<https://forest-gis.com/2018/02/poligonos-de-thiessen-como-gerar-e-para-que-utiliza-los.html>>. Acesso em: 26 mar. 2023. Citado na página 21.

PROJECT, G. *GCC, the GNU Compiler Collection*. 2023. Acesso em: 3 Dez. 2023. Disponível em: <<https://gcc.gnu.org/>>. Citado na página 45.

- PYTORCH. *PyTorch*. 2023. Acesso em: 3 Dez. 2023. Disponível em: <<https://pytorch.org/>>. Citado na página 45.
- RAMESH, A. et al. *DALL·E: Creating Images from Text*. 2021. Disponível em: <<https://openai.com/blog/dall-e/>>. Citado na página 29.
- RIZZO, I. V.; CANATO, R. L. C. Inteligência artificial: funções de ativação. *Prospectus* (ISSN: 2674-8576), v. 2, n. 2, 2020. Disponível em: <<https://www.prospectus.fatecitapira.edu.br/index.php/pst/article/view/37>>. Citado na página 32.
- RODRIGUES, D. S. M. *Diagrama de Voronoi : uma abordagem sobre jogos*. Dissertação (Mestrado) — Universidade Estadual de Maringá, Maringá, 2019. Disponível em: <<http://repositorio.uem.br:8080/jspui/handle/1/6748>>. Citado 3 vezes nas páginas 18, 21 e 22.
- RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. Disponível em: <<http://arxiv.org/abs/1505.04597>>. Citado 2 vezes nas páginas 40 e 41.
- SANTANA, W. *Games vão movimentar R\$ 1 tri em 2023 e empresas estão de olho nisso*. 2022. <<https://www.infomoney.com.br/negocios/games-movimentar-r-1-tri-em-2023-empresas-de-olho/>>. Acesso em: 15 mar. 2023. Citado na página 17.
- SANTOS, P. R. S. dos. *Diagrama de voronoi: Uma Exploração nas Distâncias Euclidiana e do Táxi*. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná - UTFPR, 2016. Citado na página 21.
- SARKER, I. H. Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, v. 2, n. 6, p. 420, Aug 2021. ISSN 2661-8907. Disponível em: <<https://doi.org/10.1007/s42979-021-00815-1>>. Citado 3 vezes nas páginas 34, 35 e 37.
- SCHUMACHER, D. *Synchronized Batch Normalization*. s. d. <<https://serp.ai/synchronized-batch-normalization>>. Acesso em: 09 jun. 2023. Citado na página 47.
- SILVA, J. A. S. d.; MAIRINK, C. H. P. Inteligência artificial. *LIBERTAS: Revista de Ciências Sociais Aplicadas*, v. 9, n. 2, p. 64–85, dez. 2019. Disponível em: <<https://famigvirtual.com.br/famig-libertas/index.php/libertas/article/view/247>>. Citado na página 28.
- SIRCAR, A. et al. Application of machine learning and artificial intelligence in oil and gas industry. *Petroleum Research*, v. 6, n. 4, p. 379–391, 2021. ISSN 2096-2495. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2096249521000429>>. Citado na página 29.
- STEFANINI. *Conheça as aplicações da Inteligência Artificial no dia a dia*. 2020. Disponível em: <<https://stefanini.com/pt-br/insights/artigos/aplicacoes-da-inteligencia-artificial-no-dia-a-dia>>. Citado na página 29.
- SZELISKI, R. *Computer Vision: Algorithms and Applications*. [S.I.]: Springer, 2022. Citado na página 26.

TAYE, M. M. Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions. *Computation*, v. 11, n. 3, 2023. ISSN 2079-3197. Disponível em: <<https://www.mdpi.com/2079-3197/11/3/52>>. Citado 4 vezes nas páginas 34, 35, 38 e 39.

TECHNOLOGIES, U. *Difference Between 2D and 3D Games*. s. d. Acesso em: 03 Dez. 2023. Disponível em: <<https://unity.com/pt/how-to/difference-between-2D-and-3D-games>>. Citado na página 24.

ULKU, I.; AKAGÜNDÜZ, E. A survey on deep learning-based architectures for semantic segmentation on 2d images. *Applied Artificial Intelligence*, Taylor & Francis, v. 36, n. 1, p. 2032924, 2022. Disponível em: <<https://doi.org/10.1080/08839514.2022.2032924>>. Citado 5 vezes nas páginas 18, 39, 40, 41 e 51.

WANG, Z.; WANG, E.; ZHU, Y. Image segmentation evaluation: a survey of methods. *Artificial Intelligence Review*, v. 53, n. 8, p. 5637–5674, Dec 2020. ISSN 1573-7462. Disponível em: <<https://doi.org/10.1007/s10462-020-09830-9>>. Citado na página 42.

WANGENHEIM, A. von. *Segmentação Semântica*. [S.l.]: Lapix, 2021. <<https://lapix.ufsc.br/ensino/visao/visao-computacionaldeep-learning/deep-learningsegmentacao-semantica/>>. Acesso em: 05 jun. 2023. Citado 3 vezes nas páginas 39, 40 e 41.

WHITTAKER Diagram. [S.l.]: Procedural Content Generation Wiki, 2018. <<http://pcg.wikidot.com/pcg-algorithm:whittaker-diagram>>. Acesso em: 05 jun. 2023. Citado na página 23.

W!N, F. T. *Video game maps*. [S.l.]: For The W!n, 2023. <<https://ftw.usatoday.com/lists/video-game-maps>>. Acesso em: 4 jun. 2023. Citado na página 17.

WOSCHANK, M.; RAUCH, E.; ZSIFKOVITS, H. A review of further directions for artificial intelligence, machine learning, and deep learning in smart logistics. *Sustainability*, v. 12, n. 9, 2020. ISSN 2071-1050. Disponível em: <<https://www.mdpi.com/2071-1050/12/9/3760>>. Citado na página 29.

XU, B. et al. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015. Disponível em: <<http://arxiv.org/abs/1505.00853>>. Citado na página 32.

YANNAKAKIS, G. N.; TOGELIUS, J. *Artificial Intelligence and Games*. [S.l.]: Springer, 2018. <<http://gameaibook.org>>. Citado 2 vezes nas páginas 21 e 51.