

Lucas da Silva dos Santos
Matheus Zanivan Andrade
Rafael Nascimento Lourenço

Geração Procedural de Mapas para Jogos através de técnicas de Segmentação de Imagem

São Paulo - Brasil

2023

Lucas da Silva dos Santos
Matheus Zanivan Andrade
Rafael Nascimento Lourenço

Geração Procedural de Mapas para Jogos através de técnicas de Segmentação de Imagem

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 de técnicas de Segmentação de Imagem

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

Dedico este trabalho, de forma especial, à minha família, expressando minha gratidão pelo apoio constante ao longo deste percurso. Agradeço também aos meus amigos cuja presença foi fundamental para a concepção deste trabalho. Aos professores, meu reconhecimento por todos os momentos compartilhados e conhecimentos transmitidos ao longo do processo acadêmico.

“Toda a educação, no momento, não parece motivo de alegria, mas de tristeza. Depois, no entanto, produz naqueles que assim foram exercitados um fruto de paz e de justiça.
(Bíblia Sagrada, Hebreus 12, 11)

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.	19
Figura 2 – Diagrama de Voronoi.	24
Figura 3 – Diagrama de Voronoi separado em solo e mar	25
Figura 4 – Diagrama de Voronoi separado em solo e mar com os cantos dos polígonos indicando a direção para o litoral	26
Figura 5 – Diagrama de Whittaker	27
Figura 6 – Resultado final da geração do mapa	27
Figura 7 – Algoritmos de detecção facial e de roupas/cabelos por cor localizam e reconhecem pessoas nesta imagem	28
Figura 8 – Segmentação de instâncias de objetos pode-se delinear cada pessoa e objeto em uma cena complexa	28
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.	29
Figura 10 – Imagem conceito de uma inteligência artificial que é um robô humanoide	31
Figura 11 – Ilustração da relação entre os principais tópicos de aprendizado de máquina	32
Figura 12 – Modelo de um neurônio não-linear.	33
Figura 13 – Gráfico da função <i>Sigmoid</i> .	34
Figura 14 – Gráfico da função <i>ReLU</i> .	34
Figura 15 – Gráfico da função <i>ReLU</i> .	35
Figura 16 – Gráfico da função <i>Softmax</i> .	35
Figura 17 – Gráficos mostrando subajuste, balanceado e sobreajuste, respectivamente.	37
Figura 18 – Comparação entre uma rede neural convencional e uma rede neural profunda.	37
Figura 19 – Camadas principais de uma rede neural convolucional	38
Figura 20 – Tipos de segmentação em redes neurais convolucionais	42
Figura 21 – Exemplo de arquitetura de rede totalmente convolucional	42
Figura 22 – Arquitetura codificador-decodificador UNet	43
Figura 23 – Exemplo da classificação dos conjuntos usados nas métricas de segmentação	45
Figura 24 – Arquitetura geral do EfficientPS	48
Figura 25 – Etapas da proposta	54
Figura 26 – Resultado inicial do EfficientPS.	55
Figura 27 – Expectativa do resultado do EfficientPS.	55
Figura 28 – Resultado do EfficientPS obtido seguindo os passos da publicação no repositório.	56

Figura 29 – Passos da seleção da saída da inteligência artificial.	57
Figura 30 – Passos da seleção da saída da inteligência artificial mais performática. .	58
Figura 31 – Ilustração do processo de criação do polígono.	59
Figura 32 – Ilustração do diagrama de Voronoi.	59
Figura 33 – Exemplo de mapa de altura.	61
Figura 34 – Ilustração da classificação de conjuntos	64
Figura 35 – Resultado no Unity sem usar filtro de desfoque no mapa de altura. . .	65
Figura 36 – Resultado no Unity usando filtro de desfoque no mapa de altura. . .	66
Figura 37 – Ilustração da classificação de conjuntos entre a imagem de entrada e o mapa de altura	66
Figura 38 – Resultados da última execução de cada combinação de imagens. . . .	71
Figura 39 – Passos do programa final.	72
Figura 40 – Passos do programa final utilizando o preenchimento por inundação. .	74

Lista de tabelas

Tabela 1 – Top 15 modelos que melhor classificam pessoas com métrica PQ em segmentação panóptica	47
Tabela 2 – Relação entre umidade e elevação para biomas	62
Tabela 3 – Resultados dos testes entre imagem de entrada e mapa de altura	66
Tabela 4 – Resultados dos testes entre imagem de entrada e output3d	67
Tabela 5 – Resultados dos testes entre mapa 2d e mapa de altura	68
Tabela 6 – Resultados dos testes entre imagem de entrada e mapa 2d	69
Tabela 7 – Resultados dos testes entre imagem de entrada e mapa de altura	70
Tabela 8 – Resultados dos testes entre mapa 2d e mapa de altura	70

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

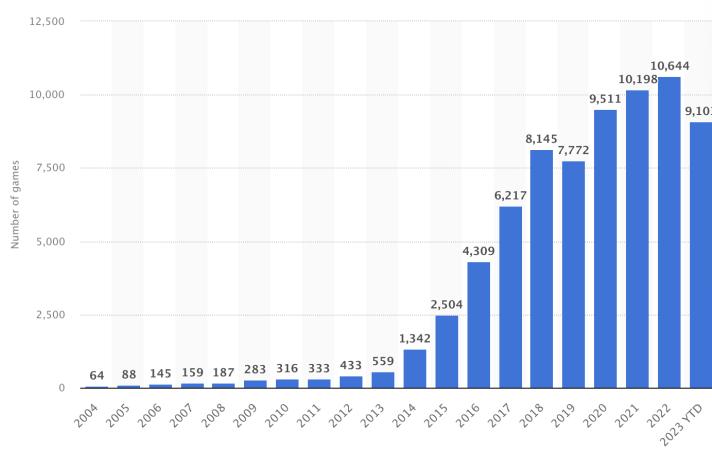
1	INTRODUÇÃO	19
1.1	Objetivos	20
2	FUNDAMENTAÇÃO TEÓRICA	23
2.1	Geração procedural de conteúdo	23
2.1.1	Diagrama de Voronoi	23
2.1.2	Geração de biomas no diagrama de Voronoi	24
2.1.3	Representação	26
2.2	Visão Computacional	28
2.2.1	PyQt5	30
2.3	Inteligência Artificial	30
2.3.1	Aprendizado de Máquina	31
2.3.1.1	Rede Neural Artificial	32
2.3.1.1.1	Rede neural convolucional	37
2.4	Trabalhos Relacionados	50
3	DESENVOLVIMENTO	53
3.1	Proposta	53
3.2	Implementação	54
3.2.1	Segmentar imagens	54
3.2.2	Selecionar contorno	56
3.2.3	Geração procedural do mapa	57
3.2.3.1	Ilha gerada no contorno	57
3.2.3.2	Unity - Mapa 3D	61
3.2.3.2.1	Terreno	63
3.2.3.2.2	Minimap	63
3.2.3.2.3	Jogabilidade	63
3.2.4	Testes	63
3.2.5	Pós-processamento	65
3.2.6	Interface Gráfica	67
4	RESULTADOS	69
4.1	Apresentação	69
4.2	Análise dos Resultados	73
5	CONSIDERAÇÕES FINAIS	75

5.1	Conclusão	75
5.2	Trabalhos Futuros	76
REFERÊNCIAS		77
APÊNDICES		85
.1	Instalação do Unity Hub	87
.2	Instalação do Miniconda	88
.3	Repositório do projeto	88
.4	Execução do projeto	89

1 Introdução

O setor de jogos digitais está em constante expansão, projetando ultrapassar os US\$ 200 bilhões em 2023, conforme previsto por [Santana \(2022\)](#). A plataforma Steam, em 2022, testemunhou o lançamento de 10.644 novos títulos, e até 6 de outubro de 2023, esse número alcançou 9.103, indicando uma notável tendência crescente, conforme revelado pela Figura 1.

Figura 1 – Número de jogos publicados na Steam.



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

No contexto dos jogos, os mapas desempenham um papel crucial, guiando os jogadores e enriquecendo suas experiências ao proporcionar uma sensação de escala e fomentar uma exploração mais rica ([W!N, 2023](#); [ZAGATA; MEDYŃSKA-GULIJ, 2023](#)).

A elaboração de mapas no universo dos jogos utiliza a técnica de geração procedural, fundamentada na criação dinâmica de conteúdo por meio de algoritmos. Inicialmente concebida para superar desafios de armazenamento, a geração procedural persiste como uma estratégia para criar conteúdos exclusivos a cada execução do programa ([KENNY, 2021](#); [FONSECA, 2022](#)).

Dentro dessa abordagem, diversas técnicas são exploradas, desde o *Perlin Noise* para criar contornos e texturas naturais, os *L-systems* para gerar vegetação, até o *Voronoi* para criar texturas e partitionar espaços 2D, e os autômatos celulares para criar cavernas em 2D, entre outras abordagens ([KŘÍŽ, 2019](#)).

Nesse contexto de geração procedural de mapas, surgem desafios consideráveis, conforme discutido por [Leite e Lima \(2015\)](#): "Um dos maiores desafios no desenvolvimento de algoritmos para geração de mapas é a dificuldade de criar cenários que sejam, ao

mesmo tempo, atraentes e diversificados, permitindo que o jogador possa explorar um novo ambiente a cada sessão de jogo" (LEITE; LIMA, 2015).

Para criar mapas diversificados, propomos estabelecer um escopo inspirado no jogo *Minecraft*, conhecido por gerar mapas proceduralmente com biomas que incluem diferentes características, como flora, fauna, relevo, temperaturas e umidade (WIKI, 2022). Nesse escopo, pretendemos gerar mapas com biomas utilizando a abordagem proposta por Patel (2010), que utiliza o diagrama de Voronoi para partitionar em regiões atribuíveis a diferentes biomas.

Adicionalmente, visando adicionar personalização e enriquecer a diversidade, planejamos implementar uma funcionalidade que permita a definição do contorno do mapa por meio da segmentação de uma imagem, originada de desenhos ou de contextos cotidianos. Para isso, empregaremos técnicas de segmentação, as quais, conforme afirmado por Team (2023), simplificam a representação ao dividir a imagem em regiões com base em características como cor, textura ou forma. Essas técnicas, úteis para rastreamento e reconhecimento de objetos, possibilitarão a segmentação de desenhos, utilizando, por exemplo, o preenchimento por inundação. Além disso, incorporaremos modelos de inteligência artificial (IA) para a segmentação de cenários mais complexos, abrangendo elementos como pessoas e veículos (TEAM, 2023; OPENCV, 2023b; ULKU; AKAGÜNDÜZ, 2022).

Portanto, o escopo de nossa pesquisa abrange a geração procedural de mapas com biomas, permitindo a escolha do contorno a partir de um desenho com técnicas de segmentação ou da segmentação de uma imagem mais complexa com modelos de IA. Esta aplicação viabilizará aos desenvolvedores a incorporação dessa nova funcionalidade em um jogo em fase de desenvolvimento, proporcionando ao usuário final a opção de selecionar ou capturar uma foto e escolher um contorno específico para gerar um mapa contendo os biomas.

Com o intuito de contribuir cientificamente, formulamos a hipótese de que quanto mais pontos o diagrama de Voronoi tiver, maior será a precisão na compatibilidade entre o mapa gerado e o contorno escolhido. Para testar essa hipótese, serão realizados testes com métricas para avaliar a qualidade da geração procedural com o contorno selecionado. Além disso, será realizado um conjunto de testes com métricas específicas para verificar a hipótese formulada: a relação direta entre a precisão na compatibilidade entre o mapa gerado e o contorno escolhido e o número de pontos no diagrama de Voronoi.

1.1 Objetivos

O objetivo principal deste trabalho é abordar um desafio inerente ao desenvolvimento de jogos digitais, mais especificamente na geração procedural de mapas. A complexidade está relacionada à criação de cenários que sejam simultaneamente atrativos

e heterogêneos, buscando proporcionar aos jogadores experiências distintas a cada sessão de jogo. A solução proposta visa aprimorar essa capacidade, concentrando-se na habilidade de identificar contornos em imagens derivadas de desenhos ou cenas cotidianas. Essa abordagem pretende possibilitar a criação de mapas contendo biomas com base nos limites predefinidos pelo usuário, visando, assim, a ampliação da diversidade e personalização dos ambientes virtuais, com potencial impacto positivo na experiência do jogador.

Adicionalmente, os seguintes objetivos específicos serão abordados:

- Selecionar conjuntos de dados contendo classes relevantes, como pessoas, carros, entre outros, para treinar um modelo IA específico para a segmentação de imagens.
- Utilizar algoritmos para criar diagrama 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 indicar o que é terreno sobre o diagrama de Voronoi.
- Gerar os biomas no diagrama de Voronoi.
- Criar testes com o intuito de mensurar a semelhança entre o contorno do mapa gerado e 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. Primeiramente, será abordado o tópico de geração procedural para criar o mapa requerido, utilizando o diagrama de Voronoi como um filtro na imagem e aplicando os biomas. Em seguida, será discutido 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. Essa prática é uma parte importante da inteligência artificial de um jogo e tem sido empregada desde a década de 1980. Essa técnica pode ser aplicada para gerar níveis, mapas, texturas, regras de jogo, histórias, entre outras coisas.

É difícil determinar qual algoritmo foi utilizado para a geração de conteúdo em jogos modernos, pois os códigos-fonte não são facilmente acessíveis. Nos jogos mais antigos, no entanto, os códigos-fonte e as estratégias utilizadas são acessíveis e bem documentados na internet. Geralmente, são empregados algoritmos de geração aleatória, classificáveis como força bruta, e são utilizados para criar estruturas ou mapas, dependendo do tipo de jogo ([DORMANS, 2010](#)).

2.1.1 Diagrama de Voronoi

Segundo [Rodrigues \(2019\)](#), o 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 no plano ([SANTOS, 2016](#)). Esse diagrama possui uma gama de utilizações, como estudar epidemias, encontrar o ponto mais próximo, calcular a precipitação de uma área, estudar os padrões de crescimento das florestas, entre outras aplicações ([POLÍGONOS..., 2018](#)).

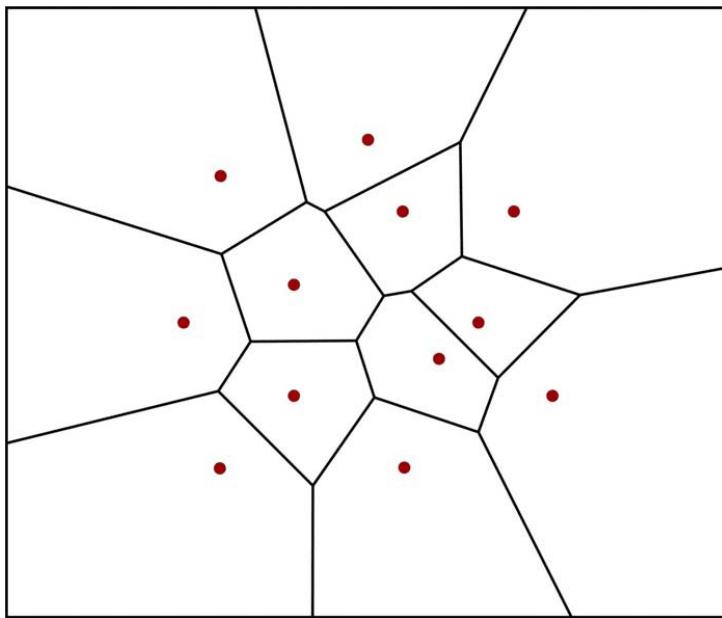
Seja um conjunto de índices $I_n = \{1, 2, 3, \dots, n\}$ e $A = \{p_1, p_2, \dots, p_n\} \subset \mathbb{R}^2$ um conjunto de pontos, onde $2 \leq n < \infty$, define-se então como região de Voronoi, descrita na Equação (2.1), o conjunto de pontos associado a p_i , onde d é a distância euclidiana:

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

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 vermelho e as retas que são perpendiculares à distância dos pontos vermelhos vizinhos.

Figura 2 – Diagrama de Voronoi.



Fonte: Medeiros (2013)

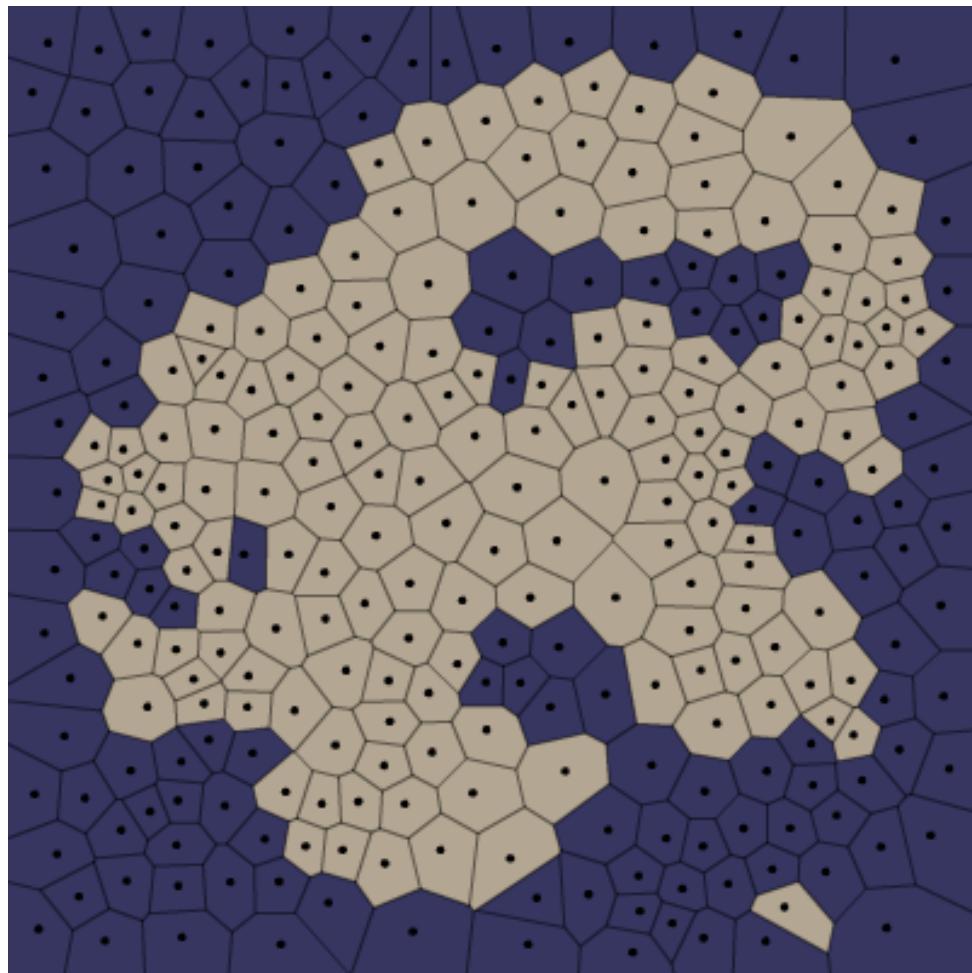
2.1.2 Geração de biomas no diagrama de Voronoi

Biomas são regiões ecológicas que possuem fauna e flora com atributos estruturais semelhantes (BIOMAS..., s. d.). Segundo Patel (2010), gera-se um mapa e seus biomas definindo-se o litoral, sendo este a borda que separa a água do solo. Na Figura 3, mostra-se essa separação, sendo os polígonos marrons representando o solo e os polígonos azuis representando 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 os polígonos indicados 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 indicam as direções das encostas.

Figura 3 – Diagrama de Voronoi separado em solo e mar



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

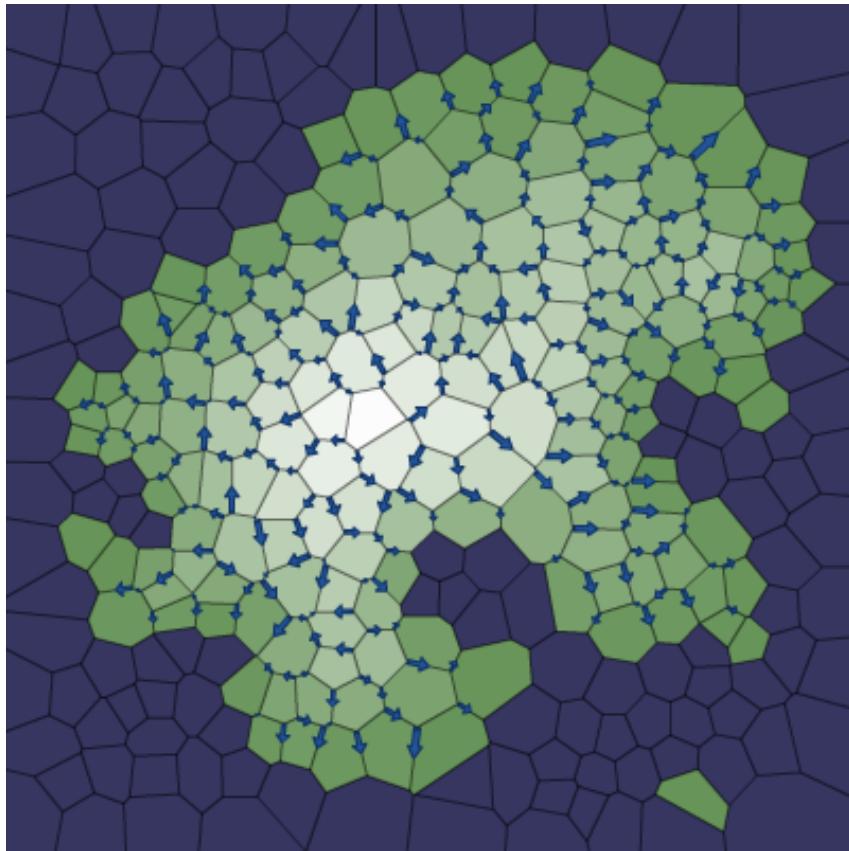
A elevação é utilizada para gerar os biomas. Por exemplo, elevações altas indicam montanhas, que, consequentemente, devem possuir neve. Adicionando mais uma camada, além da elevação, como a de umidade, podemos gerar uma variedade maior de biomas. A umidade é calculada com base na distância do polígono a 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, conforme exemplificado na Figura 5 ([WHITTAKER..., 2018](#)).

Utilizando a elevação como representante da temperatura de um bioma, é possível empregar o diagrama de Whittaker. Fazendo ajustes nesse diagrama, é possível adicionar ou remover biomas. Com essa nova camada, torna-se possível a adição de rios ao mapa ([PATEL, 2010](#)), resultando no mapa final 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



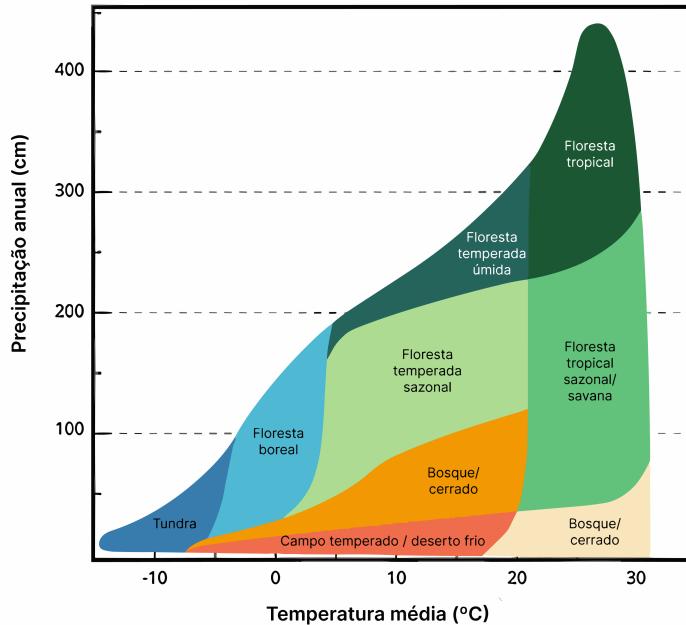
Fonte: [Patel \(2010\)](#)

2.1.3 Representação

Existem duas formas de representar os mapas criados pela geração procedural. Uma delas é a representação 2D, que possui duas dimensões de espaço, não apresenta perspectiva e é criada a partir de figuras geométricas, como 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 definido pela altura. O valor 0 (preto) representa a menor altura possível, enquanto 255 (branco) representa a maior altura possível. Isso resulta em um mapa chamado de mapa de altura. Com o mapa de altura, é possível criar uma representação 3D. Segundo [Technologies \(s. d.\)](#), os jogos 3D possuem três dimensões de espaço, utilizam perspectiva e objetos sólidos com geometria tridimensional.

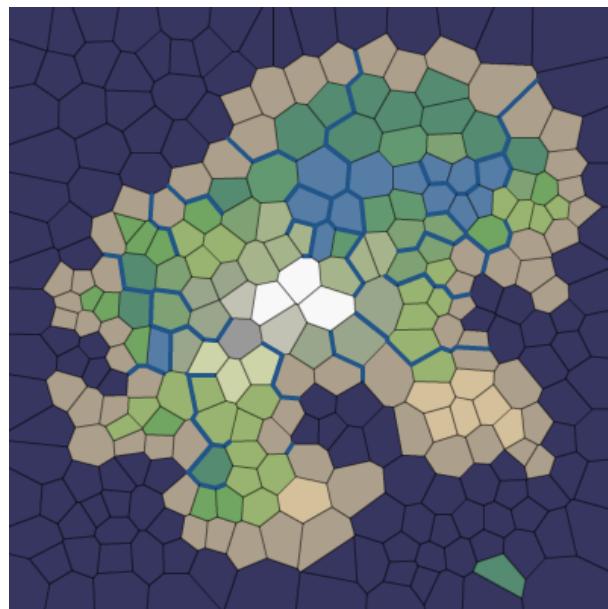
Com a integração de dois mapas distintos, torna-se viável proporcionar uma perspectiva tridimensional em um deles e uma visão superior, conhecida como minimapa, no outro. Esses mapas atuam em conjunto, complementando-se para oferecer ao jogador uma percepção de sua localização. Conforme destacado por [Microsoft \(2018\)](#), 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 integrar o Unity com o mapa de elevação, existe o pacote *Terrain Tools*, que cria um terreno a partir da imagem, e o *Terrain Toolkit 2017*, que carrega as texturas de acordo com a altura (TECHNOLOGIES, s. d.; TECHNOLOGIES, 2017).

De acordo com Correa (2015), o Unity possibilita a criação de um personagem que

oferece interação com o mundo tridimensional do Unity.

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.

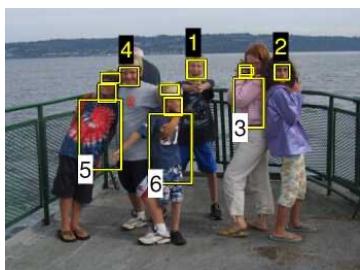
2.2 Visão Computacional

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 dedicada 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" ([SZEGLISKI, 2022](#)), nas últimas décadas ocorreram avanços significativos na busca por 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, enquanto 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 \(2023\)](#)

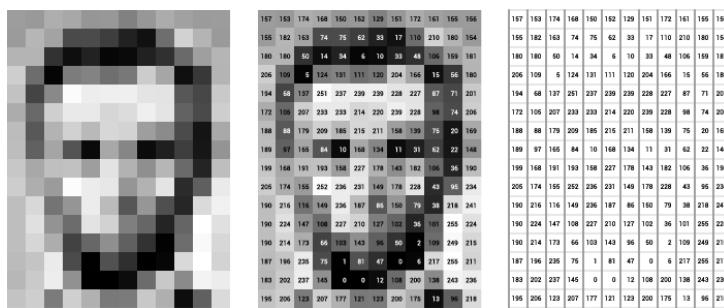
No entanto, apesar do sucesso no uso dessas técnicas, o computador ainda não consegue oferecer a mesma quantidade de detalhes na explicação de uma imagem como o

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 píxeis. 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 referentes às 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 é exemplificado na Figura 9 do presidente dos Estados Unidos, Abraham Lincoln ([AMINI, 2023](#)).

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.



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, como na 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 uma 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](#)).

Serão abordados dois algoritmos com propostas parecidas para selecionar um objeto e destacá-lo em uma imagem binária, sendo eles: selecionar imagem por cor e por inundação. Ambos utilizam a biblioteca em Python, OpenCV, uma biblioteca multiplataforma para visão computacional, com métodos para auxiliar na manipulação, por exemplo, de imagens ([OPENCV, 2023a](#)).

O método de selecionar por cor fundamenta-se em capturar 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; caso não seja, pinte como preto. Uma maneira performática de selecionar por cor é definir um espectro de cores e aplicar uma máscara 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 a mesma cor, pinta de branco, e refaz com os pixels marcados anteriormente até não ter mais 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](#)).

2.2.1 PyQt5

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

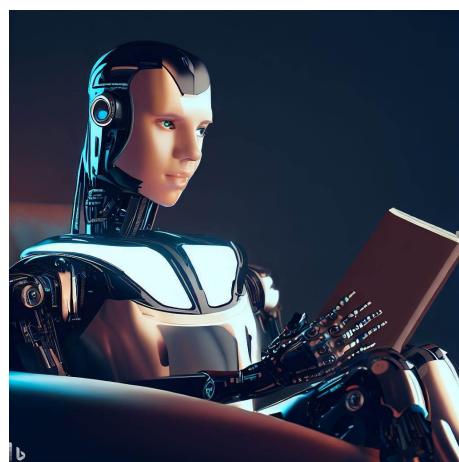
Nesta seção será abordado conceitos aninhados, sendo eles: inteligência artificial, aprendizado de máquina, rede neural artificial, rede neural convolucional, segmentação e EfficientPS.

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 da inteligência artificial foi apresentada primordialmente no artigo de Alan Turing — conhecido como o pai da computação — denominado *Computing Machinery and Intelligence*, em 1950. Outro conceito apresentado por ele foi o Teste de Turing, uma série de questionamentos que visa provar se a máquina pode executar um comportamento inteligente semelhante ao do ser humano ([BRASIL, 2023](#)).

As aplicações de IA são variadas, incluindo: controle de estoques de produtos em empresas, tanto na logística interna quanto na externa; direção autônoma de carros; reconhecimento facial com base em vídeos ou fotos; criação de imagens a partir de textos, como na Figura 10 — esta imagem foi gerada a partir da descrição: "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"—, uma imagem criada usando a plataforma DALL · E; e até mesmo a classificação de objetos e/ou pixels em imagens na área de segmentação ([STEFANINI, 2020](#); [RAMESH et al., 2021](#)).

Figura 10 – Imagem conceito de uma inteligência artificial que é um robô humanoide



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

2.3.1 Aprendizado de Máquina

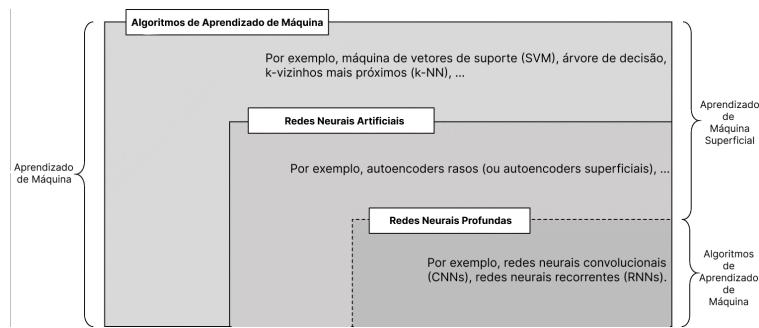
Segundo Woschank, Rauch e Zsifkovits (2020), o aprendizado de máquina é uma subcategoria da inteligência artificial que se refere à detecção de padrões importantes em uma base de dados. As ferramentas utilizadas aumentam a eficiência dos algoritmos no tratamento de grandes bases de dados.

Portanto, essa técnica permite que o computador melhore seus resultados com base na experiência, indicando uma relação direta entre a quantidade de dados consumidos pelo programa e a qualidade da solução do problema (BROWN, 2021).

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

É possível observar uma hierarquia, ilustrada na Figura 11, entre o aprendizado de máquina e os principais termos, sendo eles: algoritmos de aprendizado de máquina que estão em um subconjunto chamado de aprendizado de máquina superficial, redes neurais artificiais que estão contidas no anterior e em um subconjunto chamado de aprendizado de máquina profundo, e redes neurais artificiais profundas que estão contidas em ambos os subconjuntos (JANIESCH; ZSCHECH; HEINRICH, 2021).

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



Fonte: Janiesch, Zschech e Heinrich (2021)

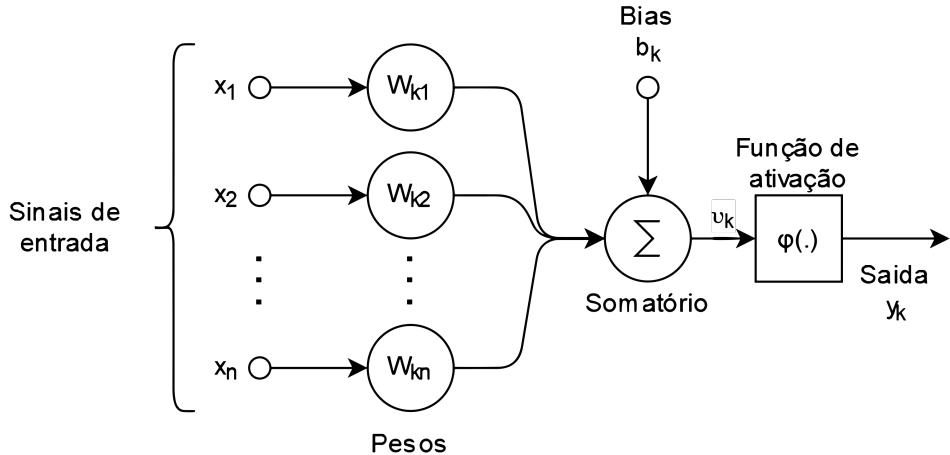
2.3.1.1 Rede Neural Artificial

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

Observa-se na Figura 12 o funcionamento de um neurônio k . Os sinais de entrada 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):

Figura 12 – Modelo de um neurônio não-linear.



Fonte: 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 , que, somado ao *bias* ou viés b_k , manipula a saída y_k do neurônio. Essa soma é colocada em uma função não-linear, nomeada de função de ativação $\varphi(\cdot)$. Essas funções mapeiam a saída em um intervalo $[0, 1]$ ou $[-1, 1]$. A função de saída pode ser representada pela 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 são ajustados 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 deslocar 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 quantificar o erro encontrado na 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 unidades de processamento chamados camadas. A primeira e a última camadas são nomeadas de camada de entrada e camada de saída, respectivamente, e as demais são chamadas 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).

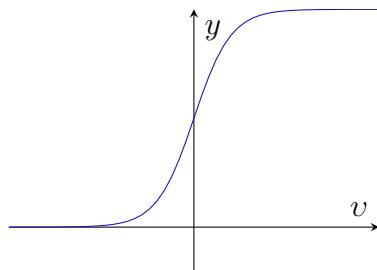
Função de ativação

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 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.

Figura 13 – Gráfico da função *Sigmoid*.

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

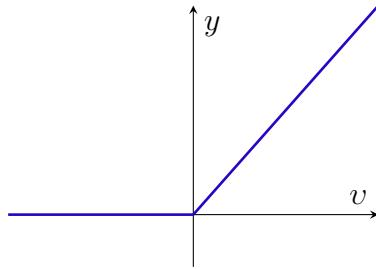


Fonte: Criação própria

2. 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 *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).
3. 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

Figura 14 – Gráfico da função *ReLU*.

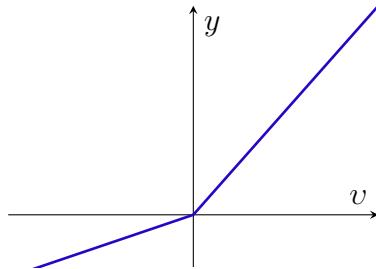
$$\varphi(v) = \max(0, v) \quad (2.5)$$



Fonte: Criação própria

Figura 15 – Gráfico da função *ReLU*.

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



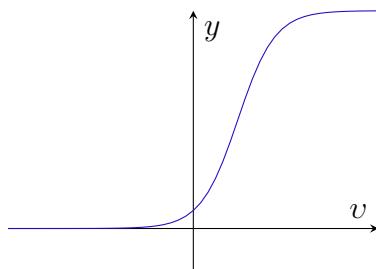
Fonte: Criação própria

sem o problema da retropropagação. ([GHARAT, 2019](#)).

4. 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 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.

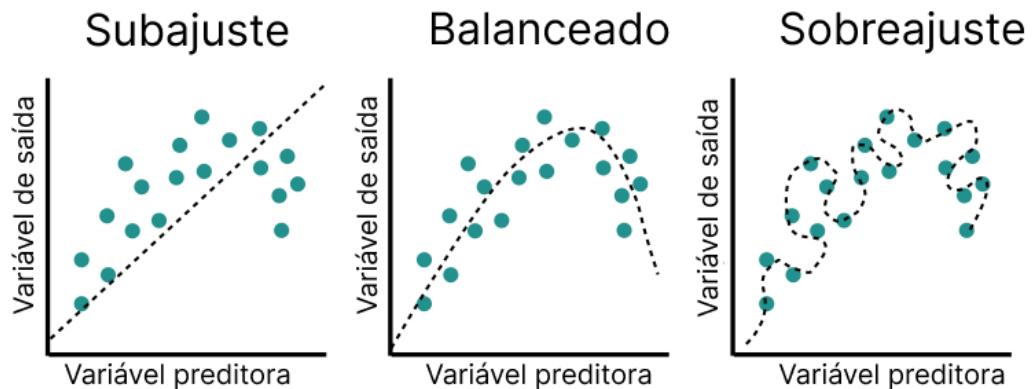
$$H(p, y) = \sum_{i=1}^N \max(0, m - (2y_i - 1)p_i) \quad (2.10)$$

Regularização

Quando se modela uma arquitetura de redes neurais, pode-se chegar a três casos, sendo eles: subajuste (*underfit*), balanceado (*optimal*) e sobreajuste (*overfit*). O sobreajuste ocorre 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). É possível visualizar na Figura 17 a relação do subajuste, balanceado e sobreajuste com uma função de regressão linear criada a partir da predição de cada um dos resultados do treinamento.

Figura 17 – Gráficos mostrando subajuste, balanceado e sobreajuste, respectivamente.

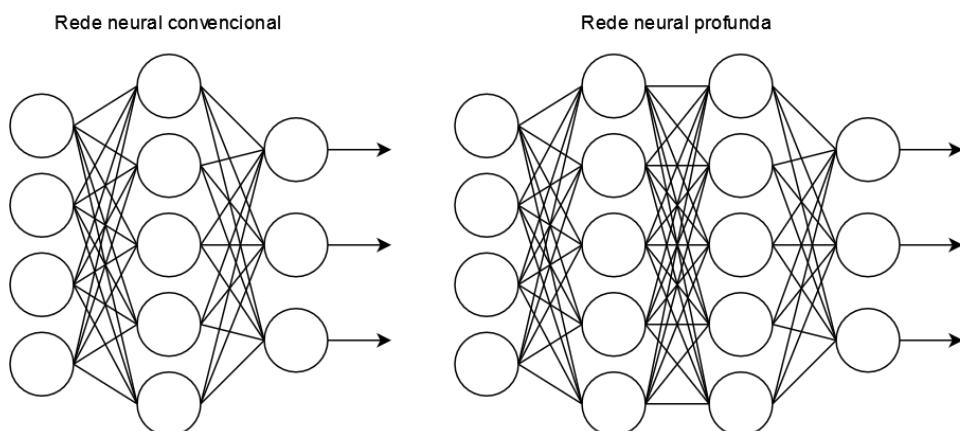


Fonte: [Educative \(2022\)](#)

Rede Neural Profunda

A principal diferença entre uma rede neural artificial e uma rede neural profunda reside na quantidade de camadas. Enquanto a rede neural profunda possui várias camadas de processamento, a rede neural artificial apresenta menos camadas (MARTI; BARROS, 2017 apud HAYKIN, 1999).

Figura 18 – Comparaçāo entre uma rede neural convencional e uma rede neural profunda.



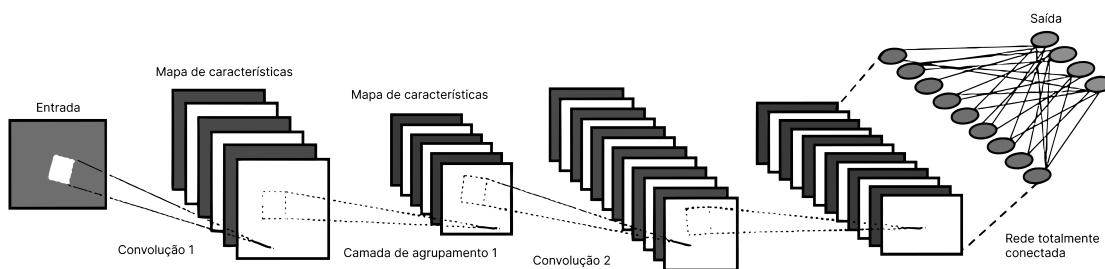
Fonte: Criação própria

2.3.1.1.1 Rede neural convolucional

Uma rede neural convolucional é análoga à rede neural artificial, isto é, é feita de neurônios que otimizam o aprendizado por meio deles mesmos. A principal diferença é que a rede neural convolucional é amplamente utilizada em soluções que detectam padrões em imagens. Portanto, existem funcionalidades específicas da própria arquitetura voltadas para essa tarefa (O'SHEA; NASH, 2015).

A arquitetura básica de uma rede neural convolucional possui as seguintes camadas: convolucional, de agrupamento e totalmente conectada, conforme ilustrado na Figura 19 (SARKER, 2021).

Figura 19 – Camadas principais de uma rede neural convolucional



Fonte: Sarker (2021)

Camada convolucional

Conforme descrito por Taye (2023), a camada convolucional é essencial para esse tipo de arquitetura. Ela utiliza um filtro — ou kernel — que é aplicado na imagem e direcionado para o próximo neurônio. Esse filtro é uma matriz de números que realiza uma operação em todos os pixels da imagem — também representada por matriz(es) —. As informações cruciais para esse filtro são: tamanho, largura e pesos. Essa abordagem é utilizada para extrair características com uma base matemática, criando uma relação direta entre um pixel e os pixels ao seu redor. Os pesos começam de forma pseudoaleatória e são ajustados ao longo do aprendizado. O resultado dessa camada é chamado de mapa de características. O tamanho da saída é baseado na fórmula abaixo, considerando os tamanhos I da imagem, F do filtro e S da saída (TAYE, 2023).

$$\begin{aligned} Ix - Fx + 1 &= Sx \\ Iy - Fy + 1 &= Sy \end{aligned} \tag{2.11}$$

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* — é usado para especificar a distância, em pixels, entre os passos da camada. No exemplo mencionado anteriormente, esse parâmetro é definido como 1, o que significa que a matriz selecionada avança 1 pixel para a direita a cada passo. Esse valor altera o tamanho da matriz resultante ([SARKER, 2021](#)).

O preenchimento — ou *padding* — é uma técnica empregada para manter o mesmo tamanho da entrada, adicionando bordas com zeros antes das operações da camada. Isso resulta em uma matriz de saída com a mesma dimensão da matriz original. A técnica é utilizada para evitar a perda de detalhes nas bordas das imagens durante o processamento de uma camada ([SARKER, 2021](#)).

Camada de agrupamento

A camada de agrupamento — ou *pooling* — tem como função primordial reduzir o tamanho do mapa de características, preservando, contudo, os padrões mais relevantes. Entre os aspectos essenciais dessa camada estão o tamanho do agrupamento e a operação a ser realizada. Um dos principais problemas dessa camada é que ela identifica apenas a presença e a localização das características, sem preservá-las integralmente. Dependendo da operação escolhida e do número de camadas, isso pode resultar em perda de características importantes, afetando negativamente o desempenho final da predição ([SARKER, 2021](#)).

Existem vários tipos de agrupamento, sendo os mais comuns o agrupamento máximo, o agrupamento médio e o agrupamento global médio. Esses tipos são explicados abaixo, com exemplos baseados em [Alzubaidi et al. \(2021\)](#).

Agrupamento máximo

Esta técnica define o resultado com base no valor máximo encontrado dentro do tamanho do agrupamento. Um exemplo pode ser ilustrado utilizando um mapa de características de tamanho 4x4 e um agrupamento de tamanho 2x2.

$$\begin{bmatrix} \boxed{4} & 25 & 44 & 10 \\ 8 & \boxed{14} & 8 & 33 \\ 17 & 2 & 16 & 34 \\ 5 & 13 & 24 & 7 \end{bmatrix} = \begin{bmatrix} \boxed{25} & 44 \\ 17 & 34 \end{bmatrix}$$

Agrupamento médio

Neste método, o resultado é definido com base na média dos valores encontrados dentro do tamanho do agrupamento. Um exemplo pode ser ilustrado utilizando um mapa de características de tamanho 4x4, com um agrupamento de tamanho 2x2. Esse processo calcula a média dos valores em cada janela de agrupamento de 2x2, gerando uma matriz de saída reduzida que preserva as informações médias das características originais.

$$\begin{bmatrix} \boxed{4} & 25 & 44 & 10 \\ 8 & \boxed{14} & 8 & 33 \\ 17 & 2 & 16 & 34 \\ 5 & 13 & 24 & 7 \end{bmatrix} = \begin{bmatrix} \boxed{12} & 23 \\ 9 & 20 \end{bmatrix}$$

Agrupamento global médio

Neste método, o resultado é obtido pela média geral do mapa de características, o que resulta, invariavelmente, em uma matriz de saída de dimensão 1x1. Por exemplo, considerando um mapa de características de tamanho 4x4, o agrupamento global médio calculará a média de todos os valores presentes neste mapa, condensando toda a informação em um único valor representativo. Essa abordagem é particularmente útil para reduzir significativamente as dimensões dos dados, mantendo uma representação global das características essenciais.

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

Camada totalmente conectada

A camada totalmente conectada é geralmente utilizada no final da arquitetura. Ela estabelece uma ligação direta de cada neurônio a cada etiqueta final, tornando essa camada extremamente pesada em termos computacionais. O número de neurônios nesta camada é equivalente ao número de classes propostas. Além disso, é nesta camada que a função de perda é calculada, iniciando o processo de retropropagação ([ALZUBAIDI et al., 2021](#); [TAYE, 2023](#)).

Aperfeiçoamento

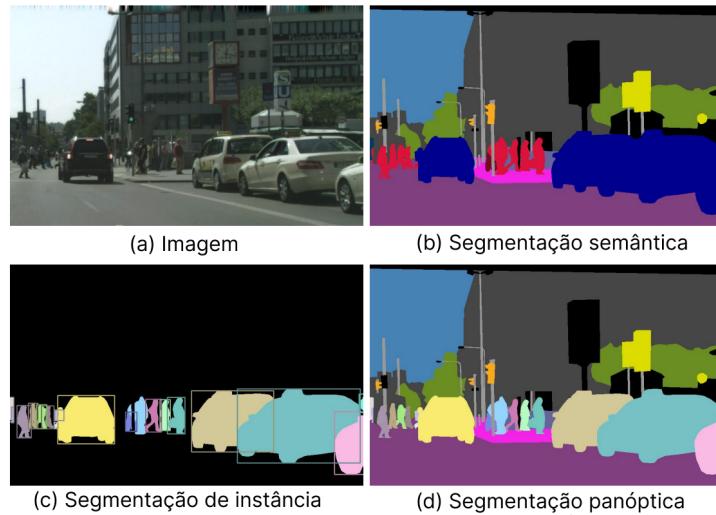
De acordo com [Alzubaidi et al. \(2021\)](#), [Taye \(2023\)](#), existem várias técnicas para aperfeiçoar os resultados do modelo, incluindo:

- Dropout: Esta técnica é amplamente utilizada para evitar o sobreajuste. Ela desativa aleatoriamente um neurônio durante o treinamento, forçando o modelo a aprender a identificar características em outros neurônios, o que possibilita a generalização do modelo.
- Aumentar o tamanho do conjunto de dados: Se não for possível obter um conjunto de dados maior, técnicas de aumento artificial podem ser empregadas, como rotacionar, recortar e inverter as imagens horizontal ou verticalmente.
- Normalização em lote: Esta técnica normaliza as saídas para acelerar o treinamento da rede.
- 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 tem três principais nichos, sendo eles: a 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 à saída de segmentação semântica, porém, separando objetos da mesma classe. Essa é a mais recente e completa. A diferença entre esses três tipos está ilustrada na Figura 20 ([ULKU; AKAGÜNDÜZ, 2022](#); [WANGENHEIM, 2021](#)).

Figura 20 – Tipos de segmentação em redes neurais convolucionais

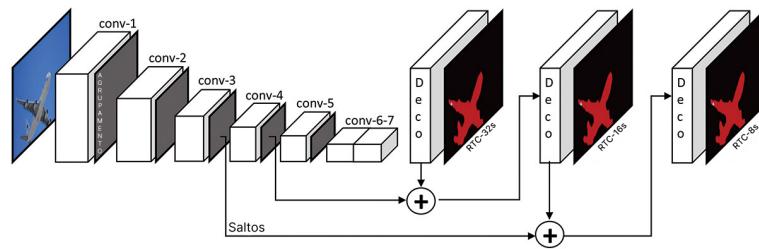


Fonte: Kirillov et al. (2019)

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

Figura 21 – Exemplo de arquitetura de rede totalmente convolucional

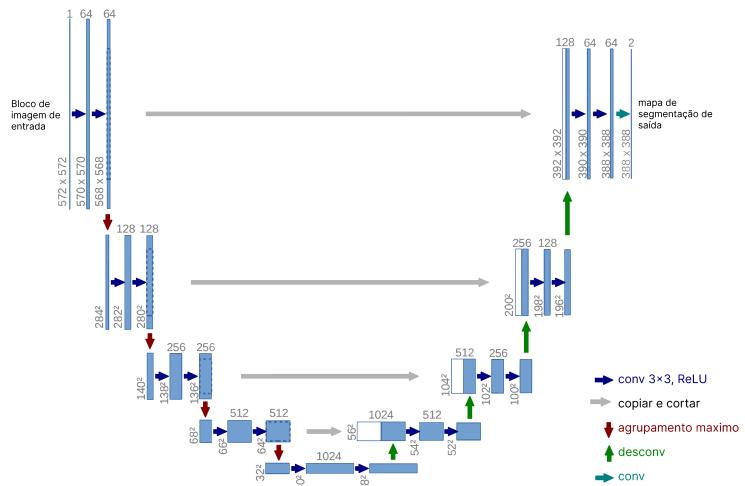


Fonte: Ulku e Akagündüz (2022)

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, na 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 a subida a de 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 máscaras nos objetos, como por exemplo a arquitetura Mask R-CNN (ULKU; AKAGÜNDÜZ, 2022; WANGENHEIM, 2021).

A arquitetura Mask R-CNN é derivada do Fast R-CNN — aprimoramento do R-CNN aplicando o conceito RoIPool para classificar —, onde há uma segmentação de máscara em cada ROI — ou Região de Interesse —, paralela à classificação da caixa delimitadora. A máscara é classificada com uma pequena RTC em cada ROI. Além disso, há 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 da 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 conjuntos 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 destacadas nesta monografia a seguir:

Classificação de conjuntos

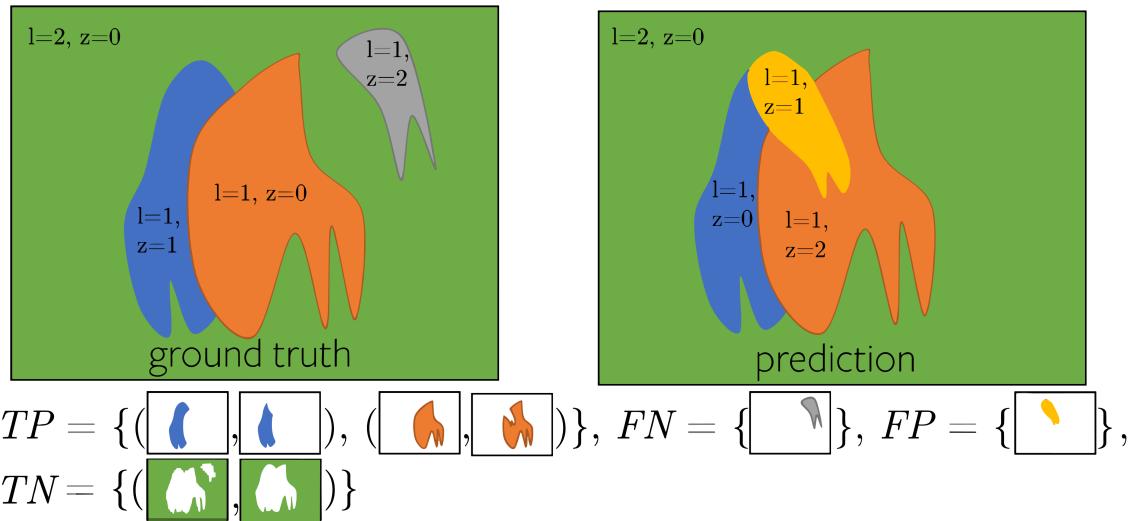
A classificação de conjuntos é uma técnica para criar relações entre a imagem de predição e a imagem do conjunto de dados. Ela se divide em quatro conjuntos, sendo eles: Positivo Verdadeiro — ou True Positive (TP) —, Negativo Verdadeiro — ou True Negative (TN) —, Falso Positivo — ou False Positive (FP) —, quando um objeto não é correspondido na imagem de predição, e, por fim, Falso Negativo — ou False Negative (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 pixels. Sua representação visual pode ser observada na Figura 23 ([KIRILLOV et al., 2019; WANG; WANG; ZHU, 2020](#)).

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 a maior eficiência. Segue a exemplificação da Equação (2.12) ([CHICCO; JURMAN, 2020](#)):

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

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



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

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 a 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 o erro obtido, dividido pelo erro mais o 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 o erro obtido, dividido pelo erro mais o 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 os conjuntos. Varia de 0 a 1, sendo 1 a 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) —, também conhecida como 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 a 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 definida 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 uma simplificação para uma fórmula que contém uma relação entre métricas de segmentação semântica e de instância.

De acordo com Ulku e Akagündüz (2022), 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 utilizada 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. Esta 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 modelos encontra-se na Tabela 1 (DATASET, 2023).

Tabela 1 – Top 15 modelos que melhor classificam pessoas com métrica PQ 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). Em seu repositório na internet Mohan e Valada (2020), há algumas recomendações de tecnologias utilizadas para desenvolver e testar o modelo. Embora possa funcionar em outros cenários, não há suporte dos criadores para tais aplicações. Essas tecnologias são: uma distribuição de sistema operacional¹ com núcleo

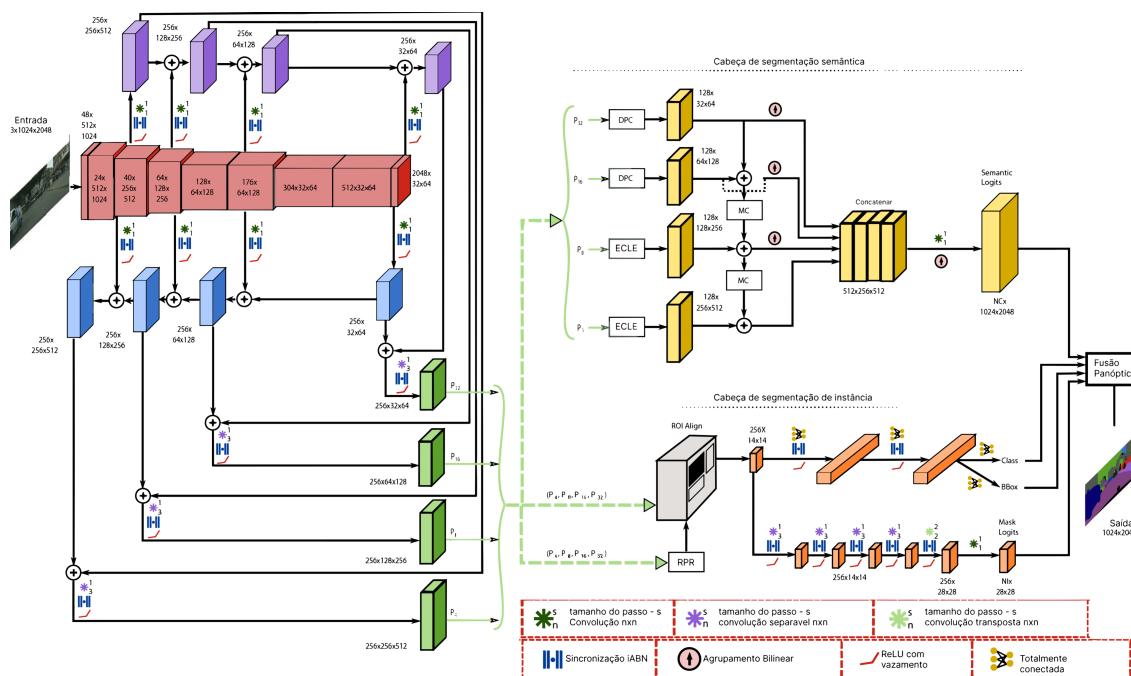
¹ Programa que gerencia a parte física de um computador; meio termo entre o usuário e a parte física do computador.

Linux, conforme descrito em [Hat \(2023\)](#); a linguagem de programação Python na versão 3.7, que, segundo [Magazine \(2023\)](#), é amplamente utilizada no campo da IA devido à sua simplicidade e às 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, visando facilitar a criação de modelos de aprendizado profundo; o CUDA Toolkit na versão 10.2, que, segundo [Developer \(2023\)](#), é uma biblioteca para proporcionar aceleração do processamento com placas de vídeo da marca Nvidia; e, por fim, o GCC nas versões 7 ou 8, que, conforme [Project \(2023\)](#), é um compilador das linguagens C e C++, frequentemente utilizados em bibliotecas Python, como o PyTorch, por exemplo.

Arquitetura

O artigo apresenta uma arquitetura que se inicia com um *backbone* — parte responsável por identificar características —, utilizando uma Rede de Pirâmide de Características (RPC)² de dois caminhos, seguida de dois cabeçotes paralelos: um para uma arquitetura de segmentação semântica, de autoria própria, e outro de instância, com modificações baseadas na topologia Mask R-CNN. Finalmente, a saída dos dois cabeçotes é combinada 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 \(2021\)](#)

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

Backbone da rede

A espinha dorsal — ou backbone — consiste em uma codificação combinada 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. No entanto, neste artigo, os autores apresentam uma solução equilibrada neste aspecto. O codificador contém nove blocos (em vermelho), mostrados na Figura 24, e as saídas 2º, 3º, 5º e 9º — da esquerda para a direita — correspondem aos fatores de redução de amostragem x4, x8, x16 e x32, respectivamente. Essas saídas se conectam à bifurcação paralela, que possui sentidos opostos, para gerar mais detecções de características. Posteriormente, é realizada uma combinação entre camadas de mesma dimensão, utilizando camadas de convolução separável em profundidade — que divide em etapas espaciais e de canal, aplicadas a cada canal e a 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, uma criação dos autores, é dividido em três módulos: o Extrator de Características em Larga Escala (ECLE) — ou Large Scale Feature Extractor (LSFE) —, destinado a capturar recursos finos em larga escala de forma eficiente; o módulo DPC, que deve ser capaz de capturar contexto de longo alcance, mas em pequena escala; e o módulo MC, responsável por 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 processadas separadamente. As entradas $P_{16} + P_{32}$ — de pequena escala — alimentam dois módulos DPC paralelos, enquanto $P_4 + P_8$ — de larga escala — alimentam dois módulos ECLE paralelos (MOHAN; VALADA, 2021).

Cabeçote de segmentação de instância

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

³ Normalização em lotes entre cores de GPU para aumentar o desempenho.

Módulo de fusão panóptica

O módulo de fusão panóptica é essencial para construir a imagem com segmentação panóptica. Nesta etapa, os resultados dos dois cabeçotes anteriormente explicados são combinados. Esta tarefa não é simples, pois requer uma lógica sofisticada para lidar com as sobreposições encontradas. O módulo foi projetado para ser adaptativo e usar as duas entradas de forma equivalente (MOHAN; VALADA, 2021).

Resumidamente, o módulo aplica técnicas para reduzir o número de instâncias baseando-se na métrica logística — um valor numérico que pontua a confiança. Ele realiza agregações entre os resultados dos dois cabeçotes e desenha com fundo preto as instâncias de maior confiança. Posteriormente, preenche com a parte de *stuff* — classes semânticas de menor importância — da entrada semântica (MOHAN; VALADA, 2021).

2.4 Trabalhos Relacionados

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

Segmentação Panóptica

No trabalho de Kirillov et al. (2019), é definida a ideia geral de segmentação panóptica, além de conceitos importantes como "coisas" e "objetos", e a métrica unificada para medir o desempenho de modelos dessa área. Foram realizados alguns testes comparando resultados humanos com um modelo simples, combinando PSPNet e Mask R-CNN, utilizando a métrica de qualidade panóptica definida por eles. Os resultados mostraram a superioridade humana na segmentação panóptica em três conjuntos de dados diferentes: 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 a 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: Segmentação Panóptica Eficiente

No trabalho de 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.

Geração de Mapas Poligonais para Jogos

No artigo de Patel (2010), é apresentada toda uma jornada de desenvolvimento de um algoritmo de geração procedural de conteúdo. São mostradas 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 aplicando isso no diagrama de Whittaker para definir o bioma do polígono. Também é apresentada uma técnica para adicionar ruídos nas arestas dos polígonos, tornando o mapa 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 estudo propõe o desenvolvimento de um protótipo voltado para a geração procedural de mapas que incorporam biomas, sendo o contorno determinado a partir de dois cenários distintos: uma representação gráfica de um desenho e uma fotografia do cotidiano.

Na imagem do desenho, a utilização de uma técnica de segmentação para a detecção de objetos será empregada, contribuindo para a seleção do contorno. Por sua vez, na fotografia do cotidiano, será adotado um modelo de segmentação panóptica. Este modelo, conforme delineado na Figura 25, processará a imagem, proporcionando ao usuário a possibilidade de escolher o contorno desejado. O resultado final replicará o contorno da seleção em ambas as dimensões, bidimensional e tridimensional.

A Figura 25 esboça os passos a serem seguidos para uma imagem urbana, onde a imagem de entrada (a) representa o ponto de partida do processo de segmentação panóptica e o resultado (imagem b) é obtido mediante o emprego do modelo EfficientPS, desenvolvido por Mohan et al. (2020). Este modelo, reconhecido por sua eficiência e elevada qualidade na classificação panóptica de pessoas, torna-se crucial em contextos urbanos contemporâneos. Após a segmentação, o usuário tem a capacidade de selecionar o contorno por meio de técnicas de escolha de cor e preenchimento de inundação, conforme detalhado por OpenCVInRange e OpenCVFloodFill, respectivamente.

O desfecho nos dois cenários — fotografia do cotidiano e desenho — é uma máscara binária, isolando o objeto na imagem, como evidenciado na imagem c, que destaca a escolha do segmento, tal como o carro amarelo na imagem b.

Posteriormente, um mapa procedural será gerado com o contorno selecionado, conforme exemplificado 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), seguindo a classificação de biomas.

Após essas etapas, a atualização automatizada do relevo do mapa tridimensional será realizada. Adicionalmente, será incorporada uma visualização conjunta do mapa em 3D e 2D no Unity, sendo o mapa 3D jogável e o 2D funcionando como minimapa, oferecendo

uma noção de localização no contexto tridimensional (Amit, 2010; First Person Movement). Testes com métricas de segmentação serão conduzidos para avaliar a similaridade entre a entrada da geração procedural e a saída, validando a hipótese de que uma maior densidade de pontos no diagrama de Voronoi resultará em maior precisão nos contornos das imagens.

Figura 25 – Etapas da proposta



Fonte: [Kirillov et al. \(2019\)](#) e autoria própria

3.2 Implementação

Pode-se dividir a implementação em seis partes: segmentação de imagens, seleção de contorno, geração procedural de mapas com biomas, criação de testes para avaliar a geração procedural, análise de casos de pós-processamento e interligação das ferramentas através de uma interface gráfica.

3.2.1 Segmentar imagens

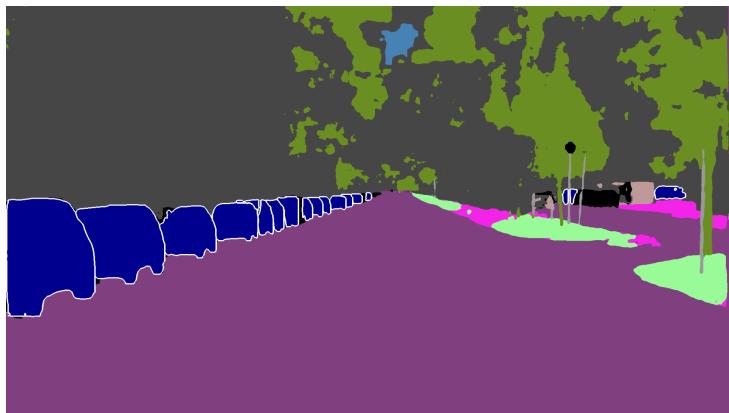
A segmentação de imagem é utilizada para classificar os pixels de uma imagem a partir de padrões reconhecidos por uma inteligência artificial. Isso permite ao usuário selecionar o contorno para gerar o mapa ([ULKU; AKAGÜNDÜZ, 2022](#); [WANGENHEIM, 2021](#)).

Para a fotografia urbana, utilizou-se o modelo EfficientPS que, segundo [Mohan e Valada \(2021\)](#), é uma solução eficiente para a segmentação panóptica. Utilizou-se também o código disponibilizado no repositório de [Mohan e Valada \(2020\)](#).

Empregou-se um arquivo pré-treinado deste 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, mas surgiram diversos desafios, incluindo a obtenção de autorização para baixar conjuntos de dados específicos, a preparação desses conjuntos para o processo de aprendizado e limitações de capacidade de processamento. Portanto, optou-se por utilizar o próprio modelo salvo, baixado do repositório.

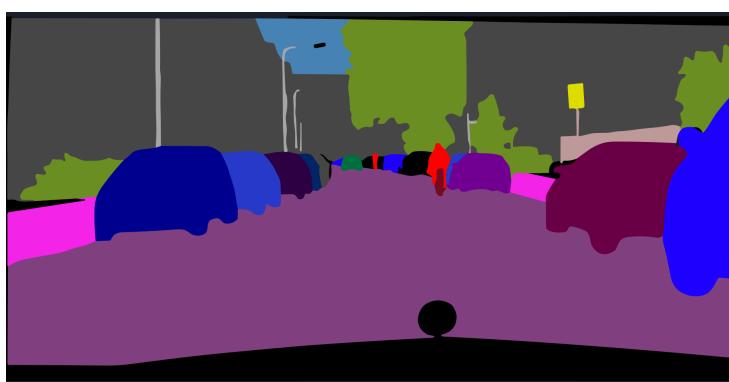
Percebeu-se que o resultado do modelo não foi o esperado, pois objetos da mesma classe, como carros, tinham a mesma cor com uma borda branca, conforme exemplificado na Figura 26. Assim, tentou-se alterar o código para gerar uma saída com objetos da mesma classe em cores diferentes, conforme ilustrado na Figura 27 no artigo de [Mohan e Valada \(2021\)](#), onde cada carro possui uma cor distinta e não há bordas entre os objetos.

Figura 26 – Resultado inicial do EfficientPS.



Fonte: Criação própria

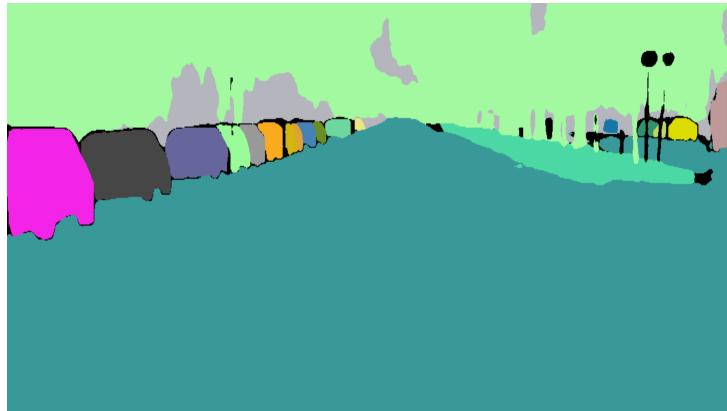
Figura 27 – Expectativa do resultado do EfficientPS.



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

Seguindo os passos citados em uma publicação de [Mohan e Gabriele \(2021\)](#) no repositório de [Mohan e Valada \(2020\)](#), o resultado obtido não foi satisfatório, pois não foi possível discernir a que classes os segmentos pertenciam. 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 conhecida como imagem binária, que, segundo [Embarcados \(2017\)](#), consiste em isolar um objeto da imagem. Esta técnica pode ser usada como uma máscara para auxiliar no processamento do mapa no contorno desejado ([AZNAG et al., 2020](#)).

Foram empregadas duas abordagens para selecionar o contorno: seleção por cor e seleção por preenchimento de inundação. A seleção por cor, conforme descrito por [OpenCV \(2023c\)](#), isola tudo na imagem que corresponde a uma determinada cor. Já a seleção por preenchimento de inundação, segundo [OpenCV \(2023b\)](#), baseia-se em um algoritmo de expansão que compara com uma faixa delimitada de cores. Ambos resultam em uma imagem binária, que também pode ser utilizada para selecionar a foto com desenho diretamente.

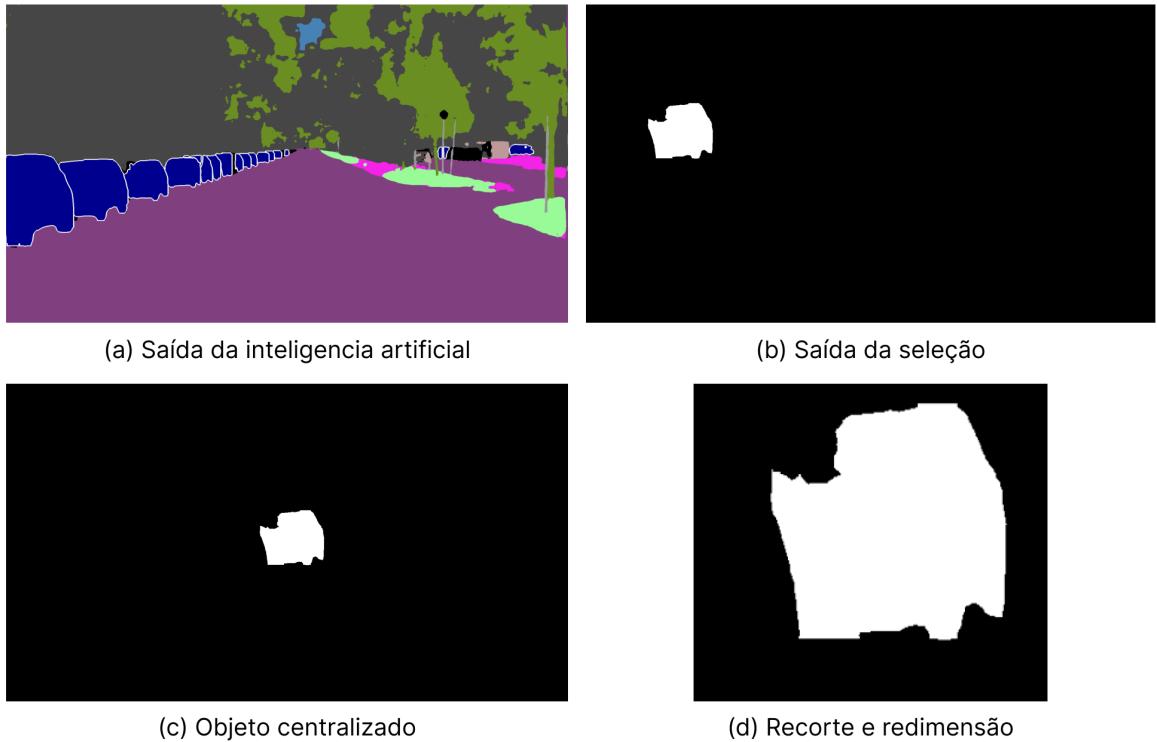
Tratamento da imagem binária

Existem duas formas de tratar a imagem binária.

A primeira, não utilizada devido à distorção que causa, ocorre após a saída dos algoritmos de seleção. O objeto selecionado é detectado, centralizado em uma nova imagem, recortado e redimensionado a partir do centro para formar uma imagem quadrada. Todos os passos são observáveis na Figura 29 ([EMBARCADOS, 2017](#)).

Já a segunda, mais eficiente, melhora os resultados evitando a distorção e a presença de bordas inadequadas. Este método localiza o objeto na imagem binária e recorta a imagem com as coordenadas e dimensões fornecidas. Em seguida, acrescenta-se uma borda na altura ou largura, conforme a folga entre as duas, transformando a imagem em quadrada. Após o redimensionamento para manter um padrão, adiciona-se novamente uma borda

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



Fonte: Autoria própria

para representar o mar. Na Figura 30, observa-se que o objeto não foi distorcido como no método anterior e permaneceu centralizado na imagem final ([EMBARCADOS, 2017](#)).

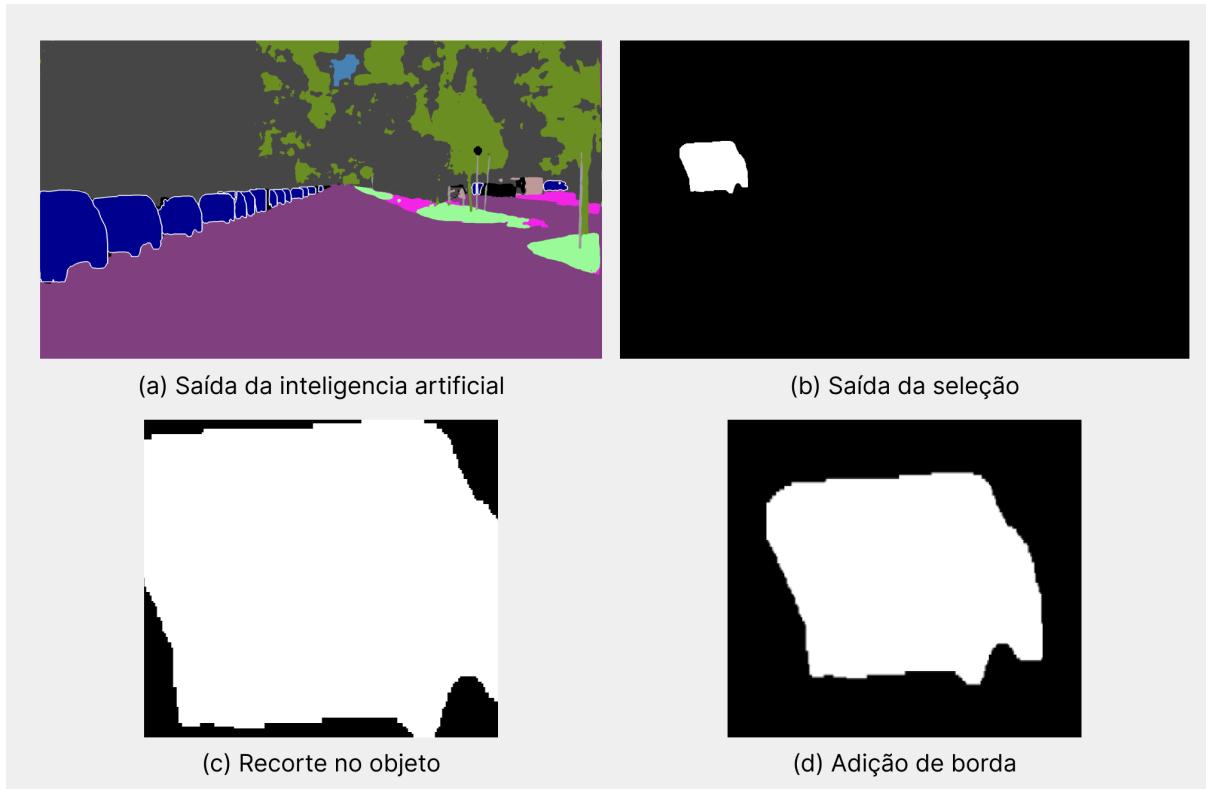
3.2.3 Geração procedural do mapa

A base para a geração procedural do mapa foi o artigo de [Patel \(2010\)](#), com uma implementação em Python realizada por [Markiewicz et al. \(2021\)](#).

3.2.3.1 Ilha gerada no contorno

Para gerar o mapa da ilha, é necessário criar um diagrama de Voronoi. Primeiro, define-se pontos de quantidade pré-estabelecida e localização pseudoaleatória, que serão os centroides dos polígonos. Os vértices dos polígonos são gerados a partir da intersecção entre retas perpendiculares aos pontos médios entre nós vizinhos. Esses nós vizinhos são encontrados por meio de uma circunferência no ponto, e, se existir algum ponto, ele é marcado como vizinho, definindo essa área como região, conforme descrito na Equação (2.1). Em seguida, cria-se outro grafo com esses pontos. A definição dos vértices é ilustrada na Figura 31, onde os pontos vermelhos são os centroides, ligados por linhas pretas para gerar pontos médios (amarelos), traçando uma reta perpendicular. A intersecção é representada

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



Fonte: Autoria própria

pelo ponto azul, que se torna o vértice, e a cor rosa representa a aresta do polígono ([PATEL, 2010; RODRIGUES, 2019](#)).

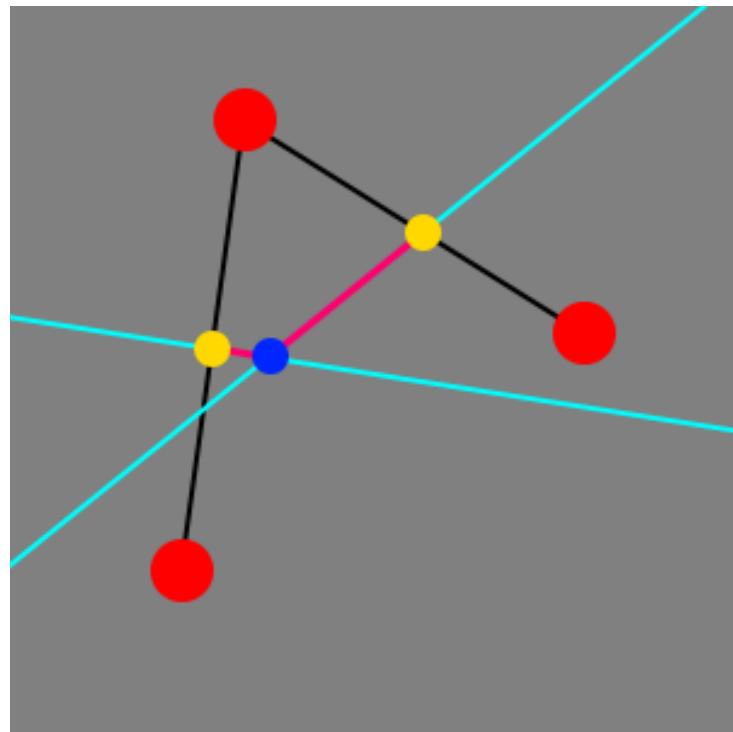
A Figura 32 mostra o modelo do diagrama de Voronoi do algoritmo, onde os pontos vermelhos são os centroides, ligados por linhas pretas. Os pontos azuis são os vértices, conectados por linhas brancas para formar as arestas do polígono.

Cada polígono criado é denominado região, e basta definir o tipo do terreno, elevação e umidade. Na lógica, todos os polígonos com o terreno do tipo oceano são marcados. Há duas formas de selecionar o contorno da imagem.

A primeira, mais lenta, pois necessita verificar todos os pixels da imagem, percorre todos os pontos da imagem de entrada e verifica se cada pixel contido no objeto da imagem binária encontra-se nos polígonos gerados. Se encontrado, o polígono é adicionado a uma lista de marcação ([OPENCV, 2023a](#)).

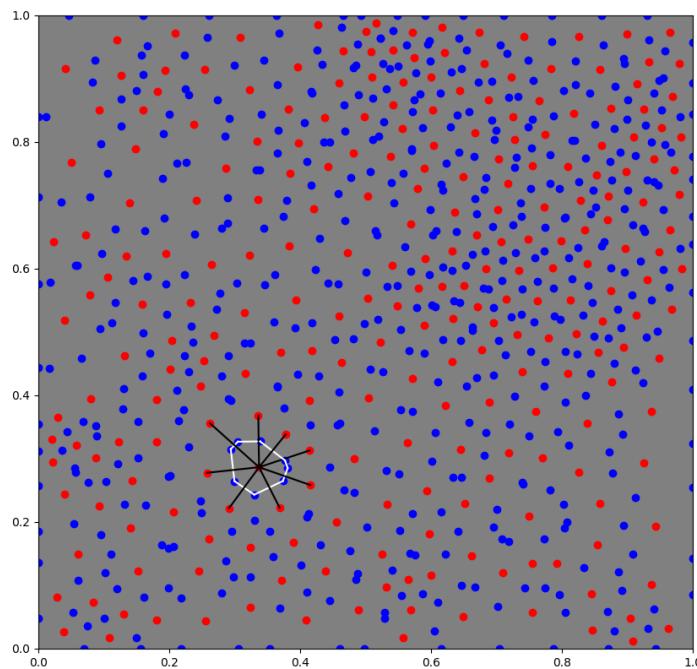
A segunda, mais performática, utiliza as bibliotecas do OpenCV. Faz-se uma iteração na lista de polígonos e, para cada um, cria-se uma imagem na qual o polígono da iteração é desenhado. Realiza-se uma comparação com a imagem binária, e o resultado dessa comparação é uma imagem com os pixels presentes em ambas as imagens (binária e do polígono). Verifica-se a existência de pixels brancos na imagem resultante e, se houver,

Figura 31 – Ilustração do processo de criação do polígono.



Fonte: Autoria própria

Figura 32 – Ilustração do diagrama de Voronoi.



Fonte: Autoria própria

adiciona-se o polígono à lista de marcação ([OPENCV, 2023a](#)).

Para os polígonos na lista de marcação, marca-se o tipo do terreno como terra. Em

seguida, verifica-se quais polígonos do tipo terra são vizinhos do tipo oceano. Se forem, define-se como do tipo litoral, e também se marcam os pontos desses polígonos que se encontram no polígono do tipo oceano (PATEL, 2010; EMBARCADOS, 2017).

Para calcular a elevação dos polígonos, inicialmente calcula-se a elevação das arestas de todos os polígonos. Isso é feito a partir de uma busca em profundidade que começa em todas as arestas tocando a borda do gráfico. A cada nova iteração, as arestas que tocam os cantos recebem o valor 0, e a altura evolui à medida que se aproxima do centro do mapa. Depois, realiza-se o cálculo de redistribuição de elevação. Com base na elevação calculada anteriormente, forma-se uma lista ordenada pela elevação, da menor para a maior. Cada item da lista é enumerado com um índice i e, com essa enumeração, calcula-se usando a Equação (3.1). Em seguida, calcula-se a Equação (3.2), sendo X a elevação do vértice. Finalmente, para calcular a elevação do polígono, faz-se a média dos vértices que pertencem ao polígono (PATEL, 2010).

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

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

Para a 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 o tipo de terreno dos vizinhos dos vértices e das arestas que foram tocadas 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 (PATEL, 2010).

Na geração de umidade, atribui-se um valor para cada vértice que possui uma aresta do tipo rio, ou se é vizinho de um lago, e coloca-se esse item em uma fila. Logo em seguida, busca-se, em profundidade, em todos os vértices adjacentes dos itens da fila, e calcula-se uma nova umidade a partir de uma multiplicação de um fator com o valor da umidade anteriormente atribuída. Verifica-se se os vértices adjacentes possuem uma umidade maior que a nova umidade calculada; se possuírem, coloca-se o vértice adjacente na fila e atribui-se o valor da nova umidade. Para terrenos do tipo oceano, é atribuído o valor máximo de umidade (PATEL, 2010).

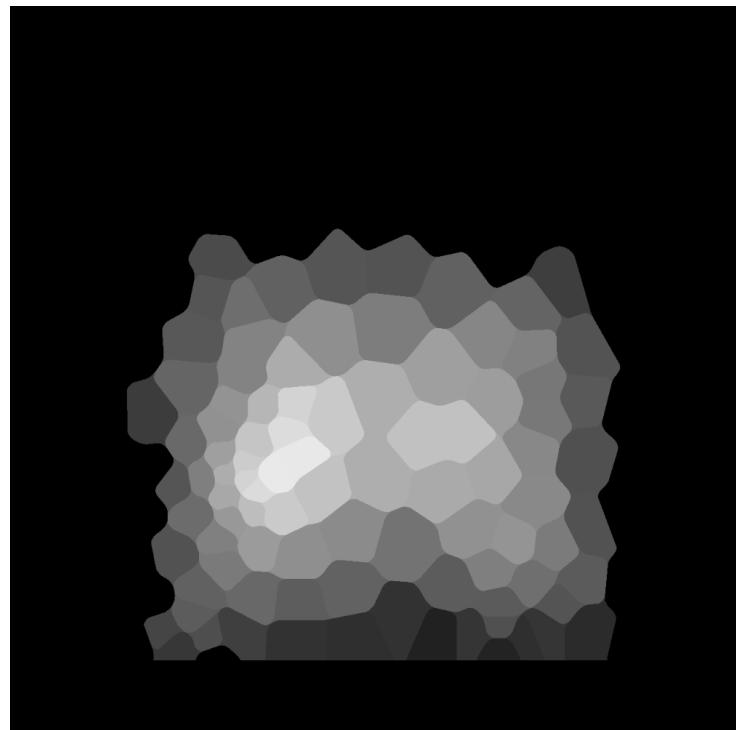
Por fim, atribui-se o valor da umidade para os polígonos e redistribui-se a umidade a partir de uma lista de polígonos com umidades ordenadas. Para cada item i , executa-se a Equação (3.3) e, assim, calcula-se o valor final da umidade (PATEL, 2010).

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

Para assinalar os biomas, verifica-se o tipo do terreno, altura e a umidade; cada um desses parâmetros liga-se a um determinado bioma. E, para a atribuição final, aplica-se a classificação descrita na Figura 5. Porém, editou-se o eixo de temperatura para elevação, para chegar nos resultados da Tabela 2, pois o diagrama de Whittaker não leva em consideração a elevação (PATEL, 2010).

Para gerar o mapa tridimensional, é gerada uma imagem denominada "mapa de altura", construída com base nos dados de elevações do grafo. Gera-se uma imagem com tonalidades de cinza, sendo que o pixel branco, com valor 255, representa o ponto mais alto do mapa, e o pixel preto, com valor 0, representa o ponto mais baixo. Isso é possível observar na Figura 33 (ZOO, s. d.).

Figura 33 – Exemplo de mapa de altura.



Fonte: Autoria própria

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: o terreno, o minimapa em 3D e a jogabilidade (TECHNOLOGIES, s. d.).

Tabela 2 – Relação entre umidade e elevação para biomas

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

3.2.3.2.1 Terreno

Para atualizar o terreno, utilizou-se primeiramente o pacote *Terrain Tools*, que facilita o processo, 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 à altura mais alta ([TECHNOLOGIES, s. d.](#)).

Porém, esse processo era manual, e a ideia era que fosse automatizado. Logo, utilizou-se um algoritmo ligado à câmera do personagem que, ao iniciar a cena, atualiza o terreno com a nova imagem de mapa 3D em RAW. O algoritmo abre a imagem e percorre-a, anotando o relevo proporcionalmente a uma matriz, e depois aplica no terreno ([TECHNOLOGIES, s. d.](#)).

Além da funcionalidade de relevo, adicionou-se um pacote para alterar a textura de acordo com a altura, a fim de diferenciar o oceano da ilha e o bioma de floresta. Utilizou-se o pacote denominado *Terrain Toolkit 2017*. No script, carregam-se as texturas e definem-se alguns parâmetros para atualizar com o terreno ([TECHNOLOGIES, 2017](#)).

3.2.3.2.2 Minimap

A saída da geração procedural visa dois principais resultados: o mapa de altura para definir o terreno e o mapa 2D que pode ser usado como minimapa ([TECHNOLOGIES, s. d.](#)).

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), atualizando as propriedades do objeto de jogo. Isso possibilita 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 ([TECHNOLOGIES, s. d.](#)).

3.2.3.2.3 Jogabilidade

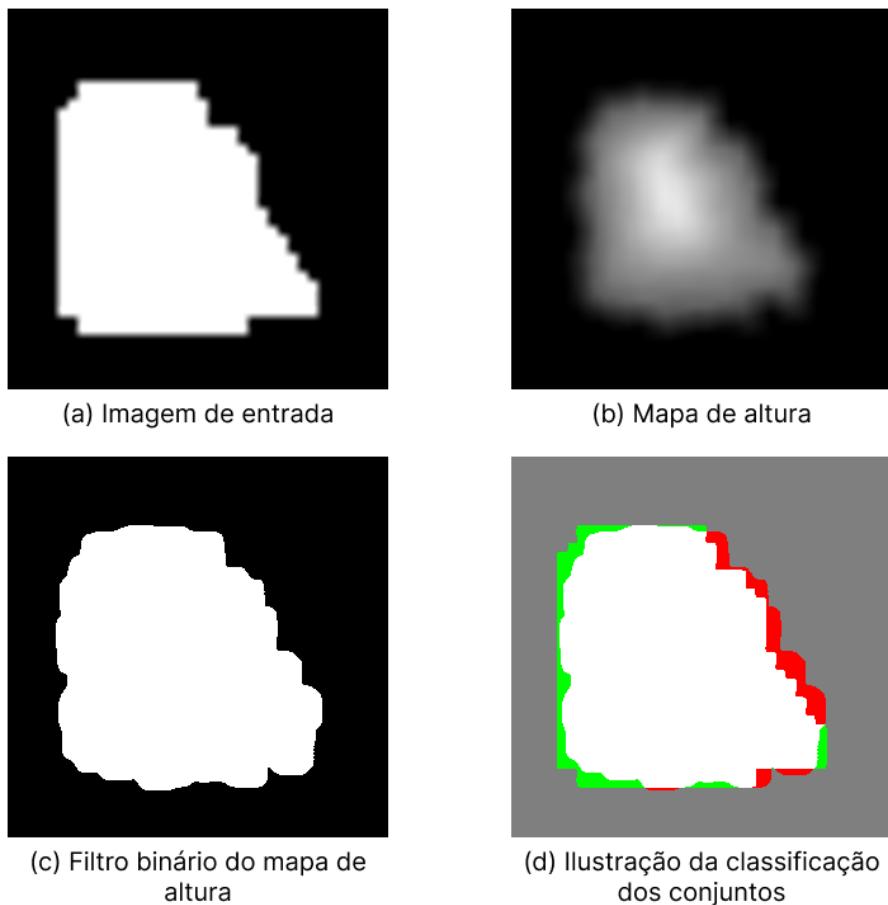
Criou-se um personagem com movimentação para andar pelo mapa, baseado no vídeo do YouTube de [BloxyDev \(2022\)](#). Adicionando os objetos e o script principal, além de criar uma tag de chão para detectar a colisão, é possível jogar no mapa, andando em todas as direções e pulando.

3.2.4 Testes

Criaram-se testes para comparar uma combinação de imagens, que podem conter a imagem de entrada do algoritmo de geração procedural ou alguma de suas saídas, como mapa de altura ou mapa 2D. Para mensurar a qualidade, utilizou-se como base a técnica descrita por [Kirillov et al. \(2019\)](#), chamada de classificação dos conjuntos.

Esta técnica cria uma nova imagem na qual cada pixel pertence a um conjunto dentre quatro: Verdadeiros Positivos, Verdadeiros Negativos, Falsos Positivos e Falsos Negativos. A Figura 34 apresenta a ilustração da classificação dos conjuntos. A imagem a é a imagem de entrada para geração procedural (saída da seleção); a imagem b, o mapa de altura, é uma das saídas da geração procedural; a imagem c representa o filtro binário a partir do mapa de altura, ou seja, apenas os pixels com cor monocromática zero são mantidos como preto, e qualquer alteração é marcada como branco; a imagem d ilustra os conjuntos entre a imagem de entrada e o filtro binário do mapa de altura, sendo a cor cinza a união entre cores pretas, a cor branca a união de cores brancas, a cor verde quando a imagem de entrada tem cor branca mas o filtro binário tem cor preta e, por fim, a cor vermelha indica que o filtro tem a cor branca, mas na imagem de entrada tem a cor preta.

Figura 34 – Ilustração da classificação de conjuntos



Fonte: Autoria própria

Posteriormente, contabilizaram-se esses pixels com cores diferentes para utilizá-los em diversas métricas baseadas na classificação dos conjuntos, tais como: F1 Score, Coeficiente de Correlação de Matthews, Taxa de Descoberta Falsa, Taxa de Falso Negativo, Acurácia e União sobre Interseção (CHICCO; JURMAN, 2020; NEDEA, 2020; BORA,

2022).

Além dessas métricas, empregou-se também uma métrica para o desfoque encontrado em uma imagem. Aplica-se um filtro para detectar bordas em x e y, calcula-se o gradiente e resulta-se no desvio padrão, o que serve para mensurar a harmonia do mapa de alturas. Outra métrica utilizada nos testes foi a duração do tempo no código, medida em segundos e contabilizando apenas a parte de geração procedural dos mapas.

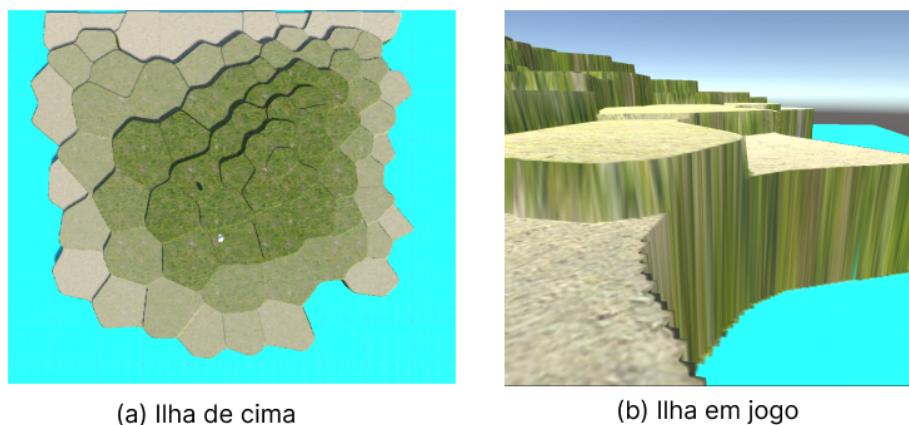
Criou-se um teste genérico para ser reutilizado em diversos testes e facilitar a reprodução em caso de mudanças. Na chamada do teste, define-se as imagens e as métricas, possibilitando definir todas as combinações possíveis. Para cada combinação, serão executadas cinco imagens de entrada pré-definidas. Em cada imagem, percorrer-se-á um laço baseado em alguma variável de geração procedural, como tamanho do filtro de desfoque, quantidade de pontos ou tamanho da borda no mapa 2D. Em cada iteração da variável, o processo será repetido três vezes.

Em cada iteração de todos esses laços aninhados, será gerado proceduralmente algum mapa ou ambos, processadas as métricas e armazenados os resultados. Após isso, calcular-se-á a média das métricas e criada uma tabela em L^AT_EX como resultado.

3.2.5 Pós-processamento

Analizando o resultado gerado no Unity do mapa 3D na Figura 35, percebe-se que não existe harmonia entre os biomas. Portanto, gerou-se uma solução aplicando um desfoque na imagem do mapa de altura para suavizar a mudança de biomas.

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

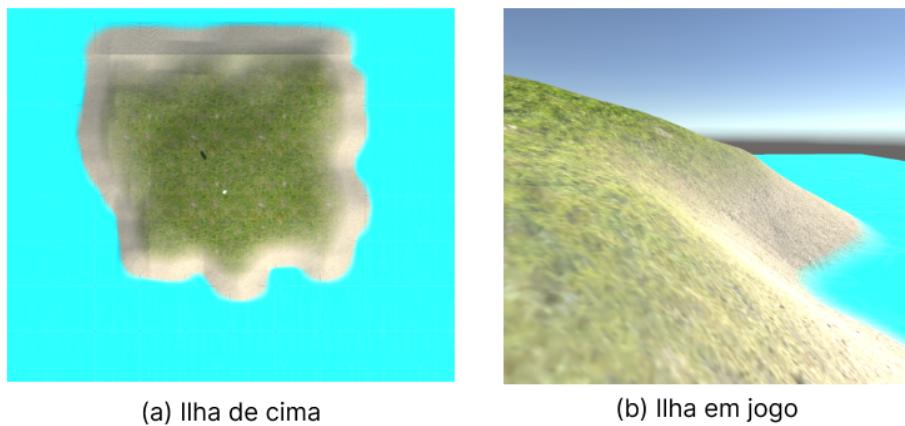


Fonte: Autoria própria

Aplicando um filtro — ou kernel — de desfoque de tamanho 100x100, obteve-se o resultado ilustrado na Figura 36. No entanto, ao analisar a Tabela 3, observa-se que as

métricas pioraram. Isso indica que o desfoque comprometeu a qualidade da equivalência dos contornos, exacerbando um problema de escala. Este problema é evidenciado pelo fato de que apenas o erro no mapa de altura foi detectado. Na Figura 37, observa-se uma representação visual da classificação dos conjuntos definida em Kirillov et al. (2019), sendo a imagem a o resultado da comparação entre a imagem de entrada e a saída do mapa de altura sem desfoque, com o vermelho indicando o erro encontrado no mapa de altura (está branco no mapa de altura, mas preto na entrada), e a imagem b, o mesmo caso, porém aplicando o filtro de desfoque.

Figura 36 – Resultado no Unity usando filtro de desfoque no mapa de altura.



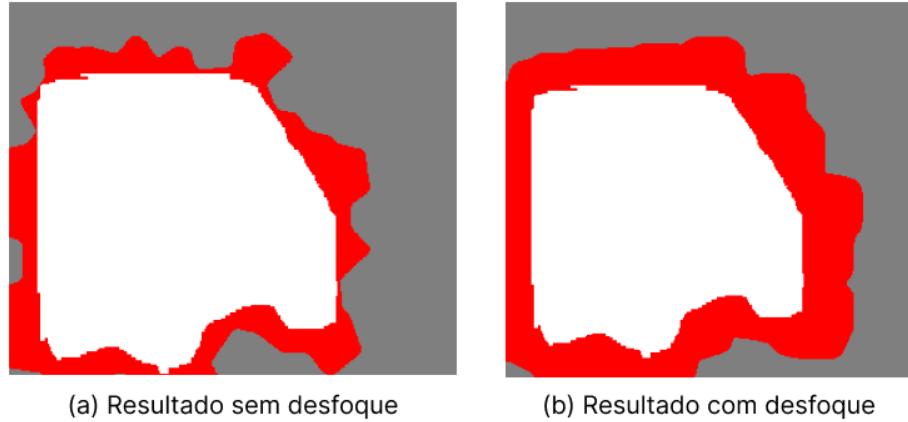
Fonte: Autoria própria

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

Para tratar esse problema, adotou-se uma solução baseada em redimensionar a imagem e adicionar uma borda, diminuindo assim o tamanho do mapa de altura. Redimensionou-se uma imagem de 1000x1000 para 800x800, reduzindo-a em 20%, e depois adicionou-se uma borda de cor preta em todos os lados de tamanho 100, retornando ao tamanho original. Para encontrar o melhor tamanho do filtro de desfoque nessas condições, executaram-se testes alterando o tamanho do filtro após a aplicação da técnica de redução do mapa. Os resultados estão na Tabela 4. Para selecionar o melhor resultado, definiu-se o critério de ter a menor métrica de desfoque — significando mais desfoque na imagem — e uma variação máxima de 1% na métrica IoU em comparação com a métrica sem o filtro de desfoque. Assim, selecionou-se o resultado com filtro de tamanho 80, pois a métrica IoU apresentou 79%, semelhante ao tamanho de filtro 0. Percebeu-se também que a diminuição

Figura 37 – Ilustração da classificação de conjuntos entre a imagem de entrada e o mapa de altura



Fonte: Autoria própria

da borda resultou em uma maior correspondência entre os erros da imagem de entrada e do mapa de altura.

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

Tamanho filtro	Desfoque	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

Após a melhoria do resultado do mapa de entrada, é necessário ajustar o mapa 2D, visto que este deve ser bastante semelhante ao mapa de altura, para informar, por exemplo, a localização exata do personagem no 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, sendo que o maior valor de IoU obtido foi de 60

3.2.6 Interface Gráfica

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

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 por 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, permite-se a seleção do contorno, cria-se uma imagem binária a partir do contorno e envia-se como argumento na geração procedural de mapas.

Além disso, para promover a usabilidade, utilizou-se um *loading* específico para o PyQt5. Para isso, foi necessário usar threads, criando classes para executar as tarefas de forma separada e síncrona.

4 Resultados

Neste capítulo, são apresentados os resultados obtidos da implementação e discutida a hipótese, o impacto e as possíveis causas.

4.1 Apresentação

Para apresentar os resultados, serão abordadas tabelas contendo testes com todas as combinações de imagens, uma imagem com os resultados finais das tabelas, além de uma imagem com todos os passos da aplicação.

A Tabela 6 apresenta os resultados comparativos entre a imagem de entrada e o mapa 2D. Esta comparação é realizada utilizando a técnica de classificação de conjuntos, conforme definido por ([KIRILLOV et al., 2019](#)). A análise foca na avaliação das métricas, conforme o número de pontos no diagrama de Voronoi, incluindo União sobre Interseção (IoU), Acurácia (Acc), F1 Score (F1), Coeficiente de Correlação de Matthews (MCC), Taxa de Descoberta Falsa (FDR), Taxa de Falso Negativo (FNR) e o tempo de execução do código ([CHICCO; JURMAN, 2020](#); [NEDEA, 2020](#); [BORA, 2022](#)).

Esta tabela é crucial, pois ela evidencia, com base em dados concretos, a semelhança do mapa 2D com o contorno do mapa de entrada. Além disso, corrobora a hipótese de que um maior número de pontos no mapa leva a melhores resultados nas métricas de IoU, Acc, F1 e MCC (onde valores mais altos indicam melhor desempenho), e simultaneamente, a uma redução nos índices de FDR e FNR (onde valores menores são preferíveis, visto que representam menor erro). Observa-se, ainda, que um aumento no número de pontos implica em um maior tempo de processamento na geração procedural do mapa.

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

Os resultados obtidos na Tabela 7 derivam da comparação entre a imagem de entrada e o mapa de altura, utilizando a técnica de classificação de conjuntos conforme definida por ([KIRILLOV et al., 2019](#)). Essa comparação foi realizada para mensurar diversos aspectos, incluindo as métricas associadas ao número de pontos no diagrama de

Voronoi. As métricas abrangem União sobre Interseção (IoU), Acurácia (Acc), F1 Score (F1), Coeficiente de Correlação de Matthews (MCC), Taxa de Descoberta Falsa (FDR), Taxa de Falso Negativo (FNR) e a duração da execução do código ([CHICCO; JURMAN, 2020](#); [NEDEA, 2020](#); [BORA, 2022](#)).

A relevância dessa tabela reside na capacidade de demonstrar, por meio de dados concretos, a proximidade entre o mapa de altura e o contorno do mapa de entrada. Além disso, ela valida a hipótese de que o aumento no número de pontos resulta em melhores desempenhos nas métricas de IoU, Acc, F1 e MCC, indicando uma maior semelhança entre os mapas. Simultaneamente, observa-se uma diminuição nas métricas de FDR e FNR, evidenciando uma redução nos erros. Vale notar que a quantidade de pontos também impacta diretamente na duração da geração procedural do mapa, sendo um fator relevante a ser considerado.

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

A tabela referenciada como Tabela 8 apresenta os resultados comparativos entre o mapa de altura e sua representação em mapa 2D. Esta comparação é realizada através do método de classificação de conjuntos proposto por ([KIRILLOV et al., 2019](#)), avaliando os resultados com base em várias métricas conforme a quantidade de pontos no diagrama de Voronoi. As métricas avaliadas incluem União sobre Interseção (IoU), Acurácia (Acc), F1 Score (F1), Coeficiente de Correlação de Matthews (MCC), Taxa de Descoberta Falsa (FDR), Taxa de Falso Negativo (FNR) e o tempo de execução do código ([CHICCO; JURMAN, 2020](#); [NEDEA, 2020](#); [BORA, 2022](#)). Esta tabela é crucial para demonstrar que o mapa de altura, que se transformará no mapa 3D, assemelha-se ao mapa 2D. Essa similaridade é fundamental no protótipo do Unity para a localização do personagem, combinando o mapa 2D como minimapa e o mapa 3D com dimensão e profundidade. Observa-se também que um aumento no número de pontos resulta em maior tempo de geração procedural do mapa.

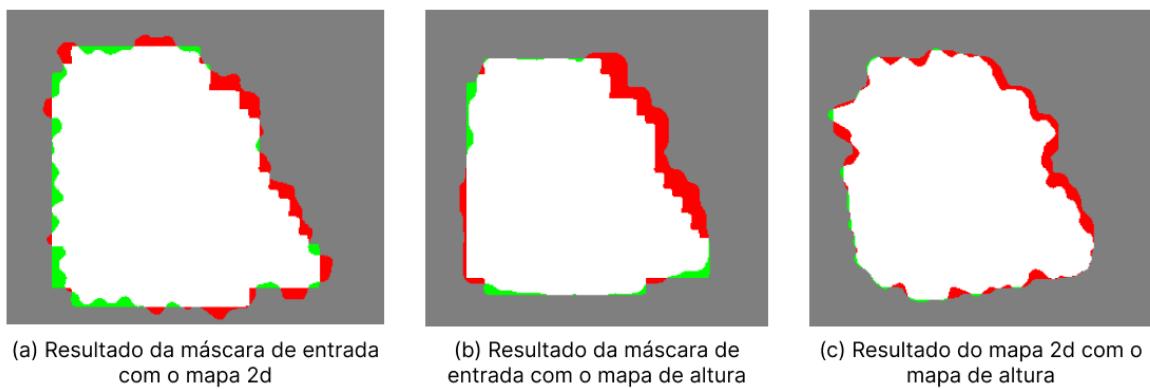
A Figura 38 contém a ilustração da classificação dos conjuntos de cada combinação. Assim, a imagem (a) ilustra a última execução da combinação entre a imagem de entrada e o mapa 2D, conforme os resultados da Tabela 6. A imagem (b) exibe a última execução da combinação entre a imagem de entrada e o mapa de altura, conforme os resultados da Tabela 7. Já a imagem (c) apresenta a última execução da combinação entre o mapa de

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

altura e o mapa 2D, conforme os resultados da Tabela 8.

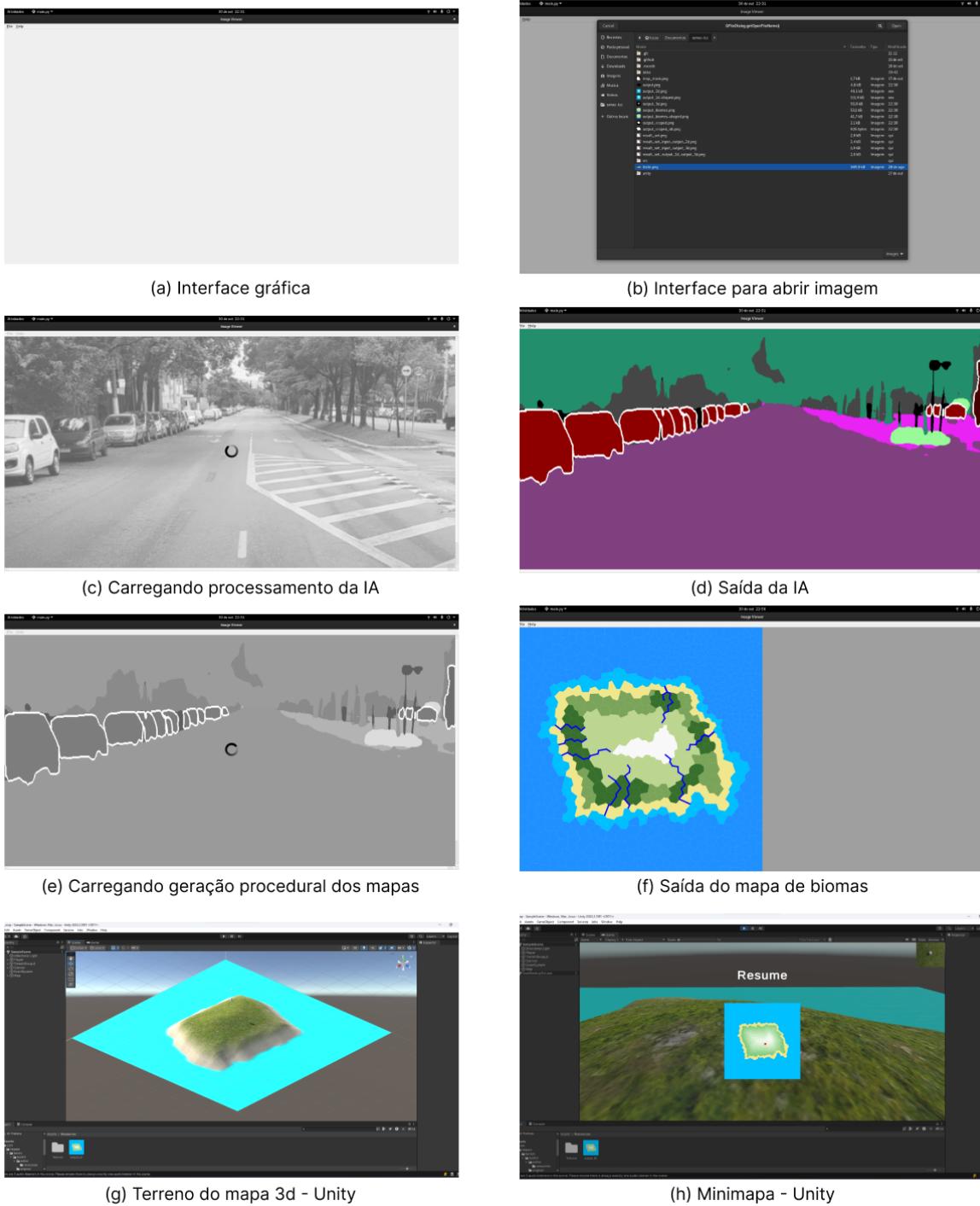
Figura 38 – Resultados da última execução de cada combinação de imagens.



Fonte: Autoria própria

A Figura 39 apresenta todos os passos da execução do programa gerado em Python. A imagem (a) mostra uma captura de tela da interface gráfica com PyQt5. A imagem (b) exibe uma captura de tela do processo de abertura de uma imagem para segmentação. Na imagem (c), é apresentada uma captura de tela da execução do modelo EfficientPS na imagem selecionada. A imagem (d) mostra uma captura de tela da saída da segmentação panóptica pelo modelo EfficientPS. Na imagem (e), temos o carregamento pós-seleção do usuário, que, no caso, selecionou um carro com o método de preenchimento por inundação. A imagem (f) exibe o resultado do mapa 2D com o contorno selecionado anteriormente. Na imagem (g), é apresentada uma captura de tela da automação usando Unity para atualizar um terreno com o mapa de altura resultante da geração procedural. Por fim, a imagem (h) mostra uma captura de tela do Unity rodando a aplicação e abrindo o minimapa (usando o mapa 2D) para oferecer uma noção de localização no mapa 3D, usando um ponto para marcar a localização atual do personagem no minimapa (mapa 2d).

Figura 39 – Passos do programa final.



Fonte: Autoria própria

Na figura Figura 40, são delineados os mesmos estágios exibidos na figura Figura 39, acrescidos de um componente relacionado ao método de reenchimento por inundação. A imagem 'a' representa a interface inicial, 'b' denota a seleção de uma imagem pelo usuário via o seletor de arquivos, 'c' ilustra a imagem escolhida pelo usuário, carregada na memória

e apresentada na interface, 'd' exibe uma interface de carregamento para o processamento da geração do mapa, 'e' revela o resultado do processamento, apresentado na interface, 'f' apresenta o mapa 3D no Unity visto de cima, com o formato da ilha, e, finalmente, 'g' mostra o mapa 2D gerado, apresentado como um minimapa no mapa 3D no Unity.

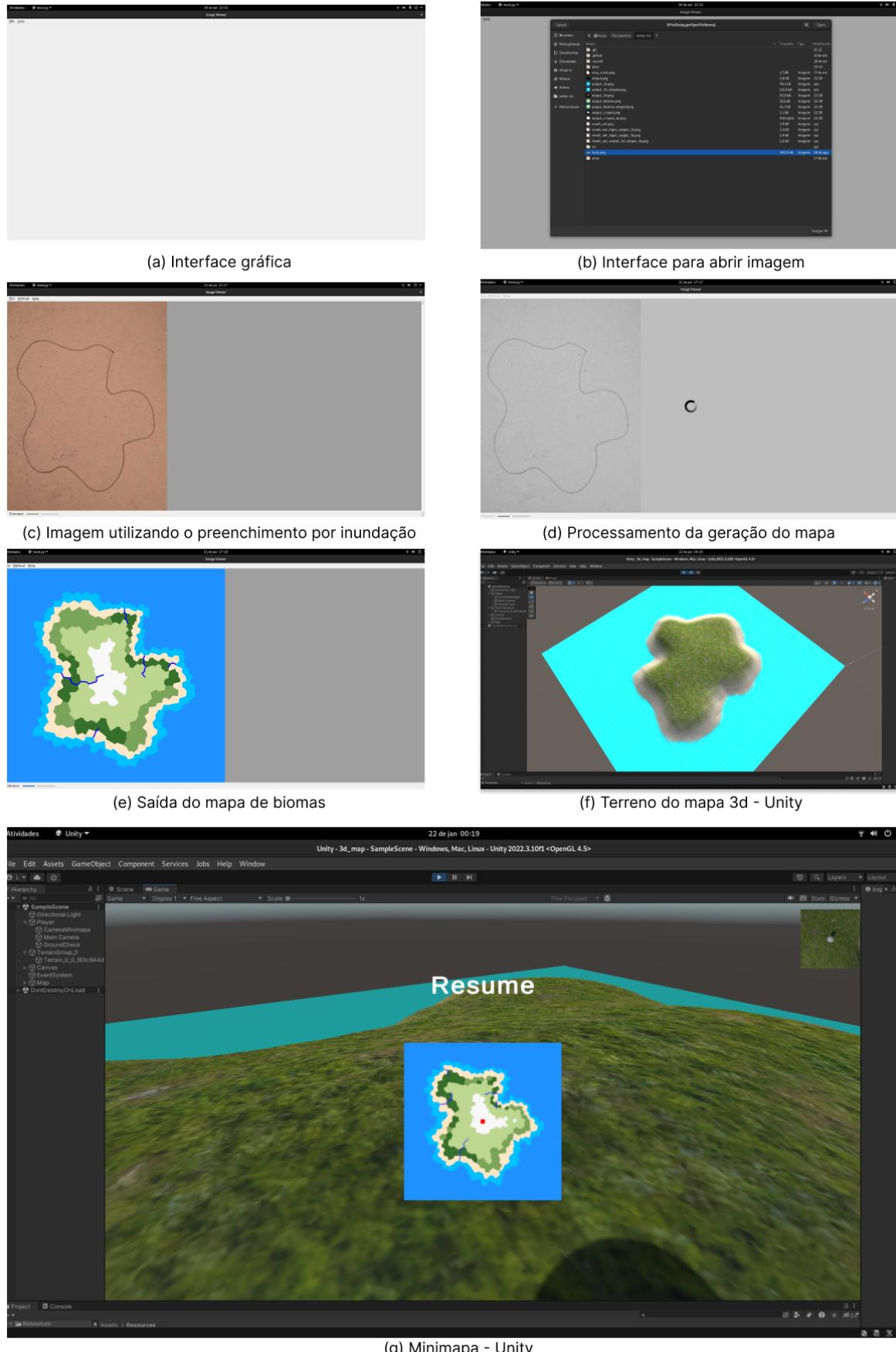
4.2 Análise dos Resultados

Com base nos dados e observações da Tabela 6 e da Tabela 7, é possível afirmar que a hipótese inicial se comprovou. Isso ocorre porque, quanto mais pontos o diagrama de Voronoi possuir, maior será a compatibilidade com o contorno. Esse fenômeno se dá devido ao aumento no número de polígonos no diagrama; com mais polígonos em um mesmo espaço, a tendência é que o tamanho desses polígonos diminua. Consequentemente, no cálculo para definir os tipos de terreno, há uma tendência à maior precisão em relação ao contorno inicial. Essa afirmação é corroborada pela tendência observada nas métricas das duas tabelas mencionadas, onde as métricas IoU, Acc, F1 e MCC aumentam a cada linha, e as métricas FDR e FNR apresentam uma diminuição no erro encontrado. Outra métrica relevante é a duração da geração procedural, que aumenta proporcionalmente ao número de pontos, indicando uma relação direta com o tempo de processamento. Portanto, é necessário analisar o cenário para o processamento e determinar qual é o melhor custo-benefício.

O resultado satisfatório é evidenciado nas ilustrações da última execução — com 300 pontos — de cada combinação de imagens nas Figura 39. Percebe-se que os erros, ilustrados em cores vermelha e verde, são mínimos, enquanto os acertos, em branco e cinza, prevalecem na grande maioria da imagem.

Além disso, a Tabela 8 mostra, com as observações realizadas, que se mantém a confiabilidade entre a imagem 2D (minimap) e o mapa de altura (que formará o mapa 3D), garantindo a funcionalidade de localização em tempo real dentro da execução do jogo.

Figura 40 – Passos do programa final utilizando o preenchimento por inundação.



Fonte: Autoria própria

5 Considerações finais

Neste capítulo, são desenvolvidas as conclusões do trabalho e os trabalhos futuros, visando evoluir com a ciência na tentativa de obter resultados melhores.

5.1 Conclusão

A presente monografia aborda uma solução que viabiliza a criação de mapas 2D e 3D a partir de contornos de dois contextos distintos: uma imagem de um desenho, utilizando seleção por preenchimento por inundação, e uma imagem urbana, empregando segmentação panóptica. A implementação do projeto compreende seis fases distintas: a primeira fase concentra-se na segmentação de imagens; a segunda, na seleção de contornos; a terceira fase envolve a geração procedural de mapas com biomas variados; a quarta etapa consiste no desenvolvimento de testes para avaliar a eficácia da geração procedural; a quinta fase aborda a análise de casos de pós-processamento; e, por fim, a sexta etapa concentra-se na integração das ferramentas mencionadas por meio de uma interface gráfica intuitiva.

Os trabalhos relacionados desempenharam um papel crucial para a conclusão desta monografia. A contribuição de Kirillov (2019) foi essencial para o entendimento e a base da segmentação panóptica, que combina a segmentação de instância e semântica, além de fornecer fundamentos para a técnica de classificação de conjuntos, permitindo a utilização de métricas para validar a hipótese inicial. O modelo eficiente de segmentação panóptica descrito por Mohan et al. (2020) foi utilizado para segmentar as imagens e selecioná-las, viabilizando a criação procedural de mapas para jogos, conforme descrito por Amit (2010), com o contorno escolhido.

A fundamentação teórica proporcionada pelos trabalhos relacionados possibilitou a construção do protótipo e a execução dos testes propostos. Além disso, a proposta inicial foi comprovada, indicando que um aumento nos pontos do diagrama de Voronoi para a geração procedural do mapa resulta em uma maior aproximação do contorno da ilha gerado a partir das técnicas de segmentação.

Conclui-se que todos os objetivos propostos foram alcançados de maneira satisfatória, obtendo resultados positivos em ambos os cenários. A diversificação e personalização dos ambientes virtuais promovem um impacto positivo na experiência do jogador. Contudo, é importante ressaltar que a relação entre a adição de mais pontos e a melhoria dos resultados, juntamente com o tempo de execução, pode apresentar desafios em dispositivos com capacidades de processamento inferiores. Além disso, a dependência da biblioteca

CUDA Toolkit na implementação da inteligência artificial pode limitar sua aplicabilidade em cenários menos ideais.

Nesse contexto, sugere-se a possibilidade de aprimorar a proposta utilizando algoritmos mais eficientes e multiplataforma, ampliando assim a funcionalidade para um público mais abrangente. A automação para Unity revela-se versátil, podendo ser adotada por desenvolvedores para a criação rápida de esboços de mapas 2D/3D ou como funcionalidade para consumidores de jogos, permitindo a geração de mapas a partir de contornos selecionados em fotos segmentadas.

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.
- Empregar paralelismo para otimizar o tempo de execução.
- Aplicar rasterização para aprimorar o tempo de execução.
- Testar o tempo de execução com outros métodos de geração procedural, visando criar 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 5 vezes nas páginas 35, 36, 38, 39 e 41.
- 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 29.
- 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 56.
- 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 29.
- 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 na página 44.
- 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 24.
- BLOXYDEV. *First Person Movement Unity 2022*. 2022. Vídeo no YouTube. Acesso em: 27 nov. 2023. Disponível em: <<https://www.youtube.com/watch?v=yI2Tv72tV7U>>. Citado na página 63.
- 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 4 vezes nas páginas 46, 64, 69 e 70.
- 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 31.
- 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 32.
- CD(3TCL) — tcl8.6-doc — Debian bookworm — Debian Manpages. 2023. Acesso em: 14 Jan. 2024. Disponível em: <<https://manpages.debian.org/bookworm/tcl8.6-doc/ed.3tcl.en.html>>. Citado na página 89.

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 6 vezes nas páginas 44, 45, 46, 64, 69 e 70.

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 19.

CONCEITOS básicos do Unity. S. D. Acesso em: 11 Jan. 2024. Disponível em: <<https://unity.com/pt/learn/get-started>>. Citado na página 87.

CONDA create — conda 23.11.1.dev52 documentation. S. D. Acesso em: 14 Jan. 2024. Disponível em: <<https://docs.conda.io/projects/conda/en/latest/commands/env/create.html>>. Citado na página 89.

CONDA init and conda activate — conda 23.11.1.dev52 documentation. S. D. Acesso em: 14 Jan. 2024. Disponível em: <<https://docs.conda.io/projects/conda/en/latest/dev-guide/deep-dives/activation.html>>. Citado na página 89.

CONDA install — conda 23.11.1.dev52 documentation. S. D. Acesso em: 14 Jan. 2024. Disponível em: <<https://docs.conda.io/projects/conda/en/latest/commands/install.html>>. Citado na página 89.

CORREA, G. *Mover personagem com Unity 3D*. 2015. Acesso em: 3 Dez. 2023. Disponível em: <<https://satellasoft.com/artigo/unity-3d/mover-personagem-com-unity-3d>>. Citado na página 27.

DATASET, C. *Panoptic Semantic Labeling Task - PQ on class-level*. 2023. <<https://www.cityscapes-dataset.com/benchmarks>>. Acesso em: 25 mai. 2023. Citado na página 47.

DEBIAN Manpages. S. D. Acesso em: 7 Jan. 2024. Disponível em: <<https://manpages.debian.org>>. Citado 2 vezes nas páginas 87 e 88.

DEVELOPER, N. *CUDA Toolkit*. 2023. Acesso em: 3 Dez. 2023. Disponível em: <<https://developer.nvidia.com/cuda-toolkit>>. Citado na página 48.

DORMANS, J. Adventures in level design: Generating missions and spaces for action adventure games. Weesperzijde 190, 1097DZ Amsterdam, The Netherlands, 2010. Citado na página 23.

EDUCATIVE. *Overfitting and underfitting*. [S.l.]: Educatice, 2022. <<https://www.educative.io/>>. Acesso em: 01 jun. 2023. Citado na página 37.

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 5 vezes nas páginas 29, 30, 56, 57 e 60.

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 33.

FONSECA, R. *Você conhece a técnica de geração procedural em jogos digitais?* 2022. <<https://www.lambda3.com.br/2022/06/voce-conhece-a-tecnica-de-geracao-procedural-em-jogos-digitais/>>. Acesso em: 3 Dez. 2023. Citado na página 19.

GETTING started with conda. S. D. Acesso em: 11 Jan. 2024. Disponível em: <<https://conda.io/projects/conda/en/latest/user-guide/getting-started.html>>. Citado na página 88.

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 34 e 35.

GIT. S. D. Acesso em: 14 Jan. 2024. Disponível em: <<https://git-scm.com/>>. Citado na página 89.

GIT - git-clone Documentation. S. D. Acesso em: 14 Jan. 2024. Disponível em: <<https://git-scm.com/docs/git-clone>>. Citado na página 89.

GMBH, H. *Instance Segmentation.* 2023. <<https://hasty.ai/docs/mp-wiki/model-families/instance-segmentor>>. Acesso em: 27 mar. 2023. Citado na página 28.

HAT, R. *O que é Linux?* 2023. Acesso em: Dez. 2023. Disponível em: <<https://www.redhat.com/pt-br/topics/linux/what-is-linux#:~:text=O%20Linux%20%C3%A9%20um%20sistema%20operacional%20open%20source,desde%20que%20fa%C3%A7a%20isso%20sob%20a%20mesma%20licen%C3%A7a.>> Citado na página 47.

HAUÑECKER 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 28.

HAYKIN, S. S. *Neural Networks: A Comprehensive Foundation.* [S.l.]: Prentice Hall, 1999. Citado 4 vezes nas páginas 32, 33, 34 e 36.

HE, K. et al. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. Disponível em: <<http://arxiv.org/abs/1703.06870>>. Citado na página 43.

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 na página 32.

KENNY. Procedural generation: An overview. *Medium*, 2021. Acesso em: 3 Dez. 2023. Disponível em: <<https://kentpawson123.medium.com/procedural-generation-an-overview-1b054a0f8d41>>. Citado na página 19.

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 45.

KIRILLOV, A. et al. *Panoptic Segmentation.* 2019. Acesso em: 05 jun. 2023. Citado 9 vezes nas páginas 42, 44, 46, 50, 54, 63, 65, 69 e 70.

KŘÍŽ, B. J. Multi-fractal terrain generation. 2019. Acesso em: 3 Dez. 2023. Citado na página 19.

LEITE, G.; LIMA, E. Soares de. Geração procedural de mapas para jogos 2d. In: . [s.n.], 2015. Disponível em: <https://www.researchgate.net/publication/297704013_Geracao_Procedural_de_Mapas_para_Jogos_2D>. Citado na página 19.

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 49.

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 30.

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 48.

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 48.

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 57.

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 29.

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 32, 33 e 36.

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 30.

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 24.

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 27.

MICROSOFT. *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 26.

- MINICONDA — miniconda documentation. S. D. Acesso em: 11 Jan. 2024. Disponível em: <<https://docs.conda.io/projects/miniconda/en/latest/>>. Citado na página 88.
- MOHAN, R.; GABRIELE, G. *EfficientPS GitHub Issue #23*. 2021. <<https://github.com/DeepSceneSeg/EfficientPS/issues/23>>. Acesso em: 27 nov. 2023. Citado na página 55.
- MOHAN, R.; VALADA, A. *Efficientps*. 2020. Repositório do GitHub. Acesso em: 27 nov. 2023. Disponível em: <<https://github.com/deepsceneseg/efficientps>>. Citado 5 vezes nas páginas 47, 54, 55, 89 e 90.
- 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 47, 48, 49, 50, 54 e 55.
- NEDEA, D. *Confusion Matrix Calculator*. 2020.
url<https://www.mdapp.co/confusion-matrix-calculator-406/>. Acesso em: 21 out. 2023. Citado 5 vezes nas páginas 45, 46, 64, 69 e 70.
- 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 29.
- OPENCV. *About OpenCV*. 2023. Acesso em: 20 out. 2023. Disponível em: <<https://opencv.org/about/>>. Citado 2 vezes nas páginas 30 e 58.
- 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 20, 30 e 56.
- 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 2 vezes nas páginas 30 e 56.
- O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015. Disponível em: <<http://arxiv.org/abs/1511.08458>>. Citado na página 37.
- 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 10 vezes nas páginas 20, 24, 25, 26, 27, 50, 51, 57, 60 e 61.
- PIP install - pip documentation v23.3.2. S. D. Acesso em: 14 Jan. 2024. Disponível em: <https://pip.pypa.io/en/stable/cli/pip_install/>. Citado 2 vezes nas páginas 89 e 91.
- PIP uninstall - pip documentation v23.3.2. S. D. Acesso em: 14 Jan. 2024. Disponível em: <https://pip.pypa.io/en/stable/cli/pip_uninstall/>. Citado na página 91.
- 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 23.
- PROJECT, G. *GCC, the GNU Compiler Collection*. 2023. Acesso em: 3 Dez. 2023. Disponível em: <<https://gcc.gnu.org/>>. Citado na página 48.

PYTORCH. *PyTorch*. 2023. Acesso em: 3 Dez. 2023. Disponível em: <<https://pytorch.org/>>. Citado na página 48.

RAMESH, A. et al. *DALL·E: Creating Images from Text*. 2021. Disponível em: <<https://openai.com/blog/dall-e/>>. Citado na página 31.

REVELE sua criatividade. S. D. Acesso em: 11 Jan. 2024. Disponível em: <<https://unity.com/pt/download>>. Citado na página 87.

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 34.

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 23, 24 e 57.

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 na página 43.

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 19.

SANTOS, P. R. S. dos. *Diagrama de voronoi: Uma Exploração nas Distâncias Euclidianas e do Táxi*. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná - UTFPR, 2016. Citado na página 23.

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 37, 38 e 39.

SCHUMACHER, D. *Synchronized Batch Normalization*. s. d. <<https://serp.ai/synchronized-batch-normalization/>>. Acesso em: 09 jun. 2023. Citado na página 49.

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 31.

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 32.

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 31.

- SZELISKI, R. *Computer Vision: Algorithms and Applications*. [S.l.]: Springer, 2022. Citado na página 28.
- 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 3 vezes nas páginas 36, 38 e 41.
- TEAM, C. M. Image segmentation | techniques & types. *saiwa*, 2023. Disponível em: <<https://saiwa.ai/blog/image-segmentation/>>. Citado na página 20.
- TECHNOLOGIES, U. *Terrain Toolkit 2017*. 2017. Acesso em: 3 Dez. 2023. Disponível em: <<https://assetstore.unity.com/packages/tools/terrain/terrain-toolkit-2017-83490>>. Citado 2 vezes nas páginas 27 e 63.
- 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 3 vezes nas páginas 26, 61 e 63.
- TECHNOLOGIES, U. *Installing the Unity Hub*. S. D. Acesso em: 2 Jan. 2024. Disponível em: <<https://docs.unity3d.com/hub/manual/InstallHub.html>>. Citado na página 87.
- TECHNOLOGIES, U. *Unity Terrain Tools Manual*. s. d. Acesso em: 3 Dez. 2023. Disponível em: <<https://docs.unity3d.com/Packages/com.unity.terrain-tools@4.0/manual/index.html>>. Citado 2 vezes nas páginas 27 e 63.
- 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 6 vezes nas páginas 20, 41, 42, 43, 47 e 54.
- 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 44.
- 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 41, 43 e 54.
- 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 25.
- WIKI, M. *Bioma*. 2022. Acesso em: 3 Dez. 2023. Disponível em: <https://minecraft.fandom.com/pt/wiki/Bioma#cite_note-mojang-1>. Citado na página 20.
- 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 19.
- 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 31.

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 34.

YANNAKAKIS, G. N.; TOGELIUS, J. *Artificial Intelligence and Games*. [S.l.]: Springer, 2018. <<http://gameaibook.org>>. Citado na página 23.

ZAGATA, K.; MEDYŃSKA-GULIJ, B. Mini-map design features as a navigation aid in the virtual geographical space based on video games. *ISPRS International Journal of Geo-Information*, v. 12, n. 2, 2023. ISSN 2220-9964. Acesso em: 3 Dez. 2023. Disponível em: <<https://www.mdpi.com/2220-9964/12/2/58>>. Citado na página 19.

ZOO, P. *Mapeamento em alto nível*. s. d. Acesso em: 3 Dez. 2023. Disponível em: <<https://www.planetzoogame.com/pt-BR/novidades/mapeamento-em-alto-nivel>>. Citado na página 61.

Apêndices

1 Instalação do Unity Hub

O Unity é um motor gráfico e uma a plataforma que é líder mundial para criar e operar conteúdo 3D interativo em tempo real (CONCEITOS..., S. D.). O Unity Hub é o gerenciador de projetos que utiliza o motor gráfico Unity (REVELE..., S. D.).

A instalação do Unity Hub foi realizada por meio do serviço de pacotes do Debian. Foi necessário utilizar o superusuário e adicionar o certificado do repositório do Unity, seguindo os comandos apresentados no Código 1 (TECHNOLOGIES, S. D.). O comando *su* permite que os comandos subsequentes sejam executados com um usuário diferente da sessão; caso não seja informado, é utilizado o usuário *root* ou superusuário. Já o comando *wget* é utilizado para baixar arquivos da internet. Enquanto o comando *gpg* é uma ferramenta que fornece funções de criptografia e assinatura digital usando o padrão OpenPGP. Por fim, o comando *tee* é utilizado para ler o arquivo da entrada padrão e gravar na saída padrão ou em um arquivo (DEBIAN..., S. D.).

Código 1 – Trecho de código com comando UNIX para adicionar o certificado do repositório do Unity (TECHNOLOGIES, S. D.)

```
su
wget -qO -https://hub.unity3d.com/linux/keys/public | \
gpg --dearmor | \
tee /usr/share/keyrings/Unity_Technologies_ApS.gpg > /dev/null
```

Com o certificado do Unity devidamente incluído, o repositório do Unity foi adicionado ao controlador de pacotes do Debian, as informações dos pacotes foram atualizadas e o Unity foi instalado. Os comandos necessários estão detalhados no Código 2. O comando *sh* no Debian serve como um link para o Dash, que, por sua vez, atua como um interpretador de comandos para o sistema. Vale ressaltar que o comando *echo* dentro do comando *sh* tem a função de imprimir uma linha de texto no console. Ao incorporar o símbolo *sinal de maior que*, o texto será direcionado para a impressão em um arquivo (DEBIAN..., S. D.).

Código 2 – Trecho de código com comando UNIX para adicionar o repositório do Unity (TECHNOLOGIES, S. D.)

```
sh -c 'echo "deb [signed-by=/usr/share/keyrings/Unity_Technologies_ApS.gpg] \
https://hub.unity3d.com/linux/repos/deb stable main" > \
/etc/apt/sources.list.d/unityhub.list'
apt update
apt-get install unityhub
```

.2 Instalação do Miniconda

Conda, uma ferramenta versátil de gerenciamento de pacotes e ambientes, é compatível com sistemas operacionais Windows, macOS e Linux ([GETTING..., S. D.](#)).

Miniconda, um instalador leve e gratuito para o Conda, representa uma versão compacta do Anaconda, incluindo apenas o Conda, Python, os pacotes essenciais para ambos, e um número limitado de outros pacotes ([MINICONDA..., S. D.](#)).

Para a administração das dependências do projeto, optou-se pelo uso do Miniconda. O procedimento de instalação desse software pode ser visualizado no Código 3 ([MINICONDA..., S. D.](#)). O comando *mkdir* é empregado para criar diretórios ou pastas, caso estes não existam. Paralelamente, o comando *wget* é utilizado para baixar arquivos; nesse contexto, ocorre o download do instalador do Miniconda. O comando *bash*, um interpretador GNU Bourne-Again SHell, é empregado para a execução do arquivo do Miniconda. Por fim, o comando *rm* é aplicado para remover arquivos e diretórios, sendo utilizado, neste caso, para excluir o script de instalação do Miniconda ([DEBIAN..., S. D.](#)).

Código 3 – Trecho de código com comando UNIX para baixar o Miniconda ([MINICONDA..., S. D.](#))

```
mkdir -p ~/miniconda3
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh \
-O ~/miniconda3/miniconda.sh
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm -rf ~/miniconda3/miniconda.sh
```

Após a instalação do Miniconda, torna-se necessário ativar o ambiente do conda para possibilitar a interação com o shell, conforme descrito na documentação do Conda ([GETTING..., S. D.](#)). A abordagem para realizar essa ativação é ilustrada no trecho de código apresentado no Código 4. O comando *conda init* é empregado com o propósito de inicializar o conda nos shells bash (interpretador GNU Bourne-Again SHell) e zsh (interpretador de comandos UNIX) ([DEBIAN..., S. D.](#); [GETTING..., S. D.](#)).

Código 4 – Trecho de código com comando para inicializar o Miniconda ([MINICONDA..., S. D.](#))

```
~/miniconda3/bin/conda init bash
~/miniconda3/bin/conda init zsh
```

.3 Repertório do projeto

Todas as informações pertinentes aos códigos, texto e anotações deste trabalho encontram-se disponíveis no repositório online hospedado no GitHub, acessível por meio do link <<https://github.com/Lds20k/senac-tcc>>.

.4 Execução do projeto

Para a realização do projeto, foi organizado um repositório contendo todas as dependências. Para obter o projeto mencionado na apêndice .3, recomenda-se utilizar o sistema de controle de versões distribuído Git (GIT, S. D.). Através do comando indicado na Código 5, é possível efetuar essa operação. O comando *git clone* realiza o download do projeto a partir de um repositório remoto (GIT..., S. D.).

Código 5 – Trecho de código com comando do git para baixar o projeto

```
git clone https://github.com/Lds20k/senac-tcc.git
```

Após ter realizado o download do projeto e instalado as dependências mencionadas nos apêndices .1 e .2, é possível configurar o ambiente para a execução do projeto. Para realizar essa configuração, acesse o diretório do projeto e siga os comandos indicados no Código 6.

O comando *conda env create* é utilizado para criar um novo ambiente, incorporando as dependências especificadas no arquivo *environment.yml*. Posteriormente, o comando *conda activate* é empregado para ativar o ambiente denominado *senac-tcc*. Adicionalmente, o comando *conda install* é empregado para instalar outras dependências que não estão incluídas no arquivo *environment.yml*. Por fim, o comando *pip install* é utilizado para instalar módulos específicos para o Python, conforme listado no arquivo *requirements* (CONDA..., S. D.a; CONDA..., S. D.b; CONDA..., S. D.c; PIP..., S. D.a). Este conjunto de ações possibilita a adequada configuração do ambiente, preparando-o para a execução bem-sucedida do projeto.

Código 6 – Trecho com instruções para instalação de dependências utilizando conda e pip
(MOHAN; VALADA, 2020)

```
conda env create -n senac-tcc --file=environment.yml
conda activate senac-tcc
conda install pytorch==1.7.0 torchvision==0.8.0 torchaudio==0.7.0 \
    cudatoolkit=10.2 cudatoolkit-dev -c pytorch -c conda-forge
pip install -r requirements.txt
```

Após a configuração adequada das dependências e do ambiente, o próximo passo consiste na instalação da implementação do EfficientNet e do EfficientPS. Para realizar essa tarefa, é necessário seguir os comandos apresentados no Código 7.

O comando *cd* é utilizado para alterar o diretório de trabalho para o caminho especificado como primeiro argumento. Por sua vez, o comando *python* é empregado para executar um script Python que realiza tanto a compilação quanto a instalação do EfficientNet e do EfficientPS, respectivamente (CD(3TCL)..., 2023; MOHAN; VALADA, 2020).

Ao seguir essas instruções, a implementação desses programas será instalada de maneira apropriada no ambiente configurado, proporcionando os recursos necessários para o desenvolvimento e execução bem-sucedidos do projeto.

Código 7 – Trecho de código com comandos para instalação da EfficientNet e EfficientPS
(MOHAN; VALADA, 2020)

```
cd src/efficientNet
python setup.py develop
cd ..
python setup.py develop
```

Realize o download do modelo pré-treinado *KITTI* disponível no repositório através do seguinte endereço:

<<https://github.com/DeepSceneSeg/EfficientPS#pre-trained-models>>

Após o download, renomeie o arquivo baixado para *model_kt.pth* e, em seguida, mova o arquivo do modelo pré-treinado para a pasta *src*. Certifique-se de que o arquivo esteja localizado corretamente dentro do diretório mencionado antes de prosseguir com o uso do modelo no projeto. Essa ação garante a correta integração do modelo pré-treinado no ambiente de desenvolvimento.

Finalmente, para a execução do projeto, é crucial observar a estrutura da pasta de execução, uma vez que o diretório de execução deve iniciar obrigatoriamente a partir da raiz do projeto. No trecho apresentado no Código 8, encontra-se o comando para a execução do projeto.

Certifique-se de que o diretório de execução está corretamente configurado para iniciar a partir da raiz do projeto antes de executar o comando mencionado. Isso garantirá que o projeto seja iniciado corretamente, utilizando os recursos e dependências configurados durante o processo anterior.

Código 8 – Trecho de código com comando para execução do projeto

```
python src/main.py
```

Para utilizar o projeto no Unity Hub, proceda da seguinte maneira: abra o projeto *3d_map* localizado dentro da pasta *unity*, na raiz do projeto. Durante o processo, assegure-se de realizar a instalação da versão recomendada das dependências, conforme indicado pela plataforma Unity Hub. Este procedimento garantirá a correta configuração do ambiente de desenvolvimento, possibilitando a utilização do projeto no Unity Hub.

Depois de abrir o projeto no Unity, acesse e abra a cena localizada dentro da pasta *Scenes*. Agora, basta pressionar o botão *play*, e o script se encarregará de carregar as imagens e arquivos gerados pelo programa de geração de mapas.

No caso de ocorrer um erro relacionado ao módulo Python do Numpy, você pode corrigi-lo desinstalando e reinstalando o módulo. O procedimento está detalhado no Código 9. O comando *pip uninstall* remove um pacote previamente instalado, enquanto o *pip install* realiza a instalação de um pacote ([PIP…, S. D.a](#); [PIP…, S. D.b](#)).

Código 9 – Trecho de código com os comandos para reinstalação do Numpy

```
pip uninstall numpy  
pip install numpy
```