UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

LEONARDO SCHULTHAIS SENNA

IDENTIFICAÇÃO DE SÍMBOLOS PRÉ-DETERMINADOS MONOCROMÁTICOS UTILIZANDO TÉCNICAS DE PROCESSAMENTO DE IMAGEM E APRENDIZAGEM DE MÁQUINA PARA APLICAÇÃO EM JOGOS DIGITAIS

VITÓRIA – ES
JULHO/2018

LEONARDO SCHULTHAIS SENNA

IDENTIFICAÇÃO DE SÍMBOLOS PRÉ-DETERMINADOS MONOCROMÁTICOS UTILIZANDO TÉCNICAS DE PROCESSAMENTO DE IMAGEM E APRENDIZAGEM DE MÁQUINA PARA APLICAÇÃO EM JOGOS DIGITAIS

Parte manuscrita do Projeto de Graduação do aluno **Leonardo Schulthais Senna**, apresentada ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para aprovação na disciplina "ELE08553 – Projeto de Graduação 2".

_____
Prof. Dr. Klaus Fabiano Côco
Orientador

_____
Prof. Dr. Patrick Marques Ciarelli
Avaliador

_____
MSc. Thaís Pedruzzi do Nascimento
Avaliadora

VITÓRIA – ES
JULHO/2018

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE GRÁFICOS

SUMÁRIO

1 INTRODUÇÃO

1.1 Apresentação

Graças ao aumento da capacidade de processamento de computadores nos últimos anos e ao amadurecimento da indústria de entretenimento, jogos têm se tornado cada vez mais complexos, aproveitando todo o poder computacional disponível para tornar a experiência mais agradável ao jogador, seja através de gráficos complexos ou algoritmos de detecção de colisão mais precisos, por exemplo.

Em jogos com ações em tempo real, onde, por exemplo, o jogador controla um avatar que obedece a comandos do jogador instantaneamente, o jogo deve fazer todo o processamento do mundo virtual em dado instante o mais rápido possível para manter a sensação de fluidez e instantaneidade. Quando executados em sistemas apropriados, jogos costumam oferecer pelo menos 60 *frames* por segundo, ou seja, faz-se todo o processamento do mundo virtual em dado instante e apresenta-se a imagem do jogo para o jogador 60 vezes por segundo. Dado o limite de tempo para os processamentos, jogos têm evoluído juntamente à computação de modo a equilibrar a qualidade, como sofisticação e aspectos de suavidade das imagens no processamento.

Uma possibilidade de dinâmica de jogo é o uso de desenho de símbolos para execução ou auxílio a execução de ações, substituindo simples apertos de botões para execução direta de ações. Para que isso seja possível, é necessário determinar se o jogador executou um símbolo válido dentro das possibilidades de símbolos do jogo, e, se representou, qual é o símbolo. Para isso, utilizam-se técnicas de processamento de imagem e aprendizado de máquina, onde tenta-se alcançar um equilíbrio entre acurácia da detecção do símbolo, poder computacional necessário para o processo e outros requisitos da detecção por parte do *design* do jogo.

Neste trabalho será abordado o reconhecimento de imagens adquiridas a partir do desenho de símbolos pré-determinados na tela do jogo, representando o gesto que o jogador realiza com o *mouse*. Deste modo, a orientação dos desenhos será sempre a mesma, definindo o plano do desenho como o plano da tela do jogador.

1.2 Identificação dos Símbolos

De modo simplificado, pode-se dividir o processo de identificação dos símbolos como a união de três tipos de processos diferentes pelos quais a imagem que se quer identificar passará:

- Tratamento da imagem, onde se divide, simplifica ou transforma a imagem para uma representação mais simples para que seja possível ou mais fácil de se trabalhar com a imagem nas próximas etapas. Essa etapa se refere a processos como fechamento e abertura de imagem para tornar o desenho mais parecido com o modelo de desenho original e limiarização para transformar uma imagem em escala de cinza em uma imagem em preto e branco;
- Extração de características, onde se identifica características genéricas pré-determinadas de uma imagem, como, por exemplo, quantidade de regiões separadas na figura;
- Decisão final, onde utiliza-se dados extraídos nas etapas anteriores para verificar se o desenho representa um símbolo válido e qual é esse símbolo. Foi utilizada uma rede neural para a realização desta etapa.

Embora possam existir ambiguidades na classificação dos processos de identificação de imagens, entende-se pela divisão dos processos que imagens, em geral, precisam ser simplificadas para que se extraiam as características usadas para a avaliação final da imagem.

Nota-se que o processo pode não ser tão simples quanto seguir as etapas na ordem descrita, podendo haver vários processos diferentes para segmentação da imagem e extração de características que servirão como entradas para o algoritmo de decisão.

1.3 Justificativa

Com o aumento da facilidade de se criar jogos com o surgimento de ferramentas para auxílio de criação de jogos e programas em geral, o mercado tem ficado cada vez mais competitivo, exigindo que criadores de jogos ofereçam dinâmicas de jogo mais adaptáveis, variadas e divertidas para o jogador.

Uma possível dinâmica de jogo que pode atender a esses requisitos é a oferta da possibilidade de o jogador desenhar símbolos para executar ou auxiliar a execução de ações, substituindo os simples apertos de botões, dinamizando mais o jogo e possibilitando um aumento de *skill ceiling* do jogo, isto é, o quão bom um jogador pode ficar em um jogo, uma característica importante principalmente para jogos competitivos como *Overwatch* ou *League of Legends*.

Técnicas de processamento de reconhecimento de imagens já são utilizadas hoje em jogos como *Magic Touch: Wizard for Hire*, onde o jogo identifica o símbolo que o usuário desenha na tela e relaciona com os balões com símbolos correspondentes; *Pokémon Sun* e *Pokémon Moon*, que reconhecem a expressão facial do jogador com a câmera do console para que o *Pokémon* possa reagir; *Mario Party*, onde um dos desafios é o jogador desenhar rapidamente o símbolo na tela utilizando o movimento do controle; e *Harry Potter: Hogwarts Mystery*, onde o jogador deve desenhar o símbolo equivalente ao gesto da varinha para realizar magias.

1.4 Conjunto de Imagens

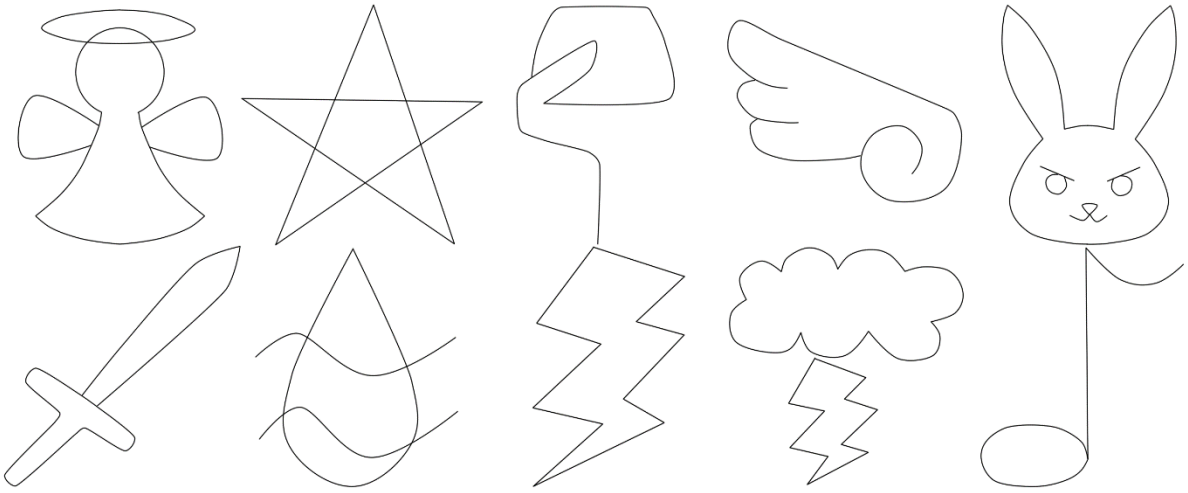
Para a utilização e avaliação apropriada dos métodos de reconhecimento de símbolos, foi criado um conjunto de imagens de símbolos, desenhados em suas formas ideais; e um conjunto de desenhos dos símbolos feitos de maneira simplificada, de modo a simular as imagens extraídas da mecânica de desenho do jogo, tendo em mente que, durante o jogo, os jogadores tenderiam a desenhar as imagens de maneira ágil e sem preocupação com o formato e proporções exatas do desenho original, assim como, por exemplo, as pessoas escrevem letras que não são simplesmente uma tentativa fiel de replicação exata da forma original da letra que aprenderam.

As imagens utilizadas no trabalho são imagens de 64x64 *pixels*, em tons de cinza e armazenadas sem perda em formato PNG. Variações nessas especificações, como mudança na escala das imagens e utilização de preto e branco ao invés de tons de cinza, não devem alterar o bom funcionamento do trabalho, desde que as consequências da mudança da especificação sejam devidamente consideradas. Por exemplo, todas as imagens utilizadas devem ter as mesmas dimensões, e a utilização apropriada da cor do fundo e do objeto depende do que as funções de tratamento de imagem e extração de características consideram como fundo e como objeto.

Para o conjunto de imagens do trabalho, foram utilizados símbolos de diferentes complexidades: desde imagens simples, com poucas áreas, traços e curvas, como uma colcheia,

até imagens mais elaboradas, como o coelho símbolo da D.Va, do jogo *Overwatch*. Além disso, priorizou-se utilizar modelos com características parecidas entre si e modelos com características diferentes entre si, para confirmar a robustez do processo, independente de quão parecidas ou diferentes as imagens são. Por fim, também foi decidido utilizar um modelo que contém outro, no caso do símbolo de raio e símbolo de raio com nuvem, para testar a capacidade de diferenciação do processo nestes casos. O conjunto de imagens contendo os objetos a serem utilizados como modelos para os desenhos é mostrado na Figura 1. A primeira linha apresenta, da esquerda para a direita, os modelos das classes 1 até 5, e a segunda linha, os modelos das classes 6 até 10.

Figura 1 - Conjunto dos modelos das classes de imagens



Fonte: O próprio autor.

As imagens que representam os desenhos dos jogadores e que foram utilizadas para treinamento, validação e teste do projeto são mostradas na Figura 2. O conjunto consiste de 18 imagens para cada uma das 10 diferentes classes, desenhadas uma a uma pelo próprio autor e em diferentes dias utilizando o mouse, simulando a obtenção dos desenhos de um jogo.

Testes mais robustos poderiam ser feitos com desenhos elaborados por outras pessoas, todavia, para efeito de teste inicial mais controlado visando o estudo da viabilidade da metodologia, optou-se pelo desenho livre de uma única pessoa.

Figura 2 - Conjunto das imagens de treinamento, validação e teste

Fonte: O próprio autor.

Para este trabalho, também foi determinado como requerimento os objetos dos desenhos serem grandes o suficiente para cobrir o máximo de área da imagem possível. Então, é exigido que os objetos dos desenhos encostem ou cheguem nas proximidades das bordas da imagem quando possível, para que não haja variação de escala das imagens. Esse requerimento não restringe a implementação em jogos, já que basta tornar os limites do objeto desenhado pelo jogador as bordas da imagem para o processo de renderização.

1.5 Objetivo

O objetivo do trabalho é a criação, teste, análise e avaliação de um processo para a classificação de desenhos de símbolos monocromáticos pré-determinados sem variação de escala adquiridos em jogos feitos por jogadores, que podem apresentar grandes variedades em características e complexidades ou similaridades, com o objetivo de implementação em um jogo de tempo real para adicionar valor competitivo ao jogo.

2 EMBASAMENTO TEÓRICO

2.1 Segmentação de Imagens

O primeiro conjunto de técnicas a ser utilizado é formato por técnicas relacionadas a segmentação das imagens, em que subdivide uma imagem em suas partes ou objetos constituintes (GONZALEZ; WOODS, 2000, p. 295). O objetivo principal da segmentação é reduzir a informação para as primeiras etapas de análise, de modo a separar e simplificar dados da imagem, que geralmente são usados em etapas futuras, de dados desnecessários para o processo de reconhecimento da imagem (YOGAMANGALAM; KARTHIKEYAN, 2013, p. 307).

2.1.1 Limiarização

A limiarização é uma operação de separação e classificação de *pixels* baseada na intensidade do *pixel* analisado. Para imagens com *pixels* de valores que, predominantemente, são claramente distintos entre áreas ou objetos da imagem, é possível separar essas áreas ou objetos estabelecendo um limiar que classifica a área ou objeto ao qual o *pixel* pertence a partir da sua intensidade. Para melhor aproveitamento da técnica em imagens com imperfeições, diferentes métodos e abordagens surgiram, como limiarização global ou local, que define se o algoritmo é aplicado em toda a imagem de uma só vez ou de região em região, além de métodos como o de Kittler, Pun ou Otsu (ABDULLAH; PIRAHANSIAH; SAHRAN, 2014).

2.1.2 Regiões

Uma imagem pode ser dividida em regiões utilizando técnicas de segmentação orientadas a regiões, como o crescimento por agregação de *pixels*, onde se define *pixels* iniciais de regiões em uma imagem e cada um desses pontos agrupam ou não pontos vizinhos à sua região a partir da avaliação de características dos *pixels*, como intensidade de cor, e cada um desses pontos vizinhos repete o algoritmo recursivamente, até que a imagem esteja devidamente dividida. Alternativamente, pode-se começar com uma imagem completamente dividida e classificada em regiões semi-aleatórias, e dividir ou unir essas regiões a partir de avaliações das características dos *pixels*, como no método citado anteriormente.

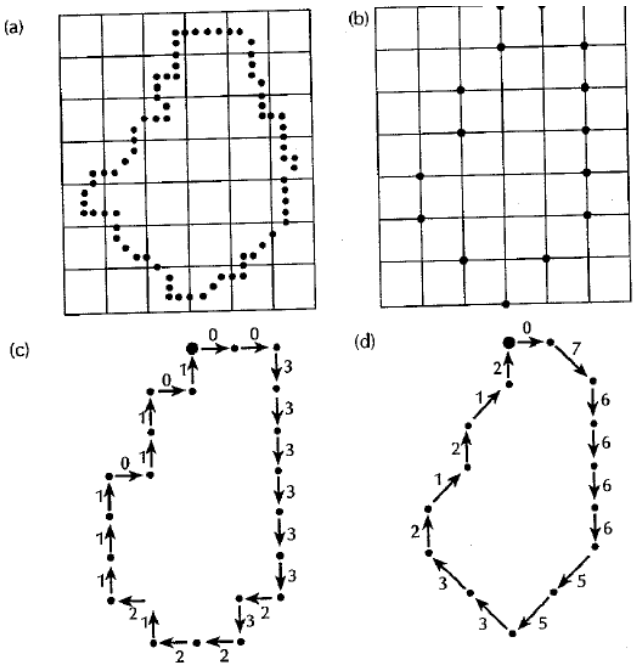
2.2 Representação e Descrição

Essa etapa consiste em métodos para se representar a imagem de maneiras mais simples para fazer sua avaliação, e definir características para a mesma, como o tamanho de uma região ou orientação de uma linha na imagem.

2.2.1 Representação

É possível representar uma imagem que consiste apenas de linhas por esquemas, em geral, mais simples para se usar em computação. Por exemplo, em uma imagem com fundo branco e objeto preto, para uma imagem composta por uma linha que forma um caminho fechado, após subamostrá-la, é possível defini-la em termos da direção na qual o próximo *pixel* preto se encontra, partindo-se de qualquer *pixel* preto da imagem. Se a subamostragem for feita corretamente, o resultado é um vetor de direções representando o polígono subamostrado da imagem, como visto na Figura 3.

Figura 3 - Aplicação de código da cadeia



Fonte: GONZALEZ; WOODS, (2000).

Segundo Gonzalez e Woods (2000), existem outras maneiras de simplificar tal imagem, como a determinação de um perímetro mínimo após uma subamostragem da imagem ou a determinação de vértices a partir da distância de uma linha traçada ao longo do interior da região dentro do caminho fechado pela linha. Essas técnicas possuem o ponto positivo de abstração de detalhes, onde se representa as características básicas de um formato e abstrai-se de detalhes.

Essa propriedade é diretamente relacionada à robustez da representação em relação ao ruído (THESSALONIKI, 1998, p. 6).

A assinatura de uma imagem é um método de representação promissor para este trabalho. De acordo com Gonzalez e Woods (2000), ela se define pela representação unidimensional de uma fronteira, onde, por exemplo, em uma representação de assinatura radial, são registradas as distâncias entre o centro escolhido para a assinatura e a linha em um determinado ângulo. Assinaturas são invariantes à translação, e algumas técnicas podem torná-las flexíveis à mudanças de rotação e escala.

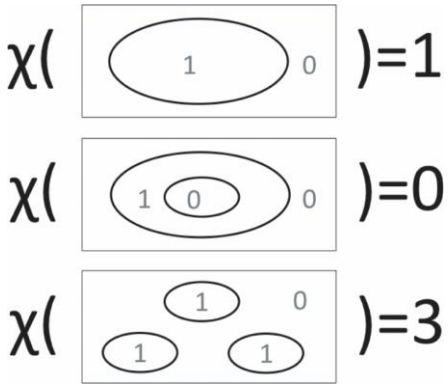
A definição do esqueleto de uma região também pode ser útil, com algoritmos para se definir segmentos interiores às regiões que representem a forma geral da região.

2.2.2 Descrição

Descritores são parâmetros para descrever características concretas sobre a imagem ou um objeto nela. Por exemplo, segundo Gonzalez e Woods (2000), o perímetro total de uma região fechada em uma imagem pode ser considerado um descritor. Descritores de fronteira mais complexos incluem os descritores de Fourier, que são obtidos aplicando a transformada discreta de Fourier para vários $s(k)$, com k sendo o índice correspondente a cada ponto que constitui uma fronteira em uma imagem, e $s(k)$ sendo definido como um número complexo, onde $x(k)$ é a parte real e $y(k)$ é a parte imaginária, sendo $x(k)$ e $y(k)$ as coordenadas do ponto de índice k. Assim, é possível representar a imagem unidimensionalmente, com precisão variável, dependendo de quantos coeficientes de Fourier forem usados para o processo.

Descritores regionais envolvem a caracterização por regiões de uma imagem. Por exemplo, uma propriedade topológica é o número de Euler, definido pela subtração do número de componentes conexos de uma região pelo número de buracos nas regiões (GONZALEZ; WOODS, 2000), sendo componentes conexos os conjuntos de *pixels* de cor diferente da cor de fundo da imagem que estão conectados, ou seja, para quaisquer dois *pixels* neste conjunto, há um caminho formado por *pixels* também pertencentes ao conjunto conectando os dois; e buracos as regiões de cor iguais à cor de fundo da imagem contidas inteiramente em componentes conexos. A Figura 4 mostra o valor do número de Euler para diferentes imagens, sendo 1 os componentes conexos e 0 os buracos.

Figura 4 - Número de Euler para diferentes casos



Fonte: MIURA; NAKADA. (2014)

Os momentos de uma imagem também podem ser úteis como descritores regionais. Estes podem ser encontrados a partir da definição de momentos para uma função contínua bidimensional $f(a,b)$, onde a e b são parâmetros quaisquer. De acordo com Gonzalez e Woods (2000), tem-se a Equação (4):

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p\, y^q f(a,b)\,da\,db \qquad (1)$$

onde m_{pq} é o momento de ordem $(p+q)$.

Segundo Gonzalez e Woods (2000), existe um conjunto de momentos de uma imagem que são invariantes à translação, rotação e escala, uma propriedade útil para o objetivo deste trabalho.

2.2.3 Morfologia

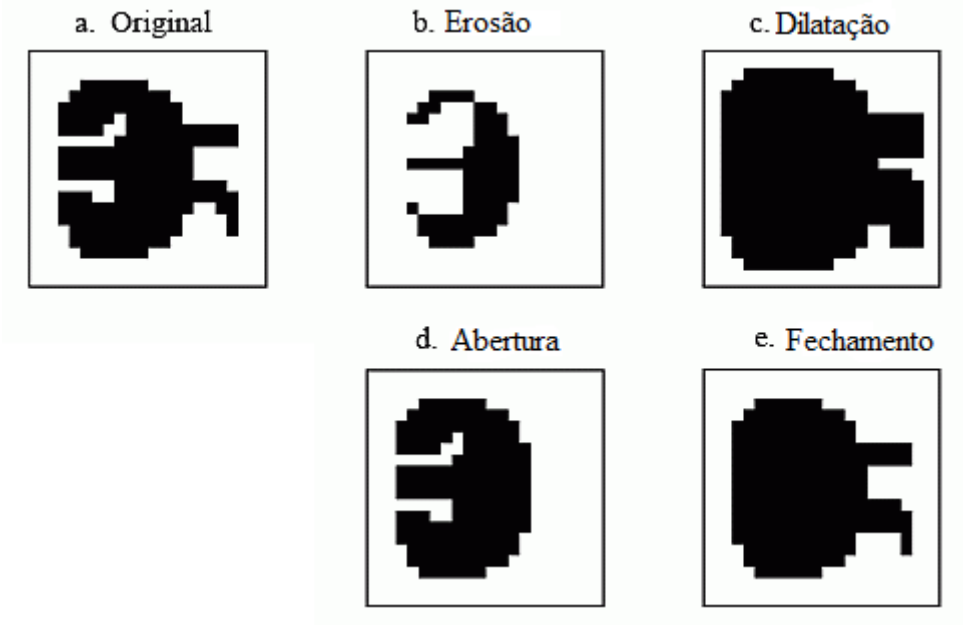
Técnicas morfológicas podem ser utilizadas em uma imagem para manipulações e análises de regiões. De acordo com Girod (2013), inicialmente focado em imagens binárias, o processamento morfológico de imagens já foi generalizado para imagens em tons de cinzas.

Duas operações morfológicas básicas são erosão e dilatação, que respectivamente reduzem ou ampliam regiões de uma imagem a partir de verificações iterativas com um elemento estruturante, que é uma região definida especialmente para essas operações. Nessas operações, o elemento estruturante percorre os pixels dos objetos nas imagens para adicionar ou remover *pixels* do objeto. A partir dessas operações, derivam-se operações mais complexas, como

abertura e fechamento, que são aplicações de erosão seguida por dilatação e dilatação seguida por erosão, respectivamente.

De acordo com Girod (2013), essas duas operações são capazes de suavizar uma estrutura sem mudar drasticamente o seu tamanho. A operação de abertura remove áreas pequenas de uma região em uma imagem, e, complementarmente, a operação de fechamento remove pequenos buracos dentro de uma região da imagem. Esses efeitos geralmente são desejáveis para detecções no processamento de imagem. As operações morfológicas apresentadas podem ser observadas na Figura 5.

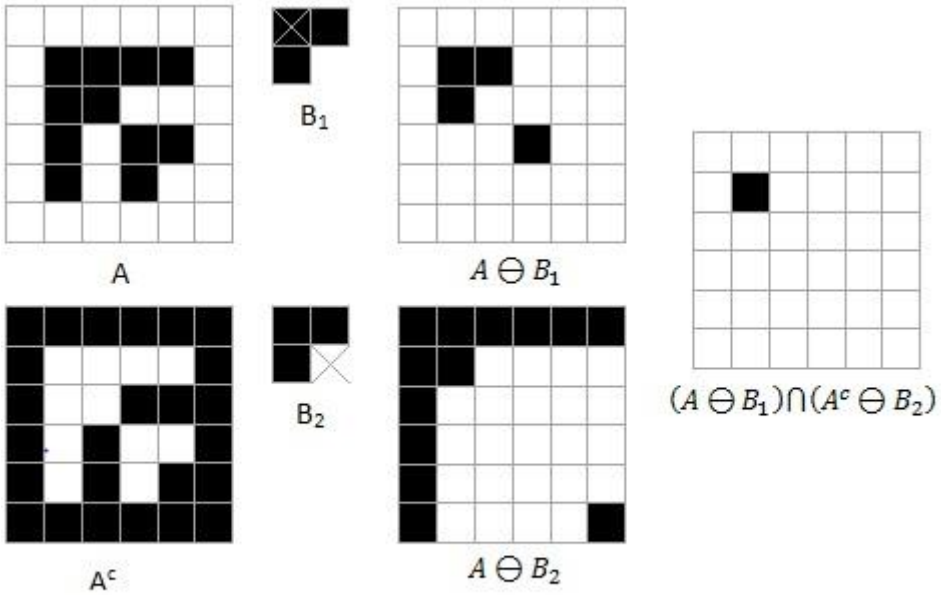
Figura 5 - Operações morfológicas básicas



Fonte: SMITH, (1997). Tradução do autor.

A transformada *hit-or-miss* é capaz de detectar formas, possivelmente uma região onde se encontra um símbolo que se queira identificar. Realizada utilizando operações lógicas em conjuntos (operações E e NÃO), erosão e dilatação, a *hit-or-miss* deixa apenas o contorno das regiões de interesse em uma imagem, eliminando todo o resto, o que não só pode servir como a própria detecção de um símbolo ou redução da área de busca, sendo que, segundo Girod (2013), a operação pode ser adaptada para ser invariante a translações, rotações e mudanças de escala. A Figura 6 ilustra um processo da transformada *hit-or-miss*, sendo “A” a imagem a ser analisada, “B_1” o elemento estruturante e a imagem à direita o resultado final da transformada.

Figura 6 - Transformada hit-or-miss



Fonte: BOCK, (2010).

Existem outras operações morfológicas derivadas a partir de simples combinações lógicas e operações morfológicas mais simples, como a extração de fronteiras, realizada a partir da diferença entre o conjunto de uma região com sua erosão, ou o preenchimento de uma região, realizado com uma dilatação de um elemento estruturante e sua interseção com o complemento da região de interesse.

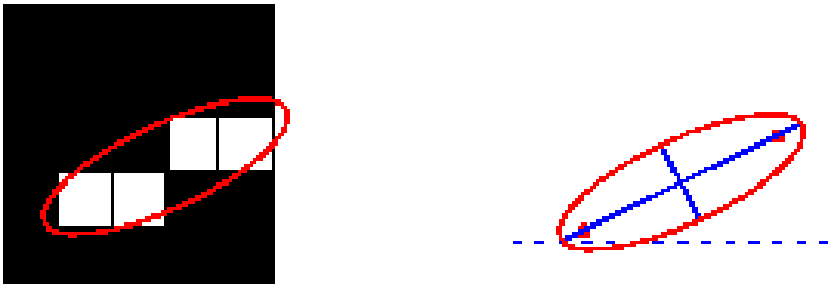
2.3 Extração de Características

Para o método de classificação de imagens por características, é necessária a obtenção de um número razoável de características da imagem para se distinguir apropriadamente cada classe. Para este trabalho, as características utilizadas foram o número de Euler da imagem (como explicado na Seção 2.2.2), a área, a área convexa, a excentricidade, os tamanhos do maior e menor eixo, e o perímetro.

A área é calculada como o número de *pixels* na região interior ao objeto na imagem. A área convexa é a área calculada a partir da envoltória convexa do objeto da imagem (um polígono contornando a imagem de modo que não haja regiões côncavas no polígono). A excentricidade é a relação da distância entre os focos da elipse mínima que envolve o objeto da imagem e o tamanho de seu maior eixo. Os tamanhos do maior e menor eixo são calculados considerando os eixos dessa mesma elipse. A Figura 7 mostra, à esquerda, um objeto envolvido por uma

elipse mínima. À direita, os pontos vermelhos mostram os focos da elipse em cima do eixo maior. O perímetro é dado pelo número total de *pixels* do objeto. Os algoritmos específicos para a obtenção de cada uma dessas características não serão abordados neste trabalho, mas são baseados nas técnicas de segmentação, representação e descrição de imagens abordadas neste trabalho para identificar as regiões dos objetos da imagem e suas características.

Figura 7 - Elipse mínima de um objeto



Fonte: MEASURE, s.d.

2.4 Reconhecimento e Decisão

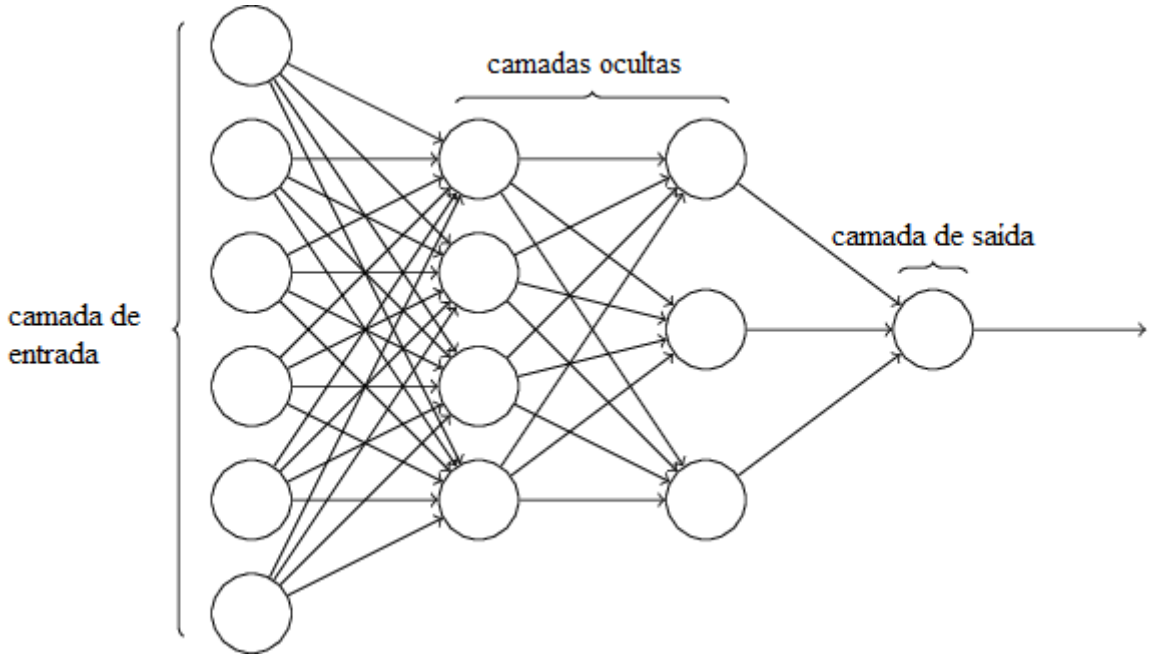
Após uma sequência de processamentos como os das etapas anteriores, geralmente é necessário um método para a avaliação final do processo, que toma as saídas das etapas anteriores como entrada, define se algum símbolo foi identificado na imagem e o classifica.

O classificador de distância mínima é um método simples de decisão, onde se compara algum valor extraído da imagem com o valor equivalente nas imagens originais dos símbolos, e, calculando o erro para cada símbolo, se decide qual símbolo que mais se aproxima da imagem, considerando aquele com menor erro. Os valores podem ser, por exemplo, conjuntos de valores de gradiente da imagem.

Uma rede neural, no entanto, pode oferecer mais flexibilidade para dados de entrada e maior precisão na decisão, inclusive com a possibilidade da decisão de que nenhum símbolo se assemelha com o desenho de entrada. De acordo com Kriesel (s.d.), uma rede neural consiste de unidades de processamento simples, os neurônios, e conexões direcionadas e com pesos entre os neurônios. Os neurônios recebem várias entradas, cada uma com um valor designado, e soma todas as suas entradas, considerando o valor designado para cada como um termo para a soma. O resultado dessa soma, então, é a saída dos neurônios, que pode servir como entrada para outros neurônios ou como saída da rede neural, ou seja, a resposta da rede neural para o processo.

Redes neurais podem variar em termos de quantidade de camadas (relacionado ao número de neurônios em série) e número de neurônios nas camadas, possibilitando, segundo Gonzalez e Woods (2000), diversos níveis de complexidade de classificação. A Figura 8 ilustra uma rede neural multicamada, com os neurônios sendo representados por círculos e as conexões representadas por setas.

Figura 8 - Rede neural multicamada



Fonte: NIELSEN, (2015). Tradução do autor.

Além disso, uma rede neural não precisa ser projetada completamente antes de ser funcional: existem algoritmos para treinamento de redes neurais, que, ao receber e processar um determinado conjunto de entradas, utilizam a saída da rede neural no estado atual e a resposta esperada como parâmetros para o ajuste dos pesos das conexões dos neurônios da rede, que se adaptam para corrigir a resposta para aquela entrada. Ou seja, com a topologia e características da rede neural definidas, a rede neural é treinada para classificar entradas similares às entradas utilizadas para seu treinamento, sem necessidade de projetá-la treinada à priori. Por exemplo, um método possível para o treinamento de uma rede neural de camada única é a regra delta, na qual o erro (diferença entre a saída e o resultado esperado) da rede neural, de acordo com Ingrid (s.d.), é iterativamente minimizado utilizando a Equação (5) para mudança de peso de uma conexão entre neurônios:

$$\Delta\omega_{ij} = r \cdot a_i \cdot e_j \quad (2)$$

onde $\Delta\omega_{ij}$ representa a mudança de peso na conexão especificada, r representa a taxa de aprendizagem, a_i representa a diferença entre o valor esperado e o valor obtido e e_j representa o valor de saída atual do neurônio.

Assim, com um conjunto de desenhos criados exclusivamente para o treinamento da rede neural e parametrização correta (quanto ao número de camadas, número de neurônios, tipos de entradas, etc.), é possível adaptá-la para funcionar como um método de decisão preciso e rápido o suficiente para aplicação em um jogo digital.

2.4.1 Rede LVQ

A rede LVQ (*Learning Vector Quantization*) é uma rede neural que combina aprendizagem competitiva com supervisão. Segundo Leung (2000), o sistema da rede neural é representado por um conjunto de N vetores, sendo N igual ao número de classes possíveis para se classificar uma entrada. Cada um dos vetores que representam a rede assume uma posição em um espaço D-dimensional, sendo D o número de entradas da rede neural.

Para classificar o vetor de entrada, a rede neural calcula as distâncias euclidianas entre o vetor de entrada e os vetores que representam o sistema da rede neural. A classe correspondente ao vetor que resultou na menor distância é tida como “classe vencedora”, e o neurônio na saída da rede correspondente a esse vetor será completamente ativado, enquanto os outros são completamente desativados, seguindo o princípio “vencedor leva tudo”.

No seu aprendizado, sendo uma rede de aprendizagem supervisionada, a classe encontrada é comparada com a classe real do vetor de entrada. Se a rede neural classificou o vetor de entrada corretamente, o ponto equivalente ao vetor representante da classe na rede neural é espacialmente movido em direção ao ponto equivalente ao vetor de entrada, reduzindo a distância entre o vetor de entrada e o vetor representante da classe. Contrariamente, no caso de classificação errada, o ponto equivalente ao vetor representante da classe errada na rede neural é espacialmente movido em direção contrária ao ponto equivalente ao vetor de entrada, aumentando as distâncias entre os dois vetores.

2.4.2 Perceptron Multicamadas

Uma perceptron multicamadas, também chamada de rede neural rasa, é um modelo simples de uma rede neural contendo apenas a camada de entrada, com o número de neurônios igual ao número de variáveis de entrada, de modo que cada neurônio de entrada recebe o valor de uma característica da imagem; uma única camada intermediária, com um número de neurônios variável definido para cada projeto; e a camada de saída, com um número de neurônios definido pelo número de classes do sistema, de modo que cada classe será associada a um neurônio de saída. De acordo com Ba e Caruana (2015), redes neurais rasas podem ter desempenho similar a redes neurais profundas mais complexas, sendo possível empregar certos algoritmos de aprendizagem para treinar melhor redes neurais rasas.

3 DESENVOLVIMENTO

Para a execução do projeto e análises das possíveis metodologias, foram desenvolvidos e analisados dois métodos diferentes: um método com o foco no uso de rede LVQ, e outro com foco na extração de características das imagens para utilização em uma rede neural rasa.

Com a visão de uso restrito à classificação de imagens assim como citado neste projeto, os dois métodos teoricamente são capazes de solucionar o problema. Porém, além de diferenças em desempenho (velocidade de processamento dos algoritmos e acurácia dos resultados), os métodos têm diferentes possibilidades e comportamentos para casos de mudanças de especificações ou refinamento e crescimento de complexidade das necessidades do projeto, como discutido em cada um dos tópicos.

Foi utilizado o *software* MATLAB para o desenvolvimento e execução dos *scripts* que realizam todas as funções de leitura, processamento de imagens, extração de características e desenvolvimento das redes neurais.

Um conjunto de 10 classes com 18 imagens cada foi utilizado no desenvolvimento deste trabalho.

3.1 Identificação com Rede LVQ

Para a classificação com rede LVQ, o *script* de MATLAB carrega o conjunto de imagens a ser utilizado em memória, descomprimidas, transformadas em uma matriz de valores entre 0 a 255 (indicando o tom de cinza que representa) e armazenadas em um vetor. As imagens foram devidamente separadas em diretórios diferentes com base em sua classe, assim, durante a leitura das imagens, também é atribuída a classe as quais elas pertencem.

Em seguida, as imagens são binarizadas utilizando o método de Otsu, que determina um valor de limiar para cada imagem para classificação dos *pixels* em preto e branco de modo que a variância seja a menor possível (OTSU 1979, p. 62-66). Os *pixels* são, então, numerados com -1 ou 1, dependendo da cor resultante da limiarização. Para a binarização, a função *imbinarize* do MATLAB foi utilizada.

As imagens resultantes, então, são ordenadas em forma de vetores unidimensionais alinhando linha por linha, e o vetor resultante utilizado como entrada para a rede LVQ. A função utilizada para criação da rede LVQ foi *lvqnet* com os parâmetros padrões do MATLAB, com exceção de parâmetros citados explicitamente neste trabalho.

3.2 Identificação Utilizando Características

A classificação de imagens utilizando características extraídas se provou um método flexível, com possibilidade de alterações drásticas em suas etapas resultando em efeitos possivelmente preferíveis. Uma das formas do processo com resultados satisfatórios foi escolhida e será apresentada nesta seção, com algumas variações pequenas apresentadas e seus resultados explicados.

Assim como no processo utilizando a rede LVQ, primeiramente, as imagens são carregadas em memória, descomprimidas, transformadas em uma matriz de valores entre 0 a 255 (indicando o tom de cinza que representa) e armazenadas em um vetor. As imagens foram devidamente separadas em diretórios diferentes com base em sua classe, assim, durante a leitura das imagens, também é recordada a classe a qual ela pertence. Em seguida, as imagens passam por alterações para facilitar a extração de suas características. Como primeiro passo, as imagens são binarizadas utilizando o método de Otsu, utilizando a função *imbinarize* do MATLAB, que determina um valor de limiar para cada imagem para classificação dos *pixels* em preto e branco de modo que a variância seja a menor possível (OTSU 1979, p. 62-66). A Figura 9 mostra o resultado deste processamento em uma das imagens utilizadas no projeto.

Figura 9 - Inversão e binarização de imagem. À esquerda, a imagem original. À direita, a imagem processada

Fonte: O próprio autor.

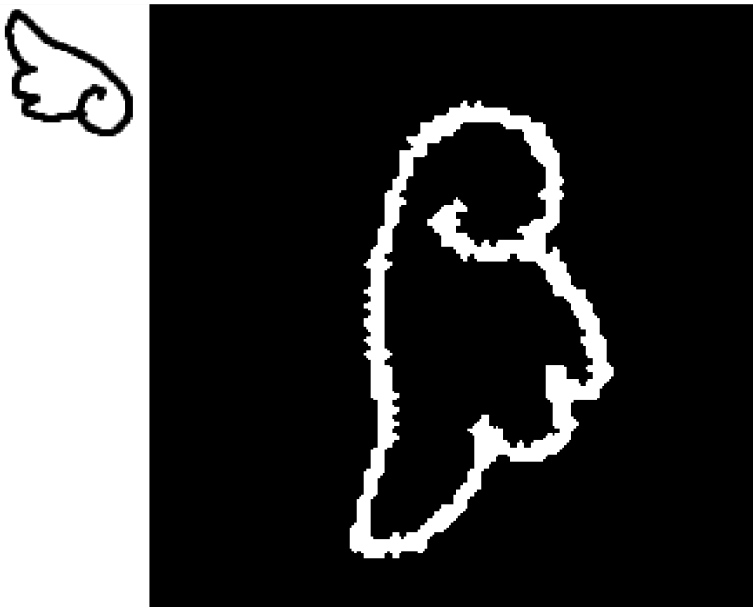
Então, as imagens podem ser rotacionadas, caso o conjunto de imagens para treinamento não possua imagens com orientações diferentes e seja desejado que as imagens possam ser identificadas independente da sua rotação.

Todas as imagens devem ter as dimensões finais iguais, independente das rotações sofridas, pois algumas das características envolvem áreas e perímetros, sendo sensíveis à mudanças de escala. Neste trabalho, foram testados casos de imagens sem rotação e redimensionamento, e casos de imagens com rotações aleatórias e aumento de escala para 4 vezes as dimensões originais (de 64x64 para 256x256).

Nos casos de processamento de imagens com rotação, para conservação das dimensões da imagem, as imagens são, primeiramente, rotacionadas, e, por consequência da rotação, redimensionada. Então, as imagens têm suas dimensões multiplicadas por 4 e divididas pelo fator de dimensionamento que foi aplicado devido à rotação. Deste modo, simula-se uma imagem redimensionada como se originalmente tivesse as mesmas dimensões que a imagem original, mas já rotacionada. Este passo é importante, pois como inicialmente não se sabe a orientação da imagem, a imagem será renderizada a partir do jogo com base em um quadrado envolvendo e encaixado no objeto desenhado.

A Figura 10 mostra o resultado do processamento da imagem com rotação e redimensionamento. Percebe-se a presença de artefatos na imagem devido à baixa resolução da imagem original e possivelmente de algoritmos não-ótimos de rotação e redimensionamento de imagem. Para a rotação e redimensionamento das imagens, foram utilizadas as funções *imrotate* e *imresize* com suas configurações padrões.

Figura 10 - Inversão, binarização, rotação e redimensionamento de imagem. À esquerda, a imagem original. À direita, a imagem processada



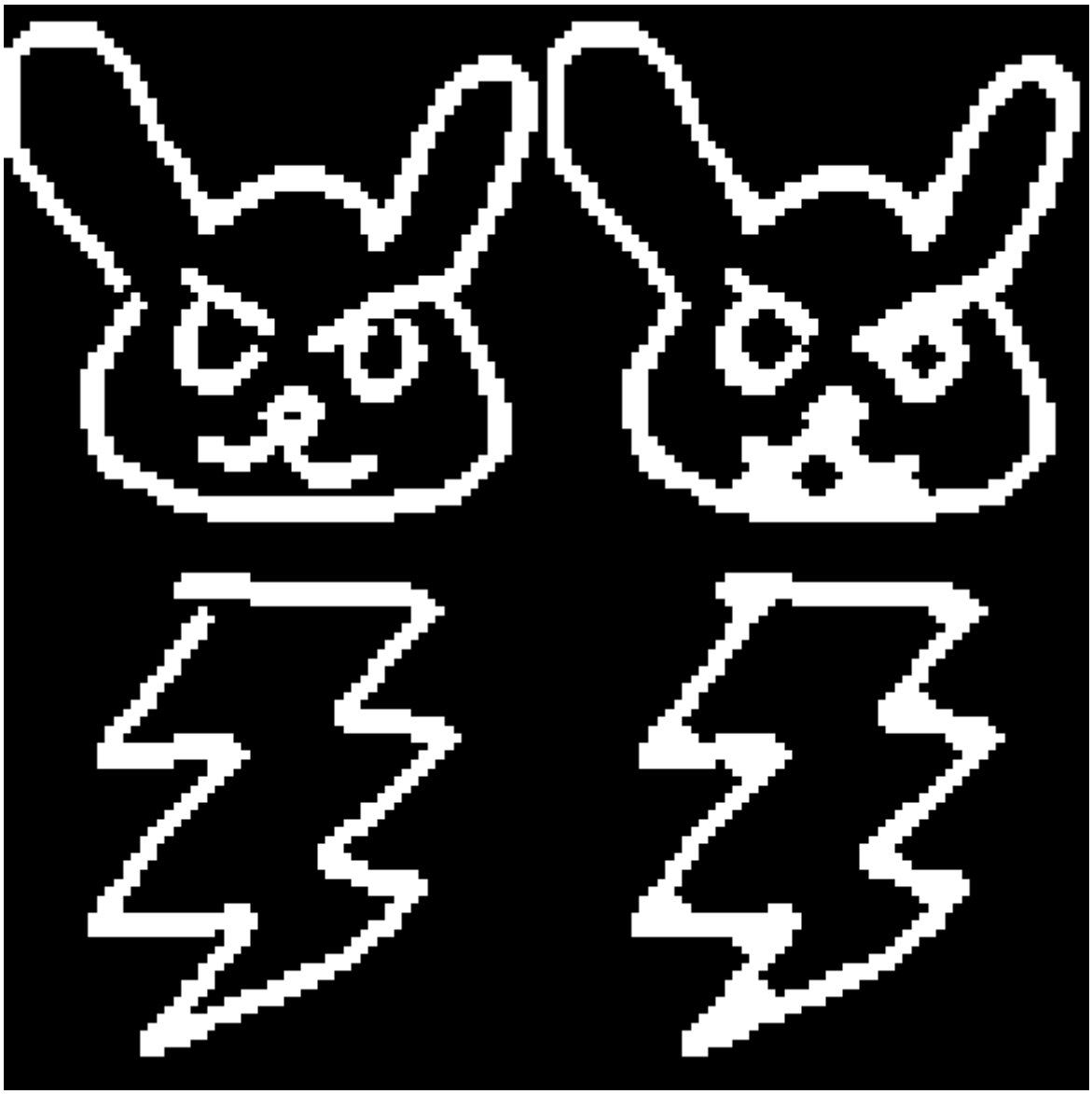
Fonte: O próprio autor.

Pode-se realizar o fechamento ou abertura da imagem, de acordo com as necessidades devidas às características da imagem renderizada. Neste projeto, foi utilizado um disco de 2x2 *pixels* como elemento estruturante para fechamento da imagem, mas não houveram diferenças notáveis na acurácia da classificação ou na velocidade de processamento independente da presença ou ausência dessa etapa. Porém, é possível que essa indiferença ocorra devido à qualidade das imagens utilizadas: as imagens possuem baixa resolução e foram feitas sem preocupação com fechamento das áreas das imagens, resultando, principalmente, em números de Euler diferentes, mesmo que seja possível exigir do desenhista que os desenhos possuam a mesma quantidade de objetos e buracos que o modelo original, e, com boa resolução das imagens, seria possível corrigir pequenos problemas de conexão de linhas com esse processo.

Na Figura 11, observa-se os efeitos do fechamento em duas imagens. Na imagem do coelho, observa-se que não só o objeto utilizado para o fechamento da imagem não foi suficiente para conectar o olho esquerdo à sobrancelha do coelho, como teve o efeito indesejado de conectar a boca com o contorno da cabeça. Por outro lado, na imagem do raio, o fechamento foi adequado,

fechando o contorno do objeto e tornando seu número de Euler de 1 para 0, aproximando essa característica do modelo original, como o desejado. Para a análise de resultados deste trabalho, não foram utilizados processos de fechamento ou abertura.

Figura 11 - Efeitos de fechamento com um disco de raio de 2 pixels nas imagens de teste. À esquerda, as imagens originais. À direita, as imagens após o fechamento



Fonte: O próprio autor.

A próxima etapa é a extração de características. Foram adquiridos o número de Euler, a área do dos objetos, a área convexa, a excentricidade, os tamanhos do maior e menor eixo e o perímetro da imagem, como discutidos na Seção 2.3, e os primeiro e segundo momentos invariantes. Os momentos do terceiro ao sétimo se mostraram variáveis demais para auxílio apropriado no reconhecimento da imagem. Devido à natureza das imagens, o valor da intensidade de cor de cada *pixel* nas imagens foi reduzido em um fator de 40 para o cálculo dos momentos da imagem,

a fim de obter valores numéricos de momentos de maior magnitude para mais fácil visualização e análise. As características foram extraídas utilizando a função *regionprops* do MATLAB, com exceção dos momentos invariantes, que foram calculados com um script próprio.

A Tabela 1 mostra, como exemplo, as características da primeira imagem das primeiras 5 classes observadas na Figura 2. A Tabela 2 e a Tabela 3 mostram as características das primeiras 5 imagens das classes 1 e 3, respectivamente, como observadas na Figura 2. Observa-se a proximidade numérica das características em imagens de mesma classe e a diferença das características entre classes diferentes.

Tabela 1 - Características de imagens de diferentes classes

	Classe 1	**Classe 2**	**Classe 3**	**Classe 4**	**Classe 5**
Número de Euler	-8	-4	0	0	1
Área	1123	782	540	545	931
Área convexa	3590	3059	2020	2052	3435
Excentricidade	0,45394	0,22071	0,60025	0,75440	0,45271
Tam. do maior eixo	77,165	64,444	63,293	75,868	75,632
Tam. do menor eixo	68,756	62,855	50,622	49,801	67,438
Perímetro	298,438	305,410	261,197	179,671	258,949
Primeiro momento	23,77398	25,89925	30,39802	37,76850	27,56588
Segundo momento	7,46122	0,41837	44,65424	225,81563	9,91099

Fonte: O próprio autor.

Tabela 2 - Características das primeiras 5 imagens de classe 1

	Imagem 1	**Imagem 2**	**Imagem 3**	**Imagem 4**	**Imagem 5**
Número de Euler	-8	-6	-5	-5	-5
Área	1123	1113	1157	988	1066
Área convexa	3590	3497	3788	3017	3480
Excentricidade	0,4539	0,5260	0,4514	0,5630	0,4765
Tam. do maior eixo	77,1652	75,9259	78,6757	71,7471	77,4216
Tam. do menor eixo	68,7566	64,5745	70,2049	59,2975	68,0683
Perímetro	298,4380	336,2580	523,1880	308,1160	325,1260
Primeiro momento	23,7740	22,3089	24,0189	21,9159	24,9173
Segundo momento	7,4612	12,8334	7,4258	17,0418	10,1850

Fonte: O próprio autor.

Tabela 3 - Características das primeiras 5 imagens de classe 3

	Imagem 1	**Imagem 2**	**Imagem 3**	**Imagem 4**	**Imagem 5**
Número de Euler	0	0	0	0	0
Área	540	491	500	510	444
Área convexa	2020	1677	1748	1859	1789
Excentricidade	0,6003	0,5403	0,6441	0,6831	0,5550
Tam. do maior eixo	63,2931	56,0530	60,4177	64,2669	60,2685
Tam. do menor eixo	50,6224	47,1676	46,2161	46,9331	50,1333
Perímetro	261,1970	232,5160	244,8220	246,1450	222,1200
Primeiro momento	30,3980	27,3118	28,9178	31,0308	34,5888
Segundo momento	44,6542	21,8073	57,3330	89,2761	39,6947

Fonte: O próprio autor.

Uma rede neural perceptron multicamadas, então, é criada, configurada e treinada. A ferramenta utilizada para a criação da rede neural rasa no MATLAB foi *fitnet* com os parâmetros padrões, com exceção dos parâmetros explicitados neste trabalho. Na camada de entrada, cada neurônio recebe o valor de uma das características a ser avaliada, e, na camada de saída, cada neurônio corresponde a uma das classes de imagem, de modo que o neurônio correspondente à classe da imagem utilizada como entrada deve ter seu valor próximo de 1 e os demais neurônios da camada de saída devem ter seus valores próximos de 0.

Apesar de ser difícil de quantificar e avaliar características exatas para a otimização da rede neural no processo, uma rede com 30 neurônios na camada oculta, uma taxa de aprendizado de 0,5 e limite de 120 épocas mostrou os melhores resultados. Taxas de aprendizado menores, assim como mais de 30 neurônios na camada oculta, tornavam o processo mais lento sem melhorar o resultado, e taxas de aprendizado maiores e quantidade menor de neurônios faziam com que a rede não conseguisse alcançar resultados tão bons.

O limite de 120 épocas foi imposto pois, passado essa quantidade, o desempenho não aumentava ou aumentava pouco. Foi utilizada a regularização bayesiana como algoritmo de treinamento, pois apresentou acurácia e velocidade de treinamento melhores que o algoritmo de Levenberg-Marquardt. Das imagens separadas para o aprendizado da rede neural, 80% foi utilizado para o treinamento e 20% foi utilizado para validação.

Por fim, a rede foi utilizada para classificar as imagens utilizadas para o treinamento da rede neural e imagens que a rede neural nunca tinha utilizado previamente, e classificações incorretas eram registradas no console do programa.

4 RESULTADOS

A rede LVQ não se mostrou adequada para a classificação das imagens, visto que sua aprendizagem se torna extremamente lenta e incapaz de alcançar níveis significantes de acurácia para o conjunto de imagens utilizado. Utilizando 30 neurônios na camada oculta, com 1 imagem de cada classe para treinamento, a rede neural levou 138 iterações em 8 segundos para ser capaz de corretamente classificar seu conjunto de treinamento, com acurácia de 60,63% para o conjunto de testes.

Com 2 imagens de cada classe para treinamento, a rede neural alcançou uma acurácia de 79,33% em 300 iterações. Porém, a rede neural alcançou seu estado final em 80 iterações em 35 segundos: não houve melhora de desempenho nas últimas 220 iterações.

Com 3 imagens de cada classe para treinamento, a rede neural alcançou acurácia de 77,33% em 300 iterações, sendo as últimas 100 iterações sem melhorias em desempenho. Redução no número de neurônios torna as iterações mais rápidas, mas reduz a capacidade de melhoria de desempenho por iteração: com 2 imagens de cada classe, foram 300 iterações em 16 segundos para 55,33% de acurácia em uma rede com 10 neurônios, e, com 20 neurônios, em 300 iterações em 26 segundos, uma acurácia de 79,33% foi alcançada.

Por outro lado, um aumento no número de neurônios aumenta o tempo das iterações, mas não melhora a acurácia da rede: com 40 neurônios, em 300 iterações em 1 minuto e 8 segundos, a rede apresentou acurácia de 76,67%. Em todos esses casos, a rede se estabiliza (deixa de apresentar mudanças no processo de treinamento) muito antes de completar as 300 iterações, indicando que não haverá mais progresso ou haverá pouco progresso no treinamento da rede. Mudanças de outros parâmetros, como a taxa de aprendizagem, mostraram resultados similares: não houve alteração ou diminuição da acurácia do sistema.

A classificação utilizando características obteve resultados tidos como satisfatórios para os fins desejados. Para imagens não rotacionadas, foi obtido um resultado consistente com múltiplas instâncias de treinamento de redes neurais de 100% de acurácia utilizando 150 imagens para treinamento e validação e 30 imagens para teste, sempre utilizando a mesma quantidade de imagens de cada classe para treinamento e teste. Com 100 imagens para treinamento e validação

e 80 para teste, a rede neural alcança acurácia de 100% na classificação das imagens de treinamento, mas apenas de aproximadamente 90% (em 8 experimentos, entre 2 a 4 erros de classificação) nas imagens de teste, mostrando que a quantidade de 15 imagens de cada classe para treinamento é o suficiente para um desempenho bom. Na Tabela 4, observa-se o resultado da rede neural para a 16ª imagem de cada classe após treiná-la com as 15 primeiras imagens de cada classe. Observa-se que, para todas as imagens, a saída correspondente é muito próxima de 1, enquanto as outras saídas são muito próximas de 0, indicando que houve pouca confusão na classificação, ou seja, a rede neural classificou as imagens corretamente e com facilidade.

Tabela 4 - Saída da rede neural treinada quando utilizadas imagens de teste nunca antes expostas à rede. Quanto mais próxima de 1 é a saída da classe, mais semelhante a rede considerou a imagem à classe

		Classe das imagens utilizadas									
		1	2	3	4	5	6	7	8	9	10
Classificação pela rede neural	1	1,000	0,000	0,000	0,000	0,000	0,000	-0,001	0,000	0,000	0,000
	2	0,000	1,002	0,000	0,000	-0,001	0,000	0,000	0,000	0,000	0,000
	3	0,012	-0,003	0,999	0,042	0,015	-0,047	0,036	0,000	0,002	-0,016
	4	-0,010	0,001	0,000	0,965	-0,008	0,036	-0,051	-0,002	-0,012	0,020
	5	-0,001	0,000	0,001	0,000	1,001	-0,001	0,000	0,000	-0,001	0,000
	6	-0,002	0,003	0,001	-0,010	-0,006	1,012	0,014	-0,001	0,010	-0,001
	7	0,000	-0,001	0,002	0,000	-0,052	-0,001	1,005	0,000	0,001	-0,001
	8	-0,003	0,000	0,000	0,004	0,057	-0,005	-0,004	1,002	-0,001	-0,002
	9	0,003	-0,001	-0,001	0,001	-0,006	0,000	0,000	0,000	1,000	0,001
	10	0,000	0,000	-0,002	-0,001	0,000	0,005	0,000	0,000	0,000	0,999

Fonte: O próprio autor.

Para imagens com rotação, a acurácia da identificação foi menor. Utilizando 120 imagens para treinamento e validação e 60 para testes, a rede neural alcança acurácia de 100% na classificação das imagens de treinamento, e aproximadamente 80% (em 8 experimentos, entre 5 a 9 erros de classificação) nas imagens de teste.

Com 150 imagens de treinamento e validação e 30 de testes, a rede neural apresentou acurácia de 100% nas imagens de treinamento e 93,2% nas imagens de teste (em 24 experimentos, geralmente entre 1 a 3 erros, alcançando até 0 ou 4 erros).

Com 170 imagens de treinamento e 10 de testes, a rede neural apresentou acurácia de aproximadamente 96,2% nas imagens de teste (em 20 experimentos, com nenhum ou um erro por experimento). Com a diferença de acurácia entre 120 e 150 imagens para treinamento, os

resultados sugerem que o projeto pode ter acurácia incrementada com mais imagens para treinamento. Como dito na seção 3.2, é possível que a baixa acurácia com imagens rotacionadas ocorra devido à resolução das imagens utilizadas, resultando em grandes variações de características que talvez não ocorreriam em tamanha magnitude com imagens de maior resolução.

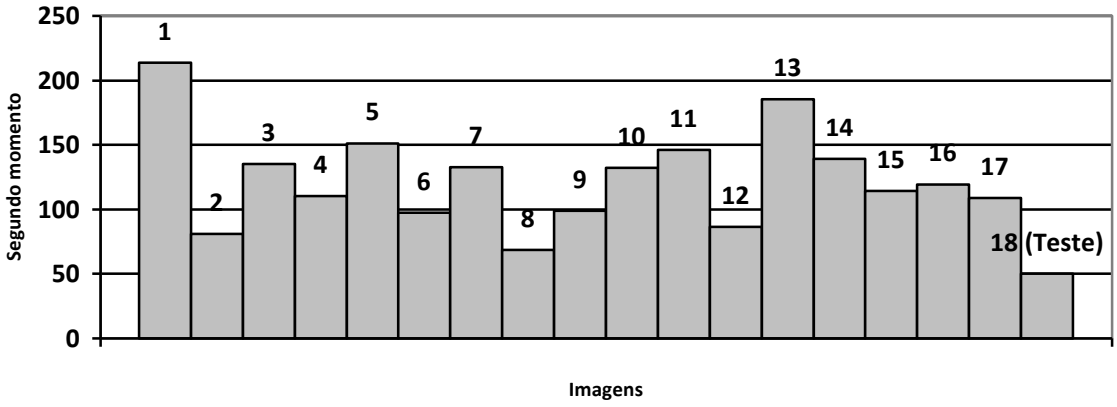
Entretanto, observa-se que nos casos com maior quantidade de imagens de treinamento, os erros ocorrem quando alguma característica extraída da imagem é discrepante do resto do conjunto utilizado para treinamento. Por exemplo, em uma instância de experimento com 170 imagens de treinamento, um erro foi detectado: a 18ª imagem da classe 8 foi classificada como classe 3. A Tabela 5 mostra características das 4 primeiras imagens da classe 8 usadas para treinamento da rede neural, assim como as características da imagem para teste. O Gráfico 1 mostra o segundo momento da imagem de teste e o segundo momento das imagens de treinamento. Como pode-se observar, o segundo momento da imagem de teste é significantemente menor do que do resto do conjunto de imagens. Ao verificar a resposta da rede neural às características dessa imagem, porém, substituindo o valor do segundo momento por 100, a rede neural classificou a imagem corretamente, como mostra a Tabela 6, comprovando a anomalia da imagem em relação ao resto do conjunto. Isso poderia ter sido evitado se essa imagem fosse utilizada no treinamento, indicando que os erros que ocorrem podem ser evitados ou reduzidos com a utilização de um maior conjunto de imagens de treinamento na rede neural.

Tabela 5 - Comparação das características da classe 8 entre a imagem de teste (18ª imagem) e as quatro primeiras imagens de treinamento

	Imagem de Teste	**Imagem 2**	**Imagem 3**	**Imagem 4**	**Imagem 5**
Número de Euler	0	0	-2	0	0
Área	6468	5541	5147	6540	7833
Área convexa	21029	19246	15461	23726	25284
Excentricidade	0,6543	0,8002	0,7224	0,7247	0,7333
Tam. do maior eixo	207,1624	224,9672	188,5346	240,5766	247,5204
Tam. do menor eixo	156,6718	134,9080	130,3691	165,7808	168,2787
Perímetro	884,4230	806,7080	774,4050	874,2910	900,3760
Primeiro momento	26,0744	31,0448	25,5191	32,6291	28,5910
Segundo momento	50,4159	213,8276	81,1751	134,9877	110,5855

Fonte: O próprio autor.

Gráfico 1 - Segundos momentos do conjunto de imagens da classe 8



Fonte: O próprio autor.

Tabela 6 - Classificação da imagem baseada nas suas características originais e nas suas características com segundo momento modificado

		Características utilizadas	
		Originais (segundo momento = 50,4159)	Modificadas (segundo momento = 100,0000)
Classificação pela rede neural	1	0,0001	0,0002
	2	-0,0002	-0,0002
	3	0,5503	0,0165
	4	0,0001	0,0002
	5	0,0000	0,0000
	6	0,0000	0,0000
	7	0,0471	0,0456
	8	0,4026	0,9378
	9	0,0000	0,0001
	10	0,0000	0,0000

Fonte: O próprio autor.

Assim como esperado, observou-se que todas as características utilizadas neste projeto são invariantes à rotação, apresentando diferenças pequenas nos valores de suas características quando rotacionadas, principalmente em comparação a diferenças de valores entre imagens diferentes da mesma classe e de classes diferentes. A Tabela 7 e a Tabela 8 mostram valores das características da imagem 1 da classe 5 e da imagem 14 da classe 6 quando rotacionadas em 0º, 20º, 90º, 180º e 250º. A variação em algumas propriedades como área e área convexa para certas rotações se dá pelo tamanho do objeto para ser encaixado na imagem. Por exemplo, um quadrado cabe perfeitamente em uma área quadrada, mas ao ser rotacionado em 45º, passa a ocupar uma área menor em proporção à area da imagem, se as pontas do quadrado ainda tocarem as bordas da imagem. Além disso, pode-se atribuir parte das diferenças à distorção devido à rotação e redimensionamento da imagem, o que não ocorreria com uma imagem que não foi rotacionada, mas sim desenhada com orientação diferente de 0º.

Tabela 7 - Características da imagem 1 da classe 5 após sofrer rotações de diferentes ângulos

	0º	**20º**	**90º**	**180º**	**250º**
Número de Euler	1	2	1	1	1
Área	14896	8877	14896	14896	8786
Área convexa	55058	33159	55065	55050	32970
Excentricidade	0,4530	0,4543	0,4530	0,4530	0,4479
Tam. do maior eixo	302,6206	232,9971	302,6206	302,6206	232,0543
Tam. do menor eixo	269,7909	207,5647	269,7909	269,7909	207,4784
Perímetro	1,112,0800	1,166,2410	1,112,0800	1,112,0800	1,813,6740
Primeiro momento	27,5852	27,4215	27,5852	27,5852	27,5705
Segundo momento	9,9470	9,9572	9,9470	9,9470	9,4472

Fonte: O próprio autor.

Tabela 8 - Características da imagem 14 da classe 6 após sofrer rotações de diferentes ângulos

	0º	**20º**	**90º**	**180º**	**250º**
Número de Euler	-1	-1	-1	-1	-1
Área	9112	5312	9112	9112	5427
Área convexa	34033	20377	34046	34036	20357
Excentricidade	0,9235	0,9227	0,9235	0,9235	0,9231
Tam. do maior eixo	372,7709	288,0151	372,7709	372,7709	285,3424
Tam. do menor eixo	143,0191	111,0022	143,0191	143,0191	109,7395
Perímetro	970,8780	794,7480	970,8780	970,8780	785,1070
Primeiro momento	43,7363	44,8379	43,7363	43,7363	43,0534
Segundo momento	1,057,1005	1,104,9870	1,057,1005	1,057,1005	1,021,4058

Fonte: O próprio autor.

Em termos de velocidade de processamento, o código feito no MATLAB, com a imagem já carregada na memória do programa, utilizando imagens 64x64 sem rotação, é capaz de realizar o processamento da imagem (binarizar com limiarização e realizar o fechamento da imagem), extrair as características, utilizar a rede neural e processar sua resposta em aproximadamente 76 milissegundos, com os tempos aproximados divididos de acordo com a Tabela 9.

Para imagens 256x256, o código executa em aproximadamente 0,185 segundos, divididos aproximadamente como indica a Tabela 10. Os tempos relativos ao rotacionamento e redimensionamento das imagens são irrelevantes, pois as imagens a serem classificadas já

devem estar previamente rotacionadas: o rotacionamento é feito neste experimento para simular conjuntos de imagens rotacionados. Similarmente, os tempos de conversão da imagem para tons de cinza e complemento de cores da imagem não são necessários, pois é uma questão da renderização da imagem: basta utilizar um método de renderização diretamente para tons de cinza e cores de fundo e de objetos corretos na prática; a conversão é utilizada neste experimento pois o conjunto de imagens é armazenado em PNG com cores de fundo e objetos invertidos aos considerados para extração de características do MATLAB.

O tempo para preparação e treinamento da rede neural é irrelevante para o escopo do trabalho, pois não é necessário que se crie a rede neural em tempo real durante o jogo, apenas que se classifique imagens – a criação e treinamento da rede neural pode ser feita antes da distribuição do jogo. O computador utilizado é um Intel Core i7-2600 de 3,40GHz, 12GB de memória RAM com Windows 10 x64.

Tabela 9 - Tempo de processamento em várias iterações para diferentes etapas do experimento na imagem 1 da classe 1 no processo sem rotação e redimensionamento de imagem

	Tempo de processamento (ms)					
	1	**2**	**3**	**4**	**5**	**Média**
Binarização	0,535	0,482	0,524	0,481	0,467	0,498
Fechamento	3,003	2,304	2,912	2,141	2,907	2,653
Número de Euler	4,722	5,088	4,756	5,162	5,822	5,110
Área	4,010	4,191	4,566	5,261	5,096	4,625
Área convexa	6,536	8,511	6,201	7,567	7,727	7,308
Excentricidade	4,088	5,133	4,264	7,220	4,868	5,115
Tam. do maior eixo	4,167	4,377	4,479	6,701	8,446	5,634
Tam. do menor eixo	4,737	4,032	4,426	4,985	5,023	4,641
Perímetro	5,015	3,979	4,365	4,319	4,029	4,341
Momentos	13,339	12,973	12,561	12,672	13,156	12,940
Classificação	21,611	22,864	24,346	22,394	23,130	22,869
Total	71,763	73,934	73,400	78,903	80,671	75,734

Fonte: O próprio autor.

Tabela 10 - Tempo de processamento em várias iterações para diferentes etapas do experimento na imagem 1 da classe 1 no processo com rotação e redimensionamento de imagem

	Tempo de processamento (ms)					
	1	**2**	**3**	**4**	**5**	**Média**
Binarização	0,526	0,471	0,494	0,459	0,512	0,492
Fechamento	3,426	3,281	3,208	4,352	3,231	3,500
Número de Euler	7,367	6,424	7,764	6,772	7,200	7,105
Área	4,264	5,161	6,255	5,623	4,991	5,259
Área convexa	8,893	9,758	9,950	8,567	9,737	9,381
Excentricidade	5,754	7,355	6,453	5,588	5,890	6,208
Tam. do maior eixo	6,307	5,931	7,072	5,222	5,968	6,100
Tam. do menor eixo	4,961	11,118	9,859	6,498	5,281	7,543
Perímetro	5,330	4,731	4,977	5,110	6,403	5,310
Momentos	162,129	176,131	163,022	158,452	160,423	164,031
Classificação	25,608	22,746	24,763	23,964	23,241	24,064
Total	234,565	253,107	243,817	230,607	232,877	238,995

Fonte: O próprio autor.

5 CONCLUSÃO

A rede LVQ se mostrou inadequada para os fins desse projeto, tendo um treinamento lento e com desempenho extremamente limitado, inviabilizando o seu uso para classificação de imagens com os requisitos e características apresentados.

Utilizando características das imagens para classificação, o projeto de classificação alcança até 100% de acurácia para os conjuntos de teste utilizados, os resultados foram satisfatórios, provando a possibilidade da implementação do sistema para uso em um jogo em tempo real.

Em termos de velocidade de processamento, a rede neural obteve resultados satisfatórios para fins de implementação em jogos de tempo real. Apesar de jogos atuais executarem normalmente entre 60 a 144 *fps*, não é necessário que o jogo seja capaz de processar e classificar a imagem em todos os *frames* o tempo todo. Um pequeno *delay* na ordem de milissegundos ainda é aceitável, principalmente considerando que possivelmente a imagem não estará em constante mudança ou que não há necessidade de classificar uma imagem antes de ela estar terminada. Além disso, utilizando um código compilado e otimizado para fins desse projeto com uso de linguagens como C ou *Assembly*, é possível reduzir esses tempos de processamento. Por fim, outras alternativas ainda podem ser utilizadas para otimização do processo, como a remoção ou substituição dos momentos das imagens no processo, já que contribuem significantemente para o tempo total de processamento; a utilização de paralelismo, fazendo com que a extração de várias características ou o cálculo de valores dos momentos e ativações dos neurônios sejam realizados ao mesmo tempo utilizando processadores multinúcleos; ou a utilização de placas de vídeo para o processamento das imagens e cálculo da rede neural, visto que placas de vídeo são otimizadas para cálculos com matrizes.

Por ser flexível, o método também possui possibilidades para implementações de diversas maneiras para objetivos e requisitos diferentes. Por exemplo, é possível implementar um sistema com uma rede neural para cada conjunto de classes com o mesmo número de Euler. A imagem a ser classificada, então, só é avaliada pela rede neural correspondente ao número de Euler calculado da imagem. Com esse método, impõem-se regras para os desenhos, como a obrigação de que pontos de encontro de traços na imagem modelo devem estar claros o suficiente no desenho a ser avaliado, reforçando certas similaridades entre os desenhos e a

imagem modelo e possivelmente aumentando a acurácia do sistema, mas criando a necessidade de um melhor tratamento morfológico da imagem para evitar erros por pequenas imprecisões no desenho. Alternativamente, a partir da detecção de regiões, poderia ser implementado um sistema para separação de objetos da imagem, classificando os objetos individualmente para maior acurácia ou resultados diferentes para combinações diversas de objetos sem a necessidade de a rede neural ser treinada com todas as combinações possíveis de objetos.

Essas possibilidades demonstram a versatilidade do método, assim como a necessidade de adaptação para os casos específicos nos quais o sistema será implementado, como o correto processamento de imagens para o método e características da renderização a serem utilizadas.

REFERÊNCIAS

ABDULLAH, Siti Norul Huda Sheikh; PIRAHANSIAH, Farshid; SAHRAN, Shahnorbanun. **Adaptative Image Thresholding Based on the Peak Signal-to-noise Ratio**. Reasearch Journal of Applied Sciences, Engineering and Technology, v. 8, n. 9, 2014, p. 1104-1116. Disponível em: <http://maxwellsci.com/msproof.php?doi=rjaset.8.1074>. Acesso em: 19 jun. 2018.

BA, Lei Jimmy; CARUANA, Rich. **Do Deep Nets Really Need to be Deep?**. 2013. Disponível em <http://datascienceassn.org/sites/default/files/Do%20Deep%20Nets%20Really%20Need%20to%20be%20Deep.pdf>. Acesso em 16 jun. 2018.

BOCK, Yannick De. **An example of the hit-or-miss transform**. 2010. Disponível em <https://commons.wikimedia.org/wiki/File:Hitormisstransformatie.jpg>. Acesso em 04 dez. 2017.

BORACCHI, Giacomo. **Edge Detection**. Disponível em: <http://home.deib.polimi.it/boracchi/teaching/IAS/EdgeDetection/EdgeDetection.html>. Acesso em: 04 dez. 2017.

GIROD, Bernd. **Digital Image Processing**: Morphological Image Porcessing. 2013. Stanford University, Stanford, 2013. Disponível em: <https://web.stanford.edu/class/ee368/Handouts/Lectures/2014_Spring/Combined_Slides/7-Morphological-Image-Processing-Combined.pdf>. Acesso em: 04 dez. 2017.

GONZALEZ, R. C.; WOODS, R. E. **Processamento de Imagens Digitais**. São Paulo: Edgard Blücher LTDA. 2000.

KRIESEL, David. **A Brief Introduction to Neural Networks**. [S.l.: S.n.]. Disponível em: <http://www.dkriesel.com/_media/science/neuronalenetze-en-zeta2-2col-dkrieselcom.pdf>. Acesso em: 04 dez. 2017.

LEUNG, K. Ming. **Learning vector Quantization**. Department of Computer and Information Science, Florida, 2013. Disponível em: <http://cis.poly.edu/~mleung/CS6673/s08/LVQ.pdf>. Acesso em: 13 jun. 2018.

MIURA, Keiji; NAKADA, Kazuki. Neural Implementation of Shape-Invariant Touch Counter Based on Euler Calculus. **IEEE Access**, jan. 2014.

NIELSEN, Michael A. **Neural networks and Deep Learning**. [S.l.]: Determination Press. 2015. Disponível em: <http://neuralnetworksanddeeplearning.com/chap1.html>. Acesso em: 04 dez. 2017.

OTSU, N. **A Threshold Selection Method from Gray-Level Histograms**. IEEE Transactions on Systems, Man, and Cybernetics, v. 9, n. 1, 1979, p. 62-66. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4310076>. Acesso em: 13 jun. 2018.

THESSALONIKI. **Digital Image Processing**. cap. 7. Disponível em: <http://wcours.gel.ulaval.ca/2013/a/GIF7002/default/5notes/diapositives/pdf_A13/lectures%20supplementaires/C09b.pdf>. Acesso em: 04 dez. 2017.

RUSSEL, Ingrid. **The Delta Rule**. Disponível em: <http://uhavax.hartford.edu/compsci/neural-networks-delta-rule.html>. Acesso em: 04 dez. 2017.

SMITH, Steven W. **The Scientist and Engineer's Guide to Digital Signal Processing**. 1. ed. [S.l.]: California Technical Pub. 1997. Disponível em: <http://www.dspguide.com/ch25/4.htm>. Acesso em: 04 dez. 2017.

YOGAMANGALAM, R.; KARTHIKEYAN, B. Segmentation Techniques Comparison in Image Processing. **International Journal of Engineering and Technology (IJET)**, v. 5, n. 1, p. 307-313, feb./mar. 2013.

MEASURE properties of image regions - MATLAB regionprops. In: MathWorks - Makers of MATLAB and Simulink - MATLAB & Simulink. Disponível em <https://www.mathworks.com/help/images/ref/regionprops.html>. Acesso em: 13 jun. 2018.

LINE Detection by Hough transformation. Disponível em <http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/HoughTrans_lines_09.pdf>. 2009. Acesso em: 04 dez. 2017.