UNIVERSIDADE FEDERAL DO PARANÁ

ARYEL MARLUS REPULA DE OLIVEIRA
CARLA MARIA DA SILVA CASSEMIRO
MAURO DA SILVA PINTO

P-FINDER

CURITIBA
2008

SUMÁRIO

LISTA DE ILUSTRAÇÕES

RESUMO

O projeto P-Finder é uma ferramenta de busca baseada em Algoritmo Genético que tem como objetivo encontrar uma placa automotiva em uma imagem, visando, assim, auxiliar, para os diversos fins, no reconhecimento dos caracteres da placa que poderá ser feito através de um software de OCR. Trabalhar com uma tecnologia que usa IA é algo complexo e está bastante suscetível a falhas, além de possuir inúmeros modos de chegar ao objetivo. O Algoritmo Genético não garante o melhor resultado, mas oferece uma solução aceitável. A modelagem estratégica referente ao mecanismo de diferenciação da placa e do meio que ela se encontrava aliado a um esquema de filtragem das imagens que pudesse auxiliar destacando as características importantes apresentou-se bastante complexo e sensível a diversos fatores como luminosidade, tamanho, posição, inclinação, cores e ruídos contidos na imagem, mas chegou-se a um resultado satisfatório após diversas tentativas. Apesar de fazer a localização da posição da placa na imagem, o P-Finder não faz a identificação dos caracteres. Esta identificação pode ser feita através de um OCR.

Palavras-chave: Algoritmo Genético. Software. Reconhecimento de placa. Placa automotiva.

ABSTRACT

The project P-Finder is a search engine based on Genetic Algorithm that aims to find an automobile plate in an image, helping, in various purposes, in recognition of the characters of the plate that could be done by an OCR software. Work with a technology that uses AI is something complex and is very susceptible to breakdowns, and there are many ways to reach the goal. The Genetic Algorithm does not guarantee to reach the best result, but it can offer na acceptable solution. The strategic modeling regarding the mechanism of plate and ambient differentiation combined with an image filter that could help highlight the important characteristics became very complex and sensitive to various factors like brightness, size, position, tilt , Color and noise in the picture, but it became a satisfactory result after several attempts. Despite finding the location of the position of the plate in the picture, the P-Finder does not identify the characters. This identification can be made by an OCR.

Keywords: Genetic Algorithm. Software. Plate recognition. Automotive plate.

1 INTRODUÇÃO

A IA (Inteligência Artificial) vem sendo cada vez mais utilizada no mercado como uma ferramenta para automação de algumas atividades que eram realizadas por pessoas. Estas automações são importantes, pois contribuem para facilitar algumas tarefas, contribuindo assim para o avanço tecnológico.

Trata-se aqui um sistema feito em plataforma Java que é uma ferramenta de busca em imagem que utiliza uma técnica de IA, o Algoritmo Genético.

Um computador não consegue abstrair informações de uma imagem como os seres humanos fazem, pois para ele, uma imagem nada mais é do que vários pixels em conjunto, não conseguindo reconhecer elementos dentro da figura, como objetos ou sinais. Já os seres humanos, o fazem naturalmente ao observar uma foto ou assistir um vídeo. Por isso se faz necessário um software que faça este processo, para que consiga chegar ao reconhecimento dos caracteres da placa.

O objetivo geral deste projeto é encontrar uma solução para a localização de placas de automóveis para o reconhecimento através de software OCR externo.

Os objetivos específicos são:

a) Entender e desenvolver um algoritmo genético funcional;
b) Adaptar a função de fitness para encontrar uma placa de automóvel em uma imagem;
c) Estudar softwares OCR que possam ser recomendados para o reconhecimento dos caracteres da placa na imagem obtida pelo P-Finder.

Esta aplicação poderá ser utilizada como parte integrante, por exemplo, dos seguintes sistemas:

a) Radares
b) Lombadas eletrônicas
c) Controles de estacionamento
d) Controles de entrada e saída de veículos de quaisquer lugares.

Neste trabalho serão apresentadas as tecnologias utilizadas em nosso projeto, bem como o histórico do desenvolvimento desta aplicação e análise dos resultados obtidos com este trabalho, apresentando sugestões para projetos futuros.

2 FUNDAMENTOS TEÓRICOS

2.1 INTELIGÊNCIA ARTIFICIAL

A palavra “inteligência” vem do latim “inter” (entre) e “legere” (escolher), portanto pode ser entendida, literalmente, como a capacidade de se realizar de forma eficiente uma escolha. A palavra “artificial” vem do latim “artificiale”, significa algo não natural, isto é, produzida pelo homem. Sendo assim, Inteligência Artificial (IA) é uma inteligência produzida pelo homem, a criação de algo que seja capaz de realizar escolhas, ou que realize outros aspectos relacionados a ela.

Porém inteligência não se resume apenas à capacidade de escolha, é muito mais complexa, e estudar como ela funciona não é uma tarefa fácil, e para se reproduzir aspectos da inteligência humana em sistemas computacionais é necessário entender como ela atua e definir como pode ser modelada de maneira que possa ser implementada. E esse é o objetivo básico da IA, estudar e modelar a inteligência, tratando-a como um fenômeno. Apesar de muitas conclusões relevantes tenham sido feitas sobre essa área, ainda não existe uma teoria completa sobre a mente humana e os processos de raciocínio (FERNANDES, 2005, p. 2).

A análise de aspectos da inteligência humana iniciou-se com filósofos e cientistas, e posteriormente passou a ser estudada de forma científica por outros campos do saber humano, tais como engenharia, psicologia, pedagogia, ciência cognitiva, neurologia, lingüística, computação entre outros, com objetivos práticos e comerciais (FERNANDES, 2005, p. 1).

Fernandes (2005, p. 3) ainda afirma que em IA, existe basicamente duas abordagens, com base nos diversos campos de estudo:

a) Abordagem cognitiva: também denominada de “descendente” ou “simbolista”, dá ênfase aos processos cognitivos, ou seja, a forma como o ser humano raciocina. Objetiva encontrar uma explicação para comportamentos inteligentes baseados em aspectos psicológicos e processos algorítmicos. Os pioneiros desta corrente foram John McCarthy, Marvin Minsky, Newell e Simon.
b) Abordagem conexionista: também denominada de “biológica” ou “ascendente”, dá ênfase no modelo de

> funcionamento do cérebro, dos neurônios e das conexões neurais. Os pioneiros desta corrente foram McCulloch, Pitts, Heldo, Rosenblatt e Widrow. Em 1943 surgiu a representação e formalização matemática dos neurônios artificiais, que fez surgir os primeiros modelos de redes neurais artificiais. A corrente conexionista sofreu grande impacto quando os cientistas Marvin Minsky e Seymour Papert publicaram, em 1969, o livro Perceptrons, no qual criticavam e sustentavam que os modelos das redes neurais não tinham sustentação matemática suficiente que lhes fosse possível atribuir alguma confiabilidade. Apesar de as pesquisas nesta área não terem parado, foi apenas na década de 1980 que o físico e biólogo do Instituto de Tecnologia da Califórnia, John Hopfield conseguiu recuperar a credibilidade da utilização de redes neurais.

Segundo Fernandes (2005, p. 3), os principais modelos de IA são: Algoritmos Genéticos, Programação Evolutiva, Lógica Fuzzy, Sistemas Baseados em Conhecimento, Raciocínio Baseado em Casos, Programação Genética e Redes Neurais.

Um resumo sobre alguns dos principais modelos de IA são citados por Fernandes (2005, p. 4):

> a) **Algoritmo Genético:** é um modelo para aprendizado de máquina, inspirado no livro Origem das Espécies. Através da seleção natural, escrito pelo naturalista inglês Charles Darwin (1809-1882), criador da teoria evolucionista, segundo a qual somente os mais aptos sobrevivem. É um método utilizado pelos Algoritmos Evolutivos, que inclui o estudo dos algoritmos genéticos, estratégia de evolução, programação evolutiva e sistemas classificatórios. Os algoritmos genéticos foram criados por Holland (1975) e objetivam emular operadores genéticos (específicos, como cruzamento, mutação e reprodução) da mesma forma como são observados na natureza. Isto é feito criando-se dentro da máquina uma população de indivíduos representados por cromossomos. Os indivíduos passam por um processo simulado de evolução, seleção e reprodução, gerando uma nova população.
>
> b) **Programação Evolutiva:** campo da AI concebido por Fogel (1960), assemelha-se aos algoritmos genéticos, sendo que dá maior ênfase na relação comportamental entre os parentes e seus descendentes. As soluções para os problemas são obtidas por meio de tentativas e transmitidas para a nossa população (simulada em programas).
>
> c) **Lógica Fuzzy:** também denominada “lógica difusa” ou “lógica nebulosa”. Foi estruturada por Lofti Zadeh, na Universidade da

Califórnia, no ano de 1965. É uma metodologia que serve para representar, manipular e modelar informações incertas.

d) **Sistemas Baseados em Regras**: são sistemas que implementam comportamentos inteligentes de especialistas humanos.

e) **Programação Genética:** é um campo de estudo da IA voltado para a contrução de programas que visam imitar o processo natural da genética. Trabalha com métodos de busca aleatória.

f) **Raciocínio Baseado em Casos**: é o campo de estudo da IA que utiliza uma grande biblioteca de casos para consulta e resolução de problemas. Os problemas atuais são resolvidos através da recuperação e consulta de casos já solucionados e da conseqüente adaptação das soluções encontradas.

g) **Redes Neurais:** possui várias denominações, dentre elas: Redes Neuronais, Modelo Conexionista, Neurocomputação, Modelo de Processamento Paralelo Distribuído, Sistemas Neuromórficos e Computadores Biológicos. São consideradas uma classe de modelagem de prognóstico que trabalha por ajuste repetido de parâmetro. Estruturalmente, uma Rede Neural consiste em um número de elementos interconectados (chamados “neurônios”) organizados em camadas que aprendem pela modificação da conexão firmemente conectando as camadas.

Muitos problemas não são possíveis de se resolver utilizando programação convencional de computadores, sendo somente capaz de se chegar à uma solução aceitável aplicando técnicas de IA. Entre eles, está a busca por objetos reais específicos em imagens digitalizadas, no sistema P-Finder, por exemplo, é realizada busca por placas automotivas.

2.2 ALGORITMOS GENÉTICOS

Existem muitos problemas específicos que são um desafio, por exemplo, uma situação onde haja muitas possibilidades de respostas para uma solução ou ainda muitas variáveis a serem consideradas, testar “manualmente” todas para tentar chegar à uma resposta iria resultar em milhares ou até dezenas de milhares de iterações computacionais (SCHWAB, 2004, p. 413). Para resolver esse tipo de situação, podemos utilizar a solução em IA chamada Algoritmo Genético (AG), que consiste em métodos adaptativos inspirados em teoria de processo genético e evolutivo da natureza, e que podem ser utilizados para resolver problemas de busca e otimização (FERNANDES, 2005, p. 115).

Segundo Fernandes (2005, p. 116), com o uso de AGs não há garantia de se encontrar a solução ótima para um problema, e se existirem técnicas especializadas para resolver o problema o mais provável é que os AGs sejam superados tanto em eficácia quanto em eficiência. Porém existem evidências empíricas de que respostas aceitáveis podem ser obtidas em um tempo real bastante razoável, e no que se trata de busca e otimização eles são realmente muito eficazes. No projeto P-Finder se escolheu utilizar essa técnica pelo problema (localizar placas automotivas em imagens) possuir muitas variáveis envolvidas (principalmente no sentido de existir muita variação de cores entre as imagens de automóveis resultante de luminosidade diferente, qualidade da foto entre outros fatores) sendo assim necessário utilizar uma técnica que buscasse pela melhor solução global e que se tornasse eficaz para a maioria das imagens de automóveis, o uso de técnicas especializadas poderia eventualmente resultar em menor generalização quanto ao uso de imagens muito diferentes entre si.

Para Norvig (1995, p. 619) Algoritmos Genéticos podem ser classificados como:

> Acontece que o que há de bom para a natureza também é bom para os sistemas artificiais. [...] o ALGORITMO GENÉTICO, que começa com um conjunto de um ou mais indivíduos e são aplicados operadores de seleção e reprodução para "evoluir" um indivíduo com sucesso, que será medido pela função fitness. Existem várias opções para os indivíduos. Eles podem ser agentes de funções inteiro, caso em que a função fitness é uma função para medir o desempenho ou recompensar a função, e a analogia para a seleção natural é maior. Eles podem ser uma componente das funções do agente, e neste caso a função fitness é o crítico. Ou eles podem ser absolutamente nada que possa ser enquadrado como uma otimização problema. (ARTIFICIAL INTELLIGENCE: a modern approach, 1995).

Para compreender melhor como os AGs funcionam, podemos primeiramente entender o processo no qual ele é inspirado: evolução de espécies.

Na natureza, simplificadamente, um indivíduo é criado a partir de cadeias de DNA, chamados cromossomos. Esses cromossomos possuem seqüências modulares chamadas de genes, que contém quatro proteínas básicas: Thymine, Adenine, Cytosine e Guanine (T, A, C e G, respectivamente). Esses genes contém as informações sobre as “configurações” dos “traços” de um determinado indivíduo.

Quando dois indivíduos reproduzem, as cadeias de DNA dos pais são divididas e os filhos são criados a partir de uma nova cadeia, construída com metade

do DNA de um dos pais, e metade do outro. Esse processo é chamado de recombinação genética ou cruzamento.

A recombinação genética resulta em uma nova seqüência para um indivíduo da espécie. Se a partir dessa nova seqüência os traços desse indivíduo gerado forem bem adaptados para ele sobreviver no ambiente em que viverá, ele terá uma vida completa e será apto a se reproduzir, e passará metade de sua cadeia de DNA para gerações futuras. Se eventualmente as características criadas a partir da recombinação genética resultarem em um indivíduo que herde características fracas ou ruins dos pais, ele pode não sobreviver por muito tempo no ambiente ou não ser apto a se reproduzir (por eventuais problemas biológicos ou até por questões sociais). Pode-se dizer que um indivíduo tem uma qualidade de aptidão para sobreviver em seu meio, e somente os que forem melhores adaptados tem mais chance de sobrevivem. Dessa maneira, ao longo das gerações, a tendência é gerar indivíduos cada vez com características melhores. Eventualmente ocorre também um fenômeno chamado mutação, onde alguns genes do indivíduo filho acabam se tornando completamente novos, não sendo “herdados” de nenhum de seus pais. O resultado disso geralmente resulta em alterações negativas, porém eventualmente pode-se ter benefícios novos (SCHWAB, 2005, p.414).

Os AGs simulam essa teoria em sistemas de computadores, contendo basicamente a mesma estrutura “funcional”, onde encontramos indivíduos, que possuem seus cromossomos, sua aptidão ao meio é numericamente medida a partir do mecanismo nomeado Fitness, eles tem capacidade de se reproduzir e conforme as gerações vão avançando eles tendem a evoluir, inclusive o fator mutação é simulado nos genes. Os indivíduos representam possíveis soluções para um problema.

Um AG simples resume-se, basicamente, aos seguintes passos, segundo Charles (2008, p. 107):

a) Inicia-se uma população de indivíduos com cromossomos aleatórios.
b) Mede-se a aptidão (fitness) de cada cromossomo criado na população.
c) Cria-se novos cromossomos a partir do acasalamento entre os indivíduos da população. E aplica-se mutação em alguns genes aleatoriamente.
d) Deleta-se os membros menos aptos da população atual.

e) Mede-se a aptidão (fitness) dos novos cromossomos e os insere na população.
f) Repete-se os passos “c” - “e” até a população convergir.

Convergência, segundo Fernandes (2005, p. 119), está relacionado com a idéia de progressão para a uniformidade dos genes, que levará ao ótimo global conforme as iterações ocorrem. Também à medida que o número de gerações aumenta é mais provável que a adaptação média (aptidão dos genes medida pela Fitness) se aproxime do melhor indivíduo. Existem problemas relacionados com a convergência prematura dos indivíduos, que são tratados utilizando as técnicas de mutação e serão verificadas mais adiante.

Porém a primeira questão que pode vir à mente é como implementar computacionalmente os indivíduos e seus genes, sua capacidade de reprodução e como definir seu cruzamento e mutação. Verificaremos isso nos próximos capítulos.

2.2.1 Genoma

Os indivíduos, que representam uma possível solução para um determinado problema dentro dos AGs, devem possuir os genes que o definem, e na maioria das vezes eles são representados por cadeias de bits. Entretanto, há como representá-los de outras maneiras, até mesmo como uma lista contendo algum tipo de informação específica (SCHWAB, 2005, p. 417).

Primeiro é preciso determinar a estrutura desses genes de acordo com o problema em questão. O que e quantos traços específicos está se buscando é uma pergunta que pode ajudar a começar a determinar a estrutura (SCHWAB, 2005, p. 416). Para exemplificar, podemos definir que procuramos retângulos em uma imagem para testá-los de alguma maneira específica qualquer. Basicamente, um retângulo pode ser facilmente desenhado graficamente se tivermos um ponto inicial (coordenadas x,y que estejam dentro dos limites da imagem), um valor para altura e um para largura. Nosso genoma poderia ser estruturado da seguinte maneira: os quatro primeiros bits representando o valor da posição X, os quatro seguintes a posição Y, outros quatro representando altura H e quatro para representar a largura W.

Portando nosso genoma representa: XYHW, o que poderia resultar um seguinte valor em bits gerado aleatoriamente:

1001110000100010

Onde os valores estão representados, os primeiros (em negrito) se referem ao valor da coordenada X:

1001110000100010

Os quatro seguintes (em negrito) se referem ao valor da coordenada Y:

1001**1100**00100010

E assim sucessivamente. Um detalhe que se pode observar, é que podemos limitar inclusive os limites de valores, no caso do exemplo, o maior número decimal possível de ser representado para cada característica é 15, pois é o valor máximo que se consegue apresentar utilizando quatro dígitos binários.

2.2.2 Seleção de Indivíduos

Os indivíduos dentro de um AG devem se reproduzir para que as populações evoluam para encontrar uma boa solução. E um método para seleção dos indivíduos que irão cruzar deverá ser utilizado.

Basicamente, cada indivíduo tem uma chance para reproduzir proporcional à sua aptidão (medida pela Fitness), ou seja, um indivíduo que possua uma maior aptidão terá maiores chances de ser selecionado para reproduzir do que outro que tenha menor, porém todos possuem chance de cruzamento, mesmo que seja pequena (CHARLES, 2008, p.111).

Para se escolher os indivíduos que irão reproduzir, foram desenvolvidas diversas técnicas, inicialmente utilizamos AG a técnica chamada de “Roleta” (Roulette Wheel Selection) (CHARLES, 2008, p.111).

A técnica de Roleta consiste em determinar uma chance aleatória de um indivíduo ser selecionado proporcional à nota de aptidão determinada pela Fitness, os que possuírem maior nota terão maior chance de serem selecionados (terão a maior fatia da “roleta”). É importante notar que essa seleção não remove da roleta o indivíduo que já foi selecionado, portando um genoma que recebeu uma boa avaliação da Fitness poderá ser selecionado várias vezes. Nota-se também, que por

se tratar de uma chance aleatória de seleção, não há garantia de que o melhor indivíduo será selecionado, por essa razão a técnica de Elitismo (copiar o melhor indivíduo de uma geração automaticamente para a próxima) é frequentemente utilizada (SCHWAB, 2005, p. 422).

Outra técnica é a chamada Stochastic universal selection, que consiste na implementação da técnica da roleta (Roulette Wheel Selection) mas agora não se "gira" a roleta, é pego um numero de indivíduos que se deseja selecionar (n), e seleciona o proprietário da fatia da roleta com 1/n incrementos ao longo da roda. Portanto, para 10 indivíduos, seleciona-se o proprietário da fatia da roleta apontando para cada 1/10 da roda. A vantagem sobre o modo regular de roleta é por se manter espalhado os valores das aptidões escolhidas baixo e ainda manter a diversidade genética (SCHWAB, 2005, p. 422).

Uma outra técnica apresentada por Schwab (2005, p. 422), é a Tournament selection. Nessa técnica, um certo número de indivíduos são retirados aleatoriamente da lista de habitantes da população, e os que tiverem os maiores valores de aptidão determinados pela Fitness irão fazer parte da próxima geração. Depois todos são colocados novamente na lista, e esse processo é repetido por quantas vezes for necessário para se obter uma nova geração.

A técnica de seleção adotada no projeto P-Finder é a chamada Asymptotic Selection, proposta por Mendonça, Pozzer, Raittz (2008 p. 3). Que consiste em basicamente três etapas:

> a) Classificar os indivíduos em ordem decrescente quanto à aptidão obtida, dentro de uma lista.
> b) Determinar um número aleatório no intervalo entre 0 e 1 e elevá-lo por uma constante determinada, normalmente entre 0,8 e 1,2 e multiplicar o resultado pelo tamanho da população.
> c) O resultado obtido será o índice do indivíduo a ser selecionado.
>
> O uso dessa técnica proporciona uma garantia matemática justa para escolha dos indivíduos sem descartar os que possuem menor aptidão, o que respeita o conceito dos AGs, e se mostrou bastante eficaz no projeto P-Finder.

2.2.3 Reprodução

Após selecionar os indivíduos se obtém aqueles que irão se reproduzir. A técnica adotada para realizar o cruzamento entre dois indivíduos para o projeto P-Finder é o Single Point Crossover (SCHWAB, 2005, p. 422), que consiste em determinar uma posição aleatória, dentro do limite de tamanho do genoma, e trocar todos os genes a partir dessa posição de um dos pais com o outro, para gerar o "descendente", por exemplo, se obtermos os seguintes genomas de 10 bits selecionados para reprodução:

"Pai" 1: 1001110101

"Pai" 2: 0010111011

O cruzamento poderia ocorrer da seguinte maneira:

Número aleatório entre 1 e 10 (limites do genoma) = 3.

Portanto teríamos a "linha de corte" na terceira posição do genoma, o descendente herdaria os três primeiros bits do primeiro "Pai" (em negrito no exemplo) e o restante do segundo (sem negrito no exemplo):

Descendente: **100**0111011

Além dessa técnica de cruzamento, existem outras várias como: Multipoint crossover, uniform crossover, discrete crossover, intermediate crossover, line crossover, partially mapped crossover, order-based crossover, position-based crossover, todos descritos por (SCHWAB, 2005, p. 422 – p. 426).

2.2.4 Mutação

Infelizmente, se apenas for realizado a reprodução dos indivíduos de uma população, pode ser que que haja regiões que nunca serão exploradas pelo AG, e eventualmente pode ser que essas regiões sejam as melhores para a busca de solução (CHARLES, 2008, p. 110). Como a população inicial é criada de maneira aleatória, pode ser que os genomas sejam evoluídos ao máximo apenas em uma região específica do problema, o que obteríamos o chamado máximo local, que consiste na melhor solução para aquele espaço pesquisado, porém pode não ser a melhor solução global. Para tentar evitar que o AG se limite à apenas uma área

eventualmente, é utilizado uma chance de mutação entre os genomas, que basicamente é a troca de um valor de bit aleatório do genoma (CHARLES, 2008, p. 110).

A técnica de mutação utilizada no projeto P-Finder é a Binary mutation, que consiste na troca do valor de um bit aleatório do genoma (SCHWAB, 2005, p.426), como observado no exemplo a seguir, onde se supõe o seguinte genoma representado por 8 bits:

Genoma antes da mutação: 10011001

Posição aleatória entre 1 e 8 (limites do genoma) = 2.

Portanto o valor binário na posição aleatória será trocado, representado em negrito no exemplo:

Genoma depois da mutação: 1**1**011001

Há outras técnicas de mutação citadas por Schwab (2005, p.426): Exchange mutation, displacement mutation, insertion mutation, real-value mutation.

Há chances de mutação em um genoma definido no algoritmo, que geralmente são baixas, segundo Carvalho (2004) a influência da taxa de mutação no algoritmo genético pode ser dada por:

> Uma baixa taxa de mutação previne que uma dada posição fique estagnada em um valor, além de possibilitar que se chegue em qualquer ponto do espaço de busca. Com uma taxa muito alta a busca se torna essencialmente aleatória.

2.2.5 Fitness

Uma vez determinados a estrutura dos genomas dos indivíduos, deve-se determinar uma estratégia para medir a aptidão das características geradas por esse genoma no “ambiente” determinado. A essência da eficácia de um AGs está nessa função, pois é ela quem irá “decidir” quão boa uma possível solução para um determinado problema é. Portanto a evolução de maneira coerente das populações geradas é diretamente relacionada à qualidade da estratégia da Fitness desenvolvida (SCHWAB, 2005, p. 420).

Podemos exemplificar como uma função Fitness pode ser implementada a partir de uma idéia simples, ao procurar por um retângulo preto em uma imagem que possua fundo branco, para isso podemos aproveitar o genoma criado para o indivíduo no capítulo 3.2.1 desde documento.

Uma possibilidade que poderia ser usada é pegar o genoma e desenhar o retângulo conforme as características dele na imagem, e verificar os valores RGB presentes na imagem a partir deste genoma. Para avaliar o quão bom esse indivíduo é, podemos considerar que os valores RGB que estiverem presentes na borda desse retângulo devem ser em sua maioria devem ser o equivalente ao branco, e os valores internos do retângulo deve ser em sua maioria preto. Se tirarmos a média entre esses valores podemos definir um valor numérico que tenderá a ser melhor para o indivíduo que tiver mais branco na borda e mais preto no centro. Com as iterações do genético provavelmente ele irá encontrar retângulos pretos na imagem.

O desempenho no quesito tempo do AG estará relacionado também com a velocidade da Fitness para avaliar um genoma. Pois a cada iteração criada a Firtness irá avaliar cada indivíduo, um por um, e muitas vezes são necessários milhares de indivíduos para se chegar à uma solução ótima.

2.2.6 Algumas Aplicações Práticas

Seguem alguns exemplos de sistemas em que podem ser utilizado AG:

a) Controle de Sistemas Dinâmicos;
b) Indução e Otimização de Bases de Regras;
c) Encontrar Novas Topologias Conexionistas;
d) Engenharia de Sistemas Neurais Artificiais;
e) Modelagem de Estruturas Neurais Biológicas;
f) Simulação de Modelos Biológicos;
g) Comportamento;
h) Evolução;
i) Evolução Interativa de Imagens;
j) Composição Musical.

3 TECNOLOGIAS UTILIZADAS

3.1 JAVA

Seu início deu-se pela Sun Microsystems, em dezembro de 1990, em um projeto conhecido como “Green” e segundo Anselmo (2003, p.10) como uma tentativa de “[...] desenvolver uma plataforma de programação para hardwares. A idéias era criar um Controle Remoto Universal que poderia à distância controlar todos os aparelhos eletrodomésticos e computadores [...]”. Somente foi chamada Java em 1995, depois de verificado que o projeto “Green” havia se tornado caro e inviável. Esta acabou tornando-se uma linguagem para aplicações feitas para a Internet.

Trabalha gerando código para uma máquina virtual, que é simulado por um JRE (Java Runtime Environment). Uma de suas vantagens é o suporte a diversos sistemas operacionais e a fácil distribuição dos pacotes de código com extensão “.jar”. Abaixo, segue a estrutura básica dos programas escritos em JAVA:

```
package exem;
//comentário - este é um pacote//
public class exemplo {
// comentário - esta é uma classe//
    public static void main (String args[]) {
    }
    private Object x;
    {
// comandos... //
    Object exemplo = x;
}
}
```

FIGURA 1 - EXEMPLO CÓDIGO ESCRITO EM JAVA.

Trata-se de uma linguagem Orientada a Objetos. Isto, de maneira simplificada, foi criado para tentar simular o mundo real dentro do computador, através da utilização de objetos para escrever programas.

3.1.1 Justificativa para a Utilização da Linguagem Java

A princípio, utilizamos Java pelo fato de ser uma linguagem atual e multiplataforma (adaptável a vários sistemas operacionais). Outra vantagem é sua fácil utilização com aplicações em UML (Unified Modeling Language) Outro fato importante é o fato de ser uma linguagem bastante conhecida, sendo utilizada em diversas empresas como única opção para o desenvolvimento de softwares, além de ser aquela na qual o grupo possui maior conhecimento técnico, que foi adquirido durante o período de graduação.

3.2 ECLIPSE

O Eclipse é um software para desenvolvimento de programas para computador, que possui código aberto. Escrito em Java, foi criado a partir Visual Age for Java, da IBM, que necessitava de um reposicionamento comercial e refaturação arquitetônica. Uma de suas características é a utilização do SWT ao invés do Swing. Trata-se de um software voltado para a agilidade no desenvolvimento, não focalizando muito a parte gráfica.

É um framework com número maior de plugins que o NetBeans , JBuilder, entre outros.

3.3 UML (UNIFIED MODELING LANGUAGE)

As primeiras versões, criadas pela Rational Software Corporation, da UML surgiram em 1996, como uma tentativa de padronizar os processos de documentação de software. Segundo Anselmo (2003, p.10), a UML é “[...] um esquema de representações gráficas, sendo a simples reunião de esquemas já

existentes para a modelagem de projetos Orientados a Objetos, daí o termo Unificada".

O texto de Booch, Rumbaugh e Jacobson (2000, p. 6), nos mostra por que utilizar a modelagem nos softwares e alguns objetivos que podem ser alcançados através dela.

> [...] Construímos modelos para compreender melhor o sistema que estamos desenvolvendo.
> Com a modelagem, alcançamos quatro objetivos:
> 1. Os modelos ajudam a visualizar o sistema como ele é ou como desejamos que seja.
> 2. Os modelos permitem especificar a estrutura ou o comportamento de um sistema.
> 3. Os modelos proporcionam um guia para a construção do sistema.
> 4. Os modelos documentam as decisões tomadas (UML Guia do Usuário, 2000, p. 6).

Segundo Melo (2004, p.42), os diagramas existentes na UML divididos em diagramas estruturais que tem por função "mostrar as características do sistema que não mudam com o tempo" e diagramas dinâmicos que "mostram como o sistema responde às requisições ou como o mesmo evolui durante o tempo". No quadro abaixo, segue a divisão com os diagramas correspondentes:

Diagramas Estruturais	• Diagrama de classes • Diagrama de objetos • Diagrama de componentes • Diagrama de pacotes • Diagrama de implantação • Diagrama de estrutura composta
Diagramas Dinâmicos	• Diagrama de casos de uso • Diagramas de Interação: - Diagrama de visão geral - Diagrama de seqüências - Diagrama temporal - Diagrama de comunicação - Diagrama de atividades - Diagrama de máquina de estados

TABELA 1 - DIAGRAMAS EXISTENTES NA UML.

Como exemplos de duas ferramentas que suportam UML, podemos citar StarUML E JUDE.

3.4 FILTROS DE IMAGEM

Filtros de Imagem ou Realce é uma técnica de melhoria de imagem que segundo Facon (2005, p.25) “são essencialmente projetados para manipular a imagem a partir das características psicofísicas do sistema de referência que é a visão humana”. De forma simplificada a definição do Laboratório de Geoprocessamento do Departamento de Geologia (Labgis) da Universidade Federal do Rio de Janeiro (UFRJ) (2007) nos diz que o filtro de imagem “Consiste de transformações e manipulações dos dados da imagem, com o objetivo de ajudar o analista humano na extração de informações para a interpretação direta ou como entrada para classificação”. Conforme Falcon (2005, p.25), os filtros de imagem “são relacionados com expansão de contraste, realce de bordas e suavização”.

Em nosso projeto não existem filtros de imagens construídos por nós. Os filtros utilizados foram retirados do site JH LABS Java Image Processing.

3.5 OCR

Tecnologia que surgiu em 1950, através das pesquisas de David Shepard e Louis Tordella, para a automação de dados da Agência de Segurança dos Estados Unidos, o OCR (Optical Character Recognition), trata-se de uma tecnologia utilizada para reconhecer caracteres de texto em imagens, transformando-os em texto para edição. O scanner é um exemplo popular de utilitário que utiliza algum programa OCR, que são usados para obter texto de páginas impressas, substituindo a digitação manual.

Um requisito necessário para o software OCR é oferecer suporte à língua Portuguesa, para que os caracteres acentuados possam ser reconhecidos.

O OCR funciona da seguinte maneira: Inicialmente, o programa examina a imagem para mapear os espaços em branco, reconhecendo títulos, colunas, parágrafos e imagens, o que permite manter a ordem correta do texto.

Na segunda etapa, os caracteres são comparados com os modelos de fontes suportadas pelo OCR. Havendo certa porcentagem de coincidência, o

caractere é reconhecido. Quando ocorre o não-reconhecimento, o item passa por uma terceira etapa, que realiza a análise geométrica de cada caractere (calculo de altura, largura, e combinações de retas, curvas e áreas em branco). Neste processo é usada a lei da probabilidade.

Se após todas estas etapas, o reconhecimento não for possível, uma última alternativa pode ser mostrar individualmente o bitmap de cada caractere não reconhecido e pedir ao usuário que o substitua pela letra correspondente.

3.5.1 Recomendação de Uso de um Software OCR para Reconhecimento dos Caracteres da Placa

Devido à complexidade de construção de um OCR, ele não faz parte do nosso escopo, porém, a criação de uma ferramenta poderia ser explorada em um trabalho futuro. Então, foi feito um estudo sobre alguns OCR's já prontos para recomendarmos como uma possível ferramenta a ser utilizada em conjunto com o P-Finder futuramente. Inicialmente, procuramos uma biblioteca em Java, o que possibilitaria uma integração direta, mas não encontramos alguma que pudesse atender às nossas necessidades.

Os softwares encontrados em nossas pesquisas foram:

a) Gocr-0.45

b) Libgocr-0.7.2

c) Mdk-0.9.1

d) Tesseract-1.03

e) Linuxtopocr

f) TopOCR

4 PROCESSO DE DESENVOLVIMENTO

4.1 O HABITANTE

É gerada uma população inicial, Habitante por Habitante, conforme a quantidade especificada na interface gráfica, ou com o valor padrão. Eles são adicionados uma lista, formando uma população inicial. A população é passada no construtor da classe Generation que chama a função de avaliação da Fitness, passando os Habitantes, um a um. Após a avaliação da população todos os integrantes têm uma nota definida. É feita uma ordenação decrescente com base na nota usando a função sort da classe Collections, passando a população como parâmetro. Posteriormente há uma interação invocada de acordo com a quantidade de geração, setada na interface ou com valor padrão, para que sejam feitas as gerações. Para se criar uma nova geração, poderá ser usado o elitismo, conforme selecionado pelo usuário na interface ou com o valor padrão. Após isso, temos uma outra interação para obtermos dois Habitantes, efetuarmos o cruzamento deles e obtermos filhos até preencher uma nova população com o tamanho da população inicial. A seleção dos Habitantes é feita através da Asyimptotic Selection.

```
Random rnd = new Random();
int index1 = 0;
int index2 = 0;

 while (newGeneration.size() < Parameters.NUM_INHABITANTS) {
   index1 = (int) (population.size() * Math.pow(rnd.nextDouble(), 1.8));
   do {
      index2 = (int) (population.size() *
            Math.pow(rnd.nextDouble(), 1.8));
   } while (index1 == index2);
Inhabitant child =
   population.get(index1).intersect(population.get(index2));
   newGeneration.add(child);
```

FIGURA 2 - IMPLEMENTAÇÃO DA ASYMPTOTIC SELECTION.

Para o cruzamento, é escolhido um ponto aleatório (entre a posição 0 e a 33, pois da 33 até a 44 é a informação da altura e esta é recalculada, mudando de valor. Assim não faz sentido que o ponto de corte esteja nessa característica) no DNA e um dos Habitantes é selecionado e pega-se o início do DNA de um e o final do outro, segundo a linha de corte, gerando um novo Habitante.

Após o cruzamento podem ser feitas mutações, onde um bit do Habitante é trocado pelo seu valor contrário (se era 1 fica 0, se 0 torna-se 1). Há um loop que é executado 10 vezes e em cada vez há uma chance de haver uma mutação. É possível alterar a chance de mutação. Isso é feito na interface, pelo usuário, ou será usado o valor padrão, definido na classe Parameters. O valor varia de 0 a 500, sendo que quanto maior o valor, maior a chance de ocorrer mutação.

E finalmente, com a nova geração, temos uma nova população que terá seus membros avaliados e voltamos para o início do ciclo até que condições como: encerrado o número de gerações ou algum Habitante tenha chegado a nota definido na classe Parameters como pertinence degree, sejam satisfeitas. Ao final, temos uma lista com os melhores Habitantes de cada geração, e na posição zero estará o melhor. O resultado é mostrado na tela e o sistema gera um recorte da imagem de acordo com as coordenadas, largura e altura do Melhor Habitante gravando na mesma pasta da imagem selecionada com o nome “result.jpg”.

Veja o caminho que o Habitante faz durante a localização da placa:

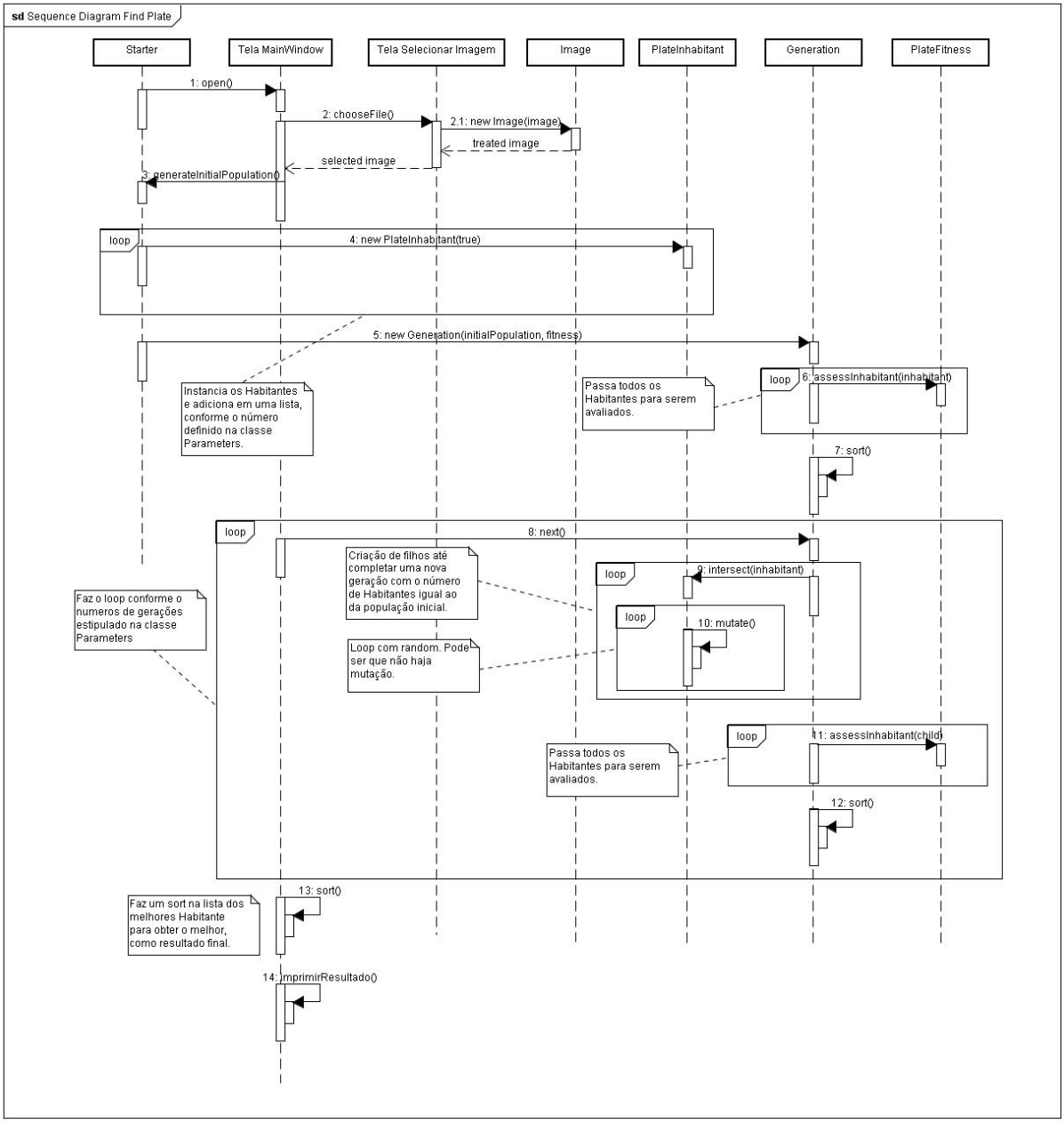

FIGURA 3 - DIAGRAMA DE SEQÜÊNCIA ENCONTRAR PLACA.

4.2 IMPLEMENTAÇÃO DO ALGORITMO GENÉTICO

Para o presente projeto, foi elaborado um Algoritmo Genético (AG) totalmente desenvolvido por nossa equipe em linguagem Java que apresenta a seguinte estrutura:

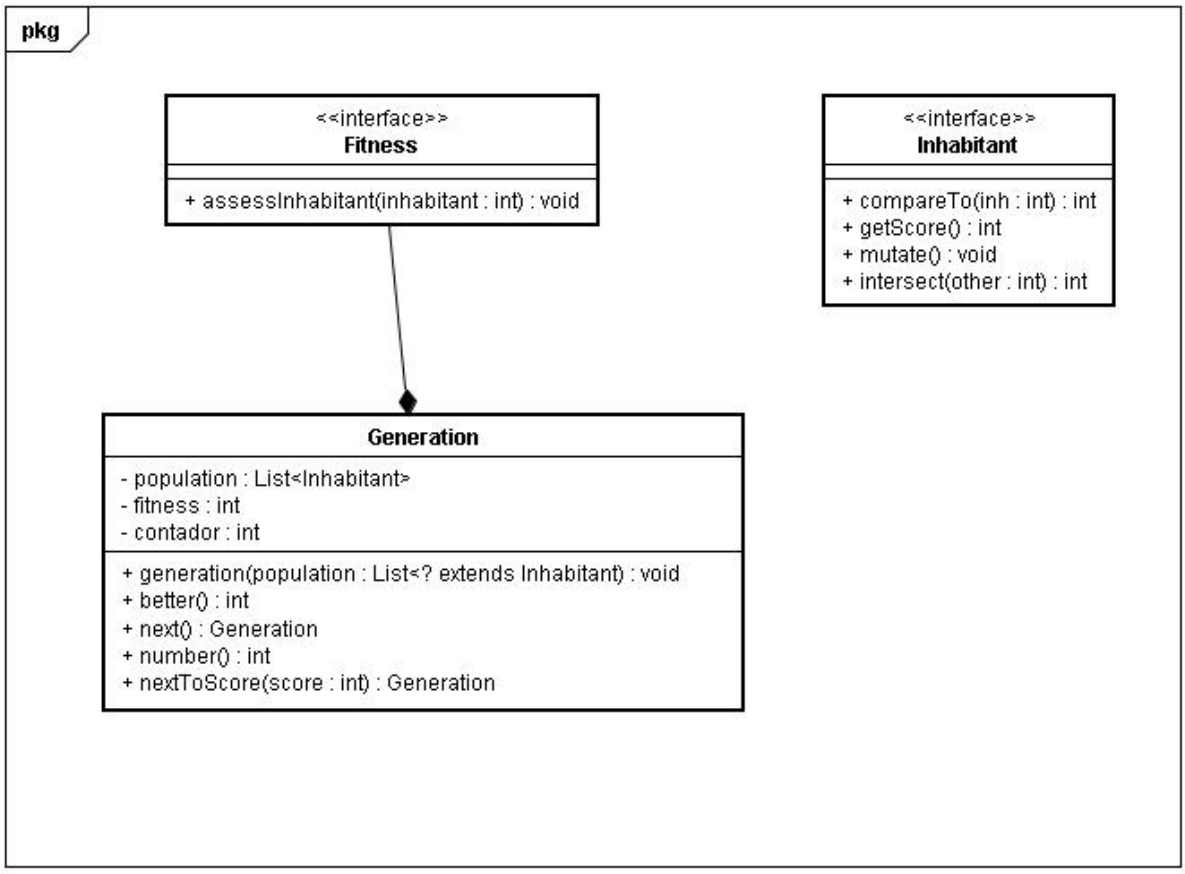

FIGURA 4 - DIAGRAMA DE CLASSE DO GENÉTICO.

Nota-se que há uma parte genérica e uma parte específica para o objetivo do AG. Determinou-se que seria conveniente fazer um AG, pois teríamos mais controle sobre o código e compreensão, além de ser um ponto positivo acadêmico a mais e servir de aprendizado.

Na Fitness e no Habitante foi usado o padrão de projeto Strategy descrito por Metsker (2004, p. 217):

> A intenção da Strategy é encapsular estratégias alternativas, ou abordagens, em classes separadas, cada uma das quais implementando uma operação em comum. A operação estratégica define as entradas e a saída de uma estratégia, mas deixa a implementação para as classes individuais. As classes que implementam as várias abordagens implementam a mesma operação e são, assim, intercambiáveis, apresentando a mesma interface dos clientes.

A Façade foi usada na classe Generation e segundo Metsker (2004, p. 49):

> Uma Façade (fachada) é uma classe com nível de funcionalidade que se situa entre um toolkit e uma aplicação completa, oferecendo um uso

> comum das classes de um pacote ou de um sistema. A interação do Façade é fornecer uma interface que se torne um sobsistema fácil de usar.

Embora não seja efetivamente uma implementação do padrão, a classe FilterApplyer foi inspirada no padrão composite de Metsker (2004, p. 61):

> Um Composite é um grupo de objetos no qual alguns deles podem conter outros; assim, um objeto pode representar grupos e outro, um item individual, ou folha (...). A intenção de um Composite é “permitir aos clientes tratar objetos individuais e composições de objetos uniformemente”.

Como a interface BufferedImageOp é muito complexa, optamos por criar apenas uma classe que gerencia a composição, sem que ela implemente a interface de suas filhas.

O Habitante possui um DNA e uma nota. Em seu DNA temos quatro informações necessárias para atender ao nosso propósito: coordenadas x e y, altura e largura. Com estas informações, monta-se um retângulo em um lugar específico da imagem, com o intuito de enquadrar a placa. O DNA é representado de forma binária sendo 44 caracteres no total. Cada conjunto de 11 representa uma característica. O Habitante também possui métodos que auxiliam na validação das características que são feitas na Fitness. Por exemplo, temos uma imagem de 500 x 500 pixels para localizar a placa e as coordenadas de um Habitante são (100, 510). Com certeza este Habitante não enquadrou a placa, pois ele está formando um retângulo qualquer fora da imagem, então ele recebe nota zero e não passa pela avaliação completa.

Exemplo de estrutura do DNA:

DNA = 00000000011000000000100000000011000000000111
Dividindo pelas características, temos:
X = 00000000011 (valor decimal: 3)
Y = 00000000010 (valor decimal: 2)
Altura = 00000000110 (valor decimal: 6)
Largura = 00000000111 (valor decimal: 7)

A nota é um número inteiro positivo de valor inicial zero. A Fitness é responsável pela avaliação do Habitante e ela seta a nota dele. Quanto maior a nota, melhor o Habitante, ou seja, melhor é o enquadramento da placa que este Habitante conseguiu, segundo suas características.

A altura, inicialmente, também não fazia parte, mas, devido à função de avaliação por seleção, foi necessário integrar a altura ao DNA, pois a altura fornecida pode não ser de um retângulo proporcional a uma placa de automóvel. A altura não é uma característica necessária no DNA, pois com a largura, podemos fazer um calculo baseado nas proporções de uma placa (40cm x 13cm) para determinar a altura e é assim que é feito quando é feita uma avaliação normal de um Habitante.

4.3 ELABORAÇÃO DA FITNESS

Fitness é a responsável por fazer a avaliação de cada Habitante de uma população. É a parte “inteligente” do sistema. Foi a parte mais demorada e trabalhosa e também a parte que mais demora a ser processada, pois há muita informação que passa por ela muitas vezes e há diversas interações. Por exemplo, para uma população de 500 Habitantes com em média 30000 pixels (para um retângulo de 300x100), 200 gerações (sem elitismo), temos a possibilidade de analisar aproximadamente nove bilhões de pixels (3 x 30000 x 500 x 200, sendo respectivamente três verificações do retângulo, uma verticalmente outra horizontalmente e outra da verificação da cor predominante de fundo, 30000 pixels de cada Habitante, 500 Habitantes e 200 gerações).

Até chegar no resultado final da Fitness, passa-se por diversas dificuldades, pois começamos a fazer nossos primeiros testes com as configurações voltadas para encontrar retângulos pretos em uma imagem branca. Ao partir para o nosso objetivo que era a placa em si, tentamos diversas técnicas, como tentar encontrar quinas na imagem, fixar uma cor específica como o fundo da placa, fazer um tratamento que deixasse as placas com fundo preto, mas isso tudo não funcionou. No caso do tratamento, as placas tinham contraste e luminosidade diferentes, o que eliminou a chance de fixar a cor de fundo da placa como a cor preta, por exemplo.

Veja o diagrama abaixo:

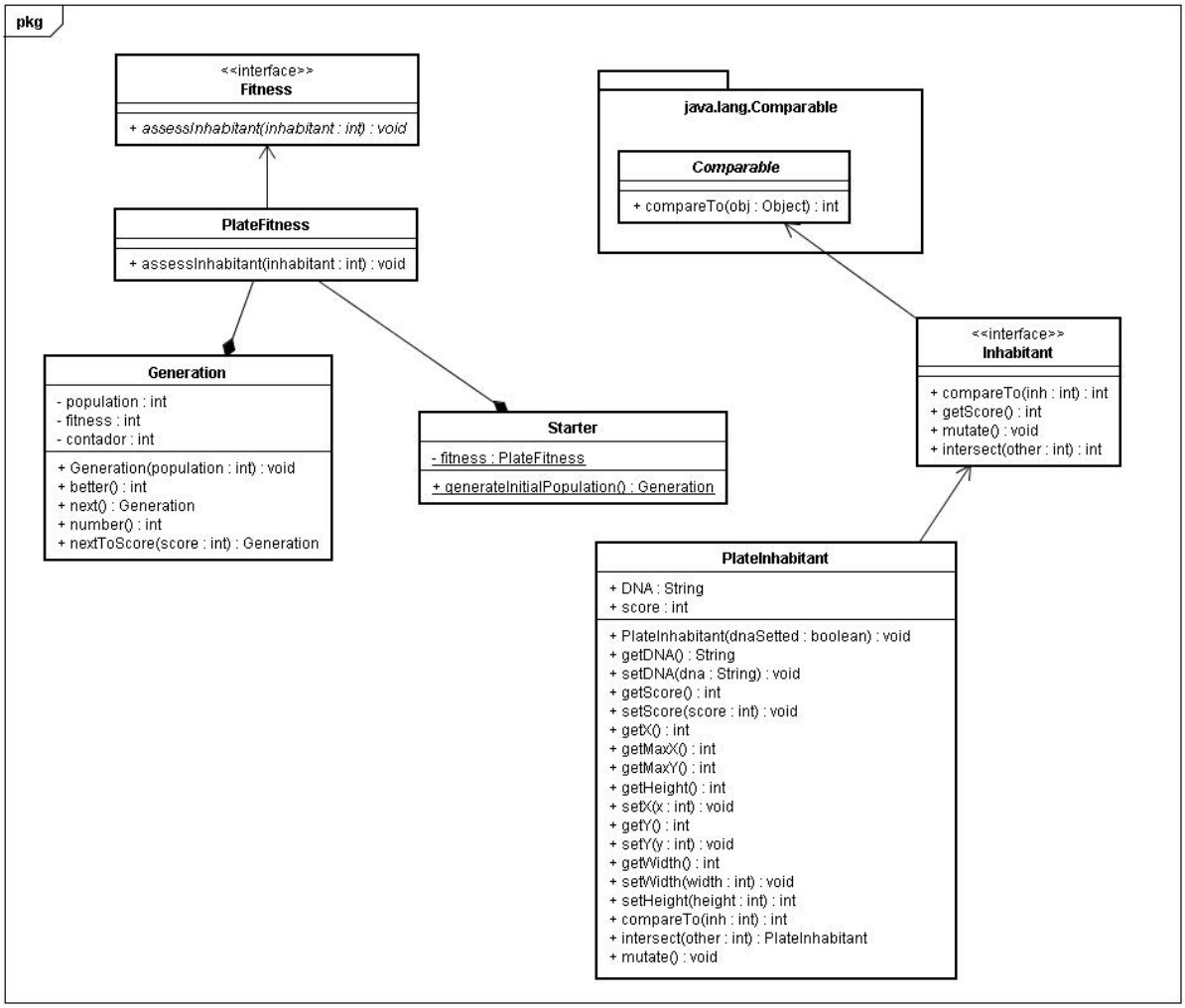

FIGURA 5 – DIAGRAMA DE CLASSE.

Há a interface Fitness que possui o método assessInhabitant(Inhabitant inhabitant) que deve ser implementado ao se criar uma fitness especializada para a resolução proposta. No presente caso, temos a classe PlateFitness implementando a interface Fitness. Em sua estrutura temos a lógica responsável pela avaliação. Inicialmente foi aplicada a técnica de códigos de guarda (Fowler, Martin). A imagem (que foi selecionada pelo usuário) é obtida da classe Parameters. Caso o Habitante, passado como parâmetro, possuir características que o inviabilizem de ser avaliado, este já recebe nota igual a zero e a avaliação é encerrada. Por exemplo: se as coordenadas do Habitante estiver fora da imagem ou se a largura do Habitante for maior ou menor que a definida pelo usuário.

Há uma outra situação que é um Habitante já avaliado ser passado para a Fitness novamente (isso ocorre quando há elitismo). Não é necessário avaliar um Habitante que já possui uma nota, pois este já passou pelo processo.

Após esta verificação inicial, caso o Habitante não corresponda às preposições, o processo continua. É feita uma verificação dos pixels internos do retângulo do Habitante para verificar qual a cor predominante. Esta cor será admitida como a cor do fundo da placa (da imagem tratada). Mesmo que a placa não tenha sido enquadrada, no decorrer das verificações, este Habitante não receberá uma nota boa.

Agora temos a cor predominante no retângulo e é feita uma nova verificação dos pixels internos do retângulo. Esta nova verificação funciona da seguinte forma: é feita uma varredura dos pixels verticalmente no retângulo, da esquerda para a direita e a ciclo que corresponde a uma linha vertical, verifica-se, de acordo com a suposta cor de fundo da placa, se esta linha corresponde a uma linha fora da placa, uma linha na placa que não corta os caracteres (Linha) ou uma linha na placa que corta algum caractere (Linha Quebrada).

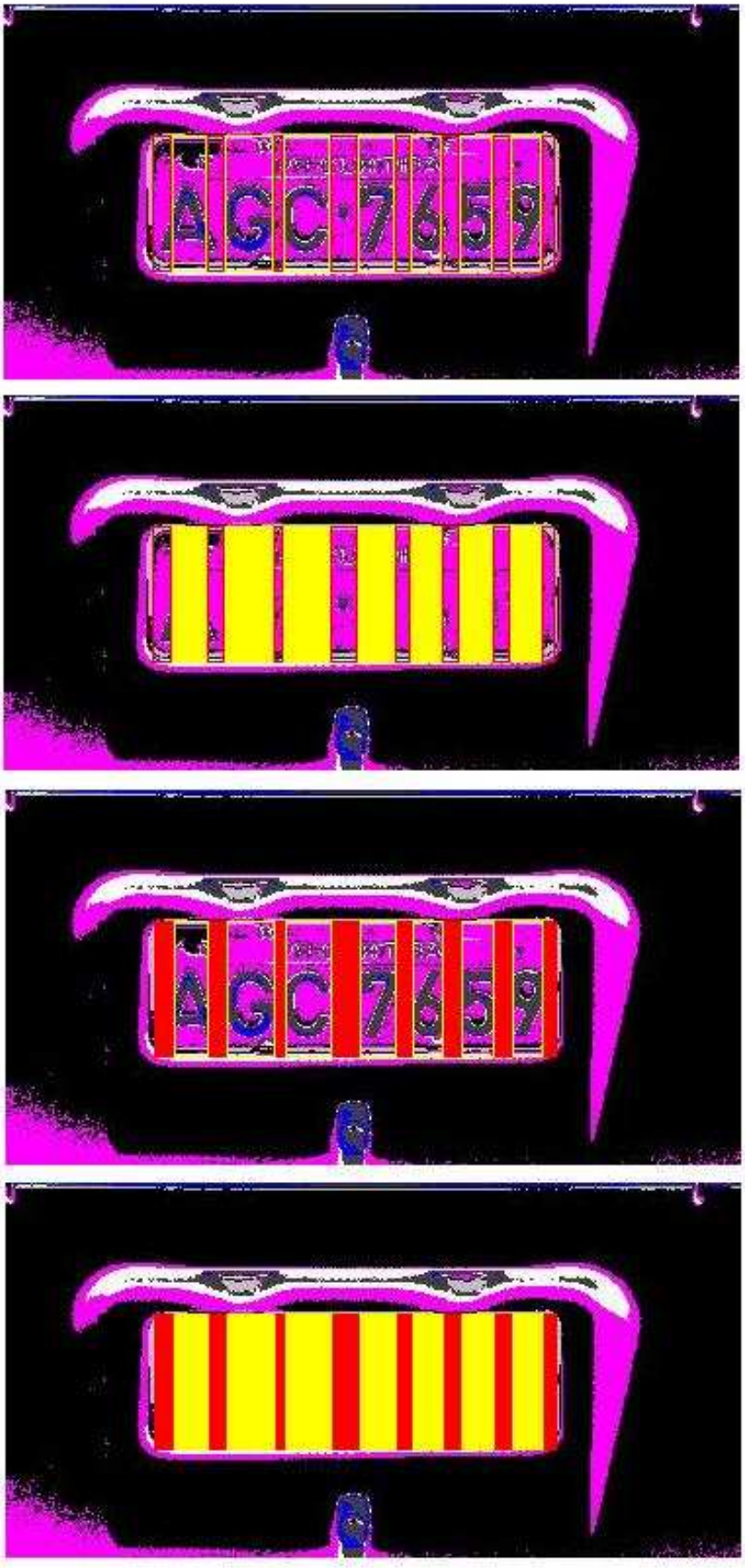

FIGURA 6 - REPRESENTAÇÃO DAS LINHAS E LINHAS QUEBRADAS.

Veja o código:

```
//Verifica qual a cor predominante interna ao retângulo
Color color = PlateColorVerify.verify(inh);
//Verificação da parte interna do retângulo verticalmente
for (int i = xHab + 1; i < xHab + lHab - 1; i++) {
 for (int ii = yHab + 1; ii < yHab + aHab - 1; ii++) {
        if(image.getRGB(i, ii) == color.getRGB()){
                //soma dos pixels pertencentes à placa
                contPlate++;
                //soma do auxiliar para calcular as linhas
                auxContLine++;
        }else{
                //soma dos pixels que são letras da placa
                contLetter++;
        }
        //soma total dos pixels
        contTot++;
}
//verificação para linha
if(auxContLine > (aHab * 0.6)){
        contLine++;
        listLines.add(true);
//verificação ára linha quebrada
}else if(auxContLine > (aHab * 0.4)){
        listLines.add(false);
        contBrokenLine++;
//soma de outras linhas
}else{
        contOtherLine++;
}
//zera para iniciar uma nova linha
auxContLine = 0;
}
```

Em seguida foi adotada uma estratégia de verificação de trocas de Linha e Linha Quebrada. Estas Linhas foram resultado do código acima e elas foram adicionadas a um ArrayList. Esta lista é varrida e verifica-se a troca de conjuntos de Linhas por Linhas Quebradas e vice-versa. A justificativa deste processo é que em uma placa, temos, da direita para a esquerda, algumas Linhas e, seguindo, logo temos Linhas Quebradas na primeira letra e entre esta e a segunda temos mais Linhas e novamente Linhas Quebradas nesta segunda letra e assim por diante. No total, este número ideal de trocas deve ser igual a 14, mas consideramos um intervalo entre 11 e 16, pois, como foi testado, descartando-se as trocas diferentes de 14, não se obtém um resultado tão bom quanto aplicando um intervalo, pois provavelmente um Habitante que tenha acertado uma parte significativa da placa seja descartado. Então a soma da nota é acrescida de 2100 (definimos este valor confirme testes, erros e acertos). Se não houve o atendimento à esta condição, há um return cancelando as verificações posteriores e mantendo o Habitante com a nota que ele estava, no caso, zero.

A próxima verificação é dos pixels da placa, agora horizontalmente a fim de verificar se, nos primeiros 40% da parte superior do retângulo há Linhas, entre os 40 e 85% Linhas Quebradas e nos últimos 15% novamente Linhas.

Prossegue-se analisando se a quantidade de outras linhas horizontais (linhas que não fazem parte da placa) é maior do que 40% da altura do Habitante, desconta-se 400 da nota; se a quantidade de Linhas (na vertical) está entre 40% e 20% da largura do Habitante, soma-se 500; se as Linhas Quebradas da vertical são maiores que 60% da largura, soma 300; se as outras linhas da vertical são maiores que 25% da largura, subtrai-se 300; se o percentual de pixels da placa é maior que 60% do total, soma o percentual * 300, se for igual ou menor, subtrai 300, se for maior que 80%, subtrai 300; se o percentual dos pixels das letras for menor que 45%, soma o percentual * 300. Após as verificações, se a nota for menor que 2500, as verificações são interrompidas e o Habitante mantém-se com a nota que veio (zero).

Analisa-se os contornos verticais do retângulo do Habitante. Se o percentual de pixels da borda da direita que pertencem à placa for menor ou igual a 20%, soma-se 500, se não, subtrai-se 300; mesma regra para a borda da esquerda. Não houve necessidade de se verificar as bordas superior e inferior, devido às todas verificações já feitas no Habitante anteriormente.

Todos estes valores citados, dentre outros comentados neste trabalho, foram fruto de testes realizados, onde houve diversas alterações até chegar nestes valores, ou seja, foi um resultado baseado em tentativas e erros, conforme descreve Brian Schwab (2004, p.447):

> [...] entender que a mistura exata a usar pode demorar muito tempo, experimentando com diferentes combinações destes fatores antes de descobrir a maneira correta de setar as condições necessárias para encontrar a solução. A única maneira real de se tornar bom em como utilizar os fatores corretos com um dado problema de AG é através de experimentação, especialmente por causa da implementação específica natural das soluções de AG.

Enfim, encerram-se as verificações e a nota é setada no Habitante. Caso ela seja menor que zero, há uma verificação que a faz receber o valor igual a zero.

4.4 TRATAMENTO DE IMAGENS

As imagens a serem passadas para a Fitness recebem um tratamento prévio com a finalidade de tornar mais fácil o processo de avaliação. A classe responsável por fazer este tratamento é a Treatment. Em uma primeira etapa, a imagem é passada para uma escala de cinza onde as informações das cores se tornam mais simples, pois se tem o valor de R (vermelho) igual ao G (verde) e igual ao B (azul).

Posteriormente, o contraste é reduzido pela metade e o brilho multiplicado por 1,4. Isso faz com que os pixels de tons próximos tenham uma diferença menor. Chegamos a estes valores conforme nossas tentativas, erros e acertos.

A última etapa é uma filtragem pelo valor de R (int entre 0 e 255), classificando cada pixel em um grupo e setando uma cor diferente para cada grupo. Desta forma temos muito menos tons para fazer a verificação (11 tons ao invés de 255), facilitando a análise no genético, pois o fundo da placa fica com um tom predominante e diferente dos caracteres.

Veja o código:

```
for (int i = 0; i < print.getHeight(); i++) {
        for (int ii = 0; ii < print.getWidth(); ii++) {
                int rgb = print.getRGB(ii, i);
                int R = (rgb >> 16) & 0xff;
                if(R < 35){
                        print.setRGB(ii, i, Color.YELLOW.getRGB());
                }else if(R < 45){
                        print.setRGB(ii, i, Color.CYAN.getRGB());
                }else if(R < 75){
                        print.setRGB(ii, i, Color.RED.getRGB());
                }else if(R < 90){
                        print.setRGB(ii, i, Color.BLUE.getRGB());
                }else if(R < 130){
                        print.setRGB(ii, i, Color.DARK_GRAY.getRGB());
                }else if(R < 150){
                        print.setRGB(ii, i, Color.WHITE.getRGB());
                }else if(R < 170){
                        print.setRGB(ii, i, Color.MAGENTA.getRGB());
                }else if(R < 190){
                        print.setRGB(ii, i, Color.BLACK.getRGB());
                }else if(R < 220){
                        print.setRGB(ii, i, Color.LIGHT_GRAY.getRGB());
                }else if(R < 240){
                        print.setRGB(ii, i, Color.PINK.getRGB());
                }else {
                        print.setRGB(ii, i, Color.ORANGE.getRGB());
                }
        }
}
```

Entenda que “print” é a imagem que está sendo tratada. As imagens a seguir são um exemplo do tratamento utilizado:

FIGURA 7 - ANTES DA APLICAÇÃO DO FILTRO NA IMAGEM 1.

FIGURA 8 - DEPOIS DA APLICAÇÃO DO FILTRO NA IMAGEM 1.

FIGURA 9 - ANTES DA APLICAÇÃO DO FILTRO NA IMAGEM 2.

FIGURA 10 - DEPOIS DA APLICAÇÃO DO FILTRO NA IMAGEM 2.

5 ANÁLISE DE RESULTADOS

A partir de testes realizados com o produto do projeto P-Finder, conseguiu-se coletar alguns dados importantes referentes à análise dos resultados obtidos. A quantidade de vezes testadas para cada variação de configuração foi de 100, obtendo assim uma apenas uma amostra do comportamento estimado do sistema. O hardware utilizado para realizar todos os teste foi um Notebook Acer Aspire 3102, com processador AMD Sempron 3200+ (1,6 GHz, 512KB de cache L2) e 2GB de memória RAM DDR2.

Primeiro verificamos o reflexo que o número de indivíduos por geração têm no percentual de acertos do AG. O teste foi realizado mantendo o número de gerações fixo em 200 e considerando os dados obtidos com os dois tipos de elitismo utilizados no projeto P-Finder, somente o melhor e com nota. Para o elitismo, manteve-se os parâmetros e alterou-se apenas o tipo de elitismo. Como pode-se observar pelo gráfico abaixo, o percentual de acerto aumenta conforme o número de indivíduos por geração cresce.

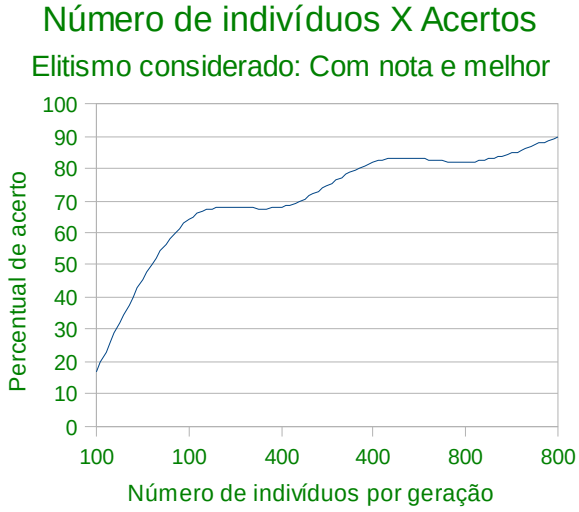

GRÁFICO 1 – NÚMERO DE INDIVÍDUOS x ACERTOS (COM NOTA E MELHOR).

Verificou-se também que o aumento no número de indivíduos por geração resulta em maior tempo de processamento para encontrar uma solução como pode

ser observado no gráfico abaixo, para este teste também foram considerados os dois tipos de elitismo utilizados no projeto P-Finder. O tempo está representado em milissegundos.

GRÁFICO 2 – NÚMERO DE INDIVÍDUOS x TEMPO.

Decidiu-se testar os dois tipos de elitismo utilizados, e pode-se notar que existem diferenças sutis entre eles na atual fitness, como podemos observar nos gráficos de número de habitantes pelo percentual de acertos divididos em cada tipo de elitismo, novamente, o número de gerações permaneceu fixo, o melhor resultado foi encontrado com o tipo de elitismo “com nota”, atingindo a marca de 96% de acerto contra 84% do tipo “somente o melhor”:

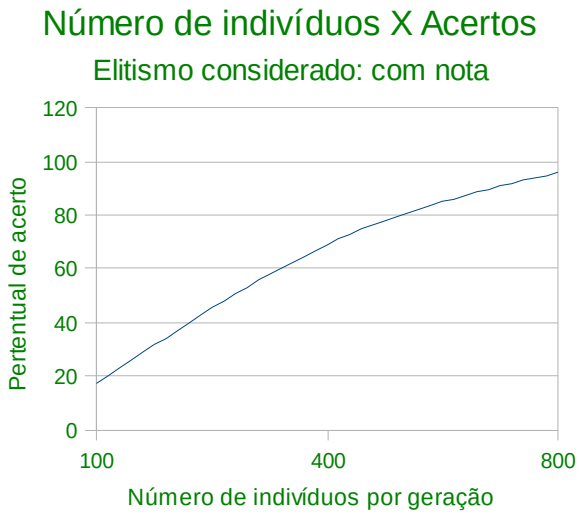

GRÁFICO 3 – NÚMERO DE INDIVÍDUOS x ACERTOS (COM NOTA).

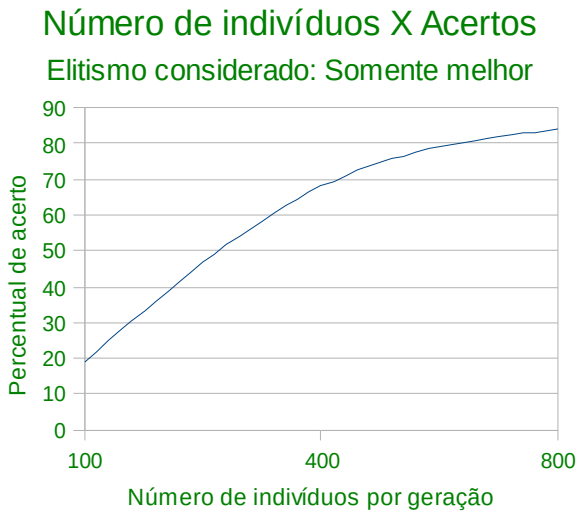

GRÁFICO 4 – NÚMERO DE INDIVÍDUOS x ACERTOS (SOMENTE MELHOR).

6 TRABALHOS RELACIONADOS - SISTEMA KAPTA

Como um programa relacionado, encontramos o sistema KAPTA. Trata-se de um sistema desenvolvido pela Universidade Federal do Rio de Janeiro (UFRJ) e que se divide em 2 módulos:

1º Captura da Imagem: Constitui o processo de digitalização, transformando-a em um padrão digital, o qual possa ser utilizado no processo de reconhecimento.

2º Reconhecimento de Caracteres: É dividido em quatro etapas:

a) Localização;
b) Segmentação;
c) Extração de características;
d) Reconhecimento propriamente dito.

6.1 KAPTA – LOCALIZAÇÃO DA PLACA

Tem por objetivo encontrar e segmentar a região que possua mais compatibilidade com o objeto procurado (no caso, a placa de um automóvel), utilizando-se de diversos filtros para tratar a imagem.

Podemos dividi-la nas seguintes etapas:

a) Redução do espaço de busca, utilizando-se de levantamentos estatísticos para descartar o que não é procurado.
b) Trabalha-se com o padrão de cor (geralmente o fundo cinza, com caracteres pretos sobrepostos). Utiliza-se um filtro nos padrões RGB, para realçar os contrastes, buscando realçar contrastes e altas freqüências existentes, começando pelas bordas dos caracteres e o contorno da placa.

Busca-se a região com maior probabilidade de serem correspondentes as bordas do caractere. Segundo o Departamento Nacional de Transito (DENATRAN), o tamanho médio dessas bordas é de 44 mm, sendo a letra I, com 10 mm o menor e os maiores contornos são os das letras A e M, com aproximadamente 54 mm.

c) O algoritmo procura por áreas com maior densidade de pontos de borda (conforme o DENATRAN, área retangular com 520cm^2). Se encontrada, a parte é recortada da imagem original e segue para a fase de segmentação. Caso o algoritmo nada encontre, é enviada uma imagem em branco.

7 O ESCOPO E O DESENVOLVIMENTO

O escopo inicial deste trabalho era:

Desenvolver uma ferramenta que auxilie no reconhecimento de placas automotivas em imagens, localizando a sua posição e podendo trabalhar em conjunto com um OCR externo. Utilizar Algoritmo Genético que deve ser implementado pela equipe podendo ter como base técnicas já utilizadas (para, por exemplo, para o cruzamento, seleção, DNA do Habitante).

Todos os requisitos foram atingidos de maneira satisfatória e novas estratégias foram percebidas durante o desenvolvimento, mas não foram implementadas por falta de tempo hábil, essas estratégias encontram-se no tópico referente às Sugestões de Trabalhos Futuros.

8 DIFICULDADES ENCONTRADAS

A maior dificuldade em se trabalhar com AG é desenvolver a parte "inteligente" situada na Fitness que é a responsável pela avaliação das características do Habitante. O computador não tem a mesma distinção que nós quando trata de uma imagem. Para ele uma imagem é somente um grupo de diversos dados, não havendo nenhuma interpretação se determinados pixels pertencem a uma placa, a um carro, ou a qualquer outra coisa. É necessário criar uma estratégia que seja capaz de identificar aquilo que se tem como objetivo e atribuir a devida nota da forma mais coerente possível. O tratamento de imagens em si já é algo complexo que demanda muito processamento computacional.

Inicialmente trabalhou-se com retângulos pretos em uma imagem de fundo branco. Com isso, alguns bugs do nosso AG foram corrigidos e diversas melhorias e correções de estrutura foram implementadas. A dificuldade maior foi a localização da placa em si, pois as imagens de automóveis possuem mais informações e maior diversidade de cores do que uma simples imagem em branco com um retângulo preto. Tentou-se diversas estratégias como, por exemplo, fixar a busca por uma cor, que seria a cor de fundo da placa, mas funcionava apenas para uma única imagem, pois em outras as cores eram muito divergentes devido à iluminação e à própria captura da câmera e assim a cor do fundo da placa era diferente em cada imagem. Outra tentativa foi procurar por uma quina da placa, analisando a variância de tons, mas a complexidade e diferença das imagens comprometia os resultados. Para corrigir essa diferença de cores tentou-se criar um filtro de modo que a cor de fundo da placa na maioria das imagens se tornasse homogênea mesmo sendo uma cor diferente em cada imagem. Finalmente, encontrou-se uma estratégia que era flexível quanto à cor da placa apresentando resultados satisfatórios em diversas imagens, tornando-se capaz de encontrar placas de cores diferentes. Não foi encontrada nenhuma procedência para a solução implementada.

Foi necessário despender bastante tempo de estudo sobre tratamento de imagens e AG. A falta de material em português mostrou-se um problema, obrigando a equipe a buscar materiais em inglês em diversos livros que continham textos com linguagem técnica, dificultando a compreensão e alguns não continham a informação aprofundada o suficiente.

9 SUGESTÕES PARA TRABALHOS FUTUROS

Esta aplicação pode ser utilizada em integração com um software OCR externo já existente ou outro que, futuramente, possa ser criado em plataforma Java. Este conjunto (P-FINDER e OCR) pode-se aplicar as seguintes funções:

a) Integração com um banco de dados para o armazenamento de placas automotivas, liberando a entrada e saída somente dos carros identificados e autorizados pelo proprietário de determinado local.
b) Utilização em radares, em conjunto com máquina fotográfica, para carros que ultrapassem a velocidade máxima permitida em rodovias.
c) Uso em estacionamentos, condomínios residenciais e comerciais que necessitem de um controle dos automóveis que passam pelo local.

Se adaptado, é possível que algoritmo faça a localização de outros tipos de placa ou objetos em uma figura.

Outra sugestão seria a geração de um arquivo em XML para a demonstração dos resultados obtidos pelo algoritmo genético no processo de localização.

10 CONCLUSÃO

O P-Finder é uma ferramenta de localização de placas automotivas em imagens que visa auxiliar no reconhecimento das mesmas por algum programa de OCR.

Um grande desafio foi enfrentado pela equipe: fazer um computador ter um discernimento similar ao humano a ponto de, em uma imagem, poder encontrar a placa de um carro. Não é fácil realizar este tipo de implementação e o resultado que se chegou foi com certeza muito inferior comparado à grande capacidade da nossa mente, mas ainda assim a equipe se felicitou bastante com os resultados, objetivos que foram alcançados e as barreiras que foram vencidas. Aprendeu-se bastante com todos os estudos relacionados às tecnologias utilizadas e pôde-se ver as dificuldades em cada etapa: planejamento, criação do AG, implementação de filtragem flexível, criação e calibragem do conjunto estratégico inteligente, testes e em paralelo a documentação. Além disso, houve um grande aprendizado não somente didático, técnico, como também humano, pois se trabalhou durante quase um ano, vencendo certos obstáculos e encontrando outros, dedicando nosso tempo em reuniões e debates, discutindo tarefas e modos de fazê-las, cobrando e sendo cobrado.

É importante fazer as coisas acontecerem em qualquer atividade que se esteja trabalhando. Nossas realizações fazem parte de nossa história e nos fazem mais completos, pois contribuem positivamente para nós mesmos e para aqueles que compartilham dos resultados alcançados.

REFERÊNCIAS

ANSELMO, F. **Aplicando Lógica Orientada a Objetos em Java.** Florianópolis: Visual Books, 2003.

ALGORTMOS GENÉTICOS: Fundamentos e Aplicações. Disponível em:

<http://www.gta.ufrj.br/~marcio/genetic.html>. Acesso em: 21/11/2008.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML guia do usuário.** Tradução de: SILVA, Fábio Freitas da. Rio de Janeiro: Campus, 2000.

CARVALHO, A. P. de L. F. de. **Algoritmos Genéticos.** Disponível em:<http://www.icmc.usp.br/~andre/research/genetic/index.htm> Acesso em: 31/10/2008.

CHARLES, D.; FYFE, C.; LIVINGSTONE, D. ; MCGLINCHEY, S. **Biologically inspired artificial intelligence for computer games.** New York: IGI Global, 2008.

FACON, J. **Processamento e Análise de Imagens**. Disponível em: <http://www.ppgia.pucpr.br/~facon/MaterialGraduacao2005/ApostilaProcImagem.pdf >. Acesso em: 25/11/2008.

FERNANDES, A. M. da R.. **Inteligência Artificial:** Noções Gerais. Florianópolis: Visual Books, 2005.

GUIA DO HARDWARE. Disponível em:

<http://www.guiadohardware.net/termos/ocr>. Acesso em: 30/10/2008.

GUIGO, B. C.; STIEBLER, G. M.; THOMÉ, A. C. G. **KAPTA:** Um sistema de reconhecimento automático de placas de veículos baseado nas técnicas de redes neurais e processamento de imagens. Disponível em: <http://www.labic.nce.ufrj.br/downloads/sucesu_2004.pdf> Acesso em: 29/09/2008.

JAVAFREE. Disponível em: <http://www.javafree.org/content/view.jf?idContent=213>.

Acesso em: 30/10/2008.

JH LABS. **Java Image Processing**. Disponível em: <http://www.jhlabs.com/ip/filters/index.html>. Acesso em: 19/11/2008.

JONES, M. T. **Artificial Intelligence**. A Systems Approach. Massachusetts: Infinity Science Press Llc, 2008.

LOZANO, F. **Eclipse No Mundo Open Source**. Disponível em: <http://www.lozano.eti.br/palestras/eclipse-oss.pdf> . Acesso em: 29/09/2008.

LOZANO, F. **Netbeans X Eclipse**: A Evolução dos IDEs A Evolução dos IDEs Livres para Java . Disponível em: <http://www.lozano.eti.br/palestras/netbeans-vs-eclipse.pdf>. Acesso em: 29/09/2008.

LUCAS, D. C. **Algoritmos Genéticos:** um estudo de seus conceitos fundamentais e aplicação no problema de grade horária. Disponível em: <www.ufpel.edu.br/prg/sisbi/bibct/acervo/info/2000/Mono-Diogo.pdf > Acesso em:

MARTIN, F. **Uml essencial**: um breve guia para a linguagem-padrao de modelagem de objetos. Tradução de: TORTELLO, João. 3ed. Porto Alegre: Bookman, 2005.

MELO, A. C. **Desenvolvendo aplicações com UML 2.0**: do conceitual à implementação. 2 ed. Rio de Janeiro: Brasport, 2004.

MENDONÇA, V. G. de; POZZER, C. T. ; RAITTZ, R. T. **A Framework for Genetic Algorithms in Games.** SBC - Proceedings of SBGames'08: Computing Track - Technical Posters. Belo Horizonte, Novembro, 2008. Disponível em: <http://www.inf.pucminas.br/sbgames08/EBooks/Proceedings-SBGames-Posters-2008-Final-EB.pdf>. Acesso em: 18/11/2008.

METSKER, S. J. **Padrões de Projetos em Java.** Porto Alegre: Bookman, 2004.

MIRANDA, M. N. **Algoritmos Genéticos:** Fundamentos e Aplicações. Disponível em: <http://www.gta.ufrj.br/~marcio/genetic.html> . Acesso em: 31/10/2008.

MONTEIRO, A. **Certificação PMP**: Otimize seu tempo de estudo na preparação para a prova de Certificação PMP: Concentre seus estudos nos tópicos mais cobrados nas provas de certificação: Questões resolvidas e comentadas. 2. ed. rev. Rio de Janeiro: Brasport, 2008.

OCR. Disponível em: <http://pt.wikipedia.org/wiki/OCR>. Acesso em: 29/09/2008.

OCR. **Termos técnicos GdH**. Disponível em: <http://www.guiadohardware.net/termos/ocr>. Acesso em: 29/09/2008.

OLIVIERA, M. H. de. E.; STEFFEN JUNIOR, V. **Estudo de otimização e casos utilizando Algoritmos Genéticos e Recozimento Simulado.** Disponível em: <http://www.famat.ufu.br/revistaset2004/artigos/ArtigoMartaValder.pdf>. Acesso em: 22/11/2008.

ROMAN, N. T. **Computação Evolutiva:** Algoritmos Genéticos & Programação Genética. Disponível em: <http://www.nortonroman.info/arquivos/mc906/aula5.pdf> Acesso em: 24/11/2008.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence**. A Modern Approach. New Jersey: Prentice-Hall, 1995.

SCHWAB, B. **AI Game Engine Programming**. Massachusetts: Charles River Media, 2004.

Software TopOCR Version 3.1 for Windows. Disponível em: <http://www.topocr.com/topocr.exe>. Acesso em: 21/11/2008.

UNIVERSIDADE ESTADUAL DO RIO DE JANEIRO (UERJ). Laboratório de Geoprocessamento do Departamento de Geologia **(**Labgis). **Realce de Imagem**. Disponível em: <http://www.fgel.uerj.br/labgis/gis_atualizada/pdi/04_realce.htm > (2007) Acesso em: 29/11/2008.

DOCUMENTOS CONSULTADOS

UNIVERSIDADE FEDERAL DO PARANÁ. Sistema de Bibliotecas. **Teses, dissertações, monografias e trabalhos acadêmicos.** Curitiba: Editora UFPR, 2007. (Normas para apresentação de documentos científicos, 2).

UNIVERSIDADE FEDERAL DO PARANÁ. Sistema de Bibliotecas. **Citações e Notas de Rodapé.** Curitiba: Editora UFPR, 2007. (Normas para apresentação de documentos científicos, 3).

UNIVERSIDADE FEDERAL DO PARANÁ. Sistema de Bibliotecas. **Referências.** Curitiba: Editora UFPR, 2007. (Normas para apresentação de documentos científicos, 4).

APÊNDICES

APÊNDICE A: PLANO DE PROJETO

1 TERMO DE ABERTURA DO PROJETO

1.1 DECLARAÇÃO DO PROBLEMA

Necessidade de desenvolver tecnologia para reconhecimento de placas automotivas a partir de uma imagem digitalizada, para uso em outros projetos.

1.2 OBJETIVOS DO PROJETO

Encontrar uma solução para reconhecimento de placas de automóveis em imagens digitalizadas utilizando algoritmos genéticos. Pesquisar softwares OCR disponíveis no mercado que consigam reconhecer os caracteres contidos em uma imagem de placa.

1.3 REQUISITOS DO PRODUTO DO PROJETO

Localizar automaticamente placas de automóveis em imagens digitalizadas, para facilitar a leitura dos caracteres contidos por softwares OCR. Ser portável para se utilizar em projetos futuros facilmente.

1.4 STAKEHOLDERS (PARTES INTERESSADAS)

Identificação dos stakeholders:

h) Universidade Federal do Paraná (UFPR) setor Escola Técnica: Instituição cliente e patrocinadora do projeto;

i) Roberto Tadeu Raittz: Orientador do projeto e representante oficial da Instituição cliente (UFPR) para este projeto;

j) Vinícius Godoy de Mendonça: Co-orientador do projeto;

k) Time de projeto, composto por: Aryel Marlus Repula de Oliveira, Carla Maria da Silva Cassemiro, Mauro da Silva Pinto.

1.5 PREMISSAS

a) Professor Orientador Senhor Doutor Roberto Tadeu Raittz durante todo o período de desenvolvimento do projeto, oferecendo auxílio e orientação no que diz respeito à inteligência artificial;

b) Professor Co-Orientador Senhor Vinícius Godoy de Mendonça durante todo o período de desenvolvimento do projeto, oferecendo auxílio e orientação no que diz respeito a padrões de codificação, gerência do projeto, documentação, modelagem do sistema e tratamento de imagens.

1.6 RESTRIÇÕES

Projeto deve ser concluído até 09/12/2008.

2 DECLARAÇÃO DO ESCOPO

2.1 DECLARAÇÃO DO ESCOPO DO PRODUTO

a) Permite ao usuário enviar uma imagem para localizar automaticamente a placa de um automóvel;

b) Apresentação em tela dos resultados encontrados para a busca da placa;

c) Permitir seleção de posições específicas na imagem para se verificar qual nota recebe do algoritmo genético;

d) Permitir criar cada geração manualmente para observar a convergência do algoritmo genético, os melhores indivíduos de cada geração devem ser apresentados em tela;

e) Permitir que se determine manualmente os principais parâmetros utilizados pelo algoritmo genético para testes.

2.2 DECLARAÇÃO DO ESCOPO DO PROJETO

a) Desenvolver um algoritmo genético;

b) Determinar uma estratégia para a Fitness do algoritmo genético que seja capaz de localizar placas automotivas em imagens digitalizadas;

c) Pesquisar e testar softwares OCR (Optical Character Recognition) que consigam reconhecer os caracteres em uma imagem de placa automotiva;

d) Pesquisar sobre soluções já implementadas sobre reconhecimento de placas automotivas;

e) Documentar o projeto e o código.

3 ESTRUTURA ANALÍTICA DO PROJETO (EAP)

A Estrutura Analítica do Projeto (EAP) ou Work Breakdown Structure (WBS) define todas as atividades que são necessárias para a realização do projeto, sendo também uma forma de visualizar claramente o escopo total do projeto (o que não está relacionado na EAP não faz parte do escopo do projeto), além de ser uma ferramenta de comunicação entre os stakeholders e de integração da equipe do projeto. Pode ser entendida com a definição de Monteiro (2008, p. 57):

> É um agrupamento dos componentes do projeto orientado aos Entregáveis, que organiza e define o escopo total do projeto. O trabalho que será feito no projeto é exatamente aquele que está representado na EAP (WBS). O que não está descrito na EAP está fora do escopo.

A EAP para o projeto P-Finder pode ser visualizada a seguir:

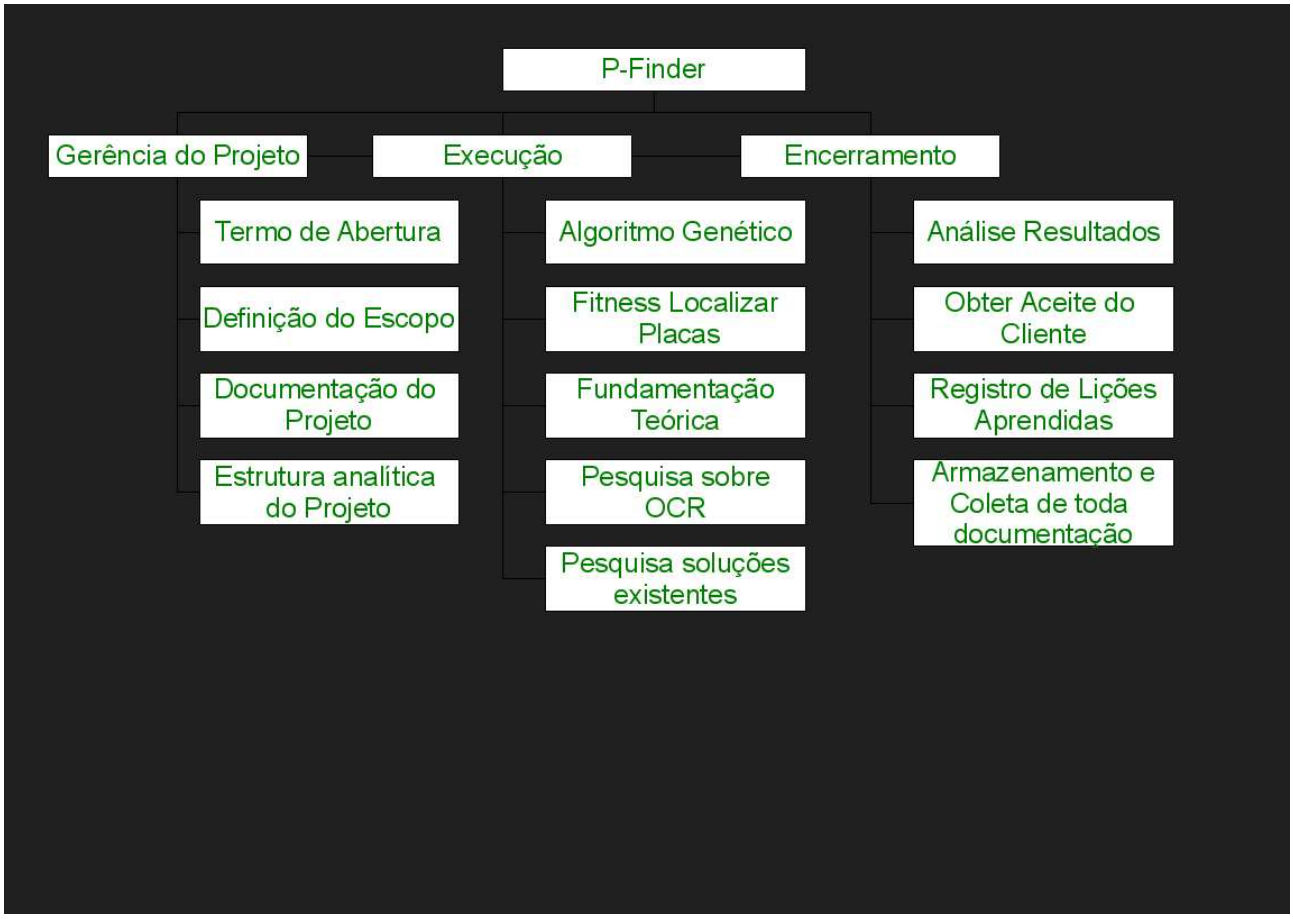

FIGURA 11 – EAP DO PROJETO P-FINDER.

APÊNDICE B - CASOS DE USO

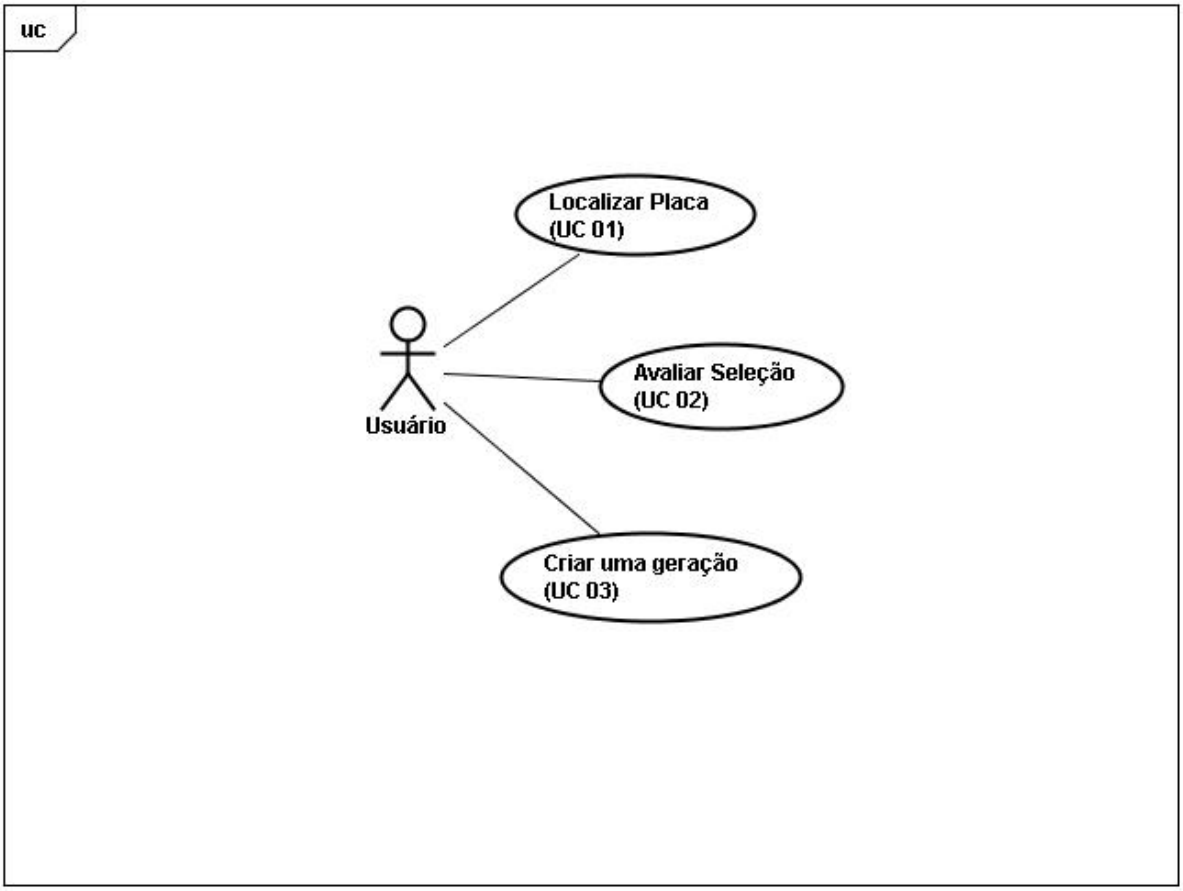

FIGURA 12 - DIAGRAMA DE CASO DE USO GERAL.

UC 01 - Caso de uso localizar placa.

Nível do objetivo: Nível do mar.

Descrição: Este caso de uso serve para localizar a placa em uma imagem.

Pré-condições: Parâmetros ajustados (UC 05); seleciona uma imagem (UC 04).

Pós-condições de sucesso: O sistema apresenta em destaque na imagem selecionada pelo usuário a localização da placa encontrada pelo sistema.

Ator primário: Usuário.

Cenário principal de sucesso:

1. Usuário solicita ao sistema para localizar placa na imagem selecionada;
2. Sistema encontra a localização da placa na imagem selecionada;
3. Sistema apresenta a imagem selecionada pelo usuário com a localização encontrada para a placa em destaque.

Extensões:

3a: Sistema não encontra uma possível localização para a placa na imagem selecionada.

.1: Sistema emite aviso ao usuário informando que não foi possível encontrar a placa e retorna à tela principal, caso de uso finaliza.

UC 02 - Caso de uso avaliar seleção.

Nível do objetivo: Nível do mar.

Descrição: Este caso de uso serve para avaliar uma seleção na imagem feita pelo usuário.

Pré-condições: seleciona uma imagem (UC 04); desenha um retângulo na imagem (UC 06).

Pós-condições de sucesso: O sistema exibe as informações sobre a seleção feita pelo usuário (coordenadas do ponto referente ao início do retângulo, largura e altura do retângulo e nota).

Ator primário: Usuário.

Cenário principal de sucesso:

c)Usuário solicita ao sistema para avaliar o desenho feito na imagem;

d) Sistema avalia o desenho;

e) Sistema apresenta informações referentes ao desenho feito pelo usuário (coordenadas do ponto de início do retângulo, largura e altura do retângulo e nota).

UC 03 – Criar geração.

Nível do objetivo: Nível do mar.

Descrição: Este caso de uso serve para criar uma e apenas uma geração de indivíduos do genético baseada nos parâmetros definidos.

Pré-condições: Parâmetros ajustados (UC 05); seleciona uma imagem (UC 04).

Pós-condições de sucesso: O sistema apresenta em destaque na imagem selecionada pelo usuário a localização do melhor indivíduo da geração criada.

Ator primário: Usuário.

Cenário principal de sucesso:

1. Usuário solicita ao sistema para criar uma nova geração do genético;
2. Sistema cria uma nova geração de indivíduos baseada nos parâmetros definidos;
3. Sistema apresenta em destaque na imagem selecionada pelo usuário a localização do melhor indivíduo da geração criada.

UC 04 – Selecionar imagem

Nível do objetivo: Nível de peixe.

Descrição: Este caso de uso serve para selecionar uma imagem.

Pré-condições:

Pós-condições de sucesso: O sistema apresenta na tela principal a imagem selecionada.

Ator primário: Usuário.

Cenário principal de sucesso:

f) Usuário solicita ao sistema para abrir uma imagem;

g) Sistema apresenta diretórios do computador e arquivos com a extensão JPG;

h) Usuário navega nos diretórios apresentados e seleciona uma imagem;

i) Sistema carrega a imagem na tela principal.

Extensões:

3a: Usuário solicita ao sistema para cancelar escolha da imagem.

.1: Sistema remove a apresentação dos arquivos e diretórios e retorna à tela principal, caso de uso finaliza.

3b: Resolução da imagem selecionada é inferior a 300x300 pixels.

.1: Sistema emite aviso ao usuário informando que é recomendável a imagem possuir resolução de pelo menos 300x300 pixels.

.2: Caso de uso retorna no passo 4 do Cenário principal de sucesso.

3c: Resolução da imagem selecionada é superior à 1200x1024 pixels.

1: Sistema emite aviso ao usuário informando que é recomendável a imagem não ser superior à 1200x1024 pixels;

.2: Caso de uso retorna no passo 4 do Cenário principal de sucesso.

UC 05 – Ajustar parâmetros.

Nível do objetivo: Nível de peixe.

Descrição: Este caso de uso serve para ajustar os parâmetros do algoritmo genético.

Pré-condições:

Pós-condições de sucesso: O sistema configura o algoritmo genético para utilizar parâmetros conforme definidos pelo usuário.

Ator primário: Usuário.

Cenário principal de sucesso:

f) Usuário solicita ao sistema para setar os parâmetros;

g) Sistema exibe os parâmetros utilizados pelo algoritmo genético;

h) Usuário ajusta os parâmetros: número de habitantes, número de gerações, chance de mutação, nota de pertinência, forma de elitismo; conforme deseja, e define a largura máxima e mínima da placa (obs. extra: a largura não deve ser inferior à real da placa na imagem, nem muito superior para melhores resultados);

i) Sistema configura o algoritmo genético para utilizar os parâmetros que foram definidos pelo usuário;

j) Sistema volta à tela principal.

Extensões:

3a: Usuário define largura máxima da placa inferior à mínima.

.1: Sistema emite aviso ao usuário informando que a largura máxima da placa não pode ser superior à mínima;

.2: Caso de uso retorna ao Cenário Principal de Sucesso no passo 2.

3b: Usuário fecha a janela de exibição dos parâmetros.

.1: Sistema mantém os parâmetros atuais do genético sem qualquer alteração. Caso de uso finaliza.

UC 06 – Desenhar retângulo na imagem.

Nível do objetivo: Nível de peixe.

Descrição: Este caso de uso serve para desenhar um retângulo na imagem selecionada.

Pré-condições: Selecionar imagem (UC 04).

Pós-condições de sucesso: O retângulo desenhado é apresentado na imagem.

Ator primário: Usuário.

Cenário principal de sucesso:

1.Usuário pressiona o botão do mouse sobre a imagem e o move para baixo e direita até a posição desejada e solta o botão;

2.Sistema desenha um retângulo na imagem, a partir do primeiro clique do mouse até onde o usuário soltou o botão;

Extensões:

1a: Usuário desenha um retângulo com largura inferior a 50 pixels.

.1: Sistema emite aviso ao usuário informando que a largura deve ser de pelo menos 50 pixels;

.2: Sistema retorna à tela inicial, caso de uso finaliza.

1b: Usuário desenha um retângulo com altura inferior a 15 pixels.

1.: Sistema emite aviso ao usuário informando que a altura deve ser de pelo menos 15 pixels;

.2: Sistema retorna à tela inicial, caso de uso finaliza.

APÊNDICE C – DIAGRAMAS DE CLASSE

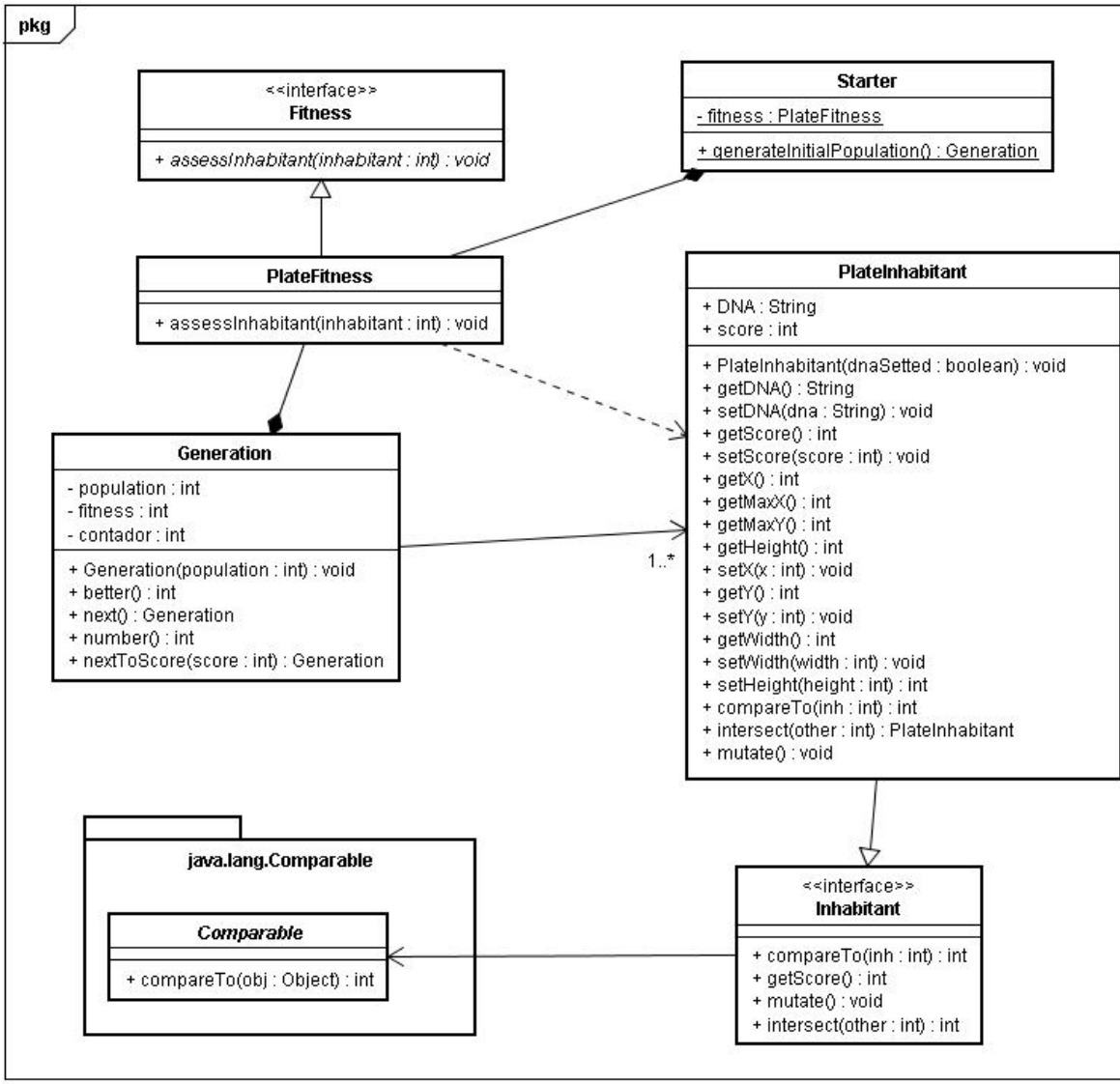

FIGURA 13 – DIAGRAMA DE CLASSE GERAL.

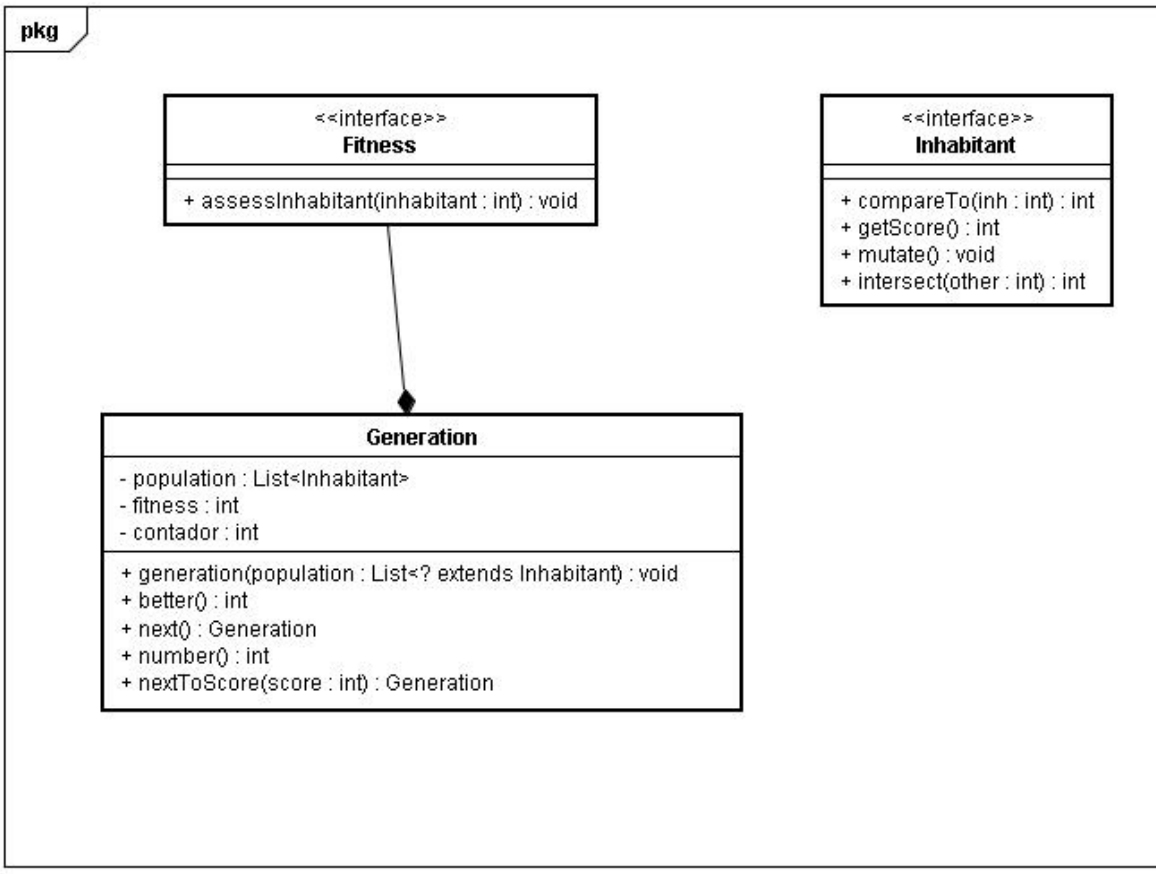

FIGURA 14 – DIAGRAMA DE CLASSE DO ALGORITMO GENÉTICO.

APÊNDICE D - DIAGRAMAS DE SEQUÊNCIA

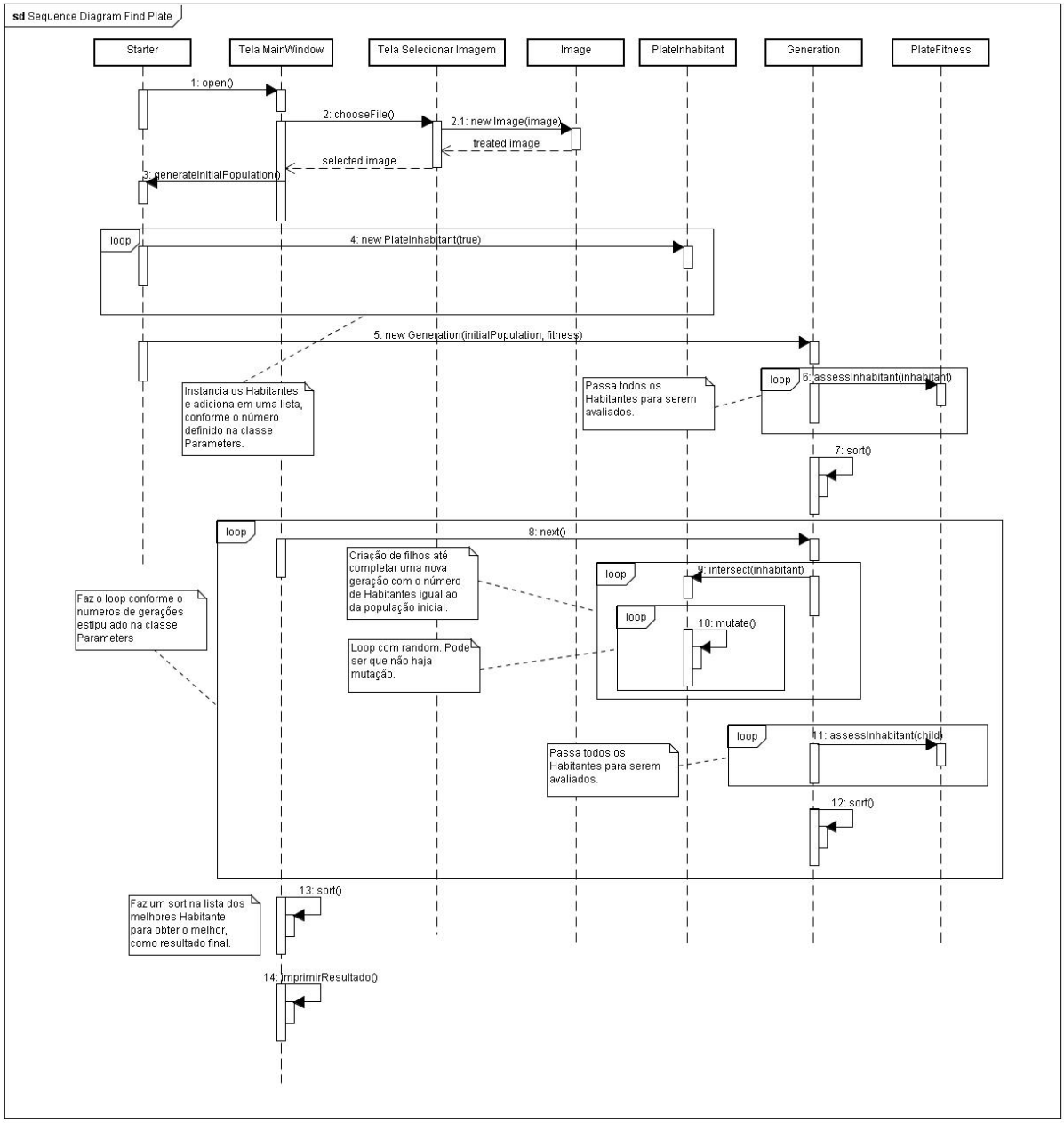

FIGURA 15 - DIAGRAMA DE SEQÜÊNCIA ENCONTRAR PLACA.

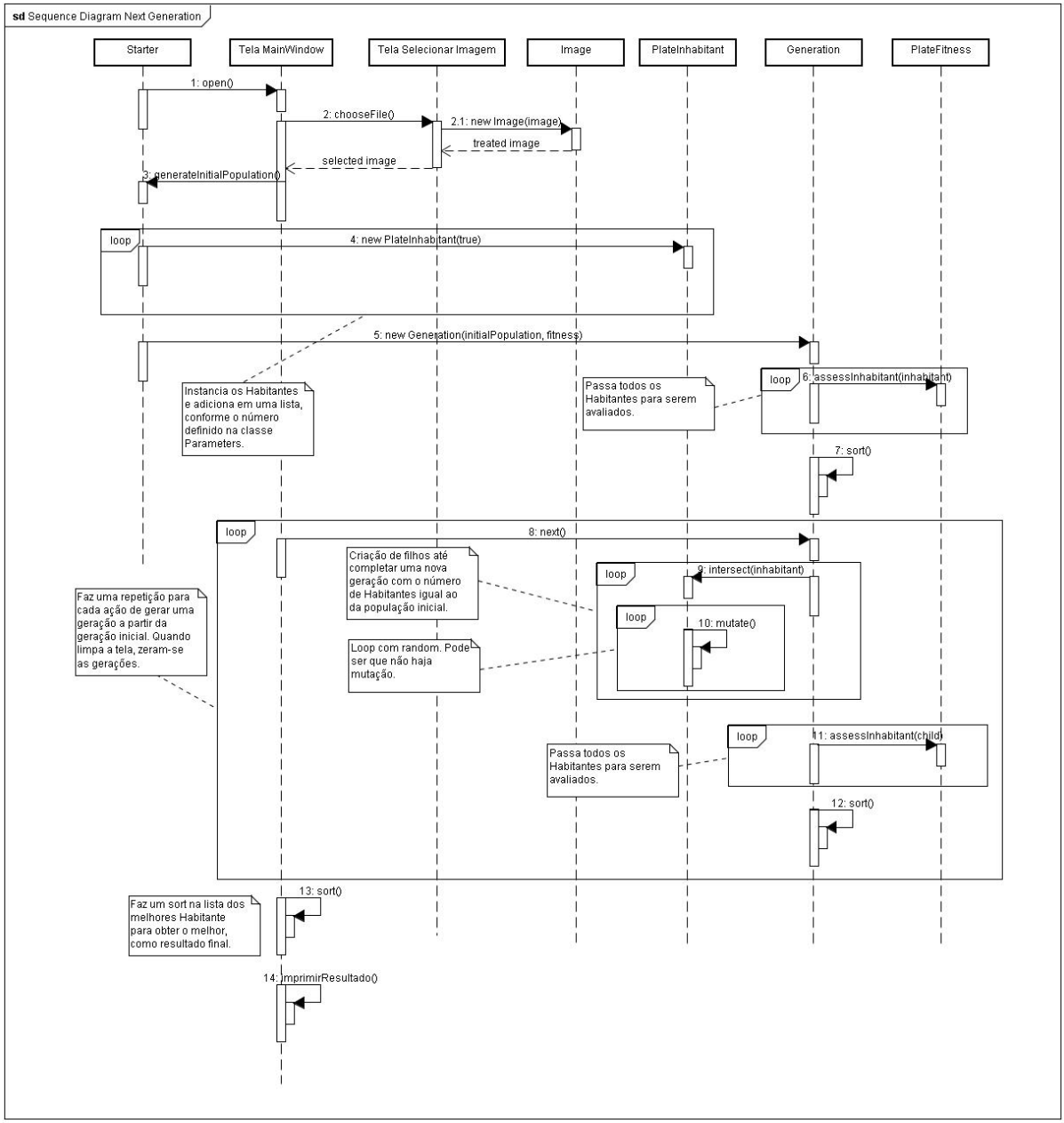

FIGURA 16 - DIAGRAMA DE SEQÜÊNCIA CRIAR UMA GERAÇÃO.

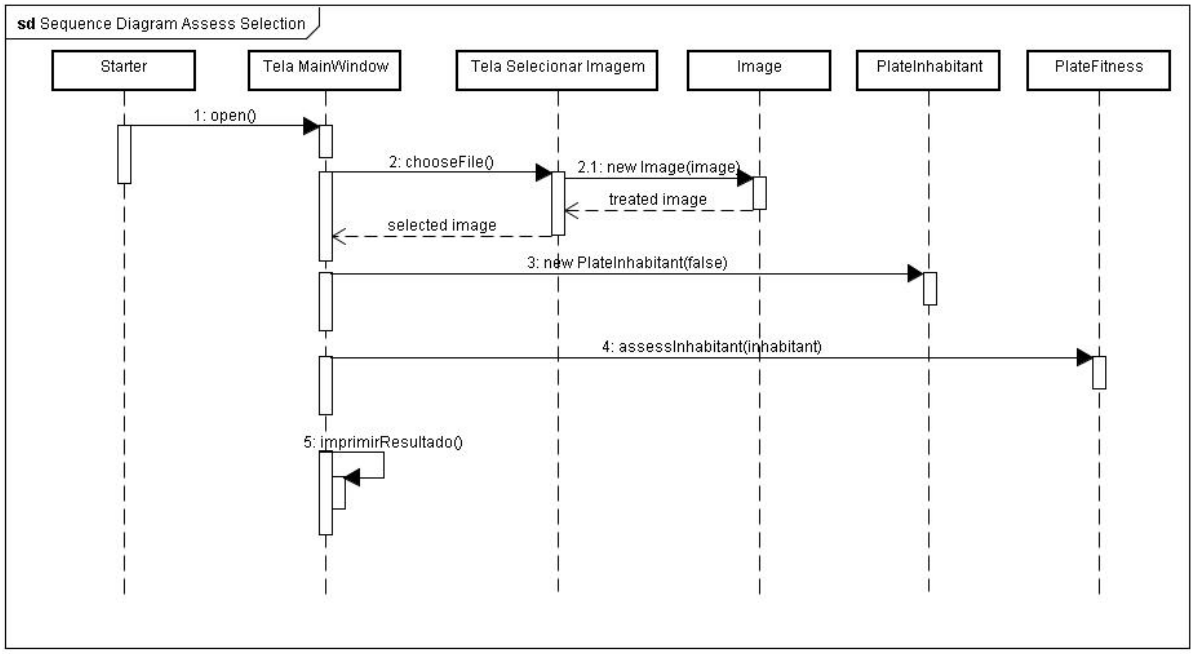

FIGURA 17 - DIAGRAMA DE SEQÜÊNCIA AVALIAR SELEÇÃO.

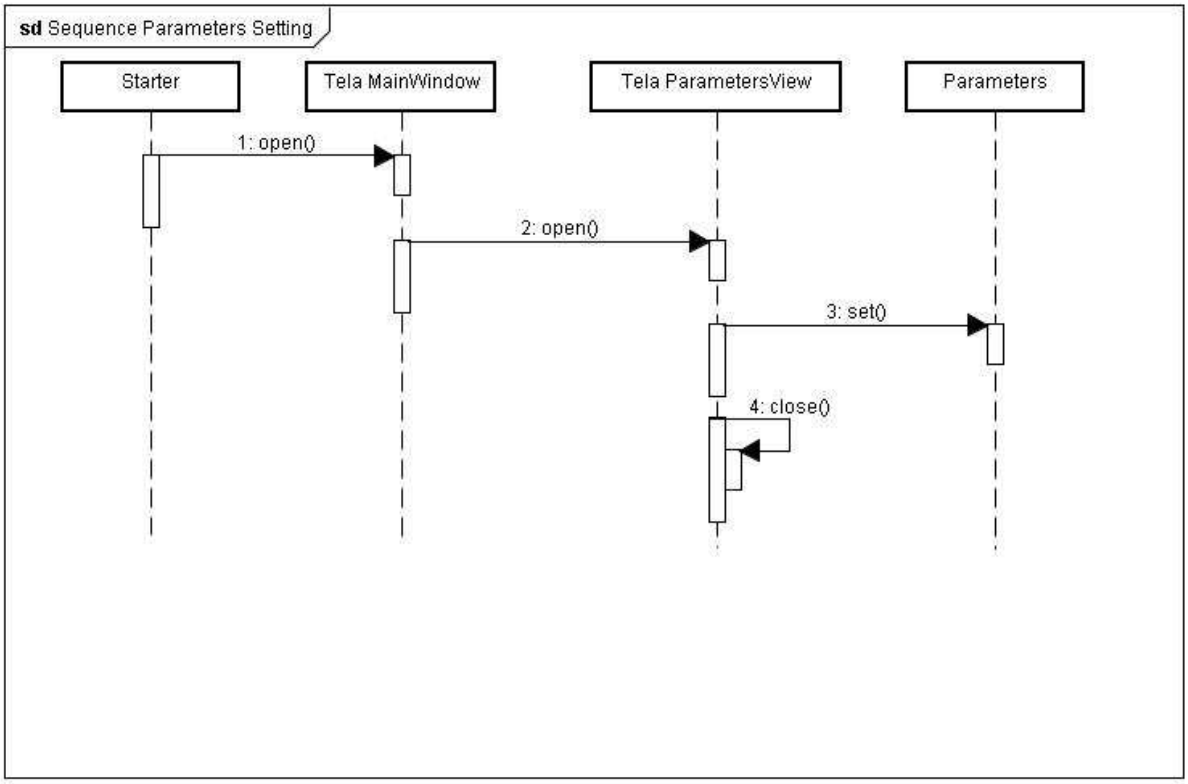

FIGURA 18 - DIAGRAMA DE SEQÜÊNCIA SETAR PARÂMETROS.

APÊNDICE E - TELAS

1 TELA PRINCIPAL

É apresentada quando o sistema é iniciado. Possui os botões: Abrir Imagem; Encontrar Placa; Avaliar Seleção; Limpar; Criar uma geração; Setar Parâmetros.

FIGURA 19 - TELA PRINCIPAL.

2 TELA ABRIR IMAGEM

É apresentada quando o usuário clica no botão “Abrir Imagem”. É possível navegar entre os diretórios do computador e os arquivos com extensão JPG são exibidos com possibilidade de seleção. Possui os botões: Up One Level; Home; Create New Folder; List; Details; Ok; Cancel. Possui as caixas de seleção: Look In; Files of Type. Possui a entrada de texto: File Name.

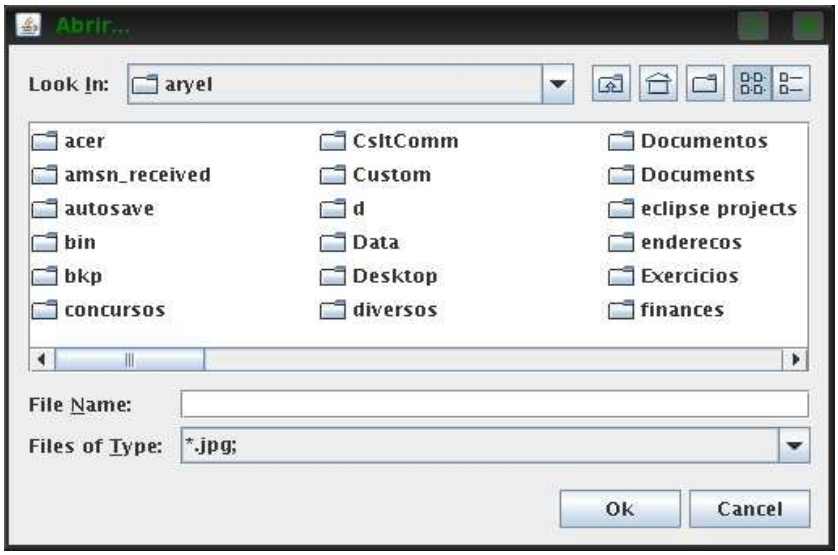

FIGURA 20 - TELA ABRIR IMAGEM.

3 TELA PRINCIPAL EXIBE IMAGEM

É apresentada quando o usuário seleciona uma imagem e clica no botão “Ok” na tela “Abrir Imagem” (Figura 20). Será exibida de maneira semelhante ao exemplo da Figura 21, com a figura escolhida pelo usuário em exibição. Possui os botões: Abrir Imagem; Encontrar Placa; Avaliar Seleção; Limpar; Criar uma geração; Setar Parâmetros.

FIGURA 21 - TELA PRINCIPAL COM IMAGEM SELECIONADA.

4 TELA SETAR PARÂMETROS

É apresentada quando o usuário clica no botão “Setar Parâmetros” na “Tela Principal”. Possui os Sliders: Número de Habitantes; Número de Gerações; Chance de Mutação; Pertinência (nota); Largura Máx da Placa; Largura Mín da Placa. Possui a caixa de seleção: Elitismo.

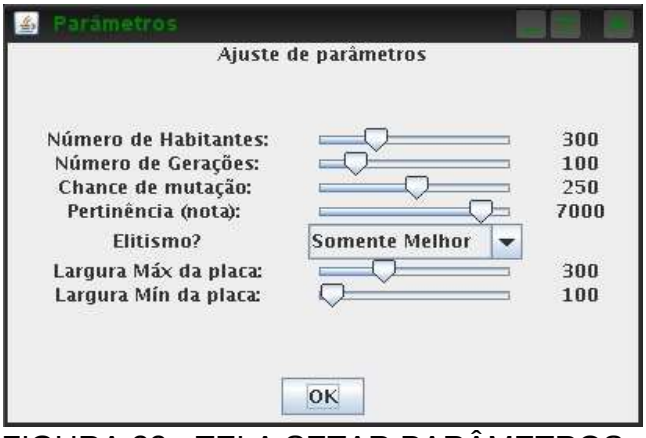

FIGURA 22 - TELA SETAR PARÂMETROS.

5 TELA NECESSÁRIO ABRIR IMAGEM

É apresentada quando o usuário clica em algum dos seguintes botões da tela principal sem previamente selecionar uma imagem: Encontrar Placa; Avaliar Seleção; Limpar; Criar uma Geração. Possui o botão: Ok.

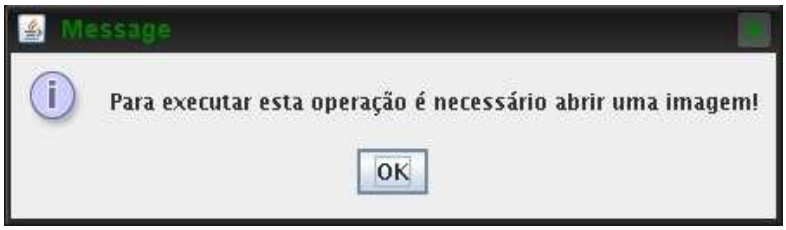

FIGURA 23 - TELA NECESSÁRIO ABRIR IMAGEM.

6 TELA LARGURA MÁXIMA MENOR QUE A MÍNIMA

É apresentada quando o usuário define a largura máxima menor que a largura mínima da placa (Como no exemplo da Figura 25) na tela “Setar Parâmetros” e clica no botão “Ok”. Possui o botão: Ok.

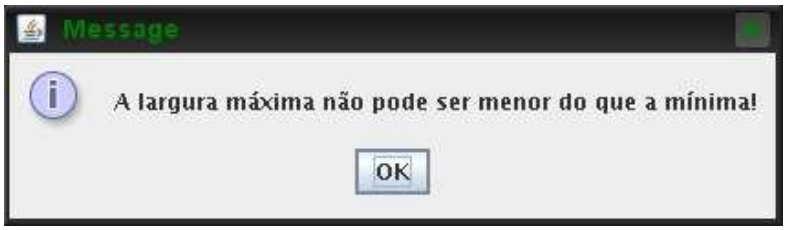

FIGURA 24 - TELA LARGURA MÁXIMA MENOR QUE A MÍNIMA.

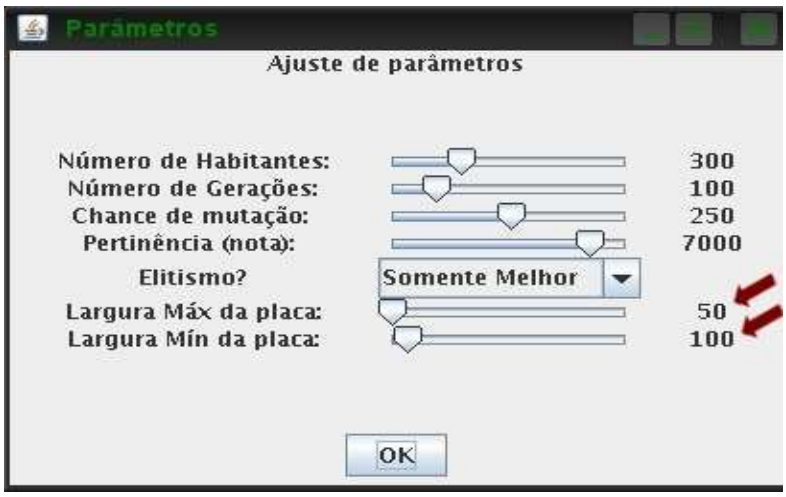

FIGURA 25 - EXEMPLO SETAR LARGURA MÁXIMA MENOR QUE MÍNIMA.

7 TELA IMAGEM MENOR QUE 300X300 PIXELS

É apresentada quando o usuário seleciona uma imagem com resolução menor que 300x300 pixels na tela “Abrir Imagem” e clica no botão “Ok”. Possui o botão: Ok.

FIGURA 26 - IMAGEM MENOR QUE 300X300 PIXELS.

8 TELA IMAGEM MAIOR QUE 1200X1024

É apresentada quando o usuário seleciona uma imagem com resolução maior que 1200x1024 pixels e na tela “Abrir Imagem” e clica no botão “Ok”. Possui o botão: Ok.

FIGURA 27 - TELA IMAGEM MAIOR QUE 1200X1024 PIXELS.

9 TELA PLACA NÃO ENCONTRADA

É apresentada quando o usuário clica no botão “Encontrar Placa” na tela “Tela Principal com Imagem Selecionada” e o sistema não encontra nenhuma possível placa na imagem definida. Possui o botão: Ok.

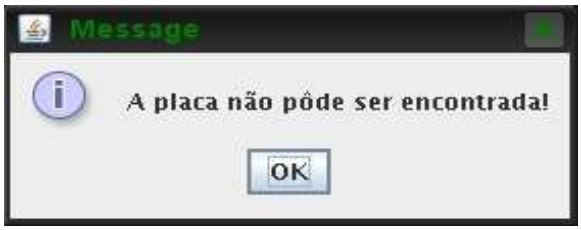

FIGURA 28 - PLACA NÃO ENCONTRADA.

10 TELA LARGURA SELEÇÃO COM MENOS DE 50 PIXELS

É apresentada quando o retângulo desenhado pelo usuário na imagem da “Tela Principal com Imagem Selecionada” tem largura inferior a 50 pixels. Possui o botão: Ok.

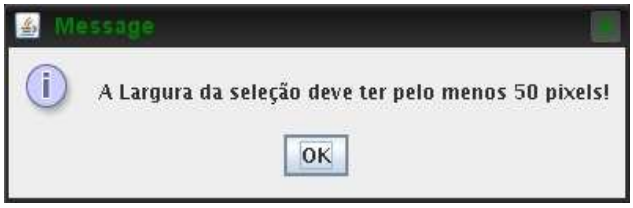

FIGURA 29 - TELA LARGURA SELEÇÃO INFERIOR A 50 PIXELS.

11 TELA ALTURA SELEÇÃO COM MENOS DE 15 PIXELS

É apresentada quando o retângulo desenhado pelo usuário na imagem da “Tela Principal com Imagem Selecionada” tem altura inferior a 15 pixels. Possui o botão: Ok.

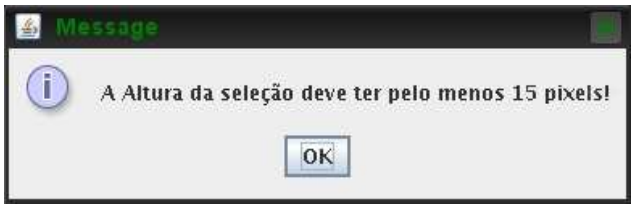

FIGURA 30 - ALTURA SELEÇÃO INFERIOR A 15 PIXELS.

12 TELA ENCONTRAR PLACA

É apresentada quando o usuário clica no botão “Encontrar Placa” na tela “Tela Principal com Imagem Selecionada”, apresenta a imagem definida pelo usuário com o local encontrado pelo sistema para a placa em destaque vermelho e os melhores indivíduos das últimas gerações em destaque azul, como observado no exemplo da figura 6.11. Possui os botões: Abrir Imagem; Encontrar Placa; Avaliar Seleção; Limpar; Criar uma geração; Setar Parâmetros.

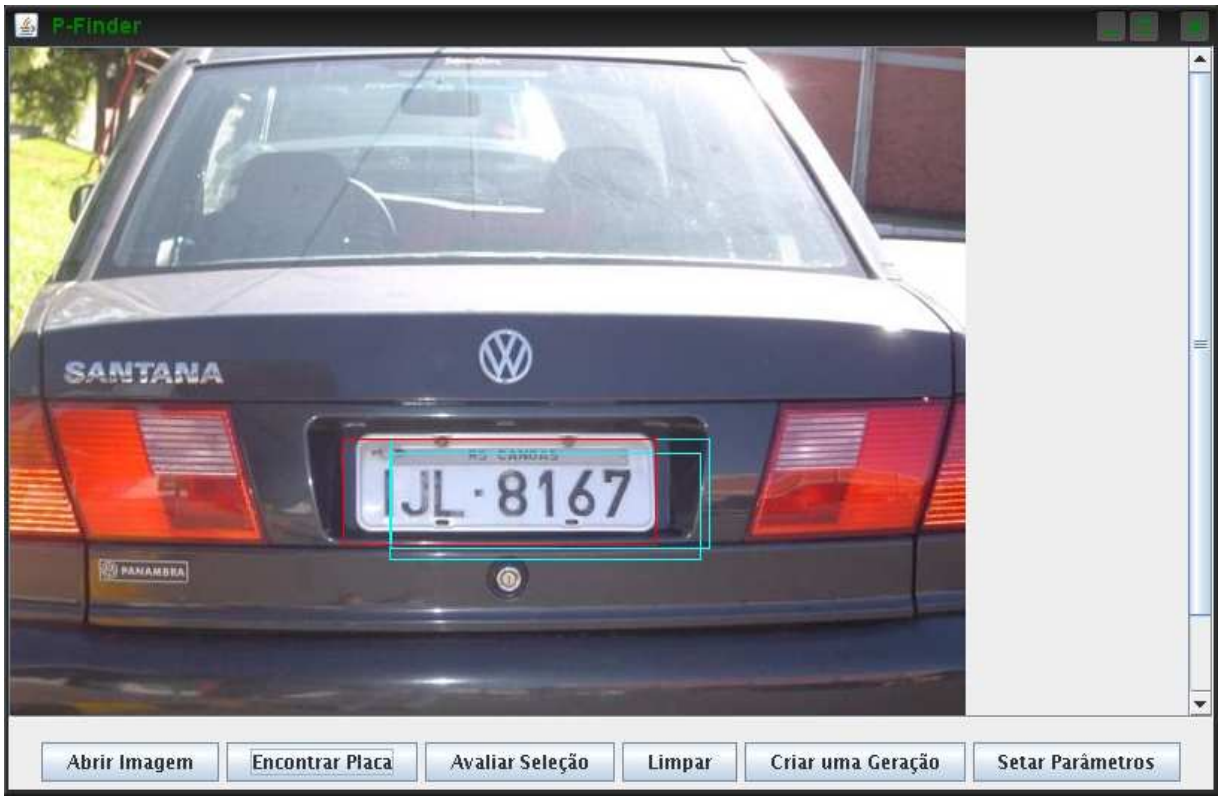

FIGURA 31 - ENCONTRAR PLACA.

13 TELA DESENHAR RETÂNGULO

É apresentada quando o usuário clica com o botão do mouse em cima da imagem exibida na tela “Tela Principal com Imagem Selecionada” e o move para baixo e direita até a posição desejada e solta o botão, um retângulo branco é desenhado na imagem dentro dos limites entre o pressionar e soltar do botão do mouse executado, como observado no exemplo da figura 6.12. Possui os botões: Abrir Imagem; Encontrar Placa; Avaliar Seleção; Limpar; Criar uma geração; Setar Parâmetros.

FIGURA 32 - DESENHAR RETÂNGULO.

14 TELA SAIR DO SISTEMA

É apresentada quando o usuário clica para fechar a janela da Tela Principal. Possui os botões: Sim; Não.

FIGURA 33 - SAIR SISTEMA.