ROBÓTICA INTELIGENTE: USO DE VISÃO COMPUTACIONAL VOLTADA PARA O CONTROLE E NAVEGAÇÃO DE ROBÔS MÓVEIS AUTÔNOMOS

Matheus Doretto Compri, Fernando Osório, Denis Wolf

Instituto de Ciências Matemáticas e de Computação (ICMC), USP/São Carlos
Departamento de Sistemas de Computação, Laboratório de Robótica Móvel (LRM)
matheusdc@grad.icmc.usp.br, fosorio@icmc.usp.br, denis@icmc.usp.br

Resumo A Robótica é um ramo da tecnologia que visa à criação e operação de um agente (real ou virtual) que realize tarefas de forma controlada ou autônoma. Robôs, como são denominamos esses agentes, podem ser usados para diversos propósitos, dentre os quais destacaremos a aplicação com fins didáticos e de pesquisa. O objetivo desse projeto é implementar um algoritmo de visão computacional que possa ser usado para controlar a navegação segura de um robô, e utilizar algoritmos genéticos, para otimizar as funções de processamento de imagens, fazendo com que o robô possa se adaptar ao meio no qual está inserido (e.g tipo de solo, obstáculos, texturas e iluminação), possibilitando um melhor tratamento das imagens e sua posterior aplicação para o controle da navegação em vários tipos de ambientes.

Palavras Chaves: Robótica, Visão Computacional, Processamento de Imagens, Algoritmos Genéticos.

Abstract: Robotics is a branch of technology that deals with design and operation of an agent (real or virtual), which executes tasks controlled or autonomously. Robots, how those agents are usually called, are used into many different applications, where in this paper we will focus on their use in educational and research applications. Our project is to create a computational vision algorithm that could be used to control secure navigation of a robot, and also to optimize parameters of image processing using genetic algorithms, in order to make possible to work with different images and environments (e.g. ground type, obstacles, textures and illumination) allowing a secure navigation in many different places.

Keywords: Robotics, Computer Vision, Image Processing, Genetic Algorithms.

1 INTRODUÇÃO

Um dos principais objetivos da visão computacional é proporcionar uma maneira de interpretar as informações captadas do ambiente através de câmeras, para transformá-las em comandos para a navegação segura de robôs, fonecendo informações suficientes para que obstáculos possam ser detectados e evitados.

Artigo Submetido para a
Mostra Nacional de Robótica Agosto/2012

Para isso utilizamos as imagens de uma câmera de vídeo (tipo webcam), e aplicamos filtros de processamento de imagens para tentar extrair informações relevantes delas (bordas dos obstáculos), e a partir das informações obtidas gerar comandos de navegação para os robôs, como avançar e desviar/virar.

2 OBJETIVOS

Nosso projeto tem como objetivo, descrever a contrução de um algoritmo de visão computacional que permita a navegação segura de um robô, isto é, permitir que o ele detecte e desvie de obstaculos, além de se adaptar as características do ambiente, pela otimização dos parâmetros adotados no processamento de imagens, utilizando para isto Algoritmos Genéticos.

3 FERRAMENTAS UTILIZADAS NA PESQUISA

Neste projeto utilizamos uma câmera tipo WebCam para a captura de imagens, e a biblioteca de código aberto OpenCV (Bradski & Kaehler 2008) para o processamento de imagens.

4 DESCRIÇÃO DO MÉTODO PROPOSTO

Esse projeto pode ser dividido em três partes principais, o processamento de imagens, a otimização dos parâmetros e a tomada de decisão.

4.1 PROCESSAMENTO DE IMAGENS

O processamento de imagens, é a primeira parte do processo, e inicia-se pela captura de imagens da câmera, que foi feita através da biblioteca OpenCV (Bradski & Kaehler 2008), a qual retorna uma imagem com os três componentes RGB, e com resolução de 640x480 pixels.

O primeiro passo para a navegação segura, é identificar quais são os obstáculos, e onde estão localizados em relação ao robô/câmera. Para isso, utilizamos algoritmos para a detecção de contornos dos objetos, pois com eles podemos detectar as bordas dos objetos e assim identificar as áreas navegaveis, ou seja, as áreas livres de obstáculos.

O piso (superfície de apoio) sobre o qual o robô/câmera estão apoiados é considerado navegável, e ele se estende da posição do robô até onde for detectada uma borda de um obstáculo, o qual será considerado como uma área não navegável (desviar).

Uma das vantagens dos algoritmos de detecção de bordas é que eles transformam as imagens de 3 canais (RGB), em apenas um canal (*Grayscale*), diminuindo assim o número de informações e melhorando o desempenho do processamento. O desempenho, é uma questão essencial para aplicações como esta, pois se espera que o robô responda em tempo real, devido ao ambiente que pode sofrer modificações a qualquer momento, e um algoritmo lento, poderia afetar o fluxo das informações, já que as etapas posteriores ao processamento necessitam de informações anteriores, e consequentemente a resposta ficaria desatualizada quando terminasse de ser processada.

Para o processamento das imagens utilizamos o filtro Canny (também da biblioteca OpenCV), a fim de obter os contornos, onde a Figura 4.1 mostra um exemplo da aplicação do filtro em um frame. A seguir, os parâmetros dessa função de detecção de bordas foram otimizados por um algoritmo genético, descrito na proxima seção, visando melhorar a imagem final obtida com os contornos.

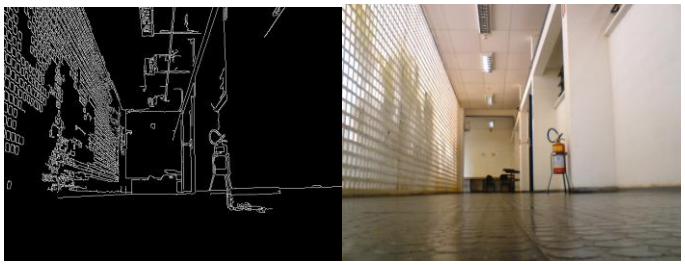

Figura 4.1 - Detecção de bordas com parâmetros otimizados

4.2 OTIMIZAÇÃO DOS PARÂMETROS

O processamento de imagens, depende de vários parâmetros, que controlam o comportamento dos filtros, e esses valores variam muito dependendo da luminosidade do ambiente, da textura e cor da superfície onde o robô está apoiado, dentre outros fatores.

Uma escolha errada de parâmetros, pode dificultar a navegação do robô, uma vez que o processamento de imagens ajusta os detalhes e modo como o robô irá perceber o ambiente. Um dos principais problemas enfrentados, é a textura da superfície de apoio do robô, pois se ela possuir padrões, ou mesmo manchas, essas podem ser interpretadas errôneamente como obstáculos, e confundir o robô durante a navegação. A Figura 4.2 mostra um exemplo de uma detecção de bordas com muitos ruidos, causada por uma escolha ruim dos parâmetros, e onde o robô certamente não teria um bom desempenho.

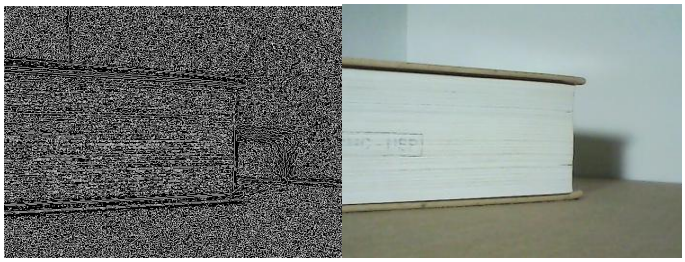

Figura 4.2 – Exemplo de detecção de bordas com ruídos

Uma boa escolha de parâmetros permite corrigir/diminuir esse problema, pois faz com que detalhes irrelevantes a navegação sejam ignorados, melhorando assim o desempenho geral do algoritmo. A maior dificuldade é descobrir quais são os parâmetros exatos, pois em nosso caso, a função Canny recebe como entrada dois valores inteiros variando de 0 a 255. Analisar todas as possibilidades de combinações destes parâmetros tornaria inviável para a aplicação em questão, pois todas possibilidades teriam que ser testadas nas imagens, onde o custo computacional de processar toda uma imagem não é negligenciável. O tempo de espera para o processamento das imagens e da análise de todas as 65536 combinações possíveis, pode ser muito grande dependendo da capacidade de processamento local do robô, sendo necessário assim encontrar uma solução baseada em algum método de otimização.

Os Algoritmos Genéticos foram escolhidos nesse projeto, devido a sua facilidade de implementação e ótimo desempenho em otimização, se comparado a outros métodos de Inteligência Artificial, como Redes Neurais Artificiais, e principalmente pela sua velocidade de convergência para soluções satisfatórias, o que pode ser comprovado empiricamente.

Uma questão importante de nosso projeto, foi como classificar se uma determinada configuração de valores dos parâmetros é boa ou ruim. Para isso, analisamos as imagens e percebemos que a parte inferior delas possuia a área navegável, e que representa a superfície sobre a qual o robô está apoiado. Os padrões e texturas da superfície de apoio, e que não são relevantes para o desvio de obstáculos na navegação, acabam apenas confundindo as funções de tomada de decisão, e devem portanto ser exluídos da imagem após a etapa de detecção de contornos dos obstáculos. Assim criamos o Algoritmo 4.1 como função de avaliação das soluções, que recebe como parâmetro uma imagem na qual foi aplicado o filtro Canny, e calcula a porcentagem dos pixels brancos (bordas), provenientes de um contorno identificado na parte inferior da imagem. É importante destacar que em uma fase inicial esta parte inferior da imagem é ajustada de tal forma que não existam bordas nela (área navegável sem obstáculos), e portanto devemos minimizar o total de pixels brancos (bordas) presentes nesta área. Assim é possível estimar um valor numérico da "qualidade de uma solução" (*fitness*) para poder comparar as soluções.

Algoritmo 4.1 para Avaliação das soluções

```
FuncaoAvaliacao(frame)
 1   inicio_algoritmo
 2   Pixel atual;
 3   Inteiro maxPixels, pixelsBrancos;
 4   maxPixels = 0;
 5   pixelsBrancos = 0;
 6   para I = Limite_Inferior_Area_Navegável até 0 faça
 7      para J = 0 até Max_Colunas_Imagem faça
 8         atual = frame(I, J);
 9         maxPixels = maxPixels + 1;
10         se pixel = Cor_Borda_Valor_255 faça
11            pixelsBrancos = pixelsBrancos + 1;
12         fim_se;
13      fim_para;
14   fim_para;
15   retorna (pixelsBrancos*100)/maxPixels;
16    fim_algoritmo;
```

Escolhemos como gene do Algoritmo Genético, o conjunto dos dois parâmetros passados para a função Canny, que representam os limites do *threshold* usado por ele. O número de gerações e o tamanho da população foram escolhidos tambem empiricamente, analisando os gráficos de evolução das populações de genes, onde esses valores foram definidos respectivamente como 10 indivíduos e 10 gerações.

O algoritmo 4.2 contem uma descrição do metódo de evolução das soluções, que segue o modelo clássico dos Algortimos Genéticos. Ele foi escrito em alto nivel para abstrair os detalhes de implementação, que podem ser consultados na bibliografia especializada (Linden, 2006). Esse processo consiste em iniciar um população de genes com valores aleatórios, e em seguida avaliá-los. Após isto, escolhe-se os indivíduos mais bem classificados e realiza-se o *crossover*, para obter novos indivíduos, que serão tambem avaliados. Posteriormente, são escolhidos os melhores genes, que prevalecerão para a próxima geração.

Algoritmo 4.2 para a Evolução das populações

```
AlgoritmoGenetico(População, maxGeracoes)
 1   inicio_algoritmo
 2   inicializar a População;
 3   para I=0 até MaxGerações faça
 4      Avaliar a População;
 5      Escolher indivíduos mais bem classificados.
 6      Realizar o crossover dos mais bem classificados.
 7      Avaliar os novos indivíduos.
 8      Escolher sobreviventes para a próxima geração.
 9   fim_para;
10   fim_algoritmo.
```

4.3 TOMADA DE DECISÃO

Para que a navegação segura efetivamente ocorra, precisamos analisar as imagens em busca de itens que nos indiquem qual comportamento deve ser assumido, por exemplo, seguir em frente (caminho livre), desviar para esquerda ou desviar para a direita.

A Figura 4.3, mostra uma situação onde a detecção de bordas encontrou objetos em um frame, e a partir dela faremos a seguir algumas análises sobre quais situações são possiveis de ocorrer, e o que pode ser interpretado a partir dela.

Figura 4.3 – Exemplo de objetos detectados

Podemos perceber que a figura pode ser dividida inicialmente em 2 partes, a parte superior da imagem, que contém os objetos que estão relativamente longe do robô, e a parte inferior da imagem que contém os objetos que estão mais próximos. Os objetos na parte superior da imagem não afetam muito as decisões a serem tomadas, pois o robô levará um tempo maior para se aproximar deles, portanto concentramos a tomada de decisão nos elementos da parte inferior da imagem, onde os objetos estão próximos e uma possivel mudança de trajetória pode ser necessária para evitar a colisão.

A parte inferior da imagem, pode ser dividida em 3 partes, a esquerda, a direita e a central. Essa divisão será util para escolher qual a direção que será seguida.

O Algoritmo 4.3, mostra como o robô pode analisar a imagem processada e escolher o melhor caminho a seguir.

Algoritmo 4.3 para a Tomada de decisão

```
TomadaDecisao(Frame)
 1   inicio_algoritmo
 2   Inteiro posicao[];
 3   Inteiro esquerda, direita, centro;
 4   Literal direcao;
 5   esquerda = 0;
 6   direita = 0;
 7   centro = 0;
 8   para I = 0 até Max_Linhas_Imagem faça
 9     para J = 0 até Max_Colunas_Imagens faça
10        posicao[J] = I;
11     fim_para
12   fim_para
13   para I = 0 até 100 faça
14     se(posicao[I] < 300) faça
15        esquerda = 1;
16     fim_se;
17   fim_para;
18   para I = 100 até 540 faça
19     se(posicao[I] < 300) faça
20        centro = 1;
21     fim_se;
22   fim_para;
23   para I = 540 até 640 faça
24     se(posicao[I] < 300) faça
25        direita = 1;
26     fim_se;
27   fim_para;
28   se(centro = 0) faça
29     direcao = “Ir para Frente”;
30   senão se(esquerda = 0) faça
31     direcao = “Ir para Esquerda”;
32   senão se(direita = 0) faça
33     direcao = “Ir para Direita”;
34   senão faça
```

```
35      direcao = “Pare”;
36    fim_se;
37    fim_algoritmo;
```

A ideia do algoritmo é percorrer a imagem, procurando pelos pixels dos contornos, e salvar o primeiro contorno em um vetor para facilitar o posterior processamento. Depois percorremos o vetor em busca de um obstáculo na parte inferior da imagem, que consideramos ser abaixo da coluna 300, onde no entanto essa altura pode variar dependendo do tamanho do robô e da imagem. Dividimos essa busca em três partes, a esquerda, no centro e a direita. Quando encontramos o obstáculo marcamos um *flag*, para indicar a sua presença na respectiva direção, e no fim escolhemos qual é a melhor direção a seguir baseado nos *flags*.

4.4 REUNINDO OS PROCESSOS

Todas as etapas descritas nos itens anteriores, devem ser combinadas no Algoritmo 4.4, que exemplifica, em linguagem de alto nível o funcionamento do processo de navegação do robô.

Algoritmo 4.4 para Navegação do Robô

```
AlgoritmoRobo( )
1    inicio_algoritmo;
2    utiliza-se o Algoritmo Genético para otimizar os parâmetros;
3    repita os passos abaixo
4      aplica o filtro Canny em um frame com os parametros já otimizados;
5      aplica a tomada de decisão;
6    fim_repita;
7    fim_algoritmo;
```

O Algoritmo 4.4 consiste inicialmente em otimizar os parâmetros do filtro Canny utilizando os Algoritmos Genéticos, e uma vez otimizados, é possivel então aplicá-los em todas as iterações posteriores, analisando cada frame e seus obstáculos a fim de tomar a decisão do melhor caminho e mais seguro a percorrer.

5 RESULTADOS

A eficácia do algoritmo de controle e navegação do robô é influenciada pela eficiência do Algoritmo Genético que utilizamos para otimizar os parâmetros do Canny. Se o Algoritmo Genético não encontrar uma boa solução e não for tão bem sucedido na otimização dos parâmetros, assim também a navegação será afetada, pois o controle de navegação do robô é realizado baseado na qualidade do processo de detecção das bordas.

Uma análise mais completa dos resultados deste trabalho pode ser feita observando-se a Figura 5.1. Esta figura representa o espaço amostral das possíveis configurações dos valores dos parâmetros e as suas respectivas avaliações (*fitness*). A figura foi obtida através do cálculo de todas as possibilidades de configuração dos parâmetros para uma determinada imagem.

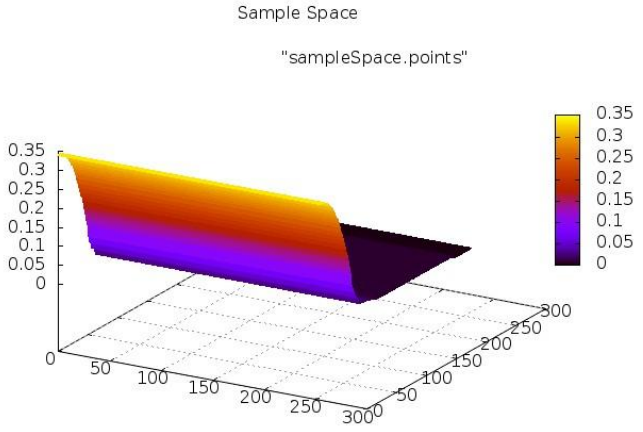

Figura 5.1 – Espaço amostral de exemplo

Neste caso, percebemos que existem duas situações específicas que são importantes de serem destacadas, as situação das piores soluções, e aquelas que são supostamente as melhores, mas que no entanto são apenas aparentemente boas, mas resultam de uma análise equivocada do valor da avaliação. As piores soluções estão na faixa onde os parâmetros do *threshold* são muito baixos, implicando assim que vários contornos desnecessários sejam detectados (falsos positivos), impedindo uma navegação adequada, que consideraria a área como estando repleta de obstáculos. Um exemplo disso pode ser visto na Figura 5.2.

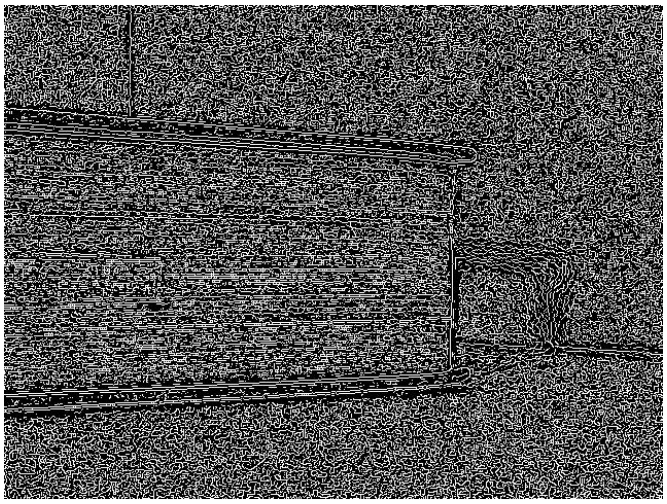

Figura 5.2 – Exemplo de detecção de bordas com valores dos parâmetros baixos

Por outro lado, podemos ser induzidos a encontrar valores dos parâmetros que levam a soluções equivocadas, que são aquelas que não possuem quase nenhum contorno (falsos negativos), seja na área navegável ou sejam também os contornos dos obstáculos (área não navegável) que simplesmente deixaram também de ser detectados gerando uma imagem sem contornos. Neste caso, o processamento da imagem irá fornecer um resultado sem nenhuma base (ou com muito poucos detalhes) para a tomada de decisão na navegação, como pode ser visto na Figura 5.3, onde algumas bordas foram eliminadas, como por exemplo a borda da mesa com a parede de fundo.

Figura 5.3 – Exemplo de detecção de bordas com valores dos parâmetros altos

Para os demais casos, as soluções são satisfatórias, pois possuem os contornos necessários, e as texturas dos pisos são em sua maioria ignoradas, como pode ser observado na Figura 5.4, que representa uma detecção de bordas bem sucedida.

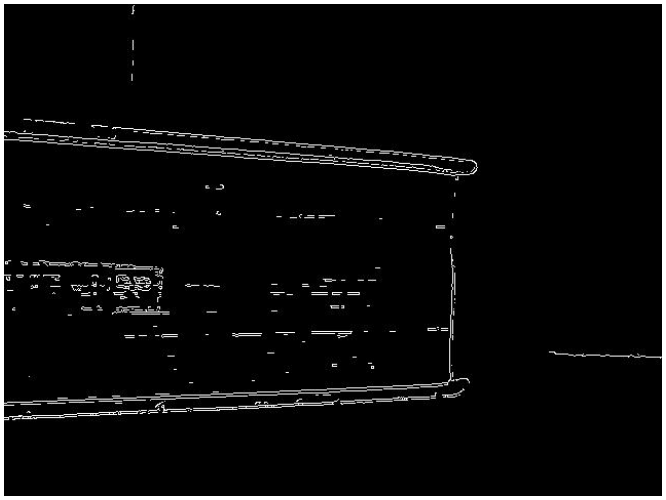

Figura 5.4 – Exemplo de detecção bem sucedia

Isto no leva a considerar a adoção de uma melhor função de avaliação das soluções, que não inclua apenas a contagem dos pixels da área navegável, mas também a contagem dos pixels da área não navegável (obstáculos). Atualmente estão sendo aperfeiçoados trabalhos de pesquisa e de testes nesta direção.

Em relação ao processo de otimização, a Figura 5.5 mostra um gráfico de evolução da população, para uma superfície de cores escuras, onde em cada geração do Algoritmo Genético, foram registradas as avaliações do melhor indivíduo, do pior, e a avaliação da média da população. Os valores da evolução do processo de otimização podem ser observados na tabela 5.1, onde constatamos uma clara convergência do Algoritmo Genético ao longo das 10 gerações analisadas.

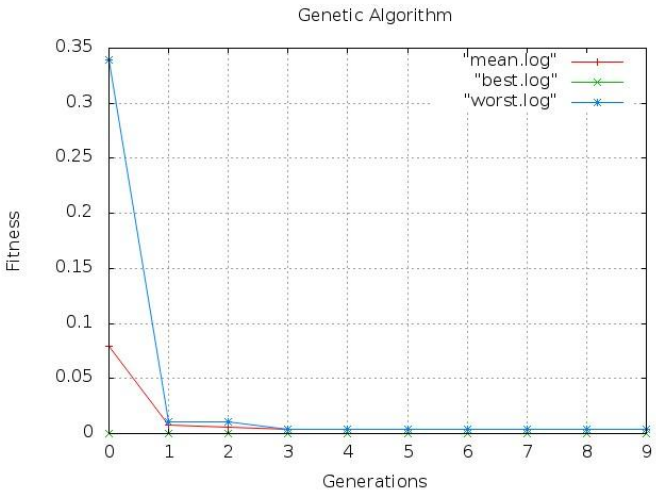

Figura 5.5 – Gráfico de evolução do Algoritmo Genético para uma superfície de cor escura

Tabela 5.1 – Valores do Gráfico da Figura 5.5

Geração	Pior	Media	Melhor
0	0.166189	0.104245	0.0591037
1	0.100501	0.0823132	0.0591037
2	0.0929715	0.0769471	0.0591037
3	0.0789559	0.0706469	0.0578931
4	0.0713674	0.0653734	0.0578931
5	0.0692513	0.0616638	0.0578931
6	0.0591037	0.0582563	0.0578931
7	0.0578931	0.0578931	0.0578931
8	0.0578931	0.0578931	0.0578931
9	0.0578931	0.0578931	0.0578931

Analisando a curva do gráfico da Figura 5.5 e os dados da Tabela 5.1, podemos perceber que ocorre a diminuição da porcentagem de pixels brancos, provenientes dos contornos, no decorrer das gerações, evidenciando assim que a evolução da população está convergindo para o resultado desejado em termos da função de avaliação (*fitness*). Sendo assim, o Algoritmo Genético é capaz de obter um conjunto de parâmetros que consiga fazer com que a textura e padrões do piso não sejam mais detectados e evitando assim o desvio desnecessário de falsos obstáculos nas regiões navegáveis. Com isto, o processo de otimização irá garantir uma melhor qualidade da tomada de decisão do robô no processo de navegação.

Já a figura 5.6, mostra tambem um gráfico de evolução mas para superfícies de cores claras. Fazendo uma análise comparativa entre os dois gráficos podemos perceber que o das superfícies claras converge um pouco mais devagar que o da superfícies escuras, mas ambos possuem um gráfico decrescente, comprovando assim a evolução do algoritmo genético para diferentes tipos de superfícies.

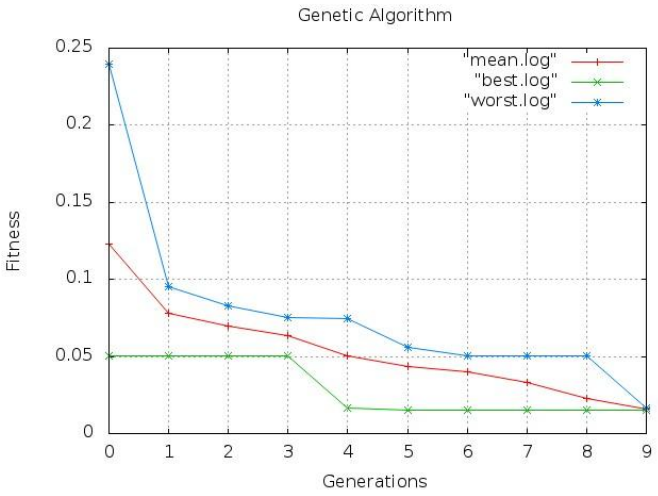

Figura 5.6 - Gráfico de evolução do Algoritmo Genético para uma superfície de cor clara

As Figuras 5.7 e 5.8 contém exemplos do funcionamento do algoritmo completo. As setas amarelas indicam a direção que ele sugere para ser tomada. Na figura 5.6 o obstáculo se encontra relativamente longe da camêra, por isso o algoritmo detecta que é seguro ir para frente. Já na figura 5.7 o obstáculo se encontra muito mais próximo, e ele indica que é necessário virar a esquerda. Podemos perceber assim que o algoritmo funciona da maneira esperada para a navegação segura, seguindo sempre em frente enquanto possivel e escolhendo o melhor caminho para seguir quando o obstáculos são encontrados.

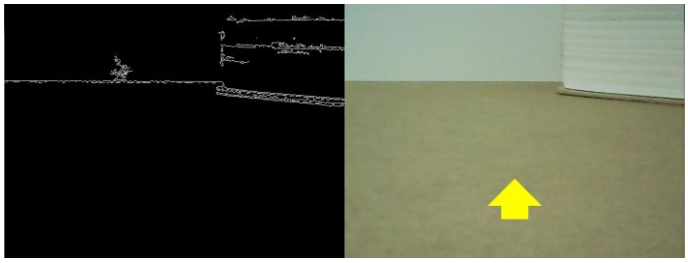

Figura 5.7 – Exemplo de navegação com obstáculo longe

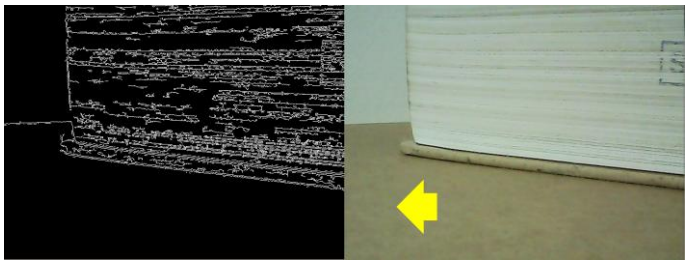

Figura 5.8 – Exemplo de navegação com obstáculo perto

6 CONCLUSÕES E TRABALHOS FUTUROS

Conforme discutido anteriormente, o Algoritmo Genético usado na otimização dos parâmetros permite eliminar o ruído (falsos positivos) na zona navegável, mas ainda precisa ser aperfeiçoado de modo a não eliminar erroneamente as bordas na zona não navegável (falsos negativos). Além disto, em alguns casos onde a textura da superfície navegável for parecida com a dos obstáculos, é possível que elas não sejam diferenciadas, pois os parâmetros foram evoluídos para eliminar ao máximo todas as texturas semelhantes.

As limitações identificadas neste trabalho estão sendo atualmente trabalhadas, mas porém, destacamos que o presente trabalho já permitiu demonstrar bons resultados na identificação das áreas navegáveis, bem como, permitiu também demonstrar a validade e o bom desempenho do uso dos Algoritmos Genéticos como uma ferramenta para a otimização dos parâmetros usados no processamento de imagens e na detecção de obstáculos.

7 AGRADECIMENTOS

Agradecemos o apoio financeiro da USP-PRP (Pró-Reitoria de Pesquisa) pelo financiamento da Bolsa PIBITI ligada a este projeto. Gostaríamos também de agradecer ao LRM - Laboratório de Robótica Móvel do ICMC/USP e INCT-SEC, bem como aos seus membros pelo apoio e equipamentos disponibilizados para a realização deste trabalho.

8 REFERÊNCIAS BIBLIOGRÁFICAS

Bradski, Gary; Kaehler, Adrian. (2008). Learning OpenCV: Computer Vision with the OpenCV Library.

Gonzalez & Woods (2002), Digital Image Processing. Prentice Hall.

Wolf, D.; Osório, F. S.; Simões, E.; Trindade, O. (2009) Robótica Inteligente: Da Simulação às Aplicações no Mundo Real. In JAI – Jornada de Atualização em Informática 2009 (Tutorial) – Congresso da SBC – Bento Gonçalves SBC – Editora da PUC Rio: Rio de Janeiro – RJ.

Linden, R.(2006) Algoritmos Genéticos: Uma importante ferramenta da Inteligência Computacional. Brasport. Rio de Janeiro – RJ.