UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

SISTEMA DE VISÃO COMPUTACIONAL UTILIZANDO RAIO LASER E CÂMERA DIGITAL

NIVALDO PÜHLER

BLUMENAU
2010

2010/2-22

NIVALDO PÜHLER

SISTEMA DE VISÃO COMPUTACIONAL UTILIZANDO RAIO LASER E CÂMERA DIGITAL

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Antônio Carlos Tavares, Mestre - Orientador

BLUMENAU
2010

2010/2-22

SISTEMA DE VISÃO COMPUTACIONAL UTILIZANDO RAIO LASER E CÂMERA DIGITAL

Por

NIVALDO PÜHLER

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Antônio Carlos Tavares, Mestre – Orientador, FURB
Membro:	Prof. Miguel Alexandre Wisintainer, Mestre – FURB
Membro:	Prof. Francisco Adell Péricas, Mestre – FURB

Blumenau, 09 de dezembro de 2010

Dedico este trabalho a todos os meus familiares, que sempre me apoiaram em toda minha graduação, a todos professores e amigos que sempre estiveram presentes em minha vida.

AGRADECIMENTOS

À Deus, pelo seu imenso amor e graça.

À minha família, que sempre esteve presente me apoiando muito.

Aos meus amigos, pelos ótimos momentos que vivemos juntos.

Ao meu orientador, Antônio Carlos Tavares, por ter acreditado na conclusão deste trabalho.

Aos colegas de trabalho que sempre me incentivaram durante toda minha graduação.

Com paciência e perseverança muito se alcança.

Théophile Gautier

RESUMO

Atualmente, muitos estudos buscam desenvolver de técnicas aplicadas à automatização de processos e medidas para desenvolver o setor de meios de transportes. O Departamento de Sistemas e Computação (DSC) – da Universidade Regional de Blumenau (FURB) – possui o projeto Veículo Autônomo Não Tripulado (VANT), onde se pretende permitir que um carro possa locomover-se sem a interferência do ser humano. Procurando incrementar esse projeto, este trabalho detalha o desenvolvimento de um sistema de posicionamento, no qual é utilizado raio laser, para criar marcos visuais em imagens captadas por uma câmera digital. Essas imagens são tratadas para que seja possível identificar a aproximação de um obstáculo, bem como sua distância do mesmo. Os resultados obtidos podem ser aplicados em um sistema de tomada de decisão, para determinar qual atitude tomar ao encontrar um obstáculo.

Palavras-chave: Visão computacional. Raio laser.

ABSTRACT

Currently, many studies seek to develop techniques applied to process automation and have sought measures to develop the sector of transport means. The department of systems and computing (DSC) - Regional University of Blumenau (FURB) - has the project unmanned autonomous vehicle (VANT), which aims to allow that a car can move without interference from humans. Looking to increase this project, this work presents the development of a positioning system, which is used ray laser, to create visual landmarks in images captured by a digital cam. These images are processed to make it possible to identify the approaching an obstacle, and their distance from it. The results can be applied in a system for making decision to determine what action to take when encountering an obstacle.

Key-words: Computer vision. Laser.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE SIGLAS

API – *Application Programming Interface*

BCC – Curso de Ciência da Computação – Bacharelado

DSC – Departamento de Sistemas e Computação

JAI – *Java Advanced Imaging*

JMF – *Java Media Framework*

RF – Requisito Funcional

RGB – *Red, Green e Blue*

RNF – Requisito Não-Funcional

USB – *Universal Serial Bus*

SUMÁRIO

1 Introdução

O emprego de sistemas robóticos utilizados para suprir necessidade de automatização de processos é uma realidade atualmente. Porém, grande parte encontra-se em ambientes industriais.

Recentemente tem se intensificado o desenvolvimento de tecnologias para aplicação em sistemas robóticos móveis não tripulados. Uma destas tecnologias é a visão computacional, que tem o objetivo de extrair informações úteis das imagens captadas do ambiente, podendo ser usada como um sensor para o posicionamento do veículo e detecção de obstáculos.

Um sensor é um dispositivo capaz de adquirir informações sobre determinados fenômenos físicos, podendo fornecer diretamente ou indiretamente um sinal que represente esta grandeza (MOREIRA, 2003, p. 08).

Para um veículo autônomo, o posicionamento relativo no ambiente que o circunda é de essencial importância tanto para as estratégias de navegação quanto para planejamento e tomada de decisões (CASTRO, 2007, p. 01).

Uma das soluções utilizadas em um sistema de visão computacional composto por apenas uma câmera para medir deslocamento de um veículo no plano horizontal, considerando-se o movimento de avanço e deriva, consiste na inserção de marcos visuais reconhecíveis pelo sistema no ambiente a ser inspecionado (BUSCARIOLLO, 2008, p. 34).

A aplicação deste sistema de visão mono é limitada para ser usada como sensor de posição. No entanto, ele pode ser aprimorado se forem inseridos novos marcos visuais de tal modo que os deslocamentos possam ser medidos independentemente do conhecimento da posição inicial e do ângulo relativo entre o veículo e o objeto alvo (BUSCARIOLLO, 2008, p. 37).

Uma sugestão é a utilização de raio laser para criar marcos visuais reconhecíveis pelo sistema a partir da incidência dos raios. Entre os métodos de sensoriamento mais utilizado para medir distância, baseados em um sistema a laser, estão o método *time-of-fight* (tempo de trajetória) e o método de triangulação (BUSCARIOLLO, 2008, p. 38).

Para obter-se uma boa solução na detecção de obstáculos, bem como sua distância em relação ao mesmo, são necessários estudos abrangendo áreas de matemática, computação gráfica, robótica e programação.

Considerando os fatores acima citados, pretende-se estudar as técnicas de

sensoriamento utilizando visão computacional e desenvolver um protótipo que capture imagens a partir de uma câmera, nas quais será possível detectar marcos visuais criados por raios laser. Estas imagens deverão ser tratadas para obter informações necessárias para a determinação da distância que o veículo autônomo não tripulado encontra-se de um obstáculo.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um sistema de posicionamento dinâmico em um veículo autônomo não tripulado, proporcionando a detecção de obstáculos.

Os objetivos específicos do trabalho são:

a) capturar imagens através de uma câmera, na qual terá um feixe de luz do canhão laser;
b) calcular a variação da posição do feixe de luz, para detectar obstáculos próximos.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado em quatro capítulos. O capítulo 2 apresenta as tecnologias envolvidas e os aspectos teóricos estudados para o desenvolvimento deste trabalho. São relatados temas como veículos autônomos, sensores, visão computacional, imagem digital, laser e bibliotecas que serão utilizadas neste projeto.

No capítulo 3 é abordado o desenvolvimento do presente trabalho, detalhando a especificação e implementação.

O capítulo 4 apresenta as conclusões deste trabalho, bem como sugestões para possíveis extensões.

2 FUNDAMENTAÇÃO TEÓRICA

Na seção 2.1 são apresentados alguns conceitos sobre veículos autônomos. Na seção 2.2 comentam-se aspectos a respeito de sensores. Na seção 2.3 encontram-se algumas informações sobre visão computacional. Na seção 2.4 encontram-se alguns conceitos sobre imagem digital. Na seção 2.5 são apresentados conceitos sobre laser. Na sessão 2.6 encontram-se informações sobre Java *Application Programming Interface* (API) utilizadas no trabalho. Na sessão 2.7 são apresentados trabalhos correlatos ao tema em questão.

2.1 VEÍCULOS AUTÔNOMOS

A propriedade de autonomia de um sistema reflete a sua capacidade de executar missões com intervenção humana reduzida ou nula. Por missão, entende-se uma coleção de atividades estruturadas de forma a assegurar a possibilidade de atingir um certo conjunto de objetivos (HEINEN, 2001, p. 32).

Os veículos autônomos têm atraído a atenção de um grande número de pesquisadores da área de Inteligência Artificial, devido ao desafio que este novo domínio de pesquisas propõe: dotar sistemas de capacidade de raciocínio inteligente e de interação com o meio em que estão inseridos. Os veículos autônomos podem perceber o ambiente em que estão inseridos através da leitura de seus sensores (infravermelho, sonar, lasers, câmeras de vídeo, etc.), e através desta percepção sensorial eles podem planejar melhor as suas ações. Atualmente temos robôs móveis atuando em diferentes áreas, como por exemplo: robôs desarmadores de bombas, robôs usados para a exploração de ambientes hostis, e a condução de veículos (carros) robotizados (HEINEN, 2001, p. 36).

Estevam (2003, p. 13) afirma que veículos autônomos caracterizam-se por serem sistemas automáticos que se deslocam sozinhos no espaço, podendo funcionar sem a presença de qualquer tripulante, seja local ou remota, e sem ligação física ou de qualquer outro tipo, a outros dispositivos, sendo de grande utilidade em aplicações como, auxilio a deficientes, condução de veículos, transporte, monitoramento e segurança em ambientes, exploração de ambientes hostis, entre outros.

2.2 SENSORES

Os sensores tem como função nos veículos autônomos a mesma função que tem nos humanos, sentir o ambiente e comandar suas reações. O uso de sensores, permite que um robô ou veículo autônomo possa interagir com o ambiente em que o rodeia de uma forma flexível. Isto não acontece, nas operações pré-programadas (mais comuns nos robôs industrais) onde um robô é ensinado como proceder para realizar tarefas repetitivas de um conjunto de funções programadas. O uso da tecnologia dos sensores, introduz nas máquinas um maior nível de inteligência para lidar com o seu meio sendo objeto de uma pesquisa intensa no campo da robótica (FORTES, 2001).

Os veículos autônomos utilizam, principalmente, dois grupos de sensores, os sensores de colisão e os de posicionamento. Para um melhor desempenho, deve-se utilizar vários destes sensores ao mesmo tempo, e o software deve ter a capacidade de integrar os dados destes sensores, fazendo com que seus atuadores se comportem da maneira correta (HEINEN, 2000, p. 19).

2.2.1 Sensores de colisão

Estes tipos de sensores são utilizados para detectar possíveis obstáculos que estejam no seu caminho. Os sensores de colisão também são utilizados para desenhar um mapa do ambiente (HEINEN, 2000, p. 19). Dentre os sensores de colisão podemos citar os sensores infravermelho, sonar, laser, câmera de vídeo e sensores de contato, os quais são descritos a seguir.

2.2.1.1 Infravermelho

Este tipo de sensor é muito utilizado, principalmente por ter baixo custo e ter funcionamento relativamente simples, apesar de ter a desvantagem de seu raio de ação ser bem reduzido.

Um diodo infravermelho emite um raio modulado, este raio atinge um objeto e uma porção da luz é refletida, sendo captada de volta através do receptor ótico e atingindo um

conjunto de fotodiodos. Dependendo da posição do objeto, o ângulo de incidência da luz refletida é diferente, com isso pode-se calcular a distância deste objeto por triangulação (HEINEN, 2000, p. 20).

2.2.1.2 Sonar

A principal vantagem do sonar é seu baixo custo e a necessidade de poucos recursos computacionais. Mas a informação retornada pelo sonar é limitada por possuir um raio de atuação muito largo, o que causa leituras imprecisas do ambiente.

O seu funcionamento é relativamente simples, um transdutor emite uma pequena onda de som em alta freqüência, quando esta onda atinge um objeto ela é refletida e é novamente captada pelo transdutor. A distância pode ser calculada usando o tempo entre a emissão e o recebimento de uma onda de som. A distância é calculada dividindo-se o tempo decorrido por dois, e multiplicando o resultado pela velocidade do som (HEINEN, 2000, p. 21).

2.2.1.3 Laser

Existem vários tipos de sensores que utilizam o laser para detectar obstáculos. O mais simples deles utiliza o mesmo princípio utilizado nos sensores infravermelhos, um feixe laser é emitido e um fotosensor capta sua reflexão e calcula o tempo que foi preciso para o laser retornar. Uma desvantagem deste sistema é que os circuitos precisam ser muito precisos, pois a velocidade do laser é muito alta.

Outros tipos de sensores laser utilizam espelhos para detectar o obstáculo. Um motor controla o ângulo do espelho até que o feixe laser atinja o fotosensor, quando isto acontece pode-se calcular a distância usando o ângulo do espelho por triangulação (HEINEN, 2000, p. 22).

2.2.1.4 Câmeras de vídeo

O sensor de visão de um veículo autônomo é implementado através de uma câmera de vídeo. As imagens obtidas são processadas pelo computador, que analisa e então “decide” o

que fazer. A câmera pode ser colorida ou preto e branco. O processamento de imagens coloridas é mais difícil e necessita de mais tempo de processamento.

Existem alguns métodos para fazer o computador “entender” as imagens. Estes métodos são chamados de métodos de reconhecimento de padrões. O computador analisa cada imagem obtida da câmera para identificar certos objetos. Por exemplo, o brilho ou a cor dos objetos normalmente são diferentes do fundo, e comparando cada dois *pixels* vizinhos o computador pode achar as bordas dos objetos, depois de processada, a imagem contem somente as bordas externas do objetos. Este é o primeiro passo do reconhecimento de objetos. À partir destas informações é necessário saber então o que são estes objetos, seu formato, propriedades e localização. Então é comparado com padrões armazenados e obtêm-se os resultados (HEINEN, 2000, p. 23).

2.2.1.5 Sensores de contato

Compostos de simples botões que são acionados quando bate em algum obstáculo. Este sensor é utilizado como último recurso, no caso de estar se movendo a velocidade máxima e não conseguir detectar um obstáculo. Isto pode acontecer quando um obstáculo não é reflexivo a luz infravermelho, e/ou absorve os pulsos de alta freqüência do sonar, ou quando o obstáculo se encontra em uma posição onde o sensor não é capaz de detectá-lo (HEINEN, 2000, p. 23).

2.2.2 Sensores de posicionamento

Para Heinen (2000, p. 24), tão importante quanto detectar um obstáculo, é a capacidade de um veículo autônomo saber exatamente onde ele está em relação ao seu ambiente, para isso ele utiliza sensores de posicionamento que medem de forma absoluta ou relativa a posição atual. Estes sensores são:

a) bússola: tem a finalidade de informar o ângulo em que um veículo autônomo encontra-se em um determinado momento. Estes sensores têm a vantagem de retornarem um valor absoluto, que não depende de um estado anterior;
b) odômetros: tem a finalidade de medir a distância percorrida, com isso, pode-se calcular sua posição relativa no ambiente. Este sensor tem uma precisão muito

baixa, necessitando que seus valores sejam validados de tempos em tempos;

c) GPS (Global Positioning System): utiliza uma rede de satélites em órbita da Terra para determinar sua longitude, latitude e altitude do veículo autônomo. Este sensor também retorna um valor absoluto, mas tem a desvantagem de não ser muito eficiente em ambientes altamente urbanizados e de sua precisão ser muito baixa para uso civil;
d) *beacons* ou faróis: são dispositivos que são instalados em pontos estratégicos do ambiente, emitindo sinais luminosos ou de rádio, que são detectados pelo veículo autônomo e a partir deles pode-se calcular sua posição. Este tipo de sensor é utilizado principalmente para eliminar erros de posicionamento causados por odômetros e outros sensores posicionais relativos.

2.3 VISÃO COMPUTACIONAL

A visão computacional é uma área que se dedica a desenvolver teorias e métodos voltados à extração de informações úteis contidas em imagens. As imagens são captadas por dispositivos imageadores, tais como câmeras de vídeo, *scanner* e outros (GONZALES; WOODS, 2000, p.244).

Sistemas de visão Computacional envolvem técnicas de processamento digital de imagens. Existem duas abordagens comuns nesses sistemas, a primeira refere-se a aplicações de reconhecimento e a segunda a inspeção automatizada. Em um sistema para reconhecimento devem-se extrair características dos objetos da imagem e usar algum tipo de inteligência computacional para proceder a distinção entre os objetos. Neste caso não é fundamental obter os valores exatos destas características, por outro lado em sistemas de inspeção a exatidão é fundamental (FELICIANO, 2005, p. 38).

Particularmente, a visão computacional é um problema de grande interesse, uma vez que é o sentido humano que apresenta uma grande complexidade de funcionamento. Dentre as inúmeras aplicações da solução desse problema está o desenvolvimento de veículos autônomos capazes de se movimentar em um determinado ambiente. Para que um sistema autônomo se movimente em um certo ambiente é necessário que ele seja capaz de captar cenários sobre esse ambiente e, a partir deles, possa tomar decisões sobre trajetórias a serem seguidas. Esse problema tem sido resolvido a partir da geração de imagens relativas aos

cenários, com a utilização de câmeras de vídeo, sonares, etc., e da identificação, nessas imagens, de padrões previamente definidos, capazes de subsidiar a tomada de decisão sobre os movimentos a realizar (FELICIANO, 2003, p. 39).

Uma questão importante em um sistema de visão computacional são as câmeras de vídeo. Normalmente, as mais empregadas são da tecnologia *Charge Coupled Device* (CCD), dispositivos de carga acoplada. O sensor CCD é uma matriz de elementos sensíveis à luz, montado sobre um *chip*, onde a luz captada é transformada num padrão de cargas elétricas. Esta tecnologia baseia-se na varredura das linhas desta matriz, captadas pelo sensor. Após as imagens serem captadas, são lidas pelo computador, por meio de uma placa de captura de vídeo (GONZALES; WOODS, 2000, p. 09).

A tarefa principal de um sistema de visão computacional é compreender a cena que uma imagem – composta por uma matriz de *pixels* – representa. Contudo, outras áreas de pesquisa também apontam tarefas similares como seu objetivo final, entre eles os campos de processamento de imagens, reconhecimento de padrões e interpretação e reconhecimento de imagens. Apesar de haver certa interdisciplinaridade entre os campos mencionados acima, cada qual possui uma história e característica particular (CRUZ, 2004, p. 14).

Aparentemente a visão é simples para os seres humanos, mas ao mesmo tempo extremamente difícil de ser desenvolvida e implementada comparativamente em um sistema de visão computacional. Há muitas razões para este fato. Em primeiro lugar, uma imagem contém implicitamente uma cena que muitas vezes não fornece informações suficientes de permitir a recuperação da mesma em um sistema computacional. Entre outros aspectos relevantes, a profundidade de um objeto é geralmente encoberta pela projeção de uma imagem em três dimensões por uma outra imagem vista em duas dimensões. Outras informações também são necessárias para resolver tais ambigüidades, sendo que as mesmas são muitas vezes baseadas em suposições lógicas ou em medições do objeto, sendo que sem este conjunto de parâmetros a tarefa da visão não pode ser executada.

Outro fator que torna a visão uma tarefa difícil são os diferentes fatores que podem distorcer uma imagem e, conseqüentemente, confundir um dado observador. A aparência de um objeto pode ser influenciada pelo índice de reflexão luminosa do material de superfície, pelas condições atmosféricas, pelo ângulo da fonte de iluminação, pela luz do ambiente, pelo ângulo da câmera e suas características, entre outros fatores. Todos esses fatores somados podem contribuir para o resultado da representação de uma cena e, além disso, é muito difícil saber qual é a contribuição de cada fator (após a digitalização da imagem) na representação de cada pixel (CRUZ, 2004, p.15-16).

2.4 IMAGEM DIGITAL

Conforme cita Estevam (2003, p. 25), uma imagem natural pode ser caracterizada por uma variação contínua de tons e cores. No caso de uma fotografia, por exemplo, os tons variam de claro a escuro e as cores variam de vermelho a azul, abrangendo desta forma todo o espectro de cores visíveis.

Já uma imagem digital é composta por pontos discretos de tons e/ou cores, ou brilho, e não por uma variação contínua. Para a criação de uma imagem digital deve-se dividir a imagem contínua em uma série de pontos que irão possuir uma determinada tonalidade ou cor (ESTEVAM, 2003, p. 25).

Pode-se considerar uma imagem digitalizada como sendo uma matriz onde os índices de linhas e colunas identificam um determinado ponto na imagem e o correspondente valor deste elemento determina o brilho médio amostrado. Aos elementos desta matriz digital dá-se o nome de *pixels* (GONZALES; WOODS, 2000, p. 22).

Imagens digitais podem ser armazenadas em dispositivos ou transmitidas por diversos meios – quando isso é feito, a estrutura que compõe uma imagem é codificada usando algum formato específico para armazenamento ou transmissão e decodificada para leitura para a memória. Existem diversos tipos de formatos de armazenamento de imagens digitais, entre eles, TIFF (*Tagged Image File Format*), PNG (*Portable Network Graphics*), JPEG (*Joint Photographic Experts Group*) e outros (SANTOS, 2004, p. 94).

2.4.1 Propriedades de uma imagem digital

Gonzalez e Woods (2000, p. 21) caracterizam uma imagem por uma função bidimensional de intensidade da luz como $f(x, y)$, onde x e y denotam as coordenadas espaciais e o valor de f em qualquer ponto (x, y) é proporcional ao brilho (ou níveis de cinza) da imagem naquele ponto.

2.4.1.1 Vizinhança

Um *pixel* de coordenadas (x, y), tem quatro vizinhos horizontais e verticais, cujas coordenadas são $(x+1, y), (x-1, y), (x, y+1), (x, y-1)$. Esses *pixels* formam a chamada "4-vizinhança". Os quatro vizinhos diagonais de um *pixel* são os de coordenadas $(x-1, y-1), (x-1, y+1), (x+1, y-1), (x+1, y+1)$ (FILHO;NETO, 1999, p. 25).

2.4.1.2 Conectividade

A conectividade entre *pixels* é um importante conceito usado para estabelecer limites de objetos e componentes de regiões de uma imagem. Para se estabelecer se dois *pixels* estão conectados, é necessário determinar se eles são adjacentes segundo algum critério e se seus níveis de cinza satisfazem a um determinado critério de similaridade. Por exemplo, em uma imagem binária, onde os *pixels* podem assumir valores 0 e 1, dois *pixels* podem ser 4-vizinhos, mas somente serão considerados 4-conectados se possuírem o mesmo valor (FILHO; NETO, 1999, p. 26).

2.4.2 Operações lógicas e aritméticas

Sabemos que após uma imagem ter sido adquirida e digitalizada, ela pode ser vista como uma matriz de inteiros e, portanto pode ser manipulada numericamente utilizando operações lógicas e/ou aritméticas. Estas operações podem ser efetuadas *pixel* a *pixel* ou orientadas a vizinhança (FILHO; NETO, 1999, p. 28).

Ao serem executadas as operações aritméticas sobre imagens, devem-se tomar cuidados especiais com os problemas de *underflow* ou *overflow* do resultado. Para contornar esses problemas, existem basicamente duas alternativas: manter os resultados intermediários em uma matriz na qual o espaço em memória alocado para cada pixel permita a representação de números negativos e/ou maiores que 255 e em seguida proceder a uma normalização destes valores intermediários; ou truncar os valores maiores que o máximo valor permitido, bem como os valores negativos, igualando-os a 255 e 0 (FILHO; NETO, 1999, p. 29).

Todas as operações lógicas conhecidas podem ser aplicadas entre imagens, inclusive a

operação de complemento (NOT). Operações lógicas podem ser efetuadas em imagens com qualquer número de níveis de cinza mais são melhor compreendidos quando vistas em imagens binárias (FILHO; NETO, 1999, p. 31).

2.4.3 Representação de imagens digitais em Java

Para Santos (2004, p. 94) de uma forma geral uma imagem digital é representada em Java usando-se uma instância de classe que representa uma imagem. Esta classe, por sua vez, contém instâncias de duas classes (ou herdeiras) que são:

a) uma instância de `Raster`, que contém os valores dos *pixels* da imagem. Um `Raster`, por sua vez, é representado por uma instância de `DataBuffer`, que contém os valores dos *pixels* e de `SampleModel`, que indica como estes valores são organizados na memória;
b) uma instância de `ColorModel`, que indica como os valores dos *pixels* serão interpretados para renderização da imagem em dispositivos. Esta instância pode conter uma referência a instância de `ColorSpace` que representa alguns dos tipos básicos de espaço de cores.

Conforme cita Santos (2004, p. 95), tanto a API (Application Programming Interface) básica de representação de imagens em Java quanto a API Java Advanced Imaging (JAI) contém métodos para criar e manipular imagens usando instâncias destas classes. Um ponto que gera alguma confusão é a existência de várias classes e interfaces, tanto em Java quanto na API JAI, que podem ser usadas para representar imagens. Algumas dessas classes e interfaces são `BufferedImage`, `RenderedImage`, `PlanarImage` e `TiledImage`.

2.5 LASER

O *laser* é um dispositivo que funciona baseado num fenômeno: inversão de população, ou seja, absorção de energia para que a maior parte dos átomos se excitem. Após a inversão da população, deve haver um regresso ao estado fundamental com liberação de fótons gêmeos. Um *laser* é montado de maneira que a emissão espontânea seja minimizada e

substituída por uma forma "organizada" de emitir luz: emissão estimulada. Para se conseguir tal emissão, os *lasers* apresentam três componentes principais: o meio ativo, o mecanismo de bombeio e o sistema de realimentação (MENDONÇA, 1997, p. 88).

A palavra *laser* é formada pelas iniciais de *Light Amplitication by Stimulated Emission of Radiation* (amplificação da luz por emissão estimulada de radiação). O princípio básico de funcionamento do *laser* está baseado nas leis fundamentais da interação da radiação luminosa, nas quais, quando um elétron se move para uma órbita com um maior nível de energia, tem uma forte tendência de voltar ao estado fundamental e quando isso ocorre libera energia na forma de um fóton (partícula de luz). Qualquer fonte de energia luminosa atinge este estágio graças aos elétrons que mudam de órbita e liberam fótons. Porém, quando ocorre de maneira espontânea, este retorno é demorado. No caso do *laser*, a emissão é estimulada e antecipada com a ajuda de um agente externo, outro fóton. Portanto, a emissão estimulada resulta em dois fótons idênticos: um emitido pelo átomo excitado ao voltar ao seu estado de energia mais baixo e o próprio fóton que acelerou ou estimulou este processo (BAGNATO, 2005, p. 01).

Conforme Buscariollo (2008, p. 37), funcionando como uma fonte de luz com características únicas, o *laser* possui propriedades especiais que o tornam um excelente instrumento de uso científico. As principais características que permitem a sua utilização são:

a) a luz é monocromática, ou seja, tem uma só cor ou comprimento de onda, enquanto uma fonte de luz incandescente é formada por vários comprimentos de ondas. Este caráter monocromático da luz *laser* vem do fato de a energia carregada pelo fóton estimulante e pelo fóton emitido serem as mesmas;
b) a potência do feixe *laser* pode ser muito grande, ao contrário das fontes de luz convencionais. Pode atingir trilhões de watts nos chamados *lasers* pulsados, em que a energia acumulada por um longo tempo é emitida toda em um intervalo de tempo muito curto, da ordem de trilionésimos de segundos (10^{-12}s) ou menor;
c) o feixe resultante é colimado, ou seja, propaga-se na mesma direção, havendo um mínimo de divergência. Esta característica é extremamente importante para a aplicação num sistema de visão computacional;
d) a luz *laser* é coerente. Isso ocorre porque as diferentes porções sucessivas de uma mesma onda luminosa oscilam para cima e para baixo de forma sincronizada.

Há diferentes tipos de *laser* que dependem exclusivamente do composto existente no meio ativo, o qual pode ser em sua maioria do tipo sólido, gasoso ou dispositivo de estado sólido (PIAZZA, 2008, p. 05).

Os *lasers* do tipo gasoso são excitados por uma descarga elétrica no interior de um

tubo adequadamente pressurizado que contenha o meio ativo (composto gasoso) e a cavidade ressonante afastada de uma distância que varia entre 5cm até 5m. Entre os mais eficientes está o *laser* de dióxido de carbono operando na faixa do infravermelho (PIAZZA, 2008, p. 21).

Os *lasers* do estado sólido são feitos de cristais e vidros isolantes. Esses *lasers* são excitados por lâmpadas. O *laser* de rubi, que foi o primeiro a ser construído, deu lugar a *lasers* sólidos mais eficientes à base de neodímio (PIAZZA, 2008, p. 22).

Buscariollo (2008, p. 138) cita que atualmente as aplicações industriais do *laser* são diversificadas, mas certamente suas utilizações como instrumento de corte, marcação e solda são as mais amplamente difundidas. Como instrumento de furo e corte, a vantagem do *laser* reside no fato de ele evapora o material no local do furo ou na linha de corte, removendo automaticamente o subproduto, sem deixar vestígios. Isso o torna mais preciso que outros meios mecânicos.

Hoje, é praticamente impossível um campo das ciências experimentais que não tenha algum uso para o *laser*. Na física, a pesquisa sobre o *laser* é uma área por si só. Normalmente denominada óptica quântica, ela se dedica exclusivamente ao estudo do desenvolvimento de teorias e modelos que expliquem as inúmeras propriedades dessa radiação e de sua interação com a matéria. O *laser* permite ainda controlar o movimento de átomos, produzindo a chamada física dos átomos frios, na qual tem sido possível realizar experimentos inéditos que revelam a natureza quântica da matéria. As técnicas de manipulação de átomos com luz fizeram surgir a chamada computação quântica. Na biologia, o *laser* ganhou terreno com as chamadas pinças ópticas, que são feixes de luz que agem como pinças mecânicas e que possibilitam movimentar ou segurar organelas celulares, por exemplo, e com técnicas modernas da microscopia (BAGNATO, 2005, p. 33).

Uma das grandes aplicações atuais do *laser* esta em seu uso nas telecomunicações. Que a luz é capaz de transmitir muito mais informações que a corrente elétrica, isto já se sabia havia muito tempo. O principal problema era que a tecnologia não estava avançando o suficiente para permitir a implementação dessa idéia. Com o advento do *laser*, esse problema foi resolvido em parte, e a transferência de informações via luz começou a despertar interesse, embora de forma modesta (BAGNATO, 1999, p. 34).

O interesse em utilizar a visão computacional combinada com *laser*, como um método de sensoriamento, tem crescido nos últimos anos devido a sua simplicidade e baixo custo.

2.6 JAVA APPLICATION PROGRAMMING INTERFACE (API)

Uma aplicação desenvolvida em Java pode utilizar várias bibliotecas nativas desenvolvidas pela Sun, e o conjunto dessas bibliotecas Java é conhecido como Java *Application Programming Interface* (API). Nessas bibliotecas estão várias classes, organizadas em pacotes. Cada um desses pacotes traz classes com funcionalidades básicas e vitais para um determinado ramo de programação Java. Os pacotes de uso mais comum em aplicações básicas são:

a) `java.applet`: que contém basicamente as classes necessárias para criação de um aplicativo a ser rodado dentro de um *web browser*;
b) `java.awt`: contém classes relacionadas à interface gráfica;
c) `java.io`: classes para entrada e saída de dados das mais variadas formas;
d) `java.lang`: são as classes que dão suporte ao modelo computacional da linguagem;
e) `java.net`: contém classes aptas a estabelecer conexões de rede;
f) `java.util`: contém uma série diversa de classes de apoio ao programador, como estruturas de dados básicas, referências à data do sistema, gerador de números randômicos, etc.

Dentre diversos conjuntos de classes e métodos com funcionalidades padrões já implementadas, estão descritas a seguir *Java Advanced Imaging* (JAI) e *Java Media Framework* (JMF), por serem bibliotecas especificas para captura e tratamento de imagem.

2.6.1 JAI

O JAI incluiu a plataforma Java 2D, além de permitir sofisticados processamentos de imagens com alto desempenho para serem incorporados a *applets* Java e em aplicações Java. O JAI encapsula os formatos dos dados das imagens e os métodos remotos de chamada, permitindo assim que um arquivo de imagem, uma imagem de um objeto de rede ou um simples fluxo real de tempo sejam processados de maneira idêntica. Entretanto, o JAI representa um simples modelo de programação que oculta a complexidade dos mecanismos internos.

A verdadeira intenção do JAI é suprir todas as necessidades das aplicações de

processamento de imagens, eliminando assim todas as funcionalidades que os desenvolvedores precisavam incorporar dentro de suas *applets* e de suas aplicações (MIRANDA, 2006, p. 15).

A JAI por ser baseada em Java, é uma biblioteca flexível, eficiente, orientada a objetos e possui interdependência de plataforma. Suas operações variam desde a aplicação de convoluções, criação de histogramas da imagem, inversão de cores da imagem, detecção de bordas com operador de gradiente, entre outras. Tais operações já estão definidas na JAI e prontas para serem usadas (MORGAN, 2008, p. 56).

Segundo Miranda (2006, p. 16), o JAI apresenta algumas vantagens em relação às outras soluções de processamento de imagens. Entre as quais se destacam: interoperabilidade, imagens distribuídas, orientação a objetos, flexibilidade e extensibilidade, dispositivos independentes, multi-plataforma, alto desempenho e o alto poder de processamento. Dentre as diversas vantagens existentes pode-se destacar a característica multi-plataforma, pois a maioria das outras APIs são desenvolvidas para um sistema operacional específico, porém a API em questão, segue o modelo de biblioteca adotada pelo Java, propiciando que a mesma funcione em qualquer plataforma. Outra característica que merece devida importância é a orientação a objetos. Nesta API as imagens e o processamento de imagens são definidos como objetos. O JAI unifica noções de imagens e operações fazendo as subclasses terem uma hierarquia em comum.

Conforme Jankowski (2001), a JAI possui aproximadamente 80 operadores com suporte nativo e desempenho otimizado para o processamento de imagens. Em Sun (2010a), podem-se encontrar tais operadores separados por categorias, que incluem: a categoria de operadores geométricos, de área, de pontuais, estatísticos, de arquivos, categoria de operadores, de extração de bordas, entre outros.

O poder de processamento do JAI é explicitado pela capacidade de suporte a complexos formatos de imagens, incluindo imagens de três dimensões. Muitas classes de algoritmos de imagens são suportadas diretamente enquanto outras são adicionadas à medida que forem sendo necessárias. Essa API também possui como benefício à interoperabilidade, pois o JAI integra-se com as outras APIs do Java como o Java 3D. Isso permite que imagens sofisticadas sejam parte da caixa de ferramenta dos programadores da tecnologia Java. Essa API é classificada como um padrão de extensão da plataforma da plataforma Java. Ela disponibiliza funcionalidades que são compatíveis com essas classes na maioria dos casos (MIRANDA, 2006, p. 17).

2.6.2 JMF

O JMF é um framework que adiciona funcionalidades de multimídia em aplicações Java tanto para desktops quanto em *applets*. Construído sob uma arquitetura que é facilmente estendida, podendo ser adicionado diferentes tipos de mídias de entrada, de processamento ou de *players*.

O JMF encontra-se atualmente na versão 2.1.1, onde a partir dela há a possibilidade de se acessar diretamente os dados que estão sendo capturados ou que estão sendo processados dando assim uma maior capacidade de extensão de novas funcionalidades, como aplicação de efeitos em sons e vídeos, *codecs*, entre outros. A nova versão é totalmente compatível com a anterior, JMF 1.0.

Os principais formatos de media suportados incluem AVI, MIDI, AU, MPEG, QuickTime e WAV, utilizando como meio de transmissão e recepção de pacotes os protocolos RTP e RTCP, disponíveis também na API do JMF.

Um aspecto importante quanto à utilização do JMF é que essa API serve de base para a especificação da API JavaTV, criado para desenvolvimento de conteúdo destinado à televisão interativa, sendo boa parte integrante do padrão de conteúdo para TV digital (FAUSTO; COPETTI, 2007, p. 25).

Segundo Fausto e Copetti (2007, p.26), quanto a arquitetura de composição definida para o JMF, encontra-se as seguintes partes:

a) `DataSource`: representa os dados de mídia propriamente ditos, podendo ser, por exemplo, *stream* de áudio, stream de vídeo ou a combinação dos dois;
b) `CaptureDevice`: representa a fonte de dados, como por exemplo, microfone ou uma *webcam*, sendo responsável por capturar dados multimídia. Sua saída é o `DataSource` que pode ser direcionado para um player, ou um processo de conversão para outro formato, entre outros;
c) `Player`: recebe a entrada de dados multimídia realizando a exibição em screen no caso de vídeo e execução de sons, da parte de áudio;
d) `Processor`: é um tipo de player, porem sua saída de processamento é colocado em um `DataSource` de saída e este por sua vez poderá servir de entrada para um player ou até mesmo outro processor;
e) `DataSink`: é responsável por capturar e manipular os dados contidos nos `DataSources`. Um exemplo seria um objeto relacionado à gravação de

`DataSource` de um vídeo para arquivo;

f) `Format`: responsável pela representação do formato de áudio e vídeo. São fornecidos por padrão os formatos H.261, H.263, JPEG, RGB, YUV e `IndexedColor`;

g) `Manager`: uma espécie de facilitador de criação de objetos como `Player`, `Processor`, `DataSource` e `DataSink`, bastando para isso, informar um deles e obtendo-se o desejado já associados para trabalharem. Tem a responsabilidade também de manter registro das implementações e extensões feitas para o JMF, os dispositivos de capturas existentes no sistema e plug-ins de processamento.

2.7 TRABALHOS CORRELATOS

A seguir são listadas referências a alguns trabalhos correlatos ao tema proposto neste trabalho, como o protótipo apresentado por Estevam (2003), o projeto ARGO desenvolvido por Fascioli e Broggi (1999) e o projeto descrito por Gavrila e Munder (2007), Protector.

Estevam (2003) apresenta um protótipo de um veículo autônomo terrestre dotado de um sistema óptico para rastreamento de trajetória, onde através de uma câmera, o veículo identifica uma linha guia, corrige desvios em relação à trajetória pretendida e desloca-se automaticamente em um ambiente pré-modelado, ou seja, um ambiente isento de variações que deveriam ser consideradas para aplicações do mundo real, como por exemplo, a ausência de luminosidade, variedade de cores e outros.

Fascioli e Broggi (1999) desenvolveram um projeto, denominado ARGO, onde é utilizada uma técnica para detecção de objetos baseada em visão binocular. Basicamente, a técnica consiste em desfazer a perspectiva em cada uma das imagens, e calcular as disparidades entre elas nas imagens remapeadas. A captura de imagens é realizada através de duas câmeras as quais permitem extrair informações tanto da estrada como do ambiente por onde se movimenta o veículo. Através desse sistema são detectados e localizados os obstáculos ao longo da estrada, enquanto o processamento da imagem capturada permite extrair a geometria da estrada que fica na frente do veículo. Além disso o veículo está equipado com uma placa de entradas e saídas usada para fazer aquisição de dados de velocidade.

Gavrila e Munder (2007), pesquisadores da DaimlerChrysler, desenvolveram o

Protector. Trata-se de sistema de detecção de pedestres baseado em visão estereoscópica. Neste projeto um mapa de profundidades é calculado inicialmente e algoritmos de busca por forma são utilizados para detectar os pedestres. Finalmente, o *bounding box* de cada pedestre é utilizado para o rastreamento ao longo do tempo.

3 DESENVOLVIMENTO

Este capítulo tem como objetivo detalhar as etapas de desenvolvimento do software. São apresentados os requisitos principais, a especificação e a implementação, demonstrando técnicas, ferramentas utilizadas e a operacionalidade do software. Para finalizar são comentados os resultados.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os requisitos apresentados encontram-se classificados em Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF), os quais são:

a) o sistema deverá capturar imagens através de uma câmera (RF);
b) o sistema deverá criar marcos visuais através de raio *laser* (RF);
c) o sistema deverá tratar as imagens obtidas para detectar os marcos visuais criados pelos raio *laser* (RF);
d) o sistema deverá detectar obstáculos dentro do campo de visão (RF);
e) o sistema deverá calcular a distância de obstáculos (RF);
f) o sistema deverá ser implementado em Java, utilizando o ambiente de desenvolvimento Eclipse (RNF);

3.2 ESPECIFICAÇÃO

A especificação do software foi desenvolvida seguindo a notação *Unified Modeling Language* (OMG, 2005) em conjunto com o software Astah Community da Change Vision (CHANGE VISION, INC, 2006). Diagramas como de atividades e classes são utilizados para dar-se o entendimento do funcionamento do software, que recebeu o nome de SVCLaserCam. Este nome foi definido por ser um sistema de visão computacional utilizando raio *laser* e câmera de vídeo.

3.2.1 Diagrama de atividades

Os diagramas de atividades estão divididos em duas etapas, sendo na primeira demonstrada e explica as ações disponíveis em cada tela do sistema, e a segunda demonstra e explica as ações realizadas no procedimento principal do sistema, desde a captura da imagem, passando pela localização do laser, detecção do obstáculo e por fim cálculo da distância até o obstáculo detectado.

O diagrama ilustrado pela Fig. 1 apresenta as principais ações que um usuário pode executar ao utilizar o software. Das atividades disponíveis neste diagrama, apenas a atividade "Iniciar Captura" é executada na tela inicial do programa, as atividades "`Definir Cor do Laser`", "`Configurar`" e "`Calibrar`" são iniciadas em telas específicas, conforme as necessidades para atender a atividade.

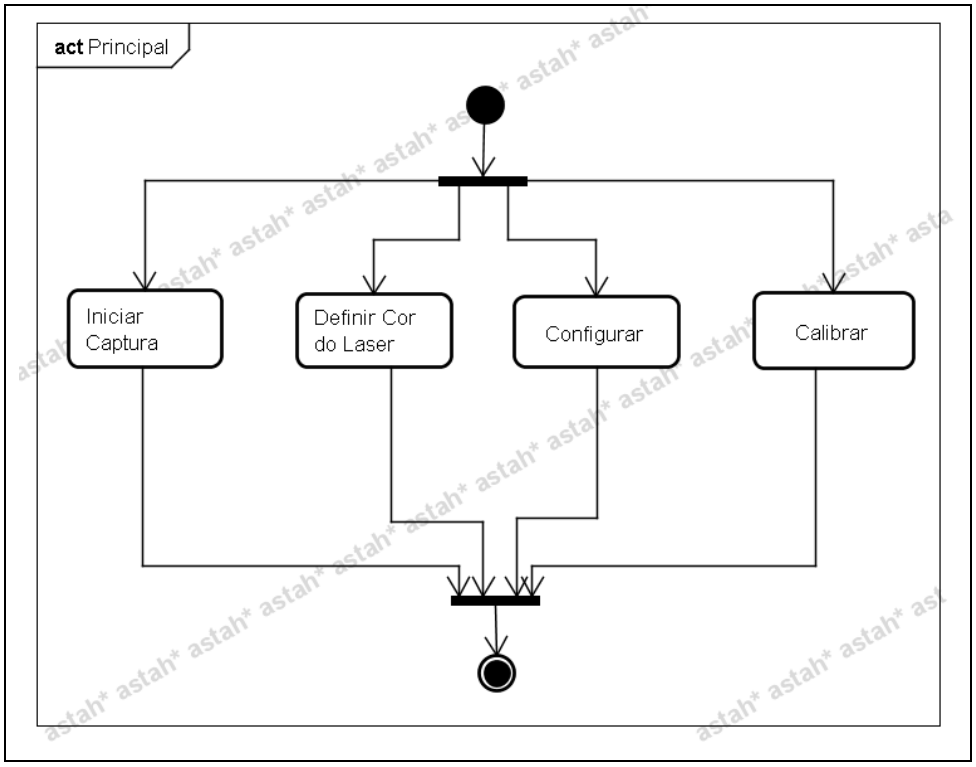

Figura 1 – Diagrama de atividades

A partir da atividade "`Iniciar Captura`", o usuário inicia o procedimento de captura e tratamento de imagens e detecção de obstáculos, bem como o cálculo da distância em que o veículo autônomo encontra-se de um obstáculo. Assim o veículo pode locomover-se e quando aproximar-se de um obstáculo, o sistema irá informar a distância atual até o obstáculo que se encontra no seu campo de visão.

As atividades "`Definir Cor do Laser`", "`Configurar`" e "`Calibrar`" são utilizadas

para definir as informações utilizadas pelo sistema para detecção de obstáculos.

O diagrama ilustrado na Fig. 2 demonstra as ações da atividade "`Definir Cor do Laser`". Como existem diversas cores de laser, para permitir uma portabilidade quanto ao laser que for escolhido para o sistema, este procedimento foi desenvolvido. Informar a cor do laser utilizada no sistema é obrigatório para funcionamento do sistema, pois através da cor informada que será feita a busca pelo marco visual criado com o laser na imagem capturada pela câmera de vídeo.

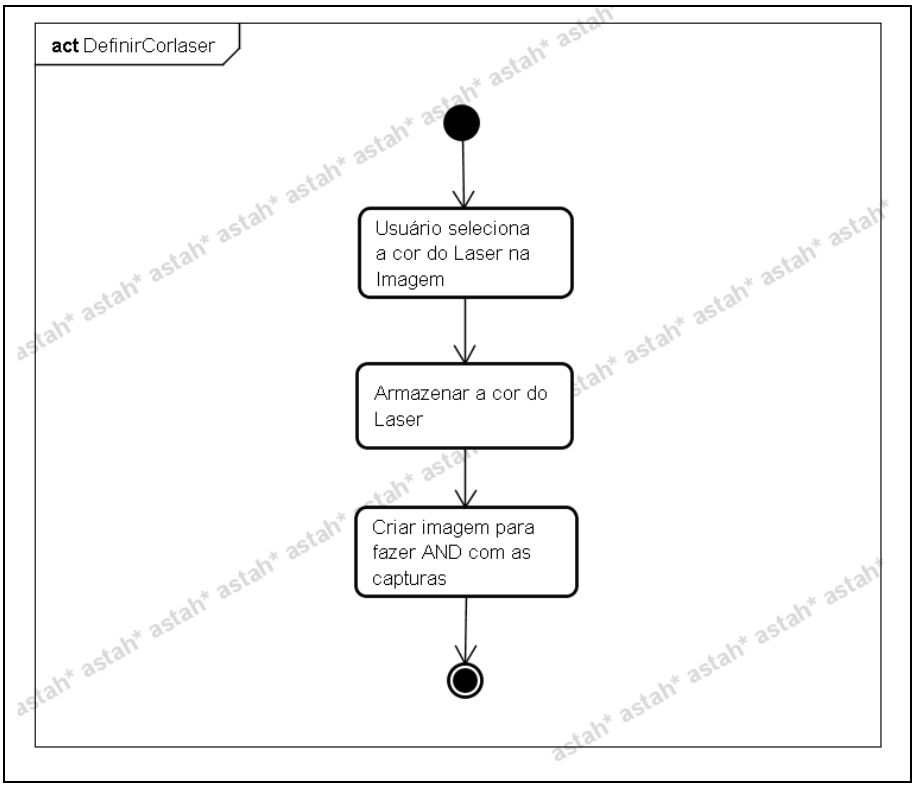

Figura 2 – Diagrama de atividades "`Definir Cor do Laser`"

O diagrama listrado na Fig. 3 demonstra as ações da atividade "`Configurar`", onde é possível definir as informações utilizadas nos cálculos para definir a distância dos obstáculos. Ao fechar a tela, é feito o cálculo utilizando as informações inseridas e armazenados os resultados em um arquivo de configurações. Os cálculos feitos após inseridas as configurações, são do ângulo em que o laser está direcionado e da posição do laser em centímetros na imagem usada para calibração.

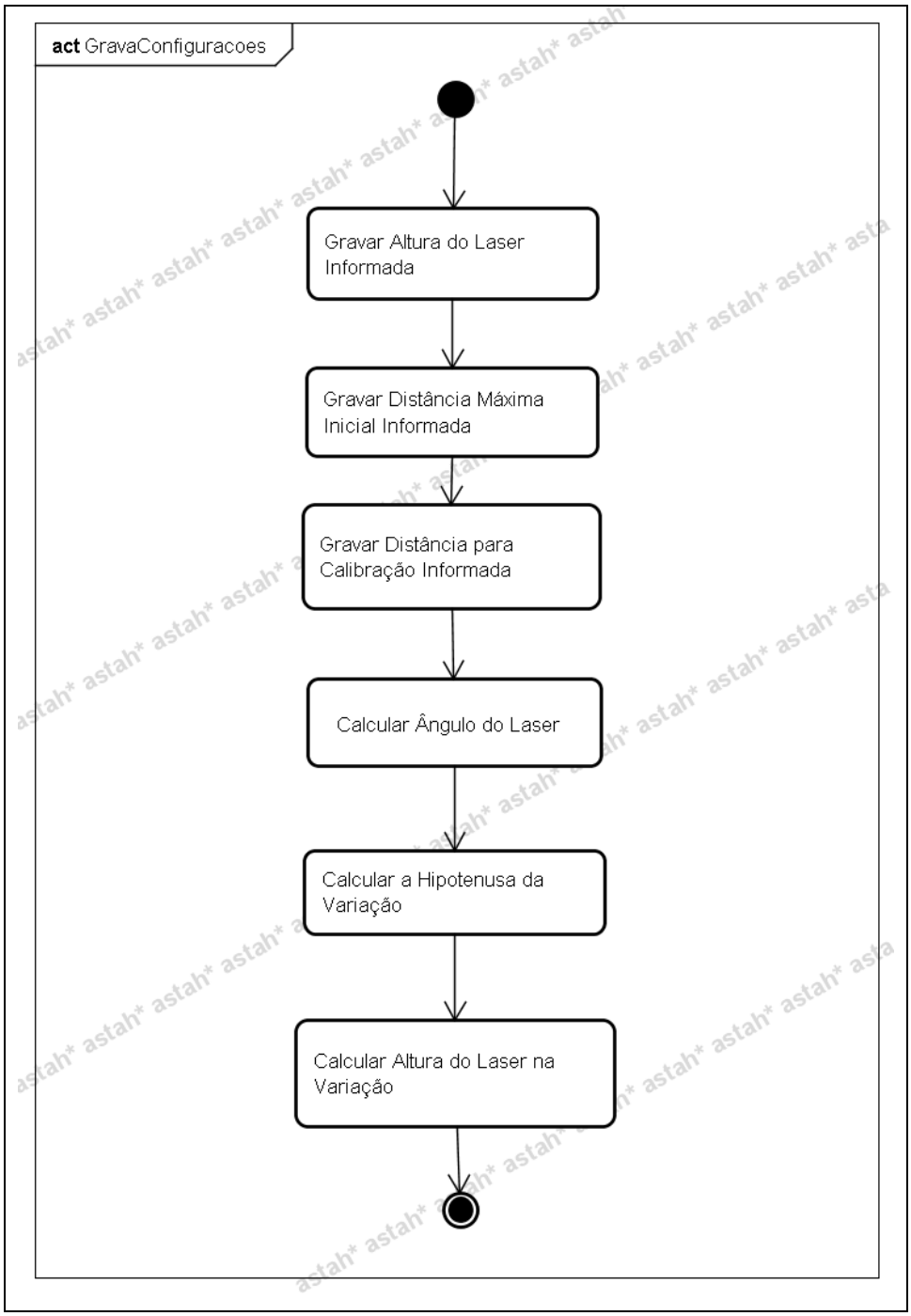

Figura 3 – Diagrama de atividades "`Configurar`"

Ao fechar a tela de configurações é feito o cálculo da altura do laser em centímetros e quando executa-se as ações da atividade "`Calibrar`", no arquivo de configurações é armazenado o *pixel* referente à posição da altura do laser na imagem capturada para calibração. O diagrama ilustrado na Fig. 4 apresenta o diagrama referente à atividade "`Calibrar`".

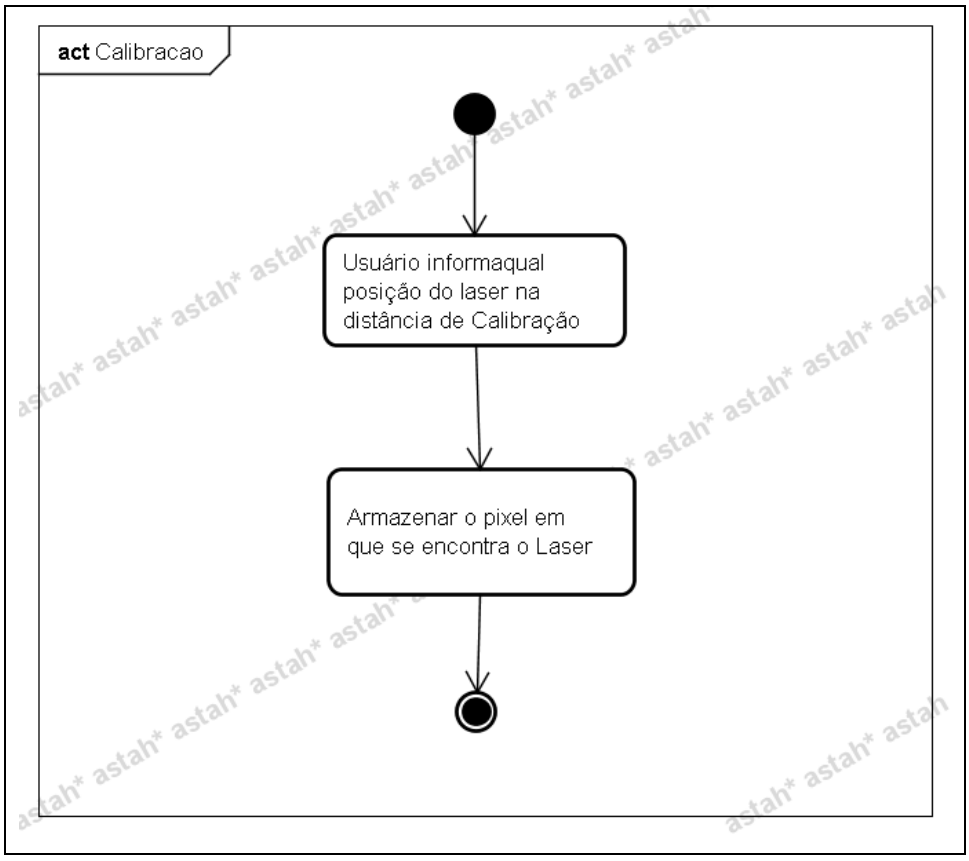

Figura 4 – Diagrama de atividades "`Calibrar`"

O diagrama ilustrado na Fig. 5 demonstra o procedimento principal, que é iniciado ao executar a atividade "`Iniciar Captura`". Este procedimento é o principal do sistema, ele utiliza todas as informações obtidas nas outras atividades.

Inicialmente é feito a captura da imagem através da câmera de vídeo, essa imagem capturada é usada numa operação AND com uma imagem na mesma cor do laser. Com esse AND é filtrada na imagem a posição exata do laser. Essa imagem com a cor exata do laser é utilizada para inicialmente encontrar-se o laser fixo na horizontal, que delimita o limite inferior do veículo. Encontrado esse laser, a imagem é recortada, para só ser utilizada na captura o que está acima do laser nos cálculos seguintes. Essa imagem recortada contém agora, não mais a imagem de dois lasers, apenas um, que é o laser que está fixado no veículo inclinado. E a partir dessa imagem é encontrado o centro do laser, que seria o ponto central do laser na imagem capturada. O valor do centro do laser é o valor correspondente à metade da soma dos extremos do laser, conforme pode ser observado na equação (1).

$$medioX = \frac{\max X - \min X}{2} \qquad medioY = \frac{\max Y - \min Y}{2} \tag{1}$$

Esse centro de massa calculado é em *pixel* e é ele que é utilizado na detecção de

obstáculos. O valor encontrado é a posição atual do laser, se for primeira captura ela apenas é armazenada para comparação com as próximas capturas. Se não for a primeira captura, ela é comparada com o valor anteriormente capturado, então é verificado se houve mudança no posicionamento da mesma. Se não houve mudança no posicionamento do laser entre as duas capturas, então não houve variação em relação a um obstáculo. Se houve mudança no posicionamento do laser, então significa que o veículo autônomo aproximou-se ou se afastou de um obstáculo. Havendo essa conclusão é realizado o cálculo da distância do objeto em centímetros, pois essa unidade de medida será muito mais entendível ao usuário do que a distância em *pixel*. Após calculada a distância do obstáculo em centímetros esta informação é passada para o usuário na tela do sistema.

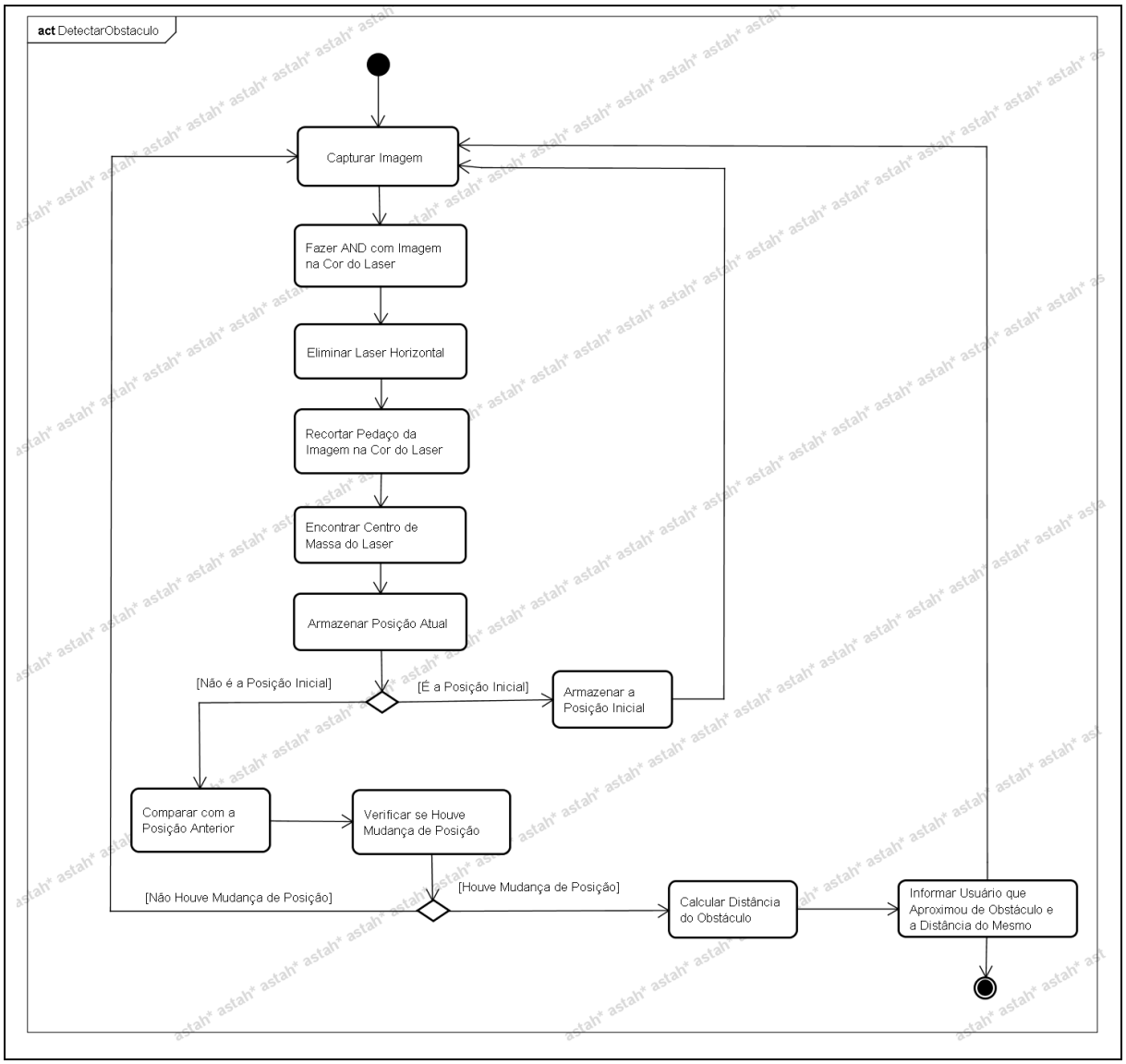

Figura 5 – Diagrama de atividades "`Iniciar captura`"

3.3 IMPLEMENTAÇÃO

Nesta seção primeiramente são apresentadas tecnologias utilizadas durante o desenvolvimento do software (item 3.3.1) em seguida mostrar como foram implementadas as funcionalidades do mesmo.

3.3.1 Técnicas e ferramentas utilizadas

Para a implementação da aplicação foi utilizada a linguagem de programação Java. O ambiente de desenvolvimento escolhido foi o Eclipse (ECLIPSE, 2010). Para o desenvolvimento da aplicação utilizou-se o framework JMF (SUN MICROSYSTEMS, 2010b), na captura de imagens e o framework JAI (SUN MICROSYSTEMS, 2010a), no tratamento das imagens capturadas.

Para testar a implementação, foram acoplados à um veículo de brinquedo, uma câmera de vídeo e dois módulos laser, o qual está ilustrado na Fig. 6. A câmera de vídeo é do modelo LG LIC-200 com resolução de 640 x 480. Os módulos laser são de 5mW, com alimentação na faixa de 3,5 volts a 4,5 volts, cada módulo foi ligado em série com um diodo, alimentados por uma fonte de 5 volts. Tanto a câmera de vídeo como os módulos laser são ligados ao computador através de portas USB.

Figura 6 – Veículo utilizado nos testes

3.3.2 Calibração do software

Para utilização do software implementado, inicialmente deve-se realizar calibrações necessárias tanto para definir informações sobre o laser como para definir informações necessárias para o cálculo da distância de um determinado obstáculo. Essa calibração subdivide-se em três etapas: definição da cor do laser, configuração das distâncias para calibração e calibração da posição do laser na imagem capturada. Estas três etapas estão descritas abaixo.

3.3.2.1 Definição da cor do laser

Esta etapa é fundamental para o funcionamento do software. Nela é definida a cor dos lasers a serem capturadas pela câmera de vídeo. Tendo em vista que se encontram disponíveis no mercado diferentes cores de laser (mais informações seção 2.5), é possível que o usuário selecione numa imagem capturada qual a cor do laser utilizado. Na tela principal do sistema, ao selecionar a opção "`Definir a Cor do Laser`", abrirá para o usuário uma nova tela, conforme podemos verificar na Fig. 7, onde o mesmo clica com o mouse, sobre a imagem capturada, na posição em que o laser se encontra na imagem capturada. Ao clicar sobre uma determinada região da imagem, é armazenada num arquivo de configurações a referência em RGB da cor do laser selecionado.

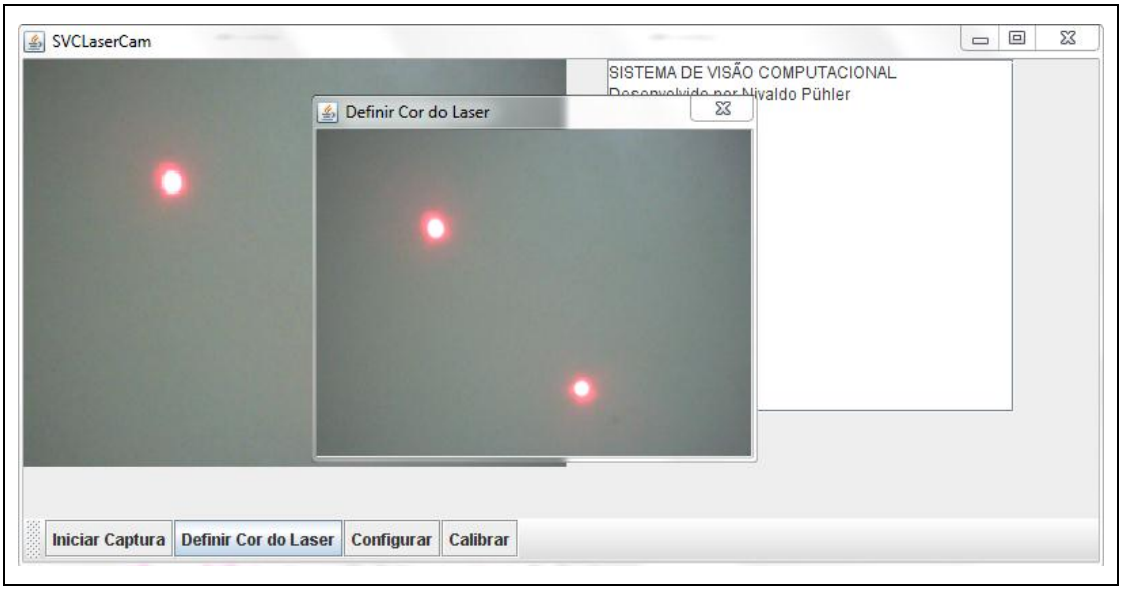

Figura 7 – Tela "`Definir Cor do Laser`"

3.3.2.2 Configuração das distâncias para calibração

Nesta etapa definem-se os parâmetros utilizados para o cálculo de calibração do sistema, para realizar o cálculo da distância de um obstáculo. Na tela principal do sistema, ao selecionar a opção "`Configurar`", abrirá uma nova tela para o usuário, ilustrada na Fig. 8, onde será possível informar a altura em que o laser está posicionado do veículo autônomo, a distância máxima em que será possível detectar um obstáculo com o veículo autônomo em questão e distância em que o veículo autônomo encontra-se no momento da calibração. Após serem informadas estas distâncias, elas são armazenadas no arquivo de configurações. Caso estas informações não forem preenchidas, influenciará no cálculo da distância de um obstáculo, pois a mesma não será calculada devido à falta dos parâmetros iniciais.

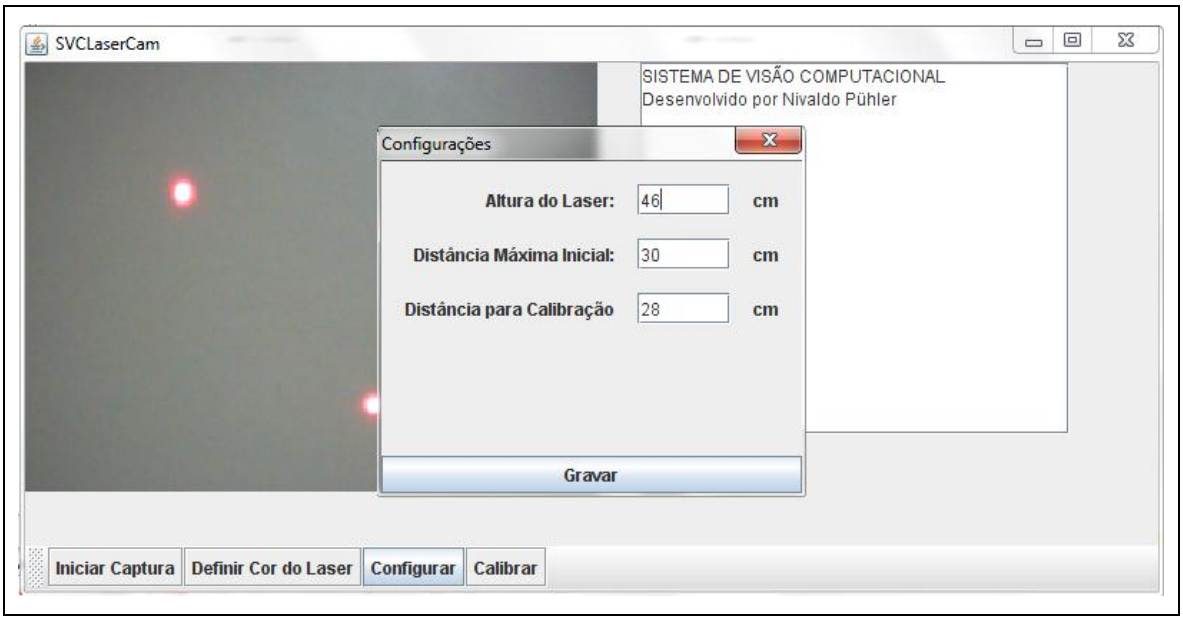

Figura 8 – Tela "`Configurar`"

3.3.2.3 Calibração da posição do laser na imagem capturada

Está etapa é responsável por detectar a posição do laser no momento da calibração, definindo então a relação entre a posição do *pixel* capturado e a distância do veículo autônomo de um obstáculo. Ao clicar em "`Calibrar`" na tela principal do sistema, o sistema faz uma captura da imagem na câmera, e calcula a posição atual do laser. Para que os cálculos sejam realizados de forma correta, é necessário que ao clicar em calibrar o veículo esteja na distante de um obstáculo, conforme valor informado na tela de Configuração, no campo

“Distância para Calibração”.

3.3.3 Capturando imagem e detectando obstáculos

Feitas as calibrações necessárias, o sistema está apto a detectar a presença de obstáculos e calcular a distância entre o veículo autônomo e o obstáculo detectado. Para perceber a presença de obstáculos através da localização dos lasers na imagem capturada, precisam-se definir algumas particularidades quanto ao posicionamento da câmera e dos lasers no veículo autônomo. O veículo autônomo deve conter uma câmera de vídeo para capturar as imagens a serem processadas.

Quanto aos lasers, o sistema necessita de dois lasers, cada um com sua função específica. Um laser deve ser fixo na horizontal, pois este será o responsável por delimitar o limite inferior de captura, indicando assim que os cálculos devem iniciar a partir dele. O que estiver abaixo do mesmo é descartado, não influenciando na detecção de obstáculos. Isso é necessário, pois a medida que o veículo vai movimentando-se, em determinados momentos, dentro do campo de visão da câmera quantidades diferentes de chão serão capturadas, pois conforme aproxima-se de um obstáculo, essa quantidade de chão vai diminuindo, conforme exemplo ilustrado na Fig. 9, onde inicia-se a captura a 100 cm do obstáculo, no caso uma parede, e aproxima-se com uma variação de 20 cm a cada captura.

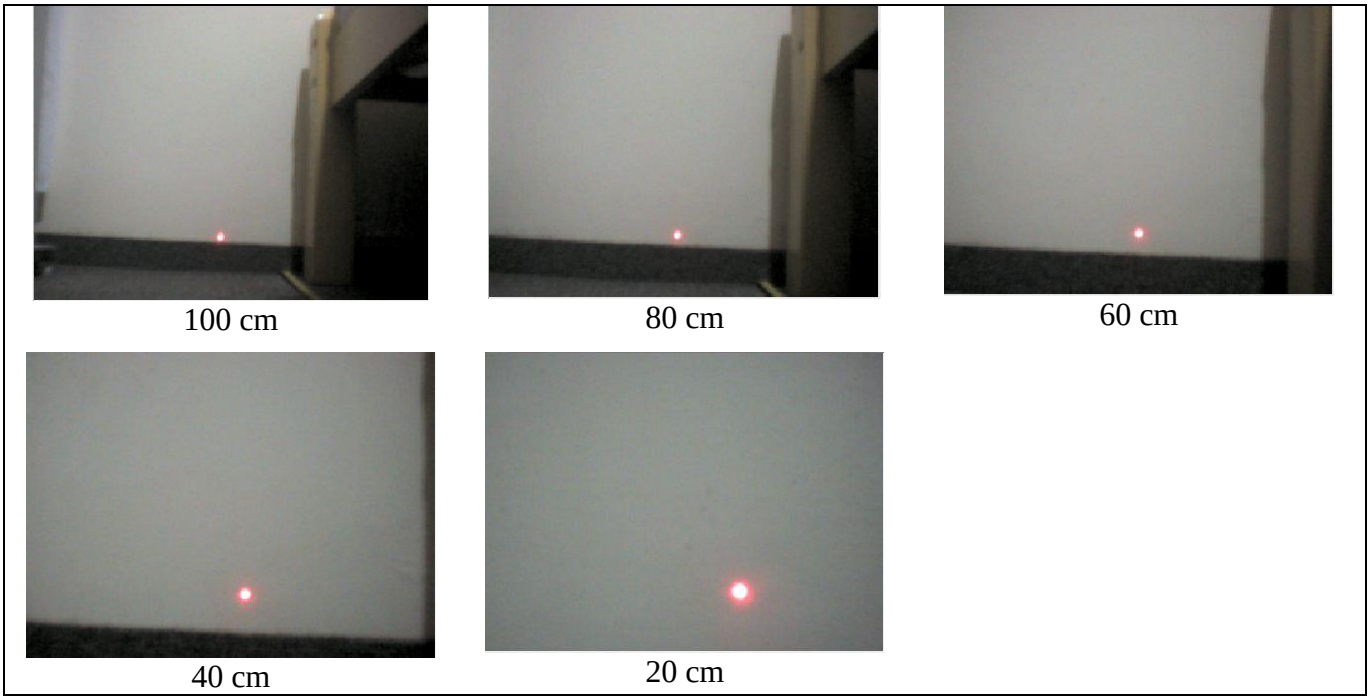

Figura 9 – Veículo autônomo aproximando de um obstáculo apenas com o laser na horizontal

O método descrito a seguir, no Quadro 1, é responsável por identificar o laser horizontal na imagem capturada, que é passada por parâmetro, e retornar uma imagem contendo somente o campo de visão que será utilizado pelo sistema, que é acima e a esquerda do laser que encontra-se na horizontal.

```
public PlanarImage EliminaLaserHorizontal(PlanarImage input, int[] corLaser){

      PlanarImage retorno = input;
      Raster ra = input.getData();
      int tamanhoX = ra.getWidth();
      int tamanhoY = ra.getHeight();
      int metadeX = tamanhoX / 2;
      int metadeY = tamanhoY / 2;
      int maiorX = ra.getMinX() -1;
      int maiorY = ra.getMinY() -1;
      int menorX = tamanhoX +1;
      int menorY = tamanhoY +1;

      for(int w=metadeX; w > ra.getMinX(); w--){
            for(int h=metadeY; h > ra.getMinY() ; h--){

                  int x = ra.getWidth() - w;
                  int y = ra.getHeight() - h;
                  int r = ra.getSample(x,y,0);
                  int g = ra.getSample(x,y,1);
                  int b = ra.getSample(x,y,2);
                  if( r == corLaser[0] && g == corLaser[1] && b == corLaser[2]){
                        if (menorX > x) {   menorX = x;};
                        if (maiorX < x) {   maiorX = x;};
                        if (menorY > y) {   menorY = y;};
                        if (maiorY < y) {   maiorY = y;};
                  }
            }
      }

      int areaX = menorX-ra.getMinX();
      int areaY = menorY-ra.getMinY();
      if (menorX>0 && menorY>0 && (maiorX-menorX)>0 && (maiorY-menorY)>0){

            retorno  =  fJAI.cortarImg(input,  ra.getMinX(),  ra.getMinY(),  areaX,
            areaY);
      }
            return retorno;
}
```

Quadro 1 – Implementação da função responsável por detectar o laser horizontal e eliminá-lo do campo de visão

O segundo laser a ser colocado no veículo autônomo deve ser fixo a uma determinada altura do chão, a qual deve ser informada ao sistema e deve estar inclinado, com o foco de luz emitido pelo laser, apontando para o chão. Através desta inclinação que poderemos determinar que o veículo esteja aproximando-se ou se afastando de um determinado obstáculo, pois quando estiver dentro do campo de captura, esse foco de laser movimentará no eixo y da imagem capturada.

A função que detecta o posicionamento do laser, através do cálculo do centro de massa

do mesmo, utiliza como base a imagem após já ser eliminado o laser que está na horizontal, pois assim, trabalha-se apenas, com o campo de visão gerado a partir do limite inferior demarcado pelo laser horizontal, que é o campo de visão a ser considerado pelo sistema. O Quadro 2, ilustra o trecho do código responsável por detectar o centro de massa do laser inclinado, utilizando como base a imagem sem o laser na horizontal, dentro do campo de visão considerado pelo sistema.

```
public int[] EncontrarCentroLaserAngulo(PlanarImage input, int[] corLaser){
      boolean encontrouLaser = false;
      input = EliminaLaserHorizontal(input, corLaser);
      Raster ra = input.getData();
      int tamanhoX = ra.getWidth();
      int tamanhoY = ra.getHeight();
      int maiorX = ra.getMinX() - 1;
      int maiorY = ra.getMinY() -1;
      int menorX = tamanhoX +1;
      int menorY = tamanhoY +1;
      int medioX = 0;
      int medioY = 0;
      int[] retorno = new int[2];

      for(int w=0; w < tamanhoX; w++){
            for(int h=0; h < tamanhoY; h++)

                  int x = ra.getMinX() + w;
                  int y = ra.getMinY() + h;
                  int r = ra.getSample(x,y,0);
                  int g = ra.getSample(x,y,1);
                  int b = ra.getSample(x,y,2);
                  if( r == corLaser[0] && g == corLaser[1] && b == corLaser[2]){
                        encontrouLaser = true;
                        if (menorX > x) {   menorX = x;};
                        if (menorY > y) {   menorY = y;};
                        if (maiorX < x) {   maiorX = x;};
                        if (maiorY < y) {   maiorY = y;};
                  }
            }
          }
```

Quadro 2 – Implementação da função responsável por detectar o laser inclinado, parte 1

```
      if (encontrouLaser){
          medioX = ((maiorX+menorX)/2);
          medioY = ((maiorY+menorY)/2);
          if (menorX>0 && menorY>0 && (maiorX-menorX)>0 && (maiorY-menorY)>0){
            imgJAI.set(fJAI.cortarImg(input,menorX,menorY,(maiorX-menorX),(maiorY-
menorY)));
          }
          if ( (atual[1]+2) == medioY  || (atual[1]-2) == medioY
                || (atual[1]+1) == medioY || (atual[1]-1) == medioY){
                  retorno[0] = medioX;                retorno[1] = atual[1];
          }else{
                  retorno[0] = medioX;                retorno[1] = medioY;
          }
        }else{
            retorno[0] = 0;                    retorno[1] = 0;
        }
      return retorno;
}
```

Quadro 3 – Implementação da função responsável por detectar o laser inclinado, parte 2

A cada captura realizada pela câmera, são executadas as funções acima descritas, a fim de encontrar o campo de visão do sistema através do laser horizontal, e detectar movimentação, através da variação do laser inclinado, comparando sempre a captura atual, com a captura anterior. Como está inclinado em direção ao chão, se o centro de massa do laser estiver mais próximo ao limite inferior do campo de visão, significa que o veículo está se afastando do obstáculo. De maneira semelhante caso o centro de massa do laser estiver mais afastado do chão, próximo ao limite superior do campo de visão significa que o veículo está aproximando-se de um obstáculo.

3.3.4 Calculando a distância do obstáculo

Após ser detectado que o veículo autônomo está aproximando-se ou se afastando de um obstáculo, realiza-se o cálculo para determinar essa distância em centímetros. Para realização desse cálculo considera-se os dados informados na calibração.

O sistema utiliza-se como base para o cálculo de distância, a relação entre a mudança

de posição do *pixel* e a distância do obstáculo. Na calibração informamos a Distância Inicial Máxima e a Distância para calibração. A diferença entre os dois valores é o resultado da distância percorrida para calibrar o veículo autônomo. Também na calibração, o *pixel* que representa a posição do laser no momento da calibração é capturado. Diminuindo o limite máximo do campo de visão do *pixel* atual, é possível encontrar a distância que esse *pixel* percorreu na imagem capturada. Unido as duas informações pode-se estabelecer esta relação. Assim para utilização nos cálculos, pode-se formar a equação (2), onde *relDesloc* é Relação de Deslocamento, que define quantos centímetros percorreu para estar no *pixel* capturado na calibração.

$$relDesloc = \frac{distMaxInicial - distCalibração}{pixelMaxCampoVisao - pixelCalibracao} \quad (2)$$

A partir da relação definida acima, no momento em que foi detectado o deslocamento pode-se calcular a distância do veículo autônomo do obstáculo, partindo do seguinte princípio: a detecção de obstáculos, processa a imagem e retorna o *pixel* atual (mais informações no item 3.3.2). Através do *pixel* retornado por esta função, posso obter a distância atual, em centímetros, do veículo autônomo em relação ao obstáculo. A fórmula utilizada para calcular essa distância atual, pode ser visualizada na equação (3).

$$distPercorrida = relDesloc * (pixelMaxCampoVisao - pixelAtual) \quad (3)$$

Assim após capturar a imagem, definir o campo de visão do sistema, e encontrar a posição atual do *pixel* que representa o centro de massa do laser inclinado, pode-se determinar a distância que o veículo autônomo encontra-se de um obstáculo.

3.4 RESULTADOS E DISCUSSÃO

O presente trabalho apresentou resultados satisfatórios, pois torna-se possível a utilização do sistema para através de alternativas de baixo custo determinar a distância de um veículo autônomo a um obstáculo. Ainda foi possível através do presente trabalho, determinar um modelo que pode servir como base para trabalhos futuros.

Para determinar como seria realizado o cálculo de distância do obstáculo, iniciou-se um estudo de qual melhor posicionamento do laser e da câmera. Após diversos testes serem realizados, pode-se chegar ao seguinte modelo: a câmera deveria ficar na horizontal, pois assim mesmo que o laser incida sobre um obstáculo inclinado, a captura ainda assim, será de

aproximada com a captura do laser incidindo sobre um obstáculo sem inclinação. O laser deveria ficar posicionado a uma determinada altura acima da câmera e inclinado em direção ao chão. Pois assim formaria um triângulo (conforme pode ser visualizado na Fig. 10) e o cálculo da distância do obstáculo, poderia ser possível calcular a distância do obstáculo, utilizando equivalência de triângulos.

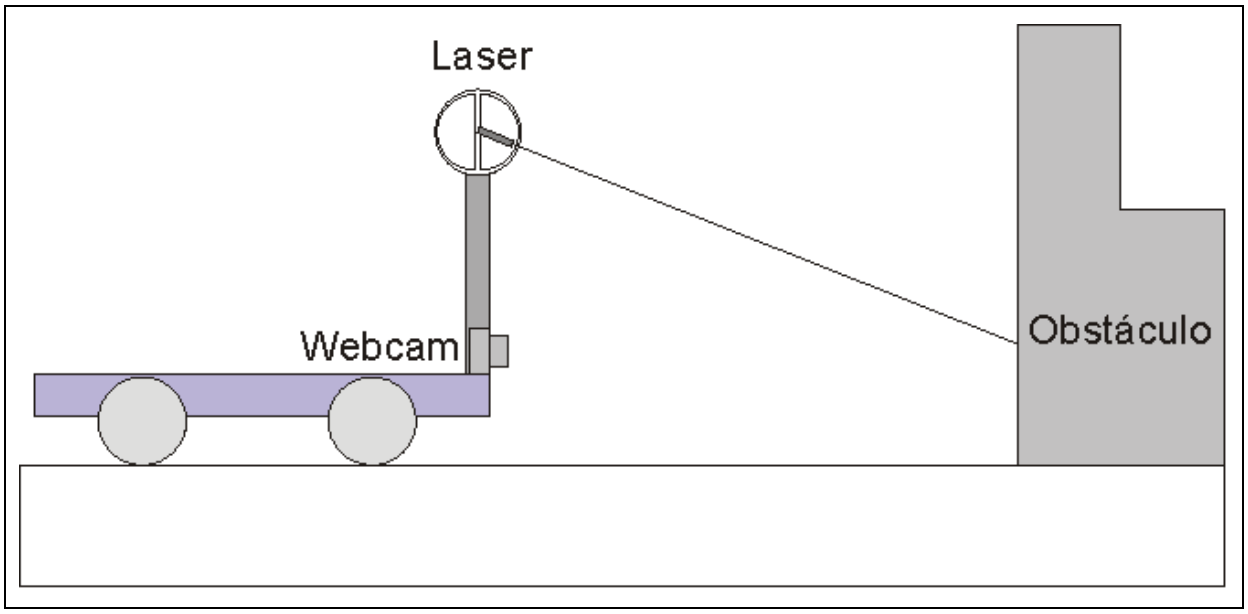

Figura 10 – Modelo inicial definido para obter a distância do obstáculo

Após determinar o modelo conforme a Fig. 10, teve inicio a implementação do sistema, baseando-se nesse modelo. Durante a implementação verificou-se a necessidade da implantação de um segundo laser. Este laser seria responsável por definir o limite inferior do veículo autônomo. Isso tornou-se necessário pois durante a implementação verificou-se que conforme o veiculo autônomo movimentava-se, a relação com o chão modificava-se, tornando assim variável, portanto, não demonstraria valores exatos nos cálculos. A partir dessa concepção pode-se chegar ao modelo conforme ilustrado na Fig. 11.

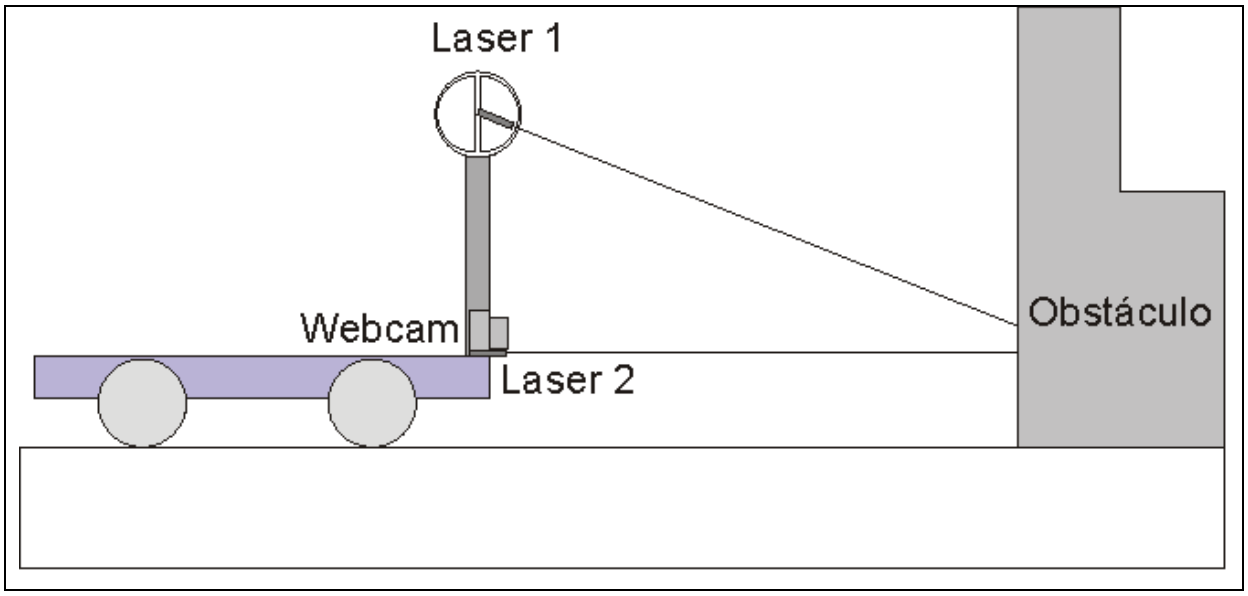

Figura 11 – Modelo definido para obter a distância do obstáculo com laser que define o limite inferior do veículo

Com o modelo acima, pode-se verificar bem a formação de um triângulo, tendo como hipotenusa a reta entre o laser inclinado e seu foco, incidindo sobre um objeto, e como catetos a base que coloca o laser a uma determinada distância da câmera, que é informada na calibração, e a reta entre o laser na horizontal e o ponto do mesmo incidindo sobre um obstáculo. A medida que aproxima-se de um obstáculo a partir da distância máxima definida no momento da calibração, vai formando-se triângulos menores dentro do triângulo maior, obtido na calibração. Neste trabalho, optou-se por determinar a distância de um obstáculo, através da relação entre *pixel* e a distância percorrida. Porém outra forma de calcular a distância do obstáculo é utilizando estas informações de triângulo, que se obtém na calibração, e através da equivalência de triângulos, determinar a distância do obstáculo.

Com a realização dos testes do sistema, observou-se que o campo de visão a ser utilizado pelo sistema, deve seguir proporções adequadas, para que a detecção do obstáculo não seja precipitada, nem tardia. Deve-se procurar utilizar um campo de visão onde seja possível detectar o obstáculo ao ponto de poder executar alguma ação para evitar a colisão. Também não é necessário que o campo de visão seja muito abrangente, pois detectar a presença de um obstáculo precipitadamente, só geraria mais processamento para o computador. Também é necessário alertar para o fato de que os apontadores laser, capturados a uma distância muito grande, podem sofrer alteração no tamanho do ponto capturado pela câmera, dependendo muito da resolução da câmera utilizada.

Esse trabalho concentrou-se apenas no desenvolvimento do modelo e da implementação do mesmo, portanto foram desconsideradas as questões de fatores que podem influenciar na imagem capturada, como fumaça, luminosidade, chuva, etc. Portanto, os testes

foram realizados em ambientes homogêneos, com luminosidade padrão, sem influência do ambiente externo.

No Quadro 4 é apresentado o comparativo das características do sistema com os trabalhos correlatos. Pode-se observar que todas as aplicações partem do princípio de captura de imagem através da câmera de vídeo, porém este trabalho é o único que utiliza dispositivos lasers como objeto para criar marcos visuais e apenas este trabalho e o de Gavrila e Munder (2007) calculam a distância do veículo em relação ao obstáculo. Em relação ao trabalho de Fascioli e Broggi (1999), ambos utilizam-se de câmeras para capturar o campo de visão, porém este trabalho utiliza-se de uma câmera, enquanto o de Fascioli e Broggi (1999), utiliza-se de duas câmeras.

Característica	Este projeto	Estevam	Fascioli e Broggi	Gavrila e Munder
Captura através de câmera de vídeo	X	X	X	X
Utilização de laser	X			
Detecção de obstáculos	X		X	X
Cálculo da distância do obstáculo	X			X

Quadro 4 – Características da aplicação e trabalhos correlatos

4 CONCLUSÕES

Este trabalho propôs a implementação de um sistema de visão computacional utilizando uma câmera de vídeo e apontador laser, capturando a imagem, encontrando o marco visual criado pelo laser e a partir da posição do laser na imagem capturada, determinar se o veículo aproximou-se ou se afastou de um determinado obstáculo, e se houve movimentação, calcular a distância em que o veículo autônomo encontra-se do obstáculo.

Durante o desenvolvimento do trabalho, notou-se a necessidade da implantação de mais um apontador laser, para definir o campo de visão a ser considerado, pois a imagem capturada, era muito variável, sem um limite inferior fixo, e esse segundo laser viria a ser o marco visual responsável por demarcar o limite inferior do campo de visão a ser utilizado pelo sistema.

Os seguintes itens podem ser destacados como resultados alcançados na elaboração deste trabalho: o sistema implementado apresentou o resultado desejado, calculando a distância caso o veículo autônomo aproxime-se ou se afaste de um obstáculo; localização do marco visual criado pelo laser; definição do campo de visão utilizado pelo sistema. Com estes resultados alcançados, pode-se concluir que os objetivos deste trabalho foram alcançados.

O sistema foi desenvolvido utilizando a linguagem de programação Java. Com a utilização das bibliotecas *Java Advanced Imaging* (JAI) e *Java Media Framework* (JMF) , todas as necessidades que surgiram durante o desenvolvimento foram atendidas.

Como limitação o sistema possui a necessidade de calibração inicial, sendo que essa calibração deve verificar se o ângulo em que o laser inclinado for posicionado for muito grande, haverá uma maior distância dentro da área de aproximação de um obstáculo, porém, não será possível detectar quando o veículo está muito próximo do obstáculo, pois o laser inclinado sairá do campo de visão. De modo semelhante, se o ângulo em que o laser inclinado for posicionado for muito pequeno, pode-se chegar bem próximo do obstáculo, mas não será possível detectar o obstáculo a uma distância muito grande.

4.1 EXTENSÕES

Como sugestão para continuação deste trabalho e melhoria do sistema, pode-se citar:

a) utilização de duas câmeras e dois lasers, tendo assim a visão estereoscópica;
b) implementar filtros para eliminar fatores do ambiente externo, que possam prejudicar a interpretação da imagem capturada;
c) calcular a distância, utilizando equivalência de triângulos;
d) incorporar este trabalho a um veículo autônomo, e conforme aproximação de um obstáculo, tomar decisões de movimento a ser executado para evitar colisão;
e) identificar o marco visual criado pelo laser independente de cor.

REFERÊNCIAS BIBLIOGRÁFICAS

BAGNATO, Valderlei S. O magnífico laser. **Revista Ciência Hoje,** São Paulo, v. 37, n. 222, p. 30-37, dez. 2005.

BUSCARIOLLO, Paulo H. **Sistema de posicionamento dinâmico baseado em visão computacional e laser.** 2008. 145 f. Tese (Doutorado de Engenharia Naval) – Escola Politécnica da Universidade de São Paulo, São Paulo.

CASTRO, Cesar D. **Sistema de navegação para veículos robóticos aéreos baseado na observação e mapeamento do ambiente.** 2007. 107 f. Dissertação (Mestrado em Engenharia de Computação) – Faculdade de Engenharia Elétrica e da Computação de Campinas, Campinas.

CHANGE VISION, INC. **Astah community.** 2006. Disponível em: <https://members.change-vision.com/members/files/astah_community/6_2_1>. Acesso em: 17 ago. 2010.

CRUZ, Ricardo C. da. **Visão ativa aplicada à robótica:** um estudo de sistemas inteligentes utilizando percepção. 2004. 74 f. Trabalho de conclusão de curso (Pós-Graduação em Especialização em Mecatrônica) – Universidade São Judas Tadeu, São Paulo.

ECLIPSE. **Eclipse.org home.** [S.l], 2010. Disponível em: <http://www.eclipse.org>. Acesso em: 15 jul. 2010.

ESTEVAM, Leonardo A. **Protótipo de um veículo autônomo terrestre dotado de um sistema óptico para rastreamento de trajetória.** 2003. 71 f. Trabalho de conclusão de curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

FASCIOLI, Alessandra; BROGGI, Alberto. **Automatic vehicle guidance:** the experience of ARGO autonomous vehiche. Danvers: World Scientific Publishing Company,1999.

FAUSTO, Julio C.; COPETTI, Rodrigo. **Ferramenta de auxílio à comunicação e ao desenvolvimento colaborativo.** 2007. 30 f. Trabalho de conclusão de curso (Bacharelado em Sistemas de Informação) – Universidade Federal de Santa Catarina, Florianópolis.

FELICIANO, Flávio F.; SOUZA, Igor L. de; LETA, Fabiana. Visão computacional aplicada à metrologia dimensional automatizada: considerações sobre sua exatidão. **Revista ENGEVISTA,** Niterói, v.7, n. 2, p. 38-50, dez. 2005.

MARQUES FILHO, Ogê; VIEIRA NETO, Hugo. **Processamento digital de imagens.** Rio de Janeiro: Brasport, 1999.

FORTES, Luciane O. **Aplicação de técnicas de aprendizado para o controle inteligente de veículos autônomos.** 2001. 75 f. Trabalho de conclusão de curso (Bacharelado em Análise de Sistemas) – Centro de Ciências Exatas e Tecnológicas, Universidade do Vale do Rio dos Sinos, São Leopoldo.

GAVRILA, Dariu M.; MUNDER, Stefan. **Multi-cue pedestrian detection and tracking from a moving vehicle.** International Journal of Computer Vision. v.73, n. 01, p. 41-59. Jun. 2007.

GONZALES, Rafael C.; WOODS, Richard E. **Processamento de imagens digitais.** São Paulo, Edgard Blücher Ltda, 2000.

HEINEN, Farlei J.. **Robótica Autônoma:** integração entre planificação e comportamento reativo. São Leopoldo: Editora Unisinos, 2000.

HEINEN, Farlei J.; OSORIO, Fernando S.; FORTES, Luciane. **Controle inteligente de veículos autônomos:** automatização do processo de estacionamento de carros. In: X SEMINCO, 2001, Blumenau. Anais do X SEMINCO. Blumenau: Editora da FURB, 2001. v. 1, p. 35-46.

JANKOWSKI, Mariusz. **Java and the digital image processing application package.** Gorham, 2001. Disponível em: <http://www.usm.maine.edu/~mjankowski/docs/mdc2001/jai.html>. Acesso em: 24 maio 2010.

MENDONÇA, Paulo E. M. F. de. O laser na biologia. **Revista Brasileira de Ensino de Física,** Pirassununga v. 20, n. 1, p. 86-94, mar. 1998.

MIRANDA, Ricardo F. **Construção de uma linguagem baseada em Java denominada J4CV para visão computacional.** 2006. 121 f. Trabalho de conclusão de curso (Bacharelado em Ciências da Computação) – Universidade do Oeste Paulista, Presidente Prudente.

MORGAN, Jolvani. **Técnicas de segmentação de imagens na geração de programas para máquinas de comando numérico.** 2008. 99 f. Dissertação (Mestrado em Engenharia de Produção) – Universidade Federal de Santa Maria, Santa Maria.

MOREIRA, Filipe. **Introdução aos sistemas de instrumentação.** Bragança Paulista, 2003. Disponível em: <http://www.ipb.pt/~fmoreira/Ensino/EI/AcetatosTeoricosEI_03.pdf>. Acesso em: 20 mar. 2010.

OMG. **UML – Unified Modeling Language specification.** Version 2.0. Needham, 2005. Disponível em: <http://www.uml.org>. Acesso em: 15 agosto 2010.

PIAZZA, Gleyson L. **Implementação de uma fonte para acionamento de raio laser.** 2008. 214 f. Dissertação (Mestrado em Engenharia de Elétrica) – Universidade Federal de Santa Catarina, Florianópolis.

SANTOS, Rodrigo B. dos. **Metodologias para geração e atualização de mosaicos de fotos aéreas no Projeto ARARA.** 2004. 120 f. Dissertação (Mestrado em Ciências da Computação e Matemática Computacional) – Universidade de São Paulo, São Paulo.

SUN MICROSYSTEMS. **Java Advanced Imaging (JAI) API.** [S.l], [2010a]. Disponível em: <http://java.sun.com/javase/technologies/desktop/media/jai>. Acesso em: 10 ago. 2010.

_____. **Java Media Framework (JMF) API.** [S.l], [2010b]. Disponível em: <http://java.sun.com/javase/technologies/desktop/media/jmf>. Acesso em: 10 ago. 2010

APÊNDICE A – Montagem do protótipo utilizado para testes

Para montagem do protótipo utilizado nos testes, são necessários os materiais listados a seguir:

a) um câmera de vídeo com saída USB para que possa ser ligada ao computador que executará o programa. Esta câmera deverá ser fixada na frente do veículo;
b) uma régua de 30 cm de metal, para fixar um módulo laser numa altura ideal. Esta régua deve ser fixada logo atrás da câmera;
c) dois módulos laser são de 5mW, ligados em série com um diodo, para poder ser alimentado através das portas USB do computador. Um laser deve ser fixado no limite inferior do veículo. O outro deverá ser fixado a uma altura ideal, inclinado para o chão;
d) um transferidor, fixado na ponta da régua, no qual deve-se fixar o laser inclinado. É utilizado para poder-se visualizar o ângulo em que o laser está inclinado;
e) um veículo, autônomo ou não, utilizado para fixação dos itens acima listados.

APÊNDICE B – Layout do arquivo de configurações

Durante a calibração do sistema diversas configurações são armazenadas para utilização em outro momento do sistema. Essas configurações são armazenadas no arquivo `config.ini`, que pode ser encontrado, no diretório raiz do sistema. Nesse arquivo pode-se encontrar as seguintes configurações:

a) `distanciaMaximaInicial`: armazena a distância máxima inicial informada na tela "`Configuração`";
b) `distaciaCalibracao`: armazena a distância de calibração informada na tela "`Configuração`";
c) `diferencaCalibracao`: armazena a diferença entre a distância máxima inicial e a distância de calibração;
d) `alturaLaser`: armazena a altura do laser informada na tela "`Configuração`";
e) `angulo`: armazena o ângulo que é calculado após o usuário informar todas as configurações acima;
f) `variacao`: armazena o *pixel* de variação que é armazenado no momento da calibração;
g) `alturaVariacao`: é a altura calculada que representa a altura que o laser encontra-se em relação ao limite inferior do veículo;
h) `relacaoDeslocamento`: é a relação entre o *pixel* de variação com a diferença de calibração. Esta relação é utilizada quando é detectada a aproximação de um obstáculo, para calcular a distância em relação ao obstáculo detectado.