UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Localização de Robôs Móveis em Ambientes Fechados Utilizando Câmeras Montadas no Teto

Rafaella Cristianne Alves do Nascimento

Orientador: Prof. Dr. Luiz Marcos Garcia Gonçalves

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da UFRN (área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Mestre em Ciências.

Número de ordem PPgEEC: M415
Natal, RN, Janeiro de 2014

UFRN / Biblioteca Central Zila Mamede

Catalogação da publicação na fonte

Nascimento, Rafaella Cristianne Alves do.

Localização de robôs móveis em ambientes fechados utilizando câmeras montadas no teto. / Rafaella Cristianne Alves do Nascimento. Natal, RN, 2014.

53 f. : il.

Orientador: Prof. Dr. Luiz Marcos Garcia Gonçalves.

Dissertação (Mestrado) Universidade Federal do Rio Grande do Norte. Centro de Tecnologia. Programa de Pós-Graduação Engenharia Elétrica e da Computação.

1. Processamento de imagens Dissertação. 2. Localização de robôs móveis Dissertação. 3. Visão global em tempo real Dissertação. I. Gonçalves, Luiz Marcos Garcia. II. Universidade Federal do Rio Grande do Norte. III. Título.

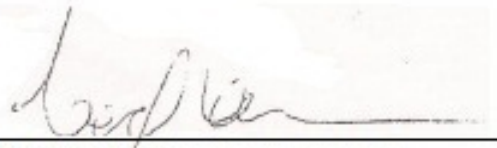
RN/UF/BCZM CDU 621.865.8

Localização de Robôs Móveis em Ambientes Fechados Utilizando Câmeras Montadas no Teto

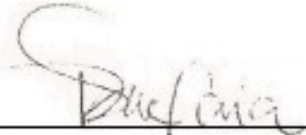
Rafaella Cristianne Alves do Nascimento

Dissertação de Mestrado aprovada em 23 de Janeiro de 2014 pela banca examinadora composta pelos seguintes membros:

Prof. Dr. Luiz Marcos Garcia Gonçalves (orientador) . . . . . DCA/UFRN



Prof. Dr. Luiz Eduardo Cunha Leite . . . . . UFRN



Prof. Dr. Rosiery da Silva Maia . . . . . UERN

Resumo

A Localização de robôs móveis em ambientes fechados encontra vários problemas que vão desde o erro acumulado pelos sensores instalados no robô até o fato das constantes mudanças que acontecem em alguns desses ambientes.

Uma técnica chamada visão global busca localizar robôs através de imagens obtidas por câmeras instaladas de maneira a cobrir o local onde o robô se locomove. Essa localização é feita através de marcas colocadas em cima do robô e algoritmos que são aplicados a tais imagens, de maneira a localizar essa marca na imagem. A partir do posicionamento da mesma obtem-se a posição e orientação do robô no ambiente.

Diante disso, a proposta aqui apresentada visa o desenvolvimento de uma sistema de localização de robôs móveis em ambientes fechados utilizando a técnica de visão global, com o objetivo de melhorar o processo de localização de plataformas robóticas móveis que venham a utilizar esse sistema.

Por ser um método de localização que só leva em consideração informações do momento atual, a localização de robôs através de imagens adequa-se bem ao objetivo do trabalho, que consiste em localizar várias plataformas robóticas que se movimentem num mesmo ambiente, podendo estar realizando tarefas similares ou não. Além disso ela proporciona também resultados mais acurados em tempo real.

Palavras-chave: Processamento de imagens, visão global, localização por câmeras.

Abstract

The localization of mobile robots in indoor environments finds lots of problems such as accumulated errors and the constant changes that occur at these places.

A technique called global vision intends to localize robots using images acquired by cameras placed in such a way that covers the place where the robots movement takes place. Localization is obtained by marks put on top of the robot. Algorithms applied to the images search for the mark on top of the robot and by finding the mark they are able to get the position and orientation of the robot.

Such techniques used to face some difficulties related with the hardware capacity, fact that limited their execution in real time. However, the technological advances of the last years changed that situation and enabling the development and execution of such algorithms in plain capacity.

The proposal specified here intends to develop a mobile robot localization system at indoor environments using a technique called global vision to track the robot and acquire the images, all in real time, intending to improve the robot localization process inside the environment.

Being a localization method that takes just actual information in its calculations, the robot localization using images fit into the needs of this kind of place. Besides, it enables more accurate results and in real time, what is exactly the museum application needs.

Keywords: image processing, global vision, real time processing.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A capacidade de realizar e empreender a tarefa de navegação, permite aos robôs móveis se movimentarem livremente no seu ambiente de trabalho, ora alcançando metas, ora desviando de obstáculos. Isto é obtido através de sistemas de navegação. De acordo com [Dev 1997], tal sistema de navegação pode ser dividido em duas tarefas básicas: a de se localizar e a de evitar obstáculos. Inúmeros são os trabalhos onde essas duas tarefas constituem o cerne da linha de pesquisa.

Uma área de pesquisa bastante difundida, e um tema recorrente dentro da área de robótica, é a linha de desenvolvimento de plataformas robóticas autônomas que ajudem as pessoas dentro das tarefas executadas por elas no dia-dia. Um exemplo disso, é a aplicação desse tipo de tecnologia dentro das nossas casas ou nosso ambiente de trabalho, seja como uma ajuda complementar nas tarefas cotidianas, seja como um meio de melhorar a segurança. Além dos ambientes internos, essas plataformas autônomas também são largamente empregadas em trabalhos que têm o ambiente externo como local de aplicação. Podemos dizer que se um robô se movimenta, alguém precisa navegar, e esse alguém deverá ser o próprio robô realizando todo o processamento do ambiente ou, podemos usar uma abordagem diferente, e realizar essa navegação através de um sistema externo que faça os cálculos e envie as informações de localização que a plataforma precisa.

Os robôs móveis encontram hoje um vasto campo nas aplicações industriais, como na limpeza do piso de prédios e fábricas, nos sistemas móveis de vigilância, no transporte de componentes em fábricas, na colheita e seleção de frutas, na medicina, nas forças armadas, dentro de museus, etc.

Quando se trabalha com uma plataforma robótica móvel uma questão extremamente importante é o fator localização. Localizar um robô móvel consiste em determinar a sua posição e orientação no espaço em um determinado instante de tempo. Sendo assim a capacidade de calcular aproximadamente a sua posição proporciona a um robô uma maior autonomia e os meios necessários para realizar múltiplas e importantes tarefas.

A extração de features de imagens adquiridas por meio de uma câmera em movimento vem sendo amplamente utilizada para estimar a posição e orientação da mesma, e conseqüentemente a posição de robôs, no caso da câmera estar montada na plataforma robótica em si. Esse procedimento é conhecido como odometria visual [Scaramuzza 2011], e utiliza uma seqüência de imagens para o processamento das informações, sendo muito útil na navegação de veículos e robôs em ambientes onde outras técnicas não podem ser utilizadas, ou mesmo onde o erro é alto e deseja-se dados mais precisos .

Tal técnica é bastante promissora pois não introduz erros relacionados com deslizamento de rodas, tamanho de eixos, entre outros. Vários estudos vem sendo feitos nessa área, entre os quais podemos citar principalmente aqueles que utilizam seqüências de imagens únicas ou os que fazem uso de seqüências de pares estéreo. Apesar de ser promissora e de não introduzir erros como os presentes nas técnicas que fazem uso principalmente de encoders, a odometria visual ainda sofre com problemas relacionados a falta de acurácia no processo de feature matching e ao acúmulo de erros em longas distâncias, entre outros.

A odometria visual, incorporada ao processo de navegação vem sendo uma técnica bastante utilizada no meio da robótica, no qual uma variedade de adaptações de navegação visual com a utilização de câmeras montadas nos robôs, vem sendo propostas. Tal técnica não é portanto uma proposta nova, pois as primeiras idéias surgiram décadas atrás. O fato do pouco uso quando comparado as outras técnicas de odometria mais utilizadas, deve-se bastante ao fato dos custos computacionais envolvidos nesse tipo de aplicação. Por isso, apesar dos vários aspectos promissores, a utilização de técnicas de visão computacional como principal fonte de dados e informações para um processo de navegação robótica, caminha mais devagar quando comparado a outras áreas relacionadas, o que não implica em pouco desenvolvimento, já que o constante avanço e surgimento de novas tecnologias e hardwares mais rápidos, possibilita a cada dia uma capacidade de processamento bem maior do que a que se possuía anos atrás.

Outra abordagem é a chamada visão global apresentada em [Kay 1993], onde uma espécie de odometria visual é descrita. O termo visão global refere-se ao fato da utilização de câmeras colocadas no ambiente onde o mesmo irá se locomover, em locais fixos, em vez de colocadas nos robôs, com o objetivo de monitorar a posição e orientação do mesmo. Nesse método são utilizadas marcas colocadas em cima da plataforma móvel que se deseja monitorar e algoritmos de processamento de imagens são utilizados no processo de localização através de imagens.

Dentro da contextualização feita anteriormente o trabalho apresentado aqui pode ser classificado como uma espécie de odometria visual, com a diferença de que a proposta aqui abordada utiliza câmeras montadas no teto do ambiente para localizar uma plata-

forma robótica móvel se movimentando no chão. Portanto a proposta consiste num sistema de câmeras responsáveis por monitorar uma determinada área e por conseguinte monitorar a posição e orientação de um robô que se movimenta num determinado plano dentro desse ambiente.

O objetivo principal consiste então, em melhorar a acurácia do posicionamento global do robô no ambiente afim de diminuir o máximo possível o erro introduzido por meio dos sensores locais presentes na plataforma robótica, bem como aliviar a carga de processamento que o robô tem de realizar enquanto está se locomovendo, tornando não só o hardware necessário para o mesmo mais simples, mas acabando por deixar mais genérico o sistema de forma que qualquer tipo de plataforma possa utilizar esse recurso de localização. A base do algoritmo de localização está em procurar por marcas propositalmente colocadas em cima da plataforma móvel. A marca utilizada é composta de duas partes, cada uma com uma cor diferente.

Devido ao fato do sistema de câmeras proposto ser um sistema de localização absoluta ele trabalha calculando a posição e orientação do robô a partir apenas de informações atuais, não ocorrendo portanto erros acumulativos.

O ambiente de trabalho é restrito a área coberta pelas câmeras, logo, para que se tenha uma cobertura total do recinto, um número apropriado de câmeras deve ser igualmente distribuído pelo ambiente de maneira que o sistema possa enxergar tudo.

Ao contrário da maioria das aplicações desenvolvidas dentro do campo de odometria visual, as câmeras na aplicação apresentada aqui serão estáticas e afixadas no teto do ambiente a ser monitorado. Tal técnica é principalmente utilizada no escopo do futebol de robôs. O sistema com várias câmeras deve ser capaz de diferenciar cada parte do ambiente coberto por cada câmera que integre o sistema, com isso ao transformarmos o ponto localizado na imagem para o ponto real no mundo, tenhamos sempre dados corretos, mesmo que o objeto tenha transitado entre áreas cobertas por câmeras diferentes, e consequentemente imagens com origem deferentes tenham sido processadas.

A principal aplicação da proposta aqui abordada será justamente a melhoria da acurácia do posicionamento de uma plataforma móvel em um ambiente real, proporcionando a mesma um método de posicionamento absoluto como alternativa para sua localização, quando a mesma estiver se locomovendo no ambiente monitorado pelo sistema.

Capítulo 2

Embasamento Teórico

A teoria envolvida em aplicações de visão computacional e processamento de imagens apresentam uma certa complexidade para serem compreendidas num primeiro momento. Muitas vezes torna-se necessário que o leitor se debruce um pouco sobre os aspectos teóricos que envolvem um determinado trabalho. Diante disso, e como forma de familiarizar o leitor com a teoria usada como base neste trabalho, apresentaremos neste capítulos algumas definições que serão referenciadas ao longo do desenvolvimento da proposta.

A visão computacional é uma ciência bastante multifacetada e difícil de ser exatamente definida. A grosso modo podemos dizer que a mesma reúne um conjunto de técnicas, linhas de pesquisa, etc, responsáveis por fazer uma máquina "enxergar"através de imagens o mundo a sua volta, extraíndo das mesmas informações significativas, permitindo a máquina executar tarefas inteligentes, simulando e até mesmo aproximando-se da inteligência humana [Ballard 1982], [Milano 2010].

A extração de características significativas e que sejam de interesse para uma determinada aplicação envolve várias tarefas, que vão desde a escolha da câmera certa para a aquisição de imagens até os cálculos envolvidos na transformação de um ponto na imagem em um ponto no mundo real. Todas essas tarefas estão agrupadas dentro da visão computacional no que chamamos de Fases de Calibração.

2.1 Fases de Calibração

Para melhor entendimento as fases de calibração podem ainda ser agrupadas em calibração com objetivo de extrair e identificar características específicas na imagem que nos permitam identificar o que é desejado no futuro, e em calibração que permita a ligação de coordenadas localizadas e identificadas na imagem em coordenadas do mundo real.

2.1.1 Calibração Radiométrica

O objetivo principal é estimar a relação entre a radiância (intensidade de luz por unidade de área idealmente emitido por cada ponto P de uma determinada superfície 3D no espaço) e irradiância da cena (intensidade de luz por unidade de área em cada ponto p do plano imagem), afim de que se possa realizar uma correção na imagem com o objetivo de que outras regiões da mesma fiquem com características semelhantes as do centro, ponto onde tais parâmetros são considerados ideais.

A equação 2.1 explicita essa relação, e a figura 2.1 ilustra a relação entre radiância e irradiância.

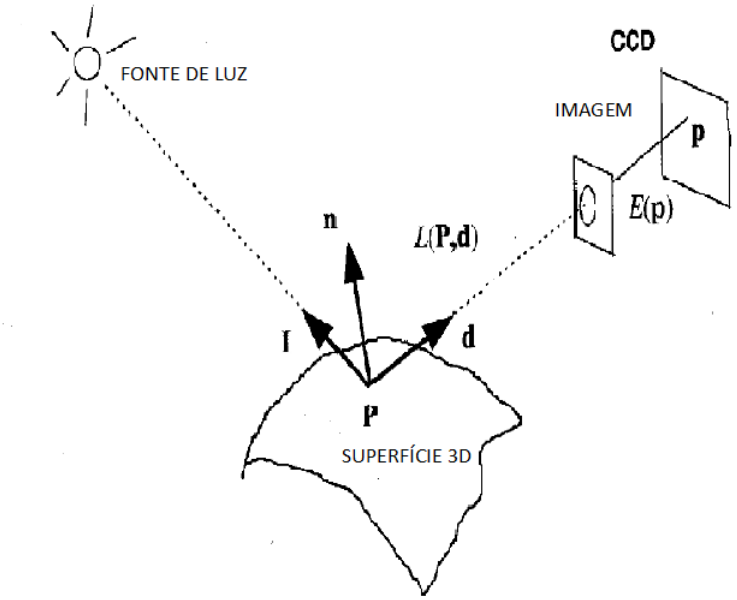


Figura 2.1: Ilustração de Radiância e Irradiância

$$E(p) = L(P)\frac{\pi}{4}(\frac{d}{f})^2 cos^4\alpha \tag{2.1}$$

- P - Ponto no mundo
- p - Ponto na imagem
- E - Quantidade de luz que chega na imagem
- L - Quantidade de Luz refletida por um ponto da cena
- d - Tamanho da lente

- f - Distância focal
- α - Ângulo entre o raio e o eixo óptico

A equação diz que a iluminação da imagem num ponto p decresce com a quarta pontência do cosseno do ângulo formado pelo raio principal através de p com o eixo principal [Trucco 1998]. Isso pode ser melhor observado na figura 2.2

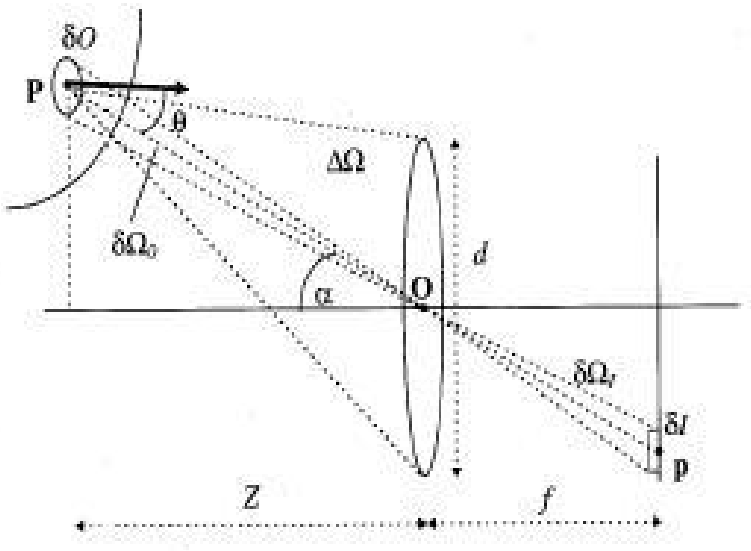


Figura 2.2: Radiometria do processo de formação de imagem

Em resumo os principais fatores radiométricos são levados em consideração e a relação entre a quantidade de luz emitida das fontes de luz, refletida pelo ambiente e que chega aos sensores é analisada, com dois objetivos principais que são modelar quanta luz é refletida pelos objetos do ambiente e quanta luz refletida realmente chega ao plano imagem da câmera.

2.1.2 Calibração de Cores

Na etapa de calibração de cores parâmetros como nível de cor de cada uma das cores utilizadas para a confecção das marcas são verificados e selecionados. O procedimento de calibração de cores visa analisar como os níveis de RGB das cores disponíveis se comportam no ambiente coberto pela câmera, afim de que seja posssível ao usuário analisar qual das cores melhor se adequa ao ambiente, com o objetivo de se chegar a um nível médio RGB que represente a cor, pois essa informação é de crucial importância para a criação de um filtro de cor utilizado para a localização das marcas no ambiente.

2.1.3 Calibração Geométrica

A calibração geométrica é responsável por identificar os parâmetros que permitirão a transformação de um ponto de interesse detectado na imagem em um ponto no mundo real.

Parâmetros Intrínsecos

Os parâmetros aqui abordados referem-se aos parâmetros internos da câmera, ou parâmetros intrínsecos, os quais refletem a relação entre as coordenadas de pixel e as coordenadas no frame de camera. São os responsáveis pela ligação entre o referencial de câmera e as coordenadas de imagem. Os parâmetros aqui envolvidos são:

$$\begin{bmatrix} \frac{-f}{s_x} & 0 & O_x \\ 0 & \frac{-f}{s_y} & O_y \\ 0 & 0 & 1 \end{bmatrix}$$

- f - Distância focal
- Sx e Sy - Tamanho real de um pixel em unidades métricas (metros, centímetros, etc) tanto na direção do eixo X como do eixo Y.
- O - Centro da imagem em relação ao posicionamento da câmera ou ponto principal da imagem (coordenadas x e y)
- Coeficientes de distorção

Esses parâmetros uma vez estimados não precisam mais ser recalculados, desde que a câmera não tenha sua posição modificada.

Parâmetros Extrínsecos

Os parâmetros aqui abordados referem-se aos parâmetros externos da câmera, ou parâmetros extrínsecos, os quais refletem a relação entre as coordenadas de camera, ou frame de camera, e as coordenadas no frame de mundo conhecido. Colocado de outra maneira é o conjunto de parâmetros que relaciona a posição e orientação de um ponto no frame de câmera com o ponto correspondente no mundo.

Nessa etapa, uma vez que os parâmetros internos tenham sido estimados, podemos calcular os parâmetros extrínsecos, já que o cálculo dos mesmos tem como entrada os parâmetros intrínsecos.

Consistem de uma matriz de rotação e de um vetor de translação. São esse vetor e essa matriz, juntamente com a matriz de parâmetros intrínsecos, aplicados a um ponto de mundo, que nos dão o ponto imagem.

A Matriz de parâmetros extrínsecos, junção matriz de rotação e translação, pode ser vista abaixo:

$$\begin{bmatrix} R11 & R12 & R13 & T_1 \\ R21 & R22 & R23 & T_2 \\ R31 & R31 & R33 & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.2 Tipos de Posicionamento

Existem basicamente duas formas mais utilizadas de se estimar a posição aproximada de um robô. Uma delas é através de posicionamento absoluto e a outra através de posicionamento relativo.

2.2.1 Posicionamento Relativo

Posicionamento relativo consiste num método que trabalha com um conjunto de dados utilizando sempre amostras anteriores a fim de encontrar a diferença entre uma nova leitura e a leitura anterior, conseguindo assim a posição atual do robô. Trata-se de um método de posicionamento que nos dá a posição local e utiliza dados coletados por sensores que estão diretamente conectados a plataforma móvel, tais como medidores de velocidade, aceleração, sensores de rotação, giroscópios e etc.

2.2.2 Posicionamento Absoluto

Posicionamento absoluto consiste num método de posicionamento global e que nos fornece justamente uma posição global da plataforma, ou seja, a localização da plataforma em relação a um referencial fixo absoluto, localizando-a assim no ambiente como um todo, utilizando apenas as informações atuais dos seus sensores. Trabalha com um conjunto de dados que não variam ao longo do tempo, não sendo necessário, portanto, estimar a diferença entre uma amostra e outra. Os dados utilizados nessa técnica provem principalmente de marcas passivas ou ativas, balizas, mapas do ambiente, sinais de satélite, como o sistema GPS, dentre outros.

2.2.3 Odometria

Uma ferramenta bastante utilizada no posicionamento de robôs é a odometria, que consiste num método de localização relativa que utiliza dados anteriormente obtidos para estimar a localização atual, bem como a orientação do robô em relação a sua posição anterior. Esses valores são medidos por dispositivos, tais como sensores de movimento, e especialmente sensores de rotação.

Tal método traz consigo uma gama de problemas entre os quais se encontram o fato de a precisão do método depender bastante da precisão dos sensores utilizados, de medidas exatas das rodas e eixos, a superfície onde o robô se desloca deve ser suave, sem grandes níveis de desnivelamento, o fato de que as rodas escorregam deve ser incorporado aos cálculos e principalmente que o erro introduzido por todos os fatores citados anteriormente é cumulativo, o que implica no fato de que esse tipo de erro só tende a crescer ao longo do tempo. Logo, um erro em determinado instante também compromete as medições dos instantes seguintes. Por isso, apesar de ser um método bastante simples para determinar a localização de um robô móvel, a odometria está sujeita a erros que fazem com que ela forneça apenas uma estimativa da localização exata do robô em um dado instante.

2.2.4 GPS

O GPS (Global Positioning System) é uma técnica comumente utilizada para diminuir esse tipo de incertezas inerente ao processo de posicionamento relativo, e consequentemente a odometria. Contudo trabalhar dentro de ambientes fechados está fora do escopo de ação do GPS, cujo grau de acurácia atingido, dentro desse tipo de ambiente, não é suficiente para satisfazer os requisitos normalmente exigidos por aplicações robóticas funcionando dentro de ambientes fechados.

2.2.5 Posicionamento Através de Marcas

Posicionamento utilizando marcas baseia-se no fato de que as mesmas possuem características distinguíveis pelos sensores dos quais o sistema responsável por detectá-las no ambiente se utiliza. Podem ser classificadas em marcas naturais e artificiais. As naturais são aquelas que já se encontram no ambiente, como por exemplo objetos ou mesmo características específicas do ambiente. As artificiais são marcas especialmente criadas e colocadas no ambiente de forma a ajudarem o robô durante o processo de navegação.

2.3 Homografia

Homografia 2D é uma transformação projetiva planar que mapeia pontos de um plano para outro plano [Hartley 2004]. Este mapeamento linear de pontos pode ser escrito em coordenadas homogêneas onde uma matriz H é a matriz de homografia que define o mapeamento de um conjunto de pontos correspondentes entre dois planos [dos Santos 2012].

2.4 Filtro de Kalman

O filtro de Kalman é um algoritmo recursivo simples e eficiente, capaz de estimar as variáveis de estado de sistemas lineares a partir de medidas ruidosas dos sensores que compõem o sistema. Por esta razão o filtro de Kalman é bastante utilizado em sistemas de navegação e rastreamento de veículos. É considerado por muitos o grande avanço da teoria da estimação do século XX [Aquino 2010].

Capítulo 3

Estado da Arte

A robótica se apresenta como uma área que proporciona uma ampla integração de metodologias ou técnicas diversas como percepção (sensores, fusão sensorial, visão computacional, etc.), meta-heurísticas (Algoritmos Genéticos, Computação Evolutiva, Inteligência Social, Redes Neurais, Lógica Difusa, etc.), navegação, planejamento inteligente, Teoria de Controle, entre outras. Robôs móveis são exemplos atuais de plataformas viáveis que permitem o desenvolvimento de aplicativos e experimentos, além do aprimoramento de conhecimentos e a verificação da aplicabilidade destas integrações, proporcionando flexibilidade e baixo custo.

O sistema aqui proposto baseia-se em vários conceitos inseridos dentro do escopo da robótica, entre os quais pode-se principalmente citar:

- Localização de robôs utilizando visão global
- Navegação sem o uso de mapas do ambiente.
- Design de Landmarks de acordo com as condições do ambiente
- Técnicas de visão computacional aplicadas na localização de robôs dentro de ambientes fechados
- Localização de robôs através de imagens
- Plataformas robóticas guiadas por métodos de visão computacional

O avanço do estudo das técnicas de visão computacional trouxeram grandes progressos tanto na navegação em ambiente fechados como em ambientes externos. No início era impossível a navegação de robôs por entre ambientes apinhados de obstáculos, bem como realizar o processamento de imagens extraindo as features desejadas de forma rápida a ponto da informação ser utilizada em aplicações em tempo real, como segurança por exemplo, tarefas comuns hoje em dia. A navegação em ambientes fechados segue basicamente três linhas de aplicações [DeSouza 2002]:

- Navegação Utilizando Mapas Previamente Construídos : Nesse tipo de navegação o robô trabalha com uma representação do ambiente, a qual pode ser mais ou menos detalhada, podendo variar desde um modelo completo em CAD a uma simples representação das interconexões que existem no ambiente. As aplicações iniciais faziam uso de uma representação em 2D dos objetos o que gerava um "mapa de ocupação em 2D"do local [Borenstein 1989]. Essa técnica evoluiu anos mais tarde para os "Campos de Força Virtuais", que funcionavam "repelindo"o robô de se aproximar do obstáculo [Moravec 1985].
- Navegação com a Construção dos Mapas Necessários Simultaneamente(SLAM): Nesse tipo de abordagem um mapa do ambiente vai sendo construído ao passo que a navegação vai sendo realizada, sem que o robô tenha conhecimento prévio.
- Navegação Sem o Uso de Mapas : Nessa categoria estão inclusas todas as abordagens que não utilizam nenhuma descrição prévia do ambiente e nem constroem mapas em nenhuma etapa da navegação. Não é necessário que a posição absoluta dos elementos seja armazenada ou mesmo conhecida previamente. Entre as técnicas mais utilizadas está a navegação baseada em "Fluxo Óptico"e aparências. Em outras palavras, essa abordagem faz uso de dados retirados de imagens que fazem com que o objeto vá se iterando do ambiente a medida que a navegação vai se desenvolvendo, sem no entanto armazenar essas informações [DeSouza 2002].

De acordo com as vertentes de desenvolvimento para aplicações de navegação robótica em ambientes fechados apresentadas, podemos classificar o sistema proposto nesse trabalho, como um sistema que realiza a tarefa de navegação sem o uso de mapas.

De acordo com [Borenstein 1997] o processamento envolvido em aplicações de visão computacional seguem geralmente os seguintes passos : Aquisição de dados, que significa obtenção das imagens, detecção das marcas ou features, a fase de matching, que consiste em determinar se o que foi encontrado é o que é desejado e por último o calculo de posição.

Um ponto que também merece atenção dentro das técnicas de visão computacional é a escolha do tipo de landmark mais adequado para cada tipo de ambiente. Não se pode ter um sistema de localização através de landmarks robusto se a concepção da mesma não for adequada. Como exemplificado em [Yoon 2002] , a marca deve ser desenvolvida levando em consideração que a mesma deva ser robusta quando a distorções geométricas e variações de iluminação. Nesse trabalho ela é desenvolvida de maneira simétrica e repetitiva e duas cores com médias bem distantes são cuidadosamente selecionadas para compô-la. É levado em consideração ainda o histograma de cada landmark no processo

de localização. Em outro trabalho por exemplo [Becker 1995], o autor testa o uso de landmarks simples de duas maneiras : colocadas no teto e colocadas na parede, afim de elucidar o que se adequa melhor ao ambiente de trabalho. Ele apresenta baseado nos testes realizados, as vantagens e desvantagens de se usar as marcas em cada uma das situações citadas.

3.1 Localização de Robôs Através de Imagens

É apresentado em [Wang 2005] um método para estimar o movimento de uma câmera dentro de uma cena estática, método também conhecido como ego-motion, através de uma câmera montada numa plataforma robótica. O objetivo consiste na verdade em rastrear o movimento do robô, baseado no da câmera, utilizando as seqüências de imagens adquiridas pela câmera utilizada. Considera-se que o deslocamento é feito sobre um chão plano, de maneira que o plano imagem e o plano de movimento podem ser considerados paralelos, fato que propicia a aplicação do conceito de homografia no momento de relacionar ambos os planos. Outro recurso utilizado é o filtro de kalman, o qual é aplicado na predição de movimento da câmera, ajudando consequentemente a diminuir a região de interesse dentro da imagem.

Em [Fiala 2004] descreve-se um sistema baseado em visão para controlar múltiplas plataformas robóticas em tempo real usando imagens de uma câmera posicionada de maneira a cobrir todo o ambiente considerado. O processo consiste em identificar os robôs na imagem através de marcas e calcular a sua posição no mundo, a qual é usada no sistema que controla o movimento do robô, fazendo com que caminhos a serem percorridos possam ser especificados e seguidos com uma taxa de erro bem menor, já que um método de posicionamento absoluto é utilizado. Aplicações desse gênero são muito comuns em competições como a Robocup [RoboCup n.d.]. Essa proposta não segue o padrão de marcas utilizando cores, mas sim faz uso de um conjunto de padrões em branco e preto que carregam informações como se fossem códigos de barras, de forma que cada padrão tem um significado específico. Uma aplicação desse tipo de marca e do sistema proposto se dá na indústria, de acordo com o autor.

Outros trabalhos onde as técnicas de posicionamento utilizando imagens e visão global estão presentes são [Se 2005] e [Orqueda 2007], com a diferênça principal de que no primeiro o problema de posicionamento global é tratado como um problema de matching, onde as características do frame atual são comparadas com a do frame anterior, que está armazenado, e a partir destes dados é feita a localização, enquanto no segundo marcas artificiais são utilizadas em cada robô, como subsídio para a concepção de um

algoritmo robusto utilizando visão global, com o objetivo de resolver o problema de coordenar o movimento de vários robôs se locomovendo num mesmo ambiente, sem que seja necessário a comunicação entre eles para a atualização de posição.

3.2 Localização de Robôs Utilizando Visão Global e Navegação sem o Uso de Mapas do Ambiente

A utilização de técnicas de odometria visual, mais precisamente de visão global, tem se mostrado uma fonte de boas aplicações e bons resultados, dentro da aplicação de posicionamento absoluto em detrimento do posicionamento relativo, quando se trata do controle eficaz de vários objetos se deslocando dentro de um mesmo ambiente.

Em [Nadarajah 2013] é feito um estudo sobre o sistema de visão utilizado no escopo do futebol de robôs. O autor primeiramente discorre sobre o local de posicionamento das câmeras, se posicionadas no teto ou de lado, e sobre a utilzação de visão global ou de visão local, nesse último caso com as câmeras montadas no robô, podendo ser divididas também em três tipos de posicionamento: omni-direcional, visão estéreo e visão monocular. É discutido também alguns tipos de algoritmos de processamento de imagem mais adequados a esse tipo de aplicação. O autor começa sua explanação sobre o tipo de câmera a ser utilizada. Um fator que deve ser levado em consideração é a velocidade de aquisição dos frames por parte da câmera escolhida. As mais sofisticadas costumam retornar cerca de 60 frames por segundo (fps) enquanto outras mais simples fornecem 30 fps. Um cuidado que se deve ter é manter a taxa de frame por segundo sempre superior a 24fps, pois menos que isso já é perceptível à visão humana os atrasos entre cada frame. Ao abordar a visão global com a câmera montada no teto e centralizada, o autor elenca as vantagens de se utilizar esse tipo de dispositivo, que são a maximização do campo de visão e diminuição das distorções nas extremidades, bem como da oclusão, fato que ocorre quando o objeto de interesse fica obstruído total ou parcialmente por outro objeto do ambiente. Segundo o autor escolher o tipo de visão a ser utilizada, se global ou local e o tipo de câmera, são somente a "ponta do iceberg", com a escolha de como será feito o processamento de imagem o ponto principal. Ele cita bibliotecas de processamento com especial atenção para a Open Computer Vision Library (OpenCV), e para os métodos de processamento Filtro de Kalman, CAMShift e Fluxo óptico.

Em [Reina 2012] é apresentado o Zeppelin, um sistema distribuído que realiza o planejamento de caminhos em ambientes dinâmicos com o objetivo de mover objetos através dele com sucesso. As câmeras são montadas no teto, fixas e os dispositivos se

comunicam por meio de wireless. Cada câmera é responsável por cobrir seu campo de visão e juntos cooperam para chegar ao caminho completo. Cada câmera é responsável por mover o robô dentro do seu campo de visão. O Zeppelin foca em lidar com o problema da superposição entre a área coberta por duas câmeras adjacentes e em como o sistema é capaz de processar os erros que aparecem em decorrência disso. Isso merece atenção pois como consiste numa aplicação para planejamento de caminhos, erros decorrentes da superposição podem gerar caminhos errados prejudicando ou tornando impossível o deslocamento do objeto. Também se preocupa com a escalabilidade de maneira que o mesmo seja adaptável a qualquer tamanho de ambiente. Toda e qualquer informação é propagada entre todos os componentes do sistema, já que o mesmo não possui um controle centralizado.

Em [Yan 2010] podemos ver outra aplicação que utiliza visão global com câmeras montadas no teto. Uma marca composta por duas cores diferentes é utilizada. O autor persegue o objetivo de processamento em tempo real e por isso, dois tipos de busca são utilizadas: a primeira realiza uma busca em toda a imagem no início, e é chamada de procura global, sendo que após a primeira posição da marca ser encontrada, passa a ser realizada apenas uma procura local, ao redor da posição previamente determinada para a localização do robô. Se a busca local falhar, então a procura global será executada, e uma busca em toda a imagem será feita.

Abordagens que seguem a mesma ideia básica são vistas em [Michel 2005] e [Roque 2005]. No primeiro, um método de navegação para um robô humanoide que realiza toda a sua computação de dados e planejamento de caminhos em tempo real é apresentado. O sistema usa uma abordagem de localização global durante o deslocamento, em virtude da necessidade de reconhecimento imediata de qualquer alteração no ambiente que venha a afetar o deslocamento do robô bípede. O uso do posicionamento global permite a separação das tarefas de monitorar o ambiente e a tarefa de deslocamento propriamente dita, o que contribui ainda mais para a otimização do processo em tempo real. Permite também que a navegação em ambientes mais cheios de obstáculos seja também mais confiável, evitando que o robô fique preso em loops. A câmera é montada no teto de forma a monitorar o movimento e marcas são utilizadas para identificar tanto o robô quanto os obstáculos. O sistema funciona basicamente com o robô sendo o cliente, e os módulos de planejamento de caminhos e calculo de posição, os servidores. No segundo, outra aplicação que realiza o planejamento de caminhos é mostrada. O trabalho apresenta um sistema de planejamento de caminhos cuja arquitetura é dividida em três grandes partes principais : um módulo de visão global, um módulo de planejamento de trajetória e o módulo de controle de navegação. O primeiro módulo captura imagens do ambiente, detectando

possíveis obstáculos e a posição do robô. O segundo coleta essas informações e as usa como base para construir um diagrama de voronoi do ambiente e que servirá de base para o modulo de controle e navegação. O terceiro trata do controle das funções do robô propriamente ditas. O módulo de visão global é composto por uma câmera CCD simples posicionada a uma altura fixa e cobrindo uma determinada área do plano que vem a ser o ambiente de deslocamento do robô. Marcas são utilizadas para a detecção dos objetos.

Uma aplicação interessante e promissora é vista em [Baltes 2007] É baseada em visão global e o autor faz uma distinção entre o procedimento de visão local, quando a câmera está montada no robô e o sistema só percebe o a si mesmo, quase como uma espécie de odometria, e a visão global, que para ele ocorre quando a câmera montada num plano acima, leva o posicionamento absoluto em consideração e vários objetos são monitorados. O trabalho mostra a evolução de um sistema baseado em visão global para monitoramento de um time de robôs indo desde a implementação inicial, onde são utilizadas landmarks simples, passando pelas fases de melhoria de calibração, adaptação das marcas ao ambiente, do aprimoramento dos padrões de marcas utilizadas, quando as mesmas passam a ser códigos em preto e branco, a utilização de redes neurais, chegando a um estágio de desenvolvimento onde o objetivo é criar um sistema que possa gerenciar o posicionamento do time de robôs sem a necessidade de padrões de localização pré-definidos, criando-se assim um sistema de localização global genérico, técnica que segundo o autor, ainda está em estudo.

Pontos que reforçam a importância do uso de posicionamento global em um sistema de localização de múltiplos objetos é mostrado em [Chen 2009], onde um grupo de robôs deve se movimentar em conjunto, mas acabam se perdendo quando não é usado um procedimento de localização global. As câmeras são montadas nos robôs, fixas e voltadas para o teto.

Uma aplicação interessante é vista em [Pinto 2012], onde a marca utilizada é uma "constelação"de LEDs colocadas no teto. O autor tem como objetivo criar um aplicação robusta quanto a variação de iluminação que venha a atingir o ambiente. Os LEDs colocados no teto "piscam"somente em determinados instantes quando recebem, através de comunicação infravermelho, um sinal emitido pelo robô, significando que o mesmo está na zona coberta pelo conjunto de LEDs. A partir daí a comunicação entre robô e marca passa a ser sincronizada, o que segundo o autor faz com que a detecção da marca seja mais robusta e imune as variações de iluminação.

Podemos destacar ainda as implementações vista em [Adorni 2001] e [Chen 2012]. A primeira apresenta um reforço da ideia de como é importante a robustez de um sistema que funcione em tempo real e que deva monitorar satisfatoriamente o movimento

de vários objetos ao mesmo tempo, principalmente em ambientes suscetíveis a constantes mudanças. O local onde a abordagem é desenvolvida é o da RoboCUP o que não impede que a mesma abordagem seja empregada em outros ambientes. Tanto câmeras fixas quanto omnidirecionais foram utilizadas nos trabalho. Já a segunda mostra uma tendência que é o uso do Wiimote em aplicações de robótica. Nesse trabalho o wiimote é colocado no teto de maneira que o deslocamento de um grupo de robôs seja coberto.

De uma maneira geral, posicionamento utilizando marcas presentes no ambiente se apóiam no fato de que as mesmas devem ser plenamente detectáveis pelos sensores, bem como pelos algoritmos de processamento que o sistema proposto utiliza. Como já citado anteriormente as marcas podem ser naturais ou artificiais. Todos os trabalhos elencados até então se utilizam de marcas artificiais.

Diante das linhas de pesquisa citadas anteriormente, a localização de robôs através de imagens, a localização de robôs utilizando visão global e a navegação sem o uso de mapas do ambiente, o sistema aqui proposto pode ser definido como um sistema que busca melhorar a acurácia do posicionamento de plataformas robóticas movimentando-se dentro de um mesmo ambiente, com a opção de localização de um ou mais objetos, utilizando técnicas de posicionamento absoluto, baseado em imagens obtidas por um sistema de câmeras posicionadas no teto, paralelamente ao plano de movimento do robô. A marca utilizada é composta por um padrão de duas cores, escolhidas de maneira a se destacarem no ambiente.

Há ainda a aplicação de técnicas de processamento de imagem empregadas na fase de pré-processamento, utilizadas para melhorar a qualidade das imagens adquiridas, buscando assim melhorar e otimizar o processo de procura das marcas, sem deixar de lado a preocupação inicial de escolher marcas bem adequadas ao ambiente de trabalho.

Esse sistema dispensa a necessidade da construção e armazenamento de mapas do ambiente por parte do robô, transferindo assim o máximo possível de processamento para uma aplicação rodando em paralelo com o sistema que controla o guia robótico. Vale salientar que o sistema de localização é totalmente independente do sistema ou método que esteja controlando o robô, deixando assim o sistema aqui proposto, como um fornecedor de informações e os robôs, seus clientes. O sistema considera que o robô se locomove no plano 2D indicado pela distância focal utilizada, que no caso é a altura do teto.

A proposta aqui especificada ajuda a diminuir a necessidade de sistemas complexos de odometria por posicionamento relativo, normalmente encontrados na maioria das aplicações utilizando robôs móveis, bem como contribui para a diminuição de erros de localização introduzidos pelos erros de odometria baseada em sensores de rotação, por exemplo.

Capítulo 4

Definição da Proposta

O fator localização é de suma importância em aplicações que utilizam plataformas robóticas móveis como principais recursos. Tal localização consiste no processo de calcular posicionamento e orientação do robô móvel num determinado instante, de maneira que esse recurso quando presente nessas plataformas, proporciona maior autonomia e capacidade de executar tarefas distintas e importantes.

Existem basicamente duas formas mais utilizadas de se estimar a posição aproximada de tais plataformas. Uma delas é através de posicionamento absoluto e a outra através de posicionamento relativo, conceitos discutidos aqui anteriormente.

Como já salientado anteriormente, esse tipo de técnica encontra grande uso e aplicação dentro de sistemas responsáveis por realizar o monitoramento e localização de uma plataforma robótica móvel, devido principalmente ao fato de marcas serem ferramentas confiáveis no cálculo da posição de robôs.

O procedimento geral de um sistema de localização que utiliza marcas, como citado em [Borenstein 1997], [Milano 2010], consiste em geral das seguintes etapas:

- Aquisição de dados : Temos aqui a parte do sistema responsável por capturar as imagens a serem analizadas.
- Tratamento dos dados: Aqui concentra-se a parte de processamento de imagens onde são realizados, por exemplo, procedimentos como correção de distorções.
- Detectar Marcas: Todo o processo de procura e recursos utilizados na mesma.
- Cálculo da posição: Depois de encontrada a marca, a posição do robô é calculada com base nesses dados.

Marcas artificiais são cuidosamente escolhidas para se destacarem no ambiente de trabalho, facilitando assim a sua identificação e o processo de navegação. Além disso a forma e a geometria de tais marcas são de antemão conhecidas. Muitos são os sistemas

de posicionamento utilizando marcas artificiais que se baseiam em técnicas de visão computacional, o que não deixa de ser diferente com a proposta aqui apresentada. A maioria desse tipo de sistema utiliza câmeras montadas no robô, o que implica que em alguns casos, a acurácia de tais sistemas dependerá muito de como as imagens do ambiente são adquiridas, ou seja, de como a geometria da marca aparecerá na imagem. Os parâmetros geométricos retirados da imagem dependerão portanto da posição relativa e do ângulo entre a câmera montada no robô e a marca [Borenstein 1996]. O que normalmente acontece é o fato de a acurácia diminuir ao passo que a distância relativa aumenta ou que o ângulo entre a câmera e a marca sai, digamos assim, de dentro de uma zona considerada razoável para o funcionamento do processo.

Uma outra abordagem do uso de marcas artificiais já citada anteriormente, é o que se chama de visão global [Kay 1993], que utiliza câmeras colocadas em locais fixos no ambiente de trabalho de modo a cobrir os locais de interesse. Nesses tipos de abordagens as marcas são colocadas nos robôs e podem ser identificadas a partir de uma imagem de uma das câmeras. Após a localização entram em ação implementações que visam predizer e diminuir o espaço de busca dentro da imagem de forma a otimizar o processo. Uma vantagem desse método é a capacidade de monitorar o robô e outras partes do ambiente ao mesmo tempo, e essa é justamente uma das principais vantagens que essa abordagem proporciona, e que se busca introduzir entre as habilidades proporcionadas pelo sistema de localização aqui proposto.

O objetivo principal é utilizar a informação do sistema de câmeras para monitorar o espaço de trabalho e localizar a uma baixa taxa de erro a posição dos objetos que se deseja monitorar dentro do ambiente de trabalho. Deve-se considerar também que pessoas vão e vem a todo momento e uma plataforma robótica deslocando-se dentro de um ambiente em constante mudança, necessita de um sistema robusto de localização que englobe outros recursos além dos fornecidos pelos sensores locais presentes no robô, pois elementos não previstos anteriormente só seriam bem identificados a distâncias menores do que um sistema baseado em visão global pode proporcionar, sem contar que as informações fornecidas por um sistema desse tipo são ricas, rápidas, apresentam melhor acurácia , tudo isso somado a uma evolução constante [Chen 2009].

A implementação e disponibilização de um sistema como esse para uma plataforma robótica deslocando-se dentro de ambientes dinâmicos envolve vários subproblemas, a começar pela localização das câmeras, passando pela aquisição de imagens e as correções necessárias, e o procedimento de encontrar e calcular a posição tanto na imagem quanto a real.

O sistema aqui proposto utiliza como plataforma de desenvolvimento base a lingua-

gem C++ e uma biblioteca muito utilizada dentro de visão computacional que é a biblioteca OpenCV. Essa biblioteca foi originalmente desenvolvida pela Intel e é caracterizada por ser multiplataforma rodando em Windows, Linux e MAC OS X. O foco dela é o processamento rápido de imagens. Entre suas áreas de aplicação encontram-se identificação de objetos, segmentação e e reconhecimento, motion tracking, robótica móvel e etc.

O processo como um todo pode ser dividido em duas partes, que são os procedimentos executados on-line e os executados off-line.

A proposta é que na parte off-line trabalhem as ferramentas de calibração, ou seja, nessa fase o usuário utilizaria as ferramentas de calibração radiométrica, de cores e geométrica para encontrar todos os parâmetros necessários. Depois com todas essas informações, seria feita então a passagem dos dados do objeto a ser localizado ao procedimento principal de monitoramento. Em resumo, na parte off-line tem-se todos os processos de calibração e especificação de parâmetros que serão utilizados pelas rotinas executadas online. Quanto a parte online, a proposta consiste em seis partes rodando em paralelo.

Essas seis partes seriam:

- Parte 1: Aquisição de Dados (imagens)
- Parte 2: Correção das Imagens
- Parte 3: Definição da Janela de Busca
- Parte 4: Busca na ROI
- Parte 5: Cálculo da posição e orientação do robô
- Parte 6: Atualização dos Dados

Cada parte trabalha exercendo a sua função e passando os dados obtidos para posterior processamento na etapa seguinte. Vale salientar que a proposta para a parte online é que a mesma funcione num modelo cliente/servidor atendendo as requisições de localização do objeto monitorado.

4.1 Procedimentos off-line

Essa estapa é realizada antes do sistema entrar em funcionamento e sua execução é necessária somente uma vez, desde que os parâmetros determinados aqui não sofram grandes modificações ao longo do tempo. Aqui são levados em consideração parâmetros ópticos, fotométricos e geométricos. Os procedimentos off-line nada mais são do que a parte de calibração de câmera discutidos no capítulo 2.

4.1.1 Entrada de dados e Aquisição de Imagens (objetivo de calibração)

Aqui parâmetros necessários ao funcionamento do sistema e que serão utilizados em etapas posteriores são introduzidos.

Nessa etapa também deve ser feita a correta disposição das câmeras ao longo do ambiente, de modo que a área coberta por cada câmera seja especificada em relação ao ambiente como um todo é também informada ao sistema.

Para a proposta aqui apresentada considera-se que as câmeras estão todas posicionadas no teto.

Na parte de aquisição de imagens com o objetivo de calibração são capturadas várias imagens dos padrões de calibração utilizados. A figura 4.1 exemplifica esses padrões. A captura aqui é controlada pelo usuário, que manipula a parte do sistema que é executado off-line realizando os procedimentos necessários.

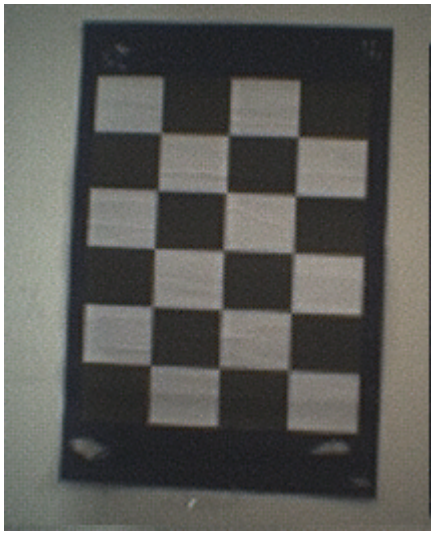


Figura 4.1: Exemplo de Padrão Utilizado

4.1.2 Calibração de Cores

Nessa etapa o nível médio de RGB de cada uma das cores disponíveis (azul, verde, rosa e amarelo) será calculado de acordo com a iluminação do ambiente que se deseja monitorar. As duas melhores cores que se destacarem mais no ambiente deverão ser escolhidas para compor a marca. O usuário deverá informar ao sistema quais foram essas duas cores e que serão consideradas pelo sistema dessa etapa em diante.

4.1.3 Calibração Radiométria

Os parâmetros determinados nessa eetapa serão utilizados na etapa de correção da imagem. Mais informações podem ser vistas no capítulo 2.

4.1.4 Calibração Geométrica

O OpenCV possui rotinas que estimam tanto os parâmetros intrínsecos quanto os extrínsecos.

Calibração de Parâmetros Intrínsecos

Ao utilizar as rotinas do OpenCV para esses cálculos, primeiramente os parâmetros internos são estimados e só então em cima dessa estimativa é que os externos são calculados. Diante disso se os primeiros forem estimados de maneira incorreta implicará também no mal dimensionamento dos outros parâmetros o que acaba por comprometer o funcionamento de todo o sistema. Devido a isso, foi utilizado tanto uma rotina em Matlab quanto uma em OpenCV durante os testes para o cálculo dos parâmetros internos. O que acabou por ser constatado ao se utilizar os valores encontrados no cálculo dos parâmetros externos foi que os dados obtidos pela rotina do Matlab proporcionou parâmetros extrínsecos que forneciam melhores resultados no processo de calculo da posição no mundo a partir da imagem. Sendo assim, optou-se por durante a fase inicial estimar os dados intrínsecos da câmera a partir da rotina em Matlab e nas fases posteriores de desenvolvimento utilizar a rotina em OpenCV. Uma imagem do toolbox do matlab utilizado pode ser vista na figura 4.2.

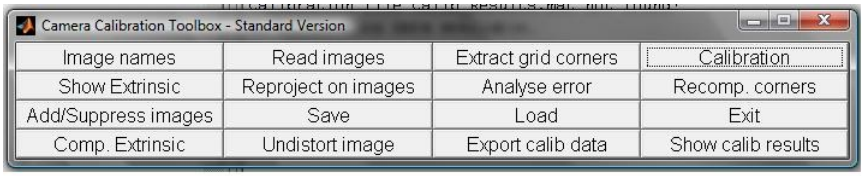


Figura 4.2: Imagem do Toolbox do Matlab

Outro fato a ser observado durante esse processo de calibração é que o padrão de imagens utilizado deve variar de posição ao longo da aquisição de dados pois imagens muito semelhantes implicam em dados insuficientes para a rotina de calibração. Em outras palavras os pontos do padrão de calibração na imagem não podem ser em sua maioria coplanares.

Calibração de Parâmetros Extrínsecos

Esses parâmetros relacionam o referencial de câmera com o referencial de mundo, ou seja, relaciona o frame de câmera com o frame de mundo. Como já falado anteriormente , o OpenCV possui funções específicas para esse fim. Nessa etapa, uma vez que os parâmetros internos tenham sido estimados no passo anterior, as funções do OpenCV os recebem como uma de suas entradas e calculam os parâmetros extrínsecos.

A calibração para encontrar os parâmetros extrínsecos consiste de três fases. Antes de tudo é necessário entretanto, que as imagens do padrão de calibração tenham sido devidamente capturadas e os ponto no mundo que o padrão representa na imagem também devem ser conhecidos. O conjunto de imagens do padrão visto é primeiramente passado por uma função que realiza um cálculo inicial para encontrar os pontos utilizados, que no caso são os pontos entre os quadrados pretos e os brancos, pontos destacados em vermelho na figura 4.3.

Figura 4.3: Exemplo de Imagem Utilizada

Essa função faz somente um cálculo inicial, por isso o mesmo conjunto de imagens e os pontos na imagem estimados anteriormente, são passados para uma segunda função para que seja realizado um cálculo mais acurado. Isso é feito utilizando os dados anteriores como base, já que a busca agora em vez de se realizar na imagem toda, é feita somente numa região ao redor de cada ponto já calculado. Após os pontos calculados e otimizados, o conjunto dos pontos imagem e seus correspondentes no mundo, juntamente com os parâmetros extrínsecos são passados para uma terceira função responsável por calcular os parâmetros extrínsecos.

Dessa forma os parâmetros extrínsecos e intrínsecos são os responsáveis por relacionar um ponto no mundo com o mesmo ponto em relação a câmera e este mesmo ainda em coordenadas de imagem, ou seja, pixels.

Finalizada essa etapa, o sistema já possui os dados necessários para realizar os procedimentos de localizar as marcas e realizar os cálculos necessários para obter o posicionamento do robô no ambiente, executados no procedimento online.

4.2 Procedimentos online

Os passos executados online são enunciados a seguir:

4.2.1 Aquisição de Dados

Nessa etapa é feita a captura das imagens que servirão de base para o funcionamento de todo o sistema.

4.2.2 Correção da imagem

Essa parte do sistema realiza a correção da imagem utilizando os coeficientes de distorção da imagem encontrados nos processos de calibração off-line. Tal correção é levada a execução sem no entanto prejudicar o funcionamento global. Uma preocupação sempre presente é que as correções executadas não devem prejudicar em demasia essa característica de funcionamento.

4.2.3 Definição da janela de busca

Nessa etapa é realizada a definição da região de interesse da imagem onde será realizada a busca pela marca. Primeiramente a busca é realizada em toda a imagem. Após encontrar o primeiro nível de RGB desejado, a busca passa a ser feita somente na área próxima.

4.2.4 Procura da marca

Aqui uma adaptação do floodfill em conjunto com um algoritmo de busca que leva em consideração o nível de cor de cada uma das duas cores utilizadas na marca são executados para realizar a busca pela mesma. Primeiramente busca-se uma cor e acha-se o centro da região que a representa na imagem. O mesmo procedimento é realizado para a outra cor da marca. Após os dois centros serem encontrados a orientação é calculada.

4.2.5 Cálculo da Posição e Orientação

Nessa etapa o cálculo propriamente dito da posição e orientação do robô é realizado.

4.2.6 Atualização dos Dados

Esse procedimento fica sempre em execução esperando as requisições de localização dos objetos monitorados.

Uma visão geral do que foi enumerado anteriormente é mostrado na figura 4.4.

4.3 Algoritmo

Com as partes da etapa online executando em conjunto e como a entrada de uma depende da saída da anterior, quando um dado é tratado por uma dessas partes, ele é imediatamente passado ao nível seguinte, afim de que novos dados possam ser tratados. Isso tudo, é claro, respeitando os requisitos de disponibilidade do nível seguinte, para que não haja perda ou conflito de informações.

Primeiramente tem-se a execução das rotinas de aquisição e correção de imagens efetuando a captura e tratamento das imagens. Mais uma vez funções OpenCv são utilizadas para isso. Quando um pedido de nova imagem é enviado, essa rotina então repassa o frame para as rotinas seguintes de definição da região de busca e busca da marca. Com já foi falado anteriormente, a definição da região de interesse baseia-se na posição anteriormente calculada. A busca pela marca tem como principal ferramenta o algoritmo flood fill, também conhecido como seed fill, o qual tem como função achar uma determinada região conexa de pixels numa imagem onde a variação de níveis RGB está dentro de um determinado limite especificado. Isso é bastante útil já que uma região de mesma cor na imagem apresenta valores RGB próximos uns dos outros. Dessa maneira a região de interesse é percorrida e quando uma região conexa de pixels semelhantes é encontrada, e é constatado que se trata de uma das regiões procuradas, as coordenadas dos pontos pertencentes a essa região são passadas para o nível seguinte, liberando assim essa etapa para uma nova procura. Dessa maneira, a maior parte do processamento encontra-se nessa parte do procedimento online.

A rotina de cálculo de posição tem por função receber as coordenadas dos pontos da região encontrada e baseando-se nelas calcular o centro da mesma. De cada uma das duas regiões recebidas o centro é calculado. Após isso, quando os centros de ambas as regiões de cores que formam o padrão utilizado para localização tiverem sido encontra-

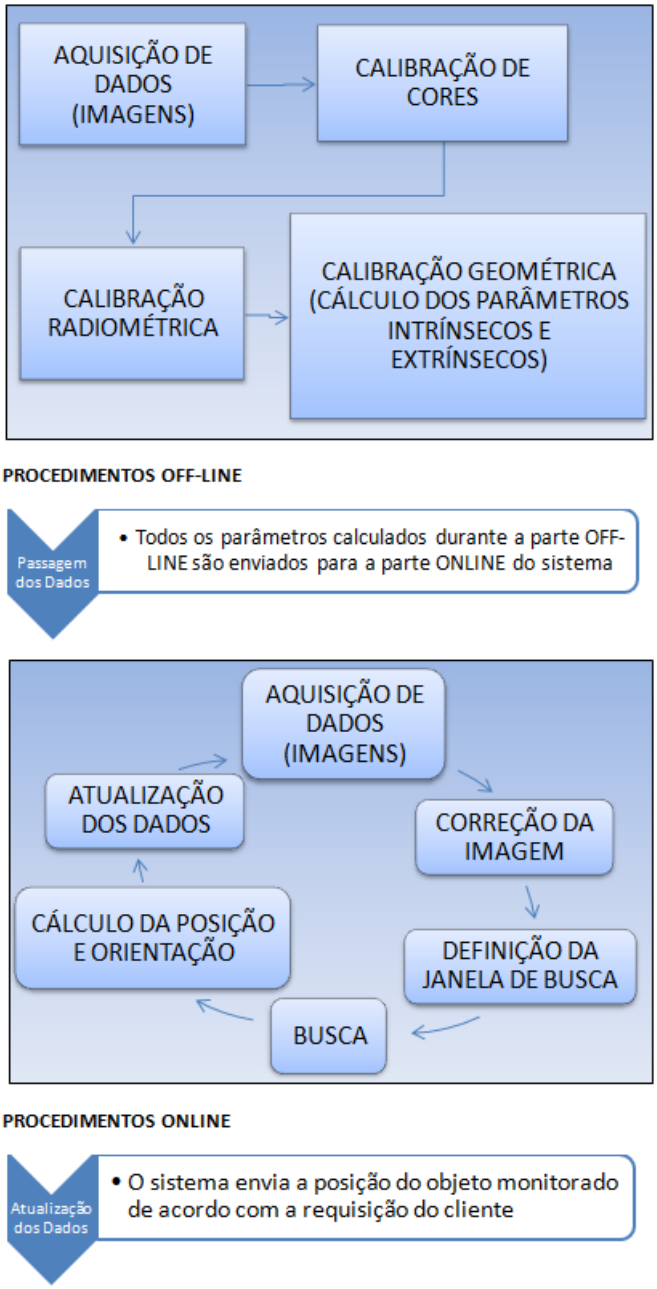


Figura 4.4: Esquema do Sistema Proposto

dos, é a partir desses dados que é calculada a posição e orientação do robô no ambiente monitorado.

Uma visão geral do algoritmo é mostrada em 4.5.

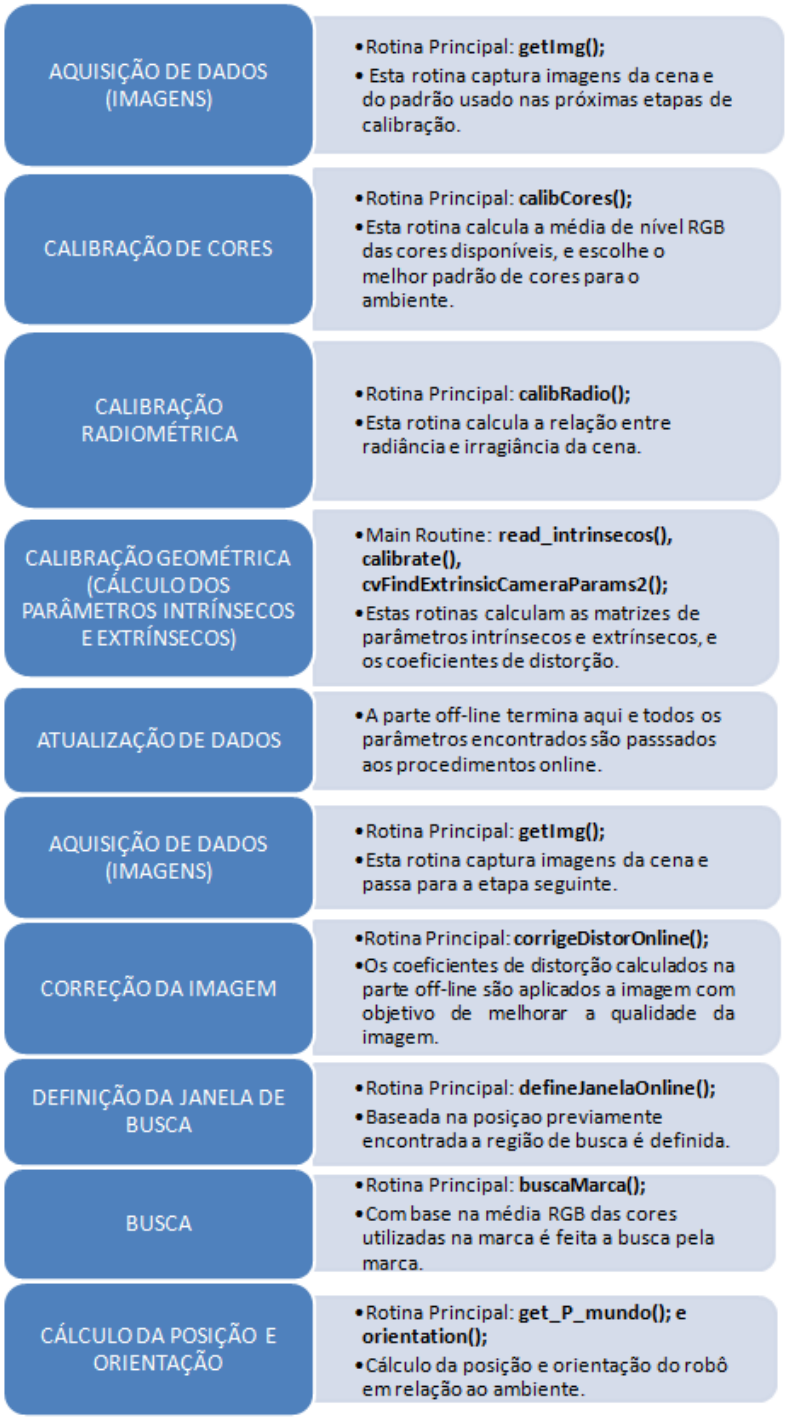


Figura 4.5: Esquema Geral do Algoritmo Proposto

4.4 Implementação

A base do algoritmo de localização, como já discutido anteriormente, está em procurar por marcas propositalmente colocadas em cima do robô.

4.4.1 Requisitos Necessários

Todo o sistema foi desenvolvido em C++ com o adicional de algumas rotinas matemáticas que foram utilizadas por proporcionarem melhores resultados. Tais rotinas podem ser visualizadas em algum programa matemático como Scilab ou MatLAB. Usou-se também a biblioteca OpenCV de visão computacional. Todo o sistema foi desenvolvido para rodar a priori em um ambiente linux.

Dessa maneira para um bom funcionamento o sistema necessita que:

- Sistema operacional linux
- A biblioteca OpenCV esteja devidamente instalada. Toda a informação necessária pode ser encontrada em http://opencvlibrary.sourceforge.net/.
- Para um melhor funcionamento um programa matemático para uma melhor visualização de resultados.
- Uma camera para a aquisicao de imagens (A camera utilizada durante o desenvolvimento e teste foi uma camera LEGO)

4.4.2 Diretório de Trabalho

Dentro da pasta de trabalho do sistema serão encontrados os seguintes arquivos, que compõem o sistema:

- calibrar.cpp: guarda o procedimento de calibração da camera. Um extrato do código pode ser visto em 4.6.
- buscarMarca.cpp: procedimento principal de busca pelas marcas na imagem. Um extrato do código pode ser visto em 4.7.
- header.h: nesse arquivo são declaradas todas as variáveis globais do sistema bem como as funções globais implementadas em header.cpp. Um extrato do código pode ser visto em 4.8.
- header.cpp: aqui são inicializadas todas as variáveis declaradas em header.h e também implementadas as funções declaradas nesse arquivo. Um extrato do código pode ser visto em 4.9.

- mainCalibrate.cpp: é o procedimento principal de calibração onde estão presentes as chamadas a todos as funções necessárias ao procedimento como um todo. s funções chamadas aqui estão todas implementadas em header.cpp, com exceção da função principal que está implementada em calibrar.cpp. Um extrato do código pode ser visto em 4.10.
- mainPosition.cpp: é o procedimento principal de localização onde estão presentes todas as chamadas ao procedimento como um todo. As funções chamadas aqui estão todas implementadas em header.cpp, com exceção da função principal que está implementada em buscarMarca.cpp e da função de aquisição de imagens. Um extrato do código pode ser visto em 4.11.
- getImages: guarda a função de aquisição de imagens. É utilizada pelo procedimento mainPosition.cpp
- MAKEFILE: é o arquivo responsável pela compilação dos códigos do sistema.
- Arquivos.txt: toda a estrutura de entrada e saida do sistema está baseada em leitura e escrita de arquivos, de forma que cada função ao precisar de dados de entrada buscará os mesmos no arquivo indicado no código. O mesmo acontece quando dados precisam ser salvos. Cada função que retorna um conjunto de dados escreve os mesmos num arquivo indicado no código. O procedimento de calibração faz uso dos arquivos intrinsecos.txt, objPoints.txt, extrinsecos.txt e Mextrinsecos.txt, onde os dois primeiros são arquivos de entrada e os dois últimos de saída. O procedimento de localização faz uso dos arquivos intrinsecos.txt, extrinsecos.txt, Mextrinsecos.txt, ParamMarcas.txt e posicao.txt, onde os quatro primeiros fornecem dados de entrada e o último é onde o algoritmo guarda a posição e orientação atual calculadas.
- Arquivos de imagem: são os arquivos padrão de entrada do sistema. Fontes de dados para o processamento executado.

4.4.3 Procedimento de Execução

De acordo com o que já foi discutido anteriormente, o sistema pode ser dividido em duas partes: a de calibração e a de localização. Primeiramente é feita a calibração, que necessita ser executada apenas uma vez, para depois ser iniciado o processo de localização, o qual utiliza os dados obtidos na calibração. Uma nova calibração somente necessitará ser executada caso a câmera seja movida de lugar.

A transformação de um ponto na imagem(2D) para um ponto no mundo (3D) necessita de um conjunto de parâmetros organizados em duas matrizes, as quais são a matriz de

parâmetros intrínsecos e a matriz de parâmetros extrínsecos, e que estão armazenadas nos arquivos intrinsecos.txt e Mextrinsecos.txt respectivamente.

O procedimento de calibração recebe justamente os parâmetros intrínsecos como entrada. A partir dessa matriz os parâmetros extrínsecos são calculados. Dessa forma é necessário que o usuário esteja de posse dos parâmetros intrínsecos da câmera que deseja utilizar para que a calibração, e posteriormente o processo de localização possam ser efetuados. Uma ótima ferramenta e de fácil uso para a descoberta dos parâmetros intrínsecos da câmera pode ser encontrada em [*www.vision.caltech.edu/bouguetj/calibdoc* n.d.]. Além da matriz de parâmetros intrínsecos essa feramenta calcula ainda a matriz de coeficientes de distorção e que também fazem parte do arquivo intrínsecos.txt. Complementando esses parâmetros temos ainda a distância da câmera até o plano onde o robô se movimenta e a área real que a câmera pode "enxergar"do mundo real. Sendo assim dentro do arquivo intrínsecos.txt devem constar, nessa mesma ordem, primeiramente um conjunto de nove parâmetros representando a matriz 3x3 de parâmetros intrínsecos, seguido pelos quatro números que constituem a matriz 1x4 de coeficientes de distorção. As linhas seguintes guardam a distância real da câmera ao plano de locomoção do robô e por fim os últimos dois números que representam a região real que pode ser coberta pela câmera.

O procedimento de localização utiliza como entrada os arquivos objPoints.txt, ParamMarcas.txt, e o arquivo Mextrinsecos.txt, o qual é gerado após a execução da calibração.

Dentro de objPoints.txt devem ser especificados quatro parâmetros, que são as distâncias no eixo x e eixo y respectivamente, da origem onde foi colocado o padrão de calibração a origem da área total vista pela câmera. Esses são os dois primeiros números. Os dois últimos representam o tamanho em x e em y de uma célula unitária do padrão de calibração. Esses parâmetros podem ser observados na figura 4.12.

A distância em Y da origem da área vista pela câmera a origem é indicada por Dy e a distância e X é dada por Dx.

O outro arquivo, ParamMarcas.txt, tem-se na primeira linha cinco número que representam os níveis RGB que devem ser procurados na imagem, três primeiros números, seguidos por dois números que representam o tamanho em pixels da marca na imagem. Isso vale para a primeira marca. Na linha seguinte tem-se esses mesmos parâmetros só que correspondendo a segunda marca a ser procurada.

O último arquivo gerado é então o arquivo posicao.txt, o qual é gerado na saída do procedimento de localização e é atualizado a cada passagem do algoritmo.

Com esses arquivos citados anteriormente devidamente completos com todos os parâmetros, o procedimento de calibração e localização podem ser então rodados. Antes disso ele devem ser compilados, o que pode ser feito digitando o comando "make"no console

do linux, dentro do diretório onde se encontram os arquivos. Após a compilação serão gerados dois executáveis intitulados 'calibracao' e 'localizacao', que deverão ser executados digitando-se ./calibracao e ./localizacao no mesmo console onde foi dado o comando "make".

Deve-se ter em mente que todos os arquivos, tanto imagens, *.txt's e demais arquivos devem todos ser mantidos no mesmo diretório.

calibCores.cpp | getImages.cpp | mainCalibrate.cpp | buscarMarca.cpp | buscaTeste.cpp | mainPosition.cpp* | headerFuncoes.cpp* | headerVariaveis.h | calibrar.cpp

(Unknown Scope)

```
#include "cv.h"
#include "cxcore.h"
#include "highgui.h"
#include <stdio.h>
#include <math.h>
#include "headerVariaveis.h"
//funcao de calibracao. A partir dos parametros intrinsecos os parametros extrinsecos sao calculados
void calibrate(CvMat* intrinsic_matrix_local, CvMat* distortion_coeffs_local, double v_obj_pt[][3], IplImage* imagem_local[],int n_iter){

  //Variáveis
  int a_local,b_local,inc_local;
  //Variáveis Internas OpenCV
  IplImage* imagem_local_thresh; IplImage* imagem_local_fixed;
  CvMat* object_points_local;
  CvMat* image_points_local;
  CvMemStorage* memstorage_local;
  CvSize etalonsize_local;
  CvPoint2D32f* vet_corners_local;
  int* cornerCount_local = new int;
  //*********************************
  CvSize win_local;
  CvSize zeroZone_local;
  //*********************************
  CvMat* rotation_vector_local;
  CvMat* translation_vector_local;
  //*********************************
  CvMat* rotation_matrix_local;
  //Variáveis de escrita
   ofstream outdata; // outdata is like cin
   double num; // list of output values

  //Inicialização Variáveis Internas
  image_points_local = cvCreateMat(CHESSBOARD_WIDTH*CHESSBOARD_HEIGHT,2,CV_32FC1);
  object_points_local = cvCreateMat(CHESSBOARD_WIDTH*CHESSBOARD_HEIGHT,3,CV_32FC1);
  memstorage_local = cvCreateMemStorage();
  etalonsize_local = cvSize(CHESSBOARD_WIDTH,CHESSBOARD_HEIGHT);
  vet_corners_local = new CvPoint2D32f[CHESSBOARD_WIDTH*CHESSBOARD_HEIGHT];

  //*****************************************************************************************
  win_local = cvSize(5,5);
  zeroZone_local = cvSize(-1,-1);
  //************************ *****
  rotation_vector_local = cvCreateMat(1,3,CV_32FC1);
  translation_vector_local = cvCreateMat(1,3,CV_32FC1);
  rotation_matrix_local = cvCreateMat(3,3,CV_32FC1);
  /* ... */
  for (int z = 0; z < n_iter; z++){
  /* ... */
    inc_local = (CHESSBOARD_WIDTH*CHESSBOARD_HEIGHT);
    for (b_local=0; b_local<(CHESSBOARD_WIDTH*CHESSBOARD_HEIGHT); b_local++){
        inc local--;

  //o processo eh realizado duas vezes. Na primeira passagem sao calculados valores iniciais e na segunda esses valores sao ajustados utilizando a imagem corrigida
  for (int z = 0; z < n_iter; z++){
  //for (int i = 0; i < 2; i++){
    //printf("Passagem %d %d \n",z,i);
    //inicialização matriz de pontos de mundo
    inc_local = (CHESSBOARD_WIDTH*CHESSBOARD_HEIGHT);
    for (b_local=0; b_local<(CHESSBOARD_WIDTH*CHESSBOARD_HEIGHT); b_local++){
        inc_local--;
        object_points_local->data.fl[b_local*3+0] = v_obj_pt[b_local][0];
        object_points_local->data.fl[b_local*3+1] = v_obj_pt[b_local][1];
        object_points_local->data.fl[b_local*3+2] = v_obj_pt[b_local][2];
        }

for (b_local=0; b_local<(CHESSBOARD_WIDTH*CHESSBOARD_HEIGHT); b_local++)
    printf("ObjPts: %f %f %f \n",object_points_local->data.fl[b_local*3+0], object_points_local->data.fl[b_local*3+1], object_points_local->data.fl[b_local*3+2]);

 imagem_local_thresh = cvCreateImage(cvSize(imagem_local[z]->width,imagem_local[z]->height), imagem_local[z]->depth,imagem_local[z]->nChannels);

 //acha os pontos de calibracao na imagem que correspondem aos valores carregados do arquivo objPoints.txt
 cvFindChessBoardCornerGuesses(imagem_local[z],imagem_local_thresh,memstorage_local,etalonsize_local, vet_corners_local, cornerCount_local);
printf(" corner count %d \n",*cornerCount_local);

//for(b_local = 0; b<5; b_local++)
    printf("%f %f \n",vet_corners_local[0].x,vet_corners_local[0].y);
printf("%f %f \n",vet_corners_local[1].x,vet_corners_local[1].y);
printf("%f %f \n",vet_corners_local[2].x,vet_corners_local[2].y);

read_Pt_img();
for (b_local=0; b_local<(CHESSBOARD_WIDTH*CHESSBOARD_HEIGHT); b_local++){
    vet_corners_local[b_local].x = v_img_pt[b_local][0];
    vet_corners_local[b_local].y = v_img_pt[b_local][1];
    printf("%f %f \n",vet_corners_local[b_local].x,vet_corners_local[b_local].y);
}

 // ajuste dos valores encontrados
 cvFindCornerSubPix(imagem_local[z],vet_corners_local,*cornerCount_local,win_local,zeroZone_local,   cvTermCriteria(CV_TERMCRIT_ITER|CV_TERMCRIT_EPS,20,0.01));
  //Escrita do arquivo
  outdata.open("/home/rafaella/Projetos/CamSis100/posicao/imgPointsFix.txt"); // opens the file
  if( !outdata ) { // file couldn't be opened
     cerr << "Error: file extrinsecos.txt could not be opened" << endl;
     exit(1);
  }

for (b_local=0; b_local<(CHESSBOARD_WIDTH*CHESSBOARD_HEIGHT); b_local++){
    printf("%f %f \n",vet_corners_local[b_local].x,vet_corners_local[b_local].y);
    outdata << vet_corners_local[b_local].x << "  ";
    outdata << vet_corners_local[b_local].y << endl;
}

  outdata<< endl;
  outdata.close();



  //inicialização matriz de coordenadas dos pontos na imagem_local
  for (b_local=0; b_local<(CHESSBOARD_WIDTH*CHESSBOARD_HEIGHT); b_local++){
        image_points_local->data.fl[b_local*2+0] = vet_corners_local[b_local].x;
        image_points_local->data.fl[b_local*2+1] = vet_corners_local[b_local].y;
        }
 //calculo dos parametros extrinsecos
 cvFindExtrinsicCameraParams2(object_points_local,image_points_local,intrinsic_matrix_local,  distortion_coeffs_local,rotation_vector_local,translation_vector_local);

  cvRodrigues2(rotation_vector_local,rotation_matrix_local);

  imagem_local_fixed = cvCreateImage(cvSize(imagem_local[z]->width,imagem_local[z]->height), imagem_local[z]->depth,imagem_local[z]->nChannels);
  //cvSaveImage("image_antes.bmp",imagem_local[z]);
  cvUndistort2(imagem_local[z],imagem_local_fixed,intrinsic_matrix_local,distortion_coeffs_local);
  //cvSaveImage("image_depois.bmp",imagem_local_fixed);
  imagem_local[z] = cvCloneImage( imagem_local_fixed );

//}
}
   //Escrita do arquivo
   outdata.open("/home/rafaella/Projetos/CamSis100/posicao/extrinsecos.txt"); // opens the file
   if( !outdata ) { // file couldn't be opened
      cerr << "Error: file extrinsecos.txt could not be opened" << endl;
      exit(1);
```

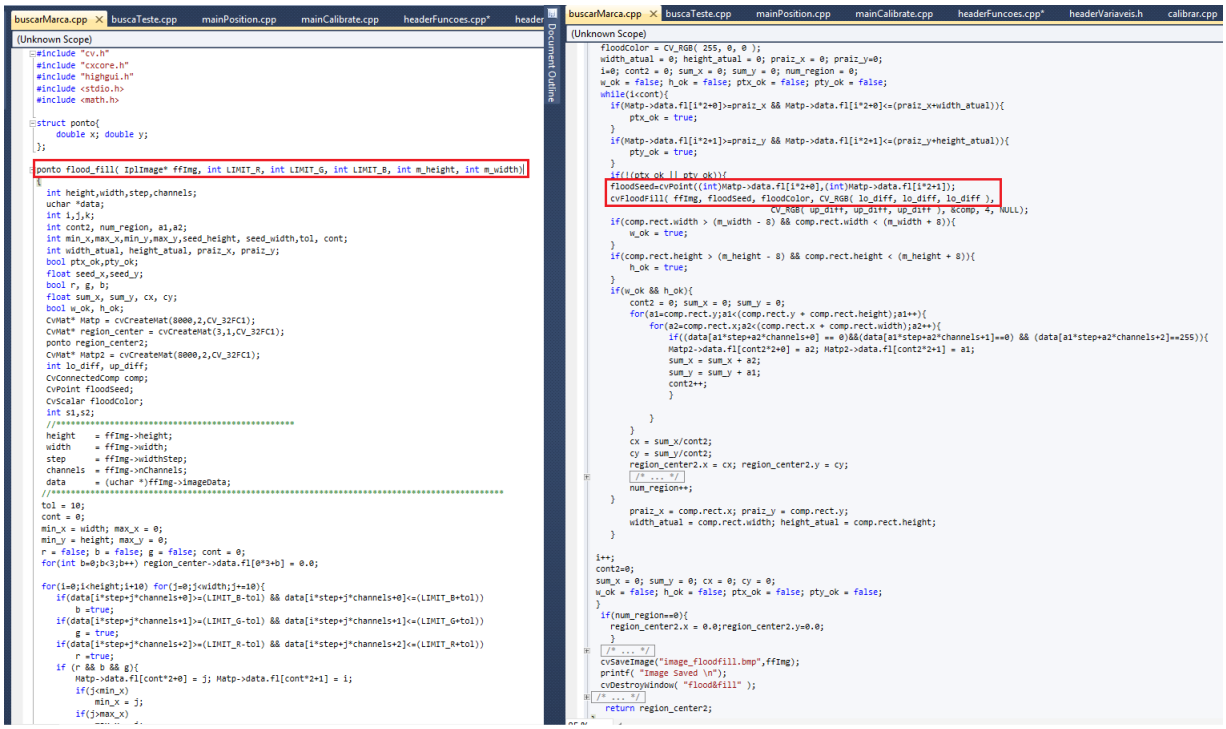


Figura 4.7: Extrato do Arquivo buscarMarca.cpp

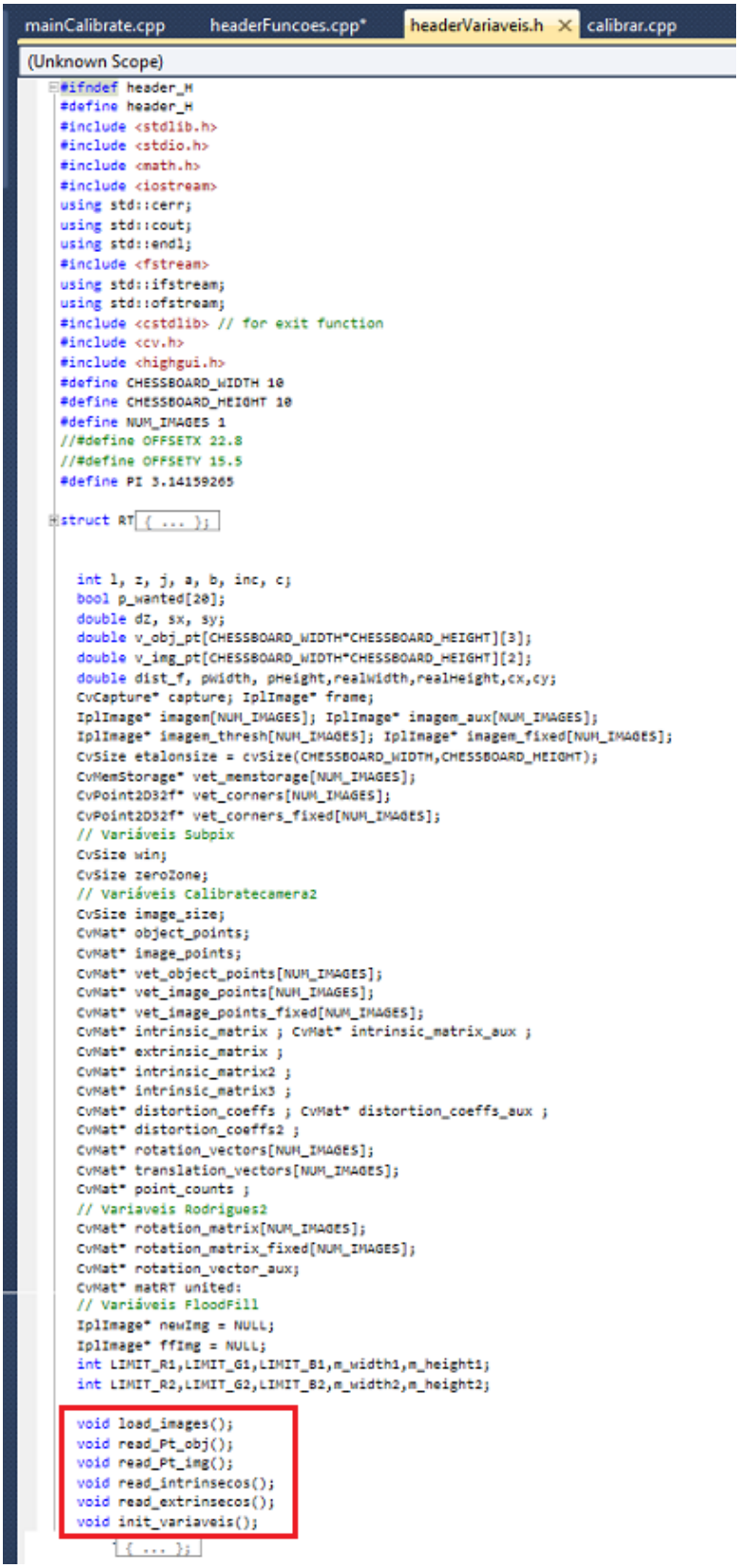
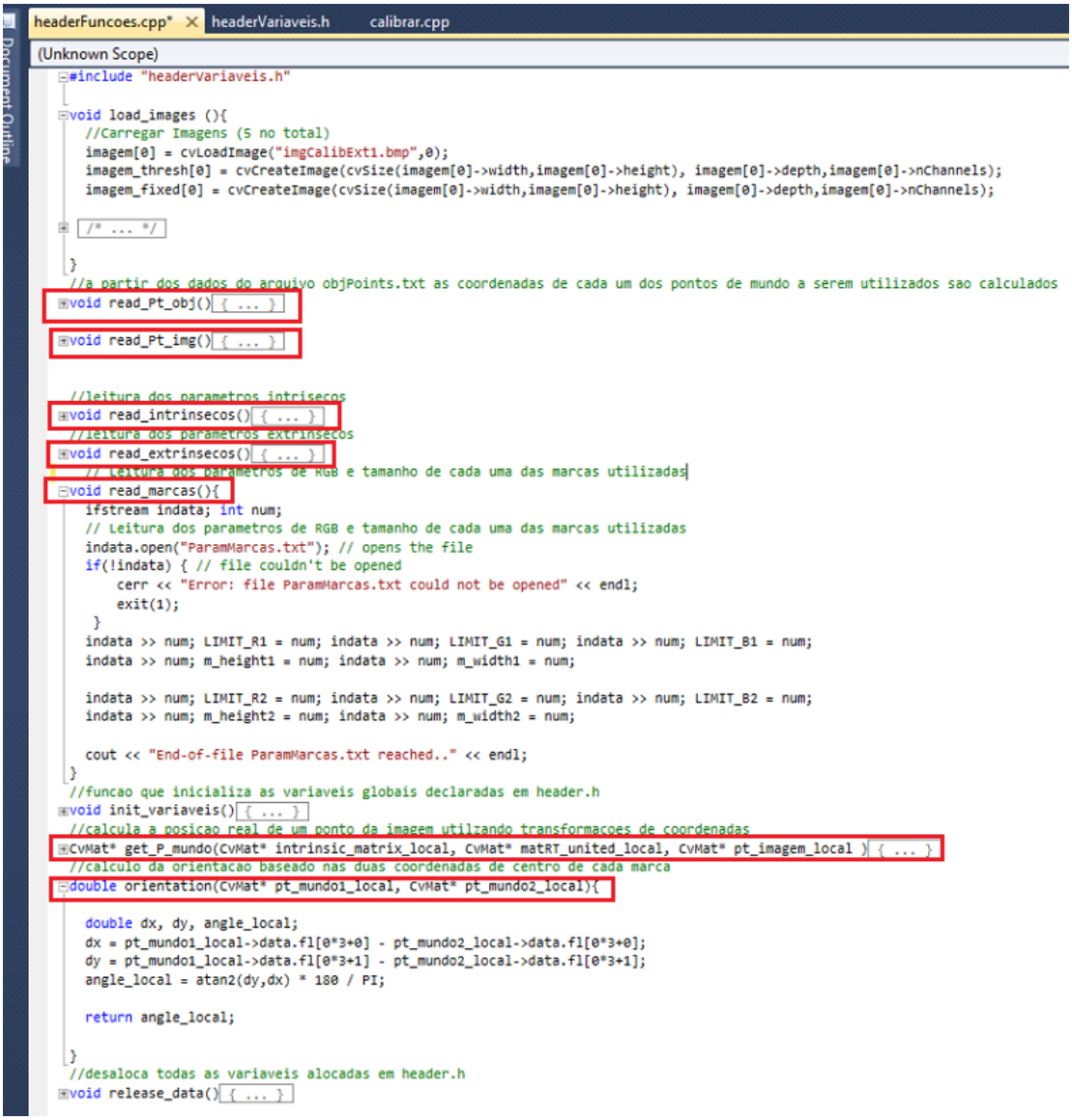


Figura 4.8: Extrato do Arquivo header.h



```
headerFuncoes.cpp*  headerVariaveis.h  calibrar.cpp
(Unknown Scope)

#include "headerVariaveis.h"

void load_images (){
    //Carregar Imagens (5 no total)
    imagem[0] = cvLoadImage("imgCalibExt1.bmp",0);
    imagem_thresh[0] = cvCreateImage(cvSize(imagem[0]->width,imagem[0]->height), imagem[0]->depth,imagem[0]->nChannels);
    imagem_fixed[0] = cvCreateImage(cvSize(imagem[0]->width,imagem[0]->height), imagem[0]->depth,imagem[0]->nChannels);

    /* ... */

}
//a partir dos dados do arquivo objPoints.txt as coordenadas de cada um dos pontos de mundo a serem utilizados sao calculados
void read_Pt_obj(){ ... }

void read_Pt_img(){ ... }


//leitura dos parametros intrisecos
void read_intrinsecos(){ ... }
//leitura dos parametros extrinsecos
void read_extrinsecos(){ ... }
// Leitura dos parametros de RGB e tamanho de cada uma das marcas utilizadas
void read_marcas(){
    ifstream indata; int num;
    // Leitura dos parametros de RGB e tamanho de cada uma das marcas utilizadas
    indata.open("ParamMarcas.txt"); // opens the file
    if(!indata) { // file couldn't be opened
        cerr << "Error: file ParamMarcas.txt could not be opened" << endl;
        exit(1);
     }
    indata >> num; LIMIT_R1 = num; indata >> num; LIMIT_G1 = num; indata >> num; LIMIT_B1 = num;
    indata >> num; m_height1 = num; indata >> num; m_width1 = num;

    indata >> num; LIMIT_R2 = num; indata >> num; LIMIT_G2 = num; indata >> num; LIMIT_B2 = num;
    indata >> num; m_height2 = num; indata >> num; m_width2 = num;

    cout << "End-of-file ParamMarcas.txt reached.." << endl;
}
//funcao que inicializa as variaveis globais declaradas em header.h
void init_variaveis(){ ... }
//calcula a posicao real de um ponto da imagem utilzando transformacoes de coordenadas
CvMat* get_P_mundo(CvMat* intrinsic_matrix_local, CvMat* matRT_united_local, CvMat* pt_imagem_local ){ ... }
//calculo da orientacao baseado nas duas coordenadas de centro de cada marca
double orientation(CvMat* pt_mundo1_local, CvMat* pt_mundo2_local){

    double dx, dy, angle_local;
    dx = pt_mundo1_local->data.fl[0*3+0] - pt_mundo2_local->data.fl[0*3+0];
    dy = pt_mundo1_local->data.fl[0*3+1] - pt_mundo2_local->data.fl[0*3+1];
    angle_local = atan2(dy,dx) * 180 / PI;

    return angle_local;

}
//desaloca todas as variaveis alocadas em header.h
void release_data(){ ... }
```

Figura 4.9: Extrato do Arquivo header.cpp

mainCalibrate.cpp | headerFuncoes.cpp* | headerVariaveis.h | calibrar.cpp

(Unknown Scope)

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <cv.h>
#include <highgui.h>
#include "headerVariaveis.h"
//#include "calibrar.cpp"

int main(int argc, char** argv)
{
  load_images();
  init_variaveis();//funcao que inicializa as variaveis globais declaradas em header.h
  read_intrinsecos();//leitura dos parametros intrinsecos guardados no arquivo intrinsecos.txt
  //read_extrinsecos();//leitura dos parametros extrinsecos
  read_Pt_obj();//leitura dos pontos correspondente ao padrao de calibracao de parametros extrinsecos
   //carregamento das imagens para a funcao de calibracao
  IplImage* Img_cena;
  IplImage* Img_aux;
  //*************************************************************************************************************
  for (int a=0; a<(NUM_IMAGES); a++)
    imagem_aux[a] = cvCloneImage(imagem[a]);
  //funcao que realiza a calibracao
  calibrate(cvCloneMat(intrinsic_matrix),cvCloneMat(distortion_coeffs),v_obj_pt,imagem_aux,1);
  //*************************************************************************************************************
  //release structures
  release_data();
  return 0;
}
```

Figura 4.10: Extrato do Arquivo mainCalibrate.cpp

buscarMarca.cpp | buscaTeste.cpp | mainPosition.cpp* | headerFuncoes.cpp* | headerVariaveis.h | calibrar.cpp

(Unknown Scope)

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <cv.h>
#include <highgui.h>
#include "header.h"
#include "buscarMarca.cpp"
#include "getImages.cpp"
int main(int argc, char** argv)
{
  init_variaveis();//funcao que inicializa as variaveis globais declaradas em header.h
  read_intrinsecos();//leitura dos parametros intrisecos
  read_extrinsecos();//leitura dos parametros extrinsecos
  read_marcas();//leitura dos parametros RGB e tamanho das marcas
  double angle = 0.0;
  IplImage* Img_cena;
  IplImage* Img_aux; IplImage* Img_aux2;
  //Variáveis de escrita
  ofstream outdata; // outdata is like cin
  //*****************************************************************************************
  while(1){
  Img_cena = getImg();//funcao que captura a imagem da camera. Ela espera 30s ateh a camera se ajustar a iluminacao
  //Img_cena = cvLoadImage("P2cor1T2.bmp",1); //Ao descomentar essa linha, deve -se comentar a anterior. Esse comando carrega uma imagem como forma de exemplificar o funcionamento do algoritmo
  //*****************************************************************************************
  //Busca pela marca 1
  Img_aux=cvCloneImage( Img_cena );
  //Img_aux=cvCloneImage( Img_aux2 );
  pt_imagem1_atual = flood_fill(Img_aux,LIMIT_R1,LIMIT_G1,LIMIT_B1,m_height1,m_width1);
  printf("Pt Imagem 1: ");
  for(b=0;b<2;b++)
    printf("%f ",pt_imagem1_atual->data.fl[0*2+b]);
    printf("\n");
  pt_mundo1_atual = get_P_mundo(cvCloneMat(intrinsic_matrix), cvCloneMat(extrinsic_matrix), cvCloneMat(pt_imagem1_atual));
  printf("Pt Mundo 1: ");
  for(b=0;b<3;b++)
    printf("%f ",pt_mundo1_atual->data.fl[0*3+b]);
    printf("\n");
  //*****************************************************************************************
  //Busca pela marca 2
  Img_aux=cvCloneImage( Img_cena );
  //Img_aux=cvCloneImage( Img_aux2 );
  pt_imagem2_atual = flood_fill(Img_aux,LIMIT_R2,LIMIT_G2,LIMIT_B2,m_height2,m_width2);
  printf("Pt Imagem 2: ");
  for(b=0;b<2;b++)
    printf("%f ",pt_imagem2_atual->data.fl[0*2+b]);
    printf("\n");
  pt_mundo2_atual = get_P_mundo(cvCloneMat(intrinsic_matrix),cvCloneMat(extrinsic_matrix), cvCloneMat(pt_imagem2_atual));
  printf("Pt Mundo 2: ");
  for(b=0;b<3;b++)
    printf("%f ",pt_mundo2_atual->data.fl[0*3+b]);
    printf("\n");
  //*****************************************************************************************
  //calculo da orientacao final do robo baseado nos centros de ambas as marcas
  for(b=0;b<3;b++)
    pt_mundo_final->data.fl[0*3+b] = (pt_mundo1_atual->data.fl[0*3+b]+pt_mundo2_atual->data.fl[0*3+b])/(2.0);
  printf("Pt Mundo Final: ");
  for(b=0;b<3;b++)
    printf("%f ",pt_mundo_final->data.fl[0*3+b]);
    printf("\n");
  //*****************************************************************************************
  //calculo da orientacao
  angle = orientation(pt_mundo1_atual,pt_mundo2_atual);
  printf("Oritentacao: %f \n",angle);
  //*****************************************************************************************
   //Escrita do arquivo
   outdata.open("posicao.txt"); // opens the file
   if( !outdata ) { // file couldn't be opened
      cerr << "Error: file posicao.txt could not be opened" << endl;
      exit(1);
   }
   for(b=0;b<3;b++)
    outdata << pt_mundo_final->data.fl[0*3+b] << "  ";
   outdata<< endl; outdata<< endl;
   outdata << angle; outdata<< endl;
   outdata.close();
   }
  //*****************************************************************************************
  //release structures
  release_data();
  return 0;
}
```

Figura 4.11: Extrato do Arquivo mainPosition.cpp

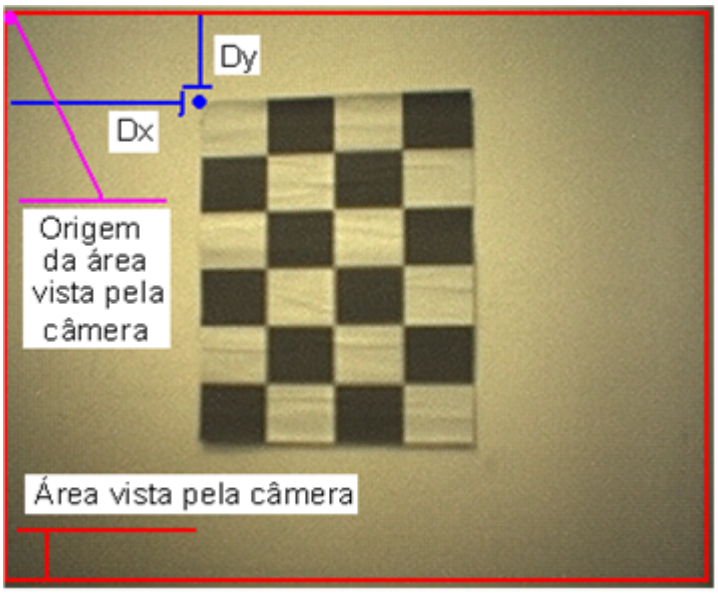


Figura 4.12: Esquema das informações armazenadas em objPoints.txt

Capítulo 5

Experimentos Realizados

Até a etapa atual de desenvolvimento foram realizados alguns experimentos, principalmente no tocante a parte de calibração. Os mais bem sucedidos são portanto mostrados aqui. Uma câmera Logitech Quickcam LEGO é utilizada. A intenção é manter o uso de webcams em virtude do baixo custo.

A cena montada para os testes contou com a câmera instalada a 133.5 cm do plano onde foram fixados os padrões de calibração utilizados. O ângulo de abertura da câmera utilizada proporciona, a essa distância, que a mesma cubra uma área de 102cm x 83 cm.

O padrão para o teste das cores pode ser visto na figura 5.1.

Figura 5.1: Padrão Teste de Cores

Utilizando-se esse padrão conseguiu-se as médias de níveis RGB mostrados na tabela 5.1 para as cores disponíveis (rosa, verde, amarelo e azul). Nessa etapa a rotina de calibração de cores foi utilizada, e as referidas médias são fruto das informações estraídas de cada pixel e dos nivéis de RGB semelhantes que compõem a região de mesma cor. O

algoritmo verifica ao redor de cada pixel se este pertence a uma região maior de pixels com níveis de RGB semelhantes.

	R	G	B
Amarelo	250	228	105
Rosa	218	128	102
Verde	105	150	82

Tabela 5.1: Tabela RGB

Para verificar se esses níveis de RGB são suficientes para identificar a marca na imagem, o algoritmo de flooffill é aplicado para que seja realizada uma busca por cada uma das médias encontradas. Os resultados são exemplificados nas figuras 5.2, 5.3 e 5.4.

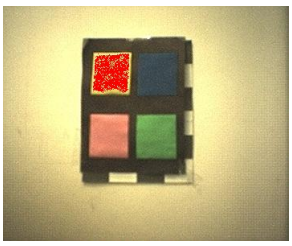


Figura 5.2: Busca Amarelo

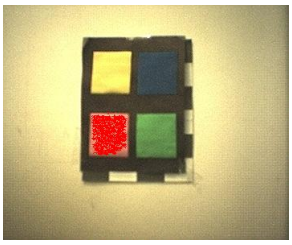


Figura 5.3: Busca Rosa

Com certeza foi notada a falta da cor azul. Isso deve-se ao fato de que para a cena montada não foi possível chegar a uma média que identificasse unicamente a marca da cor azul na imagem, pois a mesma acabava se confundindo com o preto existente. Dessa maneira, as cores utilizadas foram o rosa, o amarelo e o verde.

Quanto a parte radiométrica, ainda não se conseguiu chegar a resultados satisfatórios, de maneira que esse quesito estará entre as prioridades do desenvolvimento futuro para o sistema.

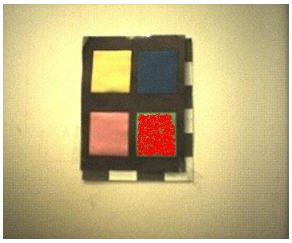


Figura 5.4: Busca Verde

Como já citado, a calibração para estimar-se os parâmetros intrínsecos da câmera apresentou melhores resultados quando feita utilizando-se uma rotina de calibração em matlab.

Nessa calibração foi utilizada uma sequência de 16 imagens vista na figura 5.5

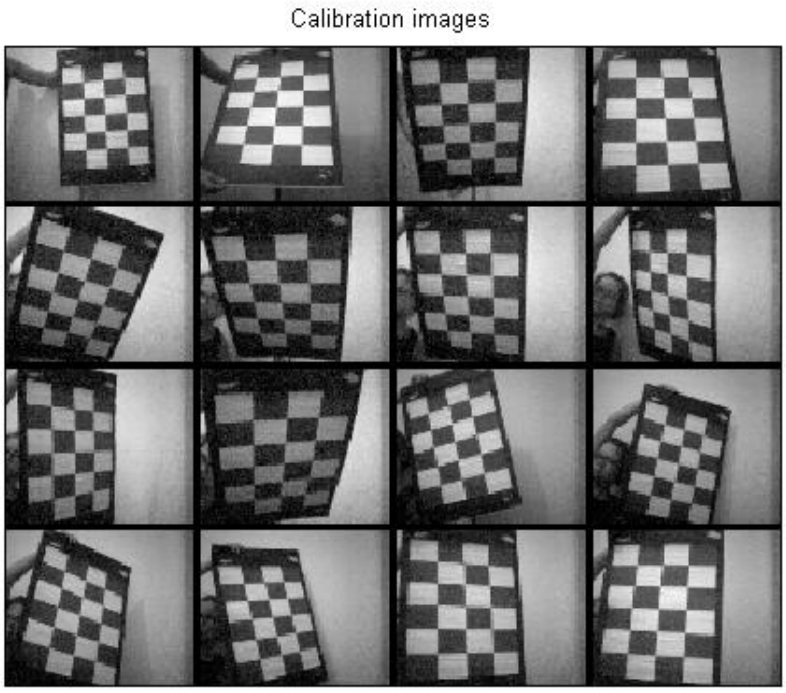


Figura 5.5: Vista das 16 Imagens Utilizadas

A partir dessas imagens conseguiu-se estimar os parâmetros intrínsecos da câmera utilizada. Tem-se a seguir, respectivamente, a matriz 3x3 que especifica tais parâmetros e outra 1x4 com os coeficientes de distorção encontrados.

- Matriz de Parâmetros Intrínsecos

$$\begin{bmatrix} 415.5375 & 0 & 132.5288 \\ 0 & 463.6427 & 121.5722 \\ 0 & 0 & 1 \end{bmatrix}$$

- Matriz de Coeficientes de Distorção

$$\begin{bmatrix} -0.136679319244274 & -0.691265870368033 & -0.025217019259358 & -0.00953064298001 \end{bmatrix}$$

Para a calibração de parâmetros extrínsecos (rotina em OpenCV), os padrões de imagem mostrados na figura 5.6 foram utilizados.

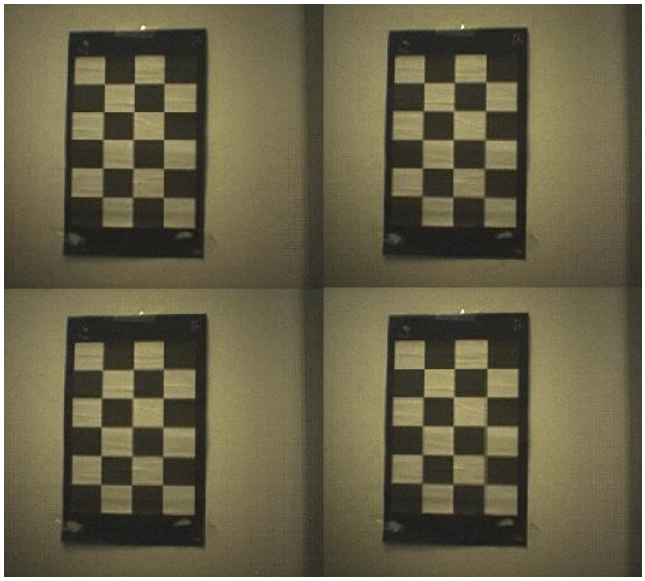


Figura 5.6: Exemplo de Imagens Utilizadas

Nessa etapa, como explanado no capítulo anterior, três funções principais atuam sobre as imagens. As duas primeiras buscam no padrão presente na imagem os pontos de mundo especificados. A última calcula os parâmetros propriamente ditos. Vale salientar que essas funções do OpenCV trabalham especificamente com esse tipo de padrão de calibração, ou seja, xadrez em branco e preto.

Na tabela 5.2 as coordenadas no mundo dos padrões mostrados na figura 5.6 são mostrados. Essas coordenadas devem constar nos arquivos .txt necessários a inicialização do sistema. Esses pontos devem ser medidos na cena real pelo usuário.

Na tabela 5.3 tem-se a saída da primeira função na primeira passagem do algoritmo quando são calculadas as coordenadas correspondentes na imagem dos pontos no mundo

Ponto	Coord. X	Coord. Y
0	54.29	55.50
1	43.79	55.50
2	33.29	55.50
3	54.29	47.50
4	43.79	47.50
5	33.29	47.50
6	54.29	39.50
7	43.79	39.50
8	33.29	39.50
9	54.29	31.50
10	43.79	31.50
11	33.29	31.50
12	54.29	23.50
13	43.79	23.50
14	33.29	23.50

Tabela 5.2: Pontos no Mundo

mostrados na tabela 5.2.

A segunda função busca melhorar as coordenadas dos pontos na imagem encontrados na tabela 5.3. Isso é feito realizando-se uma busca local ao redor de cada ponto anteriormente encontrado. Em 5.4 tem-se essas coordenadas otimizadas e recalculadas com um grau maior de certeza.

Nas tabelas 5.5 e 5.6 os passos anteriores são refeitos, na segunda passagem do algoritmo, com a diferença de que as imagens utilizadas agora, foram corrigidas com os coeficientes de distorção encontrados na etapa de calibração intrínseca. A tabela 5.5 mostra a saída da primeira função da etapa extrínseca e a tabela 5.6 mostra a saída da segunda, na segunda passagem do algoritmo.

É apresentada agora a saída da função que calcula os parâmetros da calibração extrínseca. Como já é sabido, essa saída consiste de uma matriz de rotação e de um vetor de translação.

A matriz de rotação resultante da primeira passagem do algoritmo, ou seja, sem as imagens corrigidas com os coeficientes de distorção encontrados na etapa intrínseca, pode ser vista na tabela 5.7 e a matriz otimizada, fruto das imagens corrigidas, na tabela 5.8, e que será utilizada na parte online.

Na tabela 5.9 tem-se o vetor de translação calculado a partir das imagens sem correção, e na tabela 5.10 é mostrado o vetor de translação calculado depois da correção das imagens, e o que será utilizado pelo sistema na parte online.

De posse do vetor de translação e da matriz de rotação é possível agora ligar tanto um

Saída da Primeira Função		
Ponto	Coord. X	Coord. Y
0	199.50	188.50
1	164.50	188.50
2	130.50	188.50
3	199.50	159.00
4	164.50	159.00
5	131.00	159.50
6	199.50	129.50
7	165.00	130.00
8	130.50	130.50
9	199.50	100.50
10	164.50	101.00
11	131.50	102.50
12	199.00	70.50
13	165.00	72.50
14	131.00	74.00

Tabela 5.3: Pontos na Imagem - Sem Correção

ponto no mundo a seu correspondente na imagem quando vice-versa. Utilizando as figuras 5.2, 5.3 e 5.4 como exemplo, o centro das marcas de cor amarela, rosa e azul podem ser calculados. Usando essas imagens como entrada da parte do algoritmo responsável por calcular a posição no mundo de seu respectivo ponto imagem, tem-se o resultado mostrado na tabela 5.11. Todas as medidas estão representadas em centímetros (cm).

A orientação é então calculada com esses dados recuperados da imagem.

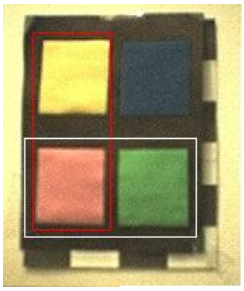


Figura 5.7: Exemplo de Localização

Para efeito de teste vamos utilizar a figura 5.7. Nela tem-se um exemplo padrão destacado em vermelho, orientado na vertical, e um outro destacado em branco, orientado na horizontal. Considerando-se os dois padrões como possível orientações, caso os mesmos estivessem posicionados em cima de um robô móvel, e passando-se os padrões pelo al-

Saída da Segunda Função		
Ponto	Coord. X	Coord. Y
0	200.57	189.43
1	165.87	189.71
2	131.53	189.40
3	200.55	159.98
4	165.76	160.16
5	131.80	160.33
6	200.56	130.18
7	165.87	131.16
8	131.86	131.73
9	200.28	101.36
10	165.68	102.15
11	131.97	102.99
12	200.03	72.07
13	165.77	73.44
14	132.34	75.06

Tabela 5.4: Pontos na Imagem - Sem Correção

goritmo de posicionamento e orientação, é retornado pelo mesmo os resultados da tabela 5.12.

Saída da Primeira Função		
Ponto	Coord. X	Coord. Y
0	200.00	188.50
1	165.50	188.50
2	131.50	188.50
3	199.00	160.00
4	165.00	159.50
5	131.50	160.00
6	200.00	130.00
7	166.00	130.50
8	131.50	131.50
9	199.50	102.00
10	165.50	102.00
11	132.00	103.50
12	199.50	72.50
13	165.50	73.50
14	132.00	75.00

Tabela 5.5: Pontos na Imagem - Com Correção

Saída da Segunda Função		
Ponto	Coord. X	Coord. Y
0	199.09	187.83
1	165.46	188.75
2	131.49	188.66
3	199.58	159.29
4	165.54	159.84
5	131.78	160.19
6	199.77	129.95
7	165.75	131.08
8	131.86	131.73
9	199.53	101.39
10	165.62	102.14
11	131.96	102.98
12	199.18	72.46
13	165.64	73.52
14	132.32	75.03

Tabela 5.6: Pontos na Imagem - Com Correção

Matriz de Rotação		
0.9999	0.0081	-0.0011
-0.0081	0.9993	-0.0361
0.0008	0.0361	0.9993

Tabela 5.7: Matriz de Rotação - Sem Correção

Matriz de Rotação		
0.9965	0.0066	-0.0827
-0.014342	0.9955	-0.0929
0.081749	0.0938	0.9922

Tabela 5.8: Matriz de Rotação - Com Correção

Vetor de Translação		
-33.8396	-36.4552	127.5624

Tabela 5.9: Vetor de Translação - Sem Correção

Vetor de Translação		
-33.6546	-36.0460	122.3237

Tabela 5.10: Vetor de Translação - Com Correção

Centros	Ponto Imagem X	Ponto Imagem Y
Amarelo	134.60	82.91
Rosa	131.48	159.73
Verde	190.12	162.70
Centros	Ponto Real X	Ponto Real Y
Amarelo	34.5	26.0
Rosa	34.3	48.5
Verde	52.8	48.5
Centros	Calculado pelo Sistema X	Calculado pelo Sistema Y
Amarelo	34.23	26.01
Rosa	33.13	47.38
Verde	51.68	48.63
Centros	Erro X	Erro Y
Amarelo	0.27	0.01
Rosa	1.17	1.12
Verde	1.12	-0.13

Tabela 5.11: Teste do Algoritmo - Cálculo de Posição - Figs 5.1, 5.2 e 5.3

Exemplo	Resultado
Padrão destacado em vermelho - Orientação :	92,96 graus
Padrão destacado em branco - Orientação :	3,84 graus

Tabela 5.12: Teste do Algoritmo - Cálculo de Orientação - Fig 5.6

Capítulo 6

Conclusão

A proposta aqui apresentada propõe a aplicação de um método conhecido como visão global [Kay 1993] como uma ferramenta dentro de um sistema de localização de robôs móveis dentro de ambientes fechados. Tal proposta visa aplicar-se ajudando na melhoria de localização bem como aliviando a capacidade de processamento que algumas plataformas robóticas tem de ter para empreender a tarefa de localizar-se, principalmente dentro de ambientes fechados.

O sistema aqui proposto tem o objetivo de tornar-se uma aplicação genérica de processamento e localização através de imagens e aplicar-se dentro de projetos diversos introduzindo nos mesmos melhorias de posicionamento ao leva-los a funcionalidade de posicionamento, abrindo caminho assim para que se tenha um controle maior sobre o movimento do robô ao longo do ambiente.

O método de visão global faz uso de várias câmeras colocadas de maneira a monitorar o ambiente com cada câmera sendo responsável por cobrir determinada área. Como consiste num método de localização absoluto, essa abordagem está sujeita a uma menor taxa de erro ao longo do tempo, já que no processo de localização apenas dados atuais são utilizados.

Um padrão de localização escolhido de acordo com as condições do local e constituído por duas cores diferentes é então colocado em cima do robô. A função principal do sistema aqui proposto é então utilizar essa marca como uma forma de localizar esse robô através das imagens obtidas com o sistema de câmeras.

Ao longo do trabalho foi apresentado o embasamento teórico e trabalhos nos quais a proposta aqui apresentada se baseia. O problema a ser resolvido foi proposto, bem como um algoritmo para resolve-lo, e uma aplicação com base na linguagem C++ foi apresentada.

O trabalho ainda não está completo pois sempre haverá muitas formas de fazer melhorias. A proposta aqui apresentada pode ser melhorada com a incorporação de técnicas

de robótica probabilística e de calibração mais apuradas, bem como estratégias que diminuam possíveis erros. Uma interface gráfica para o sistema, amigável e auto-explicativa também é uuma boa opção. Outras habilidades como detecção de obstáculos e planejamento de caminhos podem constar perfeitamente como futuros objetivos a serem alcançados, pois um sistema baseado em visão global pode ser usando também para monitorar o ambiente em busca de potenciais obstáculos cujas posições seriam passadas a plataforma robótica móvel, a qual usaria essas informações dentro do processo de planejamento de caminhos.

Referências Bibliográficas

Adorni, G.; Cagnoni, S.; Enderle S. (2001), Vision-based localization for mobile robots, *em* 'Robotics and Autonomous Systems, Volume 36', p. 103119.

Aquino, A. T.; Araújo, A. L. C. (2010), Aplicação do Filtro de Kalman a um sistema de Posicionamento de Veículo Aquático, *em* 'V CONNEPI'.

Ballard, D. H. (1982), *Computer Vision*, Prentice-Hall.

Baltes, J. ; Anderson, J. (2007), Intelligent Global Vision for Teams of Mobile Robots, *em* 'Proceedings of Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada'.

Becker, C. ; Salas, J. ; Tokusei K. ; Latornbe J. C. (1995), Reliable Navigation Using Landmarks, *em* 'Proceedings of 1995 IEEE International Conference on IEEE lnternatlonal Conference on Robotics and Automation', pp. 401–406.

Borenstein, J. ; Koren, Y. (1989), Real-Time Obstacle Avoidance for Fast Mobile Robots, *em* 'IEEE Transactions on Systems, Man and Cybernetics', pp. 1179–1187.

Borenstein, J.; Everett, H. R.; Feng L. (1996), *Where am I? Sensors and Methods for Mobile Robot Positioning*, University of Michigan.

Borenstein, J.; Everett, H. R.; Feng L.; Wehe D. (1997), Mobile Robot Positioning: Sensors and Techniques, *em* 'Invited paper for the Journal of Robotic Systems, Special Issue on Mobile Robots', pp. 231 – 249.

Caltech (n.d.), 'www.vision.caltech.edu/bouguetj/calibdoc'.

Chen, H.; Sun, D.; Yang J. (2009), Global localization of multirobot formations using ceiling vision SLAM strategy, *em* 'ELSEVIER Mechatronics', p. 617628.

Chen, M.H. ; Gu, D. ; Fu Y.T.; Pi C.H. (2012), On the Development of Autonomously Manipulation of Group Mobile Robots for Smart Living and Biomimetic Applications, *em* 'Proceedings of 2012 IEEE International Conference on Computer Science and Automation Engineering', pp. 259–263.

DeSouza, G. N.; Kak, A. C. (2002), Vision for Mobile Robot Navigation: A Survey, *em* 'IEEE Transactions On Pattern Analysis And Machine Intelligence', pp. 237–267.

Dev, A.; Krose, B.; Groen F. (1997), Navigation of a Mobile Robot on the temporal Development of the Optic Flow, *em* 'Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 558–563.

dos Santos, M. C. (2012), Revisão de Conceitos em Projeção, Homografia, Calibração de Câmera, Geometria Epipolar, Mapas de Profundidade e Varredura de Planos, *em* 'UNICAMP Universidade Estadual de Campinas, FT Faculdade de Tecnologia, Campinas'.

Fiala, M. (2004), Vision guided control of multiple robots, *em* 'Proceeding of the first Canadian conference on computer and robot vision', pp. 17–19.

Hartley, R. ; Zisserman, A. (2004), *Multiple View Geometry in Computer Vision*, Cambridge University Press.

Kay, M. G. ; Luo, R. C. (1993), Global Vision for the control of Free-Ranging AGV Systems, *em* 'Proceedings of 1993 IEEE International Conference on Robotics and Automation', pp. 14 – 19.

Michel, P.; Chestnutt, J.; Kuffner J.; Kanade T. (2005), Vision-Guided Humanoid Footstep Planning for Dynamic Environments, *em* 'Proceedings of 2005 5th IEEE-RAS International Conference on Humanoid Robots', pp. 13–18.

Milano, D. ; Honorato, L. B. (2010), VISÃO COMPUTACIONAL, *em* 'in UNICAMP Universidade Estadual de Campinas, FT Faculdade de Tecnologia, Campinas', pp. 609–620.

Moravec, H.P.; Elfes, A. (1985), High Resolution Maps from Wide Angle Sonar, *em* 'Proceedings of IEEE International Conference on Robotics and Automation', pp. 116–121.

Nadarajah, S.; Sundaraj, K. (2013), Vision in Robot Soccer: A Review, *em* 'Springer Science and Business Media Dordrecht'.

Orqueda, O. A. A. ; Fierro, R. (2007), Visual Tracking of Mobile Robots in Formation, *em* 'Proceedings of the 2007 American Control Conference', pp. 5940 – 5945.

Pinto, A. M. G.; Moreira, P.; Costa P. G. (2012), Indoor Localization System based on Artificial Landmarks and Monocular Vision, *em* 'TELKOMNIKA (Telecommunication Computing Electronics and Control)'.

Reina, A.; Gambardella, L. ; Dorigo M.; Di Caro G. A. (2012), Zeppelin: Ceiling camera networks for the distributed path planning of ground robots, *em* 'Proceedings of IRIDIA Technical Report Series'.

RoboCup (n.d.), ' www.robocup.org'.

Roque, W. L.; Iacute, D.; Doering S. (2005), Trajectory planning for lab robots based on global vision and Voronoi roadmaps, *em* 'Proceedings of Robotica Cambridge University Press', p. 467477.

Rosenfeld, A. (2000), *Image analysis and computer vision: 1999 [survey].*

Scaramuzza, D. ; Fraundorfer, F. (2011), Visual Odometry Part I : The First 30 Years and Fundamentals, *em* 'IEEE Robotics and Automation Magazine'.

Se, S. ; Lowe, D. G. ; Little J. J. (2005), Vision-Based Global Localization and Mapping for Mobile Robots, *em* 'IEEE TRansactions on Robotics', pp. 364–375.

Trucco, E. ; Verri, A. (1998), *Introductory Techniques for 3-D Computer Vision*, Prentice-Hall.

Wang, H. ; Yuan, K. ; Zou W. ; Zhou Q. (2005), Visual Odometry based on localy planar ground Assumption, *em* 'Proceedings of 2005 IEEE International Conference on Information Acquisition'.

Yan, C.; Zhan, Q. (2010), Multiple Mobile Robots Real-Time Visual Search Algorithm, *em* 'International Conference On Image Processing and Pattern Recognition in Industrial Engineering', pp. 78200U–78200U.

Yoon, K.J. ; Kweon, I. S. (2002), Landmark design and real-time landmark tracking for mobile robot localization, *em* 'Proceedings of SPIE - The International Society for Optical Engineering', p. 4573.