CÓDIGOS DE BARRAS BIDIMENSIONAIS APLICADOS À ROBÓTICA MÓVEL COM VISÃO

Marlon F. de Alcantara[1], Marcelo da Silva Hounsell[1], Alexandre Gonçalves Silva[1]

[1]*Laboratory for Research on Visual Applications* – Universidade do Estado de Santa Catarina (UDESC)
CEP – 89219-710 – Joinville – SC – Brasil

marlonmfa@gmail.com, {marcelo, alexandre}@joinville.udesc.br

Resumo. *O controle de robôs móveis depende da aquisição de informações de posição e orientação do robô em relação ao seu um ambiente. Para isto, é possível a utilização de marcadores que possuem limitações quanto à quantidade de dados que podem ser anexados. Pensando nisto, foi elaborado um novo marcador baseado em códigos de barras bidimensionais, que possuem uma capacidade de dados maior que a de marcadores comuns. Para validação foi desenvolvida uma estratégia de controle com o novo marcador para resolver o problema do Rato-Queijo. Como resultado, utilizando um robô móvel caseiro em malha aberta, o novo marcador se mostrou capaz de guiar as decisões do robô e evitando obstáculos e chegar ao seu destino.*

1. Introdução

A principal característica da robótica móvel é a locomoção que possibilita diversas aplicações, como por exemplo, exploração de ambientes ou transporte de material [COX; WILFONG, 1990]. Algumas aplicações na robótica móvel necessitam obter a posição e orientação do robô no ambiente. Para tal, diversos dispositivos são desenvolvidos a fim de obter os parâmetros necessários para o robô se mover no ambiente e decidir sobre a operação que deve ser feita ou qual trajetória deve ser percorrida. Podem ser empregados sensores de posicionamento baseados em Rádio Frequência, Infravermelho, Ultrassom, Magnéticos, dentre outros. Nesta área, a Visão Computacional (VC) pode ser empregada a fim de obter a própria posição. Sendo assim, podem ser usadas imagens padronizadas no ambiente, múltiplas câmeras, ou outra técnica que possibilite efetuar o cálculo da posição do robô.

Os marcadores convencionais utilizados na maior parte das aplicações possuem uma capacidade restrita de armazenamento de informações, apenas o suficiente para distinguir uma pequena quantidade de marcadores, o que na robótica móvel não seria suficiente para representar todas as possíveis posições de um objeto.

Os códigos de barras bidimensionais possuem uma capacidade superior de armazenamento de informações interna no código, porém não são comumente empregados neste tipo de aplicação. Fazendo o uso de códigos de barras bidimensionais em substituição aos convencionais marcadores amplia-se a gama de aplicações.

Foi observado na literatura outras técnicas utilizadas para se obter posição e orientação, e em geral fazem uso de pontos específicos no ambiente onde, de alguma forma, já se sabe a sua localização, ou por meio de objetos e marcadores de dimensões conhecidas. A posição e orientação do robô podem estar em relação ao ambiente como um todo, ou em relação a um objeto alvo.

No trabalho de Huh et al. (2006) é explorado o uso de marcadores, colocando-os sobre o chão e efetuando os ajustes de posicionamento por meio do retorno da leitura dos marcadores, e limitando a forma de disposição destes criando uma regra para posicionamento. Já as dimensões reais de objetos da cena podem ser um artifício a ser utilizado para estabelecer a posição e orientação, conforme visto em [OHYA; KOSAKA; KAK, 1998], onde o paralelismo das paredes podem estabelecer uma relação de distância e orientação.

Na aplicação de Asada et al. (1995) é utilizado o conhecimento sobre a forma como são vistos os elementos da cena para estimar a posição em que o robô se encontra, permitindo decidir quanto a direção que o robô deve tomar para ser possível colocar a uma bola no gol em um jogo de futebol de robôs, porém nesta aplicação é explorada a simplicidade dos elementos envolvidos e o conhecimento de posicionamento é o mínimo para aquele objetivo específico.

Câmeras posicionadas sobre o ambiente também foram utilizadas como tentativa no posicionamento de robôs como explorado em [MORIOKA; LEE; HASHIMOTO, 2002], onde o robô sabe sua própria posição e também da posição de outros objetos da cena devido aos dados de quatro câmeras dispostas sobre o ambiente.

Porém, por vezes, na Robótica Móvel se faz necessária a elaboração de uma estratégia de controle que faça uso das especificações de coordenadas e outras informações acerca do ambiente que possam ser fornecidas pelo marcador. Portanto, a identificação de posição e orientação em códigos de barras bidimensionais pode trazer maior aplicabilidade e flexibilidade a robôs móveis, haja vista, que a estratégia de controle pode considerar tanto a informação interna contida no código, quanto à posição relativa entre robô e o código.

Diante disso, o presente trabalho apresenta a criação de um marcador enriquecido e como este pode ser utilizado no auxílio a uma estratégia de controle de um robô móvel.

Na Seção 2, são apresentados os conceitos gerais sobre os diversos tipos de códigos de barras e, em seguida será feita uma comparação e seleção daquele mais apropriado para aplicação em robótica móvel. Na Seção 3 é detalhado o desenvolvimento do reconhecimento de dados, posição e orientação no marcador. Na seção 4 é apresentado o robô e a aplicação teste e comentam-se os sistemas de posicionamento local para então, na Seção 5, mostrar os testes e resultados obtidos. A Seção 6 conclui este texto.

2. Códigos de Barras

A codificação em barras permite que se obtenham computacionalmente informações de forma automática mediante a submissão do código a um instrumento de leitura. O uso de um código de barras propicia a melhora na eficiência operacional, a redução do tempo para aquisição e armazenamento dos dados, a redução de erros de leitura, a redução de custos pelo uso de sistemas de leitura convencionais, e outros benefícios ao usuário como a flexibilidade e padronização [SOARES, 2001]. Um código de barras pode ser unidimensional ou bidimensional. Os códigos bidimensionais possuem elementos que auxiliam uma leitura por Visão Computacional (VC) por meio de um processamento gráfico [OHBUCHI; HANAIZUMI; HOCK, 2004] e são mais indicados

a um contexto de marcadores. Os códigos bidimensionais mais comuns são o Data Matrix e o QR Code.

2.1 Data Matrix

Data Matrix (2012) é um código de barras bidimensional, visto na Figura 1, padronizado pelo ISO/IEC 16022:2000 que é uma forma de matriz binária onde cada célula representa um bit codificado. O código pode estar disposto em formato quadrado ou retangular possuindo dessa forma uma representação em duas dimensões, possibilitando uma representação maior de dados do que em uma dimensão apenas, chegando a uma capacidade de representação de até 1556 bytes. O formato do Data Matrix pode assumir qualquer valor entre 8x8 e 144x144 células de igual tamanho, e possui capacidade de correção de erros de até 30%.

Figura 1. Exemplo de um código Data Matrix

A estrutura do código possui duas linhas contínuas que são utilizadas para situar a posição do código e duas linhas alternadas que situam a posição das linhas e colunas. As quatro linhas juntas delimitam o todo, e na parte interior do marcador ficam codificados os dados [STEVENSON, 2005].

2.2 QR Code

O QR Code [DENSOWAVE, 2012] é uma representação binária de dados em um código de barras bidimensional desenvolvida pela DENSOWAVE no ano de 2000 e de especificação aberta, registrada como ISO/IEC 18004:2006. A representação bidimensional, assim como no Data Matrix, possibilita representar uma quantidade maior do que a representação em uma dimensão apenas.

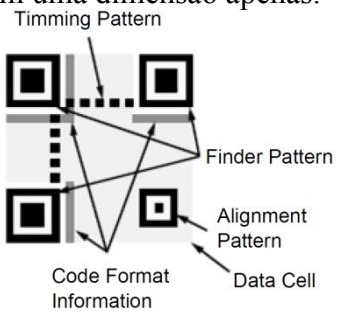

Figura 2. Elementos do QR Code

O QR Code dispõe de três elementos na imagem (marcadores) inseridos no código que objetivam situar o processo de leitura da informação, a disposição dos elementos pode ser vista na Figura 2 [OHBUCHI; HANAIZUMI; HOCK, 2004]:

- Os '*finder patterns*' são três marcadores usados para facilitar a busca pelos cantos do código, e com isso estimar o quarto canto;

- O '*timming pattern*' é utilizado para sincronizar a leitura com cada célula específica em relação a horizontal e vertical;
- Por fim, o '*alignment pattern*' é um marcador utilizado para ser possível tratar eventuais distorções quando o número de bits no código é elevado. Quanto maior o número de bits armazenados, maior será o número desses marcadores.

Anexo aos marcadores de posição ('*finder pattern*') é colocada à informação referente ao formato do código ('*Code Format Information*') para estabelecer o padrão ao qual será feita a leitura. Toda a área não ocupada pelos elementos anteriores é utilizada para o armazenamento de dados. A leitura é de forma binária, onde uma célula preta representa um bit '0' e a célula branca o bit '1'.

2.3 Comparativo e Escolha

Para realizar a escolha por um padrão de código de barras de onde pudessem ser obtidos posição, orientação e dados, um comparativo foi realizado entre os padrões de códigos de barras bidimensionais, que são mais indicados a um sistema de leitura por visão.

A capacidade de tolerância de erros contidas nos padrões QR Code e Data Matrix são equivalentes podendo ter até 30% de bits perdidos sem perda de informação. O código QR Code possui capacidade de armazenamento superior a do Data Matrix aos outros padrões de codificação (2953 bytes contra 2335bytes). Os marcadores para os vértices são outro fator importante, haja vista que estes são referências internas que facilitam a identificação dos cantos necessários para estimar a posição e orientação do marcador em relação à câmera.

Por fim, foi feita a escolha pelo QR Code para ser usado de base para elaboração de um marcador próprio, devido a: grande capacidade de armazenamento de informação (2953 bytes); possuir referências internas para os vértices, o que facilita a identificação dos cantos por VC; possuir tolerância a erros de leitura; por possuir o formato quadrado, o que facilita os cálculos de posição e orientação.

3. Detecção da Posição e Orientação no Espaço

As técnicas utilizadas para se obter a posição e a orientação comumente fazem uso de pontos específicos no ambiente onde, de alguma forma, já se sabe a sua localização; ou por meio de objetos e marcadores de dimensões conhecidas. A posição e orientação do robô podem ser em relação ao ambiente como um todo, ou em relação a um objeto alvo.

No trabalho de Huh et al. (2006) é explorado o uso de marcadores, colocando-os sobreo chão e efetuando os ajustes de posicionamento por meio do retorno da leitura dos marcadores, eliminando a forma de disposição destes criando uma regra para posicionamento. As dimensões reais de objetos podem ser utilizadas para estabelecer a posição e orientação, conforme visto em [OHYA; KOSAKA; KAK, 1998], onde o paralelismo das paredes pode estabelecer uma relação de distância e de orientação.

Na aplicação de Asada et al. (2002), é utilizado o conhecimento sobre a simplicidade dos elementos envolvidos, que no caso são dois segmentos verticais e um segmento horizontal que compõe uma meta, e uma esfera que compõe um objeto a ser posto na meta. O cálculo do posicionamento propicia a informação mínima para que o robô possa saber qual tarefa deve ser executada. O supervisionamento por câmeras posicionadas sobre o ambiente é utilizado para posicionamento no trabalho Morioka,

Lee e Hashimoto (2002), onde utilizando os dados quatro câmeras é possível estimar as distâncias entre os elementos da cena.

A fim de se obter a posição e orientação dos marcadores em relação à câmera, é necessário calcular os parâmetros que produzem a imagem projetada de um quadrado. Para isto, a DLT (*Direct Linear Transformation*) [HARTLEY; ZISSERMAN, 2003] pode ser aplicada, uma alternativa seria o cálculo por meio das matrizes de transformação assim como é feito no ARToolKit (2012), ARTag (2012) e OpenCV [DENSOWAVE, 2012]; porém, com a DLT se consegue uma independência dos parâmetros obtidos, e pode ser convertido para uma matriz de transformação caso necessário. O trabalho de [ALCANTARA; HOUNSELL; SILVA, 2011] detalha como a DLT pode ser usada para aquisição de posição, orientação em marcadores.

3.1. Integração PO/Dados

As funcionalidades a serem integradas são a capacidade de armazenamento de dados dos códigos de barras no padrão QR Code e a possibilidade de verificar a posição e orientação (PO) da câmera em relação ao marcador, podendo obter três parâmetros de saída: a posição relativa, a orientação e, os dados contidos no código de barras. Esta tecnologia foi nomeada QRPO (*Quick Response barcode with Position and Orientation*).

O QRPO foi desenvolvido fazendo uso das funcionalidades da biblioteca OpenCV 2.3 (2012), desde as funções de abertura da câmera e captura dos frames, até funções de processamento de imagens e exibição do vídeo capturado. A decodificação fez uso dos códigos da 'libdecodeqr' [NISHI, 2012], criado como uma alternativa em código aberto para o desenvolvimento de aplicações que façam uso da decodificação de códigos de barra do tipo QR Code, atendendo a licença LGPL (*Lesser General Public License*) e permitindo modificações e redistribuição dos códigos contidos na biblioteca. A libdecodeqr também faz uso da biblioteca OpenCV.

Figura 3. Exemplo do Marcador QRPO

Para aumentar a capacidade de identificação de PO foi anexada uma borda envolvendo o QRPO conforme pode ser visto na Figura 3, e o algoritmo dividiu-se então em duas partes: a identificação da borda, permitindo o cálculo da posição/orientação e; a decodificação da parte interna e, por consequência a aquisição dos dados. O sistema quando visualiza um QRPO retorna:

- **posição**: as coordenadas (*x;y;z*) de cada vértice do marcador, onde o *x* e o *y* representam respectivamente a coordenada horizontal e a coordenada vertical, a coordenada *z* representa a profundidade em que o vértice se encontra;

- **orientação**: dois ângulos que representam respectivamente a rotação da câmera em relação aos eixos *x* e *y*, lembrando que com apenas dois ângulos de rotação em torno de eixos ordenados é possível obter qualquer orientação no espaço;
- **dados**: se for possível a correta decodificação, é informada a informação contida internamente no marcador.

O algoritmo do QRPO aplica as seguintes etapas para cada *frame,* dado como entrada:

1. Converter de RGB para tons de cinza considerando os parâmetros da luminância (0.299R + 0.587G + 0.114B);
2. Limiarizar por janelamento de média com uma janela de tamanho 41×41 com o objetivo de evitar que eventuais mudanças de iluminação ao longo da imagem interfiram na limiarização [NIBLACK, 1986]. A limiarização por janelamento considera como limiar a média dos vizinhos contidos na janela, ou seja, com desvio padrão zero;
3. Aplicar o fechamento de buraco (*closing holes*) utilizando como sementes os pontos da borda da imagem, fazendo a operação AND do resultado com a imagem anterior; resultando na imagem comas regiões de ilhas (*holes*) destacadas [DOUGHERTY; LOTUFO, 2003];
4. Erodir (operação morfológica) a imagem para eliminação de ruídos e falsas zonas de borda [DOUGHERTY; LOTUFO, 2003]. Quanto maior a erosão menor a quantidade de ruídos e menor também a capacidade de identificação de um marcador conforme a distância que este se encontra da câmera, a erosão na aplicação está utilizando uma caixa de tamanho 3;
5. Varrer linearmente a imagem em busca da ilha com maior área, este passo faça com que reste na imagem apenas um componente conexo que em termos gerais significa o QRPO mais próximo da câmera;
6. A partir de uma semente da região que contém a maior ilha é feita a reconstrução com a imagem anterior a erosão para evitar a perda de informação [VICENT, 1993]; é utilizado como semente o primeiro ponto do componente encontrado na varredura;
7. É utilizada a detecção de borda com o algoritmo de Canny [CANNY, 1986], que tem por objetivo apenas diminuir o número de comparações com candidatos a cantos da imagem, haja vista que apenas um ponto da borda tem a chance de ser um canto do marcador; São assumidos como cantos os respectivos pontos: o mais a direita e mais acima, o mais a direita e mais abaixo, o mais a esquerda e mais abaixo, o mais a esquerda e mais acima. Em caso de ambiguidade é utilizado respectivamente: o maior x, o menor y, o menor x e o maior y;
8. O algoritmo calcula a DLT para os quatro cantos encontrados obtendo assim a posição/orientação;
9. Em seguida, caso seja possível, é executada a rotina do libdecodeqr para identificação dos dados internos;
10. Retornar a posição/orientação caso exista um QRPO na imagem e também os dados caso seja possível a correta leitura.

Após o recebimento do *frame* aplicam-se as etapas 1 a 4 que objetivam isolar as ilhas (parte clara envolvida por um contorno escuro) contidas na imagem. Após esta etapa, caso não exista nenhuma ilha encontrada, o retorno da função informa que não foi encontrado um QRPO; caso exista apenas uma ilha, esta é escolhida como área do QRPO, caso exista mais de uma ilha, é escolhida a ilha que possua a maior área e consequentemente esteja mais próxima da câmera (etapa 5), esta região é então reconstruída para obter a forma original do marcador (etapa 6). Então, na etapa 7, a área

escolhida é processada e os vértices da imagem são identificados. Com base nestes vértices, a etapa 8 é o cálculo da DLT e por fim, se possível, são decodificados os dados internos ao marcador na etapa 9, caso não seja possível esta etapa é pulada, e a etapa 10 por fim, retorna os dados que puderam ser obtidos.

Utilizando um marcador de 10 cm x 10 cm realizaram-se testes para relacionar a distância e o ângulo onde a câmera consegue identificar a PO, separadamente dos dados. A Figura 5 exibe os resultados.

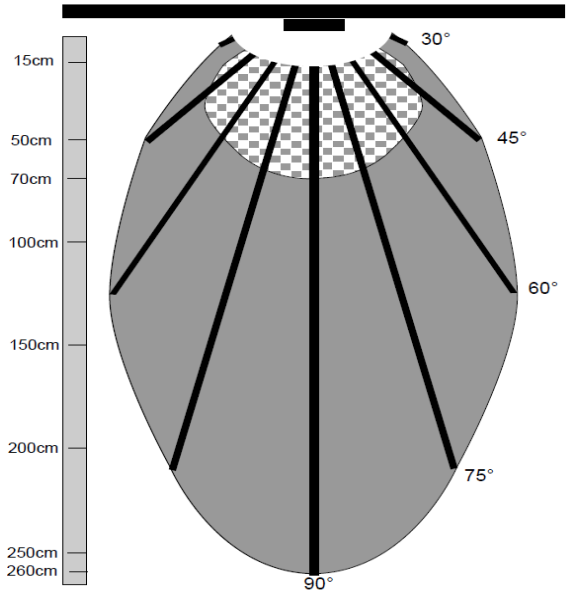

Figura 5. Alcance na Identificação de dados (grade) e posição/orientação (cor sólida) pelo QRPO.

Considerando um QRPO colocado no meio de uma parede, na parte de cima da figura, a área rachurada em xadrez identifica onde foi possível ler os dados corretamente e a área em cinza, onde foi possível identificar posição e orientação corretamente. Observe que a diferença entre a capacidade de identificação da posição/orientação (PO) e de dados é acentuada: a capacidade de identificação da PO foi de até 2,6 metros, todavia a capacidade de identificação dos dados foi de até 70cm, devido ao compartilhamento da medida de 10cm×10cm com os elementos que auxiliam a identificação da posição/orientação (borda preta e espaçamentoentre a borda e o QR Code).

4. Sistemas de Posicionamento Local

Uma forma alternativa de se obter a localização de robôs industriais é através de técnicas de Sistema de Posicionamento Local (LPS - *Local Position System*). O LPS se divide em duas categorias:

1. *Auto Posicionamento*: Um dispositivo móvel encontra sua própria localização instantânea de acordo com um ponto fixo com posição conhecida;
2. *Posicionamento Remoto*: Um dispositivo móvel encontra as posições instantâneas de outros objetos de acordo com a sua própria posição.

Tais sistemas são funcionais em qualquer ambiente *outdoor* ou *indoor*, e são tipicamente usados juntamente com centros de controle de comunicação para realizar

rastreamento, monitoramento, comando e controle. Entretanto, esses sistemas podem perder a sua localização permanentemente se houver um problema com o funcionamento dos sensores respectivo, mesmo que isto ocorra por um curto período de tempo [ZEKAVAT; TONG; TAN, 2004]. Nestes sistemas, qualquer pequeno erro de cálculo da localização pode acumular e se tornar um grande erro de posição ou levar a completa perda da informação de posição.

O escopo das técnicas de LPS citadas aqui está limitado a sistemas baseados em dispositivos que disponibilizam informação de posição absoluta. Os sistemas de posição atuais, segundo [AITENBICHLER; MUHLHAUSER, 2003], podem ser classificados pelo uso dos meios de transmissão a seguir:

- *Rádio Frequência*: oferecem uma precisão de 1 a 3 metros e não necessitam de uma linha direta de visão, todavia possui uma baixa precisão;
- *Ultrassom*: podendo localizar os alvos dentro de 9cm da sua posição real, porém sofre na precisão e interferências;
- *Magnético*: oferece alta resolução, mas é limitado a um ambiente pequeno e precisamente controlado;
- *Infra Vermelho*: É necessário um emissor e um receptor e não pode haver obstáculos entre eles, logo a capacidade depende da intensidade do feixe.

Este trabalho, diferentemente dos métodos convencionais, implementa um sistema com LPS, utilizando para isso apenas a VC e códigos de barra bidimensionais.

5. Montagem do Robô Móvel

Para teste, implementação e validação da proposta de controle, foi montado um robô utilizando uma placa Arduino (2012). O Arduino é uma plataforma de prototipagem *opensource* de hardware e software de fácil manipulação. O hardware é constituído de um microcontrolador AVR de 8 bits carregado com um *bootloader* com o papel de carregar e gravar o sistema sem necessidade de um gravador externo. Na placa acoplada consta um cristal para geração de *clock* de 16MHz, e um microcontrolador com suporte à USB (ATmega8U2) ou um conversor USB/serial (FT232RL) para comunicação do Arduino com o computador ou outros dispositivos com USB HOST já que o microcontrolador principal usa comunicação Serial UART.

A montagem contém duas rodas, uma esfera de apoio e um motor de corrente contínua para cada uma das rodas (não são motores de passo). Foi utilizada uma câmera FOSCAM IP FI8918W, com resolução 640x480 e qualidade de imagem similar as webcams convencionais. A câmera foi acoplada na posição central do robô, conforme pode ser visto na Figura 6.

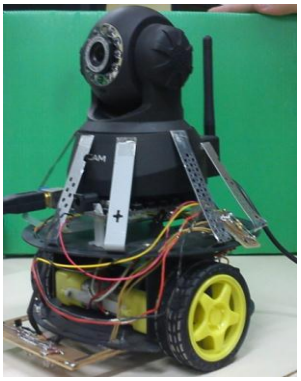

Figura 6. Montagem do robô para testes

Dessa forma, os cálculos de posição e orientação utilizam como princípio a posição da câmera como sendo a do próprio robô. A alimentação da câmera faz uso de uma fonte de 5V. A placa do Arduino é alimentada por uma bateria de 6V, e os atuadores (as duas rodas) estão conectados diretamente a esta e, portanto, suscetível a grandes variações de carga, alterando muito o seu nível de tensão. Isto impediu que a mesma fonte do Arduino fosse usada pela câmera.

5.1. Proposta de Controle

Um robô móvel obrigatoriamente necessita de uma estratégia de controle que o permita ter conhecimento do ponto onde ele se encontra e para qual direção deverá ser feito o seu deslocamento para realização da tarefa. A informação que o robô possui de sua posição em muitos casos não é verdadeira, pois podem ocorrer variações no deslocamento como por exemplo: desvios devido ao peso, patinagem das rodas, baixa rotação do motor devido a falta de carga na bateria, dentre outros.

Devido a isto um contínuo controle do posicionamento do robô se faz necessário, e a proposta deste trabalho é realizar esse controle contínuo por meio de visão utilizando como artifício para aquisição de posição e orientação, os dados contidos em QRPOs espalhados pelo ambiente. Neste caso, o QRPO está anexado aos obstáculos do ambiente e ao serem percebidos fornecem as dimensões do obstáculo para que este seja contornado e além disto, fornece a posição do obstáculo no ambiente; isto torna possível ao robô que faça uso da informação da posição do obstáculo, juntamente com a posição e orientação do robô em relação ao obstáculo, para calcular a sua própria posição e orientação num plano de coordenadas ortogonais (x, y). Desta forma, o robô sabe para qual direção ele deve prosseguir para encontrar o alvo.

Nestes termos foi constituída uma estratégia de controle. A estratégia de controle tem como foco manter sempre que possível o QRPO no campo visual do robô, de forma a evitar que o mesmo gere erros de movimentação que impossibilitem que este alcance o objetivo. O pseudocódigo está apresentado na listagem Algoritmo 1.

A ideia principal é manter o robô sempre com a referência de um QRPO ou indo na direção de um. Ao verificar que um QRPO não é um queijo, certamente este indica um obstáculo, logo a rotina Contornar Obstáculo faz uso dos parâmetros de largura e comprimento do obstáculo, que estão codificadas no QRPO. O robô sempre rotaciona na direção do queijo, a não ser que este já tenha sido encontrado.

Algoritmo1. Controle para aplicação Rato-Queijo

```
queijo← falso
Faça
Se QRPO visualizado:
posição← Leitura de Posição
orientação← Leitura de Orientação
dados← Leitura de Dados
  Enquanto Dados = NULO:
Movimenta robô na direção do QRPO
dados← Leitura de dados do QRPO
   Se dados = QUEIJO
Queijo ← verdadeiro
   Senão:
    Se Obstáculo na Direção do Queijo:
     Contornar Obstáculo
    Rotacionar na Direção do Queijo
    Movimentar na Direção do Queijo
 Senão:
Inicia rotação arbitrária para um dos lados buscando QRPOs
Enquanto queijo = falso
```

6. Testes e Resultados

Para validar o algoritmo de controle foram realizados três testes práticos:

1. O primeiro teste consiste em colocar no ambiente um robô e um queijo e verificar se o robô encontra o queijo e vai até ele se posicionando em frente ao robô;
2. No segundo teste, foi colocado um obstáculo que não obstrui a passagem do robô para verificar se o posicionamento do robô ao verificar um QRPO que informa a posição/orientação é correto;
3. Finalmente, no terceiro teste foram colocados o queijo e dois obstáculos, um dos quais se posicionava entre o robô e o queijo de forma que este devesse ser contornado a fim de se chegar ao queijo.

6.2. Cenário de teste

O QRPO pode ser utilizado como um sistema de posicionamento local, se cada marcador contiver a informação referente às suas coordenadas (no ambiente) embutida no código do QRPO, logo o robô pode procurar um objeto alvo no ambiente e reportar caso o encontre. Essa é uma aplicação conhecida como "Rato-Queijo" que pode também ser implementada com o uso de QRPOs, de forma que, se o robô (rato) é inserido no ambiente, sem nenhuma informação de posição, porém sabe as coordenadas de um objeto alvo (queijo), é possível o robô se localizar através dos QRPOs no ambiente, evitar obstáculos, e encontrar o queijo.

A única informação fornecida ao robô inicialmente são as coordenadas do queijo no plano. Não é fornecida ao robô a informação da sua posição/orientação inicial, nem quanto aos demais objetos que estão dispostos na cena.

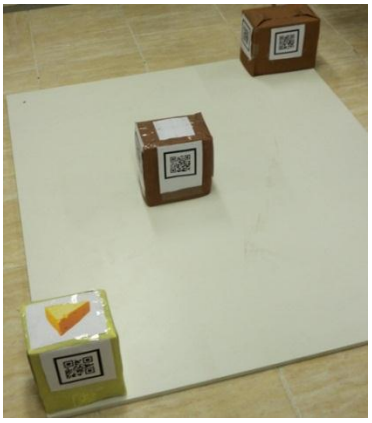

Figura 7. Montagem do robô para testes

No ambiente estão dispostos obstáculos retangulares que possuem um QRPO anexado a cada uma das quatro faces perpendiculares ao plano. O QRPO contém a informação da posição do obstáculo no plano, e para qual ponto cardeal está voltada a face que está sendo visualizada. Também são informadas no próprio QRPO as dimensões (largura e comprimento) da caixa para que o "rato" possa utilizar a informação para desviar do obstáculo caso seja necessário. Veja na Figura 7 duas caixas representando os obstáculos e representando o queijo (caixa em primeiro plano na figura).

6.3. Ambiente somente com o robô e o queijo

Este teste verifica a principal tarefa do algoritmo: se posicionar em frente ao queijo caso o localize, a última etapa do sistema que busca se posicionar em frente ao QRPO. Por se tratar de uma etapa que ocorre sem que a câmera esteja visualizando o QRPO, exige precisão nos movimentos do robô, movimentos estes, que só podem ser corrigidos depois de recuperada a visão de um marcador.

Devido à necessidade do controle contínuo diante das severas limitações da arquitetura física do robô, ocorreu a repetição do último passo de posicionamento em algumas amostragens ou, mais raramente, aconteceu do robô perder completamente a visão do marcador, porém estas exceções estão tratadas no algoritmo de controle. Este teste mostrou a importância do controle contínuo da posição do robô, sendo necessários muitos ajustes de posição para corrigir os erros de movimentação.

6.4. Ambiente com robô, queijo e um obstáculo

Este teste ilustra as situações onde é necessário utilizar a informação do obstáculo para identificar a direção do queijo, e também as situações onde o obstáculo tem de ser contornado para se chegar ao objetivo. Foi ressaltado neste teste a importância dos sucessivos ajustes de posicionamento com base no *feedback* do QRPO. Este teste mostrou a importância da conferência com o marcador lateral ao ter de contornar um obstáculo, sem essa conferência e o devido ajuste, o acúmulo de erros comprometeria o desempenho da aplicação.

6.5. Ambiente com robô, queijo e dois obstáculos

Este teste simula a situação onde ocorre de um obstáculo informar ao robô que deve ser percorrida determinada direção, porém nesta direção encontra-se outro obstáculo a ser

contornado. A configuração de ambiente com mais obstáculos equivale às mesmas situações que ocorrem neste caso.

Neste teste, pode-se observar que o robô segue sempre na direção do queijo, haja vista que quando o robô se localiza através de um obstáculo e este o redireciona a outro obstáculo, a posição do queijo fará com que o robô (em situações normais), não volte a visitar o mesmo obstáculo. Isto comprova que o robô, dentro de um tempo finito, encontrará o queijo, e que a estratégia de controle, mesmo simples, é válida e efetiva.

7. Conclusão

O recente uso de marcadores em aplicações cotidianas, impulsionados pela tecnologia de Realidade Aumentada (RA), fez surgir um recurso útil para a área de robótica móvel. O uso de marcadores numa aplicação com robôs móveis permite que o robô obtenha informações que descrevem o seu ambiente, informações estas que podem mudar bastante. Os marcadores convencionais de RA têm um limitado volume de dados que podem ser tratados o que os inviabiliza para uso em robótica.

Este trabalho mostrou como um marcador bidimensional padronizado pode ser enriquecido e como este pode ser usado de forma prática numa aplicação onde um robô tem que reconhecer seu ambiente e chegar num destino especificado.

Apesar de ter sido usado um robô de hobbistas, o uso de marcadores com capacidade de reconhecimento de dados, posição e orientação, denominada QRPO, se mostrou suficiente para garantir que o robô executasse a tarefa com sucesso.

Agradecimentos

Os autores gostariam de agradecer à FAPESC pelo apoio financeiro na forma de bolsa de mestrado e ao professor Maurício Aronne Pillon por ceder a câmera e o robô para testes.

Referências

Aitenbichler, E.; Muhlhauser, M. (2003) An ir local positioning system for smart items and devices. In: Proceedings. 23rd International Conference on Distributed Computing Systems Workshops. [S.l.: s.n.]. p. 334–339.

Alcantara, M. F.; Hounsell, M. S.; SILVA, A. G. (2011) Alternative position, orientation and data recognition algorithms for augmented reality markers. In: International Association for Development of Information Society –Conference on Applied Computing. IADIS-AC. Rio de Janeiro. 557-561.

Arduino. (2012) Home Page - An open-source electronics prototyping platform based on flexible, easy-to-use hardware and software.Disponível em: <http://www.arduino.cc/>. Acesso em: 09/04/2012

Artag. (2012) ARTag Home Page. Disponível em: <http://www.artag.net/>. Acesso em: 09/04/2012.

Artoolkit. (2012) ARToolKit – Documentation.Disponível em: <http://www.hitl.washington.edu/artoolkit/documentation/index.html>. Acesso em: 09/04/2012.

Asada, M.; Noda, S.; Tawaratsumida, S.; Hosoda, K. (2002). Vision-Based Reinforcement Learning for Purposive Behavior Acquisition. Osaka, Japão.

Canny, J. F. (1986). A computational approach to edge detection. [S.l.]: IEEE Trans - Pattern Analysisand Machine Intelligence, 679–698 p.

Cox, I. J.; Wilfong, G. T. (1990) Autonomous Robot Vehicles. 1. ed. New York, US: Springer-Verlag.

Datamatrix. (2012) Data Matrix Web Site. Disponívelem: http://idautomation.com/datamatrixfaq.html>. Acesso em: 09/04/2012.

Densowave. (2012) QR Code.Disponível em: <http://www.denso-wave.com/qrcode/indexe.html>. Acesso em: 09/04/2012.

Hartley, R.; Zisserman, R. (2003) Multiple View Geometry in Computer Vision. Reino Unido: Cambridge Univertity Press, 88–131 p.

Huh, J.; Lee, K.; Chung, W. K.; Jeong, W. S.; Kim, K. K. (2006) Mobile robot exploration in indoor environment using topological structure with invisible barcode. In: International Conference on Intelligent Robots and Systems. Beijing, China: [s.n.].

Morioka, K.; Lee, J.; Hashimoto, H. (2002) Human Centered Robotics in Intelligent Space. Tokyo, Japão.

Niblack, W. (1986) An Introduction to Digital Image Processing. Prentice-Hall, Englewood Cliffs, NJ.

Nishi, T. (2012) libdecodeqr. Disponível em: <http://trac.koka-in.org/libdecodeqr>. Acesso em: 09/04/2012.

Ohbuchi, E.; Hanaizumi, H.; Hock, L. A. (2004) Barcode readers using the camera device in mobile phones. In: Proceedings of the International Conference on Cyberworlds. Washington, US: [s.n.],

Ohya, A.; Kosaka, A.; Kak, A. (1998) Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing. IEEE Transactions on Robotics and Automation, v. 14, n. 6, p. 969–977.

OpenCV Vision Library. (2012). Disponível em: <http://sourceforge.net/projects/opencvlibrary/>. Acesso em: 09/04/2012.

Soares, R. C. (2001) Estudo de Código de Barras por Análise de Imagens. Dissertação (Mestrado em Engenharia Elétrica) — Universidade Estadual de Campinas, Campinas, SP.

Stevenson, R. (2005) Laser marking matrix codes on pcbs. In: Printed Circuit Design e Manufacture. Seattle, US: [s.n.], p. 32–36.

Zekavat, S. A.; Tong, H.; Tan, J. (2004) A novel wireless local positioning system for airport (indoor) security. In: Proceedings of SPIE. [S.l.: