NAVEGAÇÃO DE ROBÔS MÓVEIS AUTÔNOMOS E DETECÇÃO DE HUMANOS BASEADA EM SENSOR LASER E CÂMERA TÉRMICA

Matheus Chung Nin, Fernando Osório

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

Resumo No que diz respeito à robótica, pode-se notar os inúmeros benefícios que ela pode trazer a sociedade. Dentre as variadas áreas de pesquisa vinculadas a Robótica Móvel Autônoma, há a Navegação Autônoma. Esta, por sua vez, tem o objetivo de dar ao robô a função de poder locomover-se em segurança e sem a interferência humana. Robôs que possuem essa tecnologia são denominados *Autonomous Robots*. Este projeto tem por objetivo implementar algoritmos capazes de, por meio de uma câmera térmica e de um laser do tipo LIDAR (*Light Detect and Ranging*), de-tectar um ser humano (invasor) e persegui-lo, evitando obstáculos.

Palavras Chaves: Câmera térmica, visão computacional, navegação autônoma, processamento de imagens, robótica autônoma, fusão de sensores.

Abstract: Intelligent robotic solutions can provide many benefits to our society. Mobile robots and autonomous navigation systems are examples of important areas of robotics research. Their main goal is to provide to robots a mobility capacity without any human intervention. Robots which have that ability are called Autonomous Robots. This project aims to implement algorithms for mobile robots that use as input informa-tion a thermal camera and a LIDAR laser, used in order to detect and pursuit a human (intruder), and also to detect and avoid obstacles.

Keywords: Thermal camera, computer vision, autonomous navigation, image processing, autonomous robots, sensors fusion.

1 INTRODUÇÃO

A navegação em Robôs Móveis Autônomos (RMA) tem sido grande fonte de atenção dos pesquisadores na área de robótica. Dentre as linhas de pesquisa ligadas à RMA, há a navegação autônoma. Esta permite que o robô possa capturar informações do ambiente e por meio de métodos algorítmicos, possa interpretá-la e então, tomar decisões que permitam a navegação do robô no local. Neste projeto, a navegação será feita utilizando fusão de sensores e técnicas de visão computacional (Gonzalez & Woods 2002).

Artigo Submetido para a Mostra Nacional de Robótica Agosto/2011

Dentre as vantagens que a Robótica Móvel Autônoma oferece, pode-se citar a possibilidade de usar um robô para fazer a vigilância de um ambiente, onde além de detectar se este está ou não sendo invadido, pode-se também perseguir o possível invasor. Este recurso é muito importante para a sociedade uma vez que não seria mais necessário encarregar seres humanos com trabalhos relacionados a vigilância que trariam risco a vida.

2 OBJETIVO

Com este projeto, espera-se que utilizando uma câmera de captura de imagens térmicas (sensor de calor) e um sensor laser do tipo LIDAR (Light Detect and Ranging), seja possível que um robô cumpra o papel de fazer a vigilância de ambientes fechados. Com isso, não será necessário que o local monitorado seja iluminado, uma vez que nem captura termal da câmera quanto a captura do laser requerem luz no ambiente para funcionarem.

3 FERRAMENTAS USADAS NA PESQUISA

Como material da pesquisa estão sendo utilizados vídeos capturados pela câmera térmica FLIR PathFindIR. Com bases nestes vídeos foi possível executar os testes do algoritmo de detecção de calor.

Também estão sendo utilizadas as informações provientes do sensor laser. O modelo utilizado na captura das informações do ambiente foi o Hokuyo URG-04LX. Este possui um campo de visão para detecção de 240°. A distâncias mínima reportada é de 20 milimetros e a máxima de 4 metros.

Para cumprir com esse objetivo, é utilizada a biblioteca de processamento de imagens OpenCV (Bradski & Kaehler 2008) e o software Player-Stage (Collett et al. 2005).

A OpenCV fornece uma série de funções para visão computacional em tempo real. Estas facilitam o manuseio com imagens e vídeos de vários formatos e resoluções, além disso, também permite a captura de imagens e vídeos provenientes de dispositivos de captura externos.

O Player-Stage é uma plataforma que promove uma interface entre os sensores e o computador por meio de rede TCP/IP. Além disso, o Player-Stage também permite a simulação de robôs, sensores e atuadores em um ambiente bi-dimensional mapeado.

4 DESCRIÇÂO DO MÉTODO PROPOSTO

Este trabalho envolve a captura e o tratamento de informações provindas de dois sensores diferentes (câmera termal e laser), portanto, para cada sensor é feito um processamento diferente.

4.1 Processamento da Imagem Termal

A captura em tempo real da câmera é feita utilizando as funções da biblioteca OpenCV (Bradski & Kaehler 2008). Com essas funções, é possível obter uma imagem digitalizada de 3 camadas de 8 bits no formato RGB com a resolução de 640 x 480 pixels.

A imagem gerada pela câmera térmica é em escala de cinza (Figura 4.1), representando o calor presente na cena. Por conta disso, para otimizar o processo, é feita uma transformação que usa a média aritmética das 3 camadas (RGB) para obter uma imagem em escala de cinza formada por apenas uma camada, ao contrário da gerada originalmente pela função da OpenCV que possui 3 camadas.

Figura 4.1. Captura em corredor de acesso.

Com o objetivo de obter bons resultado, a um custo reduzido de processamento, foi feito o uso do histograma de tonalidades da imagem (claro=quente, escuro=frio). Logo, para cada imagem capturada é gerado um histograma, com o eixo x representando a cor (com variação de 0 a 255) e o eixo y representando o número de pixels da imagem que possuem determinada cor/temperatura (Figura 4.2). O histograma da imagem permite determinar a quantidade de calor presente em uma cena, assim como, comparando histogramas pode-se verificar a variação de calor em um ambiente.

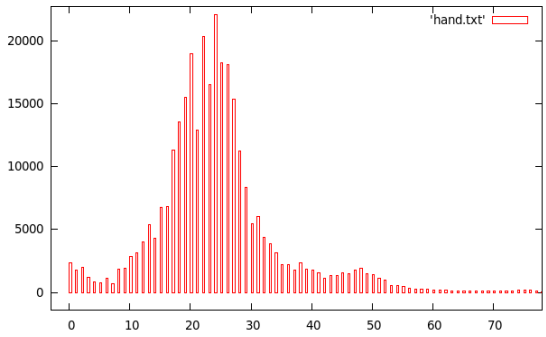

Figura 4.2 Histograma da Figura 4.1

A detecção do ser humano é feita com base na captura de calor, ou seja, sabendo que o corpo humano possui temperatura relativamente superior ao resto do ambiente, usa-se esta característica como fator principal para processar a imagem. Contudo, há no ambiente alguns objetos como lâmpadas e outros dispositivos eletrônicos que também geram calor. Devido a essa questao, não é trivial a distinção entre um objeto e um corpo humano. Para resolver este problema, é necessário que o algoritmo receba como entrada uma imagem do local gerada pela câmera térmica quando não há a presença humana, para efeitos de calibração do sistema. A partir desta imagem é gerado um histograma que é utilizado como base para a análise dos histogramas feitos a partir das novas imagens capturadas em tempo real.

Como é necessário percorrer todos os pixels da imagem tanto no momento da geração do histograma como na transformação da imagem de três camadas para uma, foi optado por executar essas duas etapas em um mesmo passo (Algoritmo 4.1). O resultado do algoritmo será a imagem processada e o seu histograma correspondente.

Algoritmo 4.1. Geração de Histograma e Imagem com uma camada.

```
gerarHistroframaImagem(
inteiro histograma[256],
imagem original, imagem processada)
1   Inteiro auxiliar;
2   Inicio
3     para todo pixel da imagem
4     inicio
5      auxiliar  =
         (original.camada1 +
         original.camada2 +
         original.camada3) / 3;
6      histograma[auxiliar]++;
7      processada.camada1 = aux;
8     fim
9   Fim.
```

Com a geração do histograma base, assim como a geração da imagem a ser processada pelo resto do algoritmo, o próximo passo é obter a faixa de posições que se destacam no histograma (zona de maior variação na comparação dos histogramas). Essas posições são descobertas calculando a razão entre os valores do eixo y dos histogramas encontrados em cada captura com o eixo y do histograma da imagem sem presença humana (denominado de histograma base). A partir desta razão descobre-se um fator de diferenciação entre as duas imagens (atual e base). Considerando este fator, é feita uma análise em blocos dos fatores encontrados em cada tonalidade (cor). Dessa forma, ao determinar um bloco cujos fatores possuem valores considerados altos (picos no histograma), determina-se que o primeiro valor (cor) do bloco é o ponto inicial a ser usado no processamento. Para calcular a posição final utiliza-se o mesmo processo para encontrar a posição inicial, sendo que nesta etapa, a leitura do histograma começa da última posição e a cor escolhida ao encontrar o bloco com grande variação é a última.

Após a obtenção do histograma, da imagem convertida para uma camada, da posição final e inicial da faixa dentro do histograma, inicia-se a etapa responsável por destacar as informações relevantes e descartar as irrelevantes.

Inicialmente, nesta parte do algoritmo calcula-se a diferença entre a posição final e inicial. Em seguida, divide-se o número total de cores (256) por esta diferença. O resultado é um novo fator a ser usado. A partir deste, faz-se uma normalização, isto é, os valores que variam da posição inicial até a posição final determinada anteriormente, passam a variar de 0 a 255. Isso destaca as informações mais relevantes pelo fato das cores que sofreram variação serem aquelas que aumentaram com a chegada de uma humano no ambiente filmado pela câmera. Portanto, ao normalizar essas faixa de cores, destaca-se os pixels que representa o humano na imagem.

Além disso, é atribuída a cor 0 (preta) a todos os pixels que possuem cores com valor menor que a cor escolhida como ponto inicial ou valor maior que a cor escolhida como ponto final (Algoritmo 4.2).

Algoritmo 4.2. Processamento da imagem para segmentação do objeto de interesse

```
processarImagem(
 inteiro minPosHist, inteiro maxPosHist,
 imagem processada)

1    float diferenca = maxPosHist -
       minPosHist;
2    float fator = 256/diferenca;
3    Inicio
4      para todo pixel de processada
5      inicio
6        se ((pixel < minPosHist) or
             (pixel > maxPosHist))
7          pixel = 0;
8        senão
9          pixel =
           (pixel - diferenca)*fator;
10     fim
11   Fim.
```

Ao final deste processo de transformação da captura da câmera, teremos uma nova imagem em que praticamente toda informação desnecessária foi anulada ao passo que o humano a ser detectado foi destacado. A partir de agora resta determinar em que local na imagem o humano está. Para isto, é feito inicialmente uma contagem em cada linha vertical da imagem de quantos pixels com cores claras são encontrados em cada linha. Com essa contagem feita, procura-se qual é o intervalo de linhas que contém a maior contagem. A esse intervalo será atribuída a posição do humano invasor dentro da cena capturada.

4.2 Processamento do Sensor Laser

A digitalização das informações gerada pelo sensor laser é feita utilizando as funções do Player-Stage (Collett et al. 2005). Estes dados são retornados em forma de um vetor contendo para cada ângulo a distância entre o sensor e um objeto.

Além disso, pode-se também utilizar o Player-Stage (Collett et al. 2005) para fazer uma captura simulada de informações. Para isso, faz-se uma mapa bi-dimensional e nele carrega-se um robô com um sensor (Figura 4.3). Com estas simulações é possível efetuar testes com os algoritmos criados mais facilmente. Contudo, não pode-se afimar que o algoritmo de fato funciona perfeitamente devido a quantidade de ruído nas informações capturadas em ambiente real, que é relativamente maior às capturas feitas em ambiente simulado.

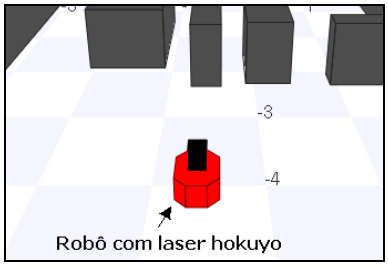

Figura 4.3. Robô simulado em ambiente mapeado.

O inicio do processo de tratamento dos dados retirados do sensor baseia-se em detectar e identificar com a captura do laser os objetos que estão próximos ao robô (paredes, obstáculos e pessoas). Também deve ser calculada as dimensões destes objetos e a posição deles em relação ao sensor.

Inicialmente, armazena-se em uma lista encadeada os pontos válidos (colisões com objetos), isto é, aqueles que retornam distâncias dentro da variação coberta pelo hardware do sensor (de 20 mm a 4 metros). Em seguida, calcula-se para cada ponto válido as distâncias em relação ao eixo das abiscissas (eixo X) e ao eixo das ordenadas (eixo Y), sendo que o ponto inicial do plano cartesiano é onde encontra-se o sensor laser.

Após obter as posições de cada ponto no plano cartesiano, é executado um processo que gera uma lista de blobs (Algoritmo 4.3), isto é, é criada uma lista de blocos que representam cada objeto detectado no ambiente varrido pelo sensor laser.

Algoritmo 4.3 Geração de blobs

```
gerarBlob(listaPontos pontos,
          listaBlobs blobs)
1    Ponto ponto;
2    Blob blob;
3    Inicio
4      ponto = pontos.primeiro;
5      enquanto (ponto != NULL)
6      inicio
7        se ((ponto < 4)and(ponto > 0.02))
8        inicio
9          se ((blob = buscaBlob(blobs,
                       ponto)) != NULL)
10         inicio
11           se (blob.Xi > ponto.X)
12              blob.Xi = ponto.X;
13           senão
14           inicio
15             se (blob.Xf < ponto.X)
16                blob.Xf = ponto.X;
17           fim
18           se (blob.Yi > ponto.Y)
19              blob.Yi = ponto.Y;
20           senão
21           inicio
22             se (blob.Yf < ponto.Y)
23                blob.Yf = ponto.Y;
24           fim
25         fim
26         senao
27         inicio
28           blob = criarBlob(ponto.X,
                              ponto.Y);
29           blobs =adicionarBlob(blob);
30         fim
31       fim
32       ponto = ponto.prox;
33     fim
34   Fim.
```

O primeiro passo deste procedimento baseia-se em pegar um ponto da lista encadeada gerada e criar um blob tendo como posição inicial e final o ponto retirado da lista. A partir deste momento, inicia-se o passo em que é pego o próximo ponto da lista. Ao pegar esse ponto, é verificado se há algum blob cujo a posição deste ponto no plano esteja dentro das dimensões do blob ou se está em uma posição próxima. Para este sensor, devido ao nível de precisão do modelo do hardware, foi considerado como próximo um ponto que está há uma distância menor que 3 centímetros dos limites do blob (Figura 4.4). Se o for determinado que o ponto encontra-se dentro das dimensões do blob, o número de pontos pertencentes àquele blob é incrementado em uma unidade. Caso seja determinado que o ponto está próximo aos limites do blob, além da adição do número de pontos pertencentes àquele blob, é também adotado este novo ponto como um novo limite da extremidade do blob. Contudo, se for detectado que este ponto não está próximo a nenhum blob já criado, então é criado com este ponto um novo blob.

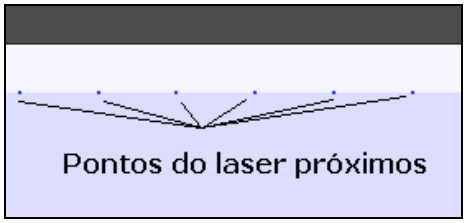

Figura 4.4. Exemplo de pontos do laser próximos.

Após todos os pontos da lista terem sido verificados pelo algoritmo, é feito um processo de refinamento dos blobs (Algoritmo 4.4). Este é feito devido aos ruídos provocados pela captura (Figura 4.5). Estes ruídos por muitas vezes podem dividir objetos em dois ou mais blobs porque alguns feixes do laser não detectam a posição corretamente.

Algoritmo 4.4 Refinamento dos blobs

```
refinarBlobs(Blobs blobs)
     Blob blob, aux;
1
2    Inicio
3      blob = blobs.primeiro;
4      enquanto ((blob != NULL) and
                 (blob.prox != NULL)
5      inicio
6        se ((blob.prox.Xi – blob.Xf) < 0.05)
7        inicio
8          blob.Xf = blob.prox.Xf;
9          se (blob.Yi > blob.prox.Yi)
10             blob.Yi = blox.prox.Yi;
11         se (blob.Yf < blox.prox.Yf)
12             blob.Yf = blox.prox.Yf;
13         blob.nPontos += blob.prox.nPontos;
14         aux = blob.prox;
15         blob.prox = blox.prox.prox;
16         liberar(aux);
17       fim
18       senao
19         blob = blob.prox;
20     fim
21   Fim.
```

O processo é feito por meio de comparação, semelhante ao processo anterior, contudo, agora é feita uma comparação entre os blobs criados e não entre um blob e ponto.

Figura 4.5. Exemplo de leitura do laser com ruído.

Esta comparação é feita pegando-se o primeiro blob e comparando a posição final deste com a posição inicial do próximo blob da lista. Caso estas posições sejam consideradas próximas entre sí, é feita uma fusão de blobs, ou seja, o número de pontos destes blobs são somados, em seguida é determinada que, a posição inicial do primeiro blob é a posição inicial do novo blob, e a posição final do segundo blob é a posição final do novo blob. Devido a precisão e a quantidade de ruídos do sensor utilizado na pesquisa, foi determinado como proximidade entre blobs a distância de até 5 centimetros.

Com o processo de refinamento, o mapeamento do am-biente coberto pelo laser é finalizado. Assim, é obtida como saída deste processo uma lista de blobs que representam todos objetos detectados pelo sensor laser.

4.3 Fusão de Sensores

Com o término do processamento dos dados da câmera térmica e do sensor laser, resta agora a fusão destas informações. Busca-se, combinar os dados provenien-tes de naturezas diferentes com o objetivo de fornecer aos sistemas informações que permitam determinar resultados mais precisos de localização (Luo & Kay 1995).

Por meio da câmera termal é possível detectar se em um ambiente há a presença humana através do calor irradiado. E assim, determinar uma posição aproxi-mada a respeito de onde o invasor se encontra em relação ao robô e a câmera que está acoplada nele.

Através do laser, são detectados/reconhecidos os objetos (humanos, paredes e obstáculos) do local monitorado. Além disso também é determinada, pelas informações dadas pelo laser, a dimensão de cada objeto e também a distância precisa de cada um em relação ao robô onde o laser encontra-se acoplado.

Tais informações (laser e câmera térmica) analisadas separdamente são úteis porém ineficientes em um monitoramento mais preciso do ambiente. Todavia, ao unir tais informações (fusão), é possível determinar dentre os objetos detectados pelo laser qual deles é o humano, relacionando a posição do invasor detectado pela câmera com a posição de um dos objetos reconhe-cidos pelo laser. Além disso, ao se determinar dentre os objetos qual deles é o invasor, pode ser feita uma perseguição do mesmo, juntamente com os desvio dos obstáculos. Logo, pode-se desviar de todos os outros objetos (frios) que não foram reconhecidos como sendo um humano invasor, ao mesmo tempo em que o robô avança direcionando-se para o alvo (quente).

Essa etapa da pesquisa ainda está em desenvolvimento no que diz respeito a parte de relacionar a posição do humano detectado pela câmera com algum objeto reconhecido pelo laser. A complexidade neste processo ocorre por conta de não haver uma função linear e direta que transforme a posição do corpo na imagem (medida em pixels) em uma posição referente ao objeto reconhecido pelo laser. Dentre os motivos para que esta transformação não seja trivial está o modo de propagação da varredura do laser, que é radial, enquanto que a câmera efetua uma captura de uma projeção bi-dimensional da cena. O laser percebe o ambiente tridimensionalmente (medidas de quais objetos estão mais perto ou mais longe do robô), enquanto a câmera gera uma imagem plana (2D) com uma projeção perspectiva da cena, isto é, por meio dela não é possível determinar facilmente a profundidade em que se encontra o invasor. Atualmente estão sendo desenvolvidos algoritmos para calibração, ajuste, e registro dos pontos capturados pelo laser e câmera.

5 RESULTADOS

Para testar a eficiência do algoritmo de detecção de pessoas utilizando a câmera térmica, foram usados os cenários A (corredor de acesso), B (sala residencial) e C (ambiente frio com captura em tempo real). Os cenários A e B foram escolhidos por serem ambientes públicos "típicos", onde optou-se por fazer filmagens com a câmera térmica e a nelas aplicar os algoritmos. O cenário C foi adotado por ser no próprio laboratório onde a pesquisa ocorria, onde a captura em tempo real pôde ser feita de modo mais prático e direto.

Nos testes feitos, em todos estes três cenários, foi obtido êxito na detecção do invasor (Figura 5.1, 5.2 e 5.3), assim como o descarte da maior parte das informações que poderiam atrapalhar no momento da perseguição do alvo, como por exemplo o reflexo do calor do invasor em uma parede e no chão (Figura 5.1).

No cenário A, embora existam objetos quentes que também ficam em destaque, no momento que é verificada a posição do corpo na imagem, esses objetos não são considerados devido ao tamanho, e consequen-temente os pixels claros que estes possuem.

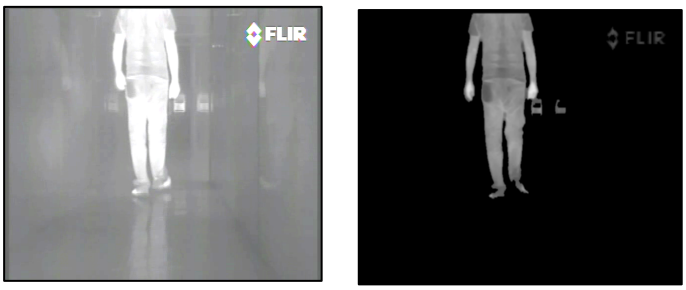

Figura 5.1. Aplicação em corredor de acesso

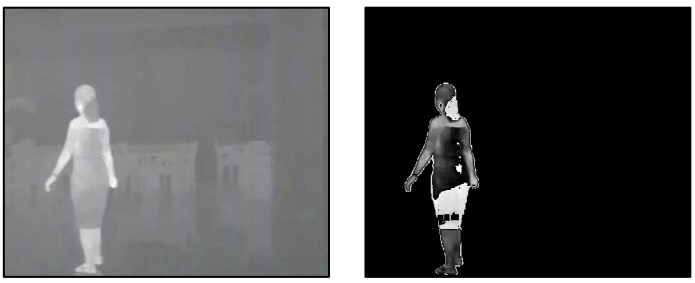

Figura 5.2. Aplicação em sala residencial

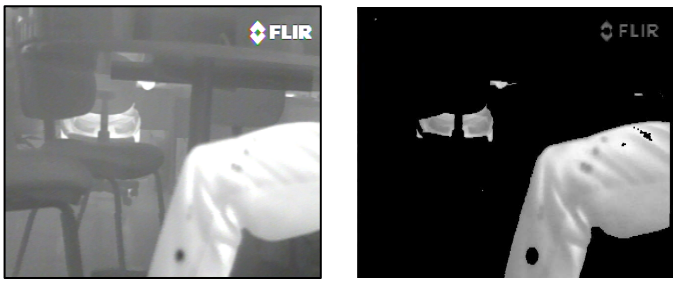

Figura 5.3. Aplicação em tempo real

Para testar e avaliar mais precisamente a eficiência do laser foram feitos testes utilizando o simulador dispo-nível no software player-stage (Collett et al. 2005). Foram utili-zados 2 cenários para fazer estes testes. O cenário A representou um final de corredor e com poucos objetos. O cenário B representou um ambiente mais populado e fechado, com paredes longas sem portas de acesso e muitos objetos a frente do robô.

Em todos testes feitos nestes cenários, o algoritmo obteve sucesso na detecção dos objetos e obstáculos, criando corretamente os blobs, mesmo quando a quantidade de ruído era relativamente alta (Figura 5.4, 5.5; Tabela 5.1 e 5.2). A eficiência do algoritmo diante do alto níve de ruído é observada na etapa de refinamento que corrige a geração dos blobs.

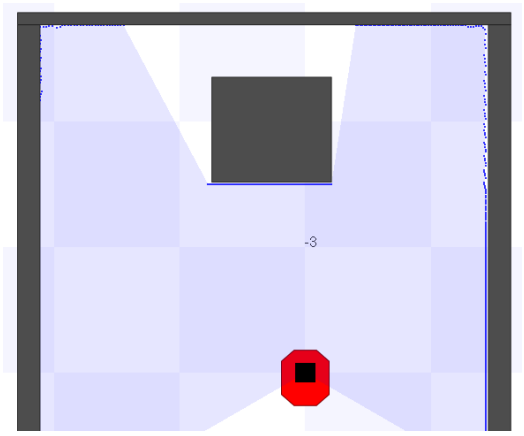

Figura 5.4 Reconhecimento em fim de corredor.

#	Xi (m)	Xf (m)	Yi (m)	Yf (m)	nPontos
1	-2.120	-1.436	0.036	2.768	176
2	-0.769	0.213	1.499	1.503	101
3	0.410	1.443	0.033	2.759	229

Tabela 5.1 Blobs gerados a partir da Figura 5.4.

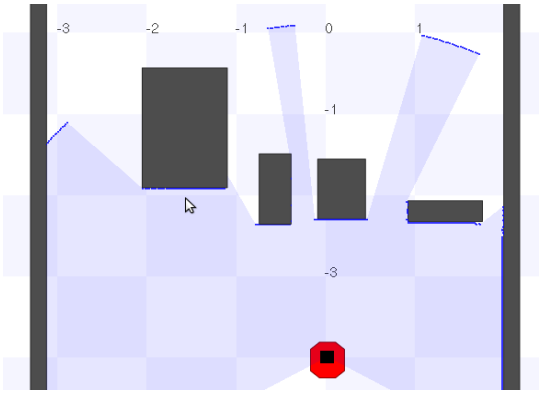

Figura 5.5 Reconhecimento de ambiente cheio.

#	Xi (m)	Xf (m)	Yi (m)	Yf (m)	nPontos
1	-3.160	-3.150	0.034	2.273	101
2	-2.056	-1.119	2.074	2.274	53
3	-0.798	-0.408	1.631	2.491	48
4	-0.135	0.445	1.689	1.690	56
5	0.889	1.714	1.628	1.908	62
6	1.959	1.982	0.033	1.845	121

Tabela 5.2 Blobs gerados a partir da Figura 5.5.

6 TRABALHOS FUTUROS

Com o término da implementação da fusão dos sem-sores, junto com a detecção e reconhecimento dos objetos no ambiente (que já está operacional), será possível saber qual é a posição exata do invasor e também de cada obstáculo. Sendo assim, a próxima etapa será a navegação do robô com a perseguição do humano tendo como base a posição real do intruso e dos obstáculos a serem evitados. Esta outra parte do projeto já está em andamento, conforme descrito em (Sales et al. 2011), onde espera-se como trabalho futuro a união destas duas pesquisas a fim de obter um sistema único e eficiente de perseguição e navegação autônoma.

AGRADECIMENTOS

Agradecemos o apoio financeiro da USP-PRP (Pró-Reitoria de Pesquisa) pelo financiamento da Bolsa de Pesquisa ligada a este projeto (BIC). Agradecemos também o apoio financeiro do CNPq e FAPESP ao INCT-SEC (Instituto Nacional de Ciência e Tecnologia em Sistemas Embarcados Críticos) processos 573963/2008-8 e 08/57870-9. Por fim, gostaríamos de agradecer ao LRM (Laboratório de Robótica Móvel do ICMC/USP) e aos seus membros pelo apoio e equipamentos disponibilizados para a realização deste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

Bradski, Gary; Kaehler, Adrian. (2008). Learning OpenCV: Computer Vision with the OpenCV Library. pag. 193-219.

Collett, Toby H.; MacDonald, Bruce A.; and Gerkey, B. P. (2005). "Player 2.0: Toward a Prac-tical Robot Programming Framework". In: Proceed-ings of the Australasian Conf. on Robotics and Au-tomation (ACRA'05), Sydney, Australia, Dec. 2005.

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

Luo, Ren C.; Kay, Michael G. (1995). Multisensor Integration and Fusion for Intelligent Machines and Systems. Norwood: Ablex Publishing Corporation, 688 p.

Sales, Daniel O.; Osório,Fernando S.; Wolf, Denis F. (2011). "Topological Autonomous Naviga-tion for Mobile Robots in Indoor Environments using ANN and FSM". In: I Conf. Brasileira em Sistemas Embarcados Críticos – CBSEC 2011. São Carlos.