Fluxo óptico em câmera embarcada para seguimento de pessoa com robô movel

Diego Santos de Oliveira
Rodolfo Bonnemasou Moreira de Castilho Cortese

Orientador: Prof. Dr. Jun Okamoto Jr.

São Paulo
05 de Dezembro de 2014

Catalogação-na-publicação

Oliveira, Diego Santos de
Fluxo óptico em câmera embarcada para seguimento de pessoa com robô móvel / D.S. de Oliveira; R.B.M.C. Cortese. – São Paulo, 2014.
45 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1.Robôs 2.Fluxo óptico I.Cortese, Rodolfo Bonnemasou Moreira de Castilho II.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II.t.

Resumo

Este trabalho propõe a implementação em um robô móvel de um sistema baseado em visão que seja capaz de seguir uma pessoa sozinha se movendo em um ambiente sem obstáculos, de maneira a manter uma distância aproximadamente constante da mesma, através do fluxo óptico calculado a partir das imagens obtidas de uma câmera embarcada. Para tanto, usou-se um robô móvel, modelo Pioneer 3-AT, e uma webcam genérica. Pretende-se, assim, que os alunos aprofundem seus conhecimentos na área de visão computacional usando processamento de imagens na detecção de movimento através de algoritmos de fluxo óptico, já que é necessária a obtenção da posição da pessoa a despeito do movimento aparente do ambiente causado pela movimentação do robô.

Quando o robô está em deslocamento atrás da pessoa alvo, o algoritmo deve ser capaz de descartar o movimento relativo do ambiente à sua volta devido a tal deslocamento e identificar a qual setor da imagem pertence apenas a pessoa, que se move de maneira diferente. Para isto, usou-se um algoritmo de reconhecimento de fluxo óptico desenvolvido por Gunnar Farnebäck, seguido de uma série de filtros diferentes para a extração da silhueta da pessoa da matriz de fluxos ópticos obtida para cada estado de movimentação do robô, e posterior aprimoramento da estimativa com o uso de um filtro de Kalman.

Abstract

This work proposes the implementation on a mobile robot of a vision based system that is capable of tracking one lone person moving in an obstacle-free environment, so as to maintain an approximately constant distance from her, using optical flow calculated from the images obtained from an onboard camera. To do so, a mobile robot, model Pioneer 3-AT, and a generic webcam was used. One of the objectives is that the students enhance their knowledge in the area of computational vision through the image processing for movement recognition using optical flow algorithms, since it is necessary to obtain the person's position despite the environment's relative movement caused by the robot's movement.

When the robot is following the target person, the algorithm must be able to discard the relative motion of the environment that is due to such displacement and identify the image sector to which belongs only the person that moves differently. To achieve thus, an optical flow recognition algorithm by Gunnar Farnebäck was used, followed by a series of different filters for the extraction of the person's silhouette from the optical flow matrix for each of the robot's movement state, and later estimate improvement using a Kalman filter.

Sumário

Lista de Figuras

1

Introdução

A identificação de pessoas sem necessidade de interação específica do usuário torna mais natural a interação homem-máquina, assim como possibilita que robôs móveis identifiquem melhor elementos do ambiente ao seu redor, permitindo, por exemplo, a supervisão de idosos e de bebês, assim como a vigilância e o reconhecimento de elementos hostis em implementações mais específicas [5].

Através de sensores como câmeras, lasers, ou sonares, é possível fornecer a um robô informações sobre o mundo real. Tais elementos, entretanto, são apenas um modelo da realidade, que precisa passar, ainda, por uma análise matemática a fim de que se identifique o que representam tais informações. Pode-se, por exemplo, fazer comparações de parte das imagens obtidas com imagens conhecidas de objetos similares [6, 7, 8], analisar padrões de cores [9], ou analisar padrões de movimento na imagem [10, 11].

O presente trabalho tem como objetivo utilizar técnicas de visão computacional para detectar padrões de movimento de uma pessoa em uma sequencia de imagens com o uso de fluxo óptico, através do desenvolvimento e implementação de um software capaz de reconhecer a presença e posição dessa pessoa em um ambiente e, então, fazer com que um robô móvel, com uma câmera embarcada, a siga. Usar-se-á um robô móvel, modelo Pioneer 3-AT, sendo, portanto, parte do escopo do projeto a programação e controle do mesmo. O sistema deve, para que se consiga seguir a pessoa, ser capaz de reconhecer os padrões de movimento da mesma mesmo quando o ambiente ao seu redor possui movimento relativo devido à movimentação do robô.

2

Detecção e rastreamento de pessoas com visão computacional

O uso das informações obtidas por uma câmera consiste em, primeiramente, identificar qual seção da imagem corresponde à pessoa a ser seguida e, então, avaliar a posição da pessoa em relação ao robô para que o mesmo possa segui-la. Há diversos algoritmos que fornecem meios para tal.

Existem algoritmos que dependem do reconhecimento da forma humana em uma imagem através da comparação de contornos, como os métodos de "*template matching*" [12], "*shape fitting*" [13], "*human modelling*" [14]. Outros tipos de algoritmos fazem a comparação de diversos setores da imagem com uma base de dados de imagens de objetos, como faz Sotelo [15] para identificar pedestres no caminho de um carro, e Braun [16] para identificar rostos humanos.

Alguns algoritmos fazem uso da separação de uma forma em movimento de uma imagem estática, como ocorre com o "*background subtraction*" [17]. Neste caso, há, entretanto, necessidade de que a imagem seja adquirida por uma câmera estática, o que não é o caso em uma câmera embarcada em um robô móvel. Outros, usam o histograma de cores da imagem para rastrear um trecho ou objeto. Esses são muito eficientes quando aliados a outras informações, mas, por si só, apresentam dificuldade para diferenciar o objeto buscado de objetos ao fundo com cores similares, como mostra o trabalho de Liem [18].

Uma outra vertente de pesquisadores usa, além de câmeras e de processamento de imagens, dados obtidos de outros sensores, como laser e ultra-som. Kobilarov [19], por exemplo, através de uma análise probabilística dos dados extraídos de um sensor a laser associada aos dados obtidos da câmera, define se o objeto é uma pessoa a ser seguida e sua distância.

Por fim, há os métodos de fluxo óptico, em que o movimento relativo de pontos da imagem é analisado. Tratam-se de métodos suficientemente eficientes do ponto de vista computacional para execução em tempo real para aplicações em que a precisão é suficiente para que a estimativa da posição da pessoa permita ao robô manter uma distância predefinida da mesma [20]. Como se tem a magnitude, direção e sentido dos deslocamentos de todos os pontos, é possível, através de um tratamento apropriado, identificar qual parte desse movimento é

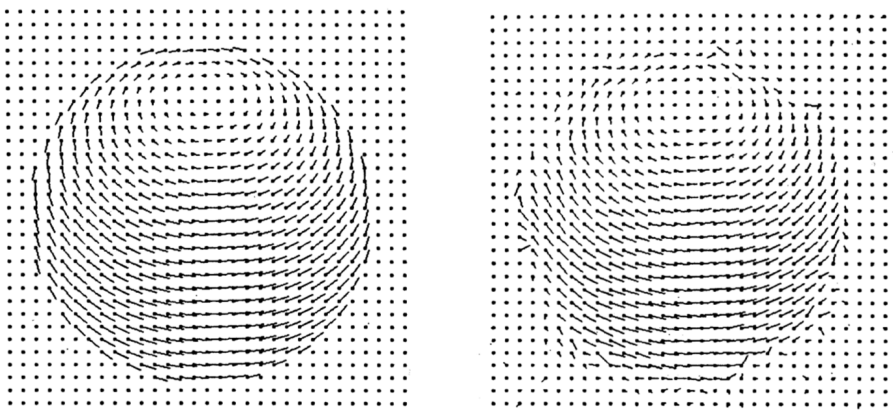
devido ao movimento da câmera e qual não é. Dada a restrição de único objeto móvel na cena ser uma pessoa, nada mais é necessário para que se tenha o tamanho e posição da mesma. Handa [21], por exemplo, obteve sucesso em fazer seu robô seguir uma pessoa mesmo em ambientes com pouca iluminação. Yamane [22], Piaggio [20] e muitos outros também chegaram a resultados similares.

Nesse projeto propõe-se o uso de fluxo óptico para detecção do movimento da pessoa à frente da câmera embarcada no robô e de um algoritmo para determinar qual deve ser a ação a ser adotada pelo robô com base na variação do fluxo óptico. Com isso, pretende-se demonstrar a aplicabilidade de métodos baseados em fluxo óptico em plataformas móveis.

2.1 Fluxo Óptico

O fluxo óptico é uma representação vetorial do deslocamento de setores de padrão de brilho equivalente entre um quadro e outro de uma sequência de imagens através da verificação da correspondência de tais setores em ambas as imagens [23], visando-se obter uma representação do campo de movimento na imagem através de um campo de fluxos ópticos, que é o resultado do cálculo dos fluxos ópticos, um campo vetorial em que cada vetor, aplicado ao ponto do primeiro quadro em relação ao qual foi calculado, representa uma estimativa da velocidade instantânea daquele ponto no par de imagens, o que possibilita a identificação e quantificação do movimento de qualquer coisa cuja representação pictórica possua velocidade diferente de sua vizinhança[24].

O campo de movimento é a projeção bidimensional do movimento tridimensional de superfícies [1], ou seja, é a projeção bidimensional do movimento real de uma cena tridimensional, enquanto o campo de fluxos ópticos é o movimento aparente de padrões de brilho na imagem, ou seja, uma representação do movimento bidimensional adquirida através de uma interpretação de um modelo da realidade, como mostra a figura 2.1.



(a) vetores de deslocamento real (b) vetores de fluxo óptico

Figura 2.1: Representação do movimento de uma esfera girando em torno de seu eixo (adaptado de [1]).

Há duas estratégias diferentes para a resolução de problemas de correspondência em visão computacional: abordagens densas, em que todos os pontos da imagem são analisados entre um quadro e outro através do uso de informações sobre a vizinhança do ponto para a resolução de suas equações, e abordagens esparsas, em que é realizada, previamente, uma redução de dimensionalidade, de maneira que sejam analisados apenas setores definidos com grande contraste em relação a sua vizinhança imediata, chamados de "features", ou atributos, que são facilmente reconhecíveis em ambos os quadros, o que torna sua execução mais rápida em troca do descarte de informaçõs de precisão menor [25, 26, 11]. Algoritmos de fluxo óptico denso, como o de Horn-Schunck [1], conseguem identificar o fluxo óptico em setores da imagem em que não se teria informação usando um algoritmo esparso, entretanto, segundo Nourani-Vatani [23], são mais suscetíveis a ruído e exigem um tempo maior para a realização do cálculo.

Devido à baixa resolução de trabalho, observou-se melhores resultados usando métodos de cálculo de fluxo óptico densos graças à maior densidade de resultados, e, portanto, o foco do presente trabalho está em tais métodos. Dada a natureza do problema, em que se deseja rastrear o movimento de uma pessoa, que representa uma parte da imagem de textura não uniforme e frequentemente contrastante de seus arredores em uma projeção bidimensional, tais métodos apresentam resultados satisfatórios para a estimativa da posição desejada.

2.1.1 Métodos esparsos

2.1.1.1 Atributos

O cálculo do fluxo óptico através de algoritmos esparsos se baseia na hipótese de que os pontos a serem rastreados possuem significativo gradiente em relação à sua vizinhança, o que torna esses pontos facilmente rastreáveis usando apenas informações sobre seus padrões de brilho. A pontos que possuem significativo gradiente em apenas uma direção se dá o nome de "edges", ou quinas, e aos que possuem elevado gradiente em ao menos duas direções não coincidentes se dá o nome de "corner", ou cantos [27]. Antes de se calcular o fluxo óptico esparso, é, portanto, necessário identificar tais setores da imagem.

"Features", ou atributos são os pontos escolhidos para terem seus fluxos avaliados. Podem ser quinas, de diferentes qualidades, ou até mesmo cantos na ausência de suficientes quinas. Atributos de má qualidade, entretanto, afetam significativamente a robustez desse tipo de algoritmo de cálculo de fluxo óptico. Bons atributos são encontrados, geralmente, em regiões da imagem que apresentam grandes variações de gradiente em um pequeno espaço, como bordas de objetos e texturas granuladas. Há diversos algoritmos diferentes para o cálculo de atributos, cada um de características bem diferentes, como mostra Nourani-Vatani [23] em testes comparando a performance de alguns dos principais.

2.1.1.1.1 Cálculo dos atributos Em 1988, Harris [27] propôs um algoritmo para a seleção de atributos conhecido como "Harris Corner Detector". Tal algoritmo funciona da seguinte maneira:

Considere uma imagem em tons de cinza I. Varre-se uma janela $w(x, y)$ de I, de comprimento u na direção x e v em y, e calcula-se a variação da intensidade através da fórmula:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

Em que:

$w(x,y)$ é a janela ao redor do ponto (x,y);

$I(x,y)$ é a intensidade de tom em (x,y);

$I(x+u, y+v)$ é a intensidade na janela deslocada $(x+u, y+v)$;

Como se busca janelas que contenham grande variação de intensidade, é preciso maximizar a equação acima, especificamente o termo:

$$\sum_{x,y}[I(x+u, y+v) - I(x,y)]^2$$

Usando expansão de Taylor:

$$E(u,v) \approx \sum_{x,y}[I(x,y) + u \cdot I_x + v \cdot I_y - I(x,y)]^2$$

Ou, na forma matricial:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \cdot M \cdot \begin{bmatrix} u \\ v \end{bmatrix}$$

Em que:

$$M = \sum_{x,y} w(x,y) \cdot \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix}$$

Calcula-se, então, para cada janela, uma pontuação arbitrária R_{Harris} que permita comparar de maneira objetiva diferentes setores, da seguinte forma:

$$R_{Harris} = \det(M) - k \cdot ((\mathrm{Tr}(M))^2$$

Sendo:

$$\det(M) = \lambda_1 \cdot \lambda_2$$
$$\mathrm{Tr}(M) = \lambda_1 + \lambda_2$$

A partir da definição de um valor limite mínimo no código, define-se que o setor é considerado um atributo se a pontuação R_{Harris} atribuída ao mesmo for maior que tal valor.

Shi [28] propôs, posteriormente, um algoritmo chamado “Good Features To Track" baseado no trabalho descrito anteriormente, cuja principal alteração é que se usa, para o cálculo da pontuação R, o seguinte:

$$R_{GFTT} = \min(\lambda_1, \lambda_2)$$

E, então, o valor de R_{GFTT} é comparado ao limite mínimo definido no código em vez de R_{Harris}. O algoritmo faz também uma comparação entre a qualidade de atributos encontrados próximos uns dos outros, de maneira a descartar atributos fracos adjacentes a outros mais fortes, fornecendo, assim, uma melhor distribuição das informações obtidas sobre a imagem. Como mostrou Nourani-Vatani [23], o algoritmo “Good Features To Track" é mais eficiente e possui praticamente o mesmo tempo de execução que o “Harris Corner Detector", e é um dos mais usados na área.

2.1.1.2 Algoritmo de Lucas-Kanade piramidal

O algoritmo Lucas-Kanade piramidal, que usa uma abordagem esparsa, com o uso de atributos, é baseado em um algoritmo desenvolvido por Lucas-Kanade em 1981 [29], um dos pioneiros nesse campo de estudo. Nesse, é assumido que os deslocamentos dos atributos são pequenos e que toda sua vizinhança se move com velocidade constante. Admite-se que a curva de brilho da região ao redor do ponto é constante em ambos os quadros e, após algumas aproximações, chega-se a uma sistema que pode ser resolvido de maneira iterativa.

Tal algoritmo, entretanto, é bastante lento por si só. Pode-se, por outro lado, utilizar uma abordagem de aplicação chamada implementação piramidal. Nessa, são criadas versões mais grosseiras da imagem original através da aplicação de repetidas reduções de resolução. O fluxo óptico é, então, calculado na imagem mais grosseira e esse resultado é usado para inicializar a busca na camada mais fina subsequente até que se chegue à imagem original, como exemplifica a figura 2.2. O cálculo nas camadas superiores da pirâmide envolve menos incógnitas e, portanto, é mais rápido, e a inicialização de cada camada a partir da anterior também significa que menos iterações são necessárias a cada camada. Por esse motivo, a implementação piramidal de algoritmos tende a ser significativamente mais rápida que a simples aplicação do algoritmo na imagem original e, como mostra Baker [30], a aplicação do algoritmo Lucas-Kanade piramidal apresenta resultados bastante satisfatórios, sendo melhor que o algoritmo original no tratamento de ruído, porém falhando em capturar objetos pequenos em movimento rápido, o que não afeta nosso trabalho.

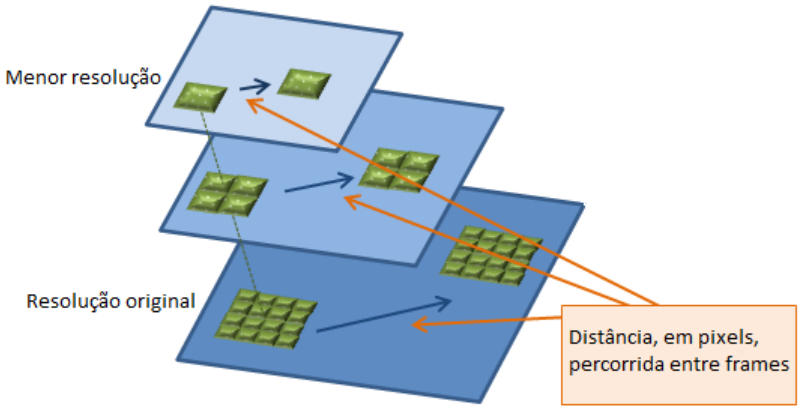


Figura 2.2: Implementação piramidal do algoritmo (adaptado de [2]).

2.1.2 Métodos densos

2.1.2.1 Horn-Schunck

Horn-Schunck [1] desenvolveu um método para o cálculo dos vetores de fluxo óptico de uma sequência de imagens baseado na observação de que a equação básica para a taxa de variação do brilho da imagem, equação 2.1, resulta em uma restrição e de que a velocidade de fluxo possui duas componentes. Seja $E(x, y, t)$ uma representação do brilho na cena, e $u = \frac{dx}{dt}$ e $v = \frac{dy}{dt}$ as componentes da velocidade de fluxo, tem-se, portanto, em uma situação ideal:

$$\frac{\partial E}{\partial x} \cdot u + \frac{\partial E}{\partial y} \cdot v + \frac{\partial E}{\partial t} = 0 \tag{2.1}$$

Para chegar a uma equação de solução única, entretanto, já que há duas componentes a serem consideradas, fez-se necessário mais uma restrição: considerou-se que o fluxo óptico varie pouco entre um ponto e outro adjacente, ou seja, que o campo de fluxos seja algo suave. Tal restrição foi imposta através da minimização da soma dos quadrados dos Laplacianos das componentes da velocidade do fluxo. Tais Laplacianos são definidos por:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \text{ e } \nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}$$

Em vez de restrições rígidas, no entanto, foram impostas penalidades para pontos que violem a restrição imposta pela equação básica da taxa de variação do brilho, e também penalidades para fluxos ópticos com taxa de variação em relação a suas vizinhanças maior que o delimitado.

Molnar [31] faz uma analogia de tal abordagem a problemas de visão computacional como sendo um método de minimização de energia, um problema de cáculo variacional, no qual, no caso de fluxo óptico, o propósito é determinar um campo de minimização de energia: tal campo deve satisfazer a restrição óptica (no caso, a taxa de variação do brilho), assim como minimizar um traço interno do campo buscado (no caso, a taxa de variação dos vetores de fluxo). Embora essa abordagem variacional não possa ser baseada em atributos [32], que possuem maior rastreabilidade que setores genéricos da imagem, já que são usadas informações sobre a vizinhança de um ponto para o cálculo do fluxo óptico no mesmo, ela fornece resultados com bastante robustez, e pequenos erros geralmente cancelam uns aos outros.

Uma omissão na abordagem feita é a negligência de fronteiras entre regiões se movendo de maneiras diferentes [32]. A abordagem variacional definida parte da ideia de que pontos vizinhos possuem fluxos similares, sem exceções, o que pode levar a erros em regiões de fronteira entre objetos em movimento em que um obscurece o outro. Há, também, uma gama de velocidades para as quais esse tipo de método funciona bem. Para grandes disparidades entre quadros sucessivos é preciso usar métodos baseados em correspondência de atributos.

2.1.2.2 Gunnar Farnebäck

Após o trabalho de Horn-Schunck vieram muitas outras abordagens densas, que usam informações da vizinhança e uma série de restrições ou de aproximações

para determinar correspondência de pontos em uma série de imagens e, consequentemente, o campo de fluxos da imagem. Algumas dessas abordagens, entretanto, precisam tratar o processo iterativo necessário para a resolução de suas equações ao longo de uma série de imagens consecutivas para que haja convergência satisfatória, como faz Schunck [1] em seu trabalho, o que limita o escopo de aplicações a vídeos em que não haja alterações significativas nos padrões de movimento em menos de algumas dezenas de quadros consecutivos. Farnebäck [33], por outro lado, trabalha apenas com um par de imagens.

Primeiramente é feita uma aproximação dos padrões de brilho nas redondezas de cada pixel, para cada quadro, por um polinômio quadrático, obtendo-se um modelo para o sinal local, expresso em um sistema de coordenadas locais, da forma:

$$f(x) \sim x^T A x + b^T x + c,$$

Em que A é uma matriz simétrica, b um vetor, e c um escalar. Os coeficiente são estimados através de um método dos mínimos quadrados ponderado aplicado à vizinhança do pixel em ambos os quadros considerados, usando-se, como peso, duas componentes: a certeza, que é nula para os pontos fora da imagem e unitária para os dentro, e a aplicabilidade, que determina significância maior a pontos da vizinhança mais próximos do pixel do que aos mais distantes, de maneira radial.

Analisa-se, então, o efeito de um deslocamento d sobre um sinal f_1, obtido do primeiro quadro, na criação de um sinal f_2, do quadro consecutivo. Assim, sendo:

$$f_1(x) = x^T A_1 x + b_1^T x + c_1$$

Tem-se:

$$f_2(x) = f_1(x - d) = (x - d)^T A_1 (x - d) + b_1^T (x - d) + c_1, \text{ logo:}$$

$$f_2(x) = x^T A_1 x + (b_1 - 2A_1 d)^T x + d^T A_1 d - b_1^T d + c_1 = x^T A_2 x + b_2^T x + c_2$$

Através da comparação dos coeficientes dos dois polinômios tem-se, então:

$$\begin{cases} A_2 = & A_1 \\ b_2 = & b_1 - 2A_1 d \\ c_2 = & d^T A_1 d - b_1^T d + c_1 \end{cases}$$

Pode-se, então, a partir dessa observação, tendo os valores dos coeficientes para ambos os sinais sido estimados pelo método supracitado, obter o valor do deslocamento do pixel, d, através de:

$$d = -\tfrac{1}{2} A_1^{-1} (b_2 - b_1)$$

A equação acima pode ser resolvida pontualmente, entretanto, isso faz com que o resultado seja extremamente sensível a ruídos [33]. Para resolver esse problema, então, os vetores de fluxo óptico são calculados localmente em cada vizinhança sem levar em conta possiveis descontinuidades no campo de velocidades. Pode-se, também, usar informações obtidas em iteração anterior na

estimativa inicial dos coeficientes, de maneira a tornar o algoritmo mais eficiente e preciso. Como mostra Aksoy [34], o algoritmo possui erro médio pequeno se comparado a outros algoritmos de fluxo óptico denso. Ainda segundo esse, apesar de o método dos mínimos quadrados usado ser computacionalmente exigente, como esses podem ser calculados através de um esquema hierárquico de convoluções, o algoritmo é muito rápido.

2.2 Reconhecimento da pessoa no campo de fluxo

O campo de fluxo óptico obtido descreve o movimento de todos os atributos reconhecíveis, no caso esparso, ou de toda a imagem, no caso denso, entre dois quadros consecutivos. É preciso, entretanto, distinguir qual parte desse movimento é devido à movimentação da pessoa e qual é consequência apenas do movimento do referencial, já que, quando o robô se move, não há mais componentes estáticos na imagem. É preciso, ainda, lidar com o ruído da imagem.

Piaggio [20], que também objetiva criar um sistema para seguir um humano, propôs resolver o problema com o uso de um filtro simples: define-se um valor limite, em relação ao qual o módulo de cada vetor de fluxo deve ser maior, usando um peso maior para a componente horizontal devido à predominância do movimento horizontal observado em um vídeo de uma pessoa caminhando nas condições propostas. Aplica-se, a esse resultado, um filtro que elimina pontos isolados da imagem, obviamente causados por ruído. Obteve-se, assim, um resultado suficientemente satisfatório para estimar, além da posição da pessoa na imagem, a largura dos ombros da mesma, o que possibilita uma estimativa da distância. Entretanto houve uma série de limitações exageradas impostas para garantir a validade de tal filtro: o robô não é capaz de executar rotação, e possui velocidade muito baixa ou, então, possui velocidade menor que a pessoa em movimento. Mesmo com essas limitações não foi obtido sucesso de maneira consistente devido à baixa frequencia de aquisição de dados e ao ruído da câmera.

Handa [21] utiliza outra metodologia para a obtenção do alvo na imagem, levando em consideração a relação entre o movimento de um atributo e o movimento dos atributos em sua vizinhança para gerar uma equação de energia de direção, capaz de desconsiderar o fluxo de um atributo se os atributos ao seu redor apresentam comportamento semelhante, isto é, direção, sentido e intensidade dentro de uma margem de erro. Tal comportamento uniforme é esperado dos elementos que se movem apenas devido ao movimento da câmera, enquanto o movimento da pessoa é mais errático, e, portanto, isolado pelo método. Para assegurar a extração da pessoa na imagem, entretanto, foi necessário incorporar ao algoritmo uma comparação da profundidade e das cores do resultado anterior com as esperadas do torso de uma pessoa.

Embora Yamane [22] apresente uma solução para se obter o alvo a partir de uma imagem estática quando o mesmo se movimenta exclusivamente paralelo ao comprimento da imagem, alguns conceitos interessantes são extraídos de seu trabalho: associa-se, à informação do fluxo óptico, regiões de brilho equivalente na imagem para adicionar robustez ao algoritmo. Para uma pessoa entrando na cena, por exemplo, uma janela circunscreve a região com fluxo óptico, gerando um retângulo de localização do alvo chamado de região de interesse. Concomitantemente, as regiões de brilho equivalentes dentro da região de interesse são extraídas e, por meio da média do fluxo óptico é determinado, então, onde se-

rão buscadas essas regiões no quadro seguinte. Regiões fora da área de interesse que eventualmente apresentem fluxo óptico devido a ruído ou a uma intercalação temporária com a pessoa e que possuam brilho semelhante a alguma das regiões de brilho buscadas são ignoradas usando uma relação de confiabilidade, que dá maior significância a regiões acompanhadas a mais tempo. Tal solução faz com que não se perca a posição da pessoa mesmo que haja uma interrupção na obtenção de fluxos ópticos por qualquer razão, usando a informação dos blocos de brilho, assim como permite que se acompanhe a pessoa mesmo que o brilho da mesma seja alterado, usando o fluxo óptico para chegar à nova informação. O algoritmo funciona bem para o caso proposto.

3

Metodologia

3.1 Requisitos do projeto

Para a demonstração da visão computacional, propôs-se programar um robô para seguir uma pessoa adulta sozinha em um ambiente sem obstáculos fechado, como corredores e salas, a uma velocidade máxima de 2 m/s. Deve ser mantida uma distância constante entre o robô e a pessoa-alvo. Para tanto, usou-se a estimativa da posição da pessoa através do campo de fluxo óptico calculado a partir de imagens adquiridas de uma câmera embarcada no robô. Em quadros de imagem em que o alvo não seja encontrado por qualquer motivo, seja sua oclusão, falta de movimento, ou excesso de ruído na imagem, uma janela de predição de posicionamento baseada em uma análise probabilística deve estimar sua localização levando em conta sua velocidade até então. O algoritmo desenvolvido deve, ao início do programa, aguardar a pessoa a ser seguida entrar em seu campo de visão ou ser identificada no mesmo e adotá-la como alvo. O robô usado é o Pioneer 3-AT (ou P3AT), de quatro rodas, com computador e bateria embarcados. Para a captação das imagens foi adaptada uma câmera ao robô.

3.2 Programação

Toda a programação foi feita em C++, linguagem escolhida tanto pela compatibilidade com as bibliotecas necessárias quanto pela familiaridade dos alunos com a mesma, usando a IDE Eclipse. Para o tratamento das imagens foi definido o uso da biblioteca openCV, que fornece funções e métodos prontos para o carregamento de imagens a partir de vídeos ou câmeras, conversão de imagens, cálculo de fluxo óptico, localização de atributos, gravação de vídeos, e diversas outras tarefas para manipulaçao de imagens. Para a comunicação com o robô usou-se a API Aria, da Pioneer, que fornece ferramentas para a obtenção dos dados dos sensores e definição das velocidades do robô.

Figura 3.1: Robô Pioneer 3-AT

3.3 Montagem

Como é necessário poder de processamento maior que o presente no computador embarcado da unidade Pioneer 3-AT, fez-se necessário executar o programa em um computador externo. Comunicação sem fio foi considerada, entretanto, devido a características da comunicação da rede sem fios do laboratório com o computador usado quando há altas taxas de transmissão de dados, que é o caso de imagens, resulta num atraso imprevisível. Assim, a melhor opção para o processamento de imagens foi a comunicação direta através da entrada serial RS-232 do robô e uma porta USB do notebook usado para o processamento de imagens e controle do robô, sendo que esse fica simplesmente posicionado sobre o robô após a inicialização do programa.

Observa-se, na execução do programa, que um aumento na resolução, apesar de trazer mais informações sobre cada quadro da imagem, aumenta consideravelmente o tempo de execução, o que reduz a taxa de amostragem das informações obtidas pelos sensores, além de melhorar apenas marginalmente a qualidade das informações sobre o fluxo óptico na cena. Decidiu-se, portanto, por uma resolução de 400x300, que mantém o aspecto original do vídeo e não apresenta suficiente perda de informação para prejudicar os cálculos, assim como permite um processamento rápido o suficiente para que se analise até 5 pares de quadros por segundo.

Com isso, poder-se-ia usar qualquer câmera para a aquisição das imagens que possua uma resolução maior ou igual a 400x300, já que é feita uma redução de resolução no programa, e que trabalhe a pelo menos 5 quadros por segundo. Neste projeto usou-se uma webcam simples de modelo Clone 11147, que trabalha em resolução VGA, 640x480, a 30 quadros por segundo, de interface USB, ligada diretamente ao notebook responsável pelo processamento das imagens.

A mesma foi adesivada na parte superior do robô, e apresentou compatibilidade com o projeto.

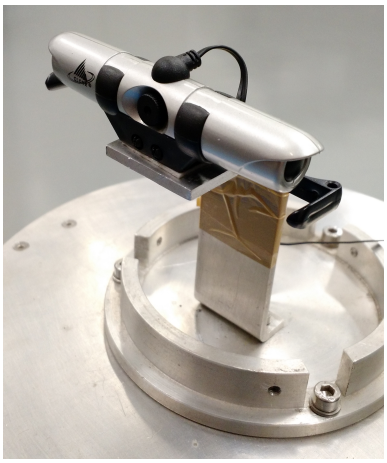


Figura 3.2: Câmera em seu suporte

3.4 Perfil de velocidade do robô

O robô usado apresenta uma curva trapezoidal de aceleração, como se pode ver na figura 3.3, o que significa que é necessário iniciar a desaceleração antes que se chegue à distância ideal, assim como é preciso levar em conta o atraso entre o processamento dos dados obtidos a partir dos sensores e a saída para o robô, que, observou-se empiricamente, chega a ser de até 0,2 segundos, sendo somente a aplicação do algoritmo de Gunnar Farnebäck responsável por metade de tal intervalo. Para resolver tal problema, a velocidade do robô é reduzida linearmente a partir de uma distância maior que a desejada em cerca de meio metro até que seja nula na distância ideal.

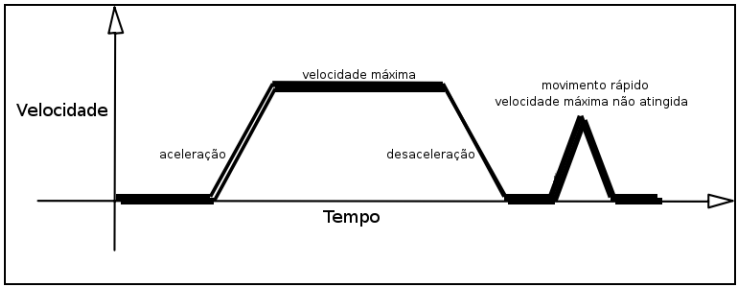


Figura 3.3: Perfil de velocidade trapezoidal do robô (adaptado de [3]).

3.5 Estados

3.5.1 Máquina de estados do robô

A concepção de máquina de estados da movimentação proposta do robô para este projeto é apresentada na figura 3.4, a seguir. Ela representa o comportamento que se espera do robô segundo a posição da pessoa identificada pelo programa.

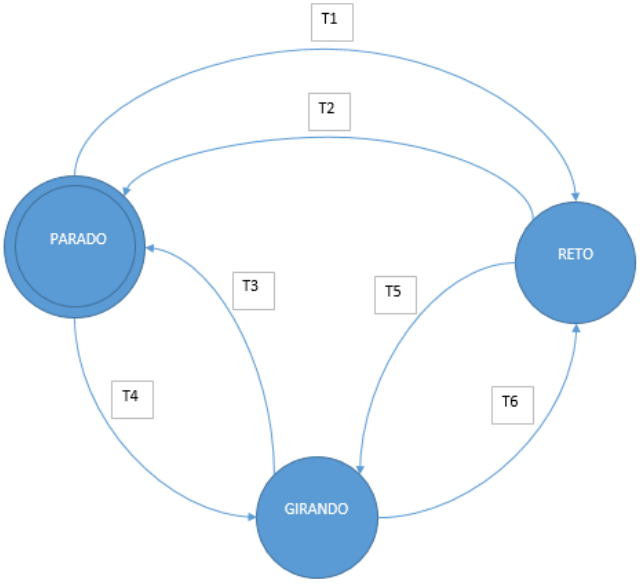


Figura 3.4: Máquina de estados do robô

Observam-se três estados – robô parado, girando e reto – e seis transições de estados – T1 a T6 -, que representam os estados finitos possíveis.

Para o controle das transições da máquina de estados do robô, dois parâmetros são usados. São eles:

1. o centro da posição do alvo;
2. a distância da pessoa ao robô, estimada usando-se a largura do alvo identificado, e comparando-se com o valor dessa largura observado na distância ideal.

Ambos com suas respectivas margens de erro.

A máquina de estados tem seu início no estado robô parado, que é o comportamento do sistema ao ser inicializado. Neste momento o robô apresenta velocidade nula e deve aguardar até que uma pessoa entre em seu campo de visão, gerando fluxo óptico. O estado reto representa a situação em que o robô se move em linha reta, e são geradas saídas para o controle de velocidade das rodas de acordo com a distância relativa entre alvo e robô. Por fim, o estado girando se dá quando há rotação do robô em torno de seu eixo.

A seguir apresenta-se a tabela de transição de estados para tal máquina:

Tabela 3.1: Transição de estados para o robô

Transição - Estado atual	Parado	Reto	Girando
T1 - A centralizado, D maior que ideal	Reto	-	Reto
T2 - A centralizado, D menor que ideal	-	Parado	Parado
T3 - A centralizado, D ideal	-	Parado	Parado
T4 - A descentralizado, D menor que ideal	Girando	Girando	-

A - Alvo

D - Distância estimada do alvo

A transição T1, exposta na figura 3.4, ocorre quando o alvo está centralizado na tela e sua largura está menor que a ideal, fazendo com que sejam acionadas as rodas com velocidade conforme a equação:

$$v_{esquerda} = v_{direita} = k \cdot (largura_{ideal} - largura)$$

Em que k é uma constante de proporcionalidade determinada experimentalmente, $largura_{ideal}$ é a largura do perfil da pessoa observado em uma distância considerada adequada, e $largura$ é a largura do perfil da pessoa observada no instante considerado. Tal tratamento faz, entretanto, com que o robô mantenha uma distância linearmente dependente da largura da pessoa a sua frente, de maneira que fique mais próximo da pessoa caso a mesma seja mais gorda que o padrão.

A transição T2 ocorre quando o alvo está centralizado e sua largura é maior que a ideal, o que faz com que as rodas sejam paradas. Em T3, o alvo continua centralizado, mas com largura ideal, e novamente há envio de velocidade nulas para as rodas.

Por fim, T4, T5 e T6 ocorrem quando o alvo está descentralizado e tem-se largura do alvo maior que o ideal, sendo, então, a velocidade das rodas calculadas segundo:

$$v_{esquerda} = -v_{direita} = k \cdot (x - x_{centro})$$

Em que$k$ é uma constante de proporcionalidade determinada experimentalmente, x é a posição concentrada do alvo, e x_{centro} é a posição do centro da imagem, equivalente a metade da resolução horizontal. Foi definida também uma zona morta ao redor do centro em que não ocorre definição de velocidade rotacional, para que pequenas oscilações sejam desconsideradas.

3.5.2 Máquina de estados do processamento de imagens

A máquina de estados implementada no código difere da máquina apresentada acima, representativa do comportamento do sistema como um todo.

Tal diferenciação é necessária para que se tenha maior confiabilidade durante o tempo de execução do programa. Na representação dos estados do comportamento do robô segundo a posição da pessoa, tem-se a transição de estados de acordo com a posição da pessoa estimada pelo próprio algoritmo, sendo, portanto, suscetível a erros ou atrasos de execução (devido à inercia de movimento do robô, por exemplo, como mostra a figura 3.3), o que ocasiona erros

no controle e risco de acidentes. Já que cada forma de movimentação do robô faz com que seja necessário um método diferente para a identificação da pessoa na imagemuma identificação incorreta do estado gera resultados inconclusivos. Sendo assim, decidiu-se trabalhar com uma segunda máquina de estados que obtem dados sobre a movimentação do robô diretamente do mesmo, conforme a figura 3.5.

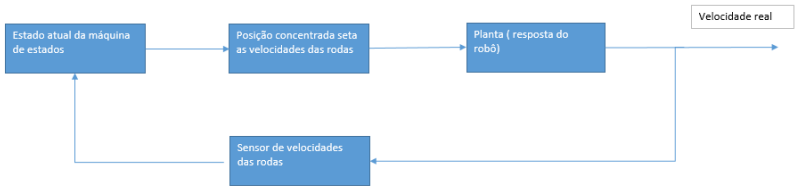


Figura 3.5: Uso da velocidade do robô para o algoritmo

O estado do robô, obtido segundo a leitura da velocidade das rodas do mesmo, configura qual método será usado para a obtenção da posição da pessoa a ser seguida. A posição concentrada e a largura do alvo são informações que geram dados de saída de velocidades que são enviados às rodas do robô, mas que não são imediatamente executadas. Um sistema de retroalimentação, então, resolve esse problema. O robô apresenta rotinas que fornecem os dados atuais de velocidades das rodas que são enviados ao programa e usadas, então, na transição de estados da máquina de estados.

Tabela 3.2: Transição de estados para o processamento de imagens

Estado atual	Parado	Reto	Girando
velocidade translacional nula	-	Parado	Parado
velocidade translacional não nula	Reto	-	Reto
velocidade rotacional não nula	Girando	Girando	-

3.6 Processamento das imagens

Para a forma como decidiu-se trabalhar com as imagens, com o uso de filtros específicos aplicados sobre toda a matriz de fluxos ópticos, faz-se necessário que se tenha informação sobre o fluxo óptico em todos os pontos da imagem, de maneira que não se pode trabalhar somente com atributos. Usou-se, portanto, para o cálculo do fluxo óptico, o algoritmo de Gunnar Farnebäck, que fornece o campo de fluxos ópticos denso e possui implementação pronta na biblioteca OpenCV. Sobre a matriz de fluxos obtida, então, são aplicados uma série de filtros diferentes para cada estado da máquina de estados do processamento de imagens.

3.6.1 Filtros para caso reto

O filtro usado para a extração dos fluxos da pessoa no caso em que o robô está em movimento retilíneo consiste em uma máscara de vetores que crescem de ma-

neira linear radial a partir do zero no centro até as extremidades, representada graficamente na figura 3.6. Para tratar casos em que haja um objeto próximo a um dos lados do robô e nada próximo do outro lado, como, por exemplo, quando o robô se move paralelo e próximo a uma parede, separou-se a imagem em duas metades, a esquerda e a direita, tratadas independentemente. Em cada uma dessas partes é buscado o maior vetor de fluxo, sendo esse, então, definido como o tamanho máximo dos vetores nas extremidades da máscara.

Calculam-se as componentes horizontal e vertical de cada vetor separadamente, conforme segue:

$$m(x) = \frac{2 \cdot x \cdot f_{max,x}}{Res_H} - f_{max,x},\ m(y) = \frac{2 \cdot y \cdot f_{max,y}}{Res_V} - f_{max,y}$$

Em que:

m é o vetor da máscara no ponto calculado;

(x, y) são as coordenadas do ponto, em pixels;

Res_H e Res_V são as componente da resolução usada no cálculo dos fluxos, no caso, 400 e 300, respectivamente;

f_{max} é o maior vetor de fluxo óptico encontrado na mesma metade da imagem em que se encontra o ponto calculado.

Essa máscara representa o que se espera dos fluxos ópticos médios dos objetos imóveis no ambiente, que surgem devido ao movimento do robô. Da matriz de fluxo original, então, subtrai-se tal máscara e desconsidera-se todos os fluxos que possuam sentido invertido por tal subtração. Objetos próximos ao robô durante o movimento fazem com que f_{max} cresça e, assim, o algoritmo é capaz de descartar a maioria dos vetores de fluxo de tais objetos sem qualquer alteração na máscara. O restante dos fluxos, então, é removido com os filtros subsequentes.

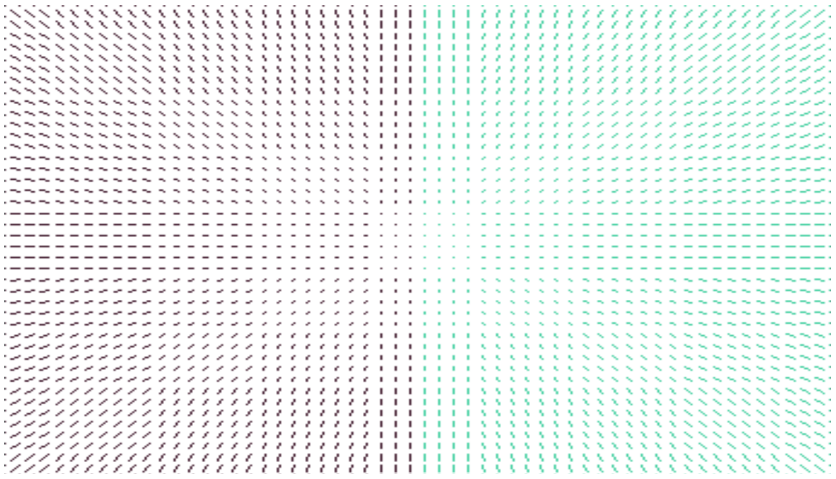


Figura 3.6: Representação gráfica da máscara vetorial

3.6.2 Filtros para caso girando

No caso em que o robô possui apenas movimento rotacional, simplesmente são anulados todos os vetores de fluxos de componente horizontal de sentido contrário ao sentido de rotação. Tal é o sentido que apresenta todo o ambiente, com exceção das leituras realizadas erroneamente devido à distorção causada na imagem pelo movimento rápido demais, que ocorre, por exemplo, no primeiro par de quadros a partir do início da rotação do robô. Devido ao movimento dos pés e das mãos, e por possuir movimento mais rápido que ou equivalente ao do robô, a pessoa apresenta bastantes vetores de fluxos de componente horizontal de sentido condizente com a rotação do robô, mesmo que inverta subitamente a direção de seu movimento, o que a torna diferenciável dos arredores.

3.6.3 Considerações gerais para todos os estados

Em todos os estados de movimento do robô são retirados, antes que se aplique qualquer outro filtro, todos os fluxos de módulo grande ou pequeno demais, que geralmente são provenientes de ruídos anormais na entrada, inconsistências no cálculo dos fluxos ópticos, ou, quando o movimento é rápido ou o objeto está próximo demais, da perda do detalhamento da imagem entre os dois quadros usados para o cálculo. Sem esse cuidado, o filtro usado no estado em que o robô apresenta movimento retilíneo, por exemplo, poderia apresentar, ocasionalmente, um tamanho máximo de vetor de tamanho fora do esperado, o que prejudica a máscara criada. Faz-se tal verificação simplesmente rejeitando qualquer vetor de fluxo encontrado que não satisfaça:

$$threshold < f < threshold + range$$

Após a aplicação do filtro específico em cada estado, retiram-se, também, os pontos identificados que não possuam um número mínimo de outros pontos identificados em sua vizinhança. Tais são, também, causados por ruídos ou perda de detalhamento, assim como pelo movimento de objetos pequenos na cena, já que o movimento da pessoa na cena gera grandes aglomerados de fluxos. Isso é feito rejeitando qualquer vetor de fluxo encontrado que não satisfaça:

$$n_{redondezas} > min_{redondezas}$$

Em uma janela ao redor do ponto de dimensão ($2 \cdot raio_{redondezas} + 1 \times raio_{redondezas} + 1$).

Verificou-se, também, no fluxo óptico gerado pela pessoa, porém não no fluxo gerado pelo movimento do robô, forte predominância da componente horizontal sobre a vertical nos vetores. Criou-se, portanto, um peso horizontal que multiplica a componente horizontal de todos os fluxos antes da aplicação de qualquer filtro, dando assim maior significância a essa informação, da forma:

$$f(x, y) = k \cdot f_x(x, y) + f_y(x, y)$$

O estado em que o robô está parado não requer um filtro específico, sendo a posição aproximada da pessoa obtível somente através dos filtros básicos aqui descritos.

3.6.4 Parâmetros de ajuste dos filtros

Durante o desenvolvimento dos filtros, fez-se necessária a criação de uma diversidade de parâmetros para o ajuste dos mesmos. São esses, então:

threshold : tamanho mínimo de vetor de fluxo aceitável como válido;

range : extensão de tamanho de vetores considerados válidos, a partir do *threshold*;

$raio_{redondezas}$: raio ao redor do ponto considerado como vizinhança para a verificação da quantidade de fluxos próximos;

$min_{redondezas}$: quantidade de fluxos próximos necessária para que o fluxo no ponto seja considerado válido;

k : peso horizontal que aumenta a significância da componente horizontal do fluxo.

Cada estado de movimentação do robô, entretanto, precisa de valores diferentes para cada um desses parâmetros que configura os filtros para que possa funcionar adequadamente. Como os cinco parâmetros são dependentes entre si, seria demasiadamente trabalhosa uma análise matemática que permitisse chegar a um resultado ótimo. Fez-se, portanto, uma rotina que analisa empiricamente uma série de parâmetros aplicados a um vídeo de exemplo, varia automaticamente os cinco parâmetros, um por vez, em uma faixa de valores definida pelo usuário e exporta os resultados desses testes em formato de texto. Para que esses dados possam ser facilmente avaliados, é preciso que o usuário defina manualmente a qual setor da imagem corresponde a pessoa no vídeo de exemplo para que a rotina efetue a contagem de fluxos encontrados dentro e fora dessa área. Pode-se, então, analisar facilmente a qualidade do resultado através de uma planilha com os dados, e volta-se a trabalhar ao redor dos melhores resultados, fazendo, assim, o refinamento, até que se obtenha resultado satisfatório.

Para que tal rotina possua um tempo razoável de execução, entretanto, foi preciso usar vídeos para teste curtos, de cerca de apenas 30 quadros, e, ainda assim, foram necessários dias de execução para a resolução de cada caso, já que ocorre o teste de dezenas de milhares de combinações de parâmetros para cada caso até que uma possa ser considerada satisfatória. Tais vídeos foram gravados usando o próprio robô, já que a alteração de fatores como a posição ou qualidade da câmera levariam a uma série de parâmetros diferentes. Outro importante fator a ser considerado é o ambiente em que o robô se encontra durante o uso. Cada ambiente apresenta uma intensidade de iluminação diferente e, em casos extremos, poderia ser necessária uma nova calibração dos parâmetros.

3.7 Filtro de Kalman

3.7.1 Estimativa do processo

O filtro de Kalman, desenvolvido em 1960 por Rudolph Emil Kalman, é uma solução recursiva para o problema de filtragem linear de dados discretos. Desde então, devido a avanços na computação digital, tal filtro tem sido alvo de extensas pesquisas, assim como amplamente aplicado, particularmente, na área de navegação autônoma ou assistida [35].

O filtro incopora toda informação disponível a seu funcionamento [36]: todas as medidas obtidas, independentemente de sua precisão, são usadas para se obter a estimativa do valor atual das variáveis de interesse, com o uso de conhecimento sobre o sistema e a dinâmica da tomada de dados, da descrição estatística de ruídos do sistema, erros de medida, e incertezas nos modelos dinâmicos, assim como de informações disponíveis sobre as condições iniciais das variáveis de interesse.

Tal filtro busca estimar o estado x_k de um processo analisado em tempo discreto pela seguinte relação linear:

$$x_k = A \cdot x_{k-1} + B \cdot u_{k-1} + w_{k-1} \tag{3.1}$$

Com uma medida z pertencente aos R^m:

$$z_k = H \cdot x_k + v_k$$

Sendo w_k e v_k o ruído do processo e da medição do mesmo, respectivamente, assumidas independentes entre si, brancas e com distribuição de probabilade normal (com média zero e variância Q e R). A matriz A, de dimensão $n \times n$, relaciona o estado no tempo anterior $(k-1)$ com o atual (k); B, de dimensão $n \times 1$, relaciona a entrada de valores de controle no modelo ao estado atual (sendo opcional) e H, de dimensão $m \times n$, relaciona o estado à medida atual, sendo que todos variam seus valores de acordo com o intervalo de tempo em questão.

3.7.2 As origens do filtro

Definem-se erros a priori e a posteriori como sendo:

$$e_k^- = x_k - x\acute{}_k^-$$

$$e_k = x_k - x\acute{}_k$$

Sendo $x\acute{}_k^-$ a estimativa do estado a priori no tempo k, conhecendo-se o processo por meio de suas equações, e $x\acute{}_k$ a estimativa a posteriori do estado no tempo k com a medida z_k.

A partir dos erros, estimam-se as matrizes de covariância como sendo:

$$P_k^- = [e_k^- \cdot e_k^{-T}]$$

$$P_k = [e_k \cdot e_k^T]$$

Por conseguinte, deve-se computar uma estimativa a posteriori para o estado estimado $x\acute{}_k$ como uma combinação linear do estado estimado a priori, $x\acute{}_k^-$, e uma diferença com peso entre a medida atual z_k e uma medida de predição dada por $H \cdot x\acute{}_k^-$, sendo essa diferença denominada resíduo. Esse valor reflete a discrepância entre a medição $H \cdot x\acute{}_k^-$ e a atual medida z_k.

$$x\acute{}_k = x\acute{}_k^- + K \cdot (z_k - H \cdot x\acute{}_k^-)$$

Temos que K é uma matriz $n \times m$ chamada ganho de Kalman, sendo:

$$K_k = P_k^- \cdot H^T \cdot (H \cdot P_k^- \cdot H^T + R)^{-1}$$

Uma interpretação sobre o valor do ganho de Kalman K é que quanto menor o valor de R (erro covariante das medidas), mais confiável é a medida z_k e o filtro tende a seguir este valor.

3.7.3 As origens probabilísticas do filtro

A equação 3.1 é baseada na probabilidade da estimativa a priori, $x\acute{}_k^-$, condicionada às medidas z_k, pela lei de Bayes. Inicialmente tem-se que o filtro de Kalman mantém os dois primeiros momentos do estado de distribuição.

$$E[x_k] = x\acute{}_k$$

$$E[(x_k - x\acute{}_k)(x_k - x\acute{}_k)^T] = P_k$$

A estimativa do estado a posteriori reflete a média, o primeiro momento do estado de distribuição. Já o erro covariante estimado a posteriori reflete a variância do estado de distribuição, como segundo momento.

$$p(x_k/z_k) \sim N(E[x_k], E[(x_k - x\acute{}_k) \cdot (x_k - x\acute{}_k)^T]) = N(x\acute{}_k, P_k)$$

3.7.4 O algoritmo do filtro de Kalman

O filtro de Kalman funciona usando uma forma de retro-alimentação, isto é, o filtro estima o estado do processo num tempo k e depois obtém o feedback na forma de medidas ruidosas. Sendo assim, as equações do filtro são divididas em duas partes: equações de atualização temporal e equações de atualização de medidas. A primeira categoria de equações é responsável por projetar em um tempo a frente, $k+1$, o estado atual e o erro covariante, obtendo-se, assim, a estimativa a priori para o próximo passo. Já a segunda categoria de equações é responsável pelo feedback, melhorando a estimativa a priori e tornando-a uma estimativa a posteriori. Uma outra maneira comum de denominar essas duas divisões é chamá-las de equações de predição e equações de correção.

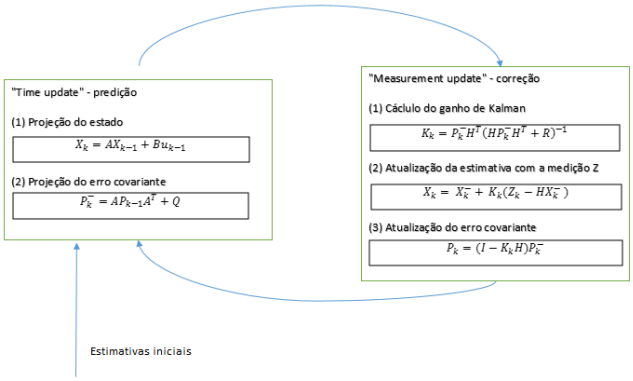


Figura 3.7: Ciclo de funcionamento do filtro de Kalman (adaptado de [4]).

Durante a execução do programa, realiza-se a computação das equações conforme figura acima, inicialmente as equações de predição seguidas pelas equações de correção, de forma iterativa. O interessante do filtro de Kalman é que as matrizes de erro covariante P_m e o ganho de Kalman K_k estabilizam-se rapidamente ao redor do valor final e se mantêm constantes durante a execução do algoritmo. Caso necessário, esses valores podem ser levantados fora da execução real do programa em que o filtro será usado.

A motivação do uso de um filtro como esse se deve à inter-relação entre o que se obtém com dados adquiridos por sensores e o que é real. Toma-se, como exemplo, um robô que se desloca num ambiente e adquire informações de posicionamento em relação à sua posição inicial por meio de um odômetro. Tem-se, então, uma malha de controle fechado, com o comando às rodas para girar e a leitura destas no tempo por meio de um encoder óptico (sensor), representado pelo seguinte diagrama de blocos da figura 3.8.

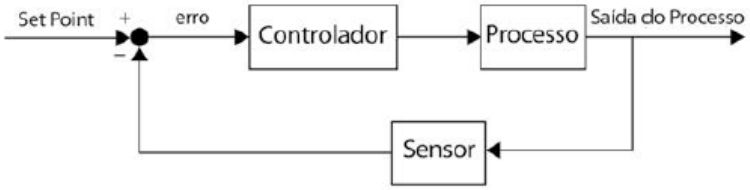
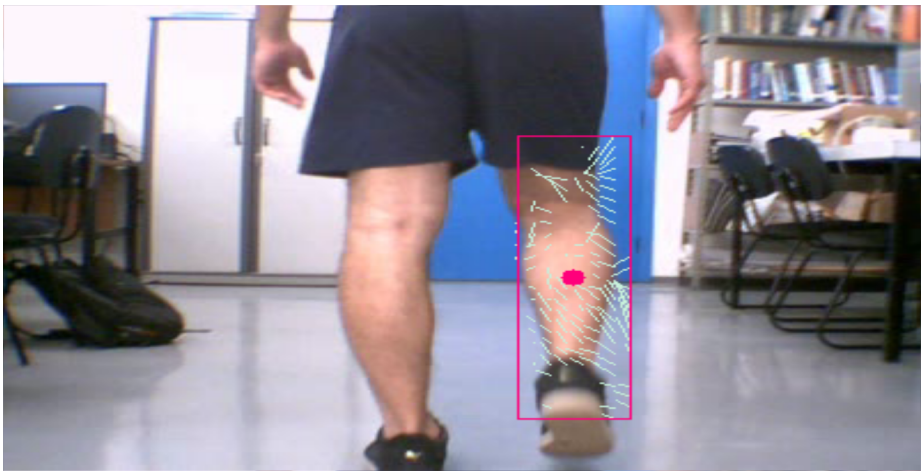


Figura 3.8: Diagrama de blocos para processo genérico com sensoriamento

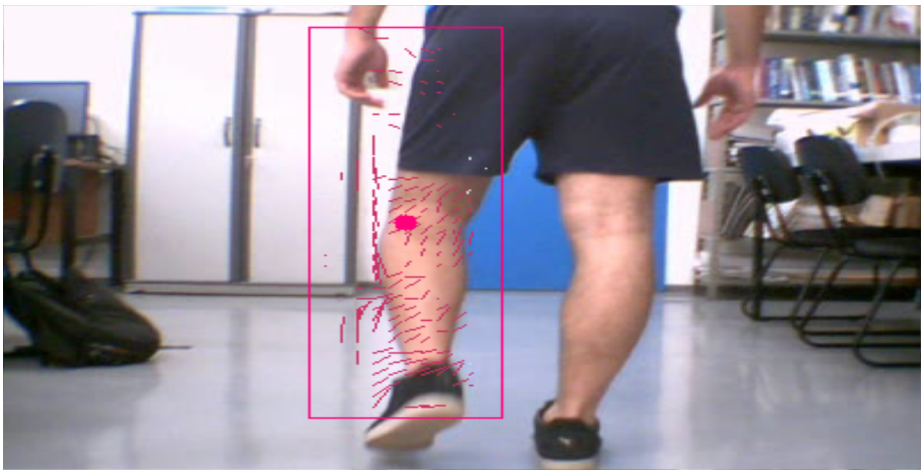
Pela integração da relação $v = w \cdot R$ no tempo , sendo w obtido pelo sensor e R o raio da roda, obtém-se o deslocamento hipotético total do robô. No entanto, essa interpretação matemática do modelo não é, de fato, o modelo real do mundo, é apenas uma representação próxima, mas com limitações, do que realmente ocorre no mundo. Em outras palavras, é imposto ao modelo variações, ditos ruídos (como escorregamento das rodas do robô no solo, leituras errôneas de pulsos no encoder óptico, etc) que modificam o estado atual do mesmo e que devem ser consideradas para se obter uma representação mais aproximada e dados confiáveis. A seguir, são apresentadas as particularidadades do uso do filtro de Kalman neste projeto.

3.7.5 Aplicação do filtro de Kalman neste projeto

Como apresentado anteriormente, o modelo do filtro de Kalman apresenta como entrada o valor da leitura do sensor – tal como um acelerômetro, giroscópio, magnetômetro, etc. Neste projeto, tem-se como saída do processamento de imagens dois pontos extremos de um retângulo que localiza o alvo a ser seguido. No entanto, devido ao padrão de movimento no caminhar dos seres humanos, em certos momentos não se verifica fluxo óptico em regiões do corpo – como as pernas – como se pode ver na figura 3.9.



(a) Apenas a perna direita é identificada



(b) Apenas a perna esquerda é identificada

Figura 3.9: Padrão de fluxos ópticos observado

Com o objetivo de se analisar a saída dos filtros do fluxo óptico, optou-se por calcular o centro geométrico do retângulo de localização e denominá-lo de ponto concentrado do alvo. Como se observa no gráfico da figura 3.10, para a execução do vídeo teste com cerca de 90 quadros no qual o robô segue uma pessoa em linha reta, há bruscas variações da posição concentrada do alvo. Destaca-se que estes dados são o resultado da aplicação de alguns filtros básicos para a extração do alvo (ainda sem a aplicação do filtro de Kalman) e que a resolução de cálculo é de 400 por 300 pixels.

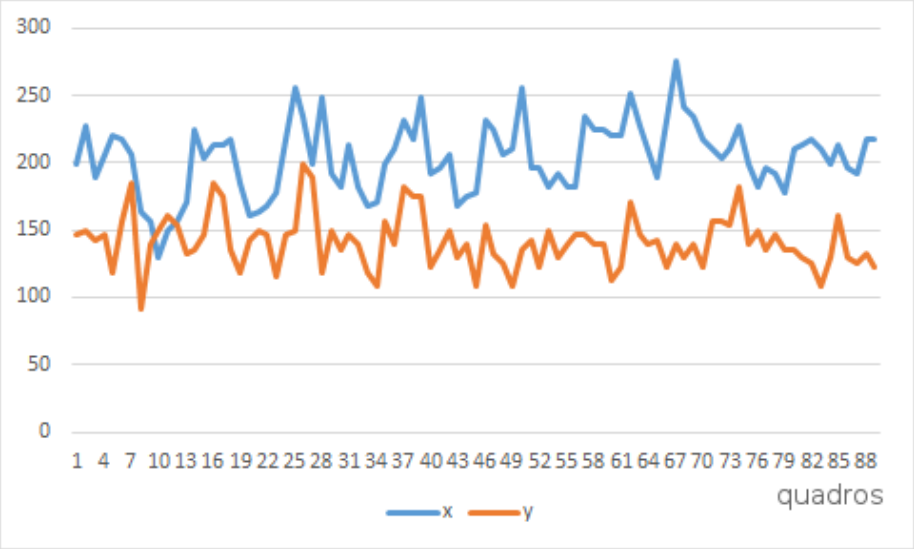


Figura 3.10: Posição concentrada sem a aplicação do filtro de Kalman

Embora os dados obtidos até então sobre a posição concentrada do alvo na imagem sejam confiáveis, deseja-se uma melhoria nesta leitura a fim de se evitar oscilações no movimento do robô. Toma-se, como exemplo, uma situação em que o robô segue um alvo que caminha em linha reta. Em alguns momentos a saída da posição concentrada obtida pelo processamento de imagens oscila ora para a direita, ora para a esquerda. Em situações extremas, o algoritmo poderia interpretar que o alvo estivesse realizando uma curva e gerar uma saída incorreta para as rodas do robô, fazendo-o girar para centralizar o alvo na imagem. O comportamento apresentado no gráfico acima, no qual há variação da leitura da localização concentrada do alvo em torno de seu real posicionamento, é um problema usual no campo da engenharia. O filtro de Kalman foi, assim, o algoritmo escolhido para a melhoria desses dados, gerando uma saída mais suave para o posicionamento do alvo com grande confiabilidade, como será exposto.

3.7.6 O filtro de Kalman e a saída do processamento de imagens

Com a saída de diversos filtros aplicados à imagem obtida pela câmera, obtém-se um retângulo representativo da localização do alvo com o problema descrito no item 3.7.5. Tal retângulo é, então, plotado na imagem, indicando-se a localização do alvo. O ponto superior e o ponto inferior são coordenadas (x, y) de pontos extremos da diagonal deste retângulo e são os dados de entrada para o algoritmo do filtro de Kalman, como apresentado na imagem 3.11.

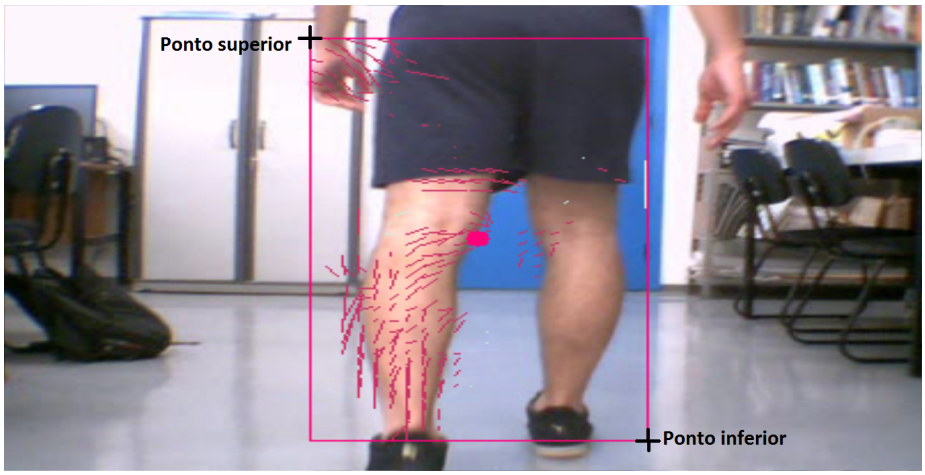


Figura 3.11: Retângulo de localização e os pontos usados para o filtro de Kalman

3.7.7 Modelo adotado

O modelo empregado para a predição é o de um sistema com movimento uniformemente variado, como segue:

$$x_k = x_{k-1} \cdot dt + 0.5 \cdot a \cdot dt^2$$

$$x_k = x_{k-1}$$

$$y_k = y_{k-1} \cdot dt + 0.5 \cdot a \cdot dt^2$$

$$y_k = y_{k-1}$$

Com $a = 0.005 pixels/s^2$ e dt o intervalo de tempo entre dois quadros consecutivos. Sendo assim, temos as seguintes matrizes para o filtro de Kalman:

$$X_k = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ x'_{k-1} \\ y'_{k-1} \end{bmatrix}, \; A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 0.5 \cdot dt^2 \\ 0.5 \cdot dt^2 \\ dt \\ dt \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

$$R = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}, Q = \begin{bmatrix} \frac{dt^4}{4} & 0 & \frac{dt^3}{2} & 0 \\ 0 & \frac{dt^4}{4} & 0 & \frac{dt^3}{2} \\ \frac{dt^3}{2} & 0 & dt^2 & 0 \\ 0 & \frac{dt^3}{2} & 0 & dt^2 \end{bmatrix}, U = a, H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

3.7.8 Análise da variância da leitura

A fim de se obter uma análise quantitativa do resultado da aplicação do filtro de Kalman no problema proposto e entender os efeitos de diferentes valores na variância da leitura desse algoritmo, desenvolveu-se um programa representativo do filtro no Matlab e aplicou-se uma entrada degrau de valores de posição de 70 a 300, conforme a figura 3.12.

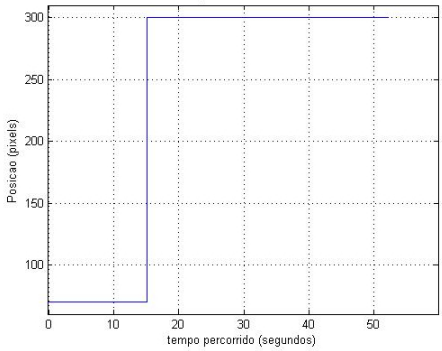


Figura 3.12: Entrada degrau para o filtro de Kalman

Para melhor aproximação da realidade, aplicou-se aos valores lidos um ruído gaussiano, como pode-se observar na figura 3.13.

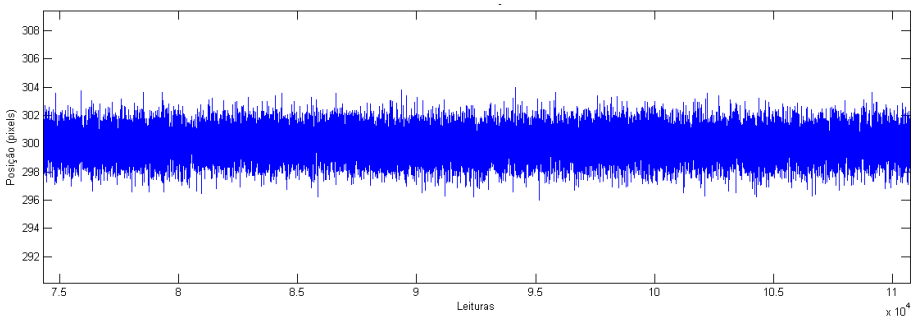
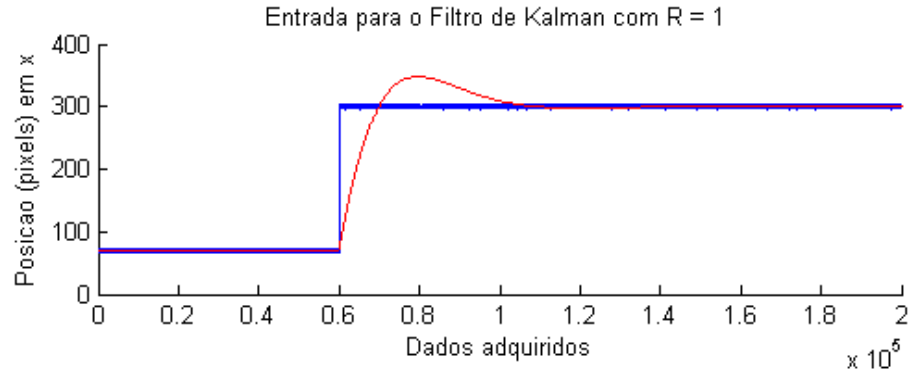
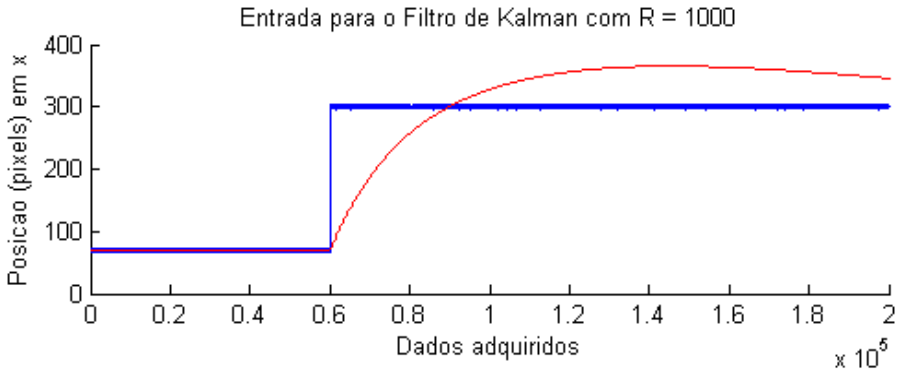


Figura 3.13: Leituras com ruído Gaussiano

Como valores iniciais, tomou-se o valor das variâncias de leitura como $\sigma_x^2 = \sigma_y^2 = 1$ na primeira simulação, e $\sigma_x^2 = \sigma_y^2 = 1000$ na seguinte, obtendo-se os resultados vistos na figura 3.14.



(a) Usando uma variância pequena



(b) Usando uma variância grande

Figura 3.14: Comportamento do filtro de Kalman com diferentes variâncias

Sendo R a matriz da variância da leitura de posicionamento, anteriormente apresentada como a variância dos dados de entrada do filtro de Kalman, nota-se que quanto mais próximos de $\sigma_x^2 = \sigma_y^2 = 1$, como na imagem 3.14a, mais confiáveis são considerados os dados de posicionamento vindos do processamento das imagens, e mais rapidamente o filtro converge. De forma contrária, usando-se, por exemplo, $\sigma_x^2 = \sigma_y^2 = 1000$, como na figura 3.14b, admite-se pouca confiabilidade nos dados adquiridos pela leitura e, portanto, demora a convergência até o valor final.

3.7.8.1 Variância adaptativa para o filtro de Kalman

Adicionalmente, foi proposta a seguinte adaptação ao valor da variância usada no filtro de Kalman: visto que é necessário avaliar em tempo real se os dados obtidos do processamento de imagens são suficientemente confiáveis, já que eles são a entrada para o filtro de Kalman, procurou-se modificar o valor da variância da leitura de acordo algum outro parâmetro para que se tenha um comportamento intermediário dentre os apresentados graficamente.

Em outras palavras, o objetivo é que o filtro confie mais no valor de posicionamento quando o valor anteriormente obtido na leitura for próximo do valor atual – e menos caso contrário. Sendo assim, a cada iteração dos quadros

processados é calculada a distância entre a posição atual do alvo e a posição anteriormente calculada, como segue:

$$distância = \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2}$$

Sendo x_{k-1} e y_{k-1}a posição em coordenadas cartesianas, em pixels, do ponto de localização concentrada do alvo no quadro anterior e x_k e y_k o equivalente para o quadro atual. De acordo com o resultado desse parâmetro, então, é configurado de forma dinâmica o valor da variância conforme o estado do robô, conforme apresentado a seguir.

3.7.8.2 Variância para o estado em movimento reto ou girando

No estado em que o robô se move em linha reta foram simuladas as aplicações de algumas curvas que relacionam o valor da variância de leitura R com a distância entre os pontos da leitura atual e anterior. A fim de se observar o valor da variância das leituras variando de acordo com a distância d, modificou-se a entrada do filtro de Kalman para uma curva sigmóide partindo de 70 pixels e chegando a 300 pixels, conforme o gráfico da figura 3.15, usando-se a equação:

$$x = \frac{1}{1+e^{-1.5\cdot(x-5)}}$$

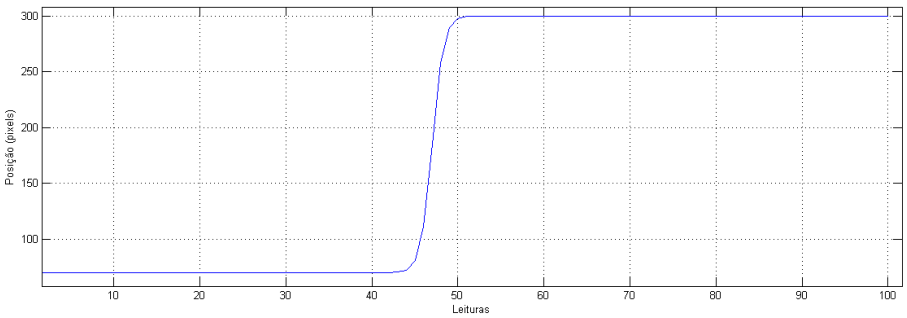


Figura 3.15: Entrada em formato de sigmóide

Para todos os casos a seguir simulados, adotou-se como padrão na obtenção das curvas os seguintes itens:

- Distância mínima (d_{min}) como 1 pixel, já que os dados são discretos;
- Distância máxima (d_{max}) como 500 pixels: já que a resolução de trabalho das imagens capturadas é de 400 por 300 pixels;
- Variância mínima (r_{min}) como 1: caso em que o filtro confia fortemente nos dados lidos;
- Variância máxima (r_{max}) como 500: caso em que há menor confiança nos dados.

Em cada item a seguir, tem-se que o primeiro gráfico apresenta uma curva em azul, que representa os valores lidos pelo sensor e que são as entradas para o filtro, e uma em vermelha, que é a saída do filtro de Kalman. O segundo é o cálculo da distância, em pixels, dos valores atual e anterior da leitura. Observa-se que, como desejado, esta distância apresenta um pico na redondeza da quinquagesima leitura, possibilitando a posterior visualização da variação da variância. O terceiro gráfico é a saída da variância ajustada de acordo com a distância d, anteriormente calculada. Por fim, o quarto gráfico é a curva R em questão adotada.

3.7.8.2.1 R constante

A simulação abaixo foi realizada para $R = 1$, e será usada com propósito comparativo, já que se deseja valores de R variáveis de acordo com a distância obtida.

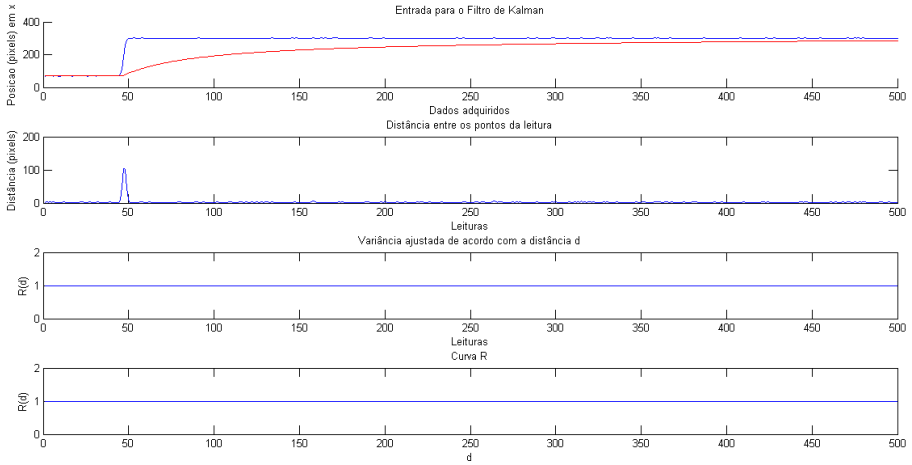


Figura 3.16: Simulação do filtro com variância constante

3.7.8.2.2 R linear

A curva padrão para $R(d)$ adotada é da seguinte forma:

$$R(d) = a \cdot d + b$$

Sendo:

$a = \frac{r_{max} - r_{min}}{d_{max} - d_{min}}$

$b = r_{min} - a \cdot d_{min}$

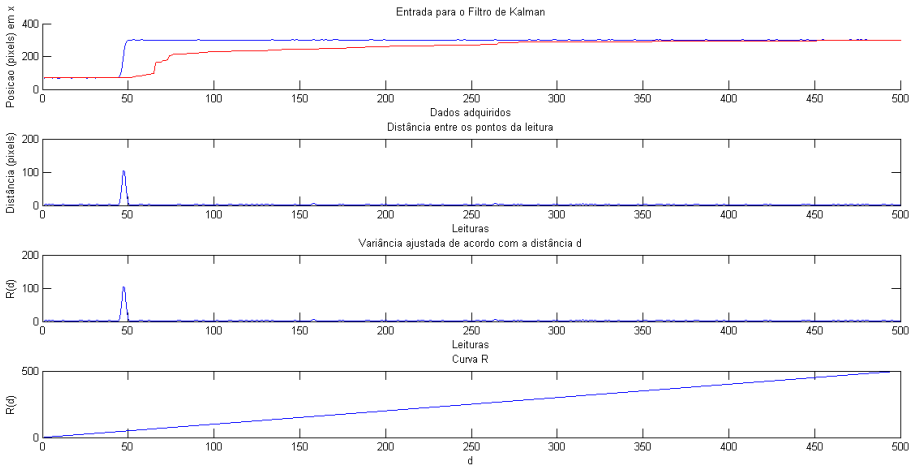


Figura 3.17: Simulação do filtro com variância linear

3.7.8.2.3 R quadrático

Curva adotada do tipo:

$$R(d) = a \cdot d^2 + b \cdot d + c$$

Sendo:

$a = \frac{r_{min} - r_{max}}{-d_{max}^2 - d_{min}^2 + 2 \cdot d_{min} \cdot d_{max}}$

$b = -2 \cdot a \cdot d_{min}$

$c = r_{max} - a \cdot d_{max}^2 - b \cdot d_{max}$

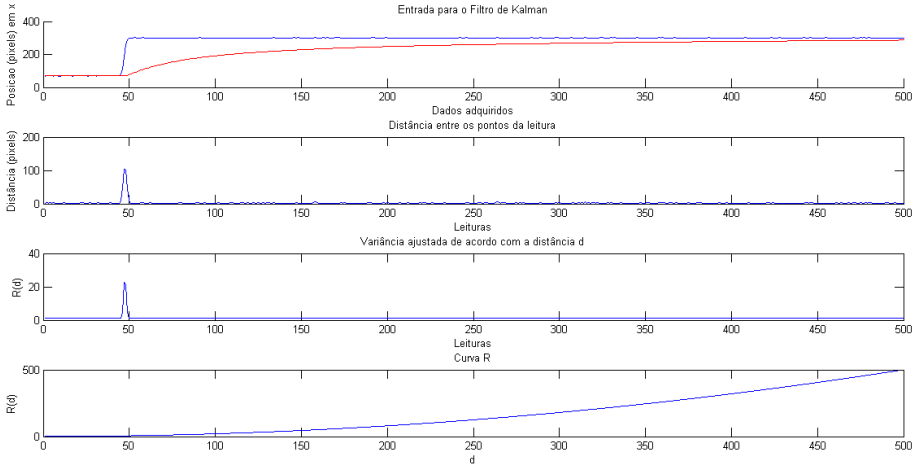


Figura 3.18: Simulação do filtro com variância quadrática

3.7.8.2.4 R exponencial

Curva adotada do tipo:

$$R(d) = 50 + e^{\frac{d}{50}}$$

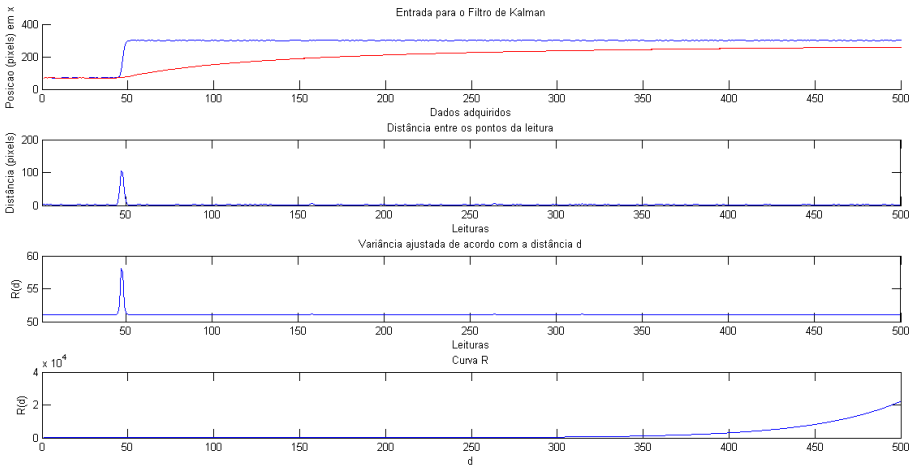


Figura 3.19: Simulação do filtro com variância exponencial

A observação dos gráficos acima demonstra que, com exceção do último caso, não há grandes diferenças entre o tipo de curva utilizado, dada a padronização dos valores r_{max}, r_{min}, d_{max} e d_{min}. Para $R(d)$ exponencial tem-se uma convergência para o valor final da leitura (300 pixels) um pouco mais demorada. Sendo assim, foi adotada a curva linear devido a sua simplicidade e eficiência suficiente.

Destaca-se que as simulações acima efetuadas representam uma situação em que, por exemplo, o alvo (de forma concentrada) a ser seguido pelo robô passou da posição 70 e foi para a posição 300 (em pixels) no eixo x do sistema de referência fixo à tela. Certamente, na prática do uso do filtro, essa situação não ocorrerá com frequência, já que se tem deslocamentos constantes do robô e do alvo, mas foi possível, assim, analisar seu comportamento e obter um maior entendimento do que ocorre.

3.7.8.3 Variância para o estado parado

A estratégia adotada para a variação da variância dos dados vindos das leituras no estado parado foi diferente do estado andando. O importante, nessa situação, é que o filtro se adeque da melhor maneira possível a uma situação com pouca densidade de fluxos em relação ao caso andando, que ignore esporádicos ruídos na imagem que ocorrem mesmo com excelentes valores de parâmetros dos filtros inicias determinados, e que ignore variações do corpo do alvo quando a quantidade de fluxos nela é pequena.

Esse último item se relaciona a um problema observado quando se tem uma pessoa praticamente parada frente a câmera, apenas com as mãos se movimentando, situação comum em que o algoritmo para a variação da variância usado

para os outros estados não traz bons resultados. Nesse cenário observa-se pequena distância entre os pontos concentrados em vários quadros consecutivos, o que resulta em pequenos valores da variância de leitura e rápida convergência do filtro de Kalman para as mãos em movimento, trazendo pouca confiabilidade no resultado global de localização do alvo por inteiro. Uma metodologia alternativa foi, então, implementada para o presente estado, em que, para cada quadro analisado, após a aplicação dos filtros para os fluxos, analisa-se a quantidade de fluxos encontrados em relação ao total possível, e ajusta-se de maneira discreta o valor da variância de leitura R correspondente, de acordo com a tabela **??**. Sendo assim, caso o número de fluxos esteja dentro de uma porcentagem do valor de número de fluxos máximo da tela, σ_x^2 e σ_y^2 recebem o valor de 1, 500, ou 1000.

Tabela 3.3: Variância no estado parado conforme porcentagem de fluxos no quadro atual

Porcentagem de fluxos	R
$0 \rightarrow 10\%$	1000
$10 \rightarrow 70\%$	1
$70 \rightarrow 100\%$	500

Esses valores foram levantados experimentalmente, e obteve-se sucesso com seu uso. Embora a pessoa movimente apenas as mãos por alguns instantes, o retângulo de localização pouco converge, assegurando confiabilidade na localização do alvo inteiro.

3.7.9 Resultados do filtro de Kalman

Uma particularidade na implementação do filtro de Kalman foi necessária para esse projeto: caso não haja fluxo óptico num quadro analisado, a entrada para o filtro fica determinada para uma posição inicial: a região central da tela. Com isto, ocorrendo falha na identificação de fluxo óptico em alguns quadros consecutivos, o que não é comum acontecer, o filtro gera uma saída que tende a levar o robô a um estado de segurança – já que este tende a parar seu movimento – e evita acidentes com pessoas e obstáculos no ambiente.

Outra característica deste projeto que merece atenção é o resultado obtido pela aplicação da máscara diferencial no caso robô andando reto: sua aplicação, juntamente com os filtros básicos, resulta em fluxos ópticos que, por si só, não são suficientes para a localização precisa do alvo – hora apenas uma perna em movimento é identificada, hora ambas. No entanto, com a configuração do filtro de Kalman, como apresentada anteriormente, pôde-se trabalhar satisfatóriamente com esse resultado, de maneira a obter a posição com robustez.

A seguir, são apresentadas as variações da implementação em cada estado do robô.

1. Estado robô em movimento reto

   Dados os resultados simulados no ambiente do Matlab acima exposto, implementou-se no programa de controle do robô o algoritmo com os parâmetros escolhidos.

Para o teste, então, do resultado na prática, usou-se como entrada do programa um vídeo em que o robô se movimenta em velocidade constante e igual à de uma pessoa caminhando a sua frente, na distância considerada ideal, e plotou-se a localização do alvo concentrado nas direções x e y, com e sem o uso do filtro de Kalman. O resultado pode ser visto na figura 3.20

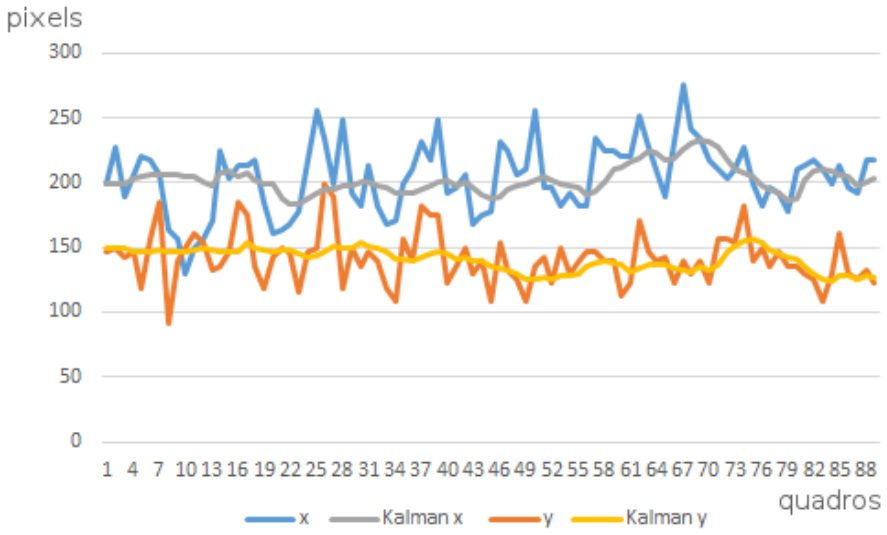


Figura 3.20: Posição concentrada da pessoa com e sem a aplicação do filtro de Kalman

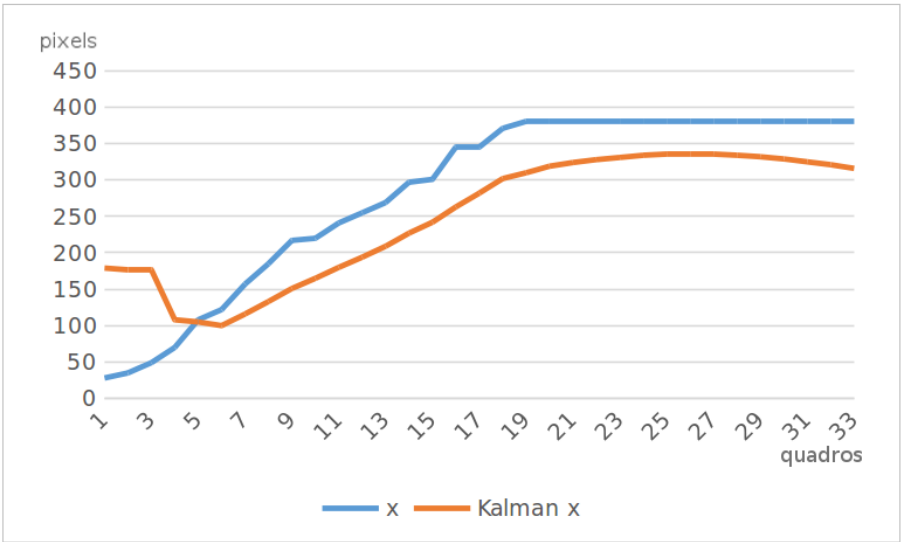
A melhoria na qualidade da saída de localização do alvo foi notória, trazendo grandes benefícios ao controle do movimento do robô, evitando-se falsos dados de localização devido à ruídos na imagem. Vale destacar que a adaptação sobre a variância da leitura do filtro funcionou muito bem. Nota-se no gráfico acima que, quando o alvo concentrado varia muito, situação representada por grandes picos no gráfico, a saída do filtro não a acompanha, pois tratam-se de dados não muito confiáveis. De forma contrária, quando se tem pouca variação, o filtro gera uma saída mais próxima destes valores. Na prática, o resultado da aplicação do filtro de Kalman foi satisfatório e trouxe grande benefício aos resultados dos algoritmos implementados.

2. Estado robô girando

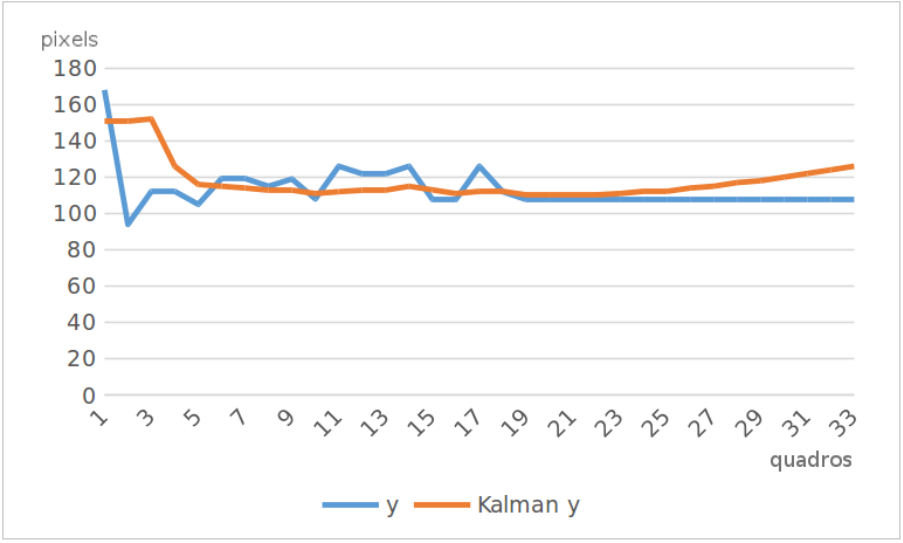
   No estado do robô girando foram usados os mesmos algoritmos para determinação da variância que o estado reto. O funcionamento se mostra eficiente e robusto: é aplicada a máscara horizontal e, se há fluxo restante, é devido às pernas da pessoa na cena, e a saída do filtro tende à pessoa; quando não há, o filtro de Kalman tende à posição central e o robô tende a parar, passando para o estado parado, no qual readquire a posição da pessoa com facilidade.

3. Estado robô parado

Usou-se como entrada um vídeo usado para testes no qual o robô se mantém parado enquanto uma pessoa entra na cena pela lateral esquerda, para no centro, e retoma o movimento para saída do lado direito da tela. Com o uso da variância conforme descrito anteriormente, tem-se o resultado mostrado na figura 3.21.



(a) Posição horizontal



(b) Posição vertical

Figura 3.21: Posição concentrada da pessoa com e sem a aplicação do filtro de Kalman

Observa-se que na componente vertical há pouquíssima variação do posicionamento concentrado do alvo, já que o movimento é predominantemente horizon-

tal. O filtro de Kalman suavizou o movimento que apresenta algumas oscilações de pequena amplitude , de cerca de 20 pixels. Já na componente horizontal, tem-se uma taxa de crescimento constante quanto ao posicionamento, até que o alvo saia da área de visão da câmera. A partir desse instante, a saída do filtro de Kalman tende a se deslocar para o centro da tela, pois não existe fluxo óptico e o sistema tende ao posicionamento de segurança. É interessante notar que a saída do Kalman, representada pelas curvas de cor laranja, começa na posição (180,150) e, quando notada a existência de fluxo óptico, converge rapidamente para o alvo. Esse fato demonstra que os valores da variância são satisfatórios e que se localiza com rapidez e robustez a pessoa seguida.

3.8 Sensores auxiliares

Não é suficiente a precisão para que se possa localizar a pessoa caso essa não possua movimento significativo, principalmente quando o robô se movimenta ou quando está próximo demais da mesma, já que não há como discernir partes específicas da pessoa. Foi necessário usar, portanto, dados de sensores alternativos para definir que se encontra próximo da distância especificada da pessoa quando a mesma não apresenta mais movimento: usou-se, inicialmente, a menor distância obtida através dos dados dos sonares frontais. Entretanto, como os mesmos apresentam uma leitura demasiadamente discretizada das distâncias, representativas apenas da distância na direção para que aponta cada sonar, há intervalos em que não se consegue uma leitura adequada, como quando as pernas da pessoa buscada estão entre duas direções analisadas. Fez-se o código para a obtenção através do laser da distância mínima na região frontal do robô, entretanto, provavelmente devido a algum mal contato ou defeito na unidade do laser, não se obteve sucesso na obtenção de tal dado, e segue-se apenas com o uso dos sonares, o que faz com que, ocasionalmente, o robô se aproxime demais da pessoa.

3.9 Saídas gráficas

Durante a execução do programa, saídas gráficas são exibidas na tela do notebook para que se possa avaliar o resultado dos algoritmos, conforme a figura 3.22.

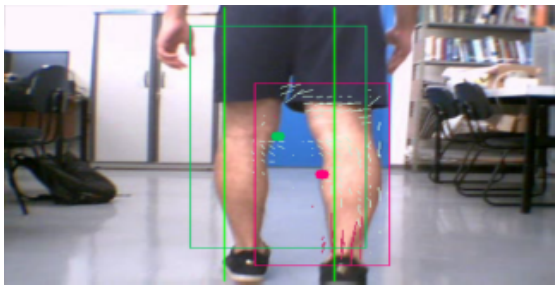


Figura 3.22: Imagem exibida durante a execução do programa

Pode-se verificar facilmente, dessa forma, algumas informações geradas pelo

programa:

- Barras verdes verticais: dividem a imagem em duas regiões laterias, e a região central, sendo usadas com o objetivo de comparar a localização concentrada do alvo a ser seguido. Representam, também, a margem de tolerância angular para que o robô gire. Essas barras estão nas posições $x = 160$ e $x = 240$ pixels, determinadas empiricamente.

- Linhas verdes ou vermelhas espalhadas pela imagem: representam o fluxo óptico após aplicação dos filtros, com exceção do filtro de Kalman, aplicado na imagem inteira. Esses traços são predominantes no alvo seguido, podendo, ocasionalmente, aparecer no ambiente devido aos ruídos, e cada cor representa um sentido da componente horizontal do fluxo (vermelho para a esquerda e branco para a direita).

- Retângulo vermelho: engloba todos os fluxos ópticos desenhados na tela, e serve como uma estimativa inicial do posicionamento do alvo. Este retângulo serve como entrada para o algoritmo do Filtro de Kalman, que atenua as variações desse posicionamento.

- Ponto vermelho: representa a posição concentrada do alvo. O cálculo do posicionamento desse círculo é uma simples média aritmética dos componentes horizontal e vertical do retângulo vermelho.

- Retângulo verde: é o resultado do Filtro de Kalman.

- Ponto verde: posição concentrada obtida a partir do retângulo verde.

4

Conclusão

Foi desenvolvido, neste projeto, a programação de um robô comercial para seguir uma pessoa em diversos ambientes através do processamento de imagens adquiridas por uma câmera embarcada em plataforma móvel. O algoritmo implementado para extrair o fluxo óptico de uma sequência de imagens foi o de Gunnar Farnebäck, e diversos filtros foram implementados para a extração do alvo na imagem. Adicionalmente, o algoritmo do filtro de Kalman, com modificações no modo de calcular a variância do posicionamento do alvo lido, foi aplicado para a melhoria dos resultados obtidos. Sensores auxiliares, já instalados no robô, como o sonar e o laser, foram usados com o objetivo de manter assegurada a distância mínima entre o robô e a pessoa durante os movimentos, assim como para evitar acidentes. Apresentou-se, também, o estado da arte do fluxo óptico.

Os resultados do projeto são um robô capaz de seguir uma pessoa em ambientes variados, mantendo uma distância mínima, por meio do processamento das imagens com o uso de fluxo óptico e auxílio de sensores. É necessário, entretanto, que a pessoa se mantenha em movimento constante, gerando fluxo óptico com as pernas, por exemplo, sem sair do campo de visão do robô para que o mesmo possa localizá-la com robustez. Um resultado bastante expressivo deste projeto é a constatação de que é possível a extração da posição de um alvo em movimento através do uso de fluxo óptico mesmo que a plataforma na qual se dá a aquisição de imagens esteja também em movimento. Com o auxílio de algoritmos, denominados máscaras, pôde-se extrair o alvo com uma precisão tal que, juntamente com o auxílio do filtro de Kalman, pôde-se obter a localização do alvo.

Referências Bibliográficas

[1] HORN, B.; SCHUNCK, B. Determining optical flow. *1981 Technical ...*, 1981. Disponível em: <http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=1231385>.

[2] WWW.MATHWORKS.COM/HELP/VISION/REF/VISION.POINTTRACKERCLASS.HTML.

[3] ADEPT. *MobileRobots. Specification Manual Pionner 3AT*. Disponível em: <http://www.mobilerobots.com/Libraries/Downloads/Pioneer3AT-P3AT-RevA.sflb.ashx>.

[4] WELCH, G.; BISHOP, G.; HILL, C. An Introduction to the Kalman Filter. p. 1–16, 2006.

[5] LEE, Y. et al. Hostile intent and behaviour detection in elevators. 2011. Disponível em: <http://digital-library.theiet.org/content/conferences/10.1049/ic.2011.0115>.

[6] PONTIL, M.; VERRI, a. Support vector machines for 3D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 20, n. 6, p. 637–646, jun. 1998. ISSN 01628828. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=683777>.

[7] ROOBAERT, D.; HULLE, M. V. View-based 3d object recognition with support vector machines. *... IX, 1999. Proceedings of the 1999 ...*, p. 77–84, 1999. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=788125>.

[8] LI, S. . K. L. C. Face recognition by support vector machines. *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, IEEE Comput. Soc, p. 196–201, 2000. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=840634>.

[9] SCHLEGEL, C. et al. Vision Based Person Tracking with a Mobile Robot. *Procedings of the British Machine Vision Conference 1998*, British Machine Vision Association, v. 3, p. 42.1–42.10, 1998. Disponível em: <http://www.bmva.org/bmvc/1998/papers/d045/h045.htm>.

[10] MOTOKUCHO, T.; ODA, N. Vision-based human-following control using optical flow field for power assisted wheelchair. *2014 IEEE 13th International Workshop on Advanced Motion Control (AMC)*, Ieee, p. 266–271, mar. 2014. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6823293>.

[11] SCARAMUZZA, D.; FRAUNDORFER, F.; SIEGWART, R. Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC. In: *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009. p. 4293–4299. ISBN 978-1-4244-2788-8. ISSN 1050-4729. Disponível em: <http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5152255>.

[12] LIPTON, A.; FUJIYOSHI, H.; PATIL, R. Moving target classification and tracking from real-time video. In: *Proceedings Fourth IEEE Workshop on Applications of Computer Vision. WACV'98 (Cat. No.98EX201)*. [S.l.]: IEEE Comput. Soc, 1998. p. 8–14. ISBN 0-8186-8606-5.

[13] BAUMBERG, A.; HOGG, D. *Learning flexible models from image sequences*. [s.n.], 1994. Disponível em: <http://link.springer.com/chapter/10.1007/3-540-57956-7_34>.

[14] ZHAO, L. Dressed human modeling, detection, and parts localization. n. 62, 2001.

[15] SOTELO, M.; PARRA, I. Pedestrian detection using SVM and multi-feature combination. *ITSC'06. IEEE*, p. 103–108, 2006. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1706726>.

[16] BRAUN, T.; SZENTPETERY, K.; BERNS, K. Detecting and following humans with a mobile robot. In: *EOS*. [S.l.: s.n.], 2005.

[17] PICCARDI, M. Background subtraction techniques: a review. In: *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*. IEEE, 2004. v. 4, p. 3099–3104. ISBN 0-7803-8567-5. ISSN 1062-922X. Disponível em: <http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=1400815>.

[18] LIEM, M.; VISSER, A.; GROEN, F. A hybrid algorithm for tracking and following people using a robotic dog. *international conference on Human robot*, ACM Press, p. 1–8, 2008.

[19] KOBILAROV, M. et al. People tracking and following with mobile robot using an omnidirectional camera and a laser. *IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, Ieee, n. May, p. 557–562, 2006. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1641769>.

[20] PIAGGIO, M. et al. An optical-flow person following behaviour. *Proceedings of the 1998 IEEE International Symposium on Intelligent Control (ISIC) held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA) Intelligent Systems and Semiotics (ISAS) (Cat. No.98CH36262)*, Ieee, p. 301–306, 1998. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=713678>.

[21] HANDA, A. et al. Person following with a mobile robot using a modified optical flow. v. 2008, n. September, 2008.

[22] YAMANE, T.; SHIRAI, Y.; MIURA, J. Person tracking by integrating optical flow and uniform brightness regions. In: *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*.

IEEE, 1998. v. 4, p. 3267–3272. ISBN 0-7803-4300-X. ISSN 1050-4729. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=680942>.

[23] NOURANI-VATANI, N.; BORGES, P.; ROBERTS, J. A study of feature extraction algorithms for optical flow tracking. *Australasian Conference on Robotics and Automation*, p. 3–5, 2012. Disponível em: <http://www.araa.asn.au/acra/acra2012/papers/pap105.pdf>.

[24] YIN, B. et al. Indirect human activity recognition based on optical flow method. In: *2012 5th International Congress on Image and Signal Processing*. IEEE, 2012. p. 99–103. ISBN 978-1-4673-0964-6. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6469639>.

[25] CAMPBELL, J. et al. A Robust Visual Odometry and Precipice Detection System Using Consumer-grade Monocular Vision. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005. p. 3421–3427. ISBN 0-7803-8914-X. Disponível em: <http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=1570639>.

[26] CORKE, P.; STRELOW, D.; SINGH, S. Omnidirectional visual odometry for a planetary rover. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2004. v. 4, p. 4007–4012. ISBN 0-7803-8463-6. Disponível em: <http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=1390041>.

[27] HARRIS, B.; STEPHENS, M. A Combined Corner and Edge Detector. In: *Proceedings of the 4th Alvey Vision Conference*. [S.l.: s.n.], 1988. p. 147–151.

[28] SHI, J.; TOMASI, C. Good features to track. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*. IEEE Comput. Soc. Press, 1994. p. 593–600. ISBN 0-8186-5825-8. ISSN 1063-6919. Disponível em: <http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=323794>.

[29] LUCAS, B. D.; KANADE, T. An iterative image registration technique with an application to stereo vision. *IJCAI*, 1981. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6469639 http://www.ic.unicamp.br/ rocha/teaching/2013s1/mc851/aulas/additional-material-lucas-kanade-tracker.pdf>.

[30] BAKER, S. et al. A Database and Evaluation Methodology for Optical Flow. *International Journal of Computer Vision*, v. 92, n. 1, p. 1–31, nov. 2010. ISSN 0920-5691. Disponível em: <http://link.springer.com/10.1007/s11263-010-0390-2>.

[31] MOLNAR, J. *Variational Methods in Machine Vision*. Tese (Doutorado), 2011. Disponível em: <www.tnkcs.inf.elte.hu/vedes/Molnar_Jozsef_Tezisek_en.pdf>.

[32] HORN, B. K. P.; SCHUNCK, B. G. Determining optical flow: a retrospective. *Artificial Intelligence*, v. 59, n. 1-2, p. 81–87, 1993. ISSN 00043702. Disponível em: <http://www.sciencedirect.com/science/article/pii/0004370293901739>.

[33] FARNEBäCK, G. Two-frame motion estimation based on polynomial expansion. *Image Analysis*, n. 1, p. 363–370, 2003. Disponível em: <http://link.springer.com/chapter/10.1007/3-540-45103-X_50>.

[34] AKSOY, S.; YE, M.; SCHAUF, M. Algorithm performance contest. *Pattern Recognition, ...*, p. 870–876, 2000. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=903054>.

[35] WELCH, G. F. HISTORY: The Use of the Kalman Filter for Human Motion Tracking in Virtual Reality. *Presence: Teleoperators and Virtual Environments*, The MIT Press, v. 18, n. 1, p. 72–91, fev. 2009. ISSN 1054-7460.

[36] MAYBECK, P. S. Stochastic models, estimation, and control. v. 1, 1979.