# Rastreamento Dinâmico de Objetos: Um Experimento Didático Integrando Conceitos de Hardware e Software

**Emerson G. Moretto**[2] **, Markus Reichel**[2] **, Hemerson Pistori**[1]

[1] Grupo de Pesquisa em Engenharia e Computação (GPEC)
Universidade Católica Dom Bosco
Caixa Postal 100, Av. Tamandaré, 6000
79117-900 Campo Grande, MS, Brasil

[2]Laboratório de Sistemas Integráveis (LSI)
Escola Politécnica - Universidade de São Paulo
Av. Prof. Luciano Gualberto Trav.3 N.158.
05508-900, São Paulo, SP, Brasil

emoretto,markus@lsi.usp.br,pistori@ucdb.br

***Abstract.*** *This paper presents a robotic vision prototype, that is able to track, circular shaped, moving objects, in real-time. The prototype, built from recycled material and free-software, serves as a non-expensive, educational tool, providing an interdisciplinary environment, that merges concepts from robotics, digital image processing, computational vision and electronics, and motivates computer engineering undergraduate students.*

***Resumo.*** *Este artigo apresenta um protótipo de sistema de visão robótica capaz de rastrear, em tempo real, determinados tipos de objetos circulares. O sistema, que possui finalidades didáticas, foi construído a partir de componentes eletro-mecânicos "reciclados" e bibliotecas de software livre. O produto final, de baixo custo, mostrou-se bastante eficaz na motivação de acadêmicos de um curso de Engenharia de Computação, para as áreas de robótica, processamento digital de sinais, visão computacional e eletrônica.*

## 1. Introdução

Novas tecnologias baseadas em visão computacional vêm sendo exploradas para solução de problemas em áreas tão diversas quanto o diagnóstico médico, a interação homem-máquina e a automação industrial [Forsyth and Ponce 2003]. Entre as diversas aplicações já desenvolvidas, ou em desenvolvimento, destacamos a detecção automática do comportamento de animais para testes de novos fármacos [Spink et al. 2001] e a interpretação de sinais realizados através dos olhos, e outras partes do corpo humano, para interação homem-máquina de pessoas com necessidades especiais [Wang and Sung 2001]. A visão computacional é também uma componente importante na robótica, para construção de sistemas capazes de interpretar e reagir a sinais obtidos por dispositivos de captura de imagens. Esse tipo de habilidade pode ser útil, por exemplo, na execução de tarefas que exigem reconhecimento e interação com seres-humanos, rastreamento e acompanhamento de objetos em movimentos e deslocamento através de corredores e ruas [Souza and Kak 2002].

A robótica é uma área que costuma despertar muito interesse em acadêmicos de cursos na área da computação. Muitas vezes, no entanto, o custo dos equipamentos necessários para realização de experimentos nessa área inviabiliza sua ampla utilização, como recurso pedagógico, em cursos de graduação. Este artigo apresenta um estudo de caso em que materiais de baixo custo, e software livre e gratuito, foram utilizados para construir um protótipo de um sistema robótico de visão computacional. Nesse protótipo, um modelo simples de $webcam$, acoplado a dois motores de passo reciclados, e a um computador pessoal, é capaz de rastrear um CD (*Compact Disc*) em movimento.

A construção do protótipo permite ao aluno a manipulação integrada de conceitos de hardware e software. O algoritmo utilizado para detecção da posição do objeto, circular, é baseado em transformadas de Hough [Toennies et al. 1998]. A próxima seção trata de transformadas de Hough e da implementação do detector de circunferências utilizado nesse trabalho, o Hough-Circles [Pistori et al. 2005]. O protótipo do sistema robótico de visão computacional é detalhado na Seção 3, juntamente com a descrição das componentes de hardware e software. As conclusões e sugestões para trabalhos futuros estão na Seção 4.

## 2. Transformadas de Hough

O problema da detecção de circunferências com raio fixo consiste em determinar quais os pontos de uma imagem pertencem a uma mesma circunferência de raio $r$. Ou seja, tem-se um conjunto de coordenadas $(x, y)$ e pretende-se encontrar valores possíveis para os parâmetros $(x_c, y_c)$, correspondentes aos pontos centrais de circunferências. Para isso, constroi-se um *espaço de Hough* [Hough 1962], que pode ser visto como uma matriz, com a mesma dimensão da imagem digital, em que as colunas e linhas representam, respectivamente, os possíveis valores de $x_c$ e $y_c$. Cada célula dessa matriz recebe, inicialmente, o valor zero, e para cada ponto $(x, y)$ da imagem, incrementa-se no espaço de Hough, todas as células $(x_c, y_c)$ representando centros de circunferências, de raio $r$, que passam por $(x, y)$. Ao final, as células contendo os valores mais altos indicarão os centros "mais prováveis" de circunferências.

**Algoritmo 1** Criação do Espaço de Hough

**entrada:** Matriz I, $n \times m$, representando a imagem binarizada.
**saída:** Matriz H, com o mesmo tamanho da imagem, representando o espaço de Hough.

```
1:  para x = 0 até n faça
2:    para y = 0 até m faça
3:      se I(x, y) = 255 então
4:        para θ = 0 até 2 * π faça
5:          x_c = x − r * cos(θ)
6:          y_c = y − r * sen(θ)
7:          H(x_c, y_c) = H(x_c, y_c) + 1
8:        fim para
9:      fim se
10:   fim para
11: fim para
```

O algoritmo 1 mostra como um espaço de Hough, $H$, pode ser criado a partir de

uma imagem digital, $I$. Depois que o espaço é criado, a detecção de circunferências passa a ser um problema simples de se encontrar pontos de máximo no espaço de Hough. É importante notar que o conceito de transformada de Hough aplica-se somente quando é possível se distinguir, na imagem original, os pontos pertencentes ao contorno, ou borda, dos objetos. O algoritmo 1 assume então que a imagem é previamente processada através de um filtro de detecção de borda e de um filtro de binarização (que "marca" os pixels pertecentes a bordas com o valor 255, e todos os outros com o valor 0).

A Figura 1 ilustra o cálculo do espaço de Hough para uma imagem contendo três objetos. O espaço foi calculado para um valor de raio igual ao raio da menor entre as duas circunferências da imagem à esquerda. A imagem à direita apresenta o espaço de Hough sem ajuste de contraste, e destaca bem a relação entre o valor máximo no espaço de Hough e o centro da circunferência que se desejava detectar. Essa técnica foi implementada através de um módulo para o ImageJ, um software livre para apoio ao desenvolvimento de sistemas de processamento de imagens digitais [Pistori et al. 2005]. A implementação, que recebeu o nome de *Hough-Circles* [1], permite ao usuário escolher o raio da circunferência, a quantidade de circunferências a serem detectadas ou um limiar a ser utilizado na determinação dos pontos do espaço de Hough correspondentes aos centros de circunferências.

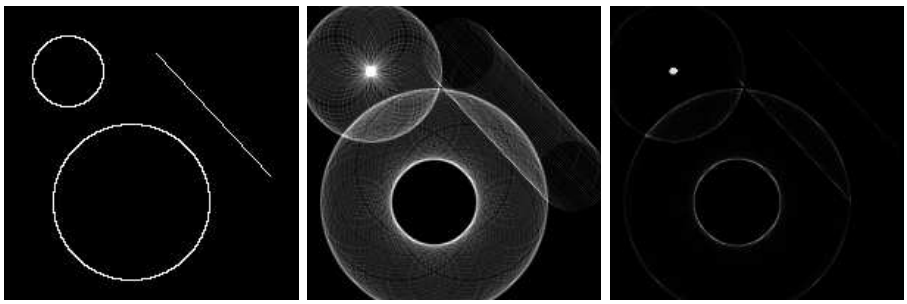


**Figura 1. Exemplo de uma imagem (esquerda) com três objetos, seu espaço de Hough com ajuste de contraste (centro) e sem ajuste de contraste (direita)**

## 3. Protótipo do Sistema de Visão Computacional

O protótipo de sistema de visão computacional implementado utilizou-se do *Hough-Circles* e de outros módulos do pacote ImageJ. Uma $webcam$ montada sobre motores de passo, envia imagens a um microcomputador. O software instalado no microcomputador detecta o centro do objeto circular, com raio pré-definido, que mais se destaca na imagem e calcula como os motores devem ser acionados para fazer com que o objeto permaneça sempre no centro do campo de visão da $webcam$. O processo, ilustrado na Figura 2, é repetido para cada quadro ($frame$) capturado, a uma taxa média de 5 quadros/s. As próximas seções detalham alguns aspectos do hardware e do software desenvolvido.

### 3.1. Componentes de Hardware

No hardware desenvolvido utilizou-se dois motores de passo para a movimentação da *webcam*, sendo um responsável pela movimentação horizontal e outro pela movimentação

[1] O Hough-circles pode ser obtido em http://rsb.info.nih.gov/ij/plugins/hough-circles.html

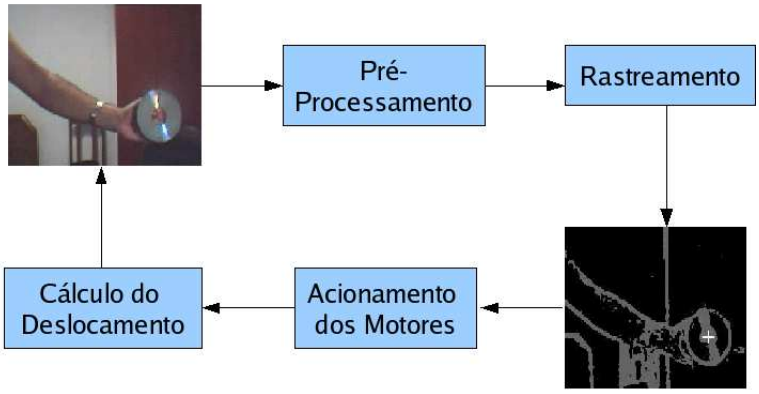



**Figura 2. Sistema de Visão Computacional**

vertical. O motor responsável pela movimentação do eixo vertical se encontra fixado perpendicularmente ao outro motor, que é responsável pela movimentação no eixo horizontal, onde a *webcam* é fixada.

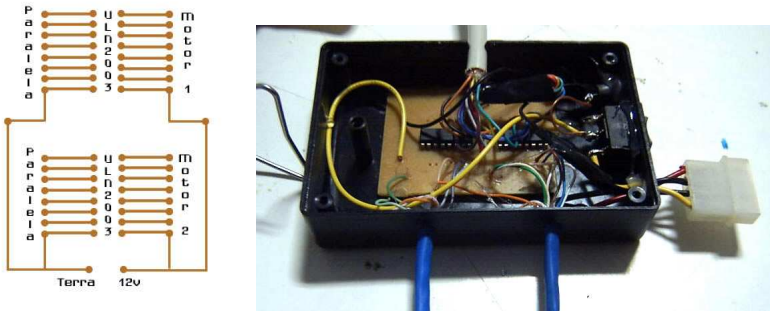
De acordo com a disposição dos motores de passo, a *webcam* obteve uma angulação máxima de 90 graus para o eixo horizontal e 180 no eixo vertical. Em virtude do cabo de ligação da *webcam* com o computador, a angulação obtida no eixo horizontal ficou limitada, pois o motor de passo utilizado na movimentação no eixo horizontal possui um baixo torque. O motor responsável pela movimentação do eixo horizontal possui um maior torque para suportar o segundo motor e a *webcam*, conforme ilustrado na Figura 3.

**Figura 3. Angulações da câmera**

Ambos os motores são alimentados com 12 volts, no entanto, a porta paralela do computador proporciona apenas 5 volts. Para isto, foi necessário o uso de um circuito integrado (ULN2003) que, de uma fonte de alimentação externa, ativa os 12 volts necessários, a partir do sinal da porta paralela. No sistema apresentado, foram utilizados dois desses circuitos integrados, um para cada motor, conforme ilustrado no esquema eletrônico da Figura 4.(a).

O controle dos motores é feito através da porta paralela, especificamente, pelo barramento de dados da porta. Cada motor necessita de 4 sinais, para realizar um passo completo. Como o barramento de dados da porta paralela possui 8 vias, não foi necessário o uso de outro barramento. A Figura 4.(b) mostra o circuito desenvolvido. O dispositivo de captura de imagens utilizado foi uma *webcam Creative Labs*, modelo PD1001, trabalhando em baixa resolução (320x240 pixels) e conectada a um microcomputador Pentium

IV, através da USB (*Universal Serial Bus* ).




$(a)$ $(b)$

**Figura 4. Componentes do hardware desenvolvido: (a) Circuito impresso para controle de motores de passo via porta paralela e (b) Foto do equipamento implementado.**

### 3.2. Componentes de Software

Antes de aplicar o algoritmo de detecção de circunferências, o sistema realiza uma fase de pré-processamento, visando a produção de uma imagem binária (apenas dois tons de cinza, 0 e 255), $I(x, y)$, em que as bordas dos objetos se destacam. Em um primeiro passo, a imagem colorida, no formato de cores RGB, e convertida para tons de cinza. No próximo passo, um filtro Gaussiano de suavização, para redução de ruídos, é aplicado, seguido de um filtro de detecção de bordas de Sobel [Forsyth and Ponce 2003], todos disponíveis no pacote livre ImageJ. As deficiências do filtro do Sobel, principalmente em relação a espessura das bordas detectadas, é compensada na implementação do cálculo de pontos de máximo do espaço de Hough, conforme descrito anteriormente. Além disso, a necessidade de processamento em tempo real, inviabilizada a utilização de filtros de detecção de bordas mais sofisticados, como o de Canny, por exemplo. O limiar utilizado para binarizar a imagem, após a aplicação do filtro de Sobel, foi escolhido empiricamente, e é igual a 80. Finalmente, com a imagem binarizada, o módulo *Hough-Circles* é aplicado. A Figura 5 ilustra as imagens após cada fase do processamento: (1) conversão para tons de cinza, (2) suavização, (3) detecção de bordas, (4) binarização, (5) espaço de Hough e (6) cálculo do ponto de máximo no espaço de Hough.

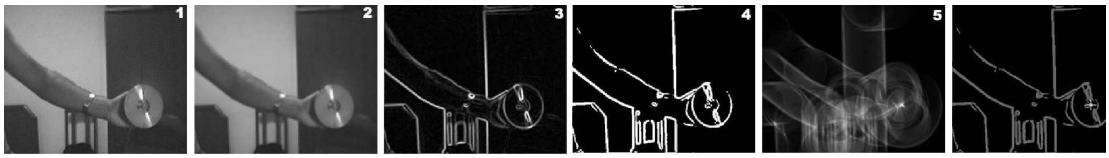



**Figura 5. Imagens em diferentes estágios de processamento**

Com o ponto central do objeto circular, $(x, y)$, detectado, cálcula-se sua distância, horizontal e vertical, ao centro da imagem, o ponto $(160, 120)$ [2]. O fator de conversão entre as distâncias em pixels, e a quantidade de passos a serem executados por cada um dos motores (vertical e horizontal), foi determinada empiricamente, considerando-se uma

[2]O sistema trabalha com imagens de tamanho 320 por 240

distância constante entre o objeto e a $webcam$, e é igual a $1/15$. Com isso, a quantidade de passos para os motores que realizam o movimento na horizontal, ($P_h$), e na vertical, ($P_v$), pode ser facilmente calculada através das seguintes fórmulas: $P_h = (160 - x)/15$ e $P_v = (120 - y)/15$.

Com o total de passos calculado, o sistema utiliza a porta paralela para enviar os sinais aos motores. Uma classe Java, *ParallelPort*, foi utilizada para implementar essa comunicação. Em ambientes *Windows XP* é necessária ainda a utilização do software *UserPort*[3], para desbloquear o acesso a porta paralela.

## 4. Considerações Finais

Este artigo apresentou um sistema de visão computacional, para fins didáticos, capaz de identificar e "perseguir" um objeto circular, em tempo real. Uma das premissas para o desenvolvimento deste sistema foi a utilização de material de baixo custo e software livre, facilitando assim a reprodução dos experimentos por acadêmicos de outras instituições, principalmente em cursos que não contam com sofisticados laboratórios de robótica. Sugere-se, para o futuro, o aprimoramento dessa implementação para permitir a detecção de elipses arbitrárias, o que possibilitará uma maior flexibilidade no sistema de rastreamento, que atualmente não permite que a distância entre o objeto e a câmera sofra muita variação. Esse aprimoramento permitirá também que os objetos circulares apresentem-se em diferentes ângulos em relação ao dispositivo de captura de imagens, pois sua projeção não precisará se restringir a circunferências.

## Agradecimentos

Este trabalho recebeu apoio financeiro da Universidade Católica Dom Bosco, UCDB.

## Referências

Forsyth, D. A. and Ponce, J. (2003). *Computer Vision: A Modern Approach*. Prentice Hall.

Hough, P. V. C. (1962). Methods and means for recognizing complex patterns. *U.S. Patent 3.069.654*.

Pistori, H., Pistori, J., and Costa, E. R. (2005). Hough-circles: Um módulo de detecção de circunferências para o imagej. In *Anais do VI Workshop de Software Livre - WSL*, Porto Alegre, RS.

Souza, G. N. and Kak, A. C. (2002). Vision for mobile robot navigation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237–267.

Spink, A. J., Tegelenbosch, R. A. J., Buma, M., and Noldus, L. (2001). The ethovision video tracking system - a tool for behavioral phenotyping of transgenic mice. *Physiology & Behavior*, 73:731–744.

Toennies, K., Behrens, F., and Aurnhammer, M. (1998). Feasibility of hough-transform-based iris localisation for real-time-application. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02)*, volume 2, pages 299–305.

Wang, J. G. and Sung, E. (2001). Gaze determination via images of irises. *Image and Vision Computing*, 19(12):891–911.

[3]Software livre, gratuito, hospedado em http://www.mattjustice.com/parport/