APLICAÇÃO DE FERRAMENTA OPEN SOURCE EM SISTEMAS DE VISÃO COMPUTACIONAL – DESENVOLVIMENTO DE UM VEÍCULO SEGUIDOR AUTÔNOMO

Riccardo Luigi Delai - riccardo_delai@hotmail.com
Alessandra Dutra Coelho – alessandra.coelho@maua.br
Escola de Engenharia Mauá (EEM/CEUN-IMT)
Praça Mauá, 1 – Bairro Mauá
09580-900 – São Caetano do Sul - SP

Resumo: *A Visão Computacional é uma área em expansão na robótica, influenciada pelos avanços da eletrônica na captação de imagens e no processamento rápido de dados. Um emprego da Visão está no reconhecimento de formas e padrões conhecidos em uma imagem real, gerando uma decisão no fluxo de um programa em função de sua presença e posição. Este artigo apresenta a OpenCV, uma biblioteca de código aberto que oferece funções para o processamento de imagens e funções relacionadas, e mostra a aplicação da detecção de um padrão conhecido em um veículo móvel autônomo, guiado pela apresentação de um símbolo pré-definido.*

Palavras-chave: *Visão Computacional, OpenCV, Veículo Autônomo, Robótica.*

1 INTRODUÇÃO

Visão Computacional, ou Visão Robótica, é um tema que está em franca ascensão. Ela é ponto chave para o desenvolvimento de diversas outras tecnologias, especialmente na área da robótica móvel. Suas aplicações são inúmeras: todas as decisões tomadas baseadas no que se pode enxergar fazem parte do escopo da Visão Computacional. A semelhança com o modo de percepção humana do ambiente (RUSS, 1998) a torna uma ferramenta extremamente poderosa. Este é um sentido que já é bastante dominado pela raça humana, pois podemos facilmente distinguir dois objetos distintos apenas ao olhar para eles, e classificá-los em categorias, como carro, casa, pessoa.

No entanto, esta tarefa simples é bastante dispendiosa computacionalmente, o que explica porque somente agora o campo da Visão está em tamanha expansão. Os motivos são o barateamento do poder de processamento e da fotografia digital, além da miniaturização de ambos. Isso tornou possível aplicações em tempo real (real-time), nas quais as imagens são processadas em tempo semelhante ao do cérebro humano, o que seria impossível com os computadores e sensores de dez anos atrás, tanto por questões de custo quanto de velocidade e tamanho. Atualmente é possível se projetar sistemas simples de visão que possam ser embarcados em robôs autônomos de pequeno porte, esta realidade pode ser vista nas competições de robótica das quais o Instituto Mauá de Tecnologia participa com os robôs desenvolvidos no Laboratório de Robôs Autônomos.

Entre as aplicações já existentes da Visão Computacional estão: câmeras inteligentes (que buscam rostos ao tirar fotos), controle de qualidade industrial por inspeção visual, sistemas de entretenimento sem controle, realidade aumentada, protótipos de veículos sem motorista, voo

não tripulado autônomo, interpretação automática de exames médicos e análise de tecidos biológicos.

A OpenCV é um conjunto de ferramentas de programação para desenvolvimento de aplicações com Visão (BRADSKI, 2008) . Ela engloba também outro conceito importante, especialmente no meio acadêmico, o software livre. A biblioteca é completamente open-source (código aberto e redistribuível), e é distribuída gratuitamente, aberta a colaborações de qualquer indivíduo ou empresa voluntários.

A biblioteca é bem abrangente, e contém funções para tarefas desde a resolução de sistemas lineares à inteligência artificial (RUSSEL, 1995). Isso diminui o tempo de desenvolvimento de aplicações, já que não é necessário reinventar a roda para as tarefas mais simples, nem programar algoritmos mais complexos que já tenham sido implementados na biblioteca. Entre as funcionalidades da biblioteca estão (Autores da OpenCV 2010):

1. Módulo matemático com operações algébricas, resolução de sistemas, operações em matrizes (soma, multiplicação, determinante, cofator, etc), funcionalidades de geometria e estatística.
2. Interface gráfica e desenho em imagens, possibilitando a criação de interfaces simples de teste e protótipo. As funções de desenho são bastante úteis para demonstrar resultados e auxiliar o desenvolvedor, já que a melhor forma de encontrar erros em um sistema de processamento de imagem é analisando as próprias imagens.
3. Interface para arquivos de imagem, vídeo e câmeras, através de funções simples. Suporte a diversos formatos de arquivos de vídeo, (TUDOR, 1995) e conexões de câmera.
4. Processamento de imagem: operações em imagens (soma, subtração, conversão de tipo), operações morfológicas (abertura, fechamento, dilatação, erosão e etc), operações geométricas (rotação, perspectiva, alteração de tamanho, recorte), blur.
5. Identificação de bordas: Sobel, Laplace, Canny, Harris (DERPANIS, 2004) e diversos outros métodos.
6. Análise de formas: cálculo de momentos, momentos invariantes de Hu, simplificação de formas geométricas.
7. Análise de cor: histogramas, comparação de histogramas.
8. Mudança de representação: log-polar, domínio da frequência (FFT) e outros.
9. Visão estéreo.
10. Inteligência artificial e métodos de aprendizado computacional: redes neurais, árvores de decisão, redes bayesianas e outros métodos em desenvolvimento.

A OpenCV contrasta com as soluções comerciais, como o IMAQ™ e o MATLAB™ por ser código aberto e por ser programada em linguagens de uso genérico e bastante difundidas, o que possibilita interação com qualquer outra biblioteca ou programa existente nessas linguagens. A OpenCV já tem versões para C, C++ e Python, mas existem ports em andamento para outras linguagens. Além disso, a OpenCV pode ser mais rápida que as aplicações citadas, por dispensar a camada extra de software existente nestas, trabalhando diretamente com o sistema operacional.

A gratuidade da OpenCV, o baixo custo do poder de máquina e a crescente qualidade das câmeras torna possível o desenvolvimento de sistemas sofisticados de Visão, com baixo investimento e custo de operação. A utilização de câmeras pode inclusive substituir outros sensores e sistemas mais caros, complexos e menos genéricos. Isso evidencia não só o crescimento da área da Visão Robótica, mas também a tendência de aproximação dos computadores à forma como os humanos entendem o ambiente ao seu redor.

Para exemplificar o estudo realizado escolheu-se implementar um sistema capaz de tornar um veículo semi-autônomo. Este sistema foi adaptado a partir de materiais de fácil acesso e

ampla comercialização, tornando um simples carrinho de controle remoto em um sistema semi autônomo, que tenta encontrar e seguir uma forma pré-determinada.

2 MATERIAIS E MÉTODOS

Para o desenvolvimento e testes do projeto, foram necessários a API (AUTORES DA OpenCV 2010) (Application Programming Interface, a relação de funções) da biblioteca, um computador com compilador C++ e a OpenCV instalada. Mais particularmente, foi utilizado o Linux como sistema operacional, por motivos de comodidade na programação e seguindo a ideologia de software livre da OpenCV, mas foram feitos também testes com Windows para que a pesquisa fosse o mais abrangente possível.

Como a OpenCV segue um ciclo rápido de desenvolvimento, foi criado um pequeno manual para garantir uma instalação comum da biblioteca, compilador e suas dependências em todos os computadores utilizados na pesquisa, evitando assim problemas de compatibilidade entre versões de código. Outra vantagem de documentar a instalação é que pessoas interessadas em reproduzir os exemplos da pesquisa podem partir do mesmo ponto, garantidamente funcional.

A pesquisa teve que ser bastante balanceada em relação à teoria do processamento de imagem e à implementação. A bibliografia voltada à programação cita apenas que determinado procedimento é mais demorado ou complexo do que outro, enquanto os livros mais teóricos ignoram os aspectos de implementação e abordam apenas os conceitos algorítmicos e matemáticos, não levando muito em conta o custo computacional. Para um curso de engenharia, no qual há tanto o conhecimento matemático quanto de implementação computacional, é necessário obter um panorama geral, que compreenda os motivos tanto teóricos quanto práticos de cada procedimento. Por ser uma biblioteca bastante dinâmica, com atualizações frequentes, muitas das soluções propostas na literatura já estão obsoletas ou foram alteradas. Assim, a consulta em fóruns especializados, documentação atualizada, changelogs (registro de mudanças entre as versões de um software) até testes para verificação devem ser feitos com frequência, até que se tenha certeza que as informações na literatura permanecem válidas.

A melhor forma de se entender os mecanismos do processamento de imagem é utilizar as próprias imagens como exemplo, analisando o que os operadores fazem sobre elas. É nesta etapa que o editor de imagens se torna interessante, para gerar imagens que se adequem aos conceitos testados. Nos casos de morfologia por exemplo, é conveniente se trabalhar com imagens bastante simples, para que se perceba imediatamente o efeito causado pelos algoritmos, enquanto nos processos com histogramas e detecção de bordas é mais interessante a demonstração em uma imagem mais sofisticada, como uma fotografia real, para que o efeito geral sobre toda a superfície seja mais notado do que o particular em pontos específicos.

Trabalhar diretamente com câmera e vídeos como exemplos não é uma boa prática. É muito difícil se ter um fundo completamente uniforme durante todos os quadros, e uma câmera completamente livre de ruídos. Assim, alguns dos processos não funcionam tão bem quando aplicados diretamente sobre uma imagem vinda de uma câmera. A utilização da câmera varia conforme a aplicação, cabendo ao implementador definir que filtros devem ser aplicados antes de partir para o processamento.

Foi dada preferência a procedimentos no domínio especial da imagem, ao invés do domínio da frequência como apresentado em grande parte da literatura teórica (GONZALEZ, 2000). As transformadas de Fourier (FFT) direta e inversa são relativamente lentas frente ao domínio especial, como as imagens se apresentam e são armazenadas, podendo comprometer a velocidade total da aplicação. Para provar os conceitos estudados sobre Visão

Computacional e demonstrar uma possível aplicação na área de robótica móvel, foi desenvolvido um sistema que identifica um símbolo pré-definido em uma imagem, vinda de uma câmera, e comanda um pequeno veículo para segui-lo.

A identificação do símbolo é mais dependente de sua forma do que de sua cor. A identificação pela forma é muito mais robusta do que pela cor, já que esta pode variar bastante com a luminosidade ambiente, enquanto essa tende a variar bem menos. Além disso, é difícil se obter precisão na identificação de cores, e vários objetos têm cores confundíveis, o que diminúi a precisão só processo.

O veículo é um carrinho de controle remoto, cujo controle foi adaptado para que fosse possível controlá-lo por um sistema externo, no caso uma plataforma Arduino. Como no controle original os movimentos de acelerar, ré, direita e esquerda eram controlados por botões simples, o microcontrolador simula uma chave variando suas saídas entre input (estado de alta impedância, chave aberta) e output nível baixo, que corresponde à chave fechada no controle. Isto dispensa componentes extra na adaptação do controle.

O Arduino é ligado a um computador via USB, que envia dados através de emulação serial para que esse altere o estado no controle. São enviados comandos de apenas uma letra, representando situações desejadas no controle, e por consequência no carrinho. O computador toma as decisões com relação a qual movimento realizar baseado nas imagens vindas da câmera de um celular, posicionado na parte superior do carrinho, que envia seus quadros via Bluetooth.

Para a etapa de comunicação câmera-computador foi necessário o uso de um software proprietário pago. Isso acontece porque o celular não permite que programas sem certificação de empresas de segurança tenham acesso a dados considerados sensíveis, como a câmera. Acessos a esta através de um software sem certificação obrigam o usuário a apertar um botão a cada quadro obtido, autorizando o programa a utilizá-lo, o que inviabilizaria o projeto.

Com a imagem da câmera, o computador inicia o processo de busca do símbolo, com base em uma referência digital deste. Repetido para cada quadro (frame) obtido, ele segue os seguintes passos:

1. Obter a imagem da câmera.
2. Suavizar (smooth) a imagem vinda da câmera, para diminuir o ruído espúrio. O mecanismo de suavização escolhido foi o blur gaussiano.
3. Converter a imagem colorida para preto-e-branco (escala de cinza).
4. Aplicar thresholds (limiares) crescentes, gerando imagens binárias (preto e branco, apenas). São aplicados valores crescentes para que o símbolo buscado possa ser visto em várias condições de luminosidade ambiente. Para cada imagem gerada neste item:
5. Aplicar rotina de detecção de contornos em toda a imagem. A imagem binária torna-se uma sequência de pontos que representam as fronteiras entre regiões pretas e brancas.
6. Simplificar os contornos, utilizando o algoritmo de Douglas-Peucker de aproximação de polígonos. Selecionar contornos quadrados pela relação de aspecto, número de vértices e tamanho mínimo. Isso cria uma lista de regiões candidatas a conter o símbolo, buscando pelo quadrado externo.
7. Obter todas as regiões quadradas candidatas novamente na forma de imagem.
8. Procurar o quadrante com maior quantidade de pixels brancos para cada região do item anterior.
9. Rotacionar as imagens do item 7 para que sigam a orientação da referência, baseado no quadrante mais branco identificado no item 8 e no ângulo formado entre a linha inferior do contorno e o eixo horizontal da imagem.
10. Calcular os momentos da imagem binária, resultado do item anterior. Os momentos utilizados são tais que a ordem em X somada com a ordem em Y seja menor ou igual a 3, com X e Y positivos.

11. Compor uma nota com o quadrado da diferença entre os momentos de cada ordem da referência e da imagem candidata, obtidos no item 10, somada de dez vezes a diferença entre a quantidade de figuras presentes na referência e no candidato. Uma figura é qualquer porção preta envolvida por branco ou branca envolvida por preto.
12. Selecionar a região candidata (dentre as obtidas no item 7) com menor nota (dada no item 11) acima de um limiar estabelecido experimentalmente. Se tal região existir, ela é considerada um match (encontro) válido, ou seja, a região encontrada é semelhante o bastante à referência. O limiar pode ser alterado durante a execução do programa, para mudar a tolerância deste. O controle é dado na forma de uma barra deslizante na tela.
13. Considerar o ponto central do match (centro do quadrado) para avaliar sua posição. Se este estiver nos primeiros 30% da imagem, na parte esquerda, o carrinho é comandado para virar à esquerda. Se estiver nos 70% mais à direita, é enviado um comando para que o carrinho vire à direita. Para determinar a distância do carrinho à figura, é utilizada a área do quadrado externo. Se esta for menor que 5% da imagem, o carrinho deve avançar. Se for maior que 12%, ele deve andar de ré e se afastar. Todos estes parâmetros podem ser alterados em tempo de execução.
14. Com o carrinho reposicionado, voltar ao passo 1.

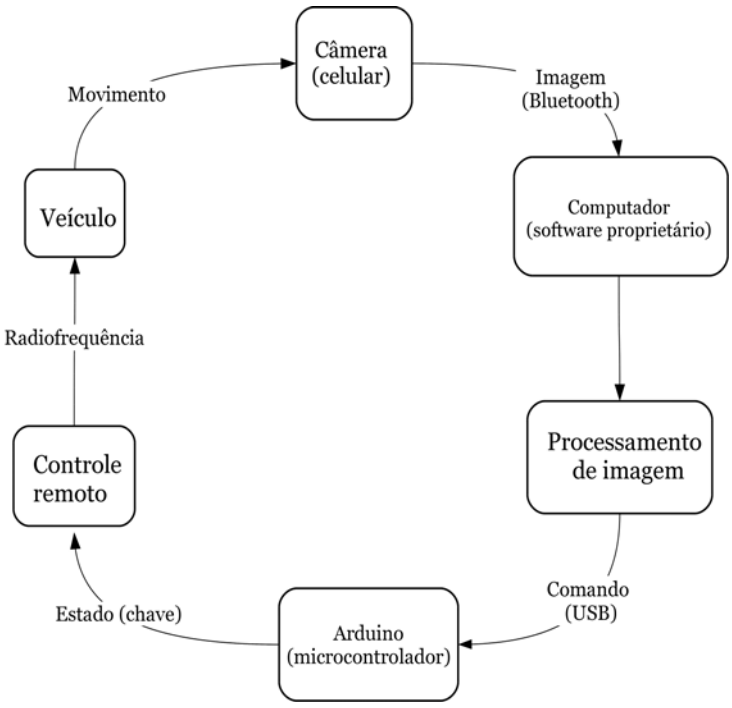

Figura 1. Esquema de funcionamento do sistema do veículo.

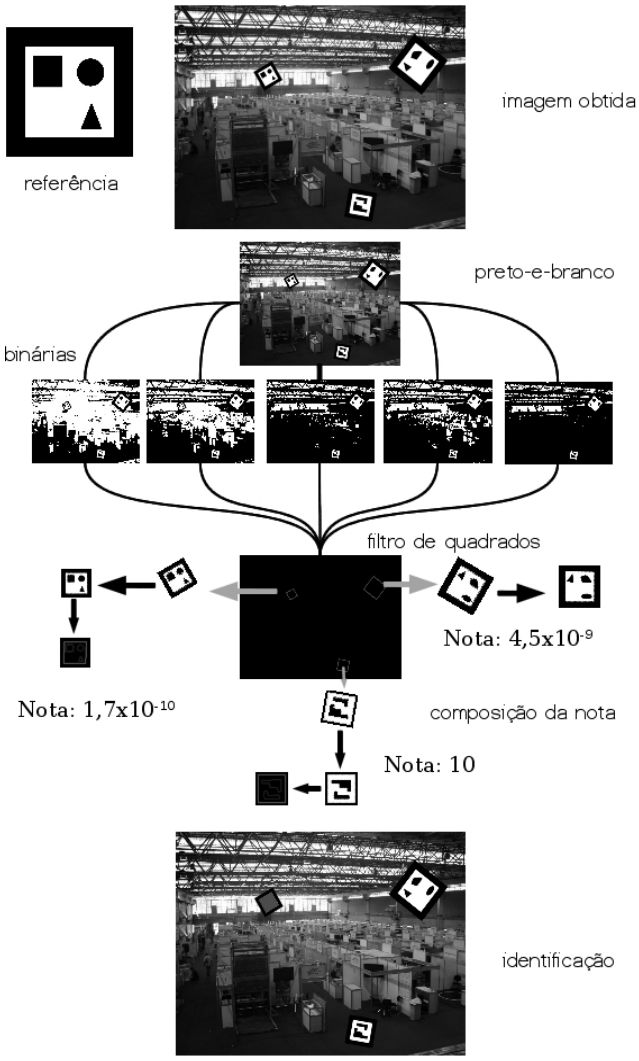

Figura 2. Processo de busca de referência no programa.

3 RESULTADOS E DISCUSSÃO

O sistema de identificação do símbolo funcionou bem quando testado em imagens estáticas, mesmo com ruído e com a adição de símbolos semelhantes ao procurado, sempre identificando corretamente a referência com a nota mais baixa.

Com a câmera adaptada no carrinho, o sistema também se comportou bem. Os maiores problemas estão relacionados com a movimentação do carrinho, que não tem grande ângulo de curva nem controle de intensidade de aceleração ou ré. Por causa disso, o carrinho tende a avançar na direção do símbolo assim que o encontra ao longe, mas ao perceber que está perto e acionar o freio, a inércia dele o movimenta além do ponto de parada esperado, deixando-o próximo demais. Quando está muito próximo, a ré é acionada para tentar manter a distância ideal, mas o mesmo acontece, e o carrinho fica em um ciclo de frente e ré sem encontrar a distância ideal prevista.

Para minimizar este problema, seria ideal que o carrinho tivesse controle de velocidade, para que acelerasse proporcionalmente à distância que se encontra da referência.

Para diminuir o custo de processamento, o programa poderia não procurar pelo símbolo a cada frame recebido, mas tentar segui-lo uma vez que o mesmo foi encontrado. Isso poderia ser feito com algum algoritmo de fluxo óptico, como o de Lucas-Kanade.

4 CONCLUSÃO

O projeto resultou em uma apostila que abrange desde conceitos simples até os mais avançados sobre a utilização da OpenCV (software livre) em sistemas de visão Computacional. Este material servirá como base para a introdução de um laboratório para difundir os conhecimentos de Visão, tanto na grade curricular como para auxiliar o laboratório de robôs autônomos.

O veículo adaptado é o início de um projeto de desenvolvimento de um carro autônomo. Ele será reproduzido em escala maior em um miniveículo elétrico, para o qual também será aumentado o grau de automação. O veículo será capaz de dirigir-se sozinho por distâncias maiores, evitando obstáculos e seguindo percursos previamente definidos, com a introdução de obstáculos dinâmicos.

REFERÊNCIAS BIBLIOGRÁFICAS

AUTORES da OpenCV . OpenCV Reference Manual v2.1, 2010.

BRADSKI, G.; KAEHLER, A. **Learning Opencv**. 1ª edição. Ed. O'Reilly Media, 2008.

DERPANIS, K. G. **The Harris Corner Detector**. Disponível em : <www.cse.yorku.ca/~kosta/CompVis_Notes/harris_detector.pdf> 2004 .

GOMES, M. M. Apresentações do Curso ECM951 - Visão Computacional. São Caetano do Sul: Escola de Engenharia Mauá. 2010.

GONZALEZ, R. C.; WOODS, R. E. **Processamento de imagens digitais. 2ª edição**. Ed. Edgard Blücher, 2000.

RUSS, J. C. **The image processing handbook. 5ª edição**. Ed. CRC Press, 1998.

RUSSEL, S.; NORVIG, P. **Artificial intelligence - a modern approach**. Ed. Prentice Hall, 1995.

TUDOR, P. Mpeg-2 video compression. **Electronics & Communication Engineering Journal,** 1995.

APPLICATION OF OPEN SOURCE TOOLS IN COMPUTER VISION SYSTEMS DEVELOPMENT OF AN AUTONOMOUS VEHICLE

Abstract: *Computer Vision is an expanding area in the field of robotics, rising mainly by the fast growing capabilities of image capturing and digital data processing. A possible use of Computer Vision lies in identifying a known shape in a real image, triggering a process on a commanding program due to its presense and location. This article presents the OpenCV, an open-source library that contains several functions for image processing and related utilities, and an application on finding a known shape on a real environment, guiding an autonomous vehicle through the presentation of a pre-defined symbol.*

Key-words: *Computer Vision, OpenCV, Autonomous Vehicle, Robotics.*