PROJETO DEMONSTRATIVO 2
TRANSFORMADA HOUGH USANDO OPENCV

Universidade de Brasília

Eduardo Furtado Sá Corrêa - 09/0111575
Princípios de Visão Computacional – Turma A - 2/2013
20/11/2013
eduardoxfurtado@gmail.com

Resumo Este relatório mostra como foi feito o desenvolvimento de uma metodologia em OpenCV utilizando c++ para a detecção de linhas e formas circulares a partir da Transformada de Hough.

Palavras Chaves: Transformada Hough, OpenCV

1 OBJETIVOS

Desenvolvimento de uma metodologia em OpenCV para a detecção de linhas e formas circulares, a partir da Transformada Hough, em imagens e vídeos em tempo real.

2 INTRODUÇÃO

A partir de métodos vistos em sala de aula e um tutorial disponibilizado[1], ao decorrer do curso de Princípios de visão computacional, para analisar e processar imagens e vídeos, foi possível extrair deles informações apresentadas nas próprias imagens. Neste caso, apontamos a presença de formar lineares ou circulares.

Para alcançar resultados de maneira eficiente, foi utilizado o que talvez há de melhor no mundo, presente na biblioteca de software livre OpenCV (Open Source Computer Vision Library).

A principal ferramenta trata-se da transformada Hough, que neste caso é usada para isolar formas particulares numa imagem, que podem então serem detectadas mediante a descrição de um modelo vetorial da forma a ser encontrada.

Os detalhes encontram-se na seção 4, de procedimentos.

Tentou-se ir além do que foi proposto, e portar a aplicação desenvolvida em C++ num ambiente Linux para o ambiente Android, com o intuito de apresentar o software funcionando em um tablet comercial. Infelizmente não se obteve sucesso, entretanto os passos seguidos serão documentados, pois ao menos um aprendizado foi gerado.

3 MATERIAIS

Linux Mint 14 com OpenCV 2.3.
G++ (Ubuntu/Linaro 4.7.2-2ubuntu1) foi o compilador C++ utilizado. A seguinte linha foi usada para compilação do programa:
g++ programa.cpp -o 090111575 ‘pkg-config –cflags –libs opencv‘;

Foi utilizado um vídeo, fornecido pela especificação, duas imagens fornecidas pela especificação, e também nove outras imagens e fotos extras.

Um tablet “Kindle Fire HD” com uma versão modificada do Android 4.2 foi utilizada.

Uma webcam de baixa qualidade foi utilizada.

4 METODOLOGIA

Inicialmente um código foi preparado permitir suporte de entrada a um arquivo de vídeo e também o streamming de vídeo fornecido por uma webcam instalada na porta Usb do Computador.

O código funciona detectando bordas no input. Isso é feito utilizando o filtro Canny, presente no OpenCV. Diversos parametros foram testados para as várias imagens de entrada. Estes parâmetros definem valores máximos e mínimos onde cada pixel é definido como borda se seu gradiente está entre estes valores.

Em seguida utiliza-se a função HoughLinesP, também do OpenCV, que gera um vetor de linhas encontradas na imagem. Neste caso, diversos parâmetros diferentes também foram testados.

Na sequencia, aplicamos a função GaussianBlur na imagem. Isso faz com que seja mais fácil detectar algumas formas, devido a, pode-se dizer, um aumento de tolerância por conta da diminuição de precisão. Diversos valores para seus parâmetros também foram testados.

Com tudo preparado, aplicamos a função HoughCircles, também presente no OpenCV que detecta os circulos presentes na imagem pré-processada até então. Diversos valores foram testados.

Finalmente são desenhadas na saída as formas detectadas.

Vários parâmetros que apresentaram melhores resultados para algumas imagens continuam presentes no código em linhas comentadas.

Além do que foi proposto na especificação, também foi feito uma tentativa de portar o software para ambiente Android. O OpenCV tem um certo suporte para *mobiles*, entretanto ainda relativamente prematura e pouco difundida, talvez pela dificuldade em configurar um ambiente de trabalho, e a dificuldade de ter que lidar com dispositivos com poder de processamento limitado.

Foi seguidos os passos de tutoriais, guias e documentação disponíveis em [5], [6], [7] e [8].

Foi instalada uma SDKs adicionais e tentou-se portar o código para Java, ou mesmo para C++ através da NDK [6], [7].

O abandono dessa parte do projeto foi dado em decorrência do tempo necessário para fazer tudo funcionar de fato.

5 PROCEDIMENTOS

O programa foi testado com as onze imagens, sendo as duas imagens padrão disponibilizadas em [2] e em [3].

O programa foi testado para o vídeo disponível em [4].

O programa foi testado utilizando a webcam como sinal de entrada. Nesta parte, imprimiu-se três imagens e estas foram expostas para captura pela webcam.

Foi feito uma tentativa de gravar o vídeo da webcam em um arquivo, para fazer o processamento como feito no vídeo para observar se os resultados seriam os mesmos observados durante a gravação.

6 RESULTADOS

O programa tem uma simples interface textual, que diz:

"This program demonstrates line and circle finding with the Hough transform."
"Usage:"
"./hough_pvc <type> <filename>"
"Type 0 = image. Type 1 = video"
"If no parameters are entered it uses your webcam"

"**"

"Detected lines are drawn in blue, circles in red with their center in green"

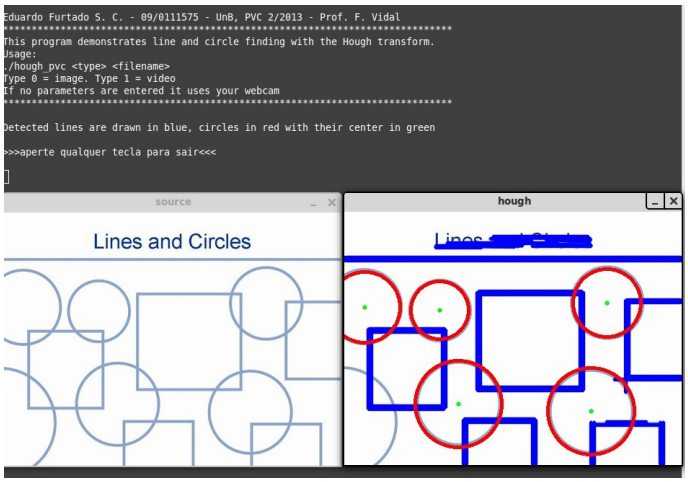

Figura 1.

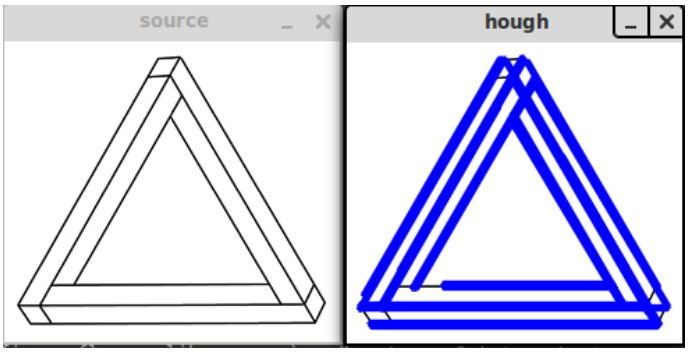

Figura 2.

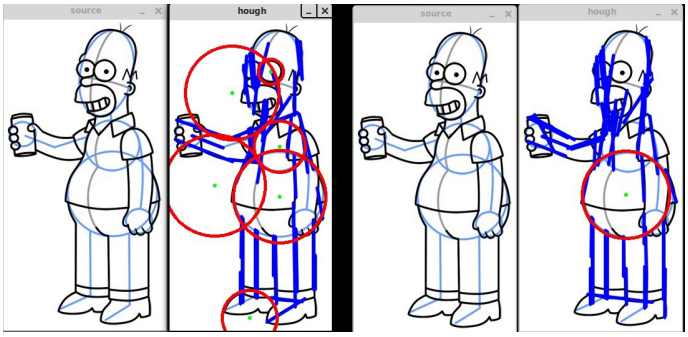

Figura 3.

A figura 1 e 2 mostram como o programa funciona bem, detectando praticamente todas as linhas presentes, e em quase todas suas extensões, além de não falhar nos círculos que estão completos.

Já na figura 3 observa-se que com uma imagem mais complexa diminui a precisão e notamos o aparecimento de artefatos. Outro problema foi também figura 1, em que se detectam artefatos em um texto, culpa do ruido.

Figura 4 – Imagem da execução em um frame de um vídeo.

Já em um vídeo os resultados são muito bons em hora, e muito ruins em outros. A razão disso é que as cenas podem ser muito diferentes umas das outras, e com a implementação apresentada, apenas uma dessas cenas pode ser otimizada pelos argumentos das funções aplicadas. Nenhum circulo foi detectado, mesmo com vários parâmetros testados.

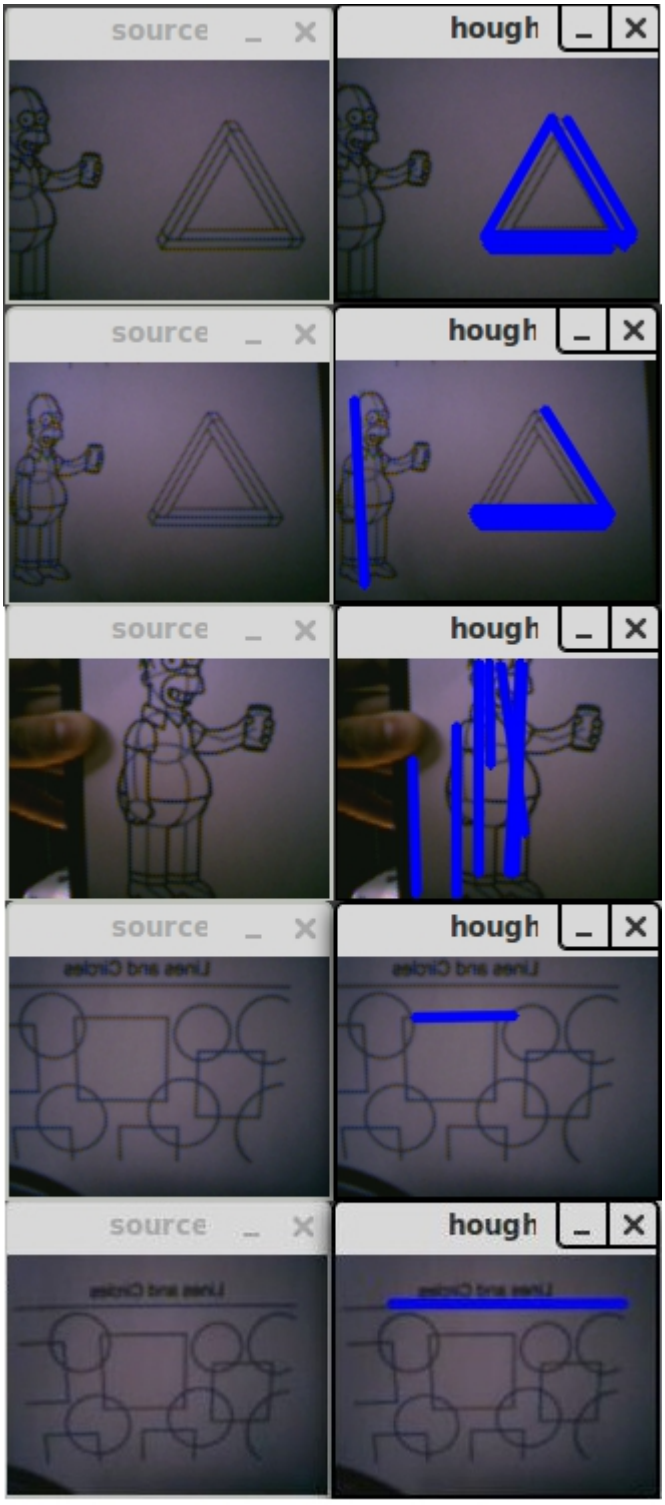

Figura 5 – Imagens de frames provenientes da webcam

Ao expor as imagens para uma webcam capturando na medíocre resolução de 160x120, como pode ser visto na figura 5, pode-se observar que os resultados estão muito distantes daqueles obtidos até então. Em primeiro lugar, isso se da ao fato já mencionado anteriormente que também acontecia nos vídeos, de que o algoritmo implementado pode apresentar somente uma configuração por execução. Soma-se a isso fatores de iluminação – foram testadas em um quarto iluminado por luz incandescente durante a noite, o mesmo local durante o dia na penumbra do sol e também com luz direta do astro.

Há muito ruido, pouca acurácia e precisão, e isso explica o fato de não terem sido observados círculos na imagem.

Por fim, ao executar o vídeo gravado pela webcam, observou-se que os resultados foram os mesmos, comparando com a imagem durante a gravação.

7 CONCLUSÕES

A transformada de Hough é um bom método para encontrar formas em imagens. Para vídeos deixa a desejar, a menos que seja implementado de forma adaptativa ao que está acontecendo a cada frame. É essencial reduzir ruído, e também simplificar as coisas, como foi feito utilizando o filtro de detecção de bordas de Canny ou o Gausian Blur.

REFERÊNCIA BIBLIOGRÁFICA

[1] http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html acessado em 20/11/2013

[2] http://bit.ly/19bJzYtC acessado em 20/11/2013.

[3] http://www.presentationmagazine.com/powerpoint-templates/00105/468slide1.jpg acessado em 20/11/2013.

[4] http://dasl.mem.drexel.edu/~noahKuntz/Flight%20Training/Firebird_Hover.avi acessado em 20/11/2013.

[6] http://www.stanford.edu/class/ee368/Android/Tutorial-1-Basic-Android-Setup-Linux.pdf acessado em 20/11/2013.

[7] http://opencv.org/platforms/android.html acessado em 20/11/2013.

[8] http://developer.android.com/tools/sdk/ndk/index.html acessado em 20/11/2013.

[9] Documentação online do OpenCV, disponível em http://docs.opencv.org acessado em 20/11/2013.