Uso de Visão Computacional em Dispositivos Móveis para o Reconhecimento de Faixa de Pedestres

Kelly Sousa, Mauricio Marengoni
Universidade Presbiteriana Mackenzie
São Paulo - SP - Brasil
kellyoliveira@gmail.com, mmarengoni@mackenzie.br

Abstract

This paper presents initial ideas on building an application that is intended to assist the visually impaired in crossing streets using mobile devices. This work also examined the challenges and some solutions to develop mobile interfaces available that perform image processing and voice recognition for crosswalks detection. Further information is given about using the OpenCV Library for the Android platform and the differences in the processing of computer vision algorithms between the mobile and other environments. The results presented are preliminary, but demonstrate the feasibility of the project.

I. Introdução

De acordo com (IBGE, 2000) no Brasil existem 148 mil pessoas cegas e 2,4 milhões de pessoas com deficiência visual, consequentemente surge a necessidade da criação de ferramentas que facilitem a execução de algumas tarefas cotidianas por parte dessa população. Sem dúvida um dos desafios para um deficiente visual é poder transitar de forma segura, eficiente e, preferivelmente, autônoma.

A propagação do uso de dispositivos móveis com maior poder de processamento abre uma nova gama de aplicações para a visão computacional, mas ao mesmo tempo adiciona novos desafios considerando as mudanças de arquitetura, usabilidade e padronização quando comparadas as aplicações desenvolvidas para computadores (BHARGAVA, 2010).

Considera-se que esse campo ainda não tem sido muito explorado e verifica-se a necessidade de adaptação de alguns algoritmos para a arquitetura móvel já que mesmo os melhores aparelhos celulares não possuem ainda a capacidade de processamento necessária para a execução em tempo real destes algoritmos.

Neste contexto este trabalho propõe a criação de um sistema que auxilie o deficiente visual na detecção de pontos seguros para travessia de ruas usando aparelhos celulares que utilizem a plataforma Android.

O projeto tem como principal objetivo o uso do processamento de imagens e visão computacional para detecção de sinais indicando que se trata de um ponto de travessia que pode ser utilizado. O sistema inicialmente está trabalhando com o reconhecimento de faixas de pedestres, mas conforme será mostrado ao longo do artigo já existe o estudo de trabalhos relacionados para o reconhecimento de outros objetos como pessoas, carros e semáforos.

Esse artigo foi organizado da seguinte forma: a seção II mostra, resumidamente, trabalhos relacionados, a seção III explica em linhas gerais o sistema proposto enquanto a seção IV mostra mais detalhes sobre o algoritmo para detecção de faixas de pedestre, a seção V mostra particularidades importantes do ambiente Android, a seção VI explica como foram conduzidos os primeiros testes do sistema e a seção VII mostra os resultados que foram obtidos até o presente momento, por último são apresentadas as conclusões e os trabalhos futuros a serem desenvolvidos.

II. Trabalhos Relacionados

Esse projeto teve como inspiração o trabalho de BHARGAVA (2010) que desenvolveu um sistema baseado em dispositivos móveis e processamento em nuvem para a travessia de pedestres em ruas. Embora a ideia deste trabalho seja semelhante, utilizou-se uma linha de desenvolvimento diferente. No trabalho de BHARGAVA o processamento de imagens não é realizado localmente, o sistema captura uma foto que é enviada a um servidor responsável pelo processamento das imagens utilizando um classificador em cascata do tipo AdaBoost (Freund et al., 1999). No trabalho aqui apresentado o processamento é realizado localmente numa plataforma móvel.

Como o objetivo deste trabalho é auxiliar pessoas com deficiências visuais na travessia de ruas, outros trabalhos estudados foram aqueles que tinham como principal alvo de pesquisa a criação de algoritmos para detecção de faixas de pedestres. Seguindo essa linha LAUSSER et al. (2008) utiliza classificadores AdaBoost para reconhecimento de faixas de pedestres. COUGHLAN (2006) utiliza para detecção de faixa de pedestres uma técnica chamada de segmentação de figuras de fundo.

Em STEPHEN SE (2000) é utilizada a estimativa de posição na detecção de faixas de pedestre. Neste trabalho três métodos diferentes são comparados: busca homográfica, convergência de linhas paralelas e convergência de linhas paralelas com 2 pontos de fuga e por último o trabalho de Udin et al. (2005) faz o reconhecimento de faixas de pedestre

através do uso da segmentação bi-polar e pontos de projeção invariantes.

Todos esses trabalhos que consideraram o uso de pontos de projeção tiveram resultados interessantes, porém os algoritmos também utilizam imagens de faixas com qualidade difícil de ser obtida em situações reais com câmeras de dispositivos móveis. Este projeto apresenta uma proposta para o reconhecimento de faixas de pedestres de forma mais realista, utilizando imagens capturadas e processadas no dispositivo móvel.

Futuramente para que o projeto tenha êxito e auxilie deficientes visuais a efetuarem travessias o sistema deve reconhecer além de faixas de pedestre, também pessoas, carros e sinalizações assistentes. Esse reconhecimento ainda não é realizado no sistema, somente a detecção de faixas de pedestre, porém esses itens estão planejados em uma etapa posterior de desenvolvimento.

Esses objetos têm características que não podem ser observadas através de verificações geométricas e para esses casos é recomendado o uso de algoritmos de classificação que, com base em amostras do mundo real, podem, após o treinamento, determinar se uma imagem possui o objeto de interesse ou não. Um algoritmo de classificação muito utilizado para essa tarefa é o classificador em cascata AdaBoost (BRADSKI, 2008).

Outra etapa importante da classificação é a seleção de características de uma imagem de forma a extrair informações nela existentes. Existem vários algoritmos para a seleção de características como é o caso do HAAR (VIOLA et al., 1999), no entanto, conforme mostra o trabalho de Chang-yeon (2008), que faz o reconhecimento de faces em sistemas móveis, seu custo de processamento nestas plataformas inviabiliza seu uso.

O trabalho de Chang-yeon (2008) também aponta uma alternativa para a execução de tarefas de reconhecimento em aparelhos celulares, que é o uso de classificadores em cascata do tipo AdaBoost juntamente com o uso de algoritmo de seleção de características LBP (*local binary pattern ou padrão binário local*), mesmo que o uso desse algoritmo apresente uma menor acurácia em relação a outros algoritmos para seleção de características.

A extração de características LBP é feita dividindo a imagem em sub-regiões onde os histogramas LBP são calculados para cada sub-região e então somados em um único histograma de características conforme ilustra a figura 1:

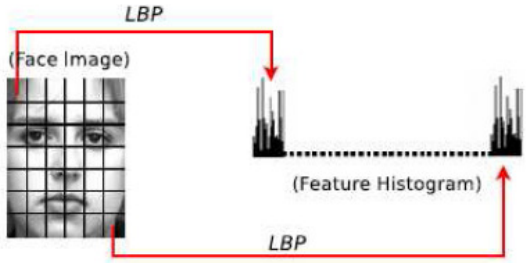

Figura 1 Ilustração do funcionamento dos algoritmos LBP
Fonte: (Chang-yeon, 2008)

III. SISTEMA PROPOSTO

O sistema para a detecção de faixas de pedestre foi criado com o intuito de ter o processamento realizado localmente em um dispositivo móvel usando plataforma Android na versão 2.3.

A plataforma Android foi escolhida devido a um conjunto de fatores: primeiro a versão OpenCV 2.3 já fornece suporte para essa plataforma, segundo a plataforma possui uma arquitetura e um kit de desenvolvimento que facilita a criação de aplicações acessíveis e, por último, de acordo com instituto GARTNER (2011), o Android possui 52% da soma do mercado mundial de *smartphones*.

A criação de uma interface voltada para os deficientes visuais é um desafio, pois aparelhos móveis de toque possuem, em essência, um forte apelo visual, porém para o público escolhido necessita-se que a interface seja baseada nos demais sentidos. Logo a audição e o tato foi o caminho escolhido para a comunicação das ações do programa criado.

A figura 2 mostra o fluxo do sistema proposto. O sistema, quando ativado pelo usuário através de um toque ou por voz, efetua o reconhecimento do ambiente para a detecção da faixa de pedestres. Ao reconhecer a faixa o sistema informa ao usuário através da vibração do aparelho ou por processamento de voz a condição de travessia na faixa de pedestre, se é segura ou não.

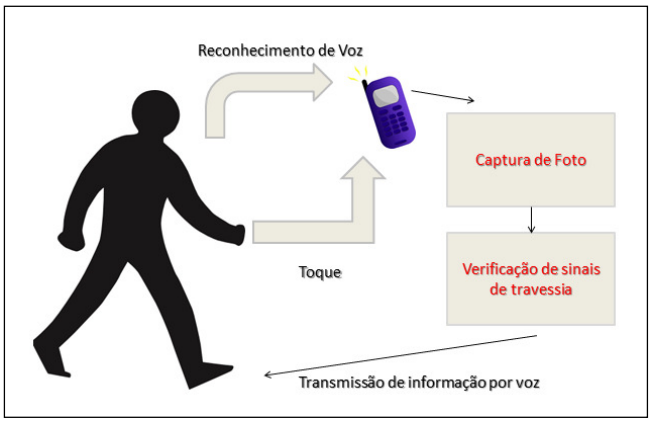

Figura 2 Fluxo da interface do sistema

Com o intuito de maximizar a experiência do usuário a aplicação criada foi dividida em três componentes básicas:

- Widget – componente criado com o intuito de tornar possível o uso do programa através de um simples toque na tela do aparelho celular. Um widget é um programa/aplicativo que fica sempre ativo na área de trabalho do aparelho móvel (Android website, 2012).
- Programa principal – é a componente responsável pelo reconhecimento dos sinais de travessia propriamente ditos.
- Serviço – componente ainda não desenvolvido cuja principal funcionalidade será o reconhecimento de voz para execução do programa mesmo que o aplicativo não esteja ativo.

Através do protótipo desenvolvido nesta primeira fase pode-se visualizar como será o comportamento do sistema e seus principais componentes. A figura 3 mostra um exemplo de execução do sistema em uma simulação de situação real. Os quadros 3a e 3b registram o uso do widget por parte do usuário acionando o programa principal e no quadro 3c o sistema detecta a faixa de pedestre emitindo um alarme sonoro sobre a detecção.

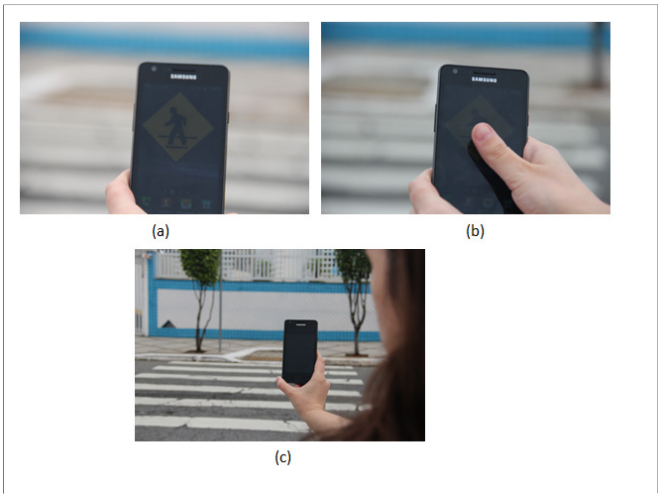

Figura 3 Execução do protótipo do sistema

IV. ALGORITMO CONSTRUÍDO PARA DETECÇÃO DE FAIXAS DE PEDESTRES

Uma das principais dificuldades para o desenvolvimento deste sistema são as variações que as faixas de pedestres possuem no mundo real, especialmente em cidades de grande porte como é o caso de São Paulo, onde existem várias faixas de pedestre que estão desgastadas e sem manutenção adequada. A figura 4 apresenta um exemplo de uma típica faixa de pedestre não uniforme com falhas na pintura que podem interferir no processo de reconhecimento.

Figura 4 Exemplo de faixa de pedestre não uniforme

Inicialmente foi tentado o uso de análise de contornos de forma a encontrar formas geométricas, porém os resultados não foram satisfatórios já que a condição de captura da imagem pela câmera do celular não é feita sempre na mesma posição o que dificulta o processamento das imagens. Também é importante destacar às irregularidades das faixas de travessia e a falta de controle de ambiente.

O sistema deve ser capaz de funcionar em diferentes condições de iluminação. A figura 5 apresenta uma faixa de pedestre onde a falta de uniformidade na iluminação do ambiente dificulta o processamento da região da faixa.

Figura 5 Imagem com diferentes condições de iluminação

O algoritmo foi desenvolvido utilizando a biblioteca OpenCV 2.3 e processado inicialmente em um computador Intel Core I5 de 64 bits com 6 GB de memória. Uma vez desenvolvido e testado o sistema foi portado para a plataforma Android. O objetivo era a comparação de resultados entre as diferentes plataformas.

É importante considerar que aplicações para celulares Android podem funcionar com orientações de tela paisagem e/ou retrato, para efeito de simplificação dos algoritmos de visão computacional considerou-se o uso da aplicação somente tipo retrato. O sistema então captura e trabalha com imagens com resolução de 480x640 em tons de cinza.

A partir desse ponto o sistema inicia algumas operações de pré-processamento, com o intuito de preparar a imagem para a separação dos objetos de interesse e primeiramente aumenta o contraste com uma equalização de histograma (LAGANIÈRE, 2011).

Como segundo passo o sistema aplica um corte de 200/255 (valores definidos empiricamente), o intuito dessa etapa é eliminar objetos que não sejam regiões consideradas candidatas a serem faixas de pedestre.

Depois deste momento o sistema trabalha com uma imagem binária considerada ideal para operações de filtros morfológicos (LAGANIÈRE, 2011). Para eliminação de pequenos orifícios é realizada uma operação de fechamento seguida da uma operação de abertura. Ambas utilizando um elemento estruturante quadrado de 5x5 (máscara definida empiricamente).

Após a finalização do pré-processamento o sistema inicia a detecção de regiões que possam ser faixas de interesse através da procura de contornos de imagem utilizando a função *Imgproc.findContours* (OpenCV WebSite, 2012). Os contornos encontrados são validados de acordo com regras estipuladas para verificação se o mesmo é um candidato a pertencer a uma faixa de pedestre. Para a validação de que a região se trata de uma faixa de pedestre é necessário que três contornos ao menos sejam considerados válidos. Este número foi definido, pois se verificou que é muito comum que um dos componentes não esteja de acordo com as regras pré-

estabelecidas devido a condições como desgastes e iluminação.

A validação dos contornos é efetuada através dos seguintes critérios:

A. O contorno para ser considerado válido deve ter uma área com um tamanho de pelo menos 1.400 pixels.

B. Calcula parâmetros como largura e altura e faz uma nova validação verificando valores mínimos de pelo menos 150 pixels de largura ou altura.

C. Determina um conjunto de momentos que, de acordo com BRADSKI et al. (2008), são informações que integram todos os pixels da região. Esses valores são calculados conforme demonstra a equação (1), onde p é a ordem-x e q a ordem y e ordem significa a potência na qual o componente correspondente é elevado na soma.

$$m_{p,q} = \sum I(x,y)\, x^P\, y^q \qquad (1)$$

D. As informações são computadas para a extração dos Momentos de Hu que representam diferentes aspectos da imagem de forma invariante em escala, rotação e reflexão (BRADSKI, 2008). O sistema atualmente utiliza somente o componente zero do vetor calculado e faz a comparação para remoção de contornos que tenham esse valor < 0.5. Dessa forma são eliminados vários contornos que na verdade são somente ruídos para a aplicação.

V. Características típicas da arquitetura Android

Conforme descrição do Google (2007) o Android é uma plataforma aberta para dispositivos móveis. A arquitetura utiliza como Kernel o sistema operacional Linux versão 2.6 e juntamente são fornecidas bibliotecas C++ para atividades como acesso de Hardware e gerenciamento de energia (MEIER, 2009).

Além do Kernel baseado em Linux outro elemento chave do Android é a máquina virtual Dalvik que não se trata de uma máquina virtual Java como a Java ME, mas de uma implementação totalmente customizada garantindo que múltiplas instâncias possam ser executadas eficientemente em um único dispositivo. A máquina virtual Dalvik utiliza o Kernel Linux para funcionalidades de baixo nível e tem como responsabilidade realizar a interface entre as aplicações desenvolvidas e o Kernel (MEIER, 2009). Desenvolvedores podem utilizar essa camada intermediária através do uso do SDK disponibilizado pelo Google.

É possível escrever códigos para acesso ao Kernel diretamente em C++ utilizando então uma interface JNI para a comunicação entre o componente escrito com código nativo e o restante da aplicação. No entanto o uso desse tipo de técnica tem o problema de criar um código dependente de hardware embora possa haver diminuição no custo de processamento. A descrição de como utilizar o NDK do Google para desenvolvimento destes componentes está fora do escopo deste artigo.

A partir do OpenCV versão 2.3 foi disponibilizada uma interface JNI (OpenCV website, 2011) para fazer essa comunicação entre as bibliotecas do OpenCV escritas em C++ com a máquina virtual Dalvik. Dessa forma não é necessário o desenvolvimento de códigos nativos para o processamento de imagens em aplicativos móveis, embora seja recomendável verificar se existe algum ganho de desempenho significativo no sistema ao utilizar código nativo. A figura 6 mostra o fluxo de execução do OpenCV dentro do sistema Android.

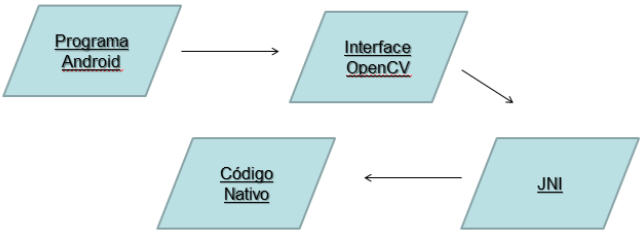

Figura 6 Arquitetura para execução do OpenCV no sistema Android

Acordo com Meier (2009) para escrever códigos eficientes em Android é necessário ter em mente que dispositivos móveis não possuem os mesmos recursos computacionais de servidores e computadores tradicionais. Aplicações neste tipo de ambiente precisam ser rápidas e eficientes do contrário podem ser interrompidas pelo sistema operacional.

Quando a tarefa envolve o processamento de imagens considera-se sempre a liberação de recursos tão logo quanto possível. Portanto após ser utilizada a imagem é descartada, o que muitas vezes não é uma preocupação no desenvolvimento em outros ambientes.

Nos experimentos feitos neste trabalho verificou-se a possibilidade do uso da câmera utilizando tanto as APIs do OpenCV quanto do Android. As bibliotecas do OpenCV tiveram um desempenho melhor e, por isso, foram utilizadas.

Transmissão de informação por voz

Para a transmissão da informação por voz são utilizadas as próprias bibliotecas disponibilizadas pelo SDK do Android, porém destaca-se que, por padrão, o Android não disponibiliza um sintetizador de voz na língua Portuguesa. Logo foi utilizado o SVOX (PFISTER, 1995) para que o sistema pudesse funcionar utilizando o Português.

O SVOX é um sintetizador de voz disponível em diversos idiomas além do Português, no entanto destaca-se que o mesmo não é gratuito, pelo menos durante o período que esse artigo foi escrito. Mais informações podem ser obtidas no website do produto.

VI. Metodologia Para testes

A ideia inicial era de se utilizar o equipamento móvel da Motorola modelo Defy 2.2, porém não foi possível utilizar este aparelho, pois a câmera não podia ser acessada pelo

OpenCV para o processamento de informações. No site oficial da biblioteca OpenCV existe uma nota indicando que este problema ocorre em alguns aparelhos com sistema Android versão 2.2. Para versões abaixo de 2.2 o OpenCV não oferece suporte. Portanto, para os testes do sistema, utilizou-se um aparelho Samsung modelo Galaxy S2 com o Android 2.3.

O algoritmo de processamento de imagens foi testado de duas formas diferentes: testava-se o aplicativo em situações reais de uso executando o protótipo do sistema em locais que possuíam ou não faixa de pedestre a fim de verificar o comportamento do programa desenvolvido e também foi feita uma validação com uma base de dados de fotos de situações armazenadas pelo aplicativo.

Foram escolhidas 41 cenários de uso em diversas condições para verificar a robustez do sistema. Exemplos são mostrados na Figura 7.

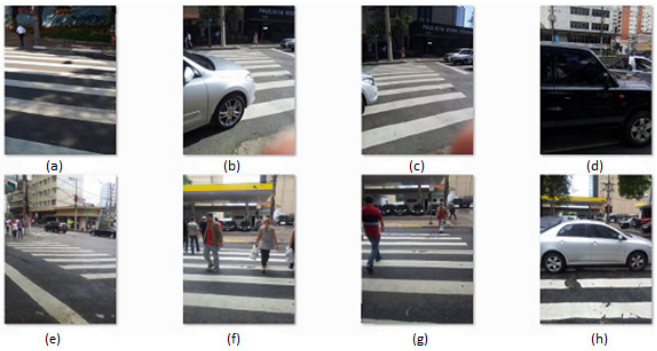

Figura 7 Exemplos de cenários testados pela aplicação

a) Faixas de pedestres com condições adversas de iluminação.
b) Cenário de travessia com carro.
c) Variação do cenário já exposto pela figura 7b.
d) Imagem sem faixa de pedestre.
e) Imagem de faixa de pedestre tirada de um ângulo diferente.
f) Cenário de travessia com pedestres incluídos.
g) Cenário de travessia com pedestres incluídos.
h) Cenário de travessia com condições adversas de iluminação e carros na faixa de pedestre.

VII. RESULTADOS

Foram utilizados 41 cenários nos testes do sistema, em 31 destes o sistema detectou corretamente a faixa de pedestres, porém em 10 o sistema falhou na detecção. Essas situações foram armazenadas com o intuito de corrigir o algoritmo para correta interpretação.

O reconhecimento de faixas de pedestre foi feito mesmo quando existiam pessoas ou carros sobre a faixa. Outro aspecto que deve ser analisado no futuro é a correlação entre alguns destes objetos e a faixa de pedestre, principalmente no que diz respeito à travessia segura de deficientes visuais.

A figura 8 apresenta exemplos de faixas que foram reconhecidas pelo sistema.

Figura 8 Faixas de Pedestres reconhecidas pelo sistema

a) Faixa de pedestre com condição de luminosidade constante.
b) Faixa de pedestre mesmo contendo pessoas também foi reconhecida pelo sistema. A presença de pessoas na cena também pode contribuir como um indicativo ao sistema de que a travessia do deficiente visual tenha um maior grau de segurança.
c) O sistema também teve um bom comportamento em relação ao reconhecimento de imagens que não possuíam faixa de pedestre.
d) O sistema reconheceu imagens com condições de tempo adversas.

Situações típicas em que o sistema apresentou problemas para reconhecimento de faixa de pedestres são mostradas na figura 9:

Figura 9 Exemplos de fotos sem reconhecimento adequado

a) Grandes variações de iluminação na imagem, devido, por exemplo, a sombras.
b) Iluminação refletida em um carro – O sistema somente reconheceu carros como parte da faixa de pedestre quando carros de cores claras tinham reflexão da luz do sol.
c) Faixas com grandes desgastes não foram reconhecidas pelo sistema.
d) O sistema reconhece faixas de pedestres que carros estejam atravessando, porém para a aplicação solicitada verifica-se que isso na verdade é um problema. Devido a isso se conclui a necessidade de reconhecimento de carros para melhora do algoritmo.

A figura 10b mostra o resultado do algoritmo na detecção de uma faixa de pedestre na figura 10a. Para casos onde o sistema não conseguiu efetuar a detecção a imagem de saída não indicava resultado algum. O sistema também emitia um aviso sonoro indicado se a faixa havia sido detectada ou não.

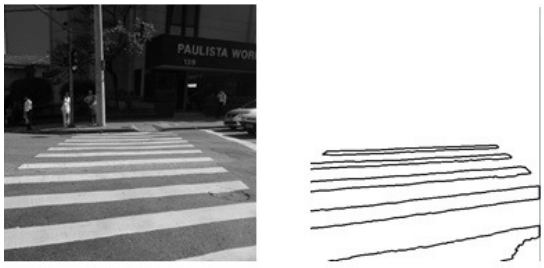

Figura 10 Reconhecimento efetuado pelo sistema

Quanto ao tempo de processamento foi definido como um dos objetivos do sistema que a detecção da faixa de pedestres fosse realizada num espaço de tempo de até 1 segundo. O tempo medido do sistema em plataforma Android ficou entre 330 e 500 milissegundos. No desktop em plataforma Windows o sistema executou a detecção de faixas de pedestre com tempo de processamento entre 88 e 140 milissegundos.

CONCLUSÕES

O sistema aqui apresentado, em seus testes iniciais, apresentou um desempenho geral da ordem de 76%, indicando a viabilidade de um sistema para detecção de faixa de pedestres em dispositivo móvel para o auxilio a deficientes visuais.

Os problemas apresentados pelo aplicativo ocorreram principalmente nas imagens que apresentavam variações climáticas e de iluminação muito grandes. O que se caracteriza como um obstáculo para que o sistema seja implantado em cenários reais. Também é necessário aumentar a quantidade de amostras de testes para a garantia de que o sistema está funcionando de forma robusta.

O tempo total de processamento na plataforma móvel ficou dentro do tempo especificado para o sistema, indicando que é possível melhorar o processamento e ainda assim manter o tempo dentro do especificado.

A interface é simples permitindo o uso por usuários não especializados.

Dessa forma o projeto contribui na investigação de questões presentes no universo da visão computacional para dispositivos móveis em cenários não controlados e a criação de interfaces para deficientes visuais.

TRABALHOS FUTUROS

O trabalho teve resultados importantes, mas para o desenvolvimento de um sistema robusto é necessário a continuação do desenvolvimento do mesmo. Conforme já citado pretende-se estender o algoritmo para reconhecimento de outros sinais que indiquem se é possível ou não atravessar uma rua. Esses sinais podem ser pessoas, carros e faróis de pedestre que auxiliarão então o aplicativo a tomar uma melhor decisão, pretende-se para isso utilizar o uso do algoritmo de classificação AdaBoost juntamente com o LBP para seleção de características.

O algoritmo de faixa de pedestre ainda possui interferência de mudanças de iluminação e estuda-se o uso de limiares adaptativos que sejam determinados de acordo com o ambiente para resolução desse problema.

Quanto à interface haverá um serviço para reconhecimento de voz no celular. É importante considerar que o serviço de reconhecimento de voz não pode utilizar Internet a fim de que o aplicativo funcione desconectado e por isso não é possível utilizar o serviço padrão do Google.

Por fim embora o tempo de resposta do sistema esteja satisfatório nestes testes iniciais serão feitos outros testes para comparação desses tempos utilizando código nativo no lugar da utilização da interface JNI disponibilizada pelo OpenCV 2.3.

REFERÊNCIAS

[1] Robert Laganiére, OpenCV 2 Computer Vision. Application Programming Cookbook, 2011.

[2] Gary Bradski and Adrian Kaehler, “Learning OpenCV”. O’REILLY, 2008.

[3] Jo Chang-yeon. Face Detection using LBP features. Dezembro 2008.

[4] B. Bhargava, P. Angin, L. Duan. A Mobile-Cloud Pedestrian Crossing Guide for the Blind. International Conference on Advances in Computing & Communication, 2011.

[5] SE S. Zebra-crossing detection for the partially sighted. Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2000). Hilton Head, 2000.

[6] J. Coughlan and H. Shen. “A Fast Algorithm for Finding Crosswalks using Figure-Ground Segmentation.” 2nd Workshop on Applications of Computer Vision, in conjunction with ECCV 2006. Graz, Austria. May 2006.

[7] L. Lausser, F. Schwenker, and G. Palm, "Detecting zebra crossings utilizing AdaBoost", in Proc. ESANN, 2008, pp.535-540.

[8] Yoav Freund and Robert E. Schapire, "A Short Introduction to Boosting", Journal of Japanese Society for Artificial Intelligence, 14(5):771-780, September, 1999.

[9] Uddin, M.S., Shioyama, T.: Robust Zebra-Crossing Detection using Bipolarity and Projective Invariant. In: Procs. 8th Intl. Symp. on Signal Processing and Its Applications, Sidney, Australia (August 2005) 571–574

[10] Paul Viola and Michael Jones. “Rapid Object Detection using a Boosted Cascade of Simple Features", Computer Vision and Pattern Recognition, 2001.

[11] IBGE. Disponível em: <http://www.ibge.gov.br/home/presidencia/noticias/27062003censo.shtm>. Acessado em: 25 de setembro de 2011

[12] Android developers oficial website. Disponível em: http://developer.android.com/index.html. Acessado em: 12 de janeiro de 2012.

[13] Gartner website. Disponível em: http://www.gartner.com/it/page.jsp?id=1848514. Acessado em: 27 de fevereiro de 2012.

[14] SVOX web site. Disponível em: http://svoxmobilevoices.wordpress.com/. Acessado em: 28 de fevereiro de 2012.

[15] OpenCV web site. Disponível em: http://opencv.willowgarage.com/wiki/. Acessado em: 27 de fevereiro de 2012.

[16] Reto Meier. Professional Android Application Development, 2009, WROX.

[17] Google Blog web site. Disponível em: http://googleblog.blogspot.com/2007/11/wheres-my-gphone.html. Acessado em: 27 de fevereiro de 2012.

[18] Beat Pfister. The SVOX Text-to-Speech System. Swiss Federal Institute of Technology Zurich, 1995.