Revista de Informática Aplicada, Volume 11, Número 1, 2015

05

Aplicativo identificador de cédulas para deficientes visuais

Victor Vequetini Teixeira[1] Juliana Cristina Braga[1] Marcelo Zanchetta do Nascimento[1]

Resumo
Existe um grande número de deficientes visuais no Brasil, mas muitos deles ainda encontram problemas de acessibilidade. Muitas dessas dificuldades podem diminuir por meio do uso das tecnologias assistivas. O potencial de uso dessas tecnologias pelos deficientes visuais é grande, mas poucos são os esforços para que elas sejam desenvolvidas. Em face da necessidade de desenvolvimento de tecnologias assistivas que auxiliam na vida dos deficientes visuais, juntamente com a importância e benefícios que elas trazem para essas pessoas, o objetivo deste estudo é contribuir para a inclusão social e digital dos deficientes visuais brasileiros propondo um algoritmo capaz de auxiliar os deficientes visuais no reconhecimento do valor de cédulas monetárias.

Palavras-Chave: Identificador de cédulas; Acessibilidade; Deficientes visuais; Android, Tecnologias Assistivas.

Abstract
There are many visually impaired people in Brazil, and most of them still have troubles with accessibility. It's possible to help with most of those difficulties using assistive technologies. The potential of visually impaired people using these technologies is good, but there are few efforts to develop them. Knowing the necessity to develop assistive technologies to help visually impaired people in their daily tasks and the benefits it might bring to them, the objective of this work is to contribute to the social and digital inclusion of visually impaired brazilians, proposing an algorithm capable of help them in recognize money bills.

Keywords: Cell Identifier; Accessibility; Visually Impaired; Android; Assistive Technologies.

[1]Universidade Federal do ABC
{victor.teixeira@aluno.ufabc.edu.br, juliana.bragaufabc.edu.br, marcelo.nascimento@ufu.br}

1.Introdução

Segundo o censo do Instituto Brasileiro de Geografia e Estatística (IBGE) de 2010, havia no Brasil uma população de mais de 190 milhões de pessoas, sendo que, dessas, 45,6 milhões (23,9%) apresentaram algum tipo de deficiência. Dentre o número de deficientes, há cerca de 35,8 milhões de pessoas com dificuldade para enxergar, sendo que 506,3 mil se declararam totalmente cegos. Apesar do grande número de deficientes visuais em nosso país, muitos deles ainda encontram problemas de acessibilidade como, por exemplo, dificuldades ao caminhar pelas ruas, realizar compras, utilizar recursos tecnológicos como: computadores, celulares, *tablets*, etc.

Algumas das dificuldades encontradas pelos deficientes visuais podem diminuir por meio do uso de Tecnologias de Informação e Comunicação (TICs). A apropriação de recursos tecnológicos modifica significativamente o estilo de vida, as interações e a conduta social ao inovar hábitos e atitudes em relação à educação, ao lazer e ao trabalho, à vida familiar e comunitária. Um exemplo disso são os cegos ou com baixa visão que usam os computadores para ler jornais, realizar pesquisas acadêmicas, fazer inscrição em concursos públicos, verificar resultados, ou simplesmente para treinar a digitação e o domínio do teclado. Desde a invenção do Código Braille, em 1829, nada teve tanto impacto nos programas de educação, reabilitação e emprego quanto o desenvolvimento dessas tecnologias para deficientes visuais (Campbell, 2001). Isso pode ser justificado pelo fato das TICs contribuírem para que esses deficientes tenham maior independência, qualidade de vida e inclusão na vida social por meio do suplemento, manutenção ou devolução de suas capacidades funcionais. As TICs ampliam as possibilidades de comunicação e de autonomia pessoal, minimizam ou compensam as restrições decorrentes da falta da visão. De acordo o livro *Internet para necessidades visuais* (UTDAD/ Guia, 1999), para a maioria das pessoas a tecnologia torna a vida mais fácil, para uma pessoa com necessidades especiais, a tecnologia torna as coisas possíveis.

Um recente estudo (BRAGA, 2012) realizado pelo grupo de pesquisa Intera, da UFABC, teve como objetivo estudar o uso das TICs no cotidiano dos deficientes visuais e identificar demandas para o desenvolvimento de novas tecnologias. Um dos resultados dessa pesquisa foi a identificação da necessidade de um aplicativo que fosse capaz de identificar as cédulas monetárias (Real). Os entrevistados relataram uma grande dificuldade em identificar os valores das notas, sendo que essa dificuldade implica na exclusão social desses deficientes em realizar tarefas corriqueiras como: realização de compras, pagamento de meios de transporte (taxi), realização de serviços bancários, etc. Alguns deficientes também relataram terem sofrido golpes motivados pelo fato de não conseguirem identificar os valores das cédulas financeiras.

2. Objetivos

Diante da necessidade de desenvolvimento de TICs que auxiliem na vida dos deficientes visuais, juntamente com a importância e benefícios que elas trazem para essas pessoas, o objetivo desse estudo é contribuir para a inclusão social e digital dos deficientes visuais brasileiros. Para tal, o presente trabalho propõe o desenvolvimento de um método capaz de auxiliar os deficientes visuais no reconhecimento do valor de cédulas monetárias. O método proposto foi dividido em três principais etapas: o pré-processamento da nota a ser reconhecida, a extração de características da imagem e, por último, sua classificação. O pré-processamento consiste na conversão da imagem para escalas de cinza. A etapa de extração de características é fundamentada no algoritmo SURF (BAY, 2008), que procura por pontos na imagem que são invariáveis, independentemente de como a imagem é capturada (distância da nota, rotação, iluminação). Na última etapa, o classificador SVM (CORTES, 1995) é aplicado nas características encontradas para o reconhecimento do valor da nota. Um banco de imagens contendo 1440 imagens foi utilizado na etapa de avaliação do método proposto.

3. Revisão bibliográfica

A. Trabalhos relacionados

Esta seção revisa alguns trabalhos relacionados, identifica o ponto de convergência e divergência com essa pesquisa.

1) LookTel Money Reader (*LOOKTEL* e l , 2013): aplicativo desenvolvido para *iPhone* (*APPLE*, 2013), que identifica e informa o usuário de forma auditiva o valor da nota (dólar americano), usando a câmera do aparelho, em tempo real. Semelhante ao aplicativo proposto, porém, será desenvolvido para a plataforma *Android* (*ANDROID*, 2013), e funcionará

para a moeda brasileira (Real), além da funcionalidade um pouco diferente.

2) Sistema de localização ubíquo acessível a deficientes visuais (Siller; Braga, 2010): propuseram uma arquitetura para um sistema de localização global embutido em um celular, facilmente acessível a deficientes visuais, por meio da recepção e emissão de comandos de voz. *É uma TIC* para auxiliar deficiêntes visuais como o projeto proposto e também foi desenvolvido na mesma plataforma (*Android*) e em linguagem *Java* (*JAVA*, 2013).

3) Uma metodologia para detectar e reconhecer placas de sinalização de trânsito (SILVA *et al*., 2012)*:* trabalho que utilizou o algoritmo SIFT (*Scale Invariant Feature Transform*) (LOWE, 2004) para reconhecimento de placas de trânsito. O projeto proposto utiliza uma técnica semelhante para a detecção das cédulas, o SURF (*Speeded Up Robust Features*) (BAY *et al*., 2010.

B. Fundamentação teórica

O presente trabalho deverá se fundamentar em dois tipos referências: o primeiro são os algoritmos de processamento e classificação de imagens, e a segunda são as bibliotecas utilizadas para o processamento de imagens e de reconhecimento e emissão de voz para *Android*. A seguir uma breve descrição teórica das referências que norteiam esse projeto.

1) Speeded Up Robust Features (SURF) (BAY *et al*., 2010): *técnica de detecção de atributos de uma imagem.* Muito utilizada em reconhecimento de objetos e reconstrução 3D. A técnica procura detectar e descrever pontos chave (*keypoints*) da imagem, para uma futura classificação ou reconstrução 3D. Cada ponto-chave (*keypoint*) é selecionado de uma localização distinta da imagem, e sua característica mais importante é poder ser detectado independentemente da condição em que a imagem foi capturada (ângulo, distancia, rotação, iluminação diferentes). A seguir, a vizinhança do *keypoint* é representada por um vetor de atributos (*descriptors*), que devem ser distintos, robustos a ruídos, erros de detecção e deformações geométricas. Os *descriptors* podem então ser comparados e classificados para o reconhecimento da imagem.

2) Support Vector Machine (SVM) (CORTES; VAPNIK, 1995; LORENA; CARVALHO, 2007): técnica computacional de aprendizado para classificação de dados, originalmente binária (linear) e posteriormente aprimorada para classificação de mais de uma classe. *É baseda na separação ótima de classes, ou seja, procura separar o máximo possível dados de classes diferentes em um hiperplano. É capaz de analizar e reconhecer padrões nos dados, separando-os em* classes (ou categorias). Para tal é necessário o treinamento da SVM, apresentando um conjunto de dados e informando a qual categoria cada um pertence. Após o treinamento, a SVM é capaz de classificar novos dados, determinando a qual categoria ele pertence. Para os testes deste trabalho foi utilizada uma *Sequential minimal optimization* (SMO), algoritmo que apresenta uma melhoria nos problemas de otimização da SVM, disponível no *software Weka* (*WAIKATO*, 2013), e também capaz de ser implementada utilizando-se a biblioteca *OpenCV* (*OPENCV*, 2013).

3) Open Source Computer Vision Library (OpenCV) (*OPENCV*, 2013): biblioteca de código aberto, livre tanto para uso comercial como acadêmico. Possui interfaces em C++, C, Python e Java, além de suportar Windows, Linux, Mac OS, iOS e *Android*. Foi desenvolvida para ser eficiente e foca em aplicações em tempo real. Foi escolhida para este projeto pela sua vasta funcionalidade, sendo que possui funções para utilização do algoritmo SURF, como também funções para implementar uma SVM. É compativel e eficiente com aplicativos para *Android.*

4) Android (*ANDROID*, 2013): sistema operacional móvel que roda sobre o núcleo Linux, sendo inicialmente desenvolvido pela Google e posteriormente pela *Open Handset Alliance*. Entretanto, a Google é a responsável pela gerência do produto e engenharia de processos. O *Android* SDK (*Software Development Kit*) fornece as ferramentas e APIs necessárias para o desenvolvimento de aplicativos nesse sistema, por meio da linguagem de programação Java, incluindo um emulador de dispositivo móvel, isto é, um dispositivo móvel virtual que roda no computador. O emulador permite desenvolver e testar aplicativos *Android* sem o uso de um dispositivo físico, porém com algumas

limitações, uma vez que não suporta, por exemplo, chamadas de áudio e suporte para câmera e captura de vídeo. Os principais componentes da arquitetura do sistema operacional *Android são: applications, escritos em linguagem Java e que incluem: cliente de e-mail, programa SMS, calendário, mapas, navegador, contatos e outros; aplication framework*, que permite aos desenvolvedores aproveitar o dispositivo do *hardware* e informações de localização de acesso, executar serviços de fundo, definir alarmes e notificações para adicionar a barra de *status*; libraries, que inclui um conjunto de bibliotecas C / C + + usadas por diversos componentes do sistema *Android*; *Android Runtime*, que inclui um conjunto de bibliotecas que fornece a maioria das funcionalidades disponíveis nas principais bibliotecas da linguagem de programação Java; Linux Kernel, que atua como um sistema central de serviços, segurança, gerenciamento de memória, gerenciamento de processos, rede de pilha, modelo e *driver*. O Kernel também atua como uma camada de abstração entre o *hardware* e o restante da pilha de *software* (*ANDROID*, 2013).
Cada aplicação *Android* roda em seu próprio processo, com sua própria instância da máquina virtual Dalvik. Esta executa os arquivos em formato Dalvik executável (.DEX), que é otimizado para o mínimo consumo de memória.

5) Waikato Environment for Knowledge Analysis (Weka) (*WAIKATO*, 2013): *software* criado na Universidade de Waikato, Nova Zelândia, que agrega várias técnicas de aprendizado de máquina. É escrito em Java e seu código é aberto. Foi utilizado para os testes de classificação de *keypoints* e *descriptors* realizados neste trabalho.

4. Metodologia

O objetivo desse trabalho foi identificar qual o melhor algoritmo de processamento de imagem para reconhecimento de cédulas. Com intuito de atingir esse objetivo, as seguintes etapas foram realizadas:
A. Estudo das técnicas de processamento de imagem que poderiam ser utilizadas
Algumas técnicas de processamento e classificação de imagem foram estudadas, porém substituídas pela técnica atual, que será descrita nas próximas seções. Dentre as técnicas estudadas podemos destacar:

1) Algoritmo de pré-processamento (descartado)

- Converter a imagem para escalas de cinza;
- Aplicar um filtro (dentre os testados, temos o filtro gaussiano, da mediana e anisotrópico [23]);
- Binarização a partir de um *threshold* (foi testado o método de *Otsu*) (GONZALES; WOODS, 2010);
- Inverter a imagem, ou seja, passar os bits brancos para preto e os pretos para branco;
- Aplicar um filtro morfológico para remover ruídos (erosão e dilatação) (GONZALES; WOODS, 2010);
- Detecção de contornos (método de *Canny*) (VALE; POZ, 2002), visando a detectar as regiões de interesse da imagem (os números da nota);

Utilizavam-se as regiões detectadas na etapa seguinte, a classificação.

2) Classificações testadas (e descartadas)

- Calcular os *Hu Invariant Moments* (sete valores calculados a partir de uma imagem que, teoricamente, são constantes independente de sua orientação, escala e translação. As equações do cálculo podem ser encontradas em (*OPENCV's k-nearest*, 2013) das regiões de interesse e do banco de imagens. Após calcular, fazer a subtração de ambos e procurar pelo valor mais próximo de zero para que ocorra assim a identificação.
- Utilizar o classificador *k-nearest-neighbor* (KNN) (*OPENCV's k-nearest*, 2013) para filtrar quais pares de *descriptors* são semelhantes, comparando assim os *descriptors* da foto com os *descriptors* obtidos do banco de imagens. O valor de *k* utilizado foi 2. A identificação ocorre para a nota com maior número de *descriptors* semelhantes. Essa forma de classificação foi testada antes da SVM e obteve resultados com muitos falso-positivos.
- Utilizar o conjunto de *keypoints* para a classificação na SVM. Os resultados obtidos utilizando-se o conjunto de *keypoints*, em vez do conjunto de *descriptors*, foram inferiores. De um total de 1440 fotos, temos 718 (49.8611%) identificações corretas e 722 (50.1389%) incorretas. O resultado, se comparado ao resultado da classificação do *dataset* de *descriptors* (ver seção VI. Resultados para uma descrição mais detalhada), é muito inferior.

B. Estudo e definição do ambiente de programação a ser utilizado

O trabalho foi desenvolvido em linguagem Java na plataforma *Android* e utiliza a biblioteca *OpenCV* para o processamento (*e.g.* conversão para escalas de cinza) e teste de diferentes técnicas para a identificação da imagem. Para a captura da imagem e testes dos diferentes métodos de identificação foi criado um protótipo do aplicativo. Para os testes de classificação, foi utilizado o *software* Weka (ver III. 5), em um momento inicial, e depois a SVM foi implementada no protótipo do aplicativo (utilizando a biblioteca *OpenCV*) para averiguar sua eficiência na prática.

C. Preparação da base de dados

O banco de imagens foi elaborado com 1440 imagens digitais, sendo 240 para cada cédula monetária brasileira (primeira família de cédulas do Real[1], com exceção da cédula de um real). As imagens foram capturadas em um ambiente controlado, com iluminação artificial de uma lâmpada fluorescente de 20 Watts. O dispositivo utilizado na captura foi um celular[2] com uma câmera com resolução máxima de 5 *megapixels*. O tamanho da imagem é de 2592 x 1944 pixels, com extensão JPEG e no padrão sRGB, sendo a resolução horizontal e vertical da imagem de 72 dpi e a intensidade de 24 bits.

No banco foram utilizadas 30 imagens de cada cédula (valor monetário: dois, cinco, vinte, cinqüenta e cem reais), sendo que cada nota possui oito imagens, variando a posição, horizontal e vertical em relação à câmera, o lado, frente e verso da nota, e uma distância de 10 e 15 centímetros na captura com a câmera. A seguir há uma tabela para exemplificar as diferentes imagens de cada um dos conjuntos de cédulas de um mesmo valor:

Tabela 1:Esquema da variação de imagens para cada nota

Notas	*Frente*				*Verso*			
	Vertical		**Horizontal**		**Vertical**		**Horizontal**	
	10 cm	**15 cm**	**10 cm**	**15 cm**	**10 cm**	**15 cm**	**10 cm**	**15 cm**

Nota1	Imagem1	Imagem2	Imagem3	Imagem4	Imagem5	Imagem6	Imagem7	Imagem8
Nota2	Imagem1	Imagem2	Imagem3	Imagem4	Imagem5	Imagem6	Imagem7	Imagem8
...								
Nota30	Imagem1	Imagem2	Imagem3	Imagem4	Imagem5	Imagem6	Imagem7	Imagem8

A Tabela I demonstra como foram capturadas as imagens de cada nota, por exemplo, para o conjunto da cédula de dois reais, temos 30 notas (coluna **Notas**), ou exemplares. Para cada nota (ou exemplar) foram capturadas oito imagens, quatros delas da frente da nota (Imagem 1 a Imagem 4) e outras quatro do verso da nota (Imagem 5 a Imagem 8). Dentre as imagens, ainda há duas subdivisões: a posição em que a nota se encontra em relação à câmera (horizontal à câmera ou vertical à câmera), e também à distância em que a imagem foi capturada (10 ou 15 centímetros).

D. Implementação do algoritmo proposto

1) Algoritmo do protótipo

As principais etapas do algoritmo são: pré-processamento, extração de características e classificação. A Figura 1 ilustra o algoritmo de forma simplificada.

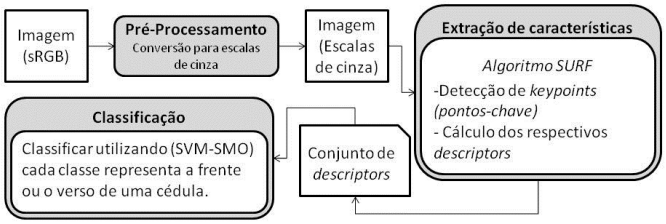

Figura 1. Diagrama do funcionamento simplificado do algoritmo.

A seguir, uma breve descrição do algoritmo implementado no protótipo:

- Capturar a foto ao tocar a tela. Foi utilizado a *Camera API (Application Programming Interface)*, inclusa no próprio *Android*. Um guia para a implementação das funções da câmera está disponível na documentação do *Android* (*CAMERA Developer*

1. Cédulas monetárias em circulação no Brasil, estão sendo substituídas aos poucos pela Segunda Família do Real, porém continuarão válidas. A Primeira Família do Real conta com as notas de um (fora de circulação), dois, cinco, dez, vinte, cinquenta e cem reais. (Primeira Família do Real, 2013)

2. Neste trabalho utilizamos um celular Sony Xperia U (2013).

Guide, 2013);

- Converter a imagem para escalas de cinza, utilizando o método "*Imgproc.cvtColor()*" (DOCUMENTAÇÃO *OpenCV*, 2013), disponível na biblioteca *OpenCV* (versão 2.4.0);
- Utilizar o algoritmo SURF para detecção de *keypoints*. A função utilizada, "*FeatureDetector. detect()*", pode ser encontrada na biblioteca *OpenCV* (versão 2.4.0) [19];
- Calcular um vetor *descriptor* para cada *keypoint* detectado. Foi utilizada a função "*DescriptorExtractor. compute()*", contida na biblioteca *OpenCV* (versão 2.4.0) (*OPENCV's DescriptorExtractor*, 2013);
- Classificar a foto utilizando o conjunto de *descriptors,* para isso uma SVM foi treinada e salva no formato XML para ser posteriormente carregada e utilizada na classificação durante a execução do aplicativo.
- Informar o valor resultante da classificação de forma sonora.

A Figura 2 contém o diagrama de classes do protótipo implementado, de forma simplificada. Note que a classe *Main* contém variáveis estáticas, tais como a que armazena a SVM treinada, uma variável que armazena a foto capturada durante a execução, para processá-la e identificá-la, entre outras.

A classe *TuplaKD* serve para modelar objetos que contenham as informações referentes a cada conjunto *keypoint* e *descriptor* encontrado na imagem capturada. O método desta classe "*compareTo*" foi sobrescrito para que a comparação seja feita pelo valor de maior *keypointHessian.* Os principais métodos de processamento estão contidos na classe *Processar*. A interação com o usuário de forma sonora fica por conta da classe *AppSounds*.

O código Java das classes do protótipo é descrito no Apêndice deste projeto.

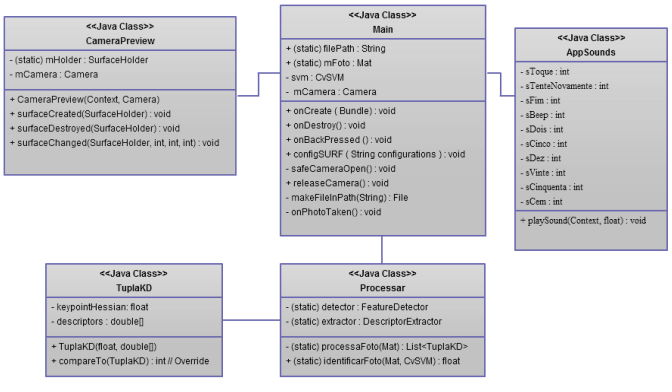

Figura 2. Diagrama de classes do protótipo.

2) Pré-processamento

A etapa de pré-processamento consistiu na conversão da imagem contendo a nota do espaço de cores sRGB para escalas de cinza. Então, foi utilizado a função "*cvtColor( )*" contida na biblioteca OpenCV que utiliza Equação (1) para a conversão (BRADSKI, 2008).

$$Y=(0.299)*R+(0.587)*G+(0.114)*B \quad (1)$$

Sendo, R, G e B os respectivos valores dos canais *red*, *green* e *blue* do pixel no espaço sRGB. Note que o valor *Alpha* não é utilizado na etapa dessa conversão.

3) Extração de características utilizando o algoritmo SURF (BAY *et al*.; VIOLA; JONES, 2002)

O algoritmo SURF (*Speeded Up Robust Features*) (BAY *et al*., 2009; CORTES, 1995) é um descritor invariante à rotação e aos efeitos de escala apresentados recentemente por Bay (2008). Essa técnica procura detectar e descrever pontos-chave (*keypoints*) da imagem para etapa de classificação.

Cada *keypoint* é selecionado de uma localização distinta da imagem. Sua característica mais importante é poder ser detectado, independentemente da condição em que a imagem foi capturada, ou seja, em escala ou rotação diferente. A vizinhança do *keypoint* é representada por um vetor de atributos denominados *descriptors*, que são distintos, robustos a ruídos, a erros de detecção e deformações geométricas.

A detecção de *keypoints* baseia-se em aproximações da matriz Hessiana, para um dado ponto na imagem, que deve estar em escalas de cinza. São detectadas estruturas em forma de borrão (*blob*), onde o determinante da matriz é máximo. A seguir, as respostas das *wavelets* de *Haar* foram calculadas em torno do *keypoint* para determinar sua orientação dominante.

A descrição de características do *keypoint* pode ser dividida em duas principais etapas. A primeira é calcular a orientação resultante para uma dada região circular em torno do *keypoint* que seja invariante a rotação. Para tal, as respostas das *wavelets* de *Haar* (MALLAT, 1996) dentro da região circular dada são calculadas tanto na horizontal como na vertical. A orientação local dominante é estimada pela soma de todas as respostas, dentro de uma fatia

da região circular de tamanho π/3, gerando um vetor da orientação. A orientação global do *keypoint* é determinada pelo maior vetor de orientação de todas as fatias. A segunda consiste em criar o *descriptor*, que será um vetor de valores. Primeiramente é construída uma região quadrada centrada no *keypoint* e orientada pelo vetor de orientação descrito na etapa anterior. Um filtro Gaussiano é aplicado (centrado no *keypoint*) para remover deformações geométricas. A região é dividida em quatro sub-regiões iguais, para cada sub-região são calculadas as respostas da *wavelet* de *Haar*, na horizontal e na vertical. O vetor *descriptor* resultante é a concatenação das quatro somas das respostas e seus respectivos módulos (chamemos de *i*), sendo *i* definido por (2).

$$i = (\sum d_x , \sum d_y , \sum |d_x| , \sum |d_y|) \qquad (2)$$

O *descriptor* resultante é um vetor de 64 valores. Para este trabalho foi utilizado uma variação do algoritmo que calcula esse vetor com 128 valores, o que deixa o processamento mais lento, porém os valores dos *descriptors* ficam mais distintos entre si, permitindo uma melhor definição de padrões para a classificação.

Neste estudo, foram elaborados o vetor contendo os atributos de cada *keypoint* da imagem e um segundo contendo os seus respectivos *descriptors*. Cada *keypoint* possui seis atributos, um ponto (localização de onde ele foi encontrado na imagem com coordenadas *x* e *y*), o tamanho (*size*) da característica, o ângulo, que varia de 0 a 360, o atributo *response* (ou *hessian*), que representa o quão proeminente um *keypoint* é, ou seja, *keypoints* com maior *response* são características mais robustas da imagem, o *octave* (ou oitavas, relacionado à escala em que o ponto foi detectado, pode estar relacionado ao atributo *size*) e *class_id*, relacionado com o valor laplaciano, sendo que pontos com os valores laplacianos diferentes não podem ser homólogos (*OPENCV*, 3012). Para cada *keypoint* deve ser calculado um *descriptor*, que, de maneira simplificada, é um vetor de 128 valores contendo a descrição do *keypoint* e sua vizinhança (BAY, 2008).

Seja um *keypoint* detectado em uma imagem "*k"*, seu respectivo *descriptor* representado por "*f(k)"*, tem-se para cada imagem um conjunto de dados formados pelos pares de *keypoint* e seu respectivo *descriptor*, representado por *"*(*k, f(k)*)".

Para a classificação é necessário um número igual de dados (*k, f(k)*) para cada imagem, porém o número *n* de (*k, f(k)*) detectados pelo algoritmo SURF varia de imagem para imagem. A solução proposta neste trabalho foi selecionar um número fixo de dados para cada imagem. Para tal, primeiramente foi verificado o número mínimo de detecções de uma imagem (neste trabalho a imagem com menos detecções possuía *n* = 89), depois foram selecionados os *n* melhores pares detectados de cada imagem. A seleção filtrou os *n* pares que possuem *keypoints* com os maiores valores de *threshold* dentro do conjunto, ou seja, os *keypoints* e seus respectivos *descriptors* mais robustos de cada imagem.

O grupo de dados resultante foi dividido em duas partes, um conjunto contendo apenas *keypoints* e outro contendo apenas *descriptors*. Esses dois conjuntos foram utilizados para a classificação na *SVM*.

4) Classificação SVM [10][11]

No contexto de classificação, o classificador *Support Vector Machines-Sequential Minimal Optimization* (*SVM-SMO*) (CORTES,1995; LORENA, 2007) vem sendo usado para a classificação de características calculadas por algoritmos como SURF e SIFT (*Scale-invariant feature transform*) (LOWE, 2004), como visto em Silva (2012).

O *SVM* é uma das técnicas de aprendizado de máquina e se baseia na teoria de aprendizagem estatística, tendo por objetivo a separação ótima de dados entre classes distintas, buscando sempre a maior minimização de erros possível. A separação ótima ocorre em um hiperplano condicional, de forma que o plano é orientado a maximizar a distância entre as bordas das diferentes classes, utilizando seus pontos mais próximos. Caso os dados não sejam linearmente separáveis, a *SVM* mapeia os dados utilizando uma função *Kernel* (neste trabalho foi utilizado um *polynomial kernel*), em um espaço de características de dimensão mais elevada, onde os dados se tornam linearmente separáveis (OSTA, 2008). A Figura 3 mostra um exemplo de separação de classes com o algoritmo *SVM*.

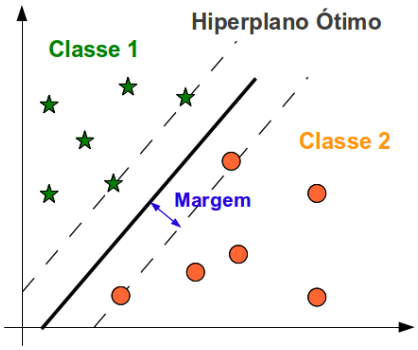

Figura 3. Ilustração da aplicação do algoritmo SVM. Fonte: Jacomini, et al. (2012).

Neste trabalho foi testado o conjunto de dados dos *keypoints* e *descriptors* de uma dada imagem. Nessa etapa, o método de rotação (*n-fold cross validation*) foi aplicado com o valor $n = 10$. Esse método utiliza 90% dos dados para treinamento e o restante, 10% dos dados para etapa de teste.

Para avaliar o desempenho do método proposto foram calculados os seguintes parâmetros: sensibilidade e especificidade. A sensibilidade representa a frequência com que o teste detectará o que está sendo testado, ou seja, um teste com sensibilidade de 90% dará um resultado positivo em 90 das 100 amostras que efetivamente apresentam o que está sendo investigado. A especificidade representa a frequência com que o resultado negativo está correto.

Também foi obtida a área sob a curva *ROC* (*Receiver Operating Characteristic*). A curva *ROC* (FAN *et al*., 2006. Understanding receiver operating characteristic (roc) curves. Canadian Journal of Emergency Medicine, 8(1), 19–20.) é criada em um gráfico de verdadeiro-positivos (eixo *y*) *versus* os falso-positivos (eixo *x*). A área abaixo da curva serve como medida de precisão do desempenho de um classificador, sendo que uma área de 1.0 uma classificação perfeita e 0.0 um classificação completamente errônea.

Tabela 2. Medida do desempenho da classificação.

Medida	Definição
Sensibilidade	Verdadeiro Positivo / Total de Positivos
Especificidade	Verdadeiro Negativo / Total de Negativos

E. Testes do algoritmo proposto

Para verificar a eficiência do algoritmo implementado (ver seção IV. D), alguns testes foram realizados. A partir dos testes foi possível detectar a forma de classificação com o melhor índice de identificações de notas corretas, ou seja, sem falso-positivos.

Os testes utilizaram o banco de imagens criado para procurar um padrão nas imagens capturadas pelo celular, e então, classificá-las. O primeiro programa criado para auxiliar nos testes foi um aplicativo *Android*, simulado no próprio emulador do *Android* SDK (*ANDROID Emulator*, 2013). O aplicativo tinha por objetivo processar as diversas fotos do banco de imagens e criar dois arquivos na extensão (.CSV) (*COMMA separated*, 2013) para cada foto: um contendo os atributos de cada *keypoint* contido na imagem e outro contendo os seus respectivos *descriptors*.

Cada *keypoint* possui seis atributos, um ponto (localização de onde ele foi encontrado na imagem com coordenadas *x* e *y*), o tamanho (*size*) da característica encontrada, o ângulo da característica (*angle*, que varia de 0 a 360), o atributo *response* (ou *hessian*), que representa o quão proeminente um *keypoint* é, ou seja, *keypoints* com maior *response* são características mais robustas da imagem, e os atributos *octave* (ou oitavas, relacionado à escala em que o ponto foi detectado, pode estar relacionado ao atributo *size*) e *class_id* (relacionado com o valor laplaciano, sendo que pontos com os valores laplacianos diferentes não podem ser homólogos) [16]. Todos os atributos, com exceção do ponto, foram utilizados na tentativa de classificação.

Para cada *keypoint* é calculado um *descriptor* (ver algoritmo do protótipo em IV. D), que, de forma simplificada, é um vetor de 128 valores contendo a descrição do *keypoint* e sua vizinhança [8].

A extração dos *keypoints* e *descriptors* foi realizada utilizado o algoritmo SURF (BAY *et al*., 2008), e então os valores foram salvos em arquivos (.CSV), um arquivo para cada foto. Um segundo programa foi feito para auxiliar nos testes, escrito em Java, e executa os passos a seguir:

- Achar qual o número mínimo *n* de *keypoints* que foi encontrado de todo o conjunto de fotos (este será o número de *keypoints/descriptors* escolhidos para o treinamento da SVM). No caso deste trabalho, $n = 89$;
- Filtrar os *n* melhores *keypoints* (juntamente com seus *n* respectivos *descriptors*) das fotos que obtiveram um número de detecção de *keypoints* maior que *n*. Foram selecionados os *keypoints* com o maior

valor do atributo *response* como "melhores *keypoints*";

- Criar dois arquivos (.ARFF), padrão do Weka, para o treinamento da SVM. Um contendo todos *keypoints* de todas as notas, e outro contendo todos os *descriptors.* Para cada linha do arquivo (.ARFF) temos todos os *keypoints/descriptors* de uma única foto e, logo após esses valores, temos a classe a que ela pertence (*e.g.* "nota-dois-frente", "nota-cem-verso", etc.). Com isso o Weka é capaz de treinar a SVM e tentar reconhecer padrões nos dados.

Após a criação dos arquivos (.ARFF), foram testados os dois *datasets* na SVM para avaliar quais dados (*keypoints* ou *descriptors*) são melhores para classificar uma imagem. Descobriu-se então que o *dataset* de *descriptors* obtém melhores resultados, que serão descritos na próxima seção.

Para testar a eficiência do algoritmo na prática, ou seja, funcionando como um aplicativo na plataforma *Android*, a SVM foi implementada em um protótipo e treinada, resultando em um arquivo com extensão XML que contém a SVM "pronta" para classificar. Para gerar o arquivo XML foram utilizadas as funções da biblioteca OpenCV de *machine learning*, em especial a função "*train_auto()*" (documentação em *OPENCV's SVM*, 2013) que recebe como parâmetro uma matriz com os dados para o treinamento e os parâmetros do treinamento. O *kernel* utilizado foi polinomial (*CvSVM.POLY*), o método de rotação (*n-fold cross validation*) foi aplicado com o valor de 10 *folds* e o critério de parada do treinamento foi definido por precisão (*TermCriteria.EPS*) com épsilon valendo e^{-12}.

Por razões de limitação no *hardware* dos celulares, o conjunto de dados de treinamento foi dividido pela metade. O conjunto de imagens com posição horizontal em relação à câmera foi removido, pois não influenciava muito na identificação das diferentes fotos, restando assim todas as imagens em que a nota está na posição vertical em relação ao celular. Em consequência da redução, o arquivo XML ficou em metade de seu tamanho e pôde ser implementado no aplicativo de maneira otimizada.

Com o intuito de melhorar a detecção do aplicativo, que se mostrou ligeiramente sensível a variação das condições impostas aos dados usados no treinamento da SVM (em especial iluminação e distância da câmera), foram feitos alguns testes com filtros para diminuir a sensibilidade a essas variações. Os testes incluíram os seguintes filtros (com suas respectivas configurações):

- *Adaptive Median filter*: variação de ruído estimada = 1000, tamanho do *kernel* = 3;
- *Anisotropic Diffusion filter*: iterações = 5, *kappa* = 40, *lambda* = 0.2;
- *Bilateral filter*: *spatial radius = 3, range radius = 50;*
- *Mean Shift filter: spatial radius = 3, color distance = 25;*

Os testes com filtros acima foram feitos com o auxilio do *software open source, ImageJ* [29], e não obtiveram bons resultados. Todos os filtros testados neste trabalho foram aplicados após a conversão da imagem para escalas de cinza e antes da detecção de *keypoints.*

Outros testes foram feitos com os filtros *gaussian* e *median* (com diferentes tamanhos de *kernel* e número de iterações de aplicação do filtro) em um subgrupo de imagens. Os filtros foram implementados no próprio aplicativo usando a biblioteca *OpenCV*. Os seguintes tamanhos de *kernel* foram utilizados: 3x3, 5x5, 7x7 e 11x11.

O subgrupo de imagens utilizado nos testes dos filtros possui quatro cédulas: uma cédula de dois, cinco, dez e vinte. Para cada iluminação diferente, o subgrupo contém imagens com a distância entre a nota e a câmera de 10 e 15 cm. A iluminação para cada nota variou de "luz natural" (luz do dia), luz de uma lâmpada amarela de 60 Watts, lâmpada fria de 20 Watts e lâmpada fria de 20 Watts mais uma luminária de 15 Watts para auxiliar. O total de imagens é 4 (cédulas: dois, cinco, dez e vinte) x 2 (frente e verso da nota) x 2(distância de 10 e 15 cm) x 4 (tipo de iluminação diferente), resultando em 64 imagens diferentes. Os resultados dos testes estão descritos na próxima seção.

5. Resultados

A ideia inicial do algoritmo era processar a foto capturada pelo deficiente visual e obter a região dos números, para posteriormente, classificá-los comparando-os a um banco de imagens salvo. Essa forma de classificação por comparação ao banco de imagens foi descartada por causa do seu custo de memória excessivo e resultados não satisfatórios. Também foram testadas algumas técnicas de processamento de imagens na foto, porém elas foram substituídas pela técnica de *Feature Detection* (detecção de características, ou atributos) da imagem,

SURF, por ser mais rápido e ter melhores resultados.
Para a validação do método proposto neste artigo, executou-se a implementação do método considerando como entrada de dados todas as notas contidas no banco de imagens descrito na seção 5.C. O método foi aplicado em 30 imagens de cada valor monetário de cédula (dois, cinco, vinte, cinquenta e cem reais, sendo considerada tanto a frente quanto o verso de cada imagem). A Tabela III mostra os resultados da validação do método para o uso dos *keypoints*. Esses resultados são apresentados por classes de valores de cédulas. Essas classes foram categorizadas na primeira coluna da Tabela III e classificadas como segue: classe (**a**): *dois-frente*; classe (**b**): *dois-verso*; classe (**c**): *cinco-frente*; classe (**d**): *cinco-verso*; classe (**e**): *dez-frente*; classe (**f**): *dez-verso*; classe (**g**): *vinte-frente*; classe (**h**): *vinte-verso*; classe (**i**): *cinquenta-frente*; classe (**j**): *cinquenta-verso*; classe (**k**): *cem-frente*; classe (**l**): *cem-verso*.

Tabela 3. Métricas obtidas com o classificador SVM usando o conjunto de informações obtidas por meio dos keypoints.

Classe	Especificidade	Sensibilidade	Área sobre curva ROC
a	0.429	0.525	0.885
b	0.570	0.642	0.926
c	0.377	0.433	0.859
d	**0.311**	0.350	0.774
e	0.313	**0.342**	0.762
f	0.546	0.492	0.890
g	0.733	**0.733**	0.964
h	0.350	0.358	0.811
i	0.509	0.467	0.899
j	0.620	0.558	0.942
k	**0.845**	0.683	0.976
l	0.545	0.400	0.868
Média	0,512	0,498	0,879

A Tabela IV mostra a matriz de confusão dos resultados de classificação da SVM (simulado no *Weka*) sobre o conjunto de *keypoints* das fotos. Para a validação foi utilizada uma *n-fold-cross-validation*, onde (n = 10). As linhas com os valores de "**a**" a "**l**" representam as classes corretas as quais as notas pertencem, e as colunas de "Classificado como" possuem o número de classificações em que as notas de cada classe foram identificadas, portanto uma identificação 100% correta deveria ter valores maiores que zero apenas na diagonal principal da tabela, valores diferentes de zero fora da diagonal principal representam falso-positivos.

Tabela 4. Matriz de confusão: resultados da SVM sobre o dataset de keypoints

	Classificado como:											
	a	b	c	d	e	f	g	h	i	j	k	l
a	63	5	23	5	6	4	0	8	1	0	1	4
b	11	77	6	5	2	2	0	9	0	6	0	2
c	22	12	52	4	20	3	0	4	1	2	0	0
d	6	7	6	42	5	4	7	18	5	7	2	11
e	13	6	22	4	41	9	9	6	5	3	1	1
f	6	8	8	9	9	59	2	14	2	1	0	2
g	0	0	1	10	9	4	88	2	3	1	0	2
h	13	12	6	18	12	8	3	43	1	2	0	2
i	4	0	2	5	20	1	8	4	56	7	7	6
j	2	3	2	17	3	6	0	10	5	67	1	4
k	0	0	4	3	1	3	2	0	19	0	82	6
l	7	5	6	13	3	5	1	5	12	12	3	48

Com relação à sensibilidade, que mede a exatidão entre as instâncias positivas, a nota de vinte-frente proporcionou melhores resultados, onde obteve desempenho superior a 0,391 em relação a nota de dez-frente. Para avaliar a exatidão entre instâncias negativas, medida pela especificidade, observa-se um desempenho diferente para as notas avaliadas, sendo as notas de cinco-verso as de piores valores.
A Tabela V representa os resultados obtidos utilizando o conjunto de dados dos *descriptors* para classificação na SVM.

Tabela 5. Métricas obtidas com o classificador SVM usando o conjunto de informações obtidas por meio dos descriptors.

Classe	Especificidade	Sensibilidade	Área sobre curva *ROC*
a	**0.915**	0,983	0.996
b	0.983	0,975	0.998
c	0.974	**0,942**	0.994
d	0.992	0,975	0.999
e	0.975	0,967	0.991
f	**1.000**	0.992	0.999
g	**1.000**	0.975	0.998
h	0.992	0.983	0.995
i	0.944	0.983	1.000
j	**1.000**	**1,000**	0.999
k	0.992	0,992	1.000
l	**1.000**	0,992	0.997
Média	0,980	0,979	0,997

A Tabela VI contém a matriz de confusão obtida

utilizando o classificador SVM no *dataset* de *descriptors*. De um total de 1440 fotos, temos 1411 (97.9861%) identificações corretas e 29 (2.0139%) incorretas.

Tabela 6. Matriz de confusão: resultados da SVM sobre o dataset de descriptors

	Classificado como:											
	a	b	c	d	e	f	g	h	i	j	k	l
a	118	1	1	0	0	0	0	0	0	0	0	0
b	3	117	0	0	0	0	0	0	0	0	0	0
c	3	0	113	0	0	0	0	1	3	0	0	0
d	2	0	0	117	0	0	0	0	1	0	0	0
e	1	0	1	0	116	0	0	0	1	0	1	0
f	0	0	0	0	0	119	0	0	1	0	0	0
g	0	0	1	0	2	0	117	0	0	0	0	0
h	1	1	0	0	0	0	0	118	0	0	0	0
i	1	0	0	0	1	0	0	0	118	0	0	0
j	0	0	0	0	0	0	0	0	0	120	0	0
k	0	0	0	0	0	0	0	0	1	0	119	0
l	0	0	0	1	0	0	0	0	0	0	0	119

Com relação à sensibilidade, a nota de cinquenta-verso proporcionou melhores resultados, onde obteve desempenho superior às demais notas avaliadas. Para avaliar a especificidade, observa-se que há várias notas com valor ideal para o sistema na classificação. Nesse caso, os piores resultados foram obtidos com a nota dois-frente com especificidade de 0,915. Ressalta-se que esse tipo de características, *descriptors*, é superior aos *keypoints*.

De acordo com os valores de área sobre a curva *ROC*, nota-se que no uso das características obtidas pelos *descriptors* o sistema proposto foi mais eficiente para avaliação das notas monetárias, tendo uma taxa média de área sobre a curva *ROC* de 0,997, o que representa uma diferença de 0,118 superior na comparação com os *keypoints*.

A *Tabela VII* mostra os resultados do subgrupo de imagens descrito na seção anterior, ao se aplicar os filtros *gaussian* e *median* com diferentes *kernels*.

Tabela 7. Quantidade de acertos após aplicação de diferentes filtros de imagem

Filtro	***kernel***	**Acertos (TOTAL: 64)**
Gaussian	3x3	30
	5x5	30
	7x7	23
	11x11	10
	(3x3) aplicado duas vezes	33
	(3x3) aplicado três vezes	22
Median	3x3	**35**
	5x5	30
	7x7	18
	11x11	3
	(3x3) aplicado duas vezes	30
SEM FILTRO		34

Mesmo com os filtros não houve melhora no problema da sensibilidade do aplicativo em relação à iluminação e distância em que a imagem é capturada. A única melhora que se pôde notar foi ao aplicar o filtro *Median* com um *kernel* de tamanho 3x3, porém não foi uma melhora significativa.

6. Conclusão e trabalhos futuros

Diante da necessidade de desenvolvimento de tecnologias que auxiliam na vida dos deficientes visuais, juntamente com a importância e benefícios que elas trazem para essas pessoas, o objetivo deste estudo foi contribuir para a inclusão social e digital dos deficientes visuais brasileiros, propondo um método capaz de auxiliar os deficientes visuais no reconhecimento do valor de cédulas monetárias. Para implementação desse método, utilizou-se a linguagem *Java*, juntamente com a biblioteca *OpenCV*. Para a etapa de classificação, foi utilizada a plataforma *WEKA*, para os testes e para extração de características foi utilizado o algoritmo *SURF*, combinado a um classificador *SVM*. O método proposto foi aplicado sobre um ambiente de iluminação controlada e um único modelo de câmera. Em trabalhos futuros, esses parâmetros serão avaliados sob diferentes condições. Os testes realizados abrangeram uma quantidade razoável de cédulas (180 cédulas diferentes, com oito variações de foto para cada cédula) e obtiveram bons resultados de uma taxa média de área sobre a curva *ROC* de 0,997 para os testes controlados. Recomenda-se a utilização do método utilizado para a identificação de cédulas monetárias, porém ainda é necessário

diminuir a sensibilidade do método para utilizá-lo em um aplicativo para deficientes visuais. Este método pode ser utilizado nas próximas etapas do trabalho e também na implementação de um aplicativo *Android* com controles de voz para auxiliar os deficientes visuais na tarefa de identificação do valor da nota. A usabilidade do referido aplicativo também deve ser testada por usuários portadores de deficiência visual para aprimorá-la.

Referências

Android (2013). Disponível em: <http://www.android.com/>. Acesso em: 20 mar. 2013.

Android Emulator (2013). Disponível em: <http://developer.android.com/tools/help/emulator.html>. Acesso em: 21 mar. 2013.

Apple iPhone (2013). Disponível em: < http://www.apple.com/br/iphone/>. Acesso em: 20 mar. 2013.

BAY, H. *et al*. "SURF: Speeded Up Robust Features'. **Computer Vision and Image Understanding** (CVIU), vol. 110, nº 3, p. 346-359, 2008.

BRAGA, J. C. *et al*. "Estudo e Relato sobre a Utilização da Tecnologia pelos Deficientes Visuais". *In*: Simpósio Brasileiro de Fatores Humanos em Sistemas Computacionais 12, 2012, Cuiabá. Anais do Simpósio Brasileiro de Fatores Humanos em Sistemas Computacionais, 2012.

Camera Developer Guide (2013). Dispovinel em: <http://developer.android.com/guide/topics/media/camera.html>. Acesso em 26 mar. 2013.

CAMPBELL, L. Trabalho e cultura: meios de fortalecimento da cidadania e do desenvolvimento humano. Revista Contato – **Conversas sobre Deficiência Visual** – Edição Especial. Ano 5, número 7 – Dezembro de 2001.

Comma separated value file format (2013). Disponivel em: <http://www.creativyst.com/Doc/Articles/CSV/CSV01.htm>. Acesso em:

CORTES C.; VAPNIK, V. "Support-vector network", 1995. **Machine Learning**, vol. 20, p. 273-297.

CUNNINGHAM, P.; DELANEY, S. J. "k-Nearest Neighbor Classifiers", 2007. Technical Report UCD-CSI-2007-4, School of Computer Science and Informatics, University College Dublin, Ireland.

Documentação do método para conversão do espaço de cores, OpenCV (2013). Disponível em: <http://docs.opencv.org/modules/imgproc/doc/miscellaneous_transformations.html>. Acesso em: 26 mar. 2013

FLUSSER, J. "On the Independence of Rotation Moment Invariants", Pattern Recognition, vol. 33, p. 1405–1410, 2000.

GONZALES, Rafael C. e WOODS, Richard C. **Processamento digital de imagens**, 3. ed. São Paulo: Pearson Prentice Hall, 2010.

ImageJ (2013). Disponivel em: <http://rsbweb.nih.gov/ij/>. Acesso em: 10 jun. 2013.

JACOMINI, R. S. ; NASCIMENTO, M. Z. ; DANTAS, R. D. "Empregando padrões binários locais em domínio wavelet para classificação de lesões benigna e maligna em mamogramas". *In*: VIII Workshop de Visão Computacional, 2012, Goiania. VIII Workshop de Visão Computacional, 2012. p. 1-6.

Java (2013). Disponível em: <http://www.java.com>. Acesso em: 20 mar. 2013.

LookTel Money Reader (2013). Disponível em: <http://www.looktel.com/moneyreader>. Acesso em: 20 mar. 2013.

LORENA, A. C. ; CARVALHO, A. C. P. L. F. "Uma introdução às Support Vector Machines". **Revista de Informática Teórica e Aplicada**, v. 14, p. 43-67, 2007.

LOWE, D.G. "Distinctive Image Features from Scale-Invariant Keypoints", 2004. International Journal of Computer Vision, vol. 60, n. 2, p. 91-110.
MALLAT, S. Wavelets for a vision, Proceedings of the IEEE 84(4): 604–614, 1996.

OpenCV (2013). Disponível em: <http://opencv.

org/>. Acesso em: 20 mar. 2013.

OpenCV's DescriptorExtractor class (2013). Disponivel em: <http://opencv.willowgarage.com/documentation/cpp/features2d_common_interfaces_of_descriptor_extractors.html>. Acesso em: 26 mar. 2013.

OpenCV's FeatureDetector class (2013). Disponível em: <http://docs.opencv.org/doc/tutorials/features2d/feature_detection/feature_detection.html>. Acesso em: 26 mar. 2013.

OpenCV's Keypoint Structure (2013). Disponível em: <http://opencv.willowgarage.com/documentation/cpp/feature_detection.html>. Acesso em: 21 mar. 2013.

OpenCV's k-nearest-neighbor (2013). Disponível em: <http://docs.opencv.org/modules/ml/doc/k_nearest_neighbors.html>. Acesso em: 27 mar. 2013.

OpenCV's SVM (2013). Disponível em: <http://docs.opencv.org/modules/ml/doc/support_vector_machines.html>. Acesso em: 10 jun. 2013.

OSTA H. ; QAHWAJI R. ; IPSONS S. "Comparisons of feature selection methods using discrete wavelet transforms and support vector machines for mammogram images", 2008. Systems,Signals and Devices. IEEE SSD 2008. 5th International Multi-Conference. p. 1 –6.

Primeira Família do Real (2013). Disponivel em: <http://novasnotas.bcb.gov.br/primeira-familia.html>. Acesso em 26/03/2013

SILLER, F. ; BRAGA, J. C. "GPS para deficientes visuais. In: Simpósio de Fatores Humanos em Sistemas Computacionais", 2010. Belo Horizonte. Anais Estendidos do IX Simpósio de Fatores Humanos em Sistemas Computacionais. Porto Alegre: Sociedade Brasileira de Computação, 2010. v. 2. p. 1-162.

SILVA, F. A. *et al*. "Uma Metodologia Para Detectar E Reconhecer Placas De Sinalização De Trânsito", 2012. Goiania. Anais Do Viii Workshop De Visão Computacional, 2012.

Sony Xperia U (2013). Disponivel em: <http://www.sonymobile.com/br/products/phones/xperia-u/>. Acesso em: 27 mar. 2013.

VALE, G. M.; POZ, A. P. "Processo de detecção de bordas de Canny**"**. Boletim de Ciências Geodésicas, Vol. 8, nº 2 (2002).

VIOLA, P.; JONES M. "Robust real-time object detection" (2002). **International Journal of Computer Vision**, 2002.

Waikato Environment for Knowledge Analysis (2013). Disponível em: <http://www.cs.waikato.ac.nz/ml/index.html>. Acesso em: 20 mar. 2013.