RECONHECIMENTO DE PLACAS UTILIZANDO MARCADORES NATURAIS

Gabriel Fontenelle Senno Silva[1]
Dr. Marcelo Hashimoto[2]

[1]Estudante do Curso de Bacharelado em Ciência da Computação; Bolsista do CNPq;
colecionador.gabriel@gmail.com

[2]Professor do Centro Universitário Senac
marcelo.hashimoto@sp.senac.br

Linha de Pesquisa: Ambientes Interativos
Projeto: Xxxxxxxxxxxxxxxx

Resumo

Palavras-chave:

Abstract

Keywords:

1. Introdução

No mundo atual em que grande quantidade de dados é fornecida visualmente para as pessoas, ser portador de deficiência visual implica em restrições ao acesso desses dados. No Brasil existem 35,8 milhões de portadores de alguma deficiência visual, entre estes 506,3 mil são cegos [1]. Embora existam soluções para mitigar dificuldades que pessoas com deficiência visual possam encontrar, como a leitura de textos com uso de aplicativos de OCR em smartphones [2], ou utilização de sistemas de identificação de placas de sinais de trânsito [3], ou reconhecimento de texto em placas de veículos com ALPR [4], essas soluções se restringem a auxiliar em seus problemas específicos,

mas não podem ser utilizadas para reconhecimento de placas de instituições e leitura de seu conteúdo em um ambiente com diversos cartazes.

Para este artigo foi realizado estudo da literatura de técnicas consideradas atualmente clássicas, utilizadas como base para sistemas de reconhecimentos atuais. Entre as diversas técnicas que podem ser utilizadas para detectar e extrair textos de placas, muitos aplicativos existentes fazem uso de marcadores não-naturais como *QR Code* como forma de identificar placas ou informativos [2] e outros utilizam OCR diretamente para a leitura de textos em documentos [5] que requer um bom posicionamento para leitura adequada. Uma técnica em particular para identificação de imagens que obteve sucesso recentemente foi a de pontos-chave [6] capaz de detectar imagens arbitrárias mesmo com algumas distorções.

O *QR Code* (*Quick Response code*) é um marcador, lançado em 1994, desenvolvido para armazenar mais informação que códigos de barras unidirecionais tradicionais e a princípio utilizado na indústria automobilística. O *QR Code* [7] foi desenvolvido para extração de muita informação com identificação independente da orientação planar do marcador por um algoritmo robusto e rápido, sendo esta sua principal vantagem, porém como o foco de seu desenvolvimento foi a criação de uma estrutura visual que diferencie dos formatos utilizados em logotipos para uma detecção rápida, o *QR Code* nem sempre combina com a identidade visual de placas por isso o consideramos um marcador não-natural, sendo essa uma desvantagem ao seu uso. Embora patenteada pela Denso Wave, o uso de *QR Code* é permitido livremente e sua especificação disponível ao público.

O *OCR* (*Optical Character Recognition*) é utilizado para reconhecimento de textos em imagens e sua conversão para caracteres, com sua a primeira patente na década de 1930 nos EUA. Na década de 1970 surgiram os primeiros algoritmos para reconhecimento de caracteres com múltiplos tipos de fontes de caracteres. Diversos algoritmos de código-fonte aberto de OCR estão disponíveis, e embora o uso de OCR não apresente problemas de conflitos com a identidade visual de placas, pois

a princípio detecta qualquer texto em imagens, há desvantagens que tornam difíceis o uso exclusivo de OCR para reconhecimento de textos apenas no interior de placas; como a detecção de qualquer texto que estiver na sua área de reconhecimento e variação de orientação da imagem resulta em identificação incorreta de caracteres.

A técnica de pontos-chave pode ser utilizada para reconhecimento de imagens arbitrárias [8], mas requer que os algoritmos de detecção sejam treinados com todas as placas que se deseja identificar, esse treinamento é uma desvantagem por requerer que toda variação de placa seja registrada, além de desnecessariamente computacionalmente custo, uma vez que placas de instituições possuem partes em comuns e o que as diferenciam são seus textos.

Marcadores não-naturais, OCR[12] e placas arbitrárias respectivamente [16].

Cada uma dessas tecnologias possui uso restrito e limitações, neste trabalho buscamos integrar técnicas a contornar as limitações na detecção de placas e reconhecimento de seu conteúdo textual.

2. Objeto da pesquisa

Nosso objetivo é o desenvolvimento de um sistema para reconhecimento de imagem e leitura de texto, mais especificamente para reconhecimento de placas de instituições e leitura de seu

conteúdo. Para isso, fazemos uso das técnicas de detecção e leitura de texto com um *pipeline* especifico visando contornar as desvantagens que cada técnica possa apresentar.

Nosso *pipeline* consiste na junção das seguintes técnicas para reconhecimento do texto: Detecção de imagens arbitrárias com uso de pontos-chave; detecção de imagens com uso de marcadores; e reconhecimento de texto com uso de OCR. Após o reconhecimento é utilizado sintetizador de voz para falar o texto a portadores de imparidade visual.

Usando a detecção de imagens por pontos-chave para detectar logotipos da instituição, que são marcadores naturais por incorporar-se bem a identidade visual das placas, restringimos o treinamento do algoritmo apenas as variações dos logotipos, e não a todo formato de placa. Com os pontos-chave identificados conseguimos corrigir distorções na imagem, que impedem o reconhecimento preciso de caracteres por OCR, e isolar o texto para que bibliotecas de OCR consigam identificar apenas o texto presente dentro das placas da instituição mesmo se esta estiver cercada por panfletos ou outras placas diversas.

3. **Metodologia**

Para extração de texto, apenas de placas da instituição, e sua leitura utilizamos o modelo de análise de pontos-chave [9], extensamente referenciado na literatura técnica atual, que possibilita localizar imagens em cenas, inclusive imagens parcialmente obstruídas com variações de posicionamento e de orientação planar.

O método de análise não utiliza todos os pontos da imagem, somente são utilizados pontos de interesses, conhecidos como pontos-chave, que são pontos que possuem maior probabilidade de estarem presentes em diversos tipos de variações na imagem, como variações na escala ou na orientação planar.

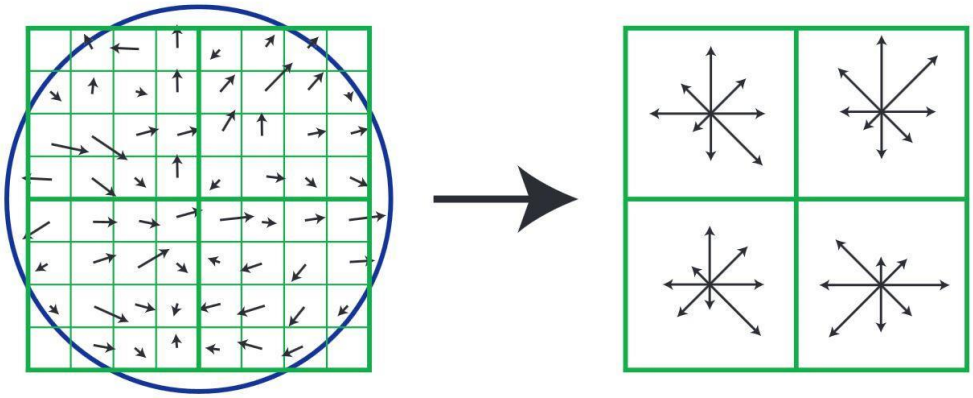
Utilizamos o modelo de análise de pontos-chave em nosso *pipeline*, que pode ser separado em 5 etapas distintas:

- **Detecção de Pontos-chave**. Cada técnica para localização de pontos-chave utiliza parâmetros próprios para definir os pontos de interesse, mas em geral os pontos-chave são obtidos procurando por pontos em áreas com grande contraste entre pontos próximos, com eliminação de pontos que não alcancem os parâmetros usados para filtragem dos pontos. Um bom detector de pontos-chave procura por pontos robustos que independe de determinadas variações, algumas das variações que ocorrem em imagens são: variação da escala; variação na orientação planar; variação angular; variação na iluminação; e variação no tom de cores.

O método de detecção de pontos-chave utilizado é o método SURF [10], desenvolvido a fim de suceder o método clássico SIFT [6]. Ambos os métodos SIFT e SURF, para seleção de pontos-chave, analisam os pixels da imagem, com seus pixels vizinhos a procura de pontos que se mantêm presentes em diversas escalas, removendo assim pontos considerados fracos por estarem em bordas e mantendo pontos fortes presentes em cantos.

Pontos-chave detectados pelo método SIFT a esquerda e pelo método SURF a direita.

- **Descrição de Pontos-chave**. Utilizando os pontos-chave selecionados pelo detector são criados descritores que representam características desses pontos-chave. Os descritores são vetores numéricos, [0, 5, 10, 2, 3, 4 ...], que representam características invariantes que se deseja extrair dos pontos de interesse, como por exemplo o vetor gradiente entre pontos próximos do ponto de interesse. Os métodos SIFT e SURF utilizam um grupo de vetores gradientes, filtrados por uma janela gaussiana, no ponto-chave para criação do descritor que represente o ponto de interesse.



Vetores gradientes no ponto de interesse na imagem à esquerda e representação do descritor gerado pelo SIFT a direita [6].

- **Correspondência de pontos-chave.** Para detectarmos o logotipo em uma imagem, com os descritores gerados pelo nosso método escolhido, SURF, fazemos uma verificação de correspondência entre os descritores presentes na imagem do logotipo e da imagem da cena.

Imagem do logotipo à esquerda e da cena à direita. Traços verdes representam as correspondências entre os descritores.

Como podem ocorrer múltiplas correspondências diferentes entre mesmos descritores, as correspondências resultantes são filtradas usando-se a distância euclidiana entre elas, mantendo assim apenas combinações próximas; por sugestão de Lowe[6] são mantidas combinações que não ultrapassem 70% da distância euclidiana.

$$\sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}.$$

Fórmula da distância euclidiana para o plano entre os pontos P = (Px, Py) e Q = (Qx, Qy). [11]

Correspondências não filtradas à esquerda e correspondências filtradas à direita.

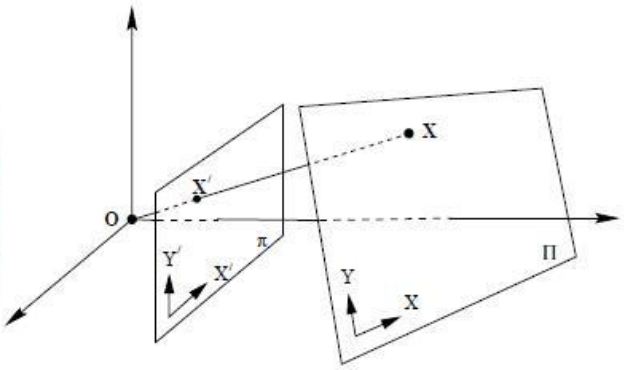
- **Estimativa de pose.** Quando a imagem da cena apresenta uma perspectiva ou uma orientação planar diferente da imagem do logotipo dizemos que a imagem está distorcida. Para corrigir a distorção da imagem e estimar a sua pose correta é necessário conhecer sua homografia.

Exemplo de distorção de imagem.
Imagem na perspectiva correta à esquerda e imagem com distorção à direita.

A Homografia é a relação existente entre duas imagens, com variação no ângulo, rotação e escala, de um mesmo cenário. Essa relação pode ser obtida analisando uma amostra de dados a procura

de um modelo matemático que melhor represente a amostra de correspondências entre imagens removendo assim *outliers.* O modelo matemático encontrado é utilizado para gerar uma matriz de transformação, conhecida como matriz de transformação homográfica, que pode, por exemplo, ser utilizada para posicionar uma imagem em uma foto, enquadrando-se na perspectiva da foto. A matriz gerada pela homografia, diferente da transformação afim, não preserva a distância entre linhas paralelas, podendo assim trabalhar com transformações de perspectiva.

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

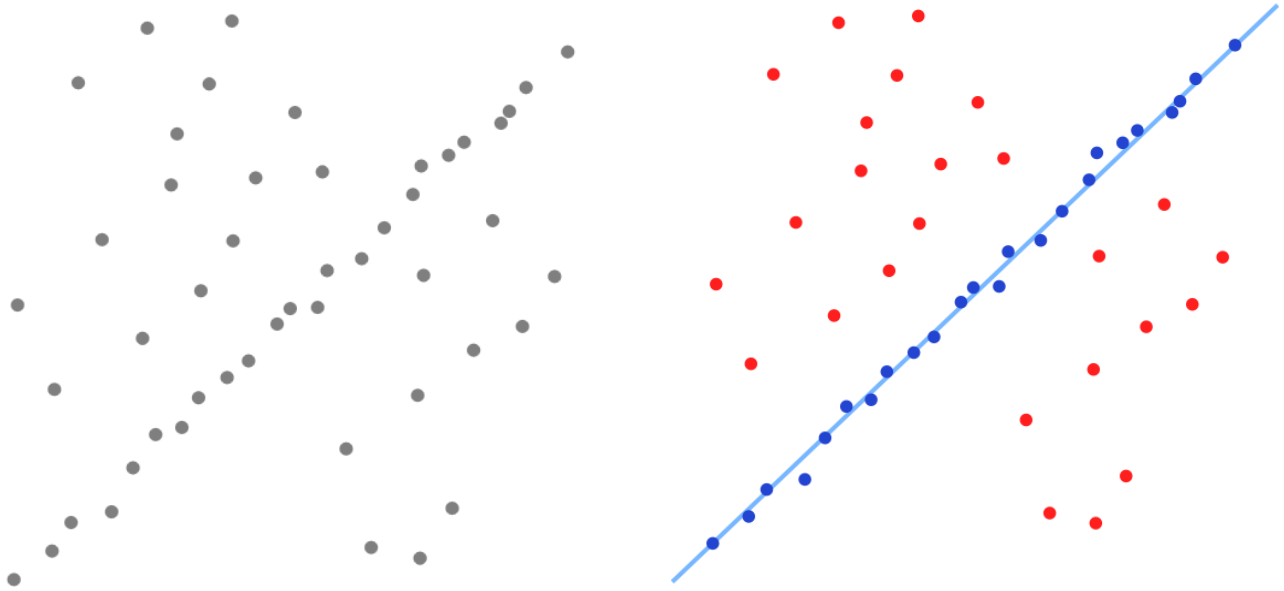


Transformação vetorial de perspectiva com matriz homográfica à esquerda [15] e representação de projeção da perspectiva à direita [15].

Para fazermos a correção da distorção por homografia entre a imagem do logotipo e imagem da cena, usamos a matriz homográfica e multiplicação de pontos da imagem na cena pela matriz a fim de obter uma nova imagem. A matriz homográfica é gerada após a extração da homografia entre as duas imagens por um método matemático não-determinístico de interpolação conhecido como RANSAC [13].

Imagem na perspectiva correta à esquerda e imagem com perspectiva corrigida à direita.

RANSAC (*Random sample consensus*) é um método utilizado para identificar modelos matemáticos a partir de pontos dispersos em uma imagem. O método funciona em iterações, a cada iteração é verificado um modelo matemático e pontos inliers ao modelo são adicionados em um vetor, esse vetor é comparado com o vetor da iteração anterior e o vetor com a menor quantidade de pontos *inliers* é descartado. Ao final das iterações se foram encontrados pontos *inliers* o melhor modelo matemático encontrado é retornado.



Exemplo de uso clássico do RANSAC para identificação de linha em um conjunto disperso de pontos. Amostra de pontos na imagem à esquerda e estimativa do modelo de linha à direita. [14]

Pseudo algoritmo do RANSAC:

```
Iterações = 0
Melhor_combinação = nenhuma
melhor_erro = algo muito grande
enquanto iterações < k {
    talvez_inliers = n valores selecionados aleatoriamente
    talvez_modelo = modelo dos parâmetros que se ajustam ao talvez_inliers
    também_inliers = conjunto vazio
    para cada ponto em dados não presentes em talvez_inliers {
```

```
      se ponto adequa-se ao talvez_modelo com um erro menor que limiar
          adiciona ponto para também_inliers
   }
   se o número de elementos em também_inliers é > número de valores mínimos para considerar o modelo adequado {
      melhor_modelo = parâmetros de modelo que se adequam a todos os pontos em talvez_inliers e também_inliers
      erro_atual = uma medida de quão bem o modelo se adequa a esses pontos
      se erro_atual < melhor_erro {
         melhor_combinação = melhor_modelo
         melhor_erro = erro_atual
      }
   }
          incrementa iterações
}
retornar melhor_combinação
```

- **OCR e Sintetização de Voz.** Com a imagem resultante da cena transformada para a perspectiva do logotipo, que se encontra alinhado corretamente na horizontal, podemos utilizar biblioteca de OCR para conversão dos caracteres do texto e assim ler o conteúdo textual ao usuário do sistema por meio de Tecnologias Assistidas como bibliotecas sintetizadoras de voz.

4. **Resultados e discussão**

Originalmente desenvolveríamos um sistema para Desktop, que poderia funcionar tanto em Windows como Linux, utilizando bibliotecas consideradas estado-da-arte, como prova de conceito do *pipeline* e após estudo de viabilidade migraríamos para uso mobile. Se tivéssemos tempo hábil após a implementação mobile, implantaríamos um sistema para adição de dados a placas utilizando Realidade Aumentada.

Para desenvolvimento Desktop escolhemos a linguagem de programação Python 2.7, biblioteca de Visual Computacional OpenCV, em sua versão estável 2.4.10, biblioteca de OCR Tesseract 3.02.02 e a biblioteca Python pyttsx, que permite utilizar as bibliotecas de sintetização nativas do Windows (SAPI5) e Linux (eSpeak).

Para desenvolvimento Mobile, optamos pelo desenvolvimento nativo em Android escolhendo utilizar a linguagem C++, a biblioteca OpenCV e a biblioteca Tesseract OCR ambas compiladas para Android com uso do NDK da Google.
Como prova de conceito para Desktop desenvolvemos um sistema capaz de reconhecer placas e ler seu conteúdo dependente da resolução da câmera disponível.

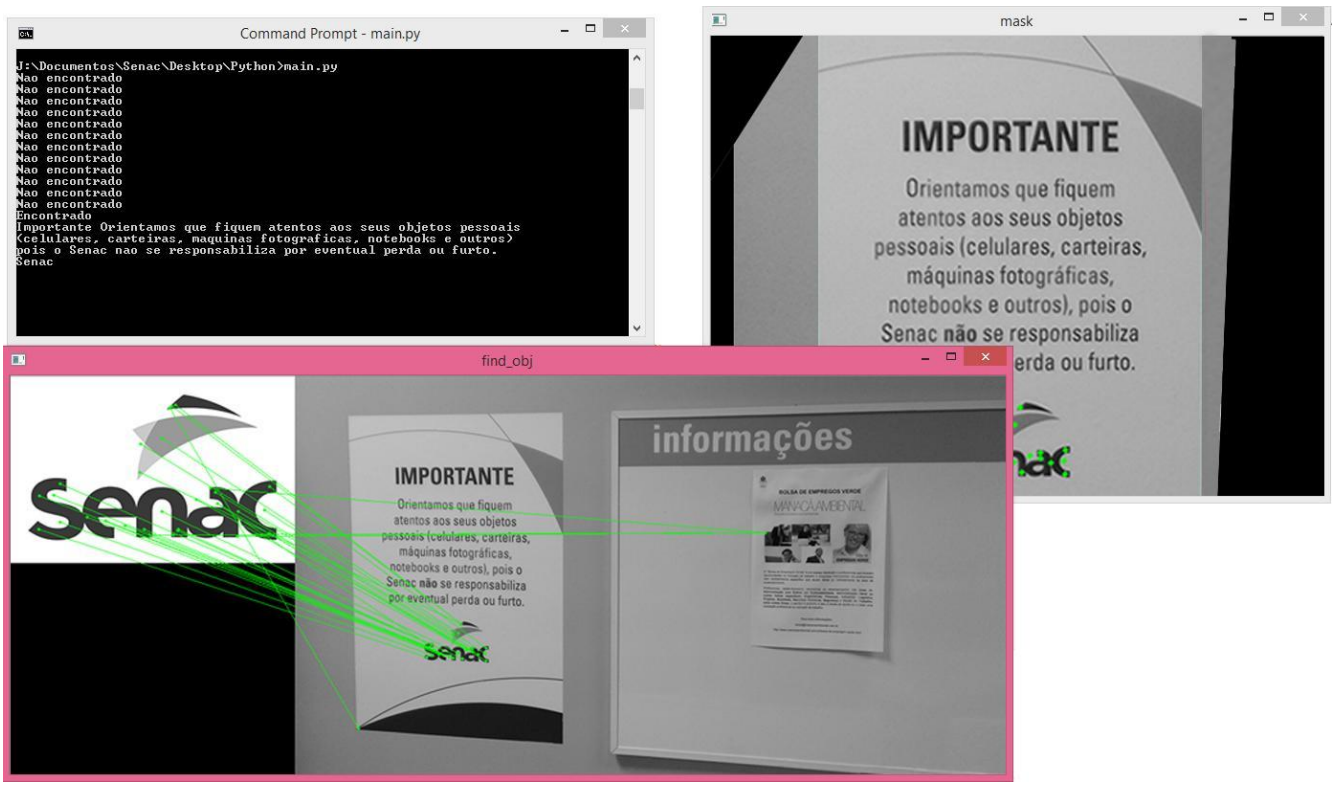


Imagem do sistema em funcionamento no Desktop. Imagem da cena à direita na parte superior com a perspectiva corrigida, imagem à esquerda na parte superior com texto extraído e reconhecido na linha de comando do terminal. Imagem inferior ilustra a correspondência de pontos-chave.

Embora o desenvolvimento da prova de conceito para Desktop tenha sido concluído com sucesso, algumas ressalvas devem ser mencionadas: durante a execução do sistema em Windows 8 no momento de utilizar a biblioteca Tesseract OCR, através da biblioteca Python python-tesseract-0.9, o interpretador do Python travou por falha de segmentação. Uma solução não foi encontrada para esse problema, em tempo hábil, nos forçando a utilizar a biblioteca Tesseract OCR por linha de comando usando a biblioteca Python commands, o que diminui a velocidade de execução do nosso

sistema no Windows. Essa de falha de segmentação não se apresenta no Linux, porém a biblioteca Python python-tesseract para Linux é fornecida para apenas a distribuição Linux Ubuntu versão 12, nos forçando a instalação de uma distribuição anterior a atual 14 para sua execução.

Não recomendamos a instalação do Linux Ubuntu 12 em máquina virtual VM Ware se a câmera utilizada for integrada como ocorre em Laptops, pois a máquina virtual pode não passar informação de drive corretamente para o Linux Ubuntu 12 impedindo a execução de captura de imagem necessária na identificação de placas.

Para portar o sistema para Android, o código desenvolvido para Python, na prova de conceito Desktop, foi alterado e adaptado para a linguagem C++, sendo feito uso da biblioteca Glue como auxílio ao desenvolvimento nativo do aplicativo Android.

Embora foi desenvolvido código em C++ para ser compilado com o NDK, ocorreram dificuldades técnicas com as bibliotecas para portar o sistema para Android e com as ferramentas de desenvolvimento.

A biblioteca OpenCV é fornecida como um aplicativo externo no Google Play, porém como os métodos para detecção e geração dos descritores SIFT e SURF são patenteados, estes não se encontram nesse aplicativo externo sendo necessário extraí-los da pasta nonfree no código-fonte do OpenCV e compilá-los com o NDK. A biblioteca Tesseract OCR não fornece um aplicativo externo e, portanto, deve ser compilada utilizando NDK.

A dificuldade técnica encontrada com as ferramentas de desenvolvimento foi a falha na execução do emulador de dispositivo móvel do Android Studio e falha na abertura do projeto com Glue devido ao "Gradle não estar pronto para execução" durante a abertura do projeto, forçando-nos a utilizar a ferramenta Eclipse com o *plugin* ADT, que deixou de receber suporte da Google [17].

Outra dificuldade técnica encontrada: compilar ambos a pasta nonfree do OpenCV em conjunto com a biblioteca Tesseract OCR. Durante testes de viabilidade, as bibliotecas foram compiladas individualmente e testadas com sucesso, embora não apresentassem uma velocidade de execução satisfatória no dispositivo testado Sony Ericsson Arc S. Porém a compilação para uso simultâneo de ambas as bibliotecas, apesar de não apresentar erro de compilação, apresentou erro de execução quando a biblioteca Tesseract não pode ser reconhecida mesmo sendo compilada.
Esses problemas inviabilizaram o progresso do desenvolvimento para mobile no tempo exigido para esse projeto.

5. **Conclusões**

Foi realizada prova de conceito para Desktop que se demonstrou funcional, com algumas ressalvas mencionadas anteriormente, só que houveram muitos problemas inesperados com as bibliotecas utilizadas tanto para portar para mobile como na implementação Desktop.

Portanto estão nos planos futuros a resolução desses problemas com essas bibliotecas ou a procura de opções alternativas para o método SURF presente na biblioteca OpenCV e para a biblioteca de OCR Tesseract-OCR, que talvez não sejam muito boas em desempenho, mas que funcionem melhor para uma primeira versão do sistema.

6. **Referências**

[1] IBGE Censo 2010. (2012) "Censo 2010: escolaridade e rendimento aumentam e cai mortalidade infantil",

http://censo2010.ibge.gov.br/pt/noticias?view=noticia&id=1&idnoticia=2125&busca=1&t=censo-2010-escolaridade-rendimento-aumentam-cai-mortalidade-infantil.

[2] All Access LLC. (2014) “About AllAccess.US”. http://allaccessmenus.weebly.com/about.html, https://itunes.apple.com/us/app/all-access-talking-menus-more/id633106012?mt=8&ign-mpt=uo%3D8.

[3] Bahlmann, C.; Ying Zhu; Ramesh, V.; Pellkofer, M.; Koehler, T. (2005) "A system for traffic sign detection, tracking, and recognition using color, shape, and motion information," Intelligent Vehicles Symposium, 2005. Proceedings. IEEE, vol., no., pp.255, 260, 6-8 June 2005.

[4] Shan Du; Ibrahim, M.; Shehata, M.; Badawy, W. (2013) "Automatic License Plate Recognition (ALPR): A State-of-the-Art Review," Circuits and Systems for Video Technology, IEEE Transactions on , vol.23, no.2, pp.311,325, Feb. 2013.

[5] Piksoft Inc. “Perfect OCR: document scanner with high quality OCR.” https://itunes.apple.com/br/app/perfect-ocr-document-scanner/id363095388?mt=8

[6] David G. Lowe. (2004) “Distinctive Image Features from Scale-Invariant Keypoints.”

[7] QR Code History. http://www.qrcode.com/en/history/

[8] Google Search Image. http://images.google.com/

[9] Tinne Tuytelaars, Krystian Mikolaczyk. (2007). Local Invariant Feature Detectors: A Survey.

[10] Bay, Herbert; Tuytelaars, Tinne; Van Gool, Luc. (2006) "SURF: Speeded Up Robust Features". Computer Vision – ECCV 2006, Lecture Notes in Computer Science, Springer Berlin Heidelberg pp.404-417.

[11] Distância Euclidiana
http://pt.wikipedia.org/wiki/Dist%C3%A2ncia_euclidiana

[12] Gautam Gupta (2012) "Making a Simple OCR Android App using Tesseract"
http://gaut.am/category/open-source/

[13] Martin A. Fischler; Robert C. Bolles. (1981) "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." Volume 24 Issue 6, June 1981, Pages 381-395, ACM New York, NY, USA

[14] Wikipedia. RANSAC.
http://en.wikipedia.org/wiki/RANSAC

[15] "Homography in Image Retrieval"
https://taotaoorange.wordpress.com/tag/cv/

[16] Placas Institucionais
http://avanceacessibilidade.com.br/placas-sinalizacao.html

[17] Google. "ADT Plugin Release Notes".
http://developer.android.com/tools/sdk/eclipse-adt.html

M Ondrej, V Zboril Frantisek, D Martin. (2007) "Algorithmic and mathematical principles of automatic number plate recognition systems", BRNO University of technology

de la Escalera, A.; Moreno, L.E.; Salichs, M.A.; Armingol, J.M. (1997) "Road traffic sign detection and classification," Industrial Electronics, IEEE Transactions on, vol.44, no.6, pp.848, 859, Dec 1997

David G. Lowe. (1999). Object recognition from local scale-invariant features.

Jun Miura; Tsuyoshi Kanda, Yoshiaki Shirai. (2000). "An Active Vision System for Real-Time Traffic Sign Recognition." (pp 52-57). Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE.

Roberto Manduchi, Sri Kurniawan, Homayoun Bagherinia. (2010) "Blind Guidance Using Mobile Computer Vision: A Usability Study."

Øivind Due Trier, Anil K. Jain, Torfinn Taxt. (1996). "Feature extraction methods for character recognition-A survey."

R. Smith. (2007) "An Overview of the Tesseract OCR Engine," 2013 12th International Conference on Document Analysis and Recognition, pp. 629-633, Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2, 2007

Smith, Ray and Antonova, Daria and Lee, Dar-Shyang. (2009). "Adapting the Tesseract Open Source OCR Engine for Multilingual OCR", Proceedings of the International Workshop on Multilingual OCR, Barcelona, Spain.

Chirag Patel, Atul Patel PhD, Dharmendra Patel. (2012) "Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study", International Journal of Computer Applications (0975–8887), Volume 55–No.10, October 2012

Ravina Mithe, Supriya Indalkar, Nilam Divekar. (2013) "Optical Character Recognition", International Journal of Recent Technology and Engineering (IJRTE), ISSN: 2277-3878, Volume-2, Issue-1, March 2013