Uma proposta de arquitetura móvel baseada em visão computacional para pessoas com deficiência visual

Alexsander V. Canez[1], Gisele M. Simas[1], Henrique de S. Conceição[1], Igor P. Maurell[1], Jônatas D. Fraga[1], Mariana P. Fernandes[1], Marlon R. C. Franco[1], Matheus C. Corrêa[1], Regina Barwaldt[1], Ricardo N. Rodrigues[1]

[1]Centro de Ciências Computacionais (C3) – Universidade Federal do Rio Grande (FURG)
Caixa Postal 474 – 96.201-900 – Rio Grande – RS – Brasil

alex.avc@hotmail.com, giselesimas@furg.br, henriqueconceicao@furg.br , igorpardo97@gmail.com, jonatas_fraga@hotmail.com, marianapereira_f@hotmail.com, marlonrcfranco@gmail.com, matheuscorrea@furg.br, reginabarwaldt@furg.br, ricardonagel@furg.br

Abstract. *This article discusses aspects about possibilities of using techniques and recent advances in the area of Computer Vision for the development of resources within the paradigm of Ubiquitous Computing, that provides the improvement in the quality of life of people with visual needs, especially in the accomplishment of daily tasks. In addition, this article proposes a Computational System for mobile that presents the tasks: i) scene classification; ii) object recognition; iii) collision detection.*

Resumo. *Este artigo discute um Sistema Computacional, os aspectos sobre possibilidades da utilização de técnicas e recentes avanços na área de Visão Computacional para o desenvolvimento de recursos a partir do paradigma de Computação Ubíqua, que proporcionem a melhoria na qualidade de vida de pessoas com necessidades visuais, especialmente na realização de tarefas do cotidiano. Além disso, este artigo propõe um Sistema Computacional para mobile que realiza as tarefas: i) classificação de cenas; ii) reconhecimento de objetos; iii) detecção de colisão. São tratados aspectos relativos à arquitetura do sistema.*

1. Introdução

Tendo em vista que cerca de 285 milhões de pessoas ao redor do mundo possuem algum tipo de deficiência visual [WHO, 2014] e são frequentemente excluídas em função de suas limitações, o desenvolvimento de Tecnologia Assistiva se torna necessário para proporcionar a inclusão das mesmas. Dessa forma, a motivação do presente projeto é a inclusão social de pessoas com deficiência visual através de soluções assistivas.

Proceedings of the XII SIBGRAPI (October 1999)

Mais especificamente, é proposta uma arquitetura *Publish/Subscribe* (*pub/sub*) para *smartphones*, que utiliza o paradigma de computação ubíqua para incorporar aplicações de técnicas recentes de visão computacional na vida dessas pessoas. A arquitetura proposta permite que as imagens capturadas pelo dispositivo sejam processadas tanto localmente por um algoritmo de detecção de colisões, quanto remotamente por algoritmos de detecção de objetos. O retorno de tais algoritmos é então informado ao deficiente visual através de síntese de voz.

O trabalho está organizado da seguinte forma: a seção 2 cita e descreve alguns trabalhos relacionados; a seção 3 apresenta a arquitetura geral e parcial proposta; na seção 4 é detalhada a implementação do sistema; a seção 5 mostra o resultado de alguns experimentos e análise qualitativa; e, finalmente, na seção 6 é apresentada a conclusão.

2. Trabalhos Relacionados

Um dos projetos da área, proposto por Ereno [Ereno, 2015], consiste em um sistema de auxílio à navegação utilizando um *Kinect* da *Microsoft* para converter vídeos em informações sonoras. A proposta é o usuário carregar o *laptop* dentro de uma mochila. Os dados do ambiente são capturados pela câmera (*Kinect*), processados pelo computador e transmitidos por fones de ouvido com tecnologia *bone conduction*. O grande problema deste projeto é mobilidade, pois é considerado um método inadequado e perigoso. O diferencial do aplicativo em andamento para o protótipo de Ereno é que, no lugar de equipamento pesados será utilizado apenas um *smartphone*.

Outro trabalho que possui uma proposta similar a do presente projeto é o *BlindTool* [Joseph, 2013], que fornece um sensor de visão para os deficientes visuais fazendo uso de uma rede neural artificial, a qual descreve os objetos assim que são identificados. A maior diferença entre o *BlindTool* e o projeto em andamento é o modo de envio das mensagens. O aplicativo de Joseph envia inúmeras imagens para a rede neural a fim de reconhecer tudo o que está em volta do usuário, enquanto a câmera do celular encontra-se em repouso.

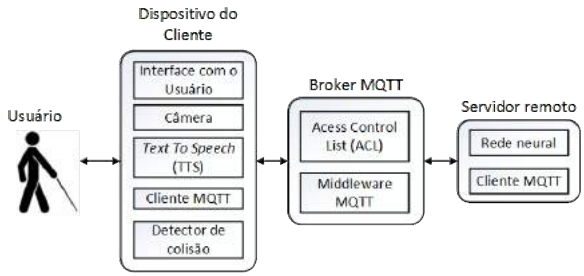
Be My Eyes [Erfurt, 2012] é um aplicativo que possui uma proposta semelhante mas aplicada de forma diferente, o usuário que necessita de ajuda solicita a assistência por meio do aplicativo. Esta requisição é enviada para um voluntário, o qual recebe o pedido, e ao aceitar, é estabelecida uma conexão de vídeo. Já o aplicativo proposto abstrai o voluntário a um sistema de classificação de objetos e detecção de colisões.

3. Arquitetura

O sistema é dividido em três partes, como apresentado na Figura 1. O Cliente é responsável pela interface com o usuário, sendo essa de fácil manuseio possibilitando que o deficiente visual inicie a captura contínua das imagens, as processando no algoritmo de colisão ou envie um *frame* específico para uma rede neural, que irá descrevê-lo. O protocolo *Message Queuing Telemetry*

Transport (MQTT) é responsável pela comunicação entre o Cliente e Servidor remoto, gerenciando a transmissão de dados em um servidor de forma segura e com privacidade.

Figura 1 - Arquitetura Geral



Fonte: Autores.

3.1. Cliente

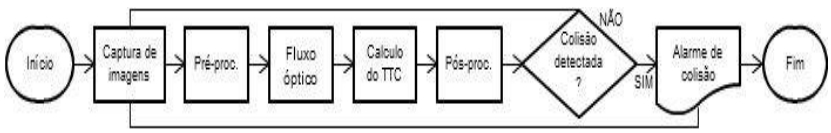
O dispositivo do cliente foi implementado em um aplicativo para *smartphone*, e a sua interface foi estruturada graficamente de maneira a simplificar o uso da mesma pelo deficiente visual, além disso é feito o uso da síntese de voz já disponível nos aparelhos. Essa parte da arquitetura capta o ambiente através da câmera e envia esse dado pela rede até o Servidor remoto e, também, aplica um algoritmo na imagem para prever futuras colisões.

Ao ser feita a captura, a imagem é enviada para o Servidor remoto utilizando o protocolo MQTT, o qual a recebe e retorna com uma *string* especificando o objeto e a cena que foram registrados. Já no sistema de colisão, a câmera inicia a captura contínua, onde um *software* que detecta obstáculos é responsável por enviar um alerta para o usuário. Com o auxílio da saída de áudio é utilizado o *plugin Text-to-Speech (TTS)*, que tem como objetivo ler essas *strings* recebidas e discursar para o usuário.

3.1.1. Sistema de Colisão

O algoritmo desenvolvido considerou a detecção de obstáculos dinâmicos e estáticos em tempo real, no qual uma colisão pode ser detectada a partir do tempo de colisão calculado através do fluxo óptico gerado pelo obstáculo. Um fluxograma básico do algoritmo é demonstrado na Figura 2, no qual suas principais etapas são brevemente comentadas a seguir:

Figura 2 - Fluxograma básico do algoritmo desenvolvido.



Fonte: Autores.

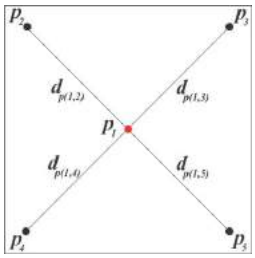
Etapa 1. Captura de imagens: são adquiridas a partir da câmera monocular não calibrada de um *smartphone*. O método não requer nenhuma calibração.

Etapa 2. Pré-processamento: as imagens são convertidas em escala de cinza e redimensionadas de 1280x720 para 384x216 *pixels*, proporcionando melhorias na eficiência computacional sem uma redução significativa no desempenho. Além disso, um filtro Gaussiano é aplicado para reduzir possíveis interferências causadas por ruídos aleatórios que possam prejudicar no fluxo óptico.

Etapa 3. Fluxo óptico: é aplicado o algoritmo de Farnebäck [Farnebäck, 2002] para encontrar no *frame* atual a correspondência entre os *pixels* do *frame* anterior. Este algoritmo calcula o fluxo óptico denso e retorna um vetor de fluxos [*flowx*, *flowy*] nas direções *x* e *y* de cada *pixel* da imagem.

Etapa 4. Cálculo do Tempo de Colisão (*Time-To-Collision - TTC*): neste passo, o fluxo óptico é utilizado para estimar o *TTC* para cada *pixel* da imagem, no qual a partir do modelo básico de uma câmera *pinhole* foi derivada uma equação fechada para calcular o *TTC*. Então para estimar este tempo, foi estabelecido um conjunto de pontos *P = {p2, p3, p4, p5}* ao redor de um ponto central, conforme a Figura 3.

Figura 3 - Ilustração da distância Euclidiana entre p1 e os outros.



Fonte: Autores.

Para todos os pontos $p \in P$, obtém-se a distância euclidiana entre *p1* e os demais para dois *frames* consecutivos. Então é definida a relação entre estas duas distâncias calculadas para cada $p \in P$. A partir dessa relação, é calculado o TTC_j do ponto *p1* no instante (*t*) para cada um dos *pixels* do *frame*, onde *j* é o *pixel* em análise.

Etapa 5. Pós-processamento: é aplicado um método baseado em uma heurística para remover possíveis erros no cálculo do TTC_j decorrentes da estimação do fluxo óptico. Para reduzir estes possíveis erros, operações morfológicas de erosão e dilatação removem pequenas regiões e preenchem buracos. As regiões resultantes são segmentadas e regiões com menos de 10% do total de *pixels* são removidas. Então é criado um mapa com as regiões remanescentes, no qual cada uma representa um possível obstáculo.

Etapa 6. Colisão detectada: para cada região resultante do passo anterior, o *TTC* é estimado pela média de todos os TTC_j dos *pixels* pertencentes a região, conforme a equação 1. Onde *R* é a região resultante e *n* é o número de *pixels* dessa região.

$$TTC \approx \frac{\sum_{i \in R} TTC_i}{n}$$ (Eq. 1)

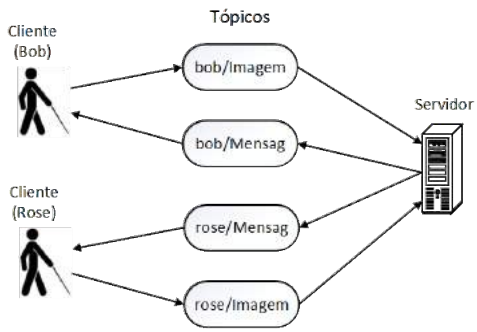
Pelos experimentos definiu-se um *threshold* de 2,5s e objetos detectados com *TTC* abaixo desse valor representam um risco de colisão.

Etapa 7. Alarme de colisão: quando e *TTC* está abaixo do *threshold,* sinais sonoros, variando na mesma proporção do risco de colisão, alertam o usuário.

3.2. Protocolo MQTT

Visando uma comunicação segura entre Cliente e Servidor remoto, a integridade da mensagem, assim como a simplicidade da comunicação, foi escolhido o protocolo MQTT, que trabalha no padrão de troca de dados *pub/sub*, onde a imagem capturada pelo dispositivo é enviada ao Servidor remoto por intermédio de um *broker*. “O princípio do modelo de comunicação *publish/subscribe* é que os componentes que estão interessados em consumir certas informações podem registrar seu interesse. Este processo de registro de um interesse é chamado de inscrição, o interessado é então chamado de um assinante” [Hunkeler, 2008]. A Figura 4 mostra um exemplo de conexão através de inscrições e publicações em tópicos no *broker*.

Figura 4 - Tópicos Pub/Sub



Fonte: Autores.

3.3. Servidor

Após receber a imagem, o servidor a processa utilizando modelos de redes neurais convolucionais (CNN, do inglês *convolutional neural networks*), que são mais rápidas e possuem uma taxa de erro menor em comparação com redes não-convolucionais [Krizhevsky et al., 2012]. Estes modelos são implementados em um Servidor remoto, com o intuito de reduzir o processamento no celular e também levando em consideração a facilidade na manutenção e na adição de novos modelos. Atualmente, como mostra a Tabela 1, são utilizados os seguintes modelos:

Tabela 1 - Modelos de Redes Neurais

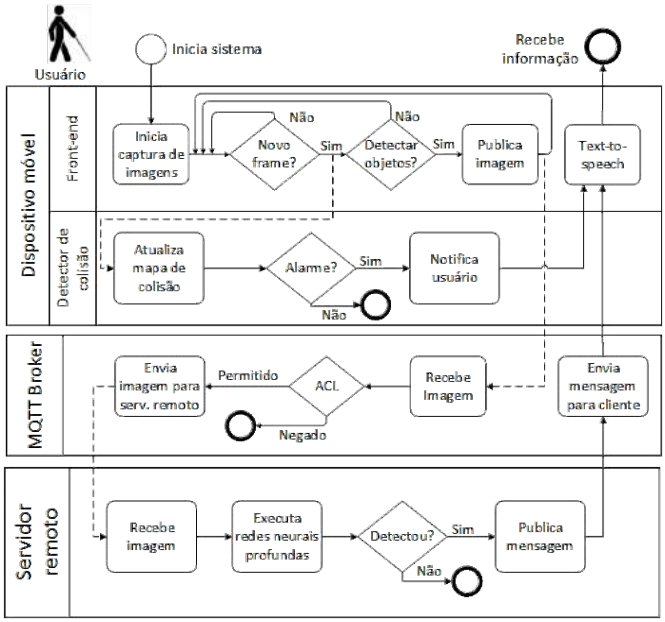
Nome	Objetivo	Classes	Idioma
GoogLeNet	Identificar Objetos	1000	Português
Places205-GoogLeNet	Identificar Cenários	205	Português
Inception-v2	Identificar Objetos	21.841	Inglês

Fonte: Autores.

3.4. Visão Geral

A Figura 5 retrata o fluxo de informações utilizando o *Business Process Model Notation*, em um exemplo onde o usuário (Bob) captura continuamente *frames* da câmera, que serão utilizados para o algoritmo de detecção de colisão e também, um *frame* específico, para comunicação com MQTT. O *broker*, que através do *Access Control List* (*ACL*), verifica se o usuário pode publicar no tópico; a imagem é então enviada para a rede neural, o qual recebe a imagem, a processa, gera uma mensagem de texto e publica no tópico de mensagem do usuário.

Figura 5 - Representação gráfica do fluxo de informações



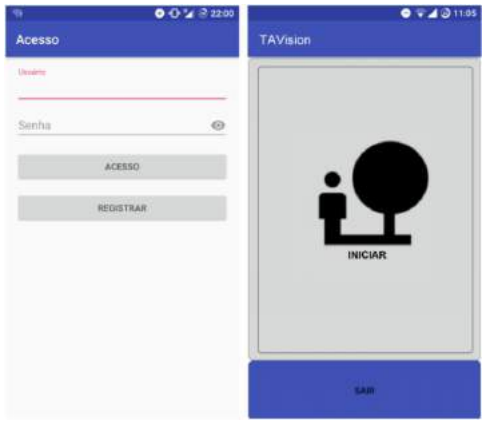
Fonte: Autores.

4. Implementação

Como ideia inicial, tem-se em vista usuários de dispositivos *Android*. Dessa forma, o cliente foi completamente implementado no *Android Studio*: uma IDE específica para desenvolver aplicações para este sistema operacional. A Figura 6 ilustra a interface do cliente: a primeira imagem é a tela de *login* e a segunda

mostra a interface principal.

Figura 6 - Interface do cliente



Fonte: Autores.

O algoritmo para detecção de colisões foi implementado em linguagem C++ usando a biblioteca de visão computacional *OpenCv*. Já para a implementação do sistema de comunicação *pub/sub* utilizou-se o *Eclipse Mosquitto*™ [Mosquitto], um *broker open source* que implementa o protocolo MQTT. Essa escolha ocorreu por conta da facilidade de configuração e implementação do *Mosquitto*™, visto que a linguagem utilizada é C++, a qual também é utilizada na rede neural.

Um recurso importante implementado no *broker* é o *ACL*, que segundo [Uzunov, 2016], limita o acesso aos tópicos a uma lista de usuários com permissão para publicarem ou se inscreverem nos mesmos. Este recurso é utilizado para permitir que somente o servidor e o usuário que criou o tópico tenham acesso a essa informação.

O servidor foi implementado na plataforma *Qt*, e possui acesso a uma rede neural de imagens. Ele é configurado de tal forma que se comporta como um super-cliente, isto é, um cliente que pode se inscrever em qualquer tópico criado pelos outros no *broker*. Deste modo, sempre que um cliente cria um tópico e publica uma imagem, o servidor pode receber esta imagem, processá-la em sua rede neural e devolver uma descrição em forma de texto em outro tópico, onde só o cliente que publicou a imagem pode se inscrever.

5. Experimentos

Esta seção apresenta experimentos realizados com o sistema implementado, descrito nas seções anteriores. Os mesmos foram realizados com sete usuários com visão saudável, instruídos a vendarem seus olhos durante o processo. O objetivo destes experimentos é obter um *feedback* qualitativo do sistema para que no futuro sejam implementadas algumas melhorias. Posteriormente serão realizados experimentos com deficientes visuais em situações do mundo real.

5.1. Classificador de Cenas e Identificador de Objetos

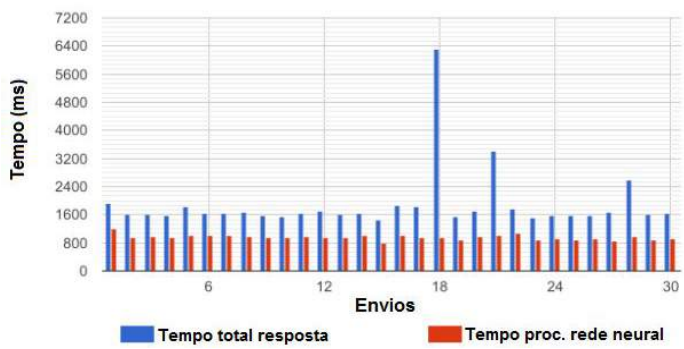
Experimento 1: Usuários foram convidados a testar aleatoriamente o classificador de cenas e o identificador de objetos em suas tarefas diárias.

Análise qualitativa: O classificador de cenas obteve avaliações positivas, sendo capaz de dar uma classificação geral na maioria do tempo. Entretanto, o identificador de objetos apresentou um maior número de discrepâncias, onde apenas objetos totalmente enquadrados foram reconhecidos, o que não é realizado facilmente sem visão. Atualmente estão sendo investigados métodos que possam aprimorar a detecção de objetos, como apresentado por [Ren et al., 2015], onde não apenas será possível reconhecer o objeto mas também determinar a sua localização mesmo em cenas imprecisas.

Experimento 2: O segundo experimento realizado consiste num teste de desempenho baseado na análise do tempo decorrido, entre o momento em que a imagem é capturada pelo dispositivo e o momento em que ocorre a execução da descrição da imagem. Para tanto foi utilizado um *smartphone* Samsung Galaxy J3, com processador Quad-Core 1.5 GHZ, memória RAM de 1.5 GB e Android 5.1.1, conectado ao *broker* através de rede *Wi-fi*; realizando 30 envios sucessivos de uma determinada imagem. Para cada envio foi registrado individualmente o tempo de processamento na rede neural (servidor) e o tempo total para o recebimento da descrição da imagem.

Análise qualitativa: Os dados de tempo de cada um dos envios foram inseridos em uma planilha e então utilizados para gerar um gráfico, apresentado na Figura 7. Embora fosse observada a ocorrência de picos isolados de tempo total de resposta, que podem ter sido causados por instabilidades na rede, o tempo de processamento permaneceu quase constante em todo o experimento. A análise do gráfico permite constatar que, para envios de um único usuário, a rede neural está funcionando adequadamente e respondendo em um tempo estável. Futuros testes com mais de um usuário enviando imagens em paralelo estão sendo desenvolvidos para que se obtenham maiores informações quanto ao desempenho do sistema em situações do mundo real.

Figura 7 - Gráfico do tempo de resposta



Fonte: Autores.

5.2. Detector de Colisões

Os vídeos foram adquiridos com a câmera de um *smartphone* com resolução de 1280x720 *pixels* e uma taxa de 30 *frames* por segundo. Foram realizados três experimentos, que são descritos a seguir:

Experimento 1: Foram gravados 20 vídeos em um ambiente controlado, onde um obstáculo móvel se desloca em direção a câmera com velocidade constante de 0,3m/s.

Experimento 2: Foram gravados 80 vídeos, sendo 5 para cada um dos 8 diferentes objetos estáticos colocados em 2 cenários predispostos a variação de luminosidade e outros ruídos, onde a câmera se desloca em direção aos objetos com velocidade média de 0,4m/s.

Experimento 3: Foram gravados 20 vídeos em 4 ambientes diferentes (2 externos e 2 internos) livres de obstáculos durante o deslocamento do usuário, sendo 5 para cada ambiente; no entanto, neste experimento haviam obstáculos laterais que não poderiam representar risco de colisão; um *smartphone* foi colocado no peito do usuário, onde ele se deslocou com velocidade desconhecida por 18s em cada cenário.

A performance do algoritmo para detecção de colisões foi dada em termos de precisão, revocação e acurácia. Já para validar o *TTC*, foi analisado o erro médio absoluto entre o *TTC* e o *ground truth* (nos experimentos 1 e 2). O *ground truth* foi estabelecido a partir da taxa de 30fps e a distância conhecida entre a câmera e o objeto.

Os resultados podem ser considerados satisfatórios, pois, de acordo com a Tabela 2 as taxas de precisão, revocação, acurácia e erro médio foram de: 93,27%, 89,06%, 92,91% e 0,29s, respectivamente (experimentos 1 e 2). Do experimento 3 obteve-se apenas a acurácia (93,22%), possibilitando avaliar a robustez do método em ambientes livres de obstáculos e predispostos a ruídos provenientes do ambiente. Entretanto, o método não foi capaz de detectar obstáculos com tempos abaixo de 1s.

Tabela 2 - Resultados dos experimentos, apresenta os valores médios.

Experimento	Precisão	Revocação	Acurácia	Erro médio
1	90,32%	93,33%	91,67%	0,18 seg
2	96,21%	84,79%	94,14%	0,40 seg
1 e 2	93,27%	89,06%	92,91%	0,29 seg
3	-	-	93,22	-

Fonte: Autores.

6. Conclusão

Este artigo apresenta um projeto em que um aplicativo possui um sistema de colisão que permite ao usuário prever possíveis obstáculos em sua trajetória

através da câmera do *smartphone*. Esse sistema é local, não necessitando de acesso à rede, o que o mantém seguro e ágil, dentro do possível, com relação às respostas. Além disso, em paralelo, o mesmo se comunica com um servidor baseado na arquitetura *pub/sub* com o intuito de capturar uma imagem e retornar uma verbalização do ambiente/objeto da mesma.

Para uma melhor integração e confiabilidade, futuramente, será implementado um sistema de amigos, onde além da rede neural, o usuário poderá receber descrições do ambiente de contatos conectados a ele, garantindo respostas mais precisas e com maior detalhamento.

Referências

ERENO, D. **Reconhecimento de ambiente. Revista Pesquisa** – FAPESP, n. 232, p. 70-71, 2015.

ERFURT, C., WIBERG, H. J., JENSEN, A. H. (2012). **BlindTool**. Disponível em: <http://bemyeyes.com/>. Acesso: 08 mar. 2017.

FARNEBÄCK, G. (2002). "**Polynomial expansion for orientation and motion estimation**", Ph.D. dissertation, Linkoping University, Computer Vision, The Institute of Technology.

HUNKELER, U., TRUONG, H. L., STANFORD-CLARK, A. (2008). "**MQTT-S - A publish/subscribe protocol for Wireless Sensor Networks**". In: Communication systems software and middleware and workshops, 2008. Comsware 2008. 3rd international conference on. IEEE. p. 791-798.

JOSEPH, P. C. (2014). **BlindTool**. Disponível em: <https://play.google.com/store/apps/details?id=the.blindtool> Acesso: 11 mar. 2017

MOSQUITTO. "**An Open Source MQTT v3.1/v3.1.1 Broker**". Disponível em: <https://mosquitto.org/>. Acesso: 11 mar. 2017.

REN, S., HE, K., GIRSHICK, R., and SUN, J. (2015). "**Faster r-cnn: Towards real-time object detection with region proposal networks**". In: Advances in Neural Information Processing Systems, 2015, p. 91–99.

UZUNOV, A.V. (2016). "**A survey of security solutions for distributed publish/subscribe systems**". Computers & Security, v. 61, p. 94-129, ago. 2016. Disponível em: <http://dx.doi.org/10.1016/j.cose.2016.04.008>. Acesso: 11 mar. 2017.

WHO (2014). "**Visual impairment and blindness**". fact sheet, n. 282, 2014. Disponível em: <http://www.who.int/mediacentre/factsheets/fs282/en/>. Acesso: 10 mar. 2017.

KRIZHEVSKY, A., SUTSKEVER, I., HINTON, G. E. (2012). "**ImageNet Classification with Deep Convolutional Neural Networks**". Nips Proceedings. Disponível em: <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks>. Acesso: 27 fev. 2017.