# SISTEMA DE NAVEGAÇÃO PARA DEFICIENTES VISUAIS BASEADO EM VISÃO ROBÓTICA

Kiyoko Marina Tukozaki – mtukozaki@gmail.com

Prof. Doutor Paulo Batista Lopes (Orientador) – e-mail

## RESUMO

A visão computacional é uma ferramenta inovadora, que vem se desenvolvendo ultimamente. A capacidade de emular um olho humano é uma utilidade poderosa para o mundo e que pode auxiliar os Deficientes Visuais em diversas tarefas cotidiana, por exemplo, a locomoção. Nas ruas, em estabelecimentos e em casa, existem muitos obstáculos que dificultam a mobilidade e/ou causam acidentes aos Deficientes Visuais. Pensando nesta situação, este projeto apresenta o desenvolvimento de um aplicativo de celular Android para auxiliar os deficientes visuais. Este recurso usufrui da tecnologia de Visão Computacional integrada à ferramenta API Google Cloud Vision para o reconhecimento visual. Esta aplicação está vinculada a plataforma Google Cloud e Firebase que foram manipulados juntos no Android Studio através da linguagem Java. Cerca de 74% dos testes mostraram êxito em reconhecer objetos, classes e grupos em uma imagem com câmera em tempo real.

Palavras chaves: Visão computacional. Deficientes Visuais. Detecção de objetos ou classes. Aplicativo.

# NAVIGATION SYSTEM FOR VISUAL DISABILITIES BASED ON ROBOTIC VISION


## ABSTRACT

The computer vision is an innovative tool that has been developing lately. The ability to emulate a human eye is a powerful utility to the world and can assist the visually impaired in various everyday tasks, for example, locomotion. On the streets, in establishments and at home, there are many obstacles that hinder mobility and/or cause accidents to the Visually Impaired. Thinking about this situation, this project presents the development of an Android mobile app to help the visually impaired. This feature takes advantage of Computer Vision technology integrated with the Google Cloud Vision API tool for visual recognition. This application is linked to the Google Cloud and Firebase platform that were manipulated together in Android Studio through the Java language. About 74% of the tests were successful in recognizing objects, classes, and groups in a real-time camera image.



Keywords: Computer vision. Visually impaired. Object or class detection. APP.


## 1 INTRODUÇÃO

Os dados da Organização Mundial da Saúde (OMS) apontam que cerca de 3,5% da população brasileira possui deficiência visual. Dentro do grupo de deficientes visuais, incluem-se pessoas que nasceram com a dificuldade visual ou aqueles que adquiriram tal perda ao longo da vida (FUNDAÇÃO DORINA, 2010). Este grupo enfrenta dificuldade na acessibilidade e mobilidade urbana devido à escassez de investimento nesta área de serviço (SOUSA, 2015). Com apenas o uso de bengalas e cães guias para a locomoção, não é possível obter 100% de segurança, visto que as bengalas possuem limitações durante a precisão ao detectar obstáculos (TAKIZAWA, 2012), enquanto os cães guias não são acessíveis a todas as classes sociais, pois custam em torno de R$60.000,00 (OLIVEIRA JR, 2016).

Estudos apontam que a maior parte das vítimas que sofrem acidentes com lesões não intencionais está inserida dentro do grupo de deficientes visuais (MANDUCHI 2011). Apesar do uso de bengala e/ou cão guia, o usuário corre risco de quedas e colisões durante a mobilidade, tornando evidente a dependência dos deficientes visuais. Segundo estudos apontados pelos pesquisadores Roberto Manduchi e Sri Kurniawan da Universidade de Santa Cruz, ao entrevistarem os deficientes visuais, chegaram à conclusão que 20% dos entrevistados tiveram acidentes casuais pelo menos uma vez por mês, mesmo com o auxílio de cães guias e/ou bengalas (MANDUCHI 2011).

Diante deste impasse, a aplicação de processamento de imagem integrado a um algoritmo de aprendizagem de máquina (Machine Learning) para o reconhecimento de obstáculos está sendo visto como um recurso inovador tecnológico melhorado e com preço acessível a todas as classes sociais (PINHO; SOUZA; OLIVEIRA; COSTA; FILHO; SOUSA; ARAÙJO, 2011). Pode-se obter dispositivos que auxiliem os deficientes visuais através de variadas aplicações e áreas de desenvolvimento com menor custo. Uma das áreas interessante dentro do contexto é a integração de visão computacional com o celular Android. O sistema tem sido usado atualmente para reconhecimento de face, detecção veicular, processamento e geração de vídeos e entre outros. O uso de aplicativos para plataforma Android (APP) com visão computacional já está inserido dentro da sociedade, otimizando diversas áreas, por exemplo, na área de negócios (AVOZDAINDUSTRIA 2019).

Devido à aplicação de o dispositivo ser no celular, o protótipo possui vantagens de custo e acessibilidade. O aplicativo permitirá acesso de qualquer smartphone Android, tornando a acessibilidade em todas as classes sociais com custo zero para o usuário, pois o smartphone é a tecnologia mais usada no país, sendo que 92% dos brasileiros são possuidores desta tecnologia (MEDEIROS 2018).

Este projeto possui o objetivo de auxiliar e facilitar a mobilidade dos deficientes visuais, utilizando Machine Learning com reconhecimento de imagem integrado à um aplicativo “APP” Android, com resposta em áudio. O desenvolvimento do aplicativo é através do Android Studio juntamente com os recursos do Firebase, programado em Java para explorar o uso do processamento de imagem, a fim de emular o comportamento do olho humano, através de entradas e saídas de imagens por meio computacional. Neste processo, a ferramenta juntamente com algoritmos de processamento Machine Learning extraídos do Kit Firebase ML, simula uma visão para o usuário, identificando e alertado dos obstáculos a frente.

## 2 REVISÃO DA LITERATURA

A área da visão computacional consiste em um estudo com aplicações em diversas áreas, por exemplo, diversão e negócio. Entretanto, é pouco explorada para questões sociais. (MEKHALFI; MELGANI; BAZI; ALAJLAN 2015). Dentro deste campo de estudos, a detecção de imagens pode ser manipulada e analisada em diversas formas exploradas dentro do conceito de visão computacional, dependendo da necessidade de cada pesquisa. Pode-se classificar um conjunto de imagens através de detecção de características e propriedades em comum, como rotação, dimensionamento, bordas e entre outras na imagem capturada. Em ênfase a esta pesquisa, é explorada estas diversas características em conjunto (CHINCHA 2011).

Muitos trabalhos utilizaram o conceito de visão computacional para auxiliar os deficientes visuais. Neste contexto, temos como exemplo a pesquisa de Ricardo Chincha e YingLi Tian da faculdade de Department of Electrical Engineering em New York. Tais pesquisadores desenvolveram um sistema que veste o usuário com câmera que detecta e reconhece os objetos para os deficientes visuais, utilizando os algorítmos Scale Invariant Feature Transform (SIFT) e Speeded-Up Robust Feature (SURF), que simplesmente extraem informações da imagem, pelo método de rotação e dimensionamento da imagem (CHINCHA 2011).

Outra forma de identificar imagens envolve a técnica de descrição grosseira, através da medida com o sensor de compressão. Mohamed Lamine Mekhalfi, Farid Melgani, Yakoub Bazi e Naif Alajlan utilizaram este método para desenvolver um dispositivo integrado a uma câmera para detecção de objetos internos para auxiliar os deficientes visuais. Com isso, eles obtiveram resultados positivos na capacidade de detectar imagens simultaneamente e houve desvantagem no design do conjunto de imagens de treinamento (MEKHALFI 2014).

Em paralelo à ideia de auxiliar os deficientes visuais, pode-se também citar a publicação de Hanen Jabnoun, Faouzi Benzarti, e Hamid Amiri que tiveram o interesse de realizar uma avaliação sobre algoritmos para reconhecimento e localização de objetos rápidos acompanhado de uma visão computacional robusta trabalhada em cena de vídeo. Eles utilizaram o algoritmo SIFT e o SURF (JABNOUN; BENZARTI; AMIRI 2014)

Conforme estudos já apontados, verifica-se que a detecção da imagem necessita de um algoritmo de processamento, seja na visão clássica ou em Machine Learning, para manipular a imagem conforme o objetivo de cada ideia. Existem diversos algoritmos para quaisquer aplicações, por exemplo, captura de imagens para manipulação robótica, reconhecimento de espécies de plantas, detecção de placas sinalizadoras de trânsitos e entre outros (NUNES 2014).

Neste trabalho foca-se em apenas trabalhar em Machine Learning devido ao projeto basear-se em métodos mais complexos para reconhecimento do objeto, por exemplo, para definir o padrão de um buraco em uma imagem é preciso uma máquina de aprendizado devido a imagens possuir formas não definidas, não sendo ideal utilizar apenas o reconhecimento de bordas, pois o objeto a ser localizado possui inúmeras características que são chamadas de pontos de interesses. Machine Learning é um sistema computacional programado para aprender atividades simples do ser humano e atividades complexas para computadores, sendo um algoritmo complexo. São programas que possuem lógicas que emulam o sistema de aprendizado humano para específicos objetivos (MITCHEL 1997). Assim, o Machine Learning é uma ferramenta que pode ser explorada dentro da plataforma Firebase, através do Kit Firebase ML, que possui APIs de reconhecimento de imagem do Google Cloud.

Neste mesmo âmbito de pesquisa, o Machine Learning pode trabalhar com reconhecimento e detecção de movimentos, Yu publicou em seu trabalho um algoritmo para reconhecimento visual de saliências utilizando como base o algoritmo denominado Frequency-Tuned (FT) que se baseia na diferença de valores médio e calculado das cores de cada pixel, diferentes efeitos são obtidos do valor final calculado. Baseada nesses cálculos a autora realiza simulações usando o software OpenCV com a ferramenta Haar cascade deste software para detecção e comparação de imagens de carros em movimento, evidenciando o funcionamento do algoritmo proposto. (YU 2018)

No campo de reconhecimento de objetos é possível que mais de um objeto seja reconhecido, tomando-se para isso diversos pontos de referência numa mesma base. Devi defende em seu trabalho que é possível reconhecer um indivíduo em uma área delimitada usando a técnica line of interest (LOI) para se contar linhas numa área delimitada. O autor estuda ainda a detecção facial em imagens reais utilizando algoritmo computacional por ele proposto, evidenciando que, dentro do contexto de reconhecimento de objetos, é possível demarcar um ou mais objetos em uma mesma figura dentro dos cálculos realizados pelo OpenCV (DEVI; PATERIYA 2017).

Um outro modelo para reconhecimento e classificação de objetos é chamado Bag-of-Words (BoW), o modelo consiste em fragmentar a imagem em um partes conhecidas e utiliza-las de base para classificação de partes na imagem não conhecidas. Esse método foi estudado por Ali (2016), que usou como base para seus estudos imagens provenientes do Google, para realizar a classificação de cada fragmento da imagem (ALI 2016).

Enquanto, a pesquisa de Kelly Aparecida Oliveira Sousa, da Universidade Presbiteriana Mackenzie, mostrou a integração do OpenCV com o sistema Android para reconhecimento de faixa de pedestres para auxiliar deficientes visuais. Neste estudo, com o auxílio do OpenCV é realizado o reconhecimento da imagem através da análise de contorno das imagens. Neste protótipo foram apontados resultados com detecção de 86% das imagens e com tempo de processamento rápido, que varia entre 76 a 325 milissegundos (SOUSA, 2013).

A aplicação do Firebase com Android Studio também é eficiente, utilizando os mesmo recursos que este projeto, foi estudado por Jean Carlos Franco da Universidade Regional de Blumenau que desenvolveu um aplicativo para deficientes visuais para reconhecimento de objetos e, possui integrado um sintetizado de voz que facilita o uso do portador do celular. Aplicou-se 30 imagens como teste obtendo 70% de assertividade das amostras e a distância foi apontada como a principal interferência para reconhecimento de imagens. Este estudo de Jean Carlos Franco foi a base deste projeto (FRANCO 2018).

Com base em tais estudos, o trabalho propõe a mesma ideia de reconhecimento de objetos através das imagens obtidas pela câmera. Verificando que os resultados de assertividade e reconhecimento de objetos é alta, sendo maiores que 70%, dependendo da utilidade de cada

dispositivo desenvolvido. Enfim, a saída da informação da imagem processada é pelo sintetizador de voz, que já está vinculado no Android pelo Java, em que será responsável em comunicar ao deficiente visual sobre os obstáculos a frente (REVISTA PROGRAMAR 2019).

# 3 METODOLOGIA

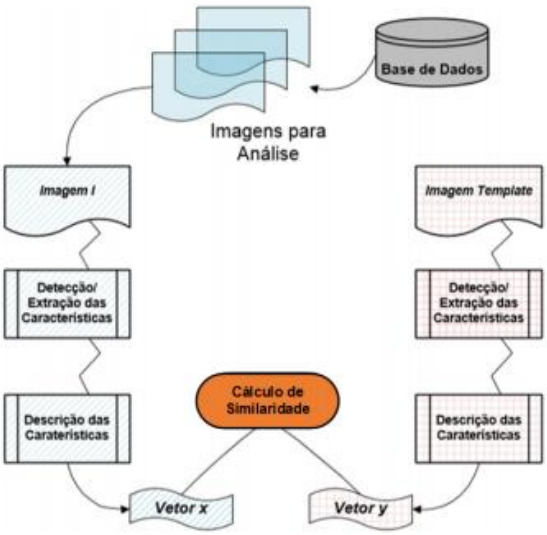
## 3.1 RECONHECIMENTO DE OBJETOS

A detecção de um objeto depende de um máquina de aprendizado que trabalha com uma determinada classe/categoria estabelecida pelo programador. Para que ocorra o reconhecimento da imagem, o Machine Learning classifica a imagem através da extração de características proeminentes que foram vistos anteriormente no objeto, seguindo uma linha de padrão, como intensidade de pixels, características das bordas, histogramas e entre outros. Ao extrair tais características da imagem é realizada uma comparação com os padrões dos caracteres alfanuméricos (NASCIMENTO 2007). Por exemplo, o método Haar-Like utilizou a diferença da soma das intensidades de pixels para caracterizar sub-regiões de uma imagem com o objetivo de detectar faces humanas (ANTONELLO).

Em processamento de imagem, o reconhecimento de objeto através de Machine Learning é considerado um Processamento de Nível Alto, que explora a emulação da inteligência artificial por meio de aprendizado da máquina com imagens. Como visto no Capítulo 2, o sistema geral de cores de cada pixel é representado por um determinado ponto (x, y) que corresponde a um vetor e seu valor escalar define a intensidade de cada cor. Cada pixel possui um conjunto de vizinhos, tendo vizinhos horizontais e verticais (vizinhança-4) e vizinhos diagonais (vizinhança-8 e vizinhança-D), conforme Figura 7. (GONZALEZ 2000).

Com a análise de vizinhança é possível explorar a diferença de intensidade, cor e textura, obtendo características específicas de pontos que distinguem-se das demais imagens, esses pontos são chamados de pontos de interesse ou pontos chaves. Tais pontos formam uma característica local que é representada pelos pontos de interesse, bordas ou trechos, formando um conjunto de padrão característico dentro da imagem. O desenvolvimento de análise desta característica local forma a base para distinguir um objeto do outro. Para isso é explorada a repetição da imagens, a distinção entre as áreas das imagens, o foco em características locais submetidas a diferentes regiões, a precisão, a eficiência e a quantidade de comparações de imagens. A próxima etapa é realizar a correspondência para comparar e classificar os objetos das imagens, através da descrição das características detectadas, como por texto ou esboço ou outra imagem, conforme Figura 1 (GHELLERE 2015).

Figura 1 – Modelo de correspondência




Fonte: GHELLERE 2015

## 3.2 MACHINE LEARNING

Machine Learning é um sistema computacional programado para aprender atividades simples do ser humano e atividades complexas para computadores. São programas que possuem lógicas que emulam o sistema de aprendizado humano para objetivos determinados. Estas máquinas utilizam algorítmos de aprendizado que evoluem através de informações alimentadas por meio de dados. (MITCHEL 1997). O Machine Learning tem sido muito explorada no campo da visão computacional, seu recurso é muito usado para classificações de imagens, texto em imagens e reconhecimentos faciais (Hosseini 2017). O princípio para iniciar um aprendizado com máquina é elaborar um conjunto de decisões que seja a base para detectar os objetos. Estas decisões estão ligadas a leitura de pixels, como as bordas, cores, iluminação e entre outras características relacionado ao processamento de imagem para diferenciar o objeto A do objeto B (BEYELER 2017).

### 3.2.1 FIREBASE/KIT ML E GOOGLE CLOUD PLATFORM/GOOGLE CLOUD VISION

A plataforma Firebase foi desenvolvida em abril de 2012 e sua principal função é Backend-as-a-Service (BaaS), integrando os serviços de infraestrutura de nuvem e Backend, como serviços de data base, autenticação e entre outros (DEVI 2018). O Firebase tornou-se uma ferramenta poderosa desde 2014, quando integrou-se com a Google e, assim, passou a servir de base para várias aplicações para uso mobile e web. Essa ferramenta fornece um vasto banco de dados, sistema de autenticação

de usuários, armazena e sincroniza dados através da aplicação database, possui um sistema Machine learning integrado e entre outros (LAKSHMI 2020).

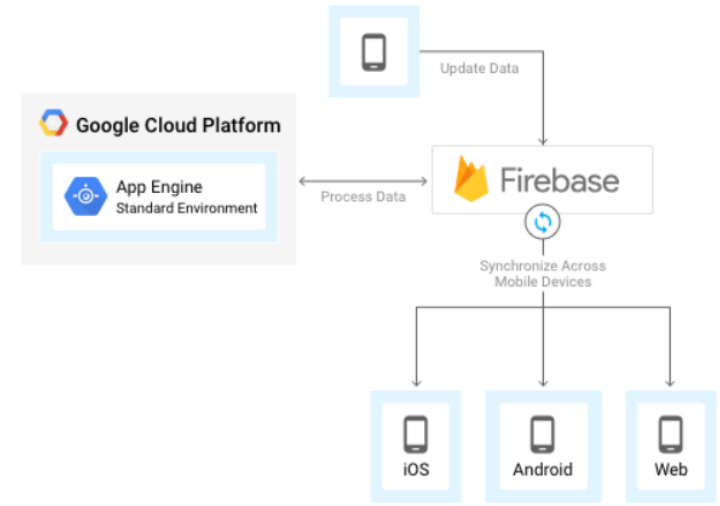
A principal aplicação utilizada neste projeto é o Firebase ML, que fornece o kit ML para desenvolvimento de aplicações voltados a Machine Learning, um pacote com utilização simples para iniciantes que é constituído de APIs treinadas para reconhecimento de texto, detecção facial, rotulação de imagens, reconhecimento de pontos de referências, leitura de código de barras, tradução de dispositivos e respostas inteligentes (Firebase 2018). O ML kit é uma camada que utiliza a tecnologia do Google Cloud Vision, Google services, Android Neural Networks API e tecnologia SDK e é compatível para desenvolvimento em diversos modelos Android e Ios. Além dos APIs pré-treinados, é possível incluir o seu próprio API treinado, através do uso de TensorFlow Lite (LAKSHMI 2020).

O Kit ML está diretamente ligado ao Google Cloud Vision que possui APIs desenvolvidas pela Google e utiliza o banco de dados do Google Cloud Platform (GCP) para treinar as imagens em Machine Learning, facilitando a extração de informações relevantes das imagens. Dentro deste API é possível executar funções de reconhecimento de objetos, textos, faces humanas e até códigos de barras, sendo ideal para programadores iniciantes (MULFARI 2016).

O Google Cloud Vision oferece a leitura por meio de fotos ou câmera ao vivo e é possível detectar no máximo 5 objetos na imagem, a identificação é explorada desde reconhecimento de objetos até reconhecimento por classificação ampla. Além disso, o objeto detectado e rastreado passa para uma nuvem back-end (ML KIT 2020).

Após um breve relato acima, percebe-se que a plataforma GCP e Firebase podem ser trabalhadas separadamente dependendo da situação. Entretanto as duas ferramentas se interligam quando o assunto é construção de aplicativos móveis ou web. Então, o Firebase necessita do GCP para ampliar suas funcionalidades back-end e o GCP precisa do Firebase para conversar com o aplicativo. A Figura 2 demonstra claramente esta relação. Portanto, neste projeto necessita do Firebase para entrosar com o Android e utilizar ferramentas do GCP devido ao Google Cloud Vision API, não sendo possível ainda trabalhar somente com o GCP no Android (Stevenson 2019).

Figura 2 – Firebase e Google Cloud Platform




Fonte: Google Cloud 2020

## 3.3 SISTEMA ANDROID/ ANDROID STUDIO

Neste projeto foi escolhida a plataforma Android como área de desenvolvimento, O Android é um software um código aberto com base em LINUX. Esta plataforma foi desenvolvida em 2005 pela Google ao adquirir a Android Inc e tem crescido desde então, chegando a mais de 6 milhões de pessoas utilizando a plataforma. Em geral, o sistema operacional do Android é trabalhado em linguagem Java através do Android SDK (Silva 2015).

O Android Studio foi a plataforma escolhida para desenvolver o ambiente Android, que possui a função de criar aplicativos móveis (apenas para Android) em linguagem Java com C++ /C ou Klotlin, podendo ser desenvolvido em plataformas com sistemas operacionais MAC, Linux e Windows. O Android Studio foi financiado pela Google e pela Open Handset Alliance (OHA) (VAN DRONGELEN 2015). No Android foi integrado funções da ferramenta Kit Firebase ML para chamar aplicações do Google Cloud Vision API, programando em linguagem Java.

### 3.3.2 SINTENTIZADOR DE VOZ

O sintetizador é uma máquina que simula o som da voz humana, podendo ser através de uma imitação de um som através de combinação de parâmetros, conhecido como Speech-To-Speech system (STS) ou a máquina realizando a imitação por meio de um sistema de Text-To-Speech (TTS).

O sistema escolhido para aplicação do projeto é o TTS, devido às respostas do Google Cloud Vision API serem por meio de texto (ARAUJO 2015).

## 4 RESULTADOS

Inicialmente, as ideias do desenvolvimento do aplicativo foram listados na tabela **1.**

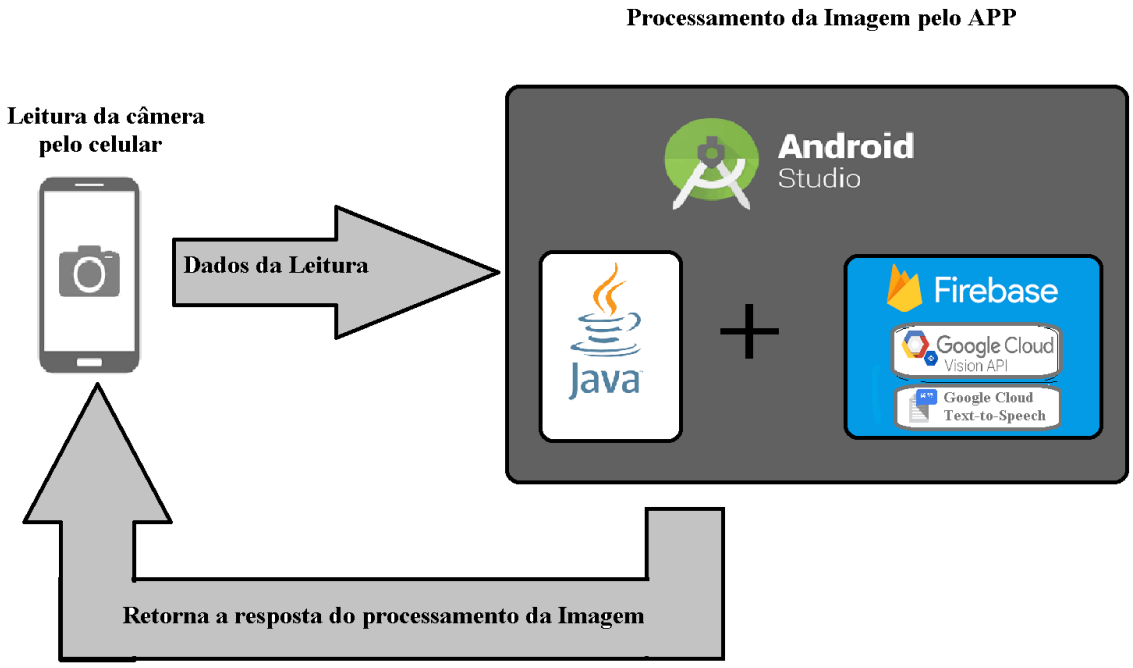
Tabela 1 – Tabela de elaboração inicial do aplicativo

| Prioridade | Problema | Respostas do problema |
|---|---|---|
| 1 | Quais imagens a serem detectadas? | A maioria dos objetos que são obstáculos a frente |
| 2 | Como realizar a detecção do objeto na imagem? | Utilizando recursos do Google Cloud Vision API |
| 3 | Como coletar estas informações? | Atrávés da câmera do celular |
| 4 | Como gerar uma função que defina o objeto ou obstáculo? | Executando funções pré-treinadas do Kit ML do Firebase |
| 5 | Como pegar o retorno desta função para ativar uma ação a ser tomada? | Utilizando recursos do Google Cloud Text-To-Speech (TTS) |
| 6 | Como comunicar este retorno para o celular? | Aplicanto sintentizador de voz |

Fonte: Própria autoria

A partir desta tabela, foi elabora uma visão geral do trabalho como mostrado na Figura 3. A câmera funciona em tempo real, recolhendo dados da imagem e transferindo ao aplicativo que seguirá as funcionalidades determinadas na programação Android Studio, com linguagem Java C/C++. O reconhecimento da imagem ocorre pelo Google Cloud Vision API e a resposta é processada pelo Text-To-Speech por intermédio de voz.

Figura 3 – Visão Geral do aplicativo




Fonte: Própria Autoria

Foi instalado o Android Studio versão 4.1.1 e Java versão 8 com atualização 271 no sistema operacional Windows 10 no notebook com processador i3 de 64 bits. A simulação do ambiente do APP foi através de um celular real para consumir menos memória do computador. Em seguida, é escolhido um nível de API acima de (Jelly Bean) e versão Gradle superior a 4.1, para rodar o Firebase. Em seguida, foi realizado a comunicação do Firebase com Android Studio.

## 4.1 IMPLEMENTANDO AS FUNCIONALIDADES DO RECONHECIMENTO DE OBJETOS DO KIT ML

Após realizar a sincronização do Firebase, foi instalada a biblioteca do Firebase Vision: “implementation 'com.google.firebase:firebase-ml-vision:24.0.3'” e “implementation 'com.google.firebase:firebase-ml-vision-object-detection-model:19.0.6”, mostradas na FIgura 4. Após isso, é criado outro arquivo com nome “ReconhecimentoActivity.java”, onde é utilizado a função “FirebaseVisionObjectDetectorOptions” para realizar a configuração da saída do Machine Learning. Para isto, existem duas opções, a escolha por detecção e rastreamento em vídeo e o outro por imagem parada. Como a câmera escolhida foi ao vivo, então implementamos o código abaixo, baseado na documentação do site Firebase. Este código teve como base os tutoriais do site Firebase para exploração da Visão Computacional (Firebase 2020).

Figura 4 – Código para chamar a funcionalidade do Firebase Vision Object

```
51        FirebaseVisionObjectDetectorOptions options =
52                new FirebaseVisionObjectDetectorOptions.Builder()
53                        .setDetectorMode(FirebaseVisionObjectDetectorOptions.STREAM_MODE)
54                        .enableClassification()  // Optional
55                        .build();
56
57
58    //for default setting:
59              FirebaseVisionObjectDetector objectDetector =
60                      FirebaseVision.getInstance().getOnDeviceObjectDetector();
```

Fonte: Própria Autoria.

Assim, por meio do objeto “FirebaseVisionImage” é executado o comando do detector da imagem, chamada pelo “objectdetector, que possui o comando direto para ler a imagem na câmera.

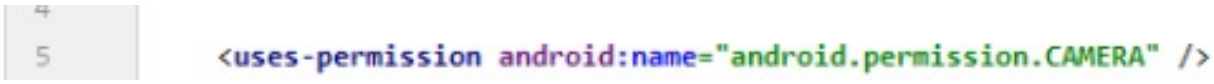
E também foi implementado um outro arquivo “RotulagemActivity.java”, que usa os mesmos recursos do Google Cloud Vision API, faz reconhecimentos de objetos também, porém a detecção são por classes mais amplas, por rótulos, como o lugar do ambiente, pessoas e entre outros, o código também teve como base os ensinamentos da área de guia do Firebase (Firebase 2020).

Após o esse processo, as respostas das funcionalidades dos arquivos deste Kit ML são chamado pelo arquivo principal “TotalActivity.java”, onde o endereços dos demais arquivos são chamado por meio do comando “private” e os comandos de botão são configurados.

## 4.2 CHAMANDO O RECURSO DA CAMERA NO APLICATIVO

Após a sincronização do Firebase com o Android Studio, a próxima etapa é programar para que seja utilizado o recurso da câmera. Para isso é necessário permitir e solicitar o uso da câmera por meio do arquivo “AndroidManifest.xml”, onde é definido as permissões de uso como demonstrado na Figura 5 (Silva 2015).

Figura 5 – Permitindo o uso da câmera




Fonte: extraído do próprio projeto

Para trabalhar com a câmera em tempo real, é necessário utilizar sobreposições de gráficos para visualização de coordenadas da imagem, melhorando a performance do Google Cloud Vision, através de taxas de quadros (KIT ML 2020). Foi criado o arquivo “SuperposiçãoGrafica.java” para configurar o ambiente deste gráfico de sobreposição, primeiro é chamado a função “overlay” para ajustar valores horizontais e verticais, suas escalas e suas coordenadas. Depois é removido todos os gráficos de sobreposição e adicionado um outro gráfico de sobreposição pelo comando “grafics.clear” e “grafics.remove “, finalmente é definido a direção e o tamanho por intermédio da câmera e desenha a sobreposição que é retornado pelo comando “graphic.draw”, tal código foi baseado na fonte no Google Github desenvolvido por Clayton Wilkison (Github 2015).

No arquivo “CameraActivity.java” foi criada para configurar entradas e saídas da câmera, a câmera é ativada somente se as funcionalidades dos arquivos “CameraFirebaseActivity.java” e “SuperposiçãoGrafica.java” estiverem ativas, sendo acionada por meio do comando “start “. Outras condições foram expostas no programa, foi implementado comandos da classe “surfaceView” que desenha superfície para preservar a qualidade da imagem (Developers 2020). O código deste arquivo teve como base no código GitHub desenvolvido pelo usuário Nagendrababu143 (Github 2018).

A “CameraFTActivity.java” foi criada para configurar os botões de controle da câmera frontal e câmera traseira e seus condicionamentos de ativação.

Logo após, é criado uma classe “TotalActivity.java”, que depende das outras três classes, “CameraFTActivity.java”, “CameraActivity.java” e “SuperposiçãoGrafica.java” para configurar o

funcionamento da câmera. Dentro deste arquivo chama-se as funcionalidades do arquivo “ReconhecimentoActivity.java” e “RotulagemActivity.java”, além de configurar a entrada e saída do botão de troca de funcionalidades do kit ML e códigos programados para identificar problemas nos demais arquivos.

Em seguida, é necessário configurar a primeira página do APP, que é configurado na pasta “layout” por meio do arquivo “activity.xml” no Android Studio, definindo a margem de 10dp em relação ao ambiente e o espaço da câmera e da sobreposição gráfica que foi definido como “math_parent” para ocupar o máximo de espaço, o botão na primeira página do APP que definem a troca da câmera frontal com a câmera traseira e vice-versa.

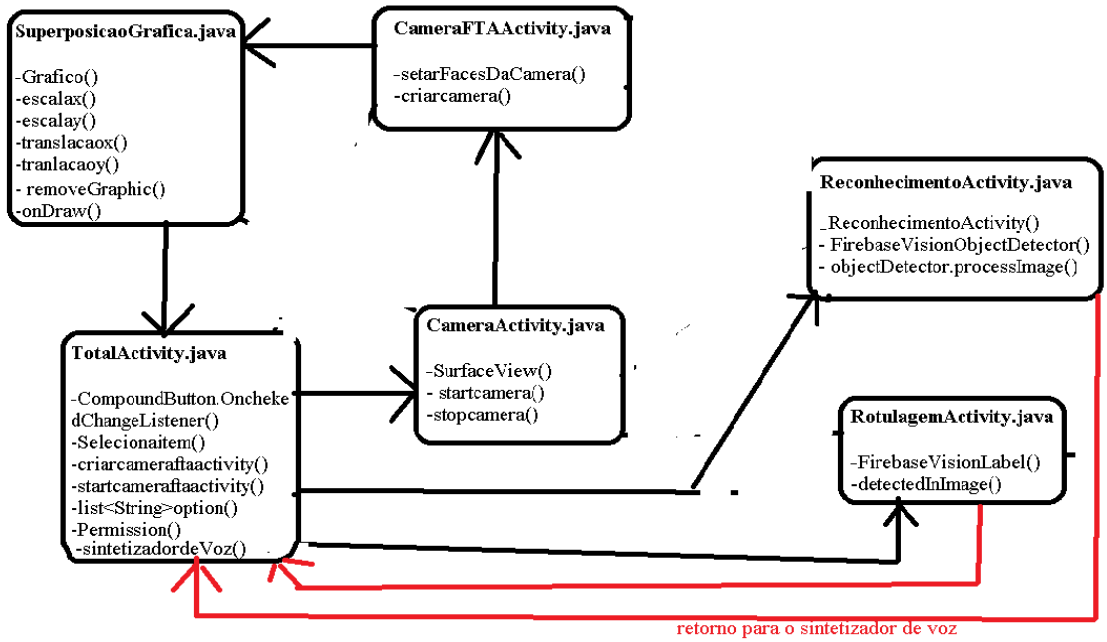
## 4.3 IMPLEMENTAÇÃO DO SINTETIZADOR DE VOZ

O sintetizador é implementado através do comando “TextToSpeech”, dentro do código do arquivo “TotalActivity.java”, através da classe “private”, que recebe o texto através da saída dos “ReconhecimentoActivity.java” e “RotulagemActivity.java”, sendo codificado pelo comando “TextToSpeech”.

## 4.4 ARQUITETURA DO PROJETO

A arquitetura do projeto mostrado na Figura 6, foi inspirada nos estudos em aula do professor Yusuf Saber do Curso Undemy de Machine Learning For Android APP Devlopment Using ML Kit.

Figura 6 - Arquitetura do projeto




Fonte: Própria Autoria

## 4.5 MANUAL DO USO

Ao clicar no Aplicativo, a página imediatamente aciona a câmera. Na figura 7 é utilizado a função de Rotulagem de Imagem, que apenas detecta imagens como um grupo, por exemplo, a parede, pessoas, cômodos da casa e entre outro. Na figura 8 é utilizado a função de Detecção de Objetos, sendo possível detectar objetos sem estar em um grupo, por exemplo, relógio, morango e entre outros. Verifica-se que a precisão foi melhor com a função rotulagem, ela detectou perfeitamente a parede, enquanto o função detecção de objetos apresentou certas falhas, apesar de ter identificado o relógio digital, ele apontou também que tinha um peixe e papel higiênico no ambiente.

Figura 7 – Rotulagem da imagem


Fonte: Própria Autoria.

Figura 8- Detecção de Objetos


Fonte: Própria Autoria

## 5 RESULTADOS E DISCUSSÕES

Foram coletados 15 amostras para a análise de detecção de objetos e 15 amostras para detectar rótulos. Na tabela 2, é apresentado os resultados para a detecção de objetos, das 15 amostras, o resultado total foi de 24% de acertos, muito abaixo da média esperada, este API não é recomendável para detectar imagens em câmera ao vivo devido a perdas de informações para detectar os objetos, conforme os resultado, havia vários momentos em que um objeto era confundido por uma banheira. Foram observados que a eficiência é melhor quando a câmera está parada e o objeto está bem próximo, um exemplo dessa eficácia é demostrada no trabalho de Jean Carlos Franco (FRANCO 2018) e no experimento de reconhecimento do relógio no item anterior, em que a câmera estava bem próxima e parada.

Tabela 2 – Teste detecção de objetos

| Detalhe do ambiente | Respostas do Aplicativo | % de acertos | O que identificou |
|---|---|---|---|
| 4 Canetas roxo, azul, laranja e lapiseira branca em cima de um caderno vermelho | Chave de fenda<br>Seringa<br>Caneta tinteiro | 34 | Objeto |
| Computador com uma mochila do lado | Laptop<br>Computador<br>Projetor | 67 | Objeto |
| Tênis azul e verde limão no chão do lado de uma parede azul e chão bege | Máscara de oxigênio<br>Balão<br>Barco pegando fogo | 0 | Obstáculo |
| Jarra de água e garrafa de água em cima da mesa | Garrafa de Água<br>Jarro de água<br>Cofrinho | 67 | Objeto |
| Pé da cama e canto de uma mesa de computador com caixa embaixo da mesa | Banheira<br>Interruptor<br>Caixa | 34 | Obstáculo |
| Obstáculos de um vaso sanitário | Banheira<br>Banheira<br>Mosquito | 0 | Obstáculo |
| Parede Branca | Parede<br>Máquina de lavar<br>Interruptor | 34 | Obstáculo |
| Cadeira com uma mesa e em cima da mesa papel toalha | Papel toalha<br>Sofá<br>Banheira | 34 | Objeto |
| Planta em um vaso | Banheira<br>Banheira<br>Berço | 0 | Objeto |
| Máquina de lavar vista de cima | Banheira<br>Banheira<br>Mosquito | 0 | Objeto |
| Parede cinza | Cortina<br>Disco<br>Papel toalha | 0 | Obstáculo |
| Teclado de computador | Barra de espaço (tecla)<br>Teclado de computador<br>Filtro | 67 | Objeto |
| Escada de cimento | Gêiser<br>Parede<br>Parede | 0 | Obstáculo |
| Sofá azul e parede branca | Terno<br>Calção de natação<br>Parede | 34 | Obstáculo |
| Um degrau bege com tapete vermelho | Asa | 0 | Obstáculo |

| | | | |
|---|---|---|---|
| | Abutre<br>Banheira | | |

Fonte: Própria Autoria

Na tabela 3 foi realizado testes com 15 amostras para o reconhecimento por rotulagem. Os resultados apontaram 74% de acertos, entretanto nota-se que o número de detecção de objetos ou classes são bem menores comparado ao API de detecção de objetos.

Tabela 3 – Teste de Reconhecimento de Rótulos

| Detalhe do ambiente | Respostas do Aplicativo | % de acertos | O que identificou |
|---|---|---|---|
| TV e hacker | Sala | 100 | Local |
| Cama, travesseiro e cobertor | Quarto | 100 | Local |
| Sofá | Sala | 100 | Local |
| Vaso sanitário, azulejo e pia | Parede | 67 | Local |
| | Banheiro | | |
| | Telha | | |
| Foi mostrado uma parte do hacker | Ladrilho da sala | 50 | Partes do local |
| Chão bege | Ladrilho | 100 | Material |
| Parede Branca | Pele | 0 | Superfície |
| Parede Branca | Parede | 100 | Superfície |
| Cadeira com mesa e objetos sobre a mesa | Cadeira | 100 | Objeto |
| Fogão, pia e panelas | Sala | 0 | Local |
| Mão | Unha | 100 | Partes do corpo |
| | Mão | | |
| Perna e pé | Pé | 100 | Partes do corpo |
| Escada | Não identificou | 0 | Obstáculo |
| Câmera apontada no céu | Céu | 100 | Espaço |
| Tanque e máquina de lavar | Não identificou | 0 | Local |
| Muro | Parede | 100 | Superfície |
| Rosto | Cabelo | 100 | Parte do corpo |
| | Óculos | | Objeto |
| Rua, carro, casa e calçada | Asfalto | | Local |
| | Veiculo | 100 | Transporte |
| | Infraestrutura | | |

Fonte: Própria Autoria.

Embora o API de rotulagem tenha mostrado eficácia de 74%, em base aos resultados levantados, apenas 12,5% conseguiu detectar objetos, 19% identificou partes do corpo, 50% detectou

locais, 18% identificou superfícies e 0,5% conseguiu classificar meios de transportes, com base nos testes realizados. O resultado demonstrou que a superposição gráfica auxiliou no desempenho da leitura pela câmera em tempo real.

Analisando o desenvolvimento dos dois APIs Machine Learning pré-treinados, a identificação de todos os obstáculos não foi possível em ambos os casos, o API de rotulagem das imagem é o mais adequado para o uso deste protótipo, conforme experimentos realizados, é possível a detecção de alguns obstáculos, com as paredes, janelas, cadeiras e veículos a frente, conforme resultados acima. Podendo ainda incrementar com a detecção do ambiente local, assim o deficiente poderá também saber em que tipo de ambiente se encontra.

A comparação dos dois resultados reforçam a importância do treinamento Machine Learning, apesar de serem APIs da mesma plataforma, tiveram treinamento diferente, causando impactos no resultado final.

## 6 CONSIDERAÇÕES FINAIS

Diante das dificuldades enfrentadas pelos deficientes visuais, este estudo foi levantado com o intuito de auxiliar os deficientes visuais através de detecção de objetos e obstáculos. Esta pesquisa mostra a eficiência da implementação da Visão Computacional em Machine Learning no celular e o quanto essa ferramenta cresceu, como recursos de altíssima qualidade, como o Google Cloud. Os resultados desta pesquisa alcançou o objetivo de detectar os obstáculos através do API Google Cloud Vision, com 74%de eficácia e detecção de alguns obstáculos, juntamente com detecção de outras classes, como locais e partes do corpos, pois foi mais eficaz em reconhecimento de Rótulos do que reconhecimento de Objetos através da câmera em tempo real.

Entretanto, o reconhecimento por rotulação não abriga um vasto repertório para a detecção de obstáculos e o API Object Detect and Tracking detecta todos os objetos na imagem, mas não localiza isoladamente o objetos especificado. Esta situação demostra a necessidade de criar novos modelos em Machine Learning que atenda a necessidade imposta com resultados melhores. Outro problema enfrentado a questão da detecção de objeto limitado a distância e ainda com a execução em câmera ao vivo, decaindo a porcentagem de acertos nos resultados.

Portanto, é proposto como trabalho futuro o desenvolvimento em Machine Learning através da ferramenta TensorFlow para treinar a máquina de aprendizado a detectar objetos que são considerados especificamente obstáculos e a melhora no desenvolvimento da Sobreposição Gráfica para a melhora do desempenho de leitura da câmera, eliminando ruídos e outras interferências. O TensorFlow abre possibilidades de desenvolvimento no Colab e na própria plataforma do Google Cloud.

## REFERÊNCIAS


ALI, Nursabillilah Mohd, et al. **Object classification and recognition using Bag-of-Words(BoW) model**. In: *2016 IEEE 12th International Colloquium on Signal Processing & Its Applications (CSPA)*. IEEE, 2016. p. 216-220.

ANTONELLO, Ricardo. **Introdução a Visão Computacional com Python e OpenCV**. Curso de Inteligência Artificial.

ARAÚJO, Fabíola Pantoja Oliveira; JÚNIOR, Aldebaro Barreto da Rocha Klautau. **Imitação da Voz Humana através do Processo de Análise-por-Síntese utilizando Algoritmo Genético e Sintetizador de Voz por Formantes**. 2015. Tese de Doutorado. Tese de doutorado, Universidade Federal de Santa Catarina.

BEYELER, Michael. **Machine Learning for OpenCV.** Birmingham, UK : Packt Publishing, 2017.

CHINCHA, Ricardo; TIAN, YingLi**. Finding objects for blind people based on SURF features.** In: *2011 IEEE International Conference on Bioinformatics and Biomedicine Workshops (BIBMW)*. IEEE, 2011. p. 526-527.

DEVELOPERS. **SurfaceView**. Disponível em: <https://developer.android.com/reference/android/view/SurfaceView> Acesso em 03 de Novembro de 2020.

DEVI, Shivali; PATERIYA, **Pushpendra Kumar. Automatic system for recognition and identification of human objects**. In: *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 2017. p. 236-244.

DUARTE, Glaucius Décio. **Introdução ao OpenCV**. Apostila do curso de Engenharia Elétrica Processamento de Imagens Digitais do Instituto Federal Sul-rio-grandense,2018.

DUARTE, Luiz. **Tudo sobre o Gradle – Android Studio**. Disponível em: <https://www.luiztools.com.br/post/tudo-sobre-o-gradle-android-studio/ > Acesso em 7 de setembro de 2020.

EZE, Ekene. Build **A Simple Blog App With Firebase In Android Studio**. Disponível em: <https://medium.com/@peterekeneeze/build-a-simple-blog-app-with-firebase-in-android-studio-b6482275408 > Acesso em 20 de Novembro de 2020

FIREBASE. **Adicionar o Firebase ao projeto para Android**. Disponível em: <https://firebase.google.com/docs/android/setup?hl=pt-br > Acesso em 17 de Agosto de 2020.

FIREBASE. **Image Labeling.** Disponível em: <https://firebase.google.com/docs/ml-kit/label-images> Acesso em 15 de Novembro de 2020.

FRANCO, JEAN CARLOS. **APLICATIVO PARA O RECONHECIMENTO DE OBJETOS EM IMAGENS UTILIZANDO A API CLOUD VISION DESTINADO A PESSOAS PORTADORAS DE DEFICIÊNCIA VISUAL**. 2018

GHELLERE, Jhonattan Salvador. **Detecção de Objetos em Imagens por Meio da Combinação de Descritores Locais e Classificadores**. Trabalho de Conclusão de Curso – Departamento Acadêmico de Computação, Universidade Tecnológico Federal do Paraná. Medianeira, p.90. 2015.

GONZALEZ, Rafael C.; WOODS, Richard E. **Processamento de imagens digitais**. Edgard Blucher, 2000.

GOOGLE CLOUD. **Build an Android App Using Firebase and the App Engine Flexible Environment**. Disponível em: <https://cloud.google.com/solutions/mobile/mobile-firebase-app-engine-flexible> Acesso em 01 de Outubro de 2020.

GOOGLE CLOUD. **Processing User-Generated Content Using the Cloud Video Intelligence and Cloud Vision APIs**. Disponível em: <https://cloud.google.com/solutions/processing-marketing-submissions-using-video-intelligence?hl=pt-br> Acesso em 28 de Agosto de 2020.

HOSSEINI, Hossein; XIAO, Baicein; POOVENDRAN, Radha**. Google's Cloud Vision API Is Not Robust To Noise**. In: IEEE INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND APPLICATIONS, 16, 2017, Cancun. Proceedings... Cancun: IEEE, 2017. p. 101-105

JABNOUN, Hanen; BENZARTI, Faouzi; AMIRI, Hamid. **Visual substitution system for blind people based on SIFT description**. In: *2014 6th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*. IEEE, 2014. p. 300-305.

LAKSHMI, K.; VANI, Atmakur; SRINIVASULU, Banda; SHAIKSHAVALI, Kadapa. **ML KIT IN FIREBASE FOR APP DEVELOPMENT**. International Research Journal of Engineering and Technology (IRJET): v.7, 2020.

MANDUCHI, Roberto; KURNIAWAN, Sri. **Mobility-related accidents experienced by people with visual impairment.** AER Journal: Research and Practice in Visual Impairment and Blindness, v. 4, n. 2, p. 44-54, 2011

MEDEIROS, Henrique.**92% dos brasileiros possuem ou usam smartphones com frequência.** Disponível em: https://www.mobiletime.com.br/noticias/18/10/2018/92-dos-brasileiros-possuem-ou-usam-smartphones-com-frequencia/. Acesso em 26/10/2019.

MEKHALFI, Mohamed Lamine, et al. **A compressive sensing approach to describe indoor scenes for blind people**. *IEEE Transactions on Circuits and Systems for Video Technology*, 2014, 25.7: 1246-1257.

ML Kit. **Object Detection and Tracking**. Disponível em: < https://developers.google.com/ml-kit/vision/object-detection> Acesso em 17 de Agosto de 2020.

MULFARI, Davide et al. Using Google Cloud Vision in assistive technology scenarios. In: **2016 IEEE Symposium on Computers and Communication (ISCC)**. IEEE, 2016. p. 214-219.

NAGENDRABABU143. **Github - nagendrababu143/CameraSourcePreview.java**. Disponível em:<https://gist.github.com/nagendrababu143/d4fc044b008fd45a7169bdcd50ff5ab8/revisions?diff=split> Acesso em 5 de Outubro de 2020.

NASCIMENTO, Maíra Carneiro. **Detecção de Objetos em Imagens**. Trabalho de Graduação – Centro Informática, Universidade Federal de Pernambuco. Recife, p. 55. 2015.

NUNES, Lucas Dal Piaz. **Algoritmos para processamento de imagens visando implementação em FPGA**. 2014.

OLIVEIRA JR, Eudes Quintino de. **O portador de deficiência visual e o cão-guia.** JusBrasil, 2016. Disponível em: <https://eudesquintino.jusbrasil.com.br/artigos/332707039/o-portador-de-deficiencia-visual-e-o-cao-guia>. Acesso em: 20 de fevereiro de 2019.

PINHO, José Mário Veras de; SOUZA, Amanda da Silva; OLIVEIRA, Denia Elice Matias de; COSTA, Evaldo Sávio Silva; FILHO, Oscar Machado da Cunha; SOUSA, Roniere da Silva; ARAÚJO, Francisco Marcelino Almeida de. Prototipagem de dispositivo de auxílio a deficientes visuais com hardware livre. **Sistema Olimpo**, 2011. Disponível em: <http://www.sistemaolimpo.org/midias/uploads/c1cb591d95fc50f7c934992cb7928cae.pdf >. Acesso em: 17 de janeiro de 2019.

Robson, Sara. Medium, Disponíve em: < https://medium.com/@srobtweets/exploring-the-cloud-vision-api-1af9bcf080b8> Acesso em 8 agosto de 2020

SILVA, Luciano Alves da. **Apostila de Android-Programando Passo a Passo Programação Básica (Versão Android Studio)**. Rio de Janeiro, RJ. 2015.

**Sobre a Interação com voz no android**. Revista Programar. Disponível em: https://www.revista-programar.info/author/lmmsoares/> Acesso em 21 de Outubro de 2019

SOUSA, Silva. Acessibilidade é essencial para que pessoas com deficiência exerçam seus direitos. **Rede Mobilizadores**, 2015. Disponível em: <http://www.mobilizadores.org.br/entrevistas/acessibilidade-e-essencial-para-que-pessoas-com-deficiencia-exercam-seus-direitos-sociais/>. Acesso em: 13 de janeiro de 2019.

SOUSA, Kelly Aparecida Oliveira, et al. **Uso de visão computacional em dispositivos móveis para auxílio à travessia de pedestres com deficiência visual**. 2013

STEVENSON, Doug. **What's the relationship between Firebase and Google Cloud?** Disponível em : <https://medium.com/google-developers/whats-the-relationship-between-firebase-and-google-cloud-57e268a7ff6f > Acesso em 27 de Agosto de 2020.

TAKIZAWA, Hotaka et al. Kinect cane**: An assistive system for the visually impaired based on three-dimensional object recognition**. In: 2012 IEEE/SICE international symposium on system integration (SII). IEEE, 2012. p. 740-745.

TOM M. MITCHELL. **Machine learning**. New York: Mcgraw-hill Science/engineering/math, 1997. 421 p.

VAN DRONGELEN, Mike. **Android studio cookbook**. Packt Publishing Ltd, 2015.
ROBINSON, Sara. **Exploring the Cloud Vision API**. Disponível em: <https://medium.com/@srobtweets/exploring-the-cloud-vision-api-1af9bcf080b8> Acesso em 25 de setembro de 2020.

**VISÃO computacional pode resolver muitos desafios da indústria**; entenda. 2019. AVOZDAINDUSTRIA. Disponível em: <http://https://avozdaindustria.com.br/gest-o/vis-o-computacional-pode-resolver-muitos-desafios-da-ind-stria-entenda>. Acesso em: 25 Oct. 2019.

Wilkinson, Clayton. **Github - googlesamples/android-vision** . Disponível em : <https://github.com/googlesamples/android-vision/blob/master/visionSamples/FaceTracker/app/src/main/java/com/google/android/gms/samples/vision/face/facetracker/ui/camera/GraphicOverlay.java> Acesso em 20 de Outubro de 2020.

YU, Lijun, et al**. The design of single moving object detection and recognition system based on OpenCV**. In: *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2018. p. 1163-1168.

ZEREEN, Aniqua Nusrat; CORRAYA, Sonia**. Detecting real time object along with the moving direction for visually impaired people**. In: *2016 2nd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE)*. IEEE, 2016. p. 1-4.

## AGRADECIMENTOS

Primeiramente, agradeço a Deus pela sabedoria e capacidade que Ele me proporcionou durante o desenvolvimento, pois toda a sabedoria e força vem dEle. “E o Espírito de Deus o encheu de sabedoria, entendimento, ciência e em todo o lavor,” Êxodo 35:31.

Agradeço a minha família e a meus amigos pelo apoio e encorajamento para prosseguir na vida acadêmica.

Agradeço ao professor e orientador Dr. Paulo Lopes Batista que me instruiu e apoiou em todo a trajetória do TCC.

Agradeço ao professor Dr. Antonio Newton Licciard Junior pela contribuição em compartilhar seus conhecimentos gerais sobre o tema e o apoio durante as dificuldades.

Agradeço ao professor Dr. Mario Olimpio de Menezes por compartilhar seus conhecimentos sobre visão computacional.

Agradeço ao professor Felipe Perrella que me auxiliou e instruiu acerca do uso do Android Studio.