UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETROTÉCNICA
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

IAGO CAVALARO MARQUES

SISTEMA DE RECONHECIMENTO ÓPTICO DE CARACTERES PARA LEITURA DE GLICOSÍMETRO CONVENCIONAL COMO FERRAMENTA DE AUXÍLIO PARA DEFICIENTES VISUAIS

CURITIBA
2015

IAGO CAVALARO MARQUES

SISTEMA DE RECONHECIMENTO ÓPTICO DE CARACTERES PARA LEITURA DE GLICOSÍMETRO CONVENCIONAL COMO FERRAMENTA DE AUXÍLIO PARA DEFICIENTES VISUAIS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina Trabalho de Conclusão de Curso, do Curso Superior de Engenharia de Controle e Automação do Departamento Acadêmico de Eletrotécnica – DAELT – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Engenheiro de Contole e Automação.

Orientador: Prof. Dr. Amauri Amorin Assef
Co-orientador: Prof. Me. Douglas Roberto Jakubiak

CURITIBA

2015

Iago Cavalaro Marques

Sistema de reconhecimento óptico de caracteres para leitura de glicosímetro convencional como ferramenta de auxílio para deficientes visuais

Este Trabalho de Conclusão de Curso de Graduação foi julgado e aprovado como requisito parcial para a obtenção do Título de Engenheiro de Controle e Automação, do curso de Engenharia de Controle e Automação do Departamento Acadêmico de Eletrotécnica (DAELT) da Universidade Tecnológica Federal do Paraná (UTFPR).

Curitiba, 09 de julho de 2015.

_____
Prof. Paulo Sérgio Walenia, Esp.
Coordenador de Curso
Engenharia de Controle e Automação

_____
Prof. Prof. Amauri Amorin Assef, Dr.
Responsável pelos Trabalhos de Conclusão de Curso
de Engenharia de Controle e Automação do DAELT

ORIENTAÇÃO

_____
Amauri Amorin Assef, Dr.
Universidade Tecnológica Federal do Paraná
Orientador

_____
Douglas Roberto Jakubiak, Me.
Universidade Tecnológica Federal do Paraná
Co-Orientador

BANCA EXAMINADORA

_____
Amauri Amorin Assef, Dr.
Universidade Tecnológica Federal do Paraná

_____
Roberto Cesar Betini, Dr.
Universidade Tecnológica Federal do Paraná

_____
Daniel Balieiro Silva, Me.
Universidade Tecnológica Federal do Paraná

A folha de aprovação assinada encontra-se na Coordenação do Curso de Engenharia de Controle e Automação

RESUMO

MARQUES, Iago. **Sistema de Reconhecimento Óptico de Caracteres para Leitura de Glicosímetro Convencional como Ferramenta de Auxílio para Deficientes Visuais**. 2015. 79f. Trabalho de Conclusão de Curso (Graduação) – Curso de Engenharia de Controle e Automação. Universidade Tecnológica Federal do Paraná. Curitiba, 2015.

Este trabalho trata sobre o estudo, desenvolvimento e aplicação do conceito de reconhecimento óptico de caractere (OCR) para leitura do *display* segmentado de um glicosímetro no âmbito da tecnologia assistiva. A pesquisa consistiu em utilizar a imagem adquirida pela câmera de um *smartphone*, transmiti-la via Wi-Fi para um computador pessoal, onde, utilizando-se funções desenvolvidas em MATLAB, foi feito o processamento da imagem, e o reconhecimento de caracteres, assim como o retorno por áudio do valor exibido no *display*. Também são discutidas as possíveis aplicações e trabalhos futuros, dificuldades e os resultados do trabalho.

Palavras-chave: Glicosimetro. MATLAB. Reconhecimento Óptico de Caractere. *Smartphone*. Tecnologia Assistiva.

ABSTRACT

MARQUES, Iago. **Optical Caracter Recognition System Applied to the Reading of Conventional Glucometer as an Aid Tool for Visually Impaired**. 2015. 79f. Term Paper (Graduation) - Control and Automation Engineering. Federal Technological University of Paraná. Curitiba, 2015.

This work envolves the study, development and application of the concept of optical character recognition (OCR) in order to read the segmented display of a glucometer within the assistive technology. In this research we use the image acquired by the camera of a smartphone, transmit it via Wi-Fi to a PC, where, using functions developed in MATLAB, the image processing and character recognition was done, and thus informing via audio the values shown on the display. Also possible applications in future work, difficulties encountered in the process and the work results were discussed

Keywords: Glucometer. MATLAB. Optical Character Recognition. Smartphone. Assistive Technology.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

CPClin	Centro de Pesquisas Clínicas
DM	Diabetes *Mellitus*
DMT1	Diabetes *Mellitus* Tipo 1
Hi-Fi	*High Fidelity*
IBGE	Instituto Brasileiro de Geografia e Estatística
IEEE	*Institute of Electrical and Electronics Engineers*
IP	*Internet Protocol*
ISM	*Industrial, Scientific and Medical*
LADA	*Latent Autoimmune Diabetes in Adults*
LAN	*Local Area Network*
LED	*Light-Emitting Diode*
MATLAB	*Matrix Laboratory*
MJPEG	*Motion Joint Photographics Experts Group*
MSER	*Maximally Stable Extremal Regions*
OCR	*Optical Character Recognition*
OMS	Organização Mundial da Saúde
PC	*Personal Computer*
PROTA	Programa de Tecnologia Assistiva
RD	Retinopatia Diabética
RGB	*Red-Green-Blue*
SBD	Sociedade Brasileira de Diabetes
TCC	Trabalho de Conclusão de Curso
Wi-Fi	*Wireless Fidelity*

SUMÁRIO

1 INTRODUÇÃO

A diabetes é um mal que atinge cerca de 370 milhões de vítimas no mundo, 13,4 milhões delas no Brasil (MELO, 2014). A doença consiste em uma síndrome metabólica de origem múltipla, devido a falta de insulina no organismo causada pela incapacidade do pâncreas em produzi-la na quantia necessária, ou quando o hormônio é incapaz de agir, como no caso da resistência à insulina, o que acaba causando um aumento de glicose no sangue. Como a função da insulina é facilitar a penetração do açúcar nas células, onde é absorvida como fonte de energia, a falta desse hormônio desencadeará um aumento de glicose no sangue e, consequentemente, a diabetes (LACERDA, 2013).

O aumento da glicose sérica pode gerar diversos problemas de ordem oftalmológica, como: diminuição da sensibilidade da córnea com propensão para o aparecimento de ceratites (tipo de inflamação da córnea); úlceras neurotróficas e defeitos epiteliais persistentes; alterações corneanas; catarata (4 vezes mais presente em diabéticos); e, tido como a consequência mais grave da diabetes, a retinopatia diabética (RD). Este último, a RD, é a maior causa de cegueira por diabetes, uma vez que, o aumento da glicose sérica gera outras alterações bioquímicas, que resultam em modificações estruturais da membrana basal dos capilares retinianos das células endoteliais e dos pericitos. Dessa forma, podem ser desencadeadas complicações da doença, como o edema macular e a neovascularização retiniana, que são as principais causas da cegueira por diabetes (FERRARO, 2012).

Quando falamos ema RD, o tempo de duração da diabetes é o fator de risco mais importante a ser levado em conta. Passados 20 anos da evolução da diabetes, a retinopatia diabética estará presente em praticamente todos os portadores de diabetes tipo 1 (insulino dependente) e em cerca de 50 a 80% dos pacientes com o tipo 2 (não insulino dependente). Assim, com base nas informações apresentadas, não é difícil concluir que um grande número de pessoas portadoras de deficiência visual são também portadoras de algum dos tipos de diabetes (FERRARO, 2012).

Foco deste trabalho, os portadores de diabetes tipo 1 precisam aplicar insulina diariamente. Esse processo requer um autoexame para verificação dos níveis de glicose no sangue, processo que é feito por meio de um glicosímetro – dispositivo capaz de medir a concentração de glicose sérica (Figura 1). Entretanto, essa não é

uma tarefa fácil para um deficiente visual, uma vez que, apesar de existirem diferentes tipos de glicosímetros, sua forma de utilização consiste basicamente em furar a ponta do dedo com uma agulha pequena chamada lanceta, a fim de obter uma pequena gota de sangue. Em seguida, coloca-se o sangue em uma tira reagente que é colocada no aparelho e em menos de um minuto o resultado é exibido em um *display*, tarefa que pode apresentar um grande grau de dificuldade para alguém que não enxerga (MINHA VIDA, 2012).

Figura 1 – Exemplo de glicosímetro comercial para medição de taxa de açúcar no sangue.
Fonte: Adaptado de Qingdao HMD Technology And Development Co., Ltd. (2015).

Dessa forma, o desenvolvimento de pesquisas e ações relacionadas à Tecnologia Assistiva, em particular para proporcionar e ampliar as habilidades funcionais de deficientes visuais, também portadores de algum dos tipos de diabetes, vão de encontro às política públicas do governo do País em relação ao bem-estar da população brasileira vítima da doença. Conforme proposto na Ata VII pelo Comitê de Ajudas Técnicas (CAT) da Secretaria de Direitos Humanos da Presidência da República: "Tecnologia Assistiva é uma área do conhecimento, de característica interdisciplinar, que engloba produtos, recursos, metodologias, estratégias, práticas e serviços que objetivam promover a funcionalidade, relacionada à atividade e participação de pessoas com deficiência, incapacidades ou mobilidade reduzida, visando sua autonomia, independência, qualidade de vida e inclusão social" (CAT, 2007).

1.1 TEMA

Na Universidade Tecnológica Federal do Paraná (UTFPR), Câmpus Curitiba, às ações relacionadas à Tecnologia Assistiva são realizadas pelo Programa de Tecnologia Assistiva (PROTA), e incluem basicamente: consultorias e o fornecimento gratuito de bengalas para deficientes visuais. A partir da constatação realizada no PROTA sobre as necessidades apresentadas pela população vítima da doença, vislumbrou-se a possibilidade de pesquisa e desenvolvimento de um sistema de inovação com âmbito da Tecnologia Assistiva. Tal sistema poderá servir como base para pesquisas futuras visando o desenvolvimento de produtos que possam ser comercializados para atendimento direto ao usuário, possibilitando facilitar o autoexame, bem como a ampliação das habilidades funcionais e, consequentemente, uma melhor qualidade de vida.

1.1.1 Delimitação do tema

O presente trabalho está focado no estudo, desenvolvimento e avaliação de um sistema portátil de bancada para reconhecimento óptico de caracteres, baseado em *smartphone* e computador PC (*Personal Computer*), com transferência de dados entre os dispositivos via rede sem fio, para leitura e sinalização sonora das taxas de glicose na corrente sanguínea, ação que é fundamental para determinação e controle das doses ideais do hormônio insulina. Nesta pesquisa, o *smartphone* tem a função de capturar a imagem do *display*, com a taxa do glicosímetro, que é enviada para o PC via rede Wi-Fi – imagem com três caracteres de medida. Neste, a imagem é sujeita às etapas de processamento e reconhecimento de caracteres (OCR – *Optical Character Recognition*) com posterior sinalização sonora do valor de glicose medido.

1.2 PROBLEMAS E PREMISSAS

Todos os portadores de diabetes tipo 1 necessitam da aplicação diária de insulina. Entretanto, este processo acaba sendo de extrema dificuldade, tratando-se

de deficientes visuais. Nesse cenário, são poucos os glicosímetros encontrados no mercado que possuem a leitura do resultado com saída áudio, o que pode faciliar a sua utilização. Entretanto, os glicosímetros com tal recurso não são encontrados no mercado brasileiro, o que torna o aparelho caro, pois além de comprar o dispositivo, seria preciso sempre importar os acessórios (lancetas e fitas retivas) compatíveis com o modelo. Além disso, o procedimento consiste em furar o dedo com uma pequena agulha (lanceta), pressioná-lo até que surja uma pequena gota de sangue, e então colocá-la numa fita reagente e, só assim, inserir no aparelho de medição – em alguns caso o procedimento deve-ser ser realizado até 7 vezes ao dia. Feito isso, o usuário ainda precisa regular a dosagem adequada, de acordo com o resultado do teste, e por fim aplicar em si mesmo a insulina.

Segundo o endocrinologista Freddy Eliaschewitz, diretor do Centro de Pesquisas Clínicas (CPClin) em São Paulo, devido às picadas, 30% dos pacientes, incluindo com e sem deficiência visual, não monitoram a glicemia como deveriam. Entretanto, não existe uma pesquisa exclusiva com pacientes portadores de deficiência visual. Pode-se concluir que esse número poderia ser menor caso o procedimento fosse mais simples, reduzindo consequentemente complicações, como por exemplo, crises severas de hipoglicemia, com desmaios e confusão mental (MELO, 2014).

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Este trabalho apresenta como objetivo principal o estudo, desenvolvimento e avaliação de um sistema de reconhecimento óptico de caracteres, utilizando *smartphone* e computador pessoal, para leitura de glicosímetro convencional. Pretende-se possibilitar estudos e aplicações futuras para o problema proposto com possibilidade futura de desenvolvimento de um aplicativo em linguagem de programação Java para processamento das imagens diretamente no *smartphone*, e como resultado, proporcionar uma maior independência do deficiente visual diabético.

1.3.2 Objetivos Específicos

- Pesquisa e seleção de aplicativo comercial para *smartphone* com captura de imagem e possibilidade de transferência de dados via rede sem fio;
- Estudo sobre transferência de imagens de aparelhos *smartphone* para computador via rede sem fio;
- Escolha do *software* do PC para comunicação com *smartphone* e processamento de imagem aplicado a reconhecimento de caracteres;
- Desenvolvimento de rotinas de comunicação com *smartphone* e processamento de imagem com reconhecimento de caracteres;
- Estudo e avaliação sobre o posicionamento do *smartphone* para captura da taxa apresentada no *display* do glicosímetro;
- Desenvolvimento de sistema mecânico de fixação e posicionamento do *smartphone* e glicosímetro para realização das leituras;
- Testes de captura de tela e reconhecimento da taxa de glicose com sinalização sonora;
- Interpretação dos resultados;
- Análise da viabilidade da utilização do sistema estudado para os fins em questão;
- Estudo de sugestões de trabalhos e aplicações futuras.

1.4 JUSTIFICATIVA

A retinopatia diabética é o fator de risco mais importante a ser levado em conta em portadores de diabetes de longa duração. Praticamente todos os portadores da doença por mais de vinte anos desenvolvem algum grau de retinopatia. Segundo Hirschi & Polak (1997), 12% dos novos cegos americanos são provenientes desta enfermidade. No Brasil essa estatística é deficiente, pois as taxas variam entre 1,42 a 9,77% (BARBOSA & DUARTE, 2010). Considerando os dados mais consistentes, 12% dos novos deficientes visuais anuais americanos, pode-se prever que destinando conhecimentos de engenharia com a finalidade proposta neste trabalho, possibilitará um grande ganho qualitativo de vida para o público alvo, uma vez que,

quando se fala da possibilidade da automedicação, o paciente passa a ser independente nesse ponto, dispensando a necessidade do auxílio de outra pessoa o medicando diariamente.

1.5 PROCEDIMENTOS METODOLÓGICOS

O estudo realizado neste trabalho tem por objetivo a avaliação de um sistema de reconhecimento óptico de caractere, utilizando *smartphone* e computador, para leitura de glicosímetro convencional para possibilitar automedicação de insulina em diabéticos portadores de deficiência visual. Para isso, o projeto foi dividido em três etapas. A primeira etapa é voltada a fundamentação teórica que sustenta esta pesquisa, envolvendo desde os estudos sobre a diabetes até as tecnologias aplicadas nesse trabalho. A segunda etapa consiste em detalhar materiais e métodos utilizados e, por fim, foi realizado uma avaliação dos resultados obtidos e as conclusões dos mesmos.

1.6 ESTRUTURA DO TRABALHO

Esse trabalho esta estruturado da seguinte forma:

Capítulo 1 – Este capítulo introduz o tema ao leitor e trata de seus problemas e premissas, assim como apresenta seu objetivo geral e específico juntamente com suas justificativas.
Capítulo 2 – Este capítulo aborda toda a fundamentação teórica que serve de base para sustentar o estudo feito e engloba desde um estudo geral sobre a diabetes e seus problemas quanto à visão até as tecnologias e *softwares* utilizados no estudo.
Capítulo 3 – Este capítulo apresenta os materiais utilizados e descreve os métodos aplicados para a execução do estudo e de seus testes.
Capítulo 4 – Este capítulo apresenta os dados, tabelas e gráficos referentes aos resultados obtidos no estudo.

Capítulo 5 – Este capítulo apresenta as conclusões e discuções levantadas pelo estudo, assim como sugestões de aplicações e trabalhos futuros.

2 FUNDAMENTOS TEÓRICOS

Esta seção tem como objetivo informar e contextualizar o leitor, fornecendo fundamentações teóricas que foram relevantes durante as escolhas e execução do projeto.

2.1 DIABETES MELLITUS

O Diabetes *Mellitus* (DM) engloba um grupo heterogêneo de distúrbios crônicos do metabolismo, causados pela deficiência absoluta ou relativa de insulina. Caracteriza-se por hiperglicemia (excesso de glicose no sangue) nos períodos pós-prandial (uma ou duas horas após ingestão calórica) e/ou de jejum. O diabetes, quando presente por longos períodos, apresenta complicações representadas por desenvolvimento de doença dos pequenos vasos (microangiopatia), envolvendo particularmente retina e glomérulo renal, além de neuropatia e aterosclerose acelerada (PIMENTA, 2003).

2.2 EPIDEMIOLOGIA DO DIABETES

Os números do diabetes mostram que trata-se de um dos maiores problemas de saúde presente em todo o mundo, e que apresenta trajetória crescente. Em 1995, estimava-se que 4,0% da população adulta mundial era portadora, em 2025, esse número girará em torno de 5,4%. No Brasil, no final da década de 1980, 8% da população residente em áreas metropolitanas, entre 30 a 69 anos de idade, possuía a doença. Esse percentual variava de 3% a 17% nas faixas de 30 a 39 e 60 a 69 anos, respectivamente. Além do que, estimativas atuais dão conta de que por volta de 11% da população igual ou superior a 40 anos, o que representa cerca de 5 milhões e meio de pessoas, são portadores da doença (população estimada IBGE 2005) (MINISTÉRIO DA SAÚDE, 2006). Vale dizer também que o diabetes apresenta alta taxa de mortalidade, além de perda significativa na qualidade de vida. É uma

das principais causas de mortalidade, insuficiência renal, amputação de membros inferiores, cegueira e doença cardiovascular.

Um levantamento da Organização Mundial da Saúde (OMS) de 1997 revelou que, após 15 anos de doença, 2% dos portadores estarão cegos e outros 10% terão deficiência visual grave. Este mesmo estudo estima que, no mesmo período de doença, 30 a 45% terão algum grau de retinopatia, 10 a 20%, de nefropatia, 20 a 35%, de neuropatia e 10 a 25% terão desenvolvido doença cardiovascular (MINISTÉRIO DA SAÚDE, 2006).

2.3 CLASSIFICAÇÃO DO DIABETES

Basicamente encontra-se duas maneiras de classificar o diabetes. Uma delas segundo os tipos de diabetes (etiológica), definidos de acordo com defeitos ou processos específicos, e a outra que leva em conta os estágios de desenvolvimento da doença, compreendidos como estágios pré-clínicos e clínicos, este último incluindo estágios avançados em que a insulina é necessária para controle ou sobrevivência (MINISTÉRIO DA SAÚDE, 2006). Entretanto, para o trabalho em questão somente o primeiro tipo classificatório será estudado e avaliado com maiores detalhes.

2.3.1 Tipos de diabetes - classificação etiológica

O diabetes tipo 1 e tipo 2 são os mais frequentes – o primeiro compreendendo cerca de 10% do total de casos, enquanto o segundo cerca de 90% do total de casos. Outro tipo de diabetes encontrado com certa frequência e cuja etiologia ainda não está esclarecida é o diabetes gestacional, que, em geral, é um estágio pré-clínico de diabetes, detectado no rastreamento pré-natal. Outros tipos específicos de diabetes menos frequentes podem resultar de defeitos genéticos da função das células beta, defeitos genéticos da ação da insulina, doenças do pâncreas exócrino, endocrinopatias, efeito colateral de medicamentos, infecções e outras síndromes genéticas associadas ao diabetes (FRAGAS; SOARES; BRONSTEIN, 2008).

a) Diabetes tipo 1

Uma publicação do Ministério da Saúde define de forma resumida que o termo tipo 1 indica destruição da célula beta (responsáveis por sintetizar e secretar o hormônio insulina) que eventualmente leva ao estágio de deficiência absoluta de insulina, quando a administração de insulina é necessária para prevenir cetoacidose (sangue fica ácido devido a falta de insulina), coma e morte. A destruição das células beta é geralmente causada por processo autoimune, que pode ser detectado por auto-anticorpos circulantes como anti-descarboxilase do ácido glutâmico, anti-ilhotas e anti-insulina, e, algumas vezes, está associado a outras doenças auto-imunes como a tireoidite de Hashimoto, a doença de Addison e a miastenia gravis. Em menor proporção, a causa da destruição das células beta é desconhecida (tipo 1 idiopático). O desenvolvimento do diabetes tipo 1 pode ocorrer de forma rapidamente progressiva, principalmente, em crianças e adolescentes (pico de incidência entre 10 e 14 anos), ou de forma lentamente progressiva, geralmente em adultos, (LADA, *latent autoimmune diabetes in adults*; doença auto-imune latente em adultos) (MINISTÉRIO DA SAÚDE, 2006).

b) Diabetes tipo 2

Já o termo tipo 2, segundo a mesma publicação, é usado para designar uma deficiência relativa de insulina. A administração de insulina nesses casos, quando efetuada, não visa evitar cetoacidose, mas alcançar controle do quadro hiperglicêmico. A cetoacidose é rara e, quando presente, é acompanhada de infecção ou estresse muito grave. A maioria dos casos apresenta excesso de peso ou deposição central de gordura. Em geral, mostram evidências de resistência à ação da insulina e o defeito na secreção de insulina manifesta-se pela incapacidade de compensar essa resistência. Em alguns indivíduos, no entanto, a ação da insulina é normal, e o defeito secretor mais intenso (MINISTÉRIO DA SAÚDE, 2006).

2.4 RETINOPATIA DIABÉTICA – RD

No ano de 2001, foi estimado que a retinopatia era a 3ª causa de cegueira em adultos no Brasil, sendo a principal em pessoas em idade produtiva (16 a 64 anos). Esta é uma das complicações mais comuns da diabetes *mellitus*, encontrada, após 20 anos de doença, em mais de 90% dos casos do tipo 1 e de 50% a 80% do tipo 2, sendo que com o crescimento da expectativa de vida dos pacientes o número de casos tem aumentado. Além da retina, as alterações vasculares costumam progredir de modo semelhante em outros órgãos como rim, coração e cérebro. Dessa maneira, a retinopatia tem correlação direta com a sobrevida desses pacientes, sendo que a detecção precoce e tratamento adequado reduzem consideravelmente os casos de cegueira (GROOS et al., 2001).

2.4.1 Fatores de risco

Pode-se dividir em genéticos e não genéticos. Os componentes hereditários ainda não foram completamente elucidados, mas sabe-se que é fundamental para o curso clínico da doença. Dentre os fatores não genéticos, o tempo de doença (acima de 5 anos) é o principal fator determinante para o desenvolvimento da RD. Por outro lado, níveis pressóricos elevados, controle glicêmico inadequado e gravidez estão mais relacionados com sua progressão (ARAGÃO; FERREIRA; PINTO, 2013).

Os principais fatores de risco não genéticos para retinopatia diabética são:

- Tempo de doença
- Glicemia mal controlada
- Hipertensão arterial
- Idade do diagnóstico
- Puberdade
- Gestação
- Nefropatia diabética (doença renal progressiva)
- Sexo masculino
- Tabagismo
- Dislipidemia

- Obesidade

2.4.2 Fisiopatologia

As lesões da RD ocorrem em progressão cronológica, exceto pelo edema macular. A hiperglicemia crônica desvia o metabolismo da glicose para vias alternativas, formando fatores inflamatórios, trombogênicos e vasoconstrictores, além de aumentar a suscetibilidade ao estresse oxidativo, resultando em oclusão e fragilidade vascular com perda de pericitos. Esse processo de enfraquecimento dos capilares causa a quebra da barreira hemato-retiniana, o que possibilita formação de microaneurismas (achados mais precoces da RD) e extravasamento de plasma para o interstício, resultando em hemorragias e edema. As principais complicações da RD são aquelas que levam à deficiência visual aguda: hemorragias, descolamento da retina e rubeose de íris (ARAGÃO; FERREIRA; PINTO, 2013).

a) Hemorragias

Vasos neoformados são mais frágeis e, portanto, mais suscetíveis a sangramentos, a maioria ocorrendo durante o sono. Quando surgem entre a retina e o vítreo, podem rapidamente causar descolamento tracional da retina (ARAGÃO; FERREIRA; PINTO, 2013).

b) Descolamento tracional da retina

Acomete cerca de 5 a 10% dos diabéticos. O tecido fibrovascular tende a crescer em direção a locais com menor resistência, como a face posterior do vítreo. Desse modo, trações vítreas podem ser transmitidas à retina, causando descolamento. A retina apresenta-se com superfície esticada, brilhante, sem deslocamento de fluido subretiniano. Se houver proliferação fibrovascular na superfície do nervo óptico, ele também pode ser tracionado, causando baixa visual (ARAGÃO; FERREIRA; PINTO, 2013).

c) Rubeose da íris

É a proliferação anterior de neovasos, alcançando a íris. Ocorre na retinopatia diabética proliferativa, sendo mais comum em paciente com isquemia severa ou descolamento de retina persistente à vitrectomia via pars plana. Pode causar glaucoma neovascular (ARAGÃO; FERREIRA; PINTO, 2013).

2.5 GLICOSÍMETRO

O glicosímetro é um equipamento portátil utilizado para medir a concentração de glicose no sangue, auxiliando a monitorar a glicose sanguínea de forma simplificada e rápida. O aparelho verifica o valor da glicemia através de uma gota de sangue, retirada geralmente do dedo da mão, e fornece o resultado quase que instantaneamente. A maioria desses aparelhos é projetada para amostras de sangue capilar e, quando usados adequadamente, apresentam resultados tão bons como os testes laboratoriais padrões (ACEDO, 2014).

Entretanto, conforme reportagem publicada no site G1, edição do dia 16 de novembro de 2014, muitos dos *kits* de glicosímetros comercializados no Brasil não apresentam manuais de instrução adequados às normas nacionais, com informações claras e de fácil entendimento. A reportagem destaca que nos teste realizados pelo Inmetro e pela Anvisa, com 75 voluntários, todas as marcas avaliadas foram reprovadas com respeito aos seguintes quesitos: manuais de instrução e os acessórios dos glicosímetros. O resultado completo dos testes pode ser encontrado em: http://estatico.redeglobo.globo.com/2014/11/16/RelatorioFinalManualGlicosimetro.pdf. Neste teste foram analisadas 15 marcas: Accu check active; Accu check performa; Biocheck gold; Breeze 2; Contour TS; Fácil trueread; Freestyle lite; G tech free; Injex sens 2; On call plus; One touch select simple; One touch ultra; One touch ultra mini; Optimum xceed; e Testline.

Esta constatação é extremamente importante e corrobora que devem ser realizados mais estudos que possibilitem a adequação e o fácil manuseio dos glicosímetros, em especial para portadores de deficiência visual.

2.5.1 Como usar o glicosímetro

A taxa de açúcar no sangue, chamada de glicemia, é considerada normal até 99 miligramas por decilitro de sangue. Se esse nível estiver mais alto ou muito mais baixo do que esse patamar, pode ser sinal de problemas de saúde, como o diabetes, por exemplo (G1, 2014).

Apesar de aparentemente fáceis de usar, é importante a leitura adequada do manual que acompanha os *kits* que trazem instruções de uso, indicações da quantidade ideal de glicose no sangue, bem como os valores que representam a hipoglicemia e a hiperglicemia. Para o presente trabalho foi adotado glicosímetro Contour TS Kit, apresentado na Figura 2 (o *kit* será descrito com maiores detalhes na seção de materiais e métodos).

Figura 2 – Glicosímetro Contour TS Kit.
Fonte: Adaptado de www.mamaosemacucar.com (2014).

Para medir a glicose no sangue, o indivíduo deve realizar um pequeno furo no dedo da mão, extrair uma gota de sangue, depositá-la numa fita apropriada e introduzi-la no aparelho. Passados alguns segundo, o aparelho exibirá em sua tela a medição referente a quantidade de glicose presente no sangue do indivíduo naquele momento (GLICOSÍMETRO, 2013). Na Figura 3 é apresentado o manual de instruções com os procedimentos de medição com o glicosímetro Contour TS.

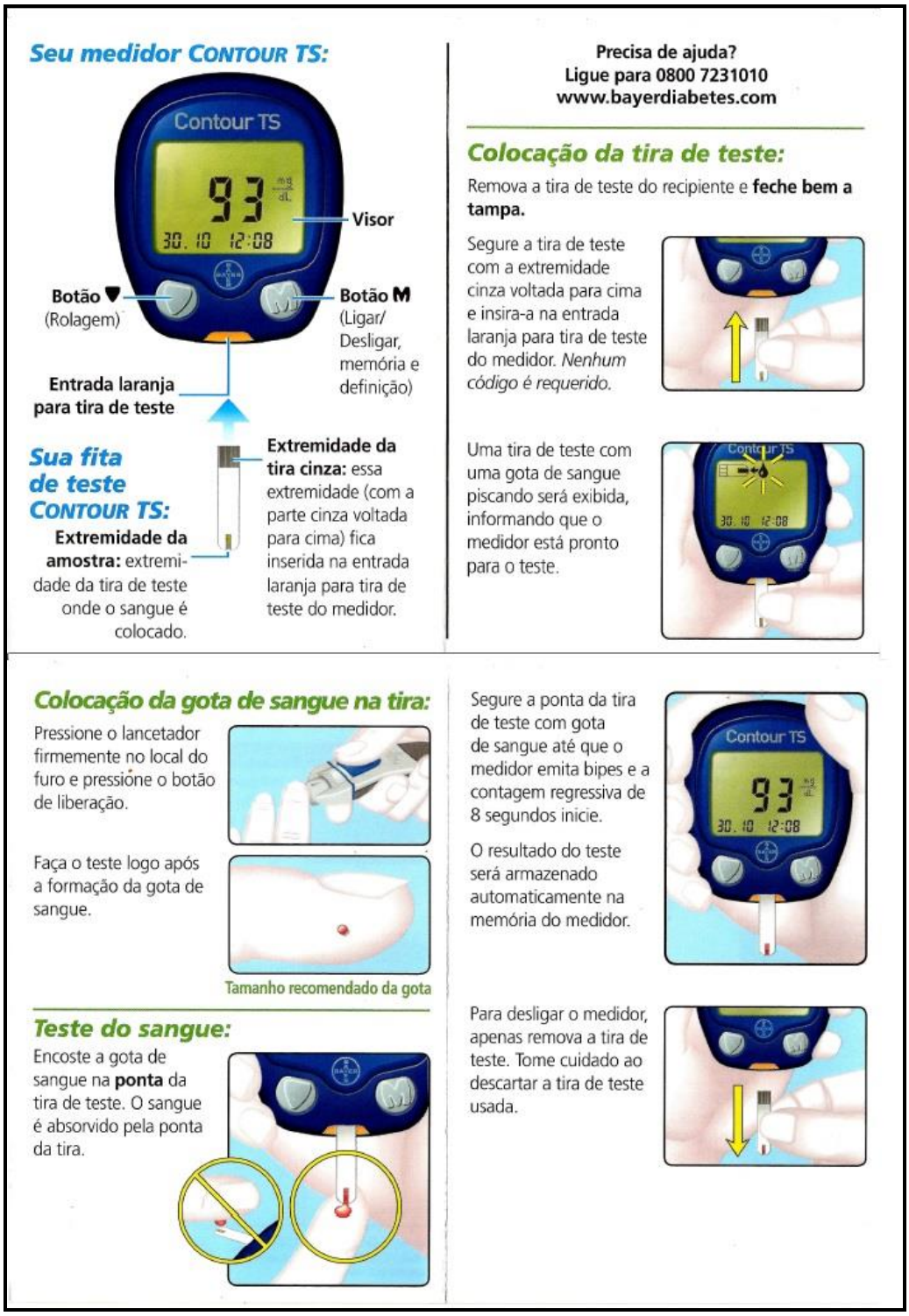


Figura 3 – Instruções de utilização do glicosímetro Contour TS.

Fonte: Adaptado de Guia de referência rápida Contour TS.

2.5.2 Quando usar o glicosímetro

O glicosímetro pode ser utilizado várias vezes ao dia. A quantidade necessária irá variar conforme a alimentação e o tipo de diabetes que o paciente possui. Indivíduos pré-diabéticos e diabéticos tipo 2 podem medir a glicose 1 ou 2 vezes ao dia. Já os insulinos dependentes podem necessitar usar o glicosímetro até 7 vezes ao dia (GLICOSÍMETRO, 2013).

2.5.3 Coleta Sanguínea

Para que o deficiente visual diabético tenha de fato sua autonomia quanto à automedição da glicose sanguínea, seria preciso um mecanismo facilitador para a coleta da gota de sangue utilizada no processo. Porém, a maioria das tiras teste, inclusive a tira Contour, possuem o orifício de absorção localizado na ponta da tira, o que torna sua localização intuitiva. Além disso, o aparelho possui uma configuração que emite bipes em algumas situações, sendo uma delas quando a fita absorve corretamente o sangue, o que serve de aviso para o usuário, uma vez que, trata-se de um sinal sonoro. Sendo assim, o trabalho proposto será simplificado, já que, não será preciso fazer uma adaptação para essa parte do processo. A Figura 4 ilustra o processo da coleta sanguínea.

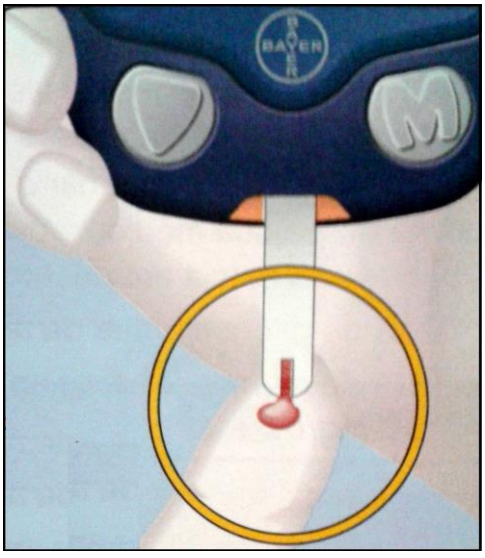


Figura 4 – Sangue sendo absorvido pela tira reativa.

Fonte: Adaptado de Guia de referência rápida Contour TS.

2.6 TECNOLOGIAS DE COMUNICAÇÃO

Para realizar a transferência da imagem capturada pelo *smartphone* para o PC foram levados em conta as duas tecnologias mais utilizadas para transmissão de dados nos dias atuais, sendo elas: Wi-Fi e Bluethooth.

2.6.1 Wi-Fi

O termo Wi-Fi trata de uma marca registrada pela Wi-Fi Alliance. Apesar disso, a expressão hoje se tornou um sinônimo para a tecnologia IEEE 802.11 – especificada pelo IEEE (*Institute of Electrical and Electronics Engineers*) –, que permite a conexão entre diversos dispositivos sem fio. Amplamente utilizado na atualidade, a origem do termo, diferente do que muito acreditam, não tem um significado específico (LANDIM, 2012). A expressão Wi-Fi surgiu como uma alusão à expressão *High Fidelity* (Hi-Fi), utilizada pela indústria fonográfica na década de 50. Assim, o termo Wi-Fi nada mais é do que a contração das palavras *Wireless Fidelity* (LANDIM, 2012).

As redes Wi-Fi funcionam por meio de ondas de rádio. Estas são transmitidas através de um adaptador, o chamado “roteador”, que recebe os sinais, decodifica e os emite a partir de uma antena. Para que um computador ou dispositivo tenha acesso a esses sinais, é preciso que ele esteja dentro um determinado raio de ação, conhecido como *hotspot* e esteja equipado para receber tal sinal (LANDIM, 2012).

O padrão 802.11 estabelece normas para criação e para uso de redes sem fio. Como a transmissão nesse tipo de rede é feita por rádio frequência, que se propaga pelo ar, podem existir inúmeros serviços utilizando esse tipo de sinal. Por isso, é necessário que cada um opere de acordo com as exigências estabelecidas pelo governo de cada país. Dessa forma, evita-se problemas principalmente interferências (ALECRIM, 2008a).

Na Tabela 1 são apresentados alguns dos prinicipais padrões da tecnologia IEEE 802.11, com as respectivas faixas de frequência, compatibilidade e distância.

Tabela 1 – Principais padrões da família IEEE 802.11.

Padrão	Faixa de Frequência	Compatibilidade	Máxima taxa de dados	Distância
802.11a	5 GHz	-	54 Mbps	Ambiente fechado: 30 a 90 m
802.11b	2,4 GHz	802.11g	11 Mbps	Ambiente aberto: 100 a 300 m (dependente do ambiente)
802.11g	2,4 GHz	802.11b	54 Mbps	

Fonte: Wi-Fi Alliance (2015).

As informações completas sobre o protocolo IEEE 802.11 para redes sem fio LAN (*Local Area Network*) pode ser encontradas no *site* da associação IEEE: http://standards.ieee.org/about/get/802/802.11.html.

2.6.2 Bluetooth

O Bluetooth é um padrão global de comunicação sem fio e de baixo consumo de energia que possibilita a troca de dados entre dispositivos próximos. A transmissão é feita por radio frequência, e permite que os dispositivos se detectem apenas estando dentro do raio de funcionamento (ALECRIM, 2008b).

Desenvolvida pela empresa de telecomunicações Ericsson, em 1994, a tecnologia foi batizada de Bluetooth em homenagem a um antigo rei da Dinamarca e da Noruega, *Harold Blatand*, mais conhecido como Harold Bluetooth (Harold Dente-Azul). O nome foi utilizado pela sua façanha de ter unificado as tribos norueguesas, suecas e dinamarquesas, já que a tecnologia é justamente uma forma de unificação de diferentes dispositivos. O logotipo e símbolo do Bluetooth também é baseado no nome de *Harold*, já que é formado pela união das runas nórdicas *Hagall* e *Berkanan*, correspondentes às iniciais do nome do rei, “H” e “B”, respectivamente (ALECRIM, 2008b).

O sistema utiliza uma frequência de rádio de onda curta (2.4 GHz) para criar uma comunicação entre aparelhos habilitados. Como seu alcance é curto e só permite a comunicação entre dispositivos próximos, seu consumo de energia é bem

baixo. A comunicação do Bluetooth se dá através de uma rede chamada *piconet*, que só permite a conexão de até oito dispositivos. Porém, para aumentar essa quantidade, é possível sobrepor mais *piconets*, capacitando o aumento de conexões pelo método chamado de *scatternet*.

Embora já existam classes de Bluetooth com alcance de 100 metros, a maioria dos dispositivos conta com alcance de 1 a 10 metros, o que, apesar de ser uma desvantagem, ajuda na segurança dos usuários. Outra garantia de ambiente seguro é que, antes de efetuar trocas de dados e arquivos entre aparelhos que dispõem do Bluetooth, normalmente deve-se ativar a função através das configurações dos dispositivos (CÂMARA, 2012).

2.6.3 Wi-Fi x Bluetooth

Depois de feita uma breve pesquisa a respeito das vantagens e desvantagens de ambas as tecnologias quanto a aplicação ao estudo em questão, constatou-se que a melhor e mais eficiente tecnologia seria a WI-FI. São várias as razões para tal decisão: primeiramente seu alcance em geral é maior, a maioria das casas possuem computadores ligados a roteadores de Wi-Fi, enquanto nem todos possuem tecnologia Bluetooth e a conexão ocorre automaticamente uma vez que conectado ao roteador. Já no Bluetooth são necessárias permissões a cada nova conexão. Além disso, tratando-se de alguns fabricantes de celular, o sistema operacional só permite troca de arquivos entre dispositivos compatíveis da mesma marca ou sistema operacional.

Outras características também podem ser elencadas, como por exemplo, consumo do s*martphone* e aplicativos para facilitar o desenvolvimento do trabalho. O Wi-Fi consome menos energia da bateria do s*martphone*, além de existirem muitos aplicativos do tipo “câmera IP”, que servem para pessoas monitorarem câmeras de segurança de suas casas via IP (*Internet Protocol*), e em alguns casos, no sentido oposto, é possível monitorar o que a câmera do celular transmite por IP.

2.7 PROCESSAMENTO DE DADOS

Uma vez transmitida ao PC, a imagem passará pela etapa de processamento. A programação para tal propósito foi totalmente desenvolvida em MATLAB versão 2014a, *software* amplamente utilizado ao longo do curso de graduação na UTFPR.

2.7.1 MATLAB

O MATLAB (*Matrix Laboratory*) é um *software* interativo de alto desempenho destinado a fazer cálculos numéricos com matrizes. O *software* dispõe de um amplo conjunto de programas de apoio especializados, denominados "*Toolboxes*" que estendem significativamente o número de funções incorporadas no programa principal. Estas *Toolboxes* cobrem praticamente todas as áreas principais no mundo da engenharia destacando entre elas a *toolbox* de processamento de imagem, sinais, controle robusto, estatística, analise financeira, cálculo matemático simbólico, redes neuronais, lógica difusa, identificação de sistemas e simulação de sistemas dinâmicos.

2.7.1.1 *Toolbox* de Processamento de Imagem

O *Image Processing Toolbox* é um conjunto de programas de apoio especializados que suportam um variado leque de operações para processamento de imagem tais como:

- Transformações espaciais;
- Operações Morfológicas;
- Operações em bloco ou pontuais;
- Filtragem de imagem linear e desenvolvimento de filtros;
- Transformadas de imagem;
- Registro de imagem;
- Segmentação.

Como operações básicas esta *toolbox* permite ler e visualizar uma imagem, verificar informações, melhorar o contraste e gravar imagens em disco. Esta *toolbox* define 4 tipos de imagem, binárias, indexadas, em tons de cinza e imagens RGB (*Red-Green-Blue*) (FARIA, 2010). Em uma imagem binária cada elemento de imagem, também denominado *pixel*, assume um valor discreto, sendo zero ou um, como mostrado na Figura 5.

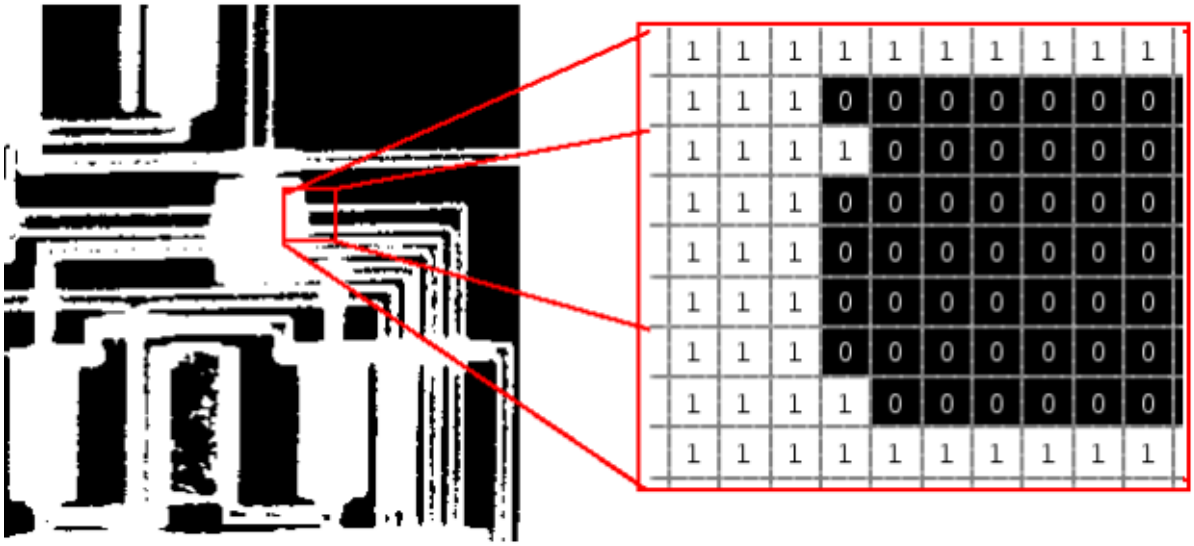


Figura 5 – Exemplo de uma imagem binária e seus valores de pixéls
Fonte: Adaptado de Faria (2010).

Uma imagem indexada consiste em uma matriz de imagem e uma escala de cores. Os *pixels* têm valores de índice referentes a uma escala de cores. Uma imagem em tons de cinza é constituída por uma matriz de dados cujos valores representam intensidades de um certo intervalo. As matrizes podem ser *uint8*, *uint16*, *int16*, *single* ou *double*. Para imagens em *single* ou *double*, os valores de intensidade dos pixéls podem tomar valores entre [0,1]. Para *uint8* os valores variam entre [0,255], para *uint16* entre [0,65535] e para *int16* os valores variam entre [-32768,32767].

Já a imagem RGB, cada pixel é especificado por 3 valores (matriz cubica 3xMxN). Um valor para a componente vermelho, um para a componente verde e um outro para a componente azul. A cor de cada pixel é assim determinada pela combinação das intensidades de vermelho, verde e azul armazenada em cada plano de cor de cada pixel (FARIA, 2010).

2.8 RECONHECIMENTO ÓPTICO DE CARACTER

O OCR se refere ao termo em inglês *Optical Character Recognition* (Reconhecimento Óptico de Caractere) e é uma técnica empregada para reconhecimento de caracteres a partir de um arquivo de imagem ou mapa de *bits*, sejam eles escaneados, escritos a mão, datilografados ou impressos. Dessa forma, através do OCR é possível obter um arquivo de texto editável por um computador.

2.8.1 OCR utilizando MATLAB

Todas as etapas envolvidas no OCR, segmentação, extração de características e classificação, pode ser implementado usando MATLAB. A segmentação, que envolve a verificação da conectividade de formas, e rotulagem para então fazer o isolamento das regiões, é a fase mais importante , pois permite que o programa extraia características de cada simbolo individualmente. A segmentação de texto escrita à mão é particularmente muito difícil, pois caracteres manuscritos tendem a ser ligados um ao outro (DUNNING, 2012).

2.8.1.1 Captura de Imagem

De acordo com um documento apresentado em um seminário dado por Lasin Laboratório de *Synergetics* na Universidade de Ljubljana, o OCR pode ser implementado em MATLAB usando recursos do *Toolbox* de Rede Neural e o de Processamento de Imagem. A primeira etapa envolve a leitura da imagem no espaço de trabalho do MATLAB como um arquivo de *bitmap*. Este é um tipo de ficheiro gráfico, no qual cada *pixel*, corresponde a um ou mais dígitos binários, ou *bit*, na memória (DUNNING, 2012). O código apresentado no Quadro 1 é usado para essa função.

```
Imagem=imread('training.bmp');
imshow(Imagem);
```

Quadro 1 – Código para ler imagem
Fonte: Autoria Própria.

2.8.1.2 Conversão da Imagem

A próxima etapa é converter a imagem colorida, que é armazenado como sobreposições vermelhos, verdes e azuis separadas, em uma imagem em tons discretos de cinza. No Quadro 2 é apresentado o código do MATLAB para converter uma imagem RGB em uma imagem em tons de cinza.

```
ImagemCinza = rgb2gray (Imagem);
imshow(ImagemCinza);
```

Quadro 2 – Código para conversão em escala de cinza.
Fonte: Autoria Própria.

Posteriormente, uma técnica conhecida como "debulha" é usada para converter a imagem de tons de cinza para uma imagem binária. Os códigos do MATLAB necessários para converter a imagem em tons de cinza em uma imagem binária são apresentados no Quadro 3:

```
ImagemBinaria = im2bw (ImagemCinza, graythresh(ImagemCinza));
imshow(ImagemBinaria);
```

Quadro 3 – Código para conversão em binário
Fonte: Autoria Própria.

2.8.1.3 Detecção de Bordas

Uma vez que a imagem binária é criada, caracteres individuais são cortadas em sub-imagens. Estes fornecem os dados brutos para a rotina recurso de extração. As sub-imagens devem ser cortadas na borda de cada caracter, se eles forem de um tamanho padrão, deste modo, a detecção da borda de cada personagem é de extrema importância (DUNNING, 2012). As bordas de uma imagem podem ser detectadas usando o código MATLAB apresentado no Quadro 4.

```
Saliencia = edge(uint8(ImagemBinaria));
imshow(Saliencia);
```

Quadro 4 – Código para detecção de bordas.
Fonte: Autoria Própria.

3 MATERIAIS E MÉTODOS

Neste capítulo são apresentados os materiais utilizados e a metodologia aplicada neste trabalho, bem como as demais informações importantes para o entendimento do mesmo.

3.1 COMPUTADOR PARA DESENVOLVIMENTO

O próprio computador pessoal do proponente foi utilizado durante o desenvolvimento deste Trabalho de Conclusão de Curso (TCC): um Ultrabook Dell Inspiron modelo 14z-5423. O sistema operacional do *notebook* é o Windows 8.1 64 *bits* e suas configurações principais são: 4 Gb de memória RAM, 464 Gb de memória utilizável em seu HD e processador com base em 64x Intel Core i5. Entretanto, qualquer máquina conectada a um roteador (sendo por cabo *ethernet* ou via Wi-Fi) e capaz de rodar o *software* MATLAB poderia ser utilizado para essa aplicação.

3.2 GLICOSÍMETRO ADOTADO NO TRABALHO

Para o presente trabalho foi adotado o Glicosímetro Contour Ts Kit (Grupo Bayer), apresentado na Figura 6. O principal motivo para tal escolha é que o aparelho em questão não necessita de *chip* de codificação para corrigir erros de medição. Tal função já vem acoplada na própria fita de análise sanguínea do glicosimetro, ou seja, o aparelho simplifica o trabalho do paciente diabético, facilitando a extração dos sinais que serão utilizados.

Outro fator considerável é o preço do *kit*, sendo bem mais em conta que os modelos tradicionais, bem como a sua alta disponibilidade no mercado nacional. Um *kit*, vendido separadamente das tiras reativas, foi comprado ao custou de R$19,99 (referência em Fevereiro de 2015) e contém o medidor Contour TS, 10 lancetas e o furador. O detalhe da tela frontal do equipamento é mostrado na Figura 7.

Figura 6 – Glicosímetro Contour TS Kit.
Fonte: Autoria Própria.

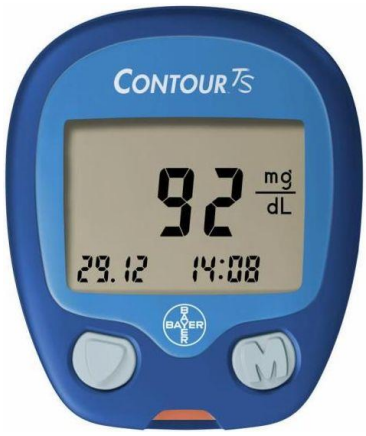


Figura 7 – Tela frontal do glicosímetro Contour TS
Fonte: Adaptado de www.polymap.net.

3.3 *SMARTPHONE*

O próprio *smartphone* do responsável pelo desenvolvimento deste TCC foi utilizado para a execução das tarefas propostas. Trata-se de um aparelho iPhone 4s produzido pela empresa Apple com o sistema operacional iOS versão 8 e câmera de 8 M*pixel*. Entretanto, qualquer outro modelo de *smartphone* certificado com Wi-Fi

802.11 e câmera acima de 5 M*pixel* pode ser utilizado, incluindo dispositivos com o sistema operacional Android.

3.4 SISTEMA DE CAPTURA DE IMAGEM NO *SMARTPHONE*

Após a pesquisa sobre métodos e aplicativos para captura e transferência de imagens para o PC foi escolhido a utilização de um aplicativo de câmera com endereço de IP, também chamado de IP *Cam*. A escolha desta tecnologia também teve impacto sobre o método de transferência de dados entre o *smartphone* e o PC. A aplicação requer por si só de um dispositivo Wi-Fi para funcionar e cria uma conexão com endereço IP para acesso da imagem capturada via computador, onde serão realizadas as etapas de processamento.

Especificamente, para a captura da imagem do *display* do glicosímetro e abertura de canal de comunicação entre o *smartphone* e o PC, foi utilizado o aplicativo IP Cam da Senstic, com logotipo apresentado na Figura 8. Trata-se de um aplicativo (*app*) pago, no valor de U$ 2,00 na loja virtual da Apple – App Store. Apesar de existirem aplicativos semelhantes e gratuitos, principalmente para Android, o *software* foi escolhido devido aos recursos avançados, qualidade de imagem e operação no sistema operacional iOS.

O IP *Cam* transforma o *smartphone* em uma câmera de vigilância, inclusive com a possibilidade de captura de áudio. Uma vez iniciado no celular é possível, por rede Wi-Fi, acessar através de um navegador vídeo e áudio ao vivo em qualquer computador ou dispositivo de sistema operacional compatível. O aplicativo também suporta MJPEG *streaming* tradicional, possibilitando acesso de imagens de qualquer outro aplicativo visualizador de câmera IP.

Figura 8 – Ícone IP Cam.
Fonte: Adaptado do site da Apple Store.

3.5 ESTRUTURA FÍSICIA DO PROTÓTIPO DE BANCADA

Durante o desenvolvimento da estrutura mecânica do protótipo para posicionamento do glicosímetro e do *smartphone* foi definida uma metodologia para facilitar a aquisição de dados e as etapas de processamento de imagem. Nesta, após feita a medição de glicose, o celular (com o aplicativo IP *Cam* iniciado) realiza a captura e transferência dos dados via rede sem fio Wi-Fi para processamento de imagem em PC utilizando o *software* MATLAB. Durante esta etapa ocorre a localização da região onde se encontram os caracteres na imagem gerada e o reconhecimento dos mesmos com retorno via áudio, bem como eventuais erros indicados pelo glicosímetro. Dessa forma, é possível demonstrar a funcionalidade e eficiência das funções de processamento geradas, assim como a viabilidade de trabalhos futuros para adaptação do código para aplicativo de celular, onde seria dispensada a utilização de um computador para o processamento das informações.

A estrutura em questão trata de um protótipo simplificado de bancada desenvolvido com materiais de baixo custo (aproximadamente 30 reais ao todo). Para a montagem foram utilizados materiais simples como protetores de porta, pote de plástico, capa flexível para celular, parafusos e porcas, LEDs de alto brilho, resistor, chave liga/desliga e tubos de silicone, todos facilmente encontrados para venda. Vale lembrar que trata-se de um protótipo para testes de viabilidade e em eventual aplicação comercial seria necessário uma pesquisa mais aprofundada para desenvolvimento de uma estrutura mais adequada.

Com os protetores de porta e parafusos e porcas foi feito uma estrutura onde é encaixado o glicosímetro. Também com os protetores e com a capa de celular flexível foi montado um suporte para o *smartphone*. Este foi colado, tampando o fundo do recipiente de plástico (o fundo foi retirado), que por sua vez encaixa no suporte feito para o glicosímetro.

Para obter uma iluminação controlada e homogênea foi feito um circulo de LEDs, alocados em um tubo de silicone para disperçar eventuais erros devido a diferenças de brilho introduzidas pela luz ambiente, bem como, minimizando o reflexo causado na tela do glicosímetro. Uma fonte conversora estabilizada CA-CC foi utilizada para gerar a alimentação de 12 V do sistema.

Nas Figuras 9 e 10 são mostradas a parte interna do sistema com os LEDs apagados e acesos, respectivamente.

.

Figura 9 – Parte interna inferior do protótipo com LEDs apagados.
Fonte: Autoria Própria.

Figura 10 – Parte interna inferior do protótipo com LEDs acesos.
Fonte: Autoria Própria.

Na Figura 11 é apresentada a visão lateral do protótipo com o celular posicionado sobre a capa de proteção que fica presa ao suporte, além da indicação da chave liga/desliga e entrada de alimentação.

Figura 11 – Visão lateral do protótipo com glicosímetro e celular posicionados.
Fonte: Autoria Própria.

Na Figura 12 é apresentada a visão superior do protótipo com glicosímetro posicionado sempre na mesma disposição, chave liga/desliga e cabo de alimentação dos LEDs. Na Figura 13 é mostrada a visão lateral do protótipo com glicosímetro e capa de celular posicionados. Durante as aquisições o *smartphone* é posicionado na capa de celular com a câmera voltada para o *display*.

Figura 12 – Visão superior do protótipo com glicosímetro posicionado.
Fonte: Autoria Própria.

Figura 13 – Visão lateral do protótipo com glicosímetro e capa de celular posicionados.
Fonte: Autoria Própria.

3.6 ACESSÓRIOS

Tendo a estrutura pronta e apta para testes, pode-se tratar dos demais componentes do sistema utilizados: glicosímetro, *smartphone* e computador. O primeiro, como descrito anteriormente, é o glicosímetro Contour, e sua funcionalidade se enquadra de forma adequada para uma pessoa deficiente visual, já que não seria necessário nenhuma configuração durante o processo de medição, pois assim que inserido a tira teste no aparelho, mesmo que desligado, esse entra em modo de espera do sangue. Após isso, ouve-se um apito e após 8 segundos, ou instantaneamente em caso de erro de medição, é exibido o valor do teste (ou código de erro) na tela, e lá permanece por um tempo considerado. Caso a pessoa demore muito, este apita e desliga, bastando pressionar o botão da direita para retornar com o valor da última medida.

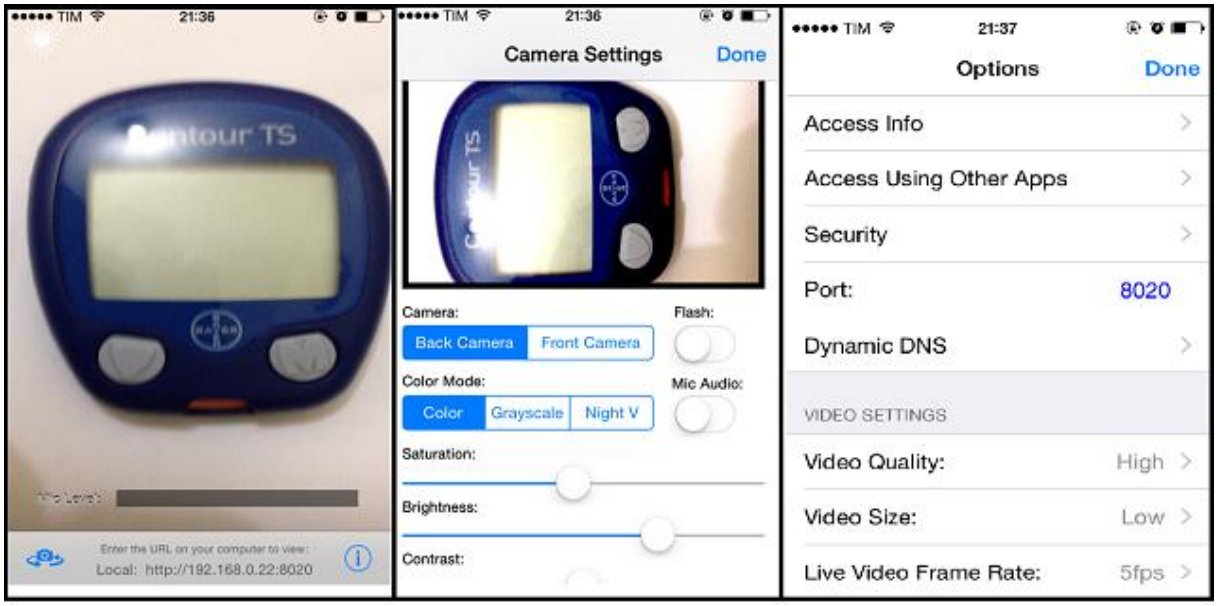
Quanto ao *smartphone* utilizado, trata-se de um celular *iphone* modelo 4s, e os motivos para sua escolha são apenas que o estudante autor deste trabalho já o possuia como celular pessoal e este possui capacidade de captura de imagem e transferência de dados via Wi-Fi, como requisitado para o processo. O mesmo serve

para o computador que deve apenas possuir requisitos para execução do *software* MATLAB e placa de rede com suporte Wi-Fi.

3.7 TRANSMISSÃO

A transmissão de dados ocorre por Wi-Fi através do aplicativo IP *Cam*. Basta abrir o aplicativo com o *smartphone* conectado a alguma rede Wi-Fi e no mesmo instante as imagens começam a ser transmitidas para qualquer dispositivo compatível que venha a acessar o endereço de IP correspondente.

Na Figura14 podem ser vistas as três telas principais do aplicativo para celular. São apresentadas a tela inicial, a tela de ajuste de parâmetros da imagem e a tela de configurações gerais nas Figura 14 (a), (b) e (c), respectivamente.



(a) **(b)** **(c)**

Figura 14 – Telas do aplicativo IP Cam. (a) Tela inicial. (b) Configuração da imagem. (c) Configurações gerais.

Fonte: Autoria Própria.

3.8 PROCESSAMENTO DE DADOS

Para o processamento de dados optou-se por utilizar o *software* MATLAB, pois trata-se de uma ferramenta poderosa e com recursos de grande utilidade para o estudo em questão, tal como o *ToolBox* de Processamento de Imagem. Vale lembrar também que o código de todas as funções do *ToolBox* é consultável, inclusive os algoritmos matemáricos utilizados pelas mesmas. Sendo assim, é extremamente viável a futura adaptação do código em MATLAB para outras linguagens, tais como C++, Java, entre outras.

O fluxograma principal de funcionamento da aplicação desenvolvida é apresentado na Figura 15.

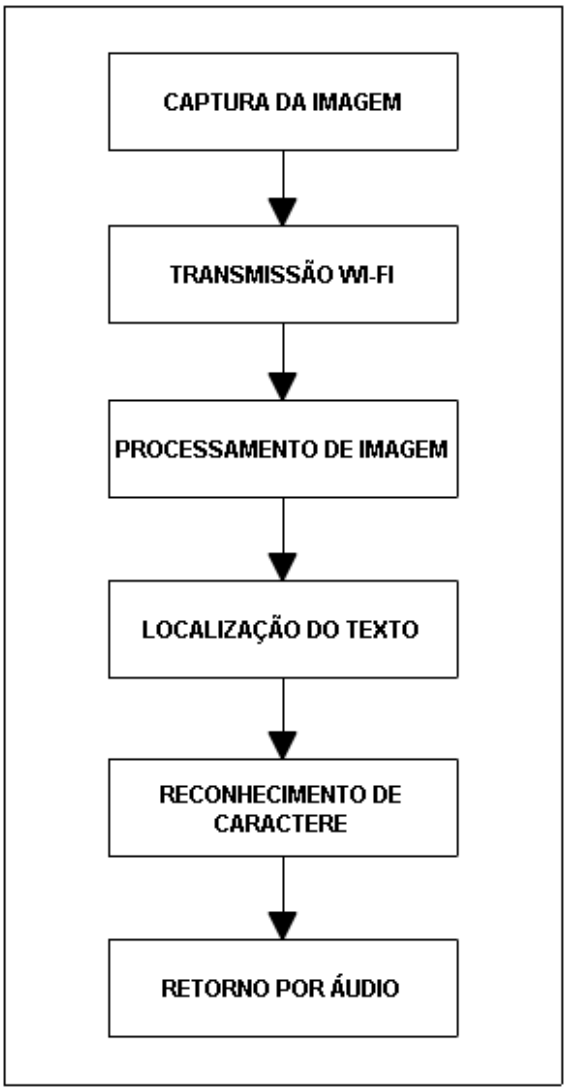


Figura 15 – Fluxograma do sistema.

Fonte: Autoria Própria.

3.8.1 O Código

O objetivo principal é reconhecer a ativação dos segmentos correspondentes aos resultados no *display*. Sabendo quais segmentos estão acesos, pode-se definir quais são os caracteres exibidos. A Figura 16 ilustra todos os segmentos encontrados no *display* do glícosimetro Contour Ts.

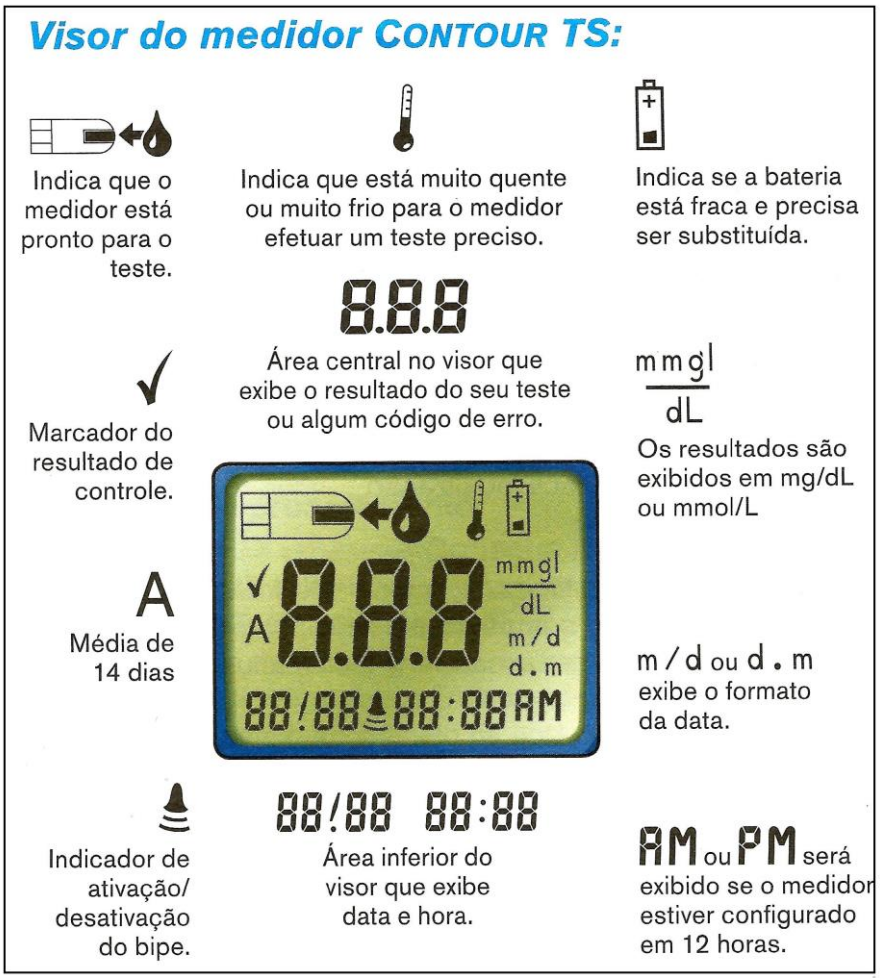


Figura 16 – Detalhamento das informações exibidas na tela do glicosímetro.
Fonte: Adaptado de Guia do usuário CONTOUR TS.

Neste estudo limitou-se em trabalhar somente com os caracteres centrais (correspondente aos resultados). Entretanto, vale lembrar que uma vez localizado a região destes segmentos, e desenvolvido o mecanismo de reconhecimento dos mesmos, fica fácil estender esse sistema para o *display* todo. Dessa forma, pode-se

localizar os segmentos complementares pela posição relativa aos segmentos centrais.

Para a finalidade de reconhecimento de caracter e retorno por áudio foi desenvolvida uma função principal e cinco funções auxiliares com finalidades específicas, porém complementares ao objetivo, sendo essas: *Captura_de_Imagem(), Processamento_de_Imagem(), Localiza_Texto(), Reconhecimento_de_Caractere()* e *Retorno_Usuario()* – todos os códigos e funções estão disponíveis no Apêndice A. Uma vez conhecida as etapas de estudo, pode-se detalhar melhor as partes do programa e suas respectivas funções.

No Quadro 5 é mostrada a estrutura de funcionamento da função principal. Na sequência são descritas as funções desenvolvidas neste trabalho.

```
%% Captura da imagem
Img = Captura_de_Imagem();

%% Processando imagem
[ImgRGB] = Processamento_de_Imagem( Img );

%% Localiza Texto
[Ib1 Ib2 Ib3] = Localiza_Texto(ImgRGB);

%% Reconhecendo caracteres
ResultadoM (1) = Reconhecimento_de_Caractere(Ib1);
if (ResultadoM (1) == 99)
    ResultadoM (1) = 0;
end
ResultadoM (2) = Reconhecimento_de_Caractere(Ib2);
ResultadoM (3) = Reconhecimento_de_Caractere(Ib3);

%% Retorno de Informação ao Usuário
Retorno_Usuario(ResultadoM)
```

Quadro 5 – Parte da função principal.

Fonte: Autoria Própria.

3.8.1.1 Captura de Imagem

Primeiramente é executada a função referente a captura de imagem através do aplicativo IP *Cam*. Pode-se fazer a aquisição das imagens (função *imread*) cap-

tadas pela câmera do *smartphone* através do número IP utilizando o código apresentado no Quadro 6:

```
function [ImagemCapturada] = Captura_de_Imagem()
%% Abre imagem no diretório do celular
url = 'http://192.168.0.22:8020/image.jpg';
ss  = imread(url);
ImagemCapturada = ss;
end
```

Quadro 6 – Função captura imagem.
Fonte: Autoria Própria.

No caso de alteração do IP, pode-se incluir no código uma simples lógica com laço do tipo "*for*" para testar os dois últimos números do IP (0 e 22) até encontrar um válido, uma vez que redes domésticas em geral utilizam a faixa "192.168".

3.8.1.2 Processamento de Imagem

Uma vez capturada e acessada, essa imagem precisará ser filtrada e processada de modo a extrair as informações de interesse. Para essa etapa foi escrito o *script* "*Processamento_de_Imagem.m*", que é responsável por receber uma variável "*ImagemCapturada*" e transformá-la em uma imagem binária, filtrando o máximo de informações desnecessárias possíveis. Para isso foi utilizado alguns dos recursos do *ToolBox* de Processamento de Imagem. Como exemplo, no Quadro 7 é apresentada a variável de entrada e na Figura 17 a respectiva imagem capturada.

```
%% Recebe imagem capturada
Img = ImgCapturada;
```

Quadro 7 – Função para abrir a imagem capturada.
Fonte: Autoria Própria.

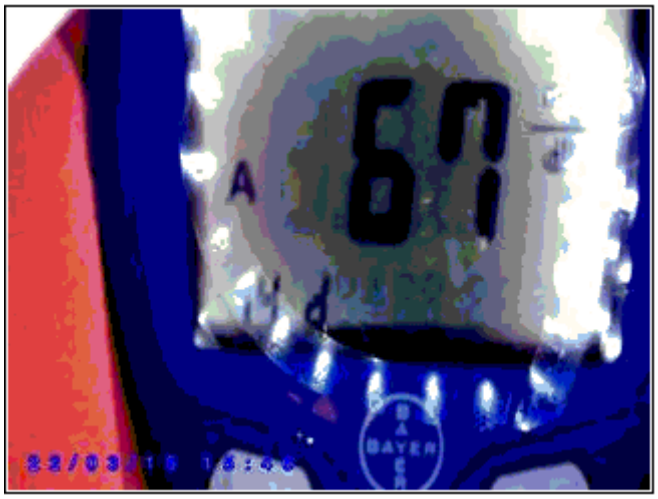


Figura 17 – Imagem capturada.
Fonte: Autoria Própria.

A função "*rgb2gray*" transforma a imagem de uma matriz de 3 camadas RGB em uma de 1 camada de valores correspondentes a uma escala de cinza. No Quadro 8 é apresentada a chamada da função e na Figura 18 a imagem convertida em escala de cinza.

```
%% Converter para imegem em escala de cinza
ImgCinza = rgb2gray(Img);
```

Quadro 8 – Função para converte imagem RGB para escala de cinza
Fonte: Autoria Própria.

Figura 18 – Imagem em escala de cinza.
Fonte: Autoria Própria.

Com o comando “*medfilt2*” (Quadro 9) aplica-se um filtro de médias 3x3, onde é feito a média das vizinhanças da matriz. Dessa forma, suaviza-se a imagem, deixando-a mais homogênea e tornando a transformação em binário mais eficiente, conforme mostrado na Figura 19.

```
%% Filtro de média das vizinhanças
ImgMedia = medfilt2(ImgCinza);
Img = medfilt2(ImgMedia);
```

Quadro 9 – Filtro média das vizinhanças
Fonte: Autoria Própria.

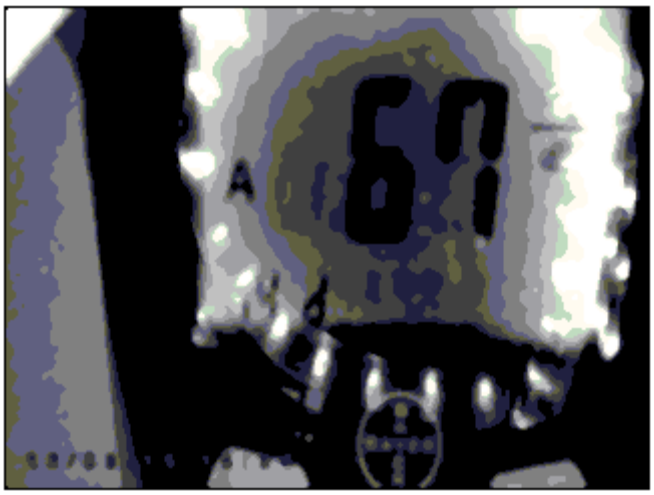


Figura 19 – Imagem com filtro de médias.
Fonte: Autoria Própria.

Tendo a imagem em condições mais adequadas pode-se, então, aplicar a função “*im2bw(Img,a)*” (Quadro 10), que transforma a imagem em binária, sendo “*Img*” a imagem a ser processada e “*a*” a variável responsável pela sensibilidade a ser considerada no processo. O resultado do processamento é mostrado na Figura 20.

```
%% Converter para binário
ImgB = im2bw(Img,0.05);
```

Quadro 10 – Função para converter imagem em escala de cinza para imagem binária.
Fonte: Autoria Própria

Figura 20 – Imagem binária.
Fonte: Autoria Própria.

Após transformada em binária, a imagem fica muito mais limpa e adequada para iniciar-se o processo de localização da região do texto.

3.8.1.3 Localização do Texto

A etapa a seguir para localização da região de texto foi baseada e adaptada das páginas de referência do MATLAB 2014b, onde é exemplificado a aplicação de algumas dessas técnicas de processamento de imagem para detecção automática e reconhecimento de textos em imagens naturais.

3.8.1.3.1 Detecção de Regiões Extremas Maximamente Estáveis

Em geral, caracteres de texto contém cores consistentes, então, inicia-se encontrando regiões de intensidades semelhantes na imagem usando o detector de regiões extremas maximamente estáveis (MSER, *Maximally Stable Extremal Regions*). Os códigos utilizados para esta detecção são apresentados no Quadro 11 com respectivo resultado apresentado na Figura 21.

.

```
%% Detectar e extrair regiões
grayImage = rgb2gray(Img);
% Faixa de tamanhos admicíveis
mserRegions = detectMSERFeatures(grayImage,'RegionAreaRange',[150 2500]);
% Extrair Regiões
mserRegionsPixels = vertcat(cell2mat(mserRegions.PixelList));

% Visualize as regiões MSER sobrepostas a imagem original
figure; imshow(Img); hold on;
plot(mserRegions, 'showPixelList', true,'showEllipses',false);
title('Regiões MSER');
```

Quadro 11 – Extração de regiões MSER.

Fonte: Autoria Própria.

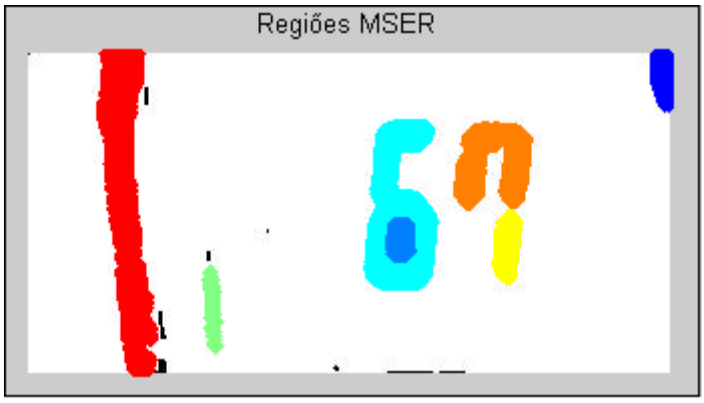


Figura 21 – Regiões MSER.

Fonte: Autoria Própria.

3.8.1.3.2 Detecção de Bordas

O texto escrito normalmente é colocado em um fundo claro, o que tende a produzir alta resposta a detecção de borda. Além disso, um cruzamento de regiões MSER com as bordas vai produzir regiões que são ainda mais prováveis de pertencem ao texto. As funções utilizadas e o resultado são apresentados no Quadro 12 e Figura 22, respectivamente.

```
%% Converter listas de pixel MSER em máscara binária
mserMask = false(size(grayImage));
ind = sub2ind(size(mserMask), mserRegionsPixels(:,2), mserRegionsPixels(:,1));
mserMask(ind) = true;

%% Detector de bordas
edgeMask = edge(grayImage, 'Canny');

%% Encontrar intersecção entre regiões MSER e bordas
edgeAndMSERIntersection = edgeMask & mserMask;
figure; imshowpair(edgeMask, edgeAndMSERIntersection, 'montage');
title('Bordas e intesecção entre bordas e regiões MSER')
```

Quadro 12 – Sequência de comando para detecção de bordas e intersecção entre bordas e regiões MSER.

Fonte: Autoria Própria.

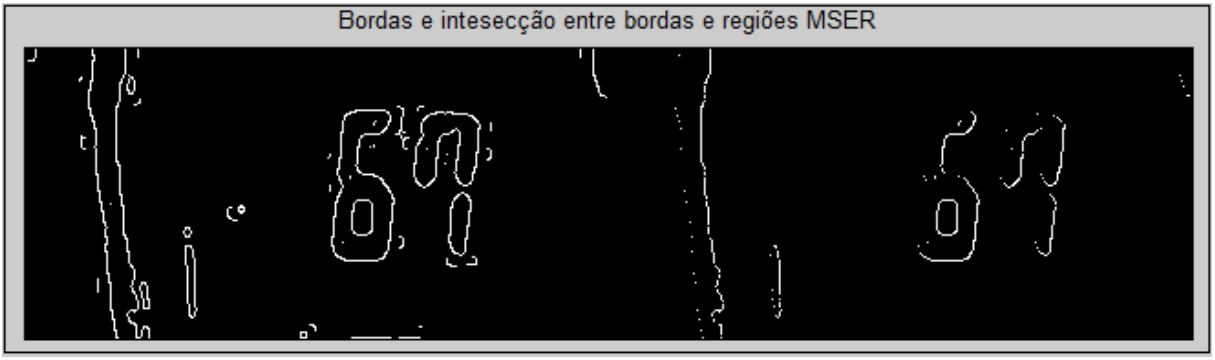


Figura 22 – Bordas (figura à esquerda) e intersecções (figura à direita) entre bordas e regiões MSER.

Fonte: Autoria Própria.

Nota-se que as regiões MSER originais em intersecção com as bordas ainda contém *pixels* que não fazem parte do texto. Dessa forma, pode-se usar a máscara de borda juntamente com o gradiente de borda para eliminar essas regiões. Na sequência são utilizados os comandos apresentados no Quadro 13 para fazer com que as bordas cresçam para fora utilizando o gradiente da imagem em torno das bordas, através da função auxiliar *"helperGrowEdges"*, com resultado na Figura 23.

```
[~, gDir] = imgradient(grayImage);
% Você tem que especificar se o texto é claro no fundo escuro ou vice versa
gradientGrownEdgesMask = helperGrowEdges(edgeAndMSERIntersection, gDir, 'LigthTextOnDark');
figure; imshow(gradientGrownEdgesMask);
title('Crescimento de bordas ao longo da direção do gradiente')
```

Quadro 13 – Código para crescimento de bordas utilizando o gradiente.

Fonte: Autoria Própria .

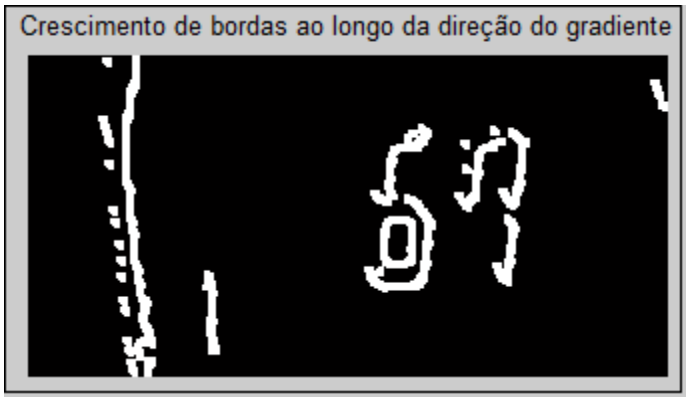


Figura 23 – Crescimento de bordas ao longo da direção do gradiente.
Fonte: Autoria Própria.

Esta máscara pode agora ser usada para remover os *pixels* que estão dentro das regiões MSER, mas provavelmente não fazem parte do texto, como apresentado no Quadro 14. Na Figura 24 é apresentado o resultado final com os segmentos bem definidos e e separados, além de ter mais alguns dos não textos filtrados (falsas bordas).

```
% Remover os pixels do gradiente de crescimento de bordas
edgeEnhancedMSERMask = ~gradientGrownEdgesMask & mserMask;

%Visualize os efeitos da segmentação
figure; imshowpair(mserMask, edgeEnhancedMSERMask, 'montage');
title('Região de MSER original e regiões MSER segmentadas')
```

Quadro 14 – Código para remover pixels do gradiente de crescimento de bordas.
Fonte: Autoria Própria.

Figura 24 – Região de MSER original e regiões MSER segmentadas.
Fonte: Autoria Própria.

3.8.1.3.3 Filtragem de Candidatos a Texto Usando Análise de Componentes Conectados

Alguns dos componentes restantes ligados agora podem ser removidos usando suas propriedades da região. Os limiares utilizados no Quadro 15 podem variar para diferentes fontes, tamanhos de imagem, ou idiomas. Na Figura 25 é apresentado o resultado da filtragem.

```
connComp = bwconncomp(edgeEnhancedMSERMask); % Enconte componentes conectados
stats = regionprops(connComp,'Area','Eccentricity','Solidity');

% Elimina as regiões que não corresponde as medidas do texto
regionFilteredTextMask = edgeEnhancedMSERMask;

regionFilteredTextMask(vertcat(connComp.PixelIdxList{[stats.Eccentricity] > .995})) = 0;
regionFilteredTextMask(vertcat(connComp.PixelIdxList{[stats.Area] < 30 | [stats.Area] > 2500})) = 0;
regionFilteredTextMask(vertcat(connComp.PixelIdxList{[stats.Solidity] < .2})) = 0;

% Visualize o resultado da filtragem
figure; imshowpair(edgeEnhancedMSERMask, regionFilteredTextMask, 'montage');
title('Candidatos a texto antes e depois de filtragem da região')
```

Quadro 15 – Código para filtragem de regiões com características desejadas.
Fonte: Autoria Própria.

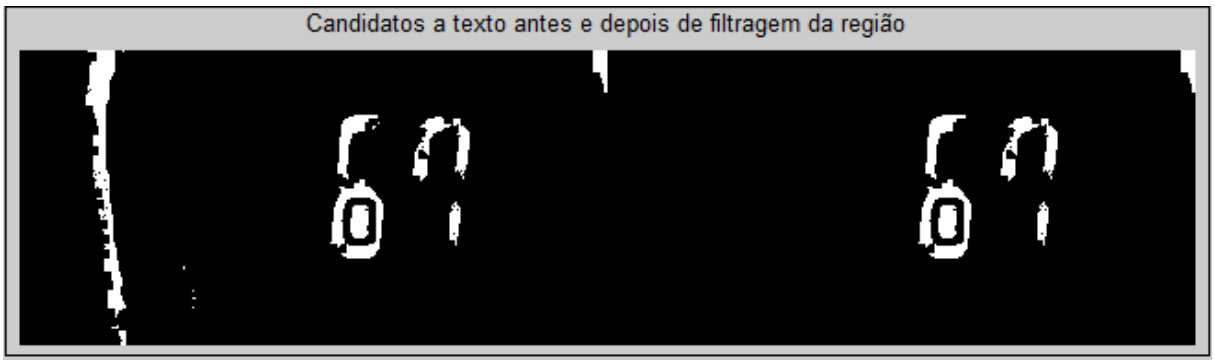


Figura 25 – Candidatos a texto antes e depois de filtragem de regiões.
Fonte: Autoria Própria.

3.8.1.3.4 Filtro para Largura do Traçado

Outro discriminador útil para textos em imagens é a variação de largura de traçado dentro de cada candidato a texto. Os caracteres na maioria dos idiomas têm uma largura de traçado ou espessura semelhante por toda sua extensão. Portanto, é útil remover regiões onde a largura do traçado exibe muita variação. Neste caso, a

função auxiliar *"helperStrokeWidth"* é utilizada (Quadro 16) para calcular a largura do traçado da Figura 26.

```
distanceImage    = bwdist(~regionFilteredTextMask);  % Compute variação de distâncias
strokeWidthImage = helperStrokeWidth(distanceImage); % Compute largura do traçado da imagem

%%Mostrar largura do traçado da imagem
figure; imshow(strokeWidthImage);
caxis([0 max(max(strokeWidthImage))]); axis image, colormap('jet'), colorbar;
title('Visualizar textos candidatos por largura de traçado')
```

Quadro 16 – Código para medição de larguras de traçado

Fonte: Autoria Própria.

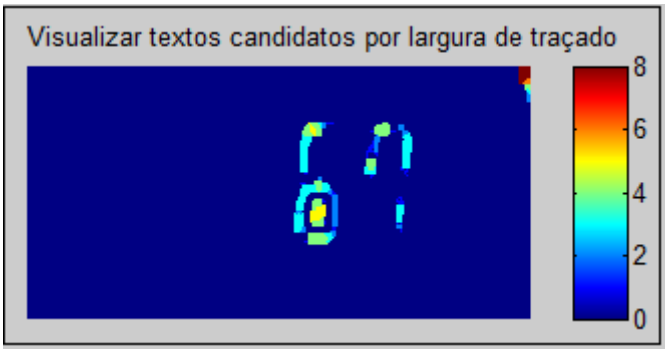


Figura 26 – Vizualização de textos candidatos por largura de traçado.

Fonte: Autoria Própria.

Nota-se que o pequeno pedaço restante que não pertence ao texto (canto superior direito) possui uma variação considerável em sua espessura, enquanto os caracteres possuem uma distância bem semelhante por toda sua extensão. Este pode agora ser filtrado, utilizando o coeficiente de variação de largura do traço (Quadro 17 e Figura 27).

```
% Encontre componentes conectados restantes
connComp = bwconncomp(regionFilteredTextMask);
afterStrokeWidthTextMask = regionFilteredTextMask;
for i = 1:connComp.NumObjects
    strokewidths = strokeWidthImage(connComp.PixelIdxList{i});
    % Computar normalização da variação de largura do traçado e comparar com o valor comum
    if std(strokewidths)/mean(strokewidths) > 0.5
        afterStrokeWidthTextMask(connComp.PixelIdxList{i}) = 0; % Remover do texto candidato
    end
end

% Visualize os efeitos do filtro de largura de borda
figure; imshowpair(regionFilteredTextMask, afterStrokeWidthTextMask,'montage');
title('Candidatos a texto antes e depois de filtragem por largura de borda')
```

Quadro 17 – Código para aplicação do filtro de largura de traçado.

Fonte: Autoria Própria.

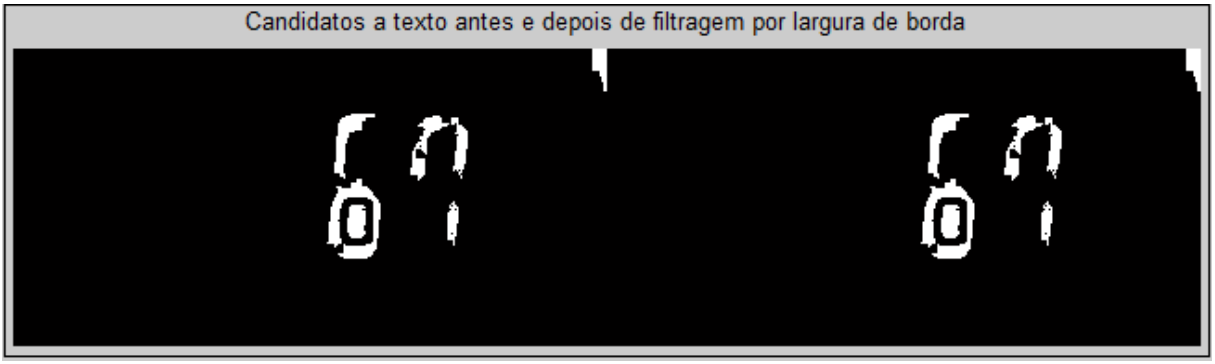


Figura 27 – Candidatos a texto antes e depois de filtragem por largura de borda.
Fonte: Autoria Própria.

3.8.1.3.5 Determinar Caixa de Delimitação da Região do Texto

Para calcular a caixa delimitadora da região do texto, deve-se primeiro mesclar os personagens individuais em um único componente ligado. Isto pode ser conseguido usando o fechamento seguido de abertura morfológica para limpar quaisquer valores aberrantes, conforme mostrado nos códigos do Quadro 18 e resultado da Figura 28.

```
se1=strel('disk',25);
se2=strel('disk',7);

afterMorphologyMask = imclose(afterStrokeWidthTextMask,se1);
afterMorphologyMask = imopen(afterMorphologyMask,se2);

% Exibir região correspondente após a máscara morfológica
displayImage = Img;
displayImage(~repmat(afterMorphologyMask,1,1,3)) = 0;
figure; imshow(displayImage);
title('Região da imagem sob máscara criada pela união de caracteres individuais')
```

Quadro 18 – Código para aplicação da caixa delimitadora.
Fonte: Autoria Própria.

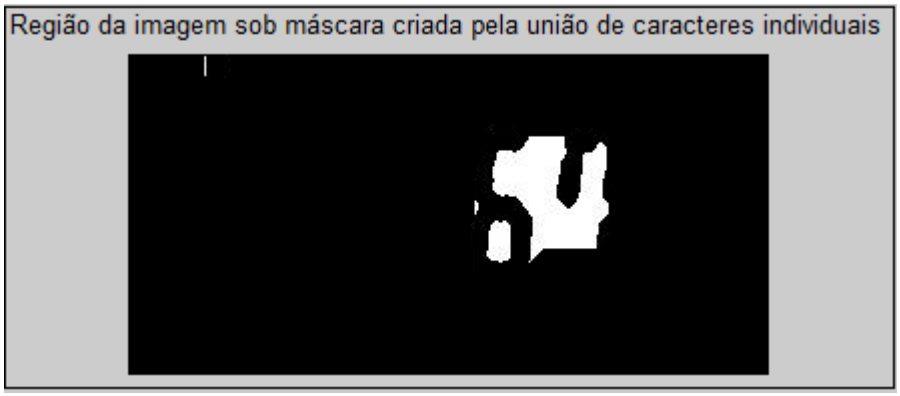


Figura 28 – Região de imagem sob máscara criada pela união de caracteres individuais.
Fonte: Autoria Própria.

3.8.1.3.6 Encontrando Caixa Delimitadora da Grande Região.

Os códigos para extração da região delimitada, bem como a região extraída pelo localizador de texto são mostrados no Quadro 19 e Figura 29, respectivamente.

```
areaThreshold = 3000; % limiar em pixels
connComp = bwconncomp(afterMorphologyMask);
stats = regionprops(connComp,'BoundingBox','Area');
boxes = round(vertcat(stats(vertcat(stats.Area) > areaThreshold).BoundingBox));

RegiaoTexto = imcrop(Img, boxes(1,:));
figure;imshow(RegiaoTexto)
```

Quadro 19 – Código para extração da região delimitada
Fonte: Autoria Própria.

Figura 29 – Região extraída pelo localizador de texto.
Fonte: Autoria Própria.

Finalmente obtem-se a região do texto. Em sequência, a imagem extraída passa por alguns recortes pré-calculados, de acordo com as dimensões estimadas dos caracteres, para separar os dígitos, como mostram as Figuras 30 e 31.

Figura 30 – Dígito dezena recortado.
Fonte: Autoria Própria.

Figura 31 – Dígito unidade recortado.
Fonte: Autoria Própria.

Tendo a imagem de cada dígito processada e armazenado separadamente, pode-se agora dar inicio a etapa de reconhecimento de caractere.

3.8.1.4 Reconhecimento de Caracter e Retorno das Informações

Nesta etapa são utilizadas três matrizes binárias correspondentes as regiões de cada um dos três dígitos. Sendo assim, e sabendo que se tratam de dígitos segmentados, a função de reconhecimento de caractere faz uma varredura em sete regiões de cada uma das matrizes, como ilustrado na Figura 32, buscando encontrar *pixels* com o valor 0 (correspondente a região preta). Dessa forma identificam-se as regiões com segmentos ativos e através de lógica de programação de teste ("*if*" e "*if-else*") pode-se classificar cada caractere.

Depois de reconhecidos os números são separados por centena, dezena e unidade, dependendo da posição em que estiverem na matriz de resultado, e então é definido quais áudios serão executados para retornar o valor medido. No caso do dígito "E", na posição do terceiro dígito da esquerda para direita, o sistema retornará o aviso referente ao erro indicado.

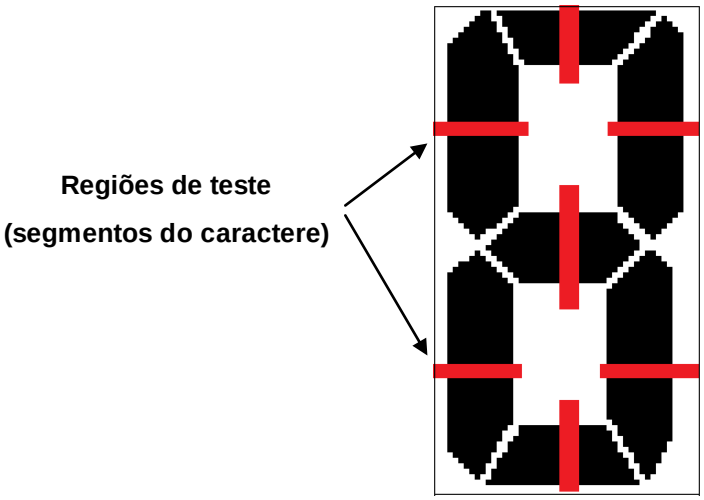


Figura 32 – Exemplo de análise de caracter.
Fonte: Autoria prórpia.

O retorno por áudio ao usuário foi feito através da execução de áudios pré-gravados, referentes a cada dígito. Para esta tarefa foi utilizado a função sintetizador de voz do Google Translate, disponível em https://translate.google.com.br/.

Para executar os áudios no MatLab foram utilizadas duas funções de sua própria biblioteca. A primeira delas, a função"*audioread"*, faz a leitura do arquivo e retorna duas informações, uma matriz de valores referentes ao sinal do áudio e a taxa de amostragem. A segunda, "*sound*", reproduz em forma de som o vetor de dados (y) na taxa de amostragem (Fs) que for escolhida. Nesse caso, utilizou-se a mesma taxa em que o arquivo foi gravado (Fs = 22050 Hz). O trecho de código a seguir exemplifica como foram utilizadas as duas funções no trabalho, onde "zero.acc" é o nome do arquivo que deseja-se abrir.

- [y,Fs] = audioread('zero.aac');
- sound(y,Fs);

Após o reconhecimento dos caracteres é executada uma sequencia de testes no Matlab para chamada da função de áudio referente ao valor encontrado para cada dígito.

4 RESULTADOS

Neste capítulo são apresentados os resultados dos testes experimentais para reconhecimento óptico de caracteres, de forma a testar a eficiência do método quando utilizado para as finalidades propostas.

Buscando verificar a eficiência do processo de reconhecimento foram feitas 9 medições sanguíneas distintas e para cada um desses resultados obtidos o sistema foi executado 50 vezes, somando um total de 450 leituras de *display*.

As medições nada mais são do que automedições feitas pelo autor desse trabalho, e para obter valores diferentes e com casas de potências distintas (valores com dois ou três dígitos) foram feitas medições em jejum e após ingestão de alimentos com alto teor de açúcar, o que explica algumas medições ultrapassando a casa da centena.

Os valores de medição encontrados foram 92, 67, 89, 94, 80, 99, 114, 100 e 102 miligramas por decilitro de sangue. Os seis primeiros tiveram as 50 tentativas de leitura executadas com sucesso, enquanto o restante obteve 48 acertos e 2 erros, 45 acertos e 5 erros e 46 acertos e 4 erros, respectivamente. Acredita-se que alguns dos erros provavelmente ocorreram devido a algum momento intermediário do foco automático da câmera ou algum outro problema com ruídos externos.

Somado a esses problemas, pode-se notar que os únicos testes que não foram unanimidade de acertos foram os com três dígitos, pois como o número que representa a centena pode ou não aparecer, dependendo do resultado, foi preciso incluir no código de localização um cálculo para que no momento de separação dos dígitos do texto seja reconheçido se são dois ou três números e com isso como separá-los adequadamente. Caso esse cálculo de “calibração” não esteja preciso é possível que em algumas leituras de *display* esse corte de imagem não seja feito adequadamente no dígito da centena e com isso reflita em erros de interpretação.

No Gráfico 1 é apresentada a porcentagem de acertos para cada uma das medições e na Tabela 1 o valores discriminados.

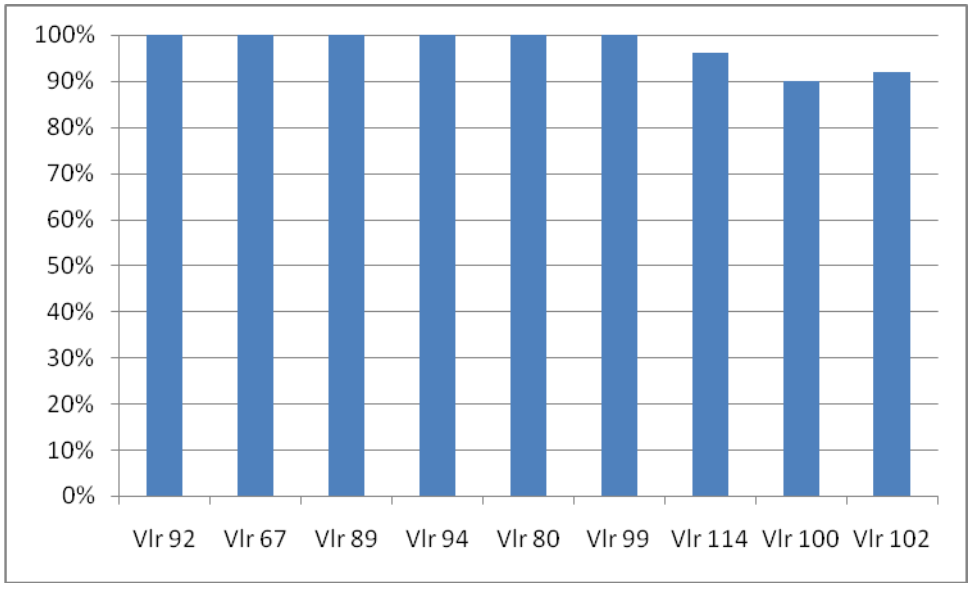


Gráfico 1 – Gráfico de porcentagem de acertos para as nove medidas diferentes.
Fonte: Autoria própria.

Tabela 2 – Número de Acertos.

Resultados da Medição	Número de Acertos	Acertos (%)
92	50	100,0
67	50	100,0
89	50	100,0
94	50	100,0
80	50	100,0
99	50	100,0
114	48	96,0
100	45	90,0
102	46	92,0

Fonte: Autoria própria.

As nove imagens correspondentes às medidas apresentadas no *display* do glicosímetro e obtidas pelas rotinas de processamento desenvolvidas são apresentadas nas Figuras 33 a 41.

Figura 33 – Imagem localizada valor 92.

Fonte: Autoria Própria.

Figura 34 – Imagem localizada valor 67.

Fonte: Autoria Própria.

Figura 35 – Imagem localizada valor 89.

Fonte: Autoria Própria

Figura 36 – Imagem localizada valor 94.

Fonte: Autoria Própria.

Figura 37 – Imagem localizada valor 80.

Fonte: Autoria Própria.

Figura 38 – Imagem localizada valor 99.
Fonte: Autoria Própria.

Figura 39 – Imagem localizada valor 114.
Fonte: Autoria Própria.

Figura 40 – Imagem localizada valor 100.
Fonte: Autoria Própria.

Figura 41 – Imagem localizada valor 102.
Fonte: Autoria Própria.

É importante citar também que foram encontrados muitos problemas causados pela variação da iluminação do ambiente, o que foi solucionado com a iluminação por LEDs, que foram posicionados para iluminar o perímetro da região a ser reconhecida, além do bloqueio parcial da luz externa.

5 DISCUSSÃO E CONCLUSÕES

Primeiramente, é importante salientar que, em princípio, o trabalho seria realizado por meio de alterações no *hardware* no próprio glicosímetro para acesso às informações das medidas. Seria colocado um microcontrolador para captar os sinais que o *display* do aparelho recebesse, para posteriormente interpretá-los e só então enviá-los para o computador. Na próxima etapa, seria realizado o processamento e retorno por áudio, assim como foi feito neste trabalho. Porém, verificou-se que seria necessário um microcontrolador com muitas entradas analógicas e digitais que pudessem verificar diferentes combinações de tensões dos mais de 30 terminais do *display*. Além de que seria muito complexo fazer toda essa adaptação física para diferentes tipos de glicosímetro. Por esse e outros motivos, já citados ao longo do trabalho, optou-se por utilizar um sistema com enfoque em *software* e que utilizasse a câmera de um *smartphone* já visando uma futura aplicação mais ampla, e de fácil adaptação para demais *smatphones* e glicosímetros.

Em relação aos gráficos e resultados obtidos, foi possível observar que a técnica utilizada no estudo se mostrou bastante eficiente para leitura de *displays* segmentados, uma vez que controlada a iluminação do *display* do glicosímetro. Seria interessante uma futura melhoria no código a fim de buscar as sensibilidades adequadas automaticamente para não necessitar de um sistema de iluminação própria na estrutura.

Quanto aos erros de leitura, seriam facilmente resolvidos fazendo algum mecanismo que reconhecesse a medição correta. Por exemplo, no caso em que o sistema a cada execução fizesse 5 leituras, das quais realizaria uma comparação com os valores encontrados e escolheria o que apareceu mais vezes e que as leituras fossem válidas, para que só então retornasse o valor mais adequado.

Alguns erros de leitura podem ter ocorridos devido a ruídos que resultam em alguns *pixels* pretos em áreas onde não deveriam existir, e por fim acabam acusando segmento ativo e gerando conflito na leitura. Para esses casos pode-se implementar um filtro simples que só considere o segmento ativo a partir de “n” *pixels* na região de varredura.

Outro ponto interessante é que apesar de foco desse trabalho ser identificar os três caracteres centrais do *display*, pode-se também, com os mesmos métodos

aplicados neste estudo, configurar o sistema de busca de texto e de reconhecimento de caracter para fazer outras buscas, como, por exemplo, para retornar todas as informações contidas na tela (data, hora, bateria, indicador de temperatura, etc...).

Vale lembrar que uma das idéias para a realização desse estudo é o desenvolvimento de um aplicativo em linguagem Java para algum sistema operacional de *smartphone*, tornando o sistema muito mais prático para o público alvo. O que impossibilitou essa aplicação direta foi a falta de tempo para um aprofundamento maior e mais adequado a linguagem Java e suas aplicações em *smartphones*. Esse motivo levou a opção pelo desenvolvimento em MATLAB, uma plataforma que o autor deste trabalho já possuia maior conhecimento.

Como sugestão de trabalho futuro, pode-se avaliar a possibilidade e desenvolvimento do sistema inteiramente em celular, dispensando a necessidade de uma rede WI-FI. Poderia ser desenvolvido também, um controle de iluminação por *software* que dispensasse a necessidade de iluminação externa. Uma última sugestão seria ampliar o sistema para mais tipos e marcas de glicosímetros.

Pode-se concluir que o objetivo principal deste trabalho foi atingido com sucesso através do desenvolvimento e avaliação inicial de um sistema de reconhecimento óptico de caracteres, utilizando *smartphone* e computador pessoal, com processamento via MATLAB, para leitura de glicosímetro convencional.

REFERÊNCIAS

ACEDO, J. 2014. Diabetes. Disponível em: <http://diabeticotipo2.blogspot.com.br/2014/11/como-saber-se-glicose-esta-alta.html>. Acesso em: 3 abr. 2015.

ALECRIM, E. O que é Wi-Fi (IEEE 802.11)?. 2008a. Disponível em: <http://www.infowester.com/wifi.php>. Acesso em: 29 mar. 2015.

ALECRIM, E. Tecnologia Bluetooth: O que é e como funciona?. 2008b. Disponível em: <http://www.infowester.com/wifi.php>. Acesso em: 14 abr. 2015.

ARAGÃO, R.; FERREIRA, B.; PINTO, H.. Retinopatia Diabética, 2013. Disponível em: <http://www.ligadeoftalmo.ufc.br/arquivos/ed_-_retinopatia_diabetica.pdf>. Acesso em: 21 ago. 2014.

BARBOSA, D.; DUARTE, D.A.. Achados Moleculares da Retinopatia Diabética em Foco Patogênese Bioquímica. 2010. Disponível em: <http://www.asmec.br/biblioteca/anais2010/Art.%20004.pdf>. Acesso em: 23 jan. 2014.

C. MELO. Chega de Dar o Sangue. Revista Veja. Publicado em: 6 de out. 2014.

CÂMARA, M.. Bluetooth: O que é e como funciona. 2012. Disponível em: http://www.techtudo.com.br/artigos/noticia/2012/01/bluetooth-o-que-e-e-como-funciona.html. Acesso em: 29 mar. 2015

COMITÊ DE AJUDAS TÉCNICAS, CAT. Secretaria Nacional de Promoção dos Direitos da Pessoa com Deficiência. Secretaria Especial dos Direitos Humanos da Presidência da República (SNPD/SDH/PR), Brasília, 2007, Ata da Reunião VII.

DUNNING, D.. The Code in MatLab for OCR. 2012. Disponível em: <http://www.ehow.com/info_12225276_code-matlab-ocr.html> Acesso em: 29 mar. 2015

FARIA, D.. Análise e Processamento de Imagem. 2010. Faculdade de Engenharia da Universidade do Porto. Disponível em:

<https://web.fe.up.pt/~tavares/downloads/publications/relatorios/MEB_Diogo_Faria_TrabPraticos.pdf> Acesso em: 23 mar. 2015

FERRARO, L. P. C. Bevacizumabe intravítreo para retinopatia diabética com neovascularização de retina persistente. 2014. Universidade de São Paulo. Faculdade de Medicina de Ribeirão Preto. Disponível em: <http://roo.fmrp.usp.br/teses/2012/lessia_de_pedro_cintra_ferraro.pdf> Acesso em: 12 de jul. 2014

FRAGAS, R.; SOARES, S.M.; BRONSTEIN, M.. Depressão e Diabetes Mellitus. 2008. Disponível em: <http://www.hcnet.usp.br/ipq/revista/vol36/s3/93.htm>. Acesso em: 21 ago. 2014.

G1. Inmetro reprova 15 marcas de glicosímetros testados. 2014. Disponível em: <http://g1.globo.com/fantastico/quadros/inmetro/noticia/2014/11/inmetro-reprova-15-marcas-de-glicosimetros-testados.html> Acesso em: 14 abr. 2015.

GLICOSÍMETRO, 2013. Disponível em: <http://www.tuasaude.com/glicosimetro/>. Acesso em: 21 ago. 2014.

GROOS, J.; SILVEIRO, S.; CAMARGO, J.; REICHELT, A.; AZEVEDO, M.. Diabetes Melito: Diagnóstico, Cassificaçâo e Controle Glicêmico. 2001. Arq Bras Endocrinol Metab vol.46 no.1 São Paulo Feb. 2002. Disponível em: <http://www.scielo.br/scielo.php?pid=S0004-27302002000100004&script=sci_arttext>. Acesso em: 21 ago. 2014

HIRSCHI, K.K.; D'AMORE, P.A. Control of Angiogenesis by the Pericyte: Molecular Mechanisms and Significance. 1997;79:419-28.

LACERDA, A.C. Doze milhoes de Brasileiros tem diabétes mas não sabe disso. 2013. Disponível em: <http://port.pravda.ru/science/11-11-2013/35603-ciencia_diabetes-0>. Acesso em: 23 jan. 2014.

LANDIM, W. O que é Wi-Fi?. 2012. Disponível em: <http://www.tecmundo.com.br/wi-fi/197-o-que-e-wi-fi-.htm>. Acesso em: 29 mar. 2015.

MAMÃO SEM AÇUCAR. Imetro reprova 15 marcas de glicosímetro. 2014. Disponível em: <www.mamaosemacucar.com>. Acesso em: 23 jun. 2014.

MATHWORKS. Automatically Detect and Recognize Texto in Natural. MatLab 2014a example. Disponível em: <http://www.mathworks.com/help/vision/examples/automatically-detect-and-recognize-text-in-natural-images.html> Acesso em: 23 mar. 2015.

MatLab. Disponível em: <http://pt.wikipedia.org/wiki/MATLAB>. Acesso em: 29 mar. 2015.

MINHA VIDA. Diabetes Tipo 1. 2012. Disponível em: <http://www.minhavida.com.br/saude/temas/diabetes-tipo-1>. Acesso em: 23 jan. 2014.

MINISTÉRIO DA SAÚDE SECRETARIA DE ATENÇÃO À SAÚDE DEPARTAMENTO DE ATENÇÃO BÁSICA. Caderno de Atenção Básica: Diabetes Mellitus. Cadernos de Atenção Básica - n.º 16 Série A. Normas e Manuais Técnicos Brasília - DF 2006.

PIMENTA, W. Diabetes Mellitus. 2003.Disponível em: <http://pt.scribd.com/doc/248178853/Diabetes-Mellitus#scribd>. Acesso em: 02 Abr. 2015.

POLAK, M.; NEWFIELD, R.S.; FIORETTO, P.; CZERNICHOW, P.; MARCHASE, R. Pathophysiology of Diabetes Complications. Diabetologia. 1997;40:B65-7.

QINGDAO HMD TECHNOLOGY AND DEVELOPMENT CO., LTD. Disponível em: < http://www.bjhmd.en.alibaba.com/ > Acesso em: 22 jun. 2015.

WI-FI ALLINCE. Disponível em: < http://www.wi-fi.org/)> Acesso em: 22 jun. 2015.

APÊNDICE A - Código Implementado em Matlab

Script Principal

```
close all
clear all
clc

a = 50; %número de execuções

for i=1:a
    %% Captura da imagem
    Img = Captura_de_Imagem();

    %% Processando imagem
    [ImgRGB] = Processamento_de_Imagem( Img );

    %% Localiza Texto
    [Ib1 Ib2 Ib3] = Localiza_Texto(ImgRGB);

    %% Reconhecendo caracteres
    ResultadoM (1) = Reconhecimento_de_Caractere(Ib1);
    if (ResultadoM (1) == 99)
        ResultadoM (1) = 0;
    end
    ResultadoM (2) = Reconhecimento_de_Caractere(Ib2);
    ResultadoM (3) = Reconhecimento_de_Caractere(Ib3);
    ResultadoM;
    %% Retorno de Informação ao Usuário
    Retorno_Usuario(ResultadoM)


    Teste(i,:) = ResultadoM %imprime resultado do teste
end
```

Captura de Imagem

```
function [ImagemCapturada] = Captura_de_Imagem()
    %% Abre imagem no diretório do celular
    url = 'http://192.168.0.16:8020/image.jpg';
    ss  = imread(url);
    ImagemCapturada = ss;
end
```

Processamento de Imagem

```
function [ ImgRGB ] = Processamento_de_Imagem( ImgCapturada )
    %% Recebe imagem capturada
    Img = ImgCapturada;

    %% Converter para imegem em escala de cinza
    ImgCinza = rgb2gray(Img);

    %% Filtro de média das vizinhanças
    ImgMedia = medfilt2(ImgCinza);
    Img = medfilt2(ImgMedia);

    %% Converter para binário
    ImgB = im2bw(Img,0.05);

    %% Converter para unint8
    A = ImgB;
    A(:,:,2) = ImgB;
    A(:,:,3) = ImgB;

    imwrite(A,'ImagemRGB.jpg','jpg')

    ImgRGB = imread('ImagemRGB.jpg');
end
```

Localiza Texto

```
function [ImgM1 ImgM2 ImgM3] = Localiza_Texto(ImgRGB)
%Img = imread('testelocalizacao.jpg');
Img = ImgRGB;
Img = Img(1:160,1:320,:);
 imshow(Img)

%% Detectar e extrair regiões
grayImage = rgb2gray(Img);
% Faixa de tamanhos admicíveis
mserRegions = detectMSERFeatures(grayImage,'RegionAreaRange',[150 2500]);
% Extrair Regiões
mserRegionsPixels = vertcat(cell2mat(mserRegions.PixelList));

% % Visualize as regiões MSER sobrepostas a imagem original
 figure; imshow(Img); hold on;
 plot(mserRegions, 'showPixelList', true,'showEllipses',false);
 title('Regiões MSER');


%% Converter listas de pixel MSER em máscara binária
mserMask = false(size(grayImage));
ind = sub2ind(size(mserMask), mserRegionsPixels(:,2), mserRegionsPix-
els(:,1));
mserMask(ind) = true;

%% Detector de bordas
edgeMask = edge(grayImage, 'Canny');

%% Encontrar intersecção entre regiões MSER e bordas
 edgeAndMSERIntersection = edgeMask & mserMask;
 figure; imshowpair(edgeMask, edgeAndMSERIntersection, 'montage');
 title('Bordas e intesecção entre bordas e regiões MSER')

[~, gDir] = imgradient(grayImage);
% Você tem que especificar se o texto é claro no fundo escuro ou vice versa
gradientGrownEdgesMask = helperGrowEdges(edgeAndMSERIntersection, gDir,
'LigthTextOnDark');
 figure; imshow(gradientGrownEdgesMask);
 title('Crescimento de bordas ao longo da direção do gradiente')

% Remover os pixels do gradiente de crescimento de bordas
edgeEnhancedMSERMask = ~gradientGrownEdgesMask & mserMask;

% %Visualize os efeitos da segmentação
 figure; imshowpair(mserMask, edgeEnhancedMSERMask, 'montage');
 title('Região de MSER original e regiões MSER segmentadas')

connComp = bwconncomp(edgeEnhancedMSERMask); % Enconte componentes conecta-
dos
stats = regionprops(connComp,'Area','Eccentricity','Solidity');

% Elimina as regiões que não corresponde as medidas do texto
regionFilteredTextMask = edgeEnhancedMSERMask;
```

```
regionFilteredTextMask(vertcat(connComp.PixelIdxList{[stats.Eccentricity] >
.995})) = 0;
regionFilteredTextMask(vertcat(connComp.PixelIdxList{[stats.Area] < 30 |
[stats.Area] > 2500})) = 0; %%TAMANHO DAS FIGURAS
regionFilteredTextMask(vertcat(connComp.PixelIdxList{[stats.Solidity] <
.2})) = 0;

% % Visualize o resultado da filtragem
 figure; imshowpair(edgeEnhancedMSERMask, regionFilteredTextMask, 'monta-
ge');
 title('Candidatos a texto antes e depois de filtragem da região')

distanceImage    = bwdist(~regionFilteredTextMask);  % Compute variação de
%distâncias
strokeWidthImage = helperStrokeWidth(distanceImage); % Compute largura do
%traçado da imagem

%%Mostrar largura do traçado da imagem
 figure; imshow(strokeWidthImage);
 caxis([0 max(max(strokeWidthImage))]); axis image, colormap('jet'), color-
bar;
 title('Visualizar textos candidatos por largura de traçado')

% Encontre componentes conectados restantes
connComp = bwconncomp(regionFilteredTextMask);
afterStrokeWidthTextMask = regionFilteredTextMask;
for i = 1:connComp.NumObjects
    strokewidths = strokeWidthImage(connComp.PixelIdxList{i});
    % Computar normalização da variação de largura do traçado e comparar
%com o valor comum
    if std(strokewidths)/mean(strokewidths) > 0.5
        afterStrokeWidthTextMask(connComp.PixelIdxList{i}) = 0; % Remover
%do texto candidato
    end
end

% % Visualize os efeitos do filtro de largura de borda
 figure; imshowpair(regionFilteredTextMask, afterStrokeWidthText-
Mask,'montage');
 title('Candidatos a texto antes e depois de filtragem por largura de bor-
da')


se1=strel('disk',25);
se2=strel('disk',7);

afterMorphologyMask = imclose(afterStrokeWidthTextMask,se1);
afterMorphologyMask = imopen(afterMorphologyMask,se2);

% % Exibir região correspondente após a máscara morfológica
displayImage = Img;
displayImage(~repmat(afterMorphologyMask,1,1,3)) = 0;
 figure; imshow(displayImage);
 title('Região da imagem sob máscara criada pela união de caracteres indi-
viduais')


areaThreshold = 3000; % limiar em pixels
connComp = bwconncomp(afterMorphologyMask);
```

```
stats = regionprops(connComp,'BoundingBox','Area');
boxes = round(vertcat(stats(vertcat(stats.Area) > areaThresh-
old).BoundingBox));

RegiaoTexto = imcrop(Img, boxes(1,:));
 figure;imshow(RegiaoTexto)

 %% Separar caracteres
RegiaoTextoAjustada = im2bw(RegiaoTexto);
[filas colunas] = size(RegiaoTextoAjustada);
imshow(RegiaoTextoAjustada)

ImgM1(79,34)  = true;
ImgM2(79,34)  = true;
ImgM3(79,34)  = true;


if (colunas >= 84 )
x = 34 - (colunas - 84);
    for (j=1:79)
        if (j>filas)
          for (i=1:34)
                ImgM1 (j,i) = 1;
          end
        else
        for (i=1:34)
            if (i<=x)
                ImgM1 (j,i) = 1;
            else
                ImgM1 (j,i) = RegiaoTextoAjustada(j,i-x);
            end
        end
        end
    end
    for (j=1:79)
        if (j>filas)
            for (i=1:34)
                ImgM2 (j,i) = 1;
            end
        else
          ImgM2 (j,:) = RegiaoTextoAjustada(j,((colunas - 73):(colunas -
40)));
        end
    end
    for (j=1:79)
        if (j>filas)
            for (i=1:34)
                ImgM3 (j,i) = 1;
            end
        else
          ImgM3 (j,:) = RegiaoTextoAjustada(j,((colunas - 33):(colunas)));
        end
    end
else
    for (j=1:79)
        for (i=1:34)
             ImgM1 (j,i) = 1;
        end
    end
```

```
for (j=1:79)
    if (j>filas)
         ImgM2(j,:) = 1;
    else
         ImgM2(j,:) = RegiaoTextoAjustada(j,(1:34));
    end
end

for (j=1:79)
    for (i=1:34)
        if ((j>filas)||((i+40)>colunas))
            ImgM3 (j,i) = 1;

        else
            ImgM3 (j,i) = RegiaoTextoAjustada(j,i+40);
        end
    end
end
end
```

Reconhecimento de Caractere

```
%% Recebe imagem binária
Img = ImagemBinaria;
%imshow(Img)
%% Declaração de variáveis
num = 100;
filtro = 0;

a = 0;
b = 0;
c = 0;
d = 0;
e = 0;
f = 0;
g = 0;

conta = 0;
contb = 0;
contc = 0;
contd = 0;
conte = 0;
conte = 0;
contf = 0;
contg = 0;

%% Definição da área de teste de segmento
A = (Img (1:14,20:22));
B = (Img (19:21,24:34));
C = (Img (59:61,24:34));
D = (Img (69:79,18:20));
E = (Img (59:61,1:11));
F = (Img (19:21,1:11));
G = (Img (35:45,20:22));

%% Teste de segmentos
for j=1:1:14
    for i=1:1:3
        if (A(j,i) == 0)
            conta = conta + 1;
            if (conta > filtro)
            a = 1;
            break
            end
        end
    end
end
for j=1:1:3
    for i=1:1:11
        if (B(j,i) == 0)
            contb = contb + 1;
            if (contb > filtro)
            b = 1;
            break
            end
        end
    end
end
for j=1:1:3
```

```
    for i=1:1:11
        if (C(j,i) == 0)
            contc = contc + 1;
            if (contc > filtro)
            c = 1;
            break
            end
        end
    end
end
for j=1:1:11
    for i=1:1:3
        if (D(j,i) == 0)
            contd = contd + 1;
            if (contd > filtro)
            d = 1;
            break
            end
        end
    end
end
for j=1:1:3
    for i=1:1:11
        if (E(j,i) == 0)
            conte = conte + 1;
            if (conte > filtro)
            e = 1;
            break
            end
        end
    end
end
for j=1:1:3
    for i=1:1:11
        if (F(j,i) == 0)
            contf = contf + 1;
            if (contf > filtro)
            f = 1;
            break
            end
        end
    end
end
for j=1:1:11
    for i=1:1:3
        if (G(j,i) == 0)
            contg = contg + 1;
            if (contg > filtro)
            g = 1;
            break
            end
        end
    end
end

MatrizSegmentos = [a b c d e f g];

%% Testando combinações de segmentos
% 1
if (a == 0)&&(b == 1)&&(c == 1)&&(d == 0)&&(e == 0)&&(f == 0)&&(g == 0)
    num = 1;
```

```
% 2
elseif (a == 1)&&(b == 1)&&(c == 0)&&(d == 1)&&(e == 1)&&(f == 0)&&(g == 1)
    num = 2;
% 3
elseif (a == 1)&&(b == 1)&&(c == 1)&&(d == 1)&&(e == 0)&&(f == 0)&&(g == 1)
    num = 3;
% 4
elseif (a == 0)&&(b == 1)&&(c == 1)&&(d == 0)&&(e == 0)&&(f == 1)&&(g == 1)
    num = 4;
% 5
elseif (a == 1)&&(b == 0)&&(c == 1)&&(d == 1)&&(e == 0)&&(f == 1)&&(g == 1)
    num = 5;
% 6
elseif (a == 1)&&(b == 0)&&(c == 1)&&(d == 1)&&(e == 1)&&(f == 1)&&(g == 1)
    num = 6;
% 7
elseif (a == 1)&&(b == 1)&&(c == 1)&&(d == 0)&&(e == 0)&&(f == 1)&&(g == 0)
    num = 7;
% 8
elseif (a == 1)&&(b == 1)&&(c == 1)&&(d == 1)&&(e ==1)&&(f == 1)&&(g == 1)
    num = 8;
% 9
elseif (a == 1)&&(b == 1)&&(c == 1)&&(d == 1)&&(e == 0)&&(f == 1)&&(g == 1)
    num = 9;
% 0
elseif (a == 1)&&(b == 1)&&(c == 1)&&(d == 1)&&(e == 1)&&(f == 1)&&(g == 0)
    num = 0;
% E
elseif (a == 1)&&(b == 0)&&(c == 0)&&(d == 1)&&(e == 1)&&(f == 1)&&(g == 1)
    num = E;
% H
elseif (a == 0)&&(b == 1)&&(c == 1)&&(d == 0)&&(e == 1)&&(f == 1)&&(g == 1)
    num = H;
% L
elseif (a == 0)&&(b == 0)&&(c == 0)&&(d == 1)&&(e == 1)&&(f == 1)&&(g == 0)
    num = L;
else
    num = 99;
end

end
```

Retorno ao Usuário

```
function [] = Retorno_Usuario( ResultadoM )
%% Retorno de Informação ao Usuário
        for i=1:1:3
            if (ResultadoM (i) == 0)
                if (i == 1)
                else
                    [y0,Fs0] = audioread('zero.aac');
                    sound(y0,Fs0);
                    pause(1);
                end
            elseif (ResultadoM (i) == 1)
                [y1,Fs1] = audioread('um.aac');
                sound(y1,Fs1);
            elseif (ResultadoM (i) == 2)
                [y2,Fs2] = audioread('dois.aac');
                sound(y2,Fs2);
                pause(1);

            elseif (ResultadoM (i) == 3)
                [y3,Fs3] = audioread('tres.aac');
                sound(y3,Fs3);
            elseif (ResultadoM (i) == 4)
                [y4,Fs4] = audioread('quatro.aac');
                sound(y4,Fs4);
            elseif (ResultadoM (i) == 5)
                [y5,Fs5] = audioread('cinco.aac');
                sound(y5,Fs5);
            elseif (ResultadoM (i) == 6)
                [y6,Fs6] = audioread('seis.aac');
                sound(y6,Fs6);
            elseif (ResultadoM (i) == 7)
                [y7,Fs7] = audioread('sete.aac');
                sound(y7,Fs7);
                pause(1);
            elseif (ResultadoM (i) == 8)
                [y8,Fs8] = audioread('oito.aac');
                sound(y8,Fs8);
                pause(1);
            else (ResultadoM (i) == 9)
                [y9,Fs9] = audioread('nove.aac');
                sound(y9,Fs9);
            end
                pause(1);
         end

        [yuni,Fsuni] = audioread('unidade.aac');
        sound(yuni,Fsuni);
        pause(1);
end
```