UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE TECNOLOGIA
CURSO DE ENGENHARIA ELÉTRICA

DANIEL MELO DOS SANTOS
GRR20142364

APLICATIVO DE SMARTPHONE PARA DETECÇÃO DE ÚLCERAS NO PÉ CAUSADAS POR DIABETES

CURITIBA
2019

DANIEL MELO DOS SANTOS

GRR20142364

APLICATIVO DE SMARTPHONE PARA DETECÇÃO DE ÚLCERAS NO PÉ CAUSADAS POR DIABETES

Plano de Trabalho apresentado ao Curso de Engenharia Elétrica da Universidade Federal do Paraná, como requisito à obtenção ao grau de Engenheiro Eletricista.

Orientadora: Profª. Drª. Giselle Lopes Ferrari Ronque

CURITIBA

2019

AGRADECIMENTOS

Agradeço à Deus, por ter me guiado durante toda a jornada.

À minha família pelo carinho e incentivo.

À professora Dr.ª Giselle Lopes Ferrari Ronque, por todo auxilio durante a execução esse trabalho.

À todos os amigos que fizeram parte dessa jornada de formação e conclusão do curso.

RESUMO

O projeto tem como objetivo o desenvolvimento de um aplicativo para auxiliar a identificação prévia de úlceras, localizadas na planta dos pés, de pacientes com caso de pé diabético.

O pé diabético é uma condição causada pela ação conjunta das complicações experimentadas pelos pacientes diabéticos. Se os sintomas não forem tratados corretamente, a formação de calos, ulceras e infeções, pode acabar por levar à necessidade de amputação do membro. Estudos tem demonstrado que é possível perceber uma variação de temperatura de aproximadamente 2 ºC em áreas com possíveis chances de ulceras. Dessa forma, como método de observação da temperatura, a termografia infravermelha tem apresentado bons resultados, pois além de não ser invasiva, a termografia também apresenta resultados rapidamente.

Nesse projeto, utilizou-se uma câmera termográfica FLIR ONE para a obtenção das imagens. O processamento das imagens foi feito separando as áreas de interesse através de dois limites, o primeiro foi obtido através do método de Otsu e o segundo é um limite próximo do máximo apresentado na imagem, cerca de 90% do maior valor.

O aplicativo é capaz de identificar as áreas de maior temperatura na planta dos pés do paciente, além de realizar o processamento de imagem separando as áreas de interesse.

Palavras chave: pé diabético, termografia, úlceras, diabetes.

Sumário

1 INTRODUÇÃO

O pé diabético é caracterizado por uma série de alterações que pode acontecer em paciente de diabetes com glicemia não controlada. Um dos principais fatores é a polineuropatia periférica, PNP, que afeta 50% dos pacientes com diabetes e que está presente em 85% dos casos de amputações. [SBEM, 2010]

Além da PNP, outros sintomas levam à caracterização de um quadro de pé diabético, como a doença arterial periférica (DAP), traumas, deformidades e também histórico de ulceras ou amputações.

Em conjunto, a baixa imunidade, a DAP, que causa má circulação de sangue, o que impede a chegada de células de defesa nas extremidades, a PNP, que causa danos aos nervos periféricos e diminui a sensação de dor do diabético, além de outros fatores dificultam a cicatrização de feridas e facilitam a infecção por micróbios. [Ministério da Saúde, 2016]

Por esses motivos, em 2010, 70% das cirurgias de amputação fossem realizadas por causa de diabetes com glicemia não controlada. [SBEM, 2010]

Úlceras por pressão, que acontece em casos de pé diabético, é uma lesão causada pela interrupção de corrente sanguínea em uma área específica. Essa lesão evolui com o tempo, gerando um calo e por fim uma ferida aberta, que é a úlcera. Geralmente leva a necrose dos músculos, ossos e estruturas de suporte. Como pode ser observado na figura 1.[Ministério da Saúde, 2016]

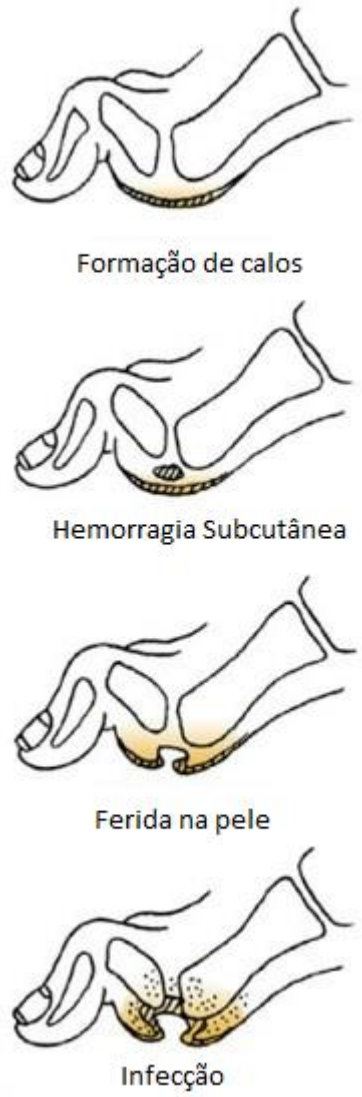


Figura 1 – Causas primárias de úlceras nos pés

Fonte: [SCHAPER, 2017]

Para uma pessoa com diabetes, um exame anual é imprescindível para a identificação do estado dos pés do diabético. Afim de determinar se algum dos sintomas de pé diabético pode ser observado, além de receber orientações sobre os cuidados que se deve tomar para evitar possuir o quadro de pé diabético. Após o exame, a pessoa pode ser designada a uma das categorias de classificação de risco da IWGDF, International Working Group on the Diabetic Foot, de 2015, que podem ser encontradas na tabela 1. [SCHAPER, 2017]

As áreas com maior risco de ulceração são mostradas na Figura 2 [SCHAPER, 2017].

Tabela 1 - Classificação de riscos IWGDF 2015

Categoria	Características	Frequência de exame
0	Sem neuropatia periférica	Uma vez por ano
1	Neuropatia periférica	Uma vez a cada 6 meses
2	Neuropatia periférica com doença arterial e/ou deformação no pé	Uma vez a cada 3-6 meses
3	Neuropatia periférica e um histórico de úlcera no pé ou amputação de membro inferior	Uma vez a cada 1 – 3 meses

Fonte: [SCHAPER, 2017]

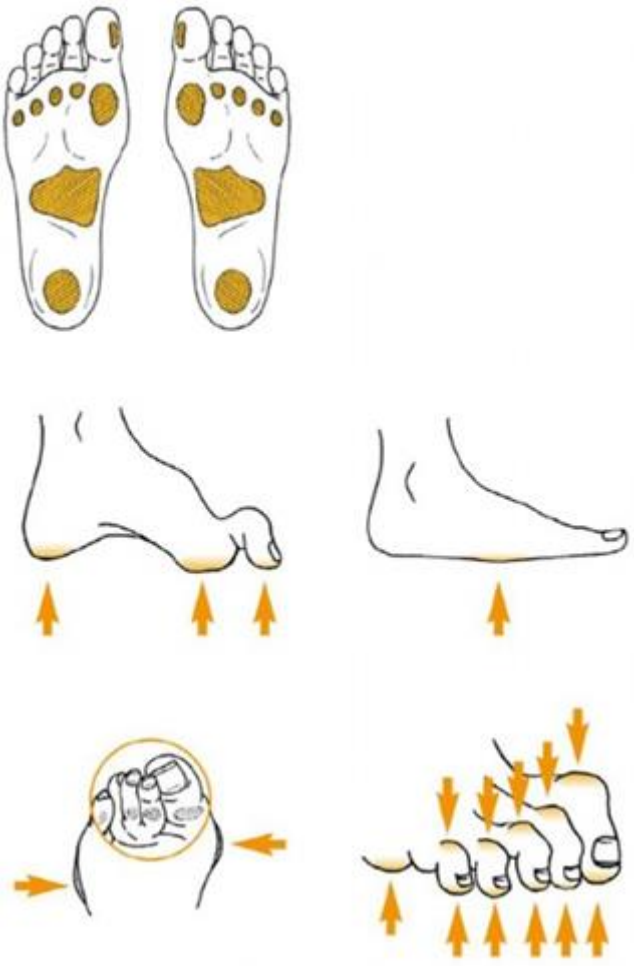


Figura 2 - Áreas de risco de úlceras
Fonte: [SCHAPER, 2017]

Pacientes que apresentarem qualquer um dos sintomas de pé diabético devem ser encorajados a usar um calçado adequado para os pés, tanto em ambientes internos quanto externos. Sendo que o calçado deve ser capaz de se adaptar a qualquer deformação que afete o pé do paciente. Principalmente caso o paciente possua neuropatia periférica. O sapato não deve ser muito apertado nem muito solto, se possível, de 1 a 2 cm maior que o pé. A largura interna deve ser igual à largura do pé e a altura deve ser suficiente para todos os dedos do pé. Caso o paciente ainda apresente dificuldades devido a amputações ou deformações nos pés, é necessário recomendar a utilização de calçados especiais, incluindo palmilhas e órteses. [SCHAPER, 2017]

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

Desenvolver, com base em trabalhos já desenvolvidos na área, um aplicativo Android que, em conjunto com a câmera Flir One, da FLIR Systems, sirva como auxílio ao diagnóstico de casos de úlceras e inflamações nos pés. O aplicativo servirá em especial para pacientes com caso de pé diabético, que, devido à neuropatia periférica, possuem menor sensibilidade nas extremidades. O aplicativo também é útil para usuários não diabéticos, pois úlceras e inflamações podem acontecer com qualquer pessoa, apesar de gerarem maiores danos em pacientes diabéticos. O aplicativo, que analisa as diferenças de temperatura nos tecidos do pé, deverá permitir ao paciente que possuir a câmera realizar o autoexame, que além de rápido, é não invasivo e de fácil realização. O que pode levar identificação prévia de caso de úlcera ou calos nos pés.

Temperaturas elevadas em regiões específicas dos pés podem aparecer até uma semana antes da úlcera neuropática. Dessa forma, a câmera térmica, que mede a temperatura a partir do calor emitido pelo corpo, pode identificar essa diferença de temperatura e permitir a identificação previa. Diferenças de temperaturas de mais de 2,2ºC comparadas com a mesma região entre os dois pés já é considerado como hipertermia, e monitorar essas diferenças é uma forma simples de identificação de formação de úlceras [FRAIWAN, 2017]

1.1.2 OBJETIVOS ESPECÍFICOS

O objetivo desse trabalho é construir um aplicativo de detecção de diferença de temperatura utilizando as imagens da câmera térmica Flir One e realizando o processamento da imagem através do método de Otsu. Para alcançar o objetivo, foram utilizadas as seguintes etapas:

- Aquisição da foto a partir do aplicativo Flir One, da FLIR Systems;
- Seleção da foto a partir do aplicativo criado;
- Processar da imagem utilizando o limite de Otsu e o limite maior;
- Identificar as temperaturas médias do pé e da área com temperatura mais elevada;

2 FUNDAMENTAÇÃO TEÓRICA

Atualmente, a empresa Siren tem como produto uma meia capaz de detectar as variações de temperatura em partes diferentes dos pés do diabético, permitindo ao médico identificar sinais de inflamações ou outros diagnósticos. A meia é feita de Neurofabric, tecido contendo vários microsensores que não são sensíveis à pele do usuário. O acompanhamento da temperatura, em comparação com o exame simplesmente visual, tem mostrado melhora de 87% dos casos de pé diabético [SIREN, 2019]

A Siren também conta com um aplicativo que notifica o usuário e mostra os dados obtidos pelas meias. Dessa forma, o paciente pode obter um serviço em que recebe 5 meias a cada 6 meses e acesso aos dados pelo site ou por aplicativo. [SIREN, 2019]

Outro exame que pode ser realizado é feito utilizando o monofilamento de 10g Semmes-Weinstein, que após ser aplicado nas mãos do paciente, para que se sabia qual a resposta inicial, pode ser aplicado em ambos os pés, assim como demonstrado nas figuras 3 e 4. O paciente, sem saber onde o monofilamento está sendo aplicado, deve responder se sente o toque e em qual local está sentindo. Caso o paciente responda corretamente duas das três aplicações, é possível considerar que a sensação está presente no local. Caso o paciente erre no mínimo duas das três aplicações é considerado que o paciente tem risco de ulceração no local. Dessa forma é importante para o profissional estar ciente das condições de uso do monofilamento, que, por tempo de desgastes pode apresentar resultados não confiáveis. [SCHAPER, 2017]

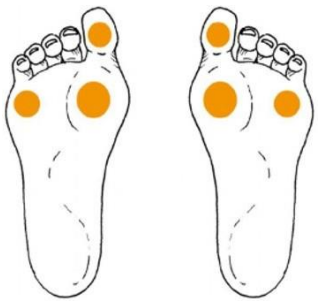


Figura 3 - Regiões do pé a serem testadas com o monofilamento
Fonte: [SCHAPER, 2017]

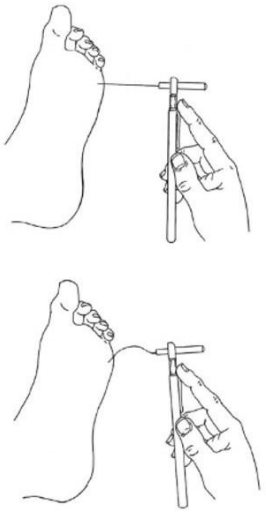


Figura 4 - Aplicação do Monofilamento
Fonte: [SCHAPER, 2017]

Outro instrumento para a verificação da sensibilidade do pé e a chance de úlceras é a o diapasão, que, da mesma forma que o monofilamento, deve ser aplicado primeiramente em nos pulsos, cotovelos ou clavícula do paciente para que o paciente saiba o que esperar. Depois, o diapasão é aplicado sobre uma parte óssea no lado dorsal da falange distal do primeiro dedo do pé, perpendicularmente e com pressão constante. Da mesma forma que no caso anterior, o paciente deve ser acertar pelo menos duas de três aplicações que possa se considerar que possui sensibilidade. A aplicação é demonstrada na figura 5 [SCHAPER, 2017]

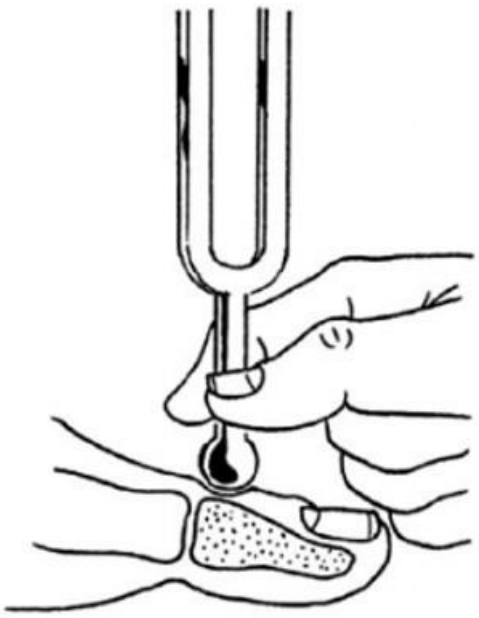


Figura 5 - Uso do diapasão para teste sensitivo no pé de pacientes
Fonte: [SCHAPER, 2017]

Em comparação com outras técnicas, uma das maiores vantagens da aquisição de imagens para medir a temperatura é a grande quantidade de dados que se pode obter em uma única imagem. E, com o avanço da tecnologia, os diagnósticos por termografia se tornam cada vez mais precisos, rápidos e com cada vez mais informação relevante. A termografia infravermelha já foi usada com sucesso em várias aplicações médias, como a detecção do câncer de mama, distúrbios vasculares, dor muscular e outros. [HERNANDEZ- CONTRERAS, 2016]

No caso do diabetes, a termografia tem sido usada para avaliar mudanças na temperatura corporal, analisar parâmetros metabólicos, estimar a glicose no sangue, etc. A aplicação mais comum da termografia na diabetes é detectar complicações do pé diabético, exibindo a distribuição de temperatura da região da planta dos pés, sobre a qual a maioria dos estudos são baseados [HERNANDEZ-CONTRERAS, 2016].

O método usual para a realização das medidas é que a pessoa atinja o equilíbrio térmico dos pés. Para isso, o paciente se mantém em uma mesma posição por cerca de 15 a 20 minutos, sem os sapatos e meias. [HERNANDEZ-CONTRERAS, 2016].

A partir de estudos anteriores, [J. VAN NETTEN, 2013], [PEREGRINA-BARRETO. 2014], foi possível notar que pacientes que não apresentam complicações nos pés demonstraram diferenças de temperatura inferiores a 2ºC, enquanto que pacientes que apresentavam ulceras não infectadas, ou calos abundantes apresentavam diferença de temperatura de temperatura maiores que 2ºC entre a mesma região em ambos os pés.

Para a identificação da área de interesse, é possível observar pelo histograma da planta dos pés, figura 8, que existem três áreas distintas com calores de temperatura: o do plano de fundo, a temperatura média dos pés e, por último, a área de temperatura mais elevada, que desejamos identificar. A partir do processamento de imagens, a área será isolada e a temperatura média calculada.

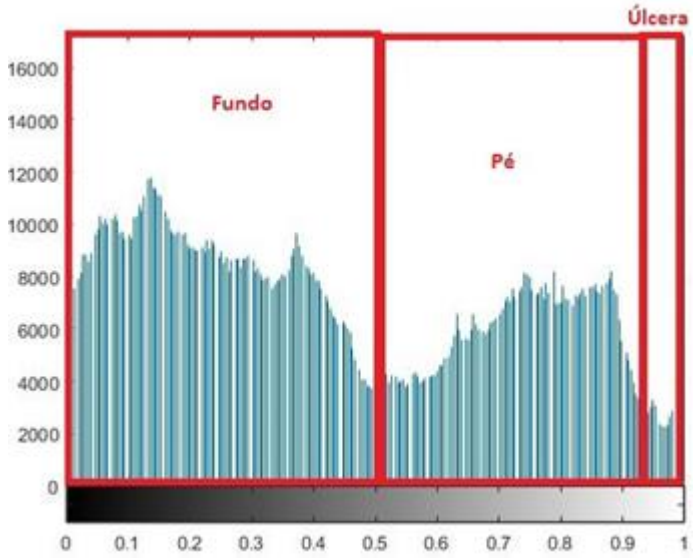


Figura 8 - Separação do Histograma entre áreas de interesse
Fonte: [P. Bach, 2018]

3 DESENVOLVIMENTO

O aplicativo foi programado em Android utilizando o software Android Studio, programa oficial para desenvolvimento de aplicativos para o sistema operacional Android, da Google. Também foi utilizada a biblioteca OpenCV para o processamento das imagens. A biblioteca OpenCV é uma biblioteca “open source” com mais de 2500 algoritmos otimizados para processamento de imagens e aprendizado de máquina. As fotos foram tiradas utilizando a câmera térmica Flir One.

O fluxograma é apresentado na figura 6:

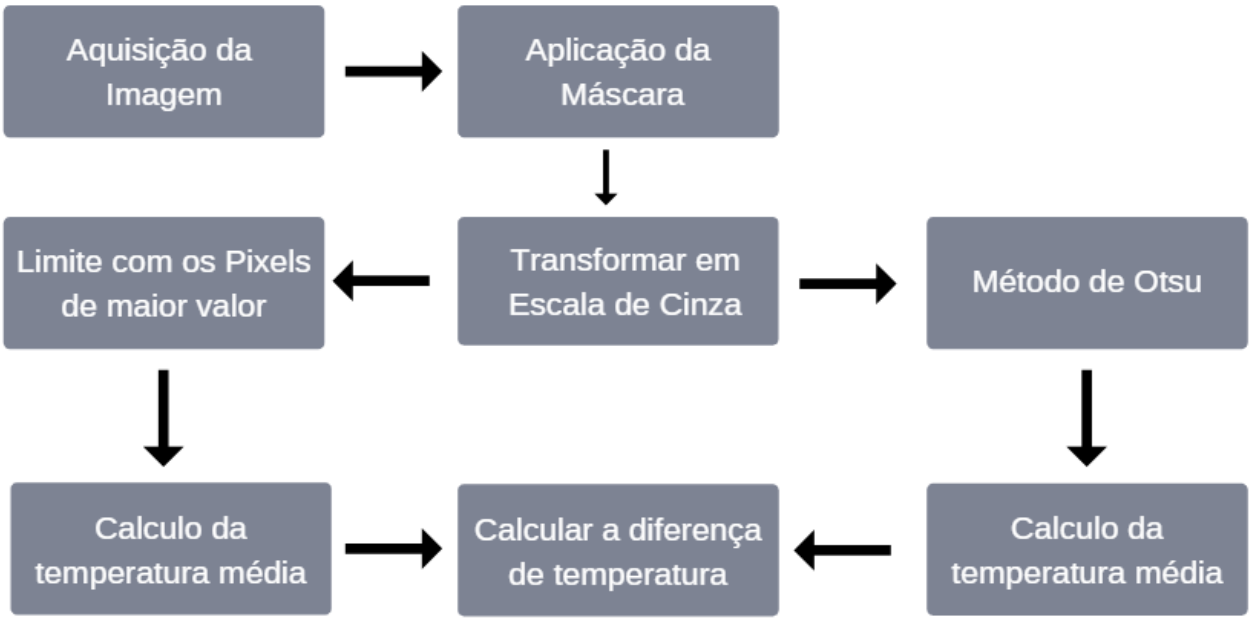


Figura 7 – Fluxograma
Fonte: Autor, 2019

3.1 SISTEMA DE AQUISIÇÃO DA IMAGEM TÉRMICA

Neste projeto, a câmera utilizada para a aquisição da imagem térmica é a Flir One, da FLIR Systems, em conjunto com o aplicativo Flir Mobile App. A câmera, mostrada na figura 9 e 10 consegue medir temperaturas de -20ºC à 120 ºC, com sensibilidade de 0,1ºC. A resolução das imagens é de 160 x 120 pixels. No aplicativo, os valores de temperatura são identificados pelo através de uma

paleta de cores, que o usuário pode selecionar. Porém, as cores, e também os valores dos pixels das imagens, são calibrados automaticamente em função dos valores máximos e mínimos de temperatura presentes na imagem. Dessa forma, o aplicativo apresenta o valor da temperatura no ponto central da imagem, porém, para o cálculo da temperatura em outros pontos da imagem, é necessário considerar os valores máximos e mínimos presentes.

As imagens obtidas pelo aplicativo são salvas no formato Joint Photographic Experts Group (JPEG), e possuem dados que permitem a identificação das temperaturas na imagem. A imagem é formada combinando-se a o espectro visual, com alto contraste, e o térmico, criando uma imagem rica em detalhes.

Figura 9 - Câmera FLIR ONE acoplada a um celular
Fonte: Autor, 2019

Figura 10 - Câmera FLIR ONE visão frontal
Fonte: Autor, 2019

3.2 AQUISIÇÃO DA IMAGEM E APLICAÇÃO DA MASCARA

A foto é tirada pelo aplicativo Flir One, da FLIR Systems. Então, no aplicativo criado, é selecionada na biblioteca criada pelo aplicativo Flir One. É criada uma cópia da foto original em formato Bitmap. É possível observar, a partir das figuras 11 e 12, que caso o usuário tire foto sem uma barreira de temperatura constante atrás dos pés, as cores do corpo do usuário se confundem com as cores que representam a temperatura dos pés. Por isso o corpo do usuário atrapalha na identificação da temperatura média do pé, ao adicionar valores no fundo da imagem.

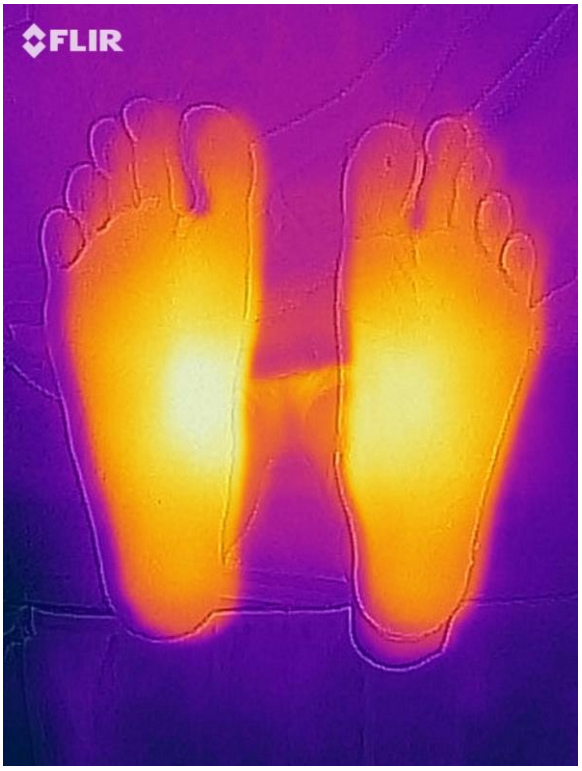


Figura 11 – Foto térmica dos pés
Fonte: Autor, 2019

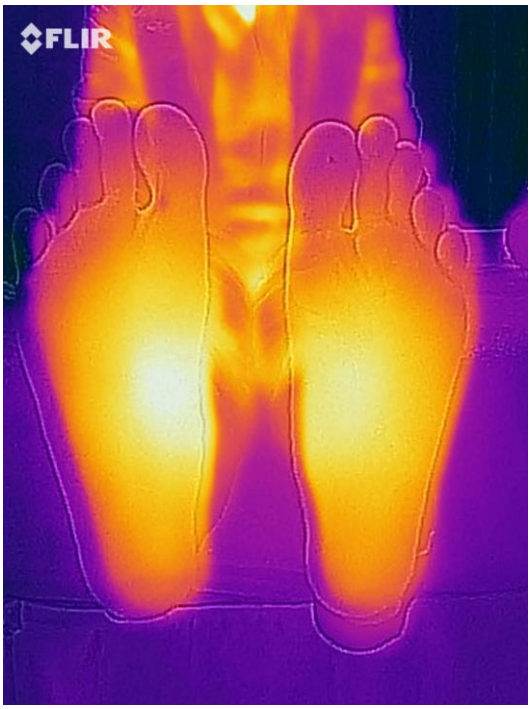


Figura 12 – Foto com ruído de fundo
Fonte: Autor, 2019

Para a utilização da máscara para filtro do ruído de fundo, a imagem em Bitmap deve possuir o mesmo tamanho da imagem de máscara. Dessa forma, foi criada uma imagem em Bitmap do mesmo tamanho da imagem da máscara.

Assim, a imagem final é uma junção da imagem original com a imagem de máscara, onde os pixels no meio e nas bordas são pixels de cor preta. Como é possível se observar na figura 13, como a imagem de máscara escolhida possui tamanho fixo, é necessário que a foto seja tirada com os pés na melhor posição possível.

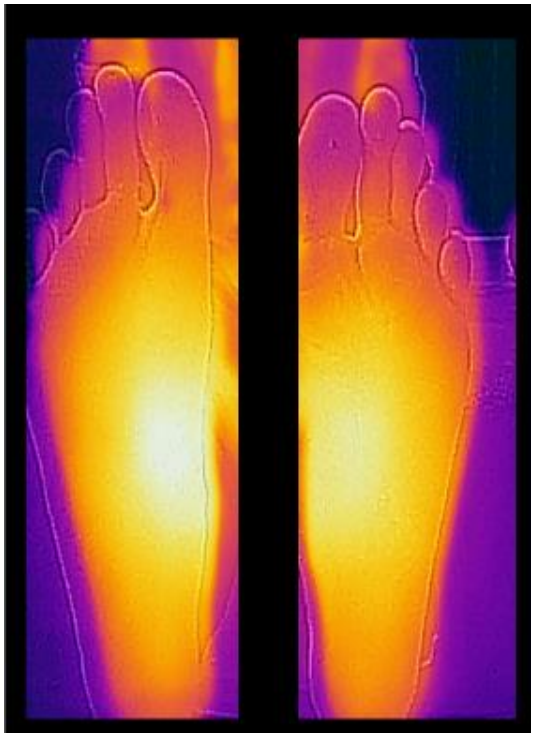


Figura 13 – Foto colorida com mascara
Fonte: Autor, 2019

Figura 14 – Foto com a máscara em escala de cinza
Fonte: Autor, 2019

3.3 MÉTODO DE OTSU

O método de Otsu, criado por Noboyuki Otsu, é um método iterativo que permite a obtenção de um valor de limite entre todos os possíveis valores de uma imagem em escala de cinza. Após determinar esse limite, é possível separar a imagem em duas áreas, a parte de fundo da imagem e a parte da frente, que são chamadas de classes.

A versão computacional do método determina o melhor valor de limite através do cálculo do maior valor da variância entre classes. É possível também determiná-lo através do menor valor de variância intra-classe, como pode ser visto nas equações abaixo:

$$\text{Within Class Variance} \quad \sigma_W^2 = W_b\,\sigma_b^2 + W_f\,\sigma_f^2 \quad \text{(as seen above)} \quad (1)$$

$$\text{Between Class Variance} \quad \sigma_B^2 = \sigma^2 - \sigma_W^2 \quad (2)$$

$$= W_b(\mu_b - \mu)^2 + W_f(\mu_f - \mu)^2 \quad (\text{where } \mu = W_b\,\mu_b + W_f\,\mu_f) \quad (3)$$

$$= W_b\,W_f\,(\mu_b - \mu_f)^2 \quad (4)$$

Figura 7 – Variância entre classes
Fonte: Labbookpages, 2010

Onde, W é o peso da classe, calculado a partir da soma de todos os valores dos pixels dentro da classe dividido pelo da soma de todos eles. O valor de μ é a média dos valores dos pixels dentro da classe. Como é possível observar pela equação, tanto as médias quanto os pesos são calculados para a classe da frente quanto do fundo, simbolizadas pelas letras b e f.

O método de Otsu pode ser exemplificado pelas figuras 15, 16 e 17. Sendo a figura 15 a imagem original, com o seu histograma, e as figuras 16 e 17 demonstrando os cálculos da menor variância intra-classe e entre classes, respectivamente.

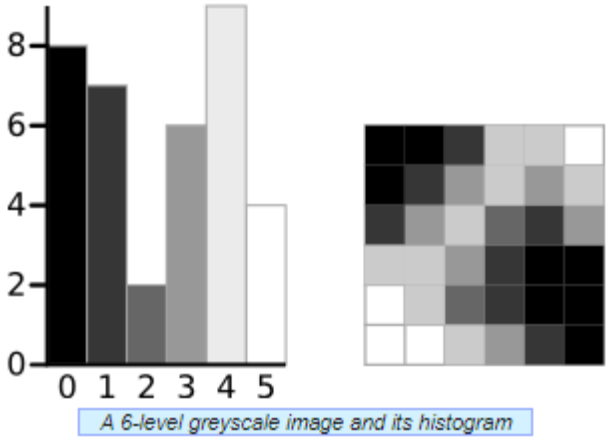


Figura 15 – Metodo de Otsu
Fonte: Labbookpages, 2010

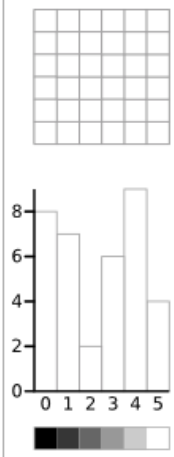
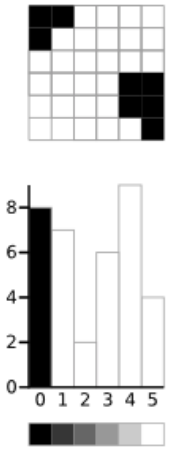
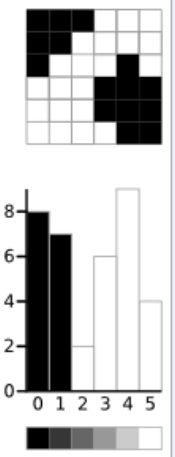
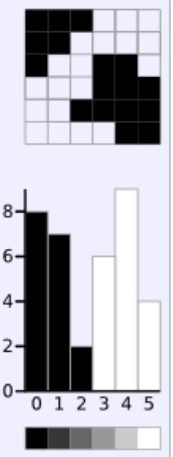
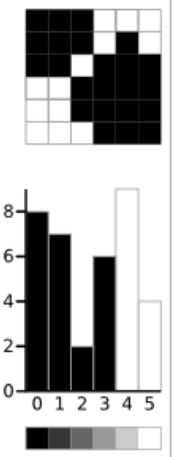
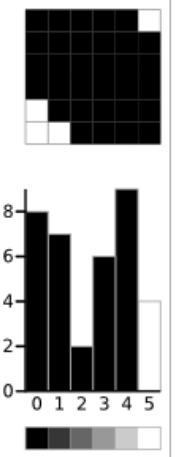
Threshold	T=0	T=1	T=2	T=3	T=4	T=5
						
Weight, Background	W_b = 0	W_b = 0.222	W_b = 0.4167	W_b = 0.4722	W_b = 0.6389	W_b = 0.8889
Mean, Background	M_b = 0	M_b = 0	M_b = 0.4667	M_b = 0.6471	M_b = 1.2609	M_b = 2.0313
Variance, Background	σ^2_b = 0	σ^2_b = 0	σ^2_b = 0.2489	σ^2_b = 0.4637	σ^2_b = 1.4102	σ^2_b = 2.5303
Weight, Foreground	W_f = 1	W_f = 0.7778	W_f = 0.5833	W_f = 0.5278	W_f = 0.3611	W_f = 0.1111
Mean, Foreground	M_f = 2.3611	M_f = 3.0357	M_f = 3.7143	M_f = 3.8947	M_f = 4.3077	M_f = 5.000
Variance, Foreground	σ^2_f = 3.1196	σ^2_f = 1.9639	σ^2_f = 0.7755	σ^2_f = 0.5152	σ^2_f = 0.2130	σ^2_f = 0
Within Class Variance	σ^2_W = 3.1196	σ^2_W = 1.5268	σ^2_W = 0.5561	**σ^2_W = 0.4909**	σ^2_W = 0.9779	σ^2_W = 2.2491

Figura 16 – Metodo de Otsu
Fonte: Labbookpages, 2010

Threshold	T=0	T=1	T=2	T=3	T=4	T=5
Within Class Variance	σ^2_W = 3.1196	σ^2_W = 1.5268	σ^2_W = 0.5561	**σ^2_W = 0.4909**	σ^2_W = 0.9779	σ^2_W = 2.2491
Between Class Variance	σ^2_B = 0	σ^2_B = 1.5928	σ^2_B = 2.5635	**σ^2_B = 2.6287**	σ^2_B = 2.1417	σ^2_B = 0.8705

Figura 17 – Metodo de Otsu
Fonte: Labbookpages, 2010

A partir do método de Otsu, foi possível diferenciar o plano de fundo da região onde se encontra os pés. Pois dessa forma o valor médio da temperatura no fundo, considerando a máscara, é diferente do valor médio na região dos pés. A imagem resultante foi suavizada, afim de diminuir os erros gerados pela segmentação, e foi possível diferenciar a região dos pés da parte do plano de fundo, como pode-se ver a figura 18.

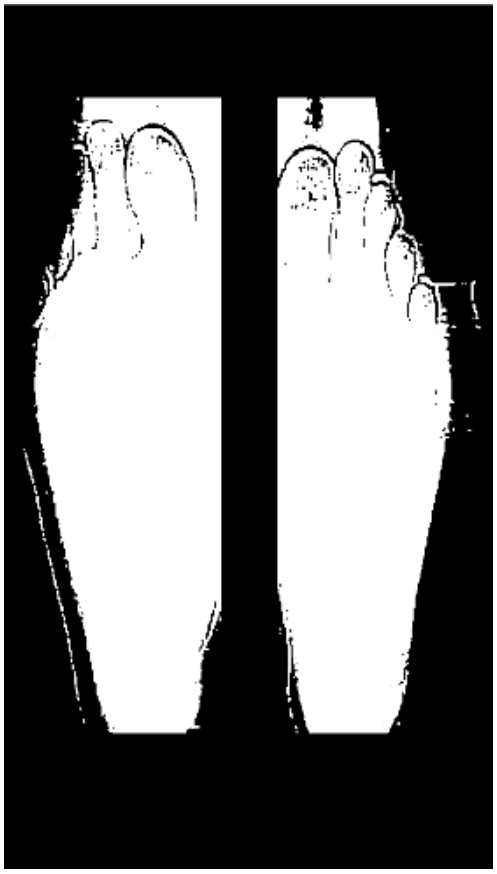


Figura 18 – Máscara pelo método de Otsu
Fonte: Autor, 2019

O resultado da imagem através do método de Otsu gera uma segunda máscara para a imagem em escala de cinza. Da mesma forma, se deseja identificar as regiões de maior temperatura da imagem. Para isso, o método utilizando foi selecionar somente os pixels de valores iguais ou superiores a 90% do maior valor presente na imagem. Esse valor foi selecionado levando em conta o histograma da figura 8, onde se pode identificar que a região de maior temperatura está localizada nas regiões com aproximadamente 90% do valor limite para a imagem[P. Bach, 2018]. Como a imagem salva pelo aplicativo da FLIR Systems sempre é ajustada para a temperatura ambiente, o maior valor de temperatura na imagem sempre corresponderá a 255, que é o maior valor possível para um pixel na escala de cinza. Por isso, o valor do limite escolhido é de 230, aproximadamente 90% do máximo em todas as imagens possíveis.

As regiões de maior temperatura foram representadas pela imagem 19.

Figura 19 – Máscara das regiões de temperatura mais alta
Fonte: Autor, 2019

Aplicando ambas as imagens como máscaras na imagem original em escala de cinza tempos as seguintes figuras 20 e 21.

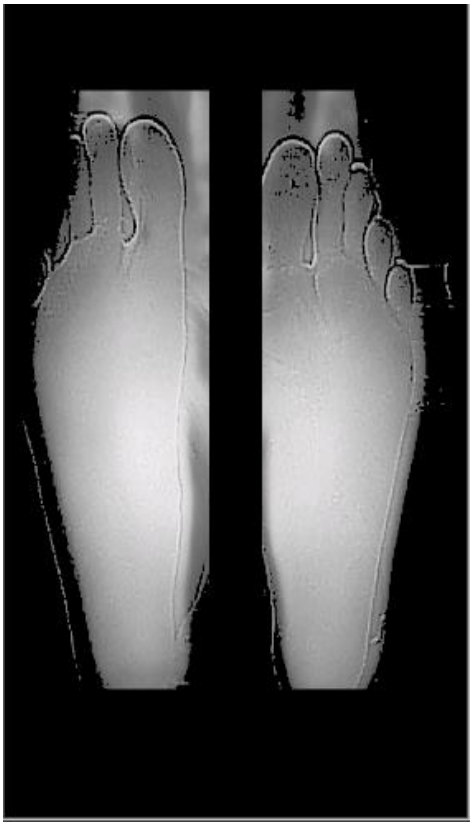


Figura 20 – Seleção da região do pé
Fonte: Autor, 2019

Figura 21 – Seleção da região de temperatura mais alta
Fonte: Autor, 2019

3.4 CÁLCULO DA TEMPERATURA

Para poder identificar a se a região de maior temperatura excede em aproximadamente 2ºC a temperatura média do pé do usuário, e nesse caso recomendar-se que o usuário busque auxílio médico, é necessário identificar a temperatura média de toda a região do pé. Também é necessária a temperatura média da região identificada como de maior temperatura.

O método utilizado foi a obtenção do valor médio de temperatura das figuras 20 e 21, que representam a região do pé e a região de maior temperatura.

Os valores de maior e menor temperatura devem ser obtidos através do aplicativo da FLIR Systems para que se possa identificar o valor correspondente em temperatura para cada pixel da imagem.

A equação utilizada foi a seguinte:

$$Temp.média = \frac{\Sigma\ Valores\ dos\ Pixels}{Numero\ total\ de\ pixels} * \left(\frac{temp.\max - temp.min}{255.0}\right) + temp.\ min$$

Onde o somatório dos "Valores dos Pixels" é a soma de todos os valores de pixels identificados na imagem, esses valores devem ser maiores do que 0 devido à aplicação das máscaras na imagem. O "Numero total de pixels" é obtido a partir da contagem de todos os pixels cujo valor é maior que 0. Os valores de "temp. max" e "temp. min" são os valores inseridos pelo usuário sobre as temperaturas máxima e mínima.

4. RESULTADOS

A figura 22 foi tirada esperando-se 15 minutos a partir do momento que se retirou o calçado e a meia.

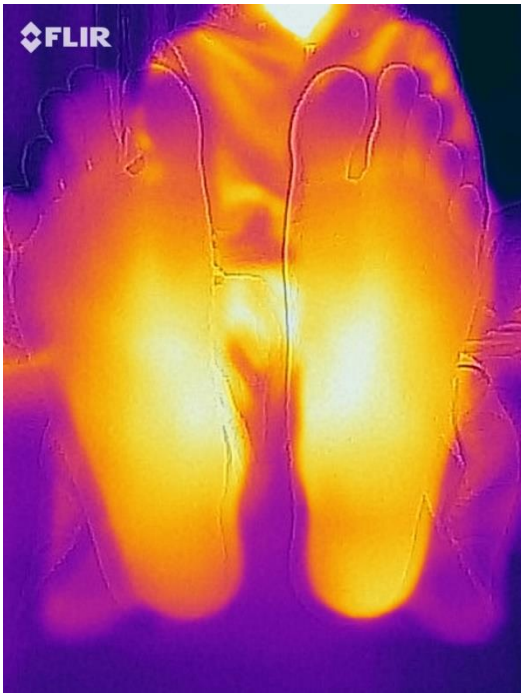


Figura 22 - Modelagem do pé saudável
Fonte: Autor, 2018

Após a seleção da imagem foram colocados os valores limites de temperatura medidos através da própria câmera, foi identificado tanto o valor médio da temperatura dos pés, quanto o valor médio da região de maior temperatura. Como se pode ver na figura 23.

É possível perceber pelo resultado que a temperatura da região mais quente está aproximadamente 2ºC mais quente que a temperatura média dos pés. Porém, como é possível observar pela imagem da região mais quente, a imagem de baixo, é possível notar que ambos os pés apresentam a temperatura elevada na mesma região, o que mostra que os pés não apresentam chances de ulcera nessas regiões.

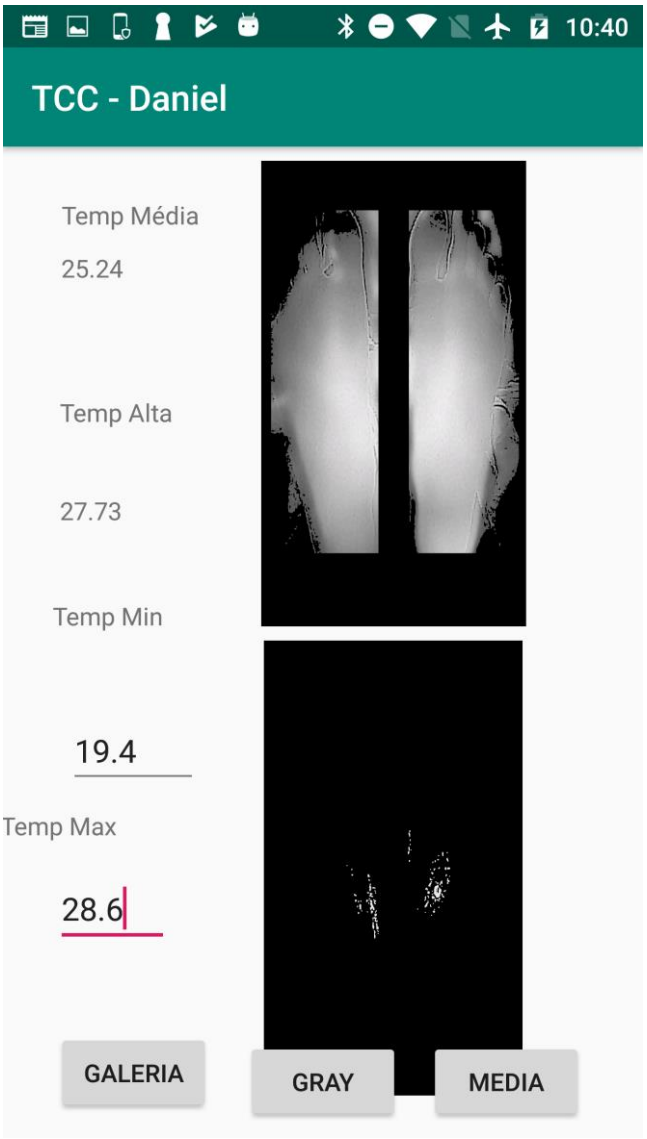


Figura 23 - Modelagem do pé saudável
Fonte: Autor, 2018

5 CONSIDERAÇÕES FINAIS

Os objetivos do trabalho foram alcançados com sucesso, pois foi possível identificar as áreas de maior temperatura na planta dos pés, assim como calcular o valor médio da temperatura dos pés e da temperatura da região mais quente. Também foi possível, ao comparar as regiões de temperatura dos dois pés, identificar se as regiões de maior temperatura aparecem em um único pé ou não, o que permite identificar se certa região apresenta uma temperatura que é comum esperada ou não, pois os dois pés tendem a apresentar a mesma distribuição de temperatura na planta dos pés.

Para o cálculo da temperatura, FLIR System disponibiliza uma biblioteca para desenvolvimento de aplicativos Android, porém, não foi possível adicionar essas funções durante o projeto. A implementação da biblioteca da FLIR permitirá o cálculo mais preciso da temperatura, não dependendo mais da aquisição da menor e da maior temperatura presentes na imagem.

A máscara é fixa permitiu a retirada de parte do ruído de fundo da imagem, porém, ainda é necessário realizar testes com as fotos tiradas, a fim de que a posição do pé se torne a melhor possível para retirar o ruído de fundo. Um modo de corrigir esse problema é a identificação automática dos pés, aplicando um filtro nas áreas que não são de interesse.

Dessa forma, ainda existem algumas modificações a serem realizadas até que o aplicativo esteja completamente funcional para um futuro teste em hospitais.

6 REFERÊNCIAS

Brasil. Ministério da Saúde. Secretaria de Atenção à Saúde. Departamento de Atenção Básica. Manual do pé diabético: estratégias para o cuidado da pessoa com doença crônica / Ministério da Saúde, Secretaria de Atenção à Saúde, Departamento de Atenção Básica. – Brasília: Ministério da Saúde, 2016.

SINGH, Nalini. Preventing Foot Ulcers in Patients with Diabetes. JAMA Network. Janeiro, 2005

N. C. SCHAPER. Prevention and management of foot problems in diabetes: A Summary Guidance for Daily Practice 2015, based on the IWGDF guidance documents. Science Direct. Fevereiro, 2017.

JM BOULTON, Andrew. The global burden of diabetic foot disease. The Lancet. Novembro, 2015.

FRAIWAM. Diabetic foot ulcer mobile detection system using smart phone thermal camera: a feasibility study; BioMedical Engineering OnLine. Outubro, 2017.

Siren. Disponível em: <https://siren.care/technology/> Acesso em: 09 jun 2019.

HERNANDEZ-CONTRERAS. Narrative review: Diabetic foot and infrared thermography. Science Direct. Setembro, 2016.

J. VAN NETTEN, Jaap. Infrared Thermal Imaging for Automated Detection of Diabetic Foot Complications. Journal of Diabetes Science and Technology. Volume 7, Edição 5, Setembro 2013.

PEREGRINA-BARRETO. Quantitative Estimation of Temperature Variations in Plantar Angiosomes: A Study Case for Diabetic Foot. Computational and Mathematical Methods in Medicine. Volume 2014, Fevereiro 2014.

Flir ONE. Disponível em: <https://www.flir.com/flirone> Acesso em: 10 jun 2019

OpenCV. Disponível em:< https://opencv.org/android/> Acesso em: 10 jun 2019

Lab Book Pages. Disponivel em:< http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html> Acesso em: 10 jun 2019

ANEXO

Código do programa:

.XML:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity" tools:layout_editor_absoluteY="81dp">

    <EditText
            android:layout_width="65dp"
            android:layout_height="wrap_content"
            android:ems="10"
            android:id="@+id/temp_max"
            android:inputType="number|numberSigned|numberDecimal"
            app:layout_constraintStart_toStartOf="parent"
android:layout_marginStart="8dp"
            app:layout_constraintEnd_toEndOf="parent"
android:layout_marginEnd="8dp"
            app:layout_constraintHorizontal_bias="0.075"
android:layout_marginTop="12dp"
            app:layout_constraintTop_toBottomOf="@+id/textView4"/>
    <Button
            android:text="Galeria"
            android:layout_width="wrap_content"
            android:onClick="openGaleria"
            android:layout_height="wrap_content"
            android:id="@+id/button"
            android:layout_marginBottom="13dp"
            app:layout_constraintBottom_toBottomOf="parent"
android:layout_marginStart="8dp"
            app:layout_constraintStart_toStartOf="parent"
android:layout_marginEnd="8dp"
            app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.084"
            app:layout_constraintTop_toBottomOf="@+id/textView4"/>
    <Button
            android:text="Gray"
            android:layout_width="wrap_content"
            android:onClick="gray"
            android:layout_height="wrap_content"
            android:id="@+id/button2"
            app:layout_constraintStart_toStartOf="parent"
            android:layout_marginStart="8dp"
app:layout_constraintEnd_toEndOf="parent" android:layout_marginEnd="8dp"
            android:layout_marginBottom="8dp"
app:layout_constraintBottom_toBottomOf="parent"/>

    <ImageView
            android:layout_width="231dp"
            android:layout_height="263dp" app:srcCompat="@mipmap/ic_launcher"
            android:id="@+id/imageView"
            app:layout_constraintEnd_toEndOf="parent"
            android:layout_marginEnd="18dp"
app:layout_constraintStart_toStartOf="parent"
            android:layout_marginStart="8dp" android:layout_marginTop="8dp"
app:layout_constraintTop_toTopOf="parent"
            android:layout_marginBottom="8dp"
app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintHorizontal_bias="0.935"
```

```
app:layout_constraintVertical_bias="0.0"
            app:layout_constraintStart_toEndOf="@+id/textView5"/>
    <ImageView
            android:layout_width="238dp"
            android:layout_height="257dp" app:srcCompat="@mipmap/ic_launcher"
            android:id="@+id/imageView2"
            android:layout_marginBottom="8dp"
            app:layout_constraintBottom_toBottomOf="parent"
android:layout_marginTop="8dp"
            app:layout_constraintVertical_bias="0.0"
            android:layout_marginEnd="20dp"
app:layout_constraintEnd_toEndOf="parent" android:layout_marginStart="8dp"
            app:layout_constraintStart_toStartOf="parent"
app:layout_constraintHorizontal_bias="0.984"
            app:layout_constraintStart_toEndOf="@+id/textView4"
app:layout_constraintTop_toBottomOf="@+id/imageView"/>
    <Button
            android:text="media"
            android:onClick="media"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/button4"
            app:layout_constraintEnd_toEndOf="parent"
            android:layout_marginEnd="8dp"
app:layout_constraintStart_toStartOf="parent"
            android:layout_marginStart="8dp" android:layout_marginBottom="8dp"
            app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintHorizontal_bias="0.902"/>
    <TextView
            android:layout_width="88dp"
            android:layout_height="45dp"
            android:id="@+id/textView"
            app:layout_constraintStart_toStartOf="parent"
            android:layout_marginStart="32dp"
app:layout_constraintEnd_toEndOf="parent" android:layout_marginEnd="8dp"
            app:layout_constraintHorizontal_bias="0.003"
android:layout_marginTop="8dp"
            app:layout_constraintTop_toBottomOf="@+id/textView5"
            android:text="Valor"/>
    <TextView
            android:layout_width="80dp"
            android:layout_height="51dp"
            android:id="@+id/textView3"
            android:text="Valor" app:layout_constraintStart_toStartOf="parent"
android:layout_marginStart="32dp"
            app:layout_constraintEnd_toEndOf="parent"
android:layout_marginEnd="8dp"
            app:layout_constraintHorizontal_bias="0.0"
android:layout_marginTop="196dp"
            app:layout_constraintTop_toTopOf="parent"
android:layout_marginBottom="8dp"
            app:layout_constraintVertical_bias="0.011"
app:layout_constraintBottom_toTopOf="@+id/textView2"/>
    <TextView
            android:text="Temp Alta"
            android:layout_width="79dp"
            android:layout_height="51dp"
            android:id="@+id/textView6"
            app:layout_constraintStart_toStartOf="parent"
android:layout_marginStart="32dp"
            app:layout_constraintEnd_toEndOf="parent"
android:layout_marginEnd="8dp"
            app:layout_constraintHorizontal_bias="0.0"
android:layout_marginTop="144dp"
            app:layout_constraintTop_toTopOf="parent"
android:layout_marginBottom="8dp"
            app:layout_constraintBottom_toTopOf="@+id/textView3"/>
    <TextView
```

```
            android:text="Temp Min"
            android:layout_width="99dp"
            android:layout_height="49dp"
            android:id="@+id/textView2"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintStart_toStartOf="parent"
            android:layout_marginStart="28dp"
app:layout_constraintEnd_toEndOf="parent" android:layout_marginEnd="8dp"
            app:layout_constraintHorizontal_bias="0.0"/>
    <TextView
            android:text="Temp Max"
            android:layout_width="104dp"
            android:layout_height="0dp"
            android:id="@+id/textView4"
            app:layout_constraintTop_toBottomOf="@+id/textView5"
app:layout_constraintEnd_toStartOf="@+id/imageView2"
            android:layout_marginBottom="103dp"
app:layout_constraintBottom_toTopOf="@+id/button"
            app:layout_constraintStart_toStartOf="parent"
            android:layout_marginStart="8dp" android:layout_marginEnd="8dp"/>
    <EditText
            android:layout_width="74dp"
            android:layout_height="wrap_content"
            android:inputType="number|numberSigned|numberDecimal"
            android:ems="10"
            android:id="@+id/temp_min"
            app:layout_constraintStart_toStartOf="parent"
            android:layout_marginStart="28dp"
app:layout_constraintEnd_toEndOf="parent" android:layout_marginEnd="8dp"
            android:layout_marginTop="12dp"
app:layout_constraintTop_toBottomOf="@+id/textView2"
            app:layout_constraintHorizontal_bias="0.033"
android:layout_marginBottom="8dp"
            app:layout_constraintBottom_toTopOf="@+id/textView4"/>
    <TextView
            android:text="Temp Média"
            android:layout_width="94dp"
            android:layout_height="0dp"
            android:id="@+id/textView5"
            android:layout_marginTop="29dp"
            app:layout_constraintEnd_toStartOf="@+id/imageView"
android:layout_marginBottom="323dp"
            app:layout_constraintBottom_toTopOf="@+id/textView4"
app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            android:layout_marginStart="33dp" android:layout_marginEnd="35dp"
            app:layout_constraintHorizontal_bias="0.0"/>

</android.support.constraint.ConstraintLayout>
```

Kotlin:

```
package com.example.daniel.novotesteopencv

import android.app.Activity
import android.app.VoiceInteractor
import android.content.Context
import android.content.Intent
import android.graphics.Bitmap
import android.graphics.BitmapFactory
import android.net.Uri
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.provider.MediaStore
import android.view.View
import android.widget.ImageView
import android.widget.Toast
```

```
import kotlinx.android.synthetic.main.activity_main.*
import org.opencv.android.*
import org.opencv.imgproc.Imgproc
import java.io.IOException
import java.net.URI
import android.util.Log
import org.opencv.core.*
import org.opencv.imgcodecs.Imgcodecs
import java.util.*
import android.R.attr.bitmap
import android.graphics.Color
import android.opengl.ETC1.getWidth
import android.opengl.ETC1.getHeight
import android.widget.EditText
import android.widget.TextView
import com.flir.flironesdk.FrameProcessor
import com.flir.flironesdk.Device
import com.flir.flironesdk.Frame
import com.flir.flironesdk.RenderedImage
import java.text.DecimalFormat
import java.text.NumberFormat
import kotlin.math.ceil


class MainActivity : AppCompatActivity(){

    lateinit var imageView: ImageView
    lateinit var uri_image: Uri
    lateinit var imageBitmap_foto_original: Bitmap
    lateinit var e: IOException
    lateinit var mat_origem_2gray: Mat
    lateinit var mat_destino_2gray: Mat
    lateinit var destino_bitmap_2gray: Bitmap
    lateinit var mascara: Bitmap
    lateinit var mascara_mat_shift: Mat
    lateinit var mascara_bitmap_original_resized: Bitmap
    lateinit var mascara_mat_original_resized: Mat
    lateinit var mascara_mat_sem_ruido: Mat
    lateinit var mascara_bitmap_sem_ruido: Bitmap
    //Gray
    lateinit var mat_origem_2gray_mascara: Mat
    lateinit var destino_bitmap_2gray_mascara: Bitmap
    lateinit var mat_destino_2gray_mascara: Mat
    lateinit var mat_otsu_mascara: Mat
    lateinit var destino_bitmap_2gray_otsu: Bitmap
    lateinit var destino_bitmap_2gray_TH_alto: Bitmap
    lateinit var mat_230_mascara: Mat
    lateinit var mat_otsu_mascara_intermed: Mat
    lateinit var mat_230_mascara_intermed: Mat
    //media
    var contagemPixel = 0L
    var valor_otsu_double = 0.0
    var valor_TH_alto_double = 0.0
    var valor_otsu_long = LongArray(10)
    var valor_TH_alto_long = LongArray(10)
    var contadorVetor = 0
    lateinit var valor_otsu_String: String
    lateinit var valor_TH_alto_String: String
    var temperatura_divisor = 0f
    lateinit var temperatura_max: EditText
    lateinit var temperatura_min: EditText


    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setTitle("TCC - Daniel")
        setContentView(R.layout.activity_main)
        imageView = findViewById(R.id.imageView) as ImageView
```

```
        mascara = BitmapFactory.decodeResource(resources, R.drawable.mascara)

        OpenCVLoader.initDebug()
    }

    fun openGaleria(v: View) {
        val myintent = Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI)
        startActivityForResult(myintent, 100)
    }

    override fun onActivityResult(requestCode: Int, resultCode: Int, data:
Intent?) {
        super.onActivityResult(requestCode, resultCode, data)

        if (requestCode == 100 && resultCode == Activity.RESULT_OK && data !=
null) {
            uri_image = data.data as Uri
            try {
                imageBitmap_foto_original =
MediaStore.Images.Media.getBitmap(this.contentResolver, uri_image)
            } catch (e: IOException) {
                e.printStackTrace()
            }

            imageView.setImageBitmap(imageBitmap_foto_original)
            imageView2.setImageBitmap(mascara)
        }
    }

    fun gray(v: View) {
        //Utilizando a mascara
        // transformando a imagem de mascara em Bitmap em escala de cinza
        mascara_mat_shift = Mat(mascara.width, mascara.height, CvType.CV_8UC1)
        Utils.bitmapToMat(mascara, mascara_mat_shift)

        //mudando o tamanho da imagem original em gray para aplicar a mascara
        mascara_bitmap_original_resized =
            Bitmap.createScaledBitmap(imageBitmap_foto_original, mascara.width,
mascara.height, true)
        mascara_mat_original_resized =
            Mat(mascara_bitmap_original_resized.width,
mascara_bitmap_original_resized.height, CvType.CV_8UC1)
        Utils.bitmapToMat(mascara_bitmap_original_resized,
mascara_mat_original_resized)

        //aplicando a mascara
        mascara_mat_sem_ruido =
            Mat(mascara_bitmap_original_resized.width,
mascara_bitmap_original_resized.height, CvType.CV_8UC1)
        Core.copyTo(mascara_mat_original_resized, mascara_mat_sem_ruido,
mascara_mat_shift)
        mascara_bitmap_sem_ruido = Bitmap.createBitmap(mascara.width,
mascara.height, Bitmap.Config.RGB_565)
        Utils.matToBitmap(mascara_mat_sem_ruido, mascara_bitmap_sem_ruido)
        imageView2.setImageBitmap(mascara_bitmap_sem_ruido)


        //grayscale
        mat_origem_2gray =
            Mat(mascara_bitmap_original_resized.width,
mascara_bitmap_original_resized.height, CvType.CV_8UC4)
        mat_destino_2gray =
            Mat(mascara_bitmap_original_resized.width,
mascara_bitmap_original_resized.height, CvType.CV_8UC1)
        destino_bitmap_2gray = Bitmap.createBitmap(
            mascara_bitmap_original_resized.width,
```

```
            mascara_bitmap_original_resized.height,
            Bitmap.Config.RGB_565
        )
        Utils.bitmapToMat(mascara_bitmap_original_resized, mat_origem_2gray)
        Imgproc.cvtColor(mat_origem_2gray, mat_destino_2gray,
Imgproc.COLOR_RGB2GRAY)
        Utils.matToBitmap(mat_destino_2gray, destino_bitmap_2gray)
        imageView.setImageBitmap(destino_bitmap_2gray)


        // Graycale da imagem com Mascara
        mat_origem_2gray_mascara = Mat(mascara_bitmap_sem_ruido.width,
mascara_bitmap_sem_ruido.height, CvType.CV_8UC4)
        mat_destino_2gray_mascara = Mat(mascara_bitmap_sem_ruido.width,
mascara_bitmap_sem_ruido.height, CvType.CV_8UC1)
        destino_bitmap_2gray_mascara =
            Bitmap.createBitmap(mascara_bitmap_sem_ruido.width,
mascara_bitmap_sem_ruido.height, Bitmap.Config.RGB_565)
        Utils.bitmapToMat(mascara_bitmap_sem_ruido, mat_origem_2gray_mascara)
        Imgproc.cvtColor(mat_origem_2gray_mascara, mat_destino_2gray_mascara,
Imgproc.COLOR_RGB2GRAY)
        Utils.matToBitmap(mat_destino_2gray_mascara, destino_bitmap_2gray_mascara)
        imageView.setImageBitmap(destino_bitmap_2gray_mascara)

        // otsu com mascara
        destino_bitmap_2gray_otsu =
            Bitmap.createBitmap(mascara_bitmap_sem_ruido.width,
mascara_bitmap_sem_ruido.height, Bitmap.Config.RGB_565)
        mat_otsu_mascara = Mat(mascara_bitmap_sem_ruido.width,
mascara_bitmap_sem_ruido.height, CvType.CV_8UC1)
        Imgproc.threshold(mat_destino_2gray_mascara, mat_otsu_mascara, 50.0,
255.0, Imgproc.THRESH_OTSU)

        //segundo TH com mascara
        destino_bitmap_2gray_TH_alto =
            Bitmap.createBitmap(mascara_bitmap_sem_ruido.width,
mascara_bitmap_sem_ruido.height, Bitmap.Config.RGB_565)
        mat_230_mascara = Mat(mascara_bitmap_sem_ruido.width,
mascara_bitmap_sem_ruido.height, CvType.CV_8UC1)
        Imgproc.threshold(mat_destino_2gray_mascara, mat_230_mascara, 230.0,
255.0, Imgproc.THRESH_BINARY)

        //limpeza da imagem
        //erosao
        Imgproc.erode(mat_otsu_mascara, mat_otsu_mascara, Mat())
        //Imgproc.erode(mat_230_mascara,mat_230_mascara, Mat())

        //dilatacao
        Imgproc.dilate(mat_otsu_mascara, mat_otsu_mascara, Mat())
        Imgproc.dilate(mat_230_mascara, mat_230_mascara, Mat())

        //usando Otsu e o segundo TH como mascaras para a imagem em gray
        mat_otsu_mascara_intermed = Mat(mat_otsu_mascara.width(),
mat_otsu_mascara.height(), CvType.CV_8UC1)
        mat_otsu_mascara_intermed = mat_otsu_mascara
        mat_230_mascara_intermed = Mat(mat_230_mascara.width(),
mat_230_mascara.height(), CvType.CV_8UC1)
        mat_230_mascara_intermed = mat_230_mascara

        Core.copyTo(mat_destino_2gray, mat_otsu_mascara,
mat_otsu_mascara_intermed)
        Core.copyTo(mat_destino_2gray, mat_230_mascara, mat_230_mascara_intermed)

        //mostrar as imagens
        //otsu
        Utils.matToBitmap(mat_otsu_mascara, destino_bitmap_2gray_otsu)
        imageView.setImageBitmap(destino_bitmap_2gray_otsu)
        //TH_alto
```

```
        Utils.matToBitmap(mat_230_mascara, destino_bitmap_2gray_TH_alto)
        imageView2.setImageBitmap(destino_bitmap_2gray_TH_alto)
    }

    fun media(v: View) {

        contadorVetor = 0
        valor_otsu_double = 0.0
        valor_TH_alto_double = 0.0
        for (x in 0..9) {
            valor_otsu_long[x] = 0L
        }
        for (x in 0..9) {
            valor_TH_alto_long[x] = 0L
        }
        contagemPixel = 0L

        temperatura_max = findViewById(R.id.temp_max)
        temperatura_min = findViewById(R.id.temp_min)

        temperatura_divisor = (temperatura_max.text.toString().toFloat()) -
(temperatura_min.text.toString().toFloat())

        //calculo do valor medio da imagem com OTSU
        for (y in 0 until mat_otsu_mascara.height()) {
            for (x in 0 until mat_otsu_mascara.width()) {
                var c = mat_otsu_mascara.get(y, x)
                if (c[0] == 0.0) {
                } else {
                    contagemPixel++
                    valor_otsu_long[contadorVetor] += c[0].toLong()

                }

                if (contagemPixel == (200000 * (contadorVetor + 1)).toLong()) {
                    contadorVetor++
                }
            }
        }
        for (x in 0..9) {
            valor_otsu_double += ((valor_otsu_long[x] / contagemPixel).toDouble()
* (temperatura_divisor / 255.0))
        }
        valor_otsu_double += (temperatura_min.text.toString().toDouble())

        contagemPixel = 0L
        contadorVetor = 0
        //calculo do valor medio da imagem com threshold mais alto
        for (y in 0 until mat_230_mascara.height()) {
            for (x in 0 until mat_230_mascara.width()) {
                var c = mat_230_mascara.get(y, x)
                if (c[0] == 0.0) {
                } else {
                    contagemPixel++
                    valor_TH_alto_long[contadorVetor] += c[0].toLong()

                }
                if (contagemPixel == (200000 * (contadorVetor + 1)).toLong()) {
                    contadorVetor++
                }
            }
        }
        for (x in 0..9) {
            valor_TH_alto_double += ((valor_TH_alto_long[x] /
contagemPixel).toDouble() * (temperatura_divisor / 255.0))
        }
        valor_TH_alto_double += (temperatura_min.text.toString().toDouble())
```

```
        //arredondamento
        valor_otsu_String = String.format("%.2f",valor_otsu_double)
        valor_TH_alto_String = String.format("%.2f",valor_TH_alto_double)
        textView.setText(valor_otsu_String)
        textView3.setText(valor_TH_alto_String)
    }
}
```