Detector de Veias com OpenCV

Posted on *19 de abril de 2022*

Neste estudo darei um exemplo de aplicação prática usando o OPENCV.

Este artigo faz parte do trabalho de pós graduação apresentado na disciplina de OpenCV, da UNINOVE.

RA dos Alunos:

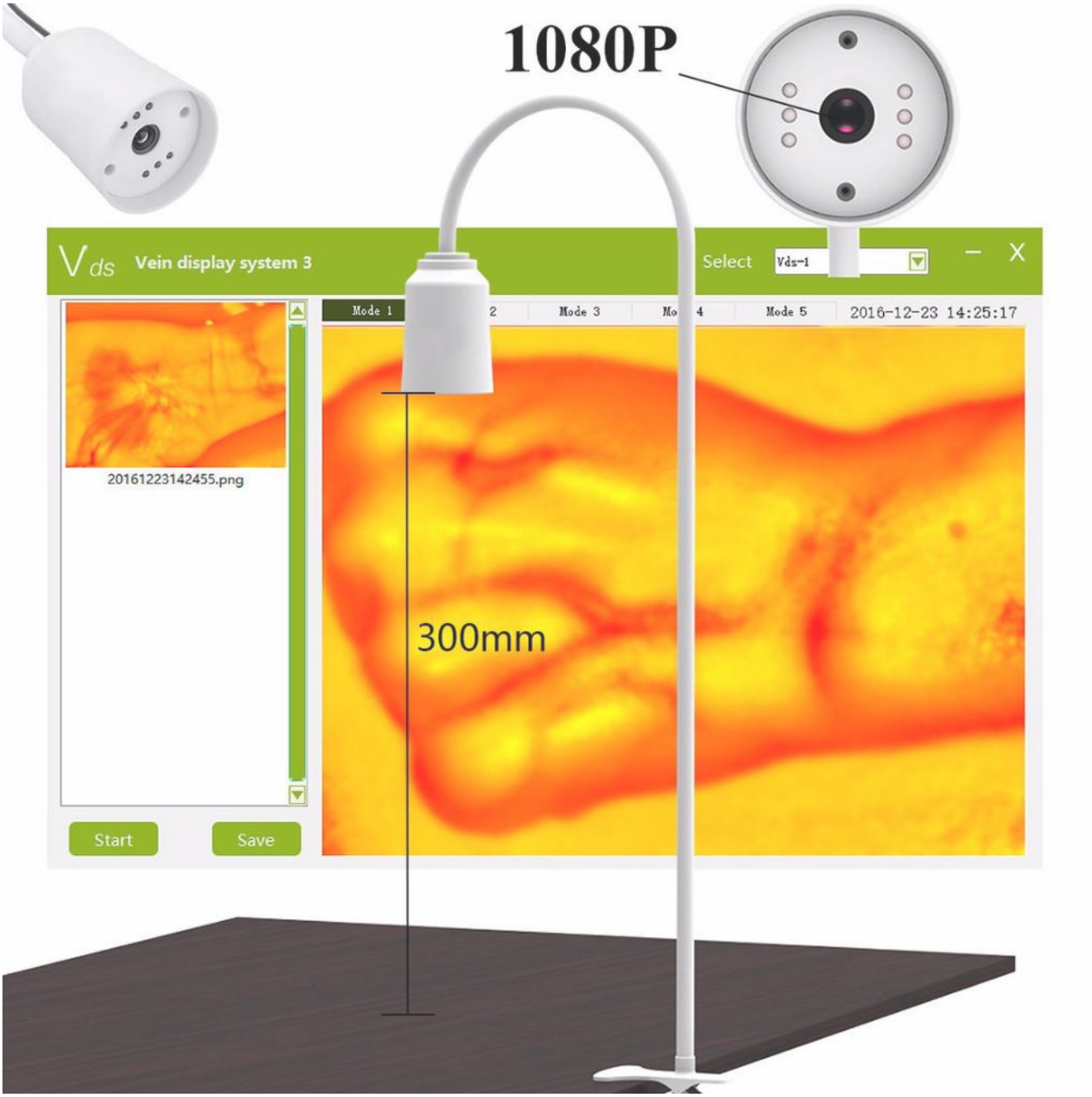
- 621200322 – Marcelo Maurin Martins
- 621201522 – Aland Montano
- 621202498 – Luciano Braga
- 621200985 – Jhone Fontenele
- 622137013- Douglas Campos

Equipamento:

Para realizar este estudo comprei o VEIN DISPLAY INSTRUMENT ZY-500.

Caracteristicas:

- 800-1000nm infrared
- IR 1080P
- Diametro 48mm



Equipamento ZY-500

Como ele Funciona

Ele é basicamente uma webcam USB 2.0, com resolução de 1080 pixel.

Este equipamento emite uma luz infra vermelha, que parcialmente penetra na pele (derme), e é refletida.

Porem em áreas onde existe muito sangue, a luz infra vermelha é absorvida.

Sabendo disso, fica facil identificar as veias proximas da pele.

Abaixo vemos um vídeo onde demonstro a visualização da camera sem nenhum tratamento.

Demonstração em Vídeo

Aqui apresentaremos detalhes do funcionamento da camera.

Neste exemplo mostro a imagem pura, sem processamento

Podemos ver em uma câmera normal, que as veias não se destacam.

Web cam normal

Porem podemos ver exatamente as veias na camera infravermelha.

Nesta imagem, sem filtros, podemos ver as veias claramente.

Imagem sem filtros

Inicio do processo de desenvolvimento

A imagem acima foi obtida, através do processamento do seguinte script PYTHON.

Iremos aplicar as técnicas aprendidas no curso, para avaliar a viabilidade destas na aplicação da solução. Lembrando que como qualquer processo cientifico, a experimentação empírica faz parte do processo.

E nem todas as técnicas serão utilizadas no projeto final.

Exemplo de código em Python

```
import cv2


device = 5

captura = cv2.VideoCapture(device)

while(1):
    ret, frame = captura.read()
    cv2.imshow("Camera Detector Veia", frame)

    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

captura.release()
cv2.destroyAllWindows()
```

A máquina em questão possui diversos devices (Cameras) acopladas, porem o número pode subir pois scripts em python tem o estranho habito de travar, e as vezes é necessário desligar o dispositivo, e religalo, o que causa um incremento no numero do device devido ao travamento do recurso.

Este travamento é temporario.

Vendo a banda RED

Separando a banda RED e verificando a melhora.

No fragmento de código abaixo, seleciono apenas a banda RED do RGB, onde analiso se há mudança significativa.

```
import cv2

device = 5

captura = cv2.VideoCapture(device)

while(1):
    ret, frame = captura.read()

    (B, G, R) = cv2.split(frame)
    cv2.imshow('RED',R)

    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

captura.release()
cv2.destroyAllWindows()
```

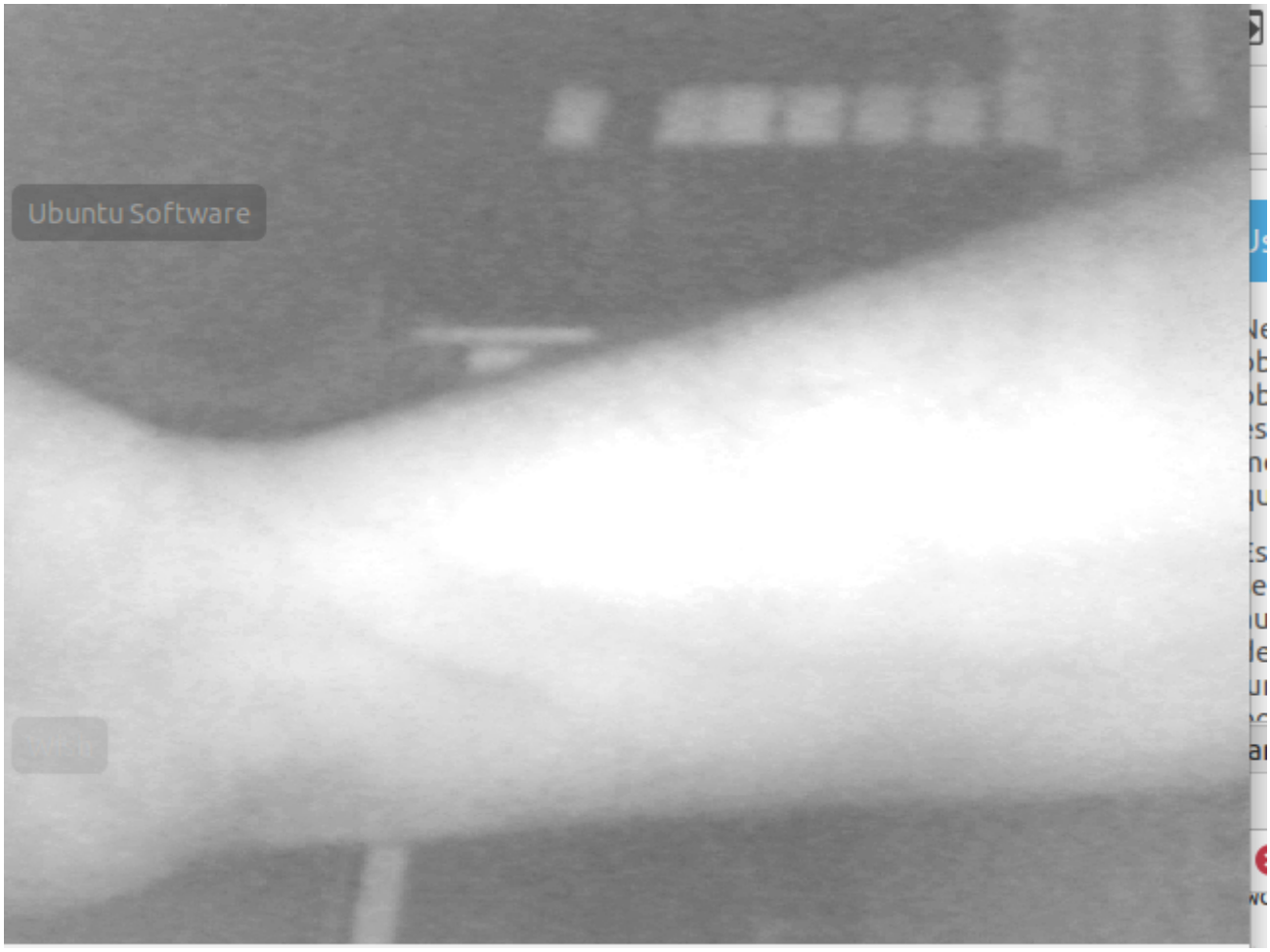
Sobre o meu ponto de vista, pegar apenas a banda R não melhorou, mas piorou um pouco a imagem.

Banda R do padrão RGB

Isolei tambem a Banda G (green)

Banda G do RGB

A banda B já piorou ainda mais



Banda B – Blue

Convertendo em Gray

Conforme a explicação do site abaixo:

https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Changing_ColorSpaces_RGB_HSV_HLS.php

Pudemos realizar a conversão das bandas em outros padrões.

```
import cv2
from PIL import Image
import numpy as np

device = 5

captura = cv2.VideoCapture(device)

while(1):

    ret, frame = captura.read()


    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    cv2.imshow('gray',gray)

    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

captura.release()
cv2.destroyAllWindows()
```

Pude perceber uma pequena melhora, neste padrão.

Padrão Gray

Analisando por HSV

HSV analisaremos a cor por suas caracteristicas de COR (H), S é a saturação, V é o brilho.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 15 16:00:05 2022

@author: mmm
"""
import cv2
from PIL import Image
import numpy as np

device = 5

captura = cv2.VideoCapture(device)

while(1):
    ret, frame = captura.read()



    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    #define range of blue color in HSV
    lower_blue = np.array([110,50,50])
    upper_blue = np.array([130,255,255])

    #low_red = np.array([161, 155, 84])
    #high_red = np.array([179, 255, 255])

    #low_green = np.array([25, 52, 72])
    #high_green = np.array([102, 255, 255])

    # Threshold the HSV image to get only blue colors
    mask = cv2.inRange(hsv, lower_blue , upper_blue )

    # Bitwise-AND mask and original image
    res = cv2.bitwise_and(frame,frame, mask= mask)

    cv2.imshow('frame',frame)
    cv2.imshow('mask',mask)
    cv2.imshow('res',res)

    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

captura.release()
cv2.destroyAllWindows()
```

Analisando a cor Blue, percebemos que muito pouco podemos aproveitar desta analise.

Analise de Saturação no Blue

Analise baseada em contorno

Agora iremos analisar a imagem baseada no seu contorno.

https://pyimagesearch.com/2021/04/28/opencv-color-spaces-cv2-cvtcolor/

No fragmento abaixo, tentamos aplicar o filtro gaussiano , conforme apresentado a seguir.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 15 16:00:05 2022

@author: mmm
"""
import cv2
#from PIL import Image
import numpy as np
import imutils


device = 2

captura = cv2.VideoCapture(device)



while(1):
    ret, frame = captura.read()


    cv2.imshow("Camera Detector Veia", frame)
    (B, G, R) = cv2.split(frame)

    # Our operations on the frame come here
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray1 = cv2.GaussianBlur(frame, (2, 2), 0)
    edged = cv2.Canny(frame, 30, 70)

    cv2.imshow('gray',gray)
    cv2.imshow('gray1',gray1)
    cv2.imshow('edged',edged)


    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

captura.release()
cv2.destroyAllWindows()
```

Podemos ver que não conseguimos ver os detalhes da veia, porem os contornos ficam aparentes.

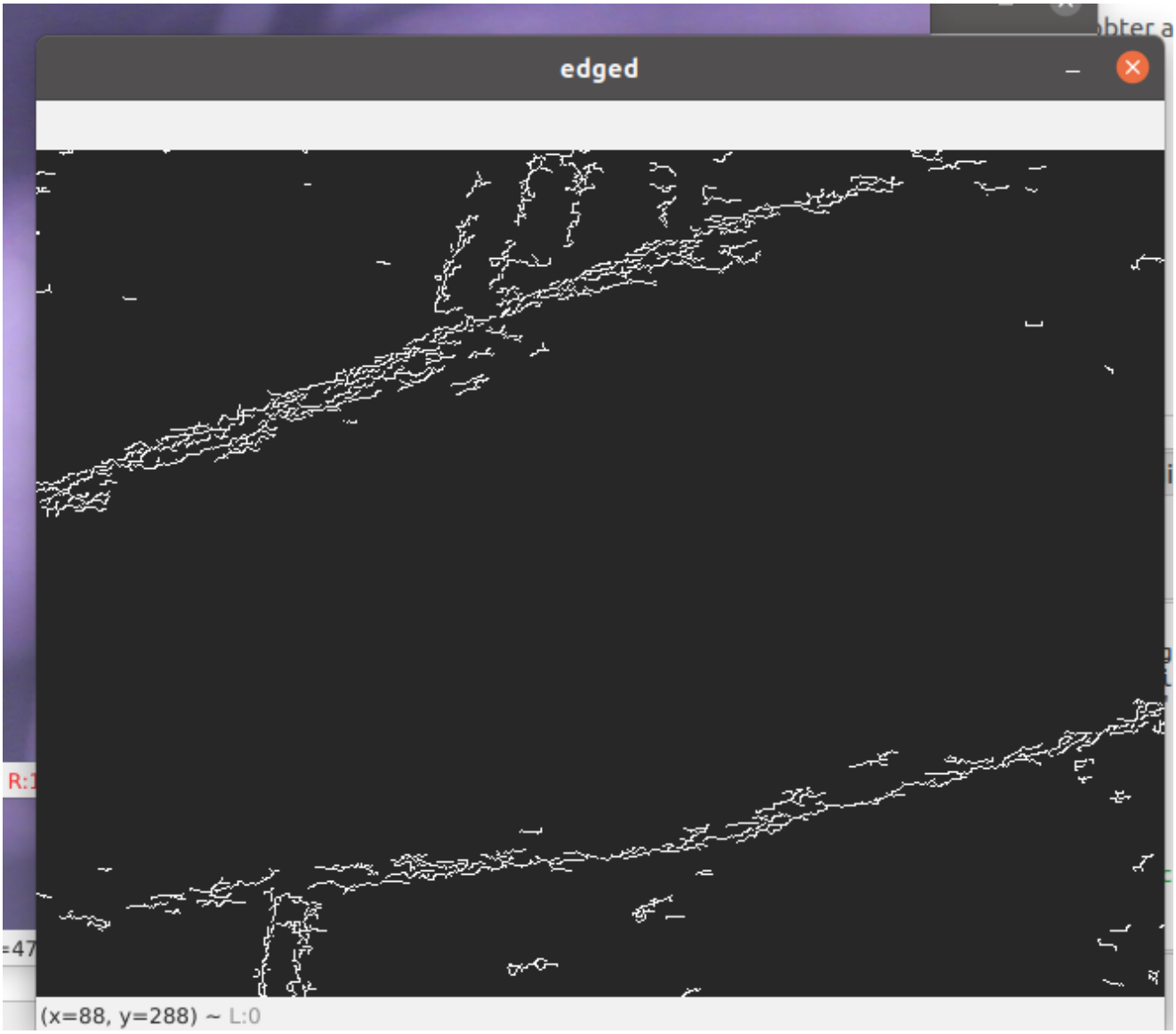


Imagem baseada em contorno.

Na forma que foi realizado, o contorno não se aplica de forma prática. Porem poderia ser aproveitado, na determinação do espaço do braço.

Retirando o fundo

Nesta técnica irei retirar o fundo para isolar o braço.

Usarei a limiarização da imagem para determinar o espaço do braço.

Podemos ver detalhes dessa técnica no artigo:

https://acervolima.com/python-tecnicas-de-limiarizacao-usando-opencv-conjunto-1-limiar-simples/

O nosso código, já possui a técnica contendo o resultado final empregado.

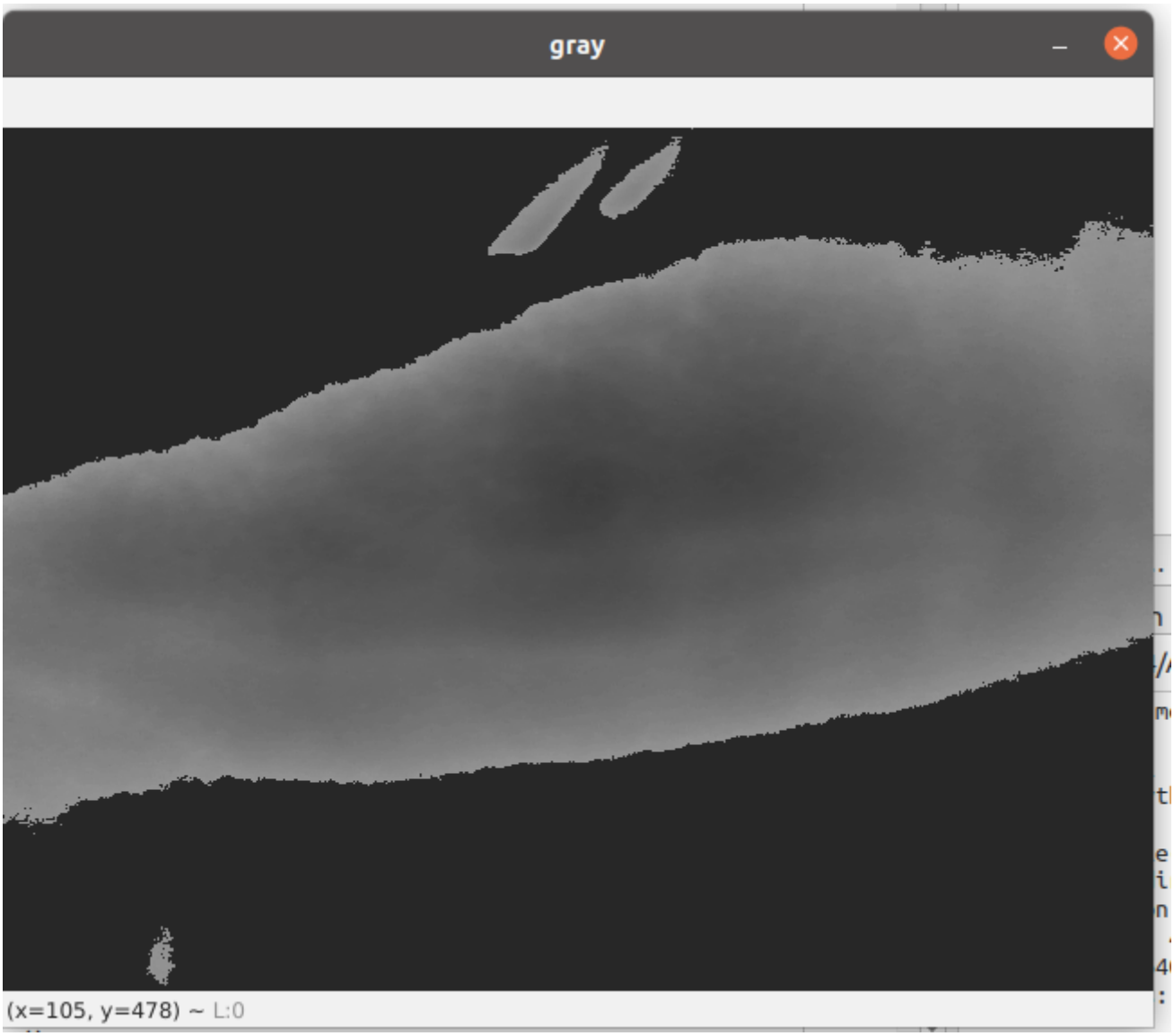
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 15 16:00:05 2022

@author: mmm
"""
import cv2
#from PIL import Image
import numpy as np
import imutils
import pip
import importlib, os
from matplotlib import pyplot as plt



#import Image
from PIL import Image, ImageChops


global fundo
global frame
global gray1
global edged
global ret

device = 2
captura = cv2.VideoCapture(device)



while(1):
    ret, frame = captura.read()

    cv2.imshow("Camera Detector Veia", frame)
    (B, G, R) = cv2.split(frame)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

num_lim = frame.shape[0]
num_col = frame.shape[1]
num_bandas = frame.shape[2]


gray2= np.zeros((num_lim , num_col), dtype="uint8")
gray2= B+G+R//3

#[num_lim, num_col, num_bandas] = frame.shape


dimensions = frame.shape
```

```
print('Image Dimension:',dimensions)
print('Image Height :',num_lim)
print('Image width:',num_col)
print('Image Channels:',num_bandas)

thresh = 135


fundo2 = cv2.threshold(gray,thresh,255,cv2.THRESH_BINARY)[1] #imagem limiral
izada
#[thresh, fundo2] = cv2.threshold(gray, thresh, 255, cv2.THRESH_OTSU)
cv2.imshow("fundo2",fundo2)


img_filtro= np.zeros((num_lim , num_col, num_bandas), dtype="uint8")

for l in range(num_lim):
   for c in range(num_col):
        img_filtro[l,c] = ((fundo2[l,c]) & (frame[l,c] ^ fundo2[l,c]))
        #img_filtro[l,c] =  (frame[l,c] ^ fundo2[l,c])

#img_filtro = ImageChops.difference(fundo, frame)

cv2.imshow("filtrado", img_filtro)

while(1):
    #gaus = cv2.GaussianBlur(img_filtro, (3, 3), 0)
    #edged = cv2.Canny(img_filtro, 40, 103)

    #janela = np.ones((3,3),np.float32)*-1
    #palta = cv2.filter2D(img_filtro,-6,janela)


    #gaus=cv2.GaussianBlur(img_filtro,(0,0),5)
    #palta2=cv2.Scharr(img_filtro, ddepth=-1, dx=1, dy=0, scale=1, borderTyp
e=cv2.BORDER_DEFAULT)


    #filtro = ImageChops.difference(fundo,gray)
    #filtro = ImageChops.logical_xor(fundo,gray)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break
RGB = cv2.cvtColor(img_filtro, cv2.COLOR_BGR2RGB)
grayold = gray
gray = cv2.cvtColor(RGB, cv2.COLOR_RGB2GRAY)
#gray = grayold

while(1):
    cv2.imshow('gray',gray)


    #http://www.lps.usp.br/hae/apostila/filtconv-ead.pdf
    #Calculo de gradiente usando Scharr
    schar=cv2.Scharr(gray, ddepth=-1, dx=0, dy=1, scale=1, borderType=cv2.BO
RDER_DEFAULT)
    #cv2.imshow("gaus", gaus)
    #cv2.imshow("Canny", edged)
```

```
        cv2.imshow("Scharr", schar)
        sobel=cv2.Sobel(schar,-2,1,1)

        kernel = np.ones((5,5),np.uint8)
        dilation = cv2.dilate(gray,kernel,iterations = 1)

        maurin= np.zeros((num_lim , num_col), dtype="uint8")
        for l in range(num_lim):
            for c in range(num_col):
                if(dilation[l,c]>=74 and dilation[l,c]<=76):
                    maurin[l,c] = 0
                else:
                    maurin[l,c] = dilation[l,c]

        thresh = 80
        binimg=np.zeros((num_lim,num_col), dtype="uint8")


        binimg=cv2.threshold(gray2, thresh, 255, cv2.THRESH_TOZERO)[1]


        #cv2.imshow('gaus',gaus)
        #cv2.imshow('edged',edged)
        cv2.imshow('sobel',sobel)
        cv2.imshow('dilation',dilation)
        cv2.imshow('maurin:',maurin)
        cv2.imshow('bin',binimg)



        k = cv2.waitKey(30) & 0xff
        if k == 27:
            break

    captura.release()
    cv2.destroyAllWindows()
```

Neste ultimo fragmento, iremos retirar o fundo do braço, e trabalhar com a imagem para dar destaque.

O resultado final é o que se apresenta.



Na imagem acima, podemos perceber que retiramos o braço dos demais objetos da imagem.

Tambem tratamos a imagem em tons de cinza, que permite posteriormente um trato mais simples da imagem.

Qual a vantagem da imagem acima da original. Primeiramente, a imagem original apesar de ser bem visivel a veia para nós, não conseguimos determinar um tom unico, para apresentar como veias. Na imagem processada, o tom da veia, esta mais distinto da imagem.

O que foi feito

Inicialmente capturamos a camera, através de um device.

Em seguida armazenamos a imagem na variavel frame.

Convertemos a imagem em cinza, através do comando:

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Agora, feita a primeira conversão, iremos pegar as caracteristicas da imagem, através da propriedade shape.

As propriedades de Altura (Height), Width(Largura) e channel (numero de bandas).

Agora iremos separar o braço do fundo, através da Limiarização:

```
thresh = 127

#fundo2 = cv2.threshold(gray,thresh,255,cv2.THRESH_BINARY)[1] #imagem limira
izada
[thresh, fundo2] = cv2.threshold(gray, thresh, 255, cv2.THRESH_OTSU)
```

O processamento da limiarização foi armazenado em fundo2.

O fundo é comparado bit a bit, fazendo um processamento, onde deixamos apenas com a imagem o braço.

```
img_filtro[l,c] = ((fundo2[l,c]) & (frame[l,c] ^ fundo2[l,c]))
```

No exemplo acima, aplicamos um xor entre a imagem e o fundo,

Extraída e processada da imagem original.

O resultado desta pesquisa bit a bit, resultou de uma imagem copia da original.

Servindo apenas para uso futuro.

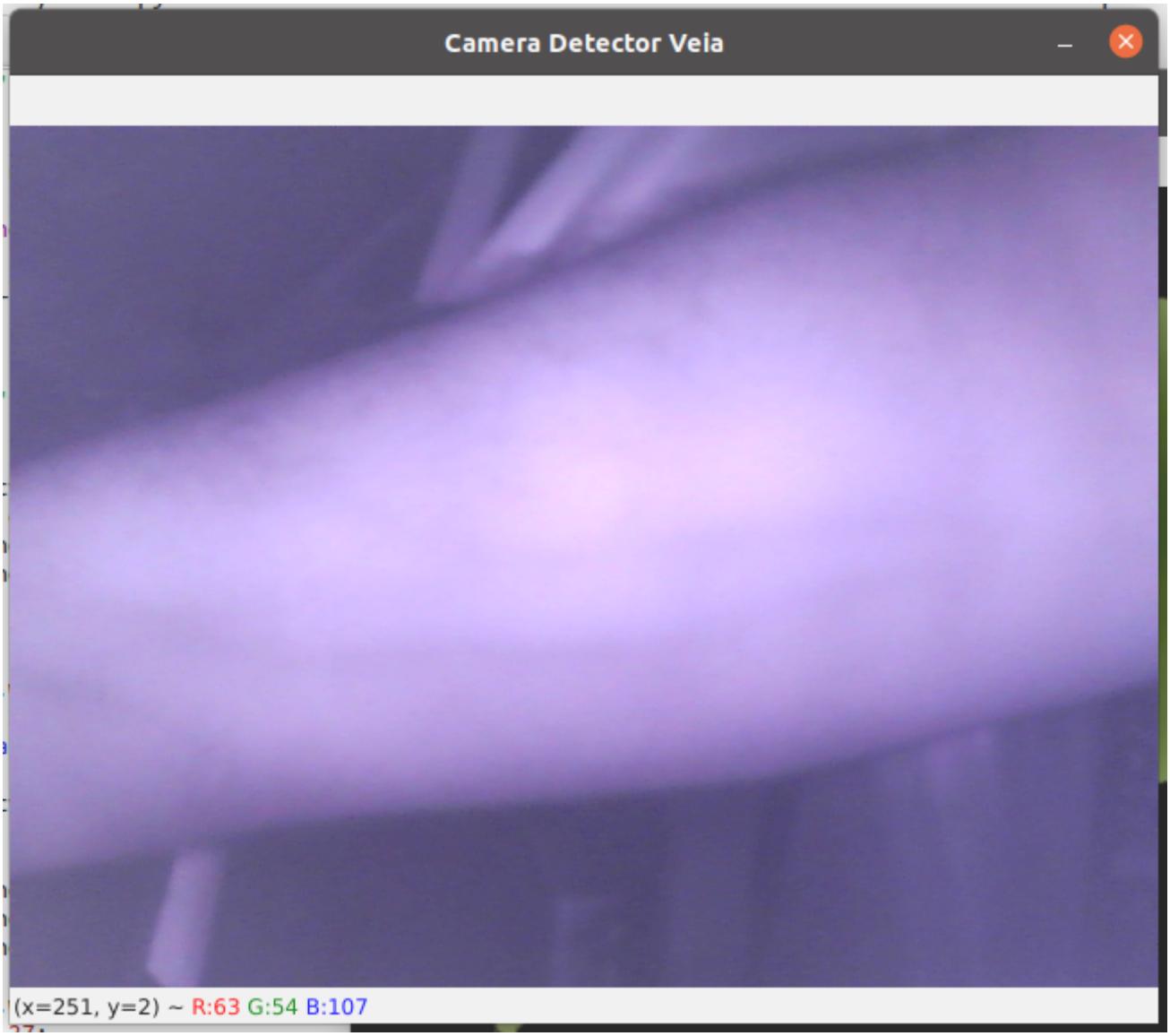
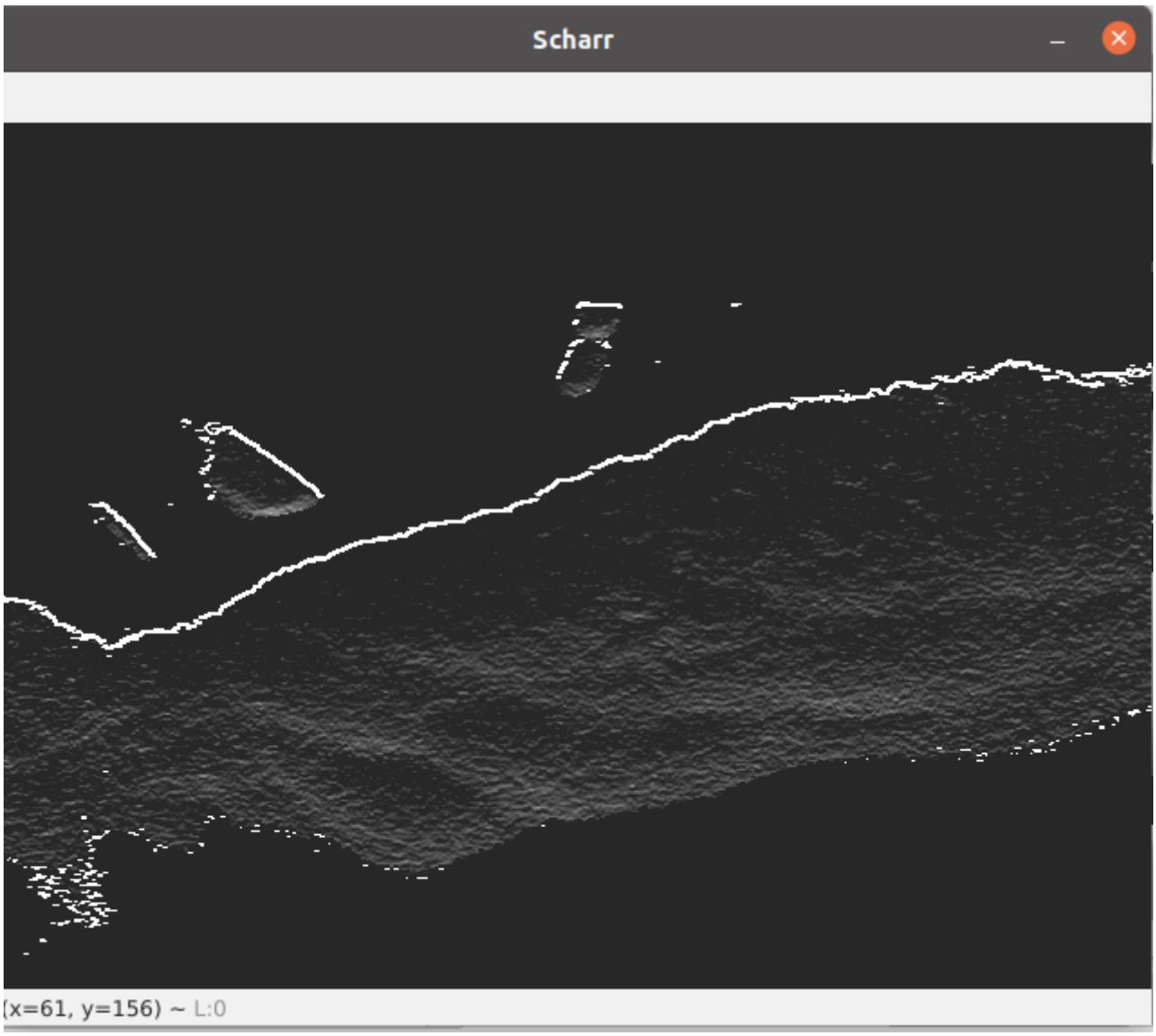


imagem original

Filtro Scharr

É um filtro que gera um gradiente 3D em imagens 2D, dando aspecto de relevo.



Filtro Scharr

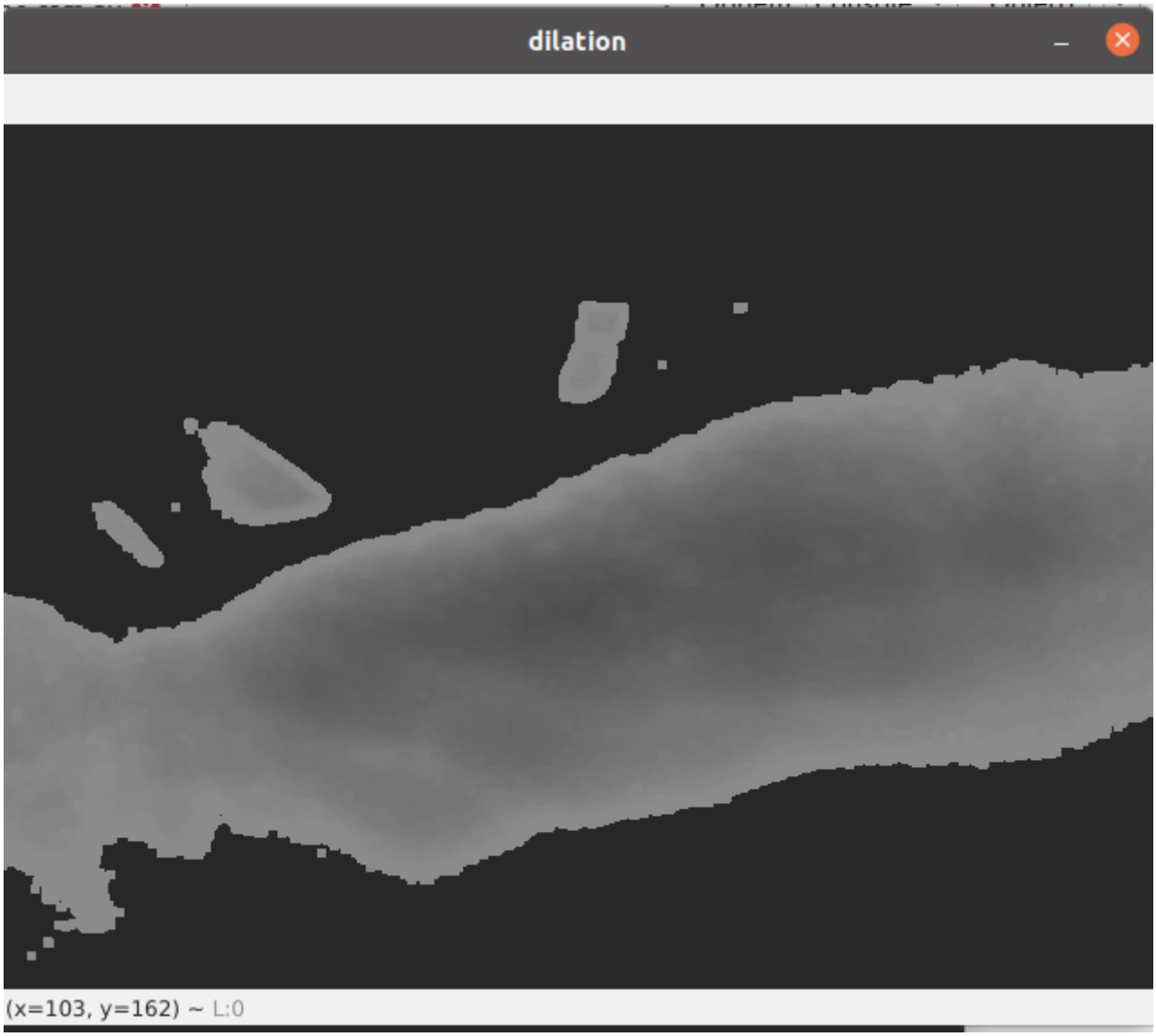
Podemos perceber que o filtro melhorou bastante a impressão da veia.

A chamada do scharr, fica conforme fragmento abaixo:

```
schar=cv2.Scharr(gray, ddepth=-1, dx=0, dy=1, scale=1, borderType=cv2.BORDER
_DEFAULT)
```

Filtro de dilatação

O Filtro de dilatação, torna as marcas das veias mais marcantes.



Dilatation

Sua sintaxe é conforme apresentada.

Dando um incremento a visão da veia.

```
dilation = cv2.dilate(gray,kernel,iterations = 1)
```

Por fim apresentamos o vídeo final.

Vídeo detector de veias

Conclusão

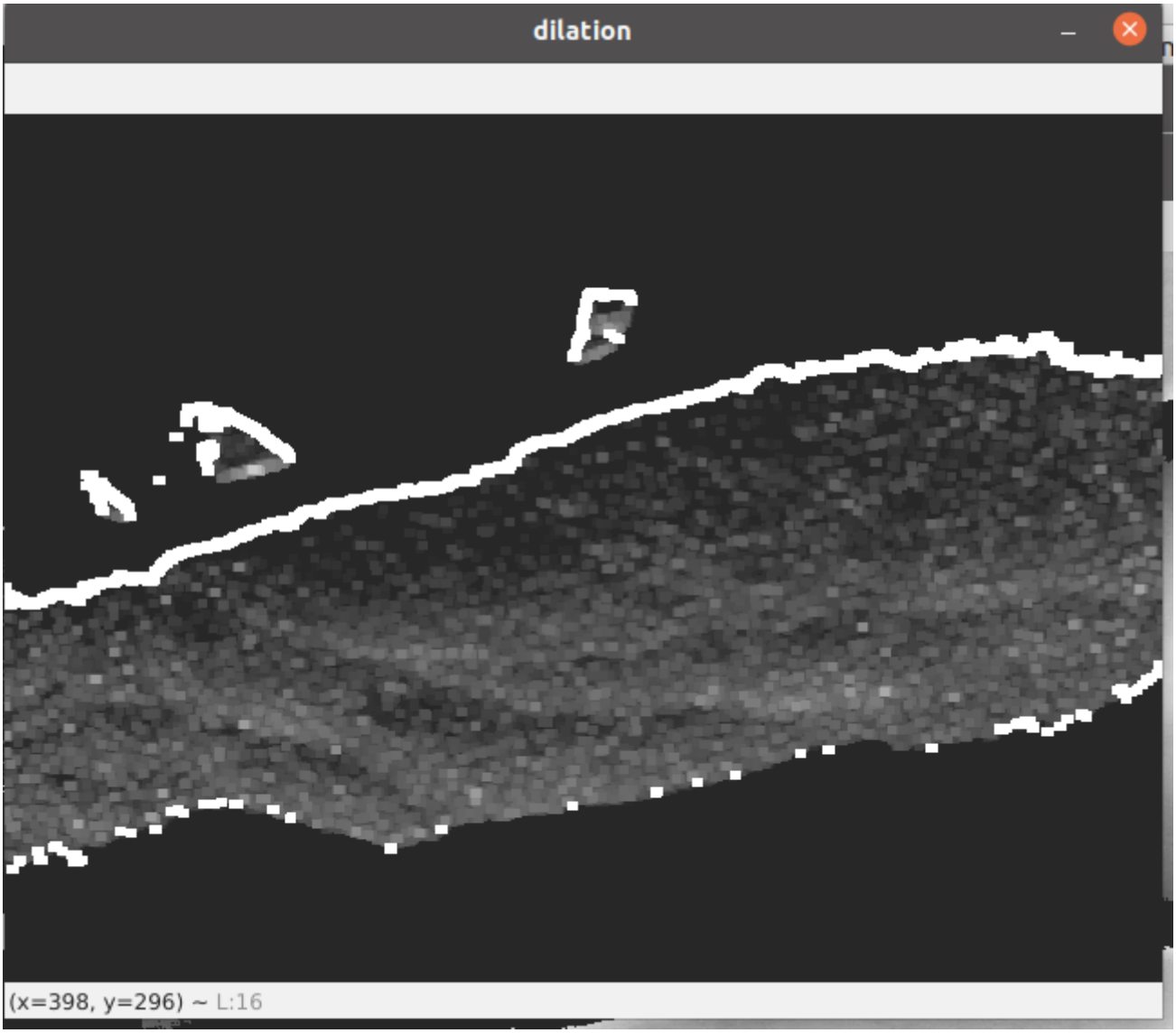
Este estudo demonstra que através de técnicas de manipulação de imagem, sem o uso de redes neurais conseguimos trabalhar com imagens, realçando e separando partes que desejamos trabalhar.

Outra parte importante, é a associação de operações:

```
img_filtro[l,c] = ((fundo2[l,c]) & (frame[l,c] ^ fundo2[l,c]))
```

Onde associamos o fundo da imagem com a imagem, a fim da extração do fundo e aplicação do formato RGB, desassociado ao fundo.

Concomitante com o uso do Dilate (filtro de dilatação) associado ao Scharr, apresentou ótimos resultados, conforme apresentado a seguir.



Associação do Dilate associado ao Scharr, apresentou ótimo resultado.

Tendo como resultado final o código gerado:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 15 16:00:05 2022

@author: mmm
"""
import cv2
#from PIL import Image
import numpy as np
import imutils
import pip
import importlib, os
from matplotlib import pyplot as plt



#import Image
from PIL import Image, ImageChops


global fundo
global frame
global gray1
global edged
global ret

device = 2
captura = cv2.VideoCapture(device)



while(1):
    ret, frame = captura.read()

    cv2.imshow("Camera Detector Veia", frame)
    (B, G, R) = cv2.split(frame)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

num_lim = frame.shape[0]
num_col = frame.shape[1]
num_bandas = frame.shape[2]


gray2= np.zeros((num_lim , num_col), dtype="uint8")
gray2= B+G+R//3

#[num_lim, num_col, num_bandas] = frame.shape


dimensions = frame.shape
```

```
print('Image Dimension:',dimensions)
print('Image Height :',num_lim)
print('Image width:',num_col)
print('Image Channels:',num_bandas)

thresh = 135


fundo2 = cv2.threshold(gray,thresh,255,cv2.THRESH_BINARY)[1] #imagem limiral
izada
#[thresh, fundo2] = cv2.threshold(gray, thresh, 255, cv2.THRESH_OTSU)
cv2.imshow("fundo2",fundo2)


img_filtro= np.zeros((num_lim , num_col, num_bandas), dtype="uint8")

for l in range(num_lim):
   for c in range(num_col):
        img_filtro[l,c] = ((fundo2[l,c]) & (frame[l,c] ^ fundo2[l,c]))
        #img_filtro[l,c] =  (frame[l,c] ^ fundo2[l,c])

#img_filtro = ImageChops.difference(fundo, frame)

cv2.imshow("filtrado", img_filtro)

while(1):
    #gaus = cv2.GaussianBlur(img_filtro, (3, 3), 0)
    #edged = cv2.Canny(img_filtro, 40, 103)

    #janela = np.ones((3,3),np.float32)*-1
    #palta = cv2.filter2D(img_filtro,-6,janela)


    #gaus=cv2.GaussianBlur(img_filtro,(0,0),5)
    #palta2=cv2.Scharr(img_filtro, ddepth=-1, dx=1, dy=0, scale=1, borderTyp
e=cv2.BORDER_DEFAULT)


    #filtro = ImageChops.difference(fundo,gray)
    #filtro = ImageChops.logical_xor(fundo,gray)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break
RGB = cv2.cvtColor(img_filtro, cv2.COLOR_BGR2RGB)
grayold = gray
gray = cv2.cvtColor(RGB, cv2.COLOR_RGB2GRAY)
#gray = grayold

while(1):
    cv2.imshow('gray',gray)


    #http://www.lps.usp.br/hae/apostila/filtconv-ead.pdf
    #Calculo de gradiente usando Scharr
    schar=cv2.Scharr(gray, ddepth=-1, dx=0, dy=1, scale=1, borderType=cv2.BO
RDER_DEFAULT)
    #cv2.imshow("gaus", gaus)
    #cv2.imshow("Canny", edged)
```

```
        cv2.imshow("Scharr", schar)
        sobel=cv2.Sobel(schar,-2,1,1)

        kernel = np.ones((5,5),np.uint8)
        #dilation = cv2.dilate(gray,kernel,iterations = 1)
        dilation = cv2.dilate(schar,kernel,iterations = 1)

        maurin= np.zeros((num_lim , num_col), dtype="uint8")
        for l in range(num_lim):
            for c in range(num_col):
                if(dilation[l,c]>=74 and dilation[l,c]<=76):
                    maurin[l,c] = 0
                else:
                    maurin[l,c] = dilation[l,c]

        thresh = 80
        binimg=np.zeros((num_lim,num_col), dtype="uint8")


        binimg=cv2.threshold(gray2, thresh, 255, cv2.THRESH_TOZERO)[1]


        #cv2.imshow('gaus',gaus)
        #cv2.imshow('edged',edged)
        cv2.imshow('sobel',sobel)
        cv2.imshow('dilation',dilation)
        cv2.imshow('maurin:',maurin)
        cv2.imshow('bin',binimg)



        k = cv2.waitKey(30) & 0xff
        if k == 27:
            break

    captura.release()
    cv2.destroyAllWindows()
```

Neste trabalho, podemos verificar que ainda é longo e árduo o trabalho que separa o pré processamento da imagem, bem como não se trata de um trabalho acabado. Há muitas técnicas aqui não empregadas, que poderiam ser ajustadas.

Porem a analise que foi feita em várias das técnicas apresentadas durante o curso, tiveram resultados significativos na identificação e destaque das veias.

Agradecimento especial ao Professor **Sidnei Alves de Araujo**, da **UNINOVE**, que incentivou e permitiu a execução de trabalho especial.

This entry was posted in *IA*, *OpenCV*, *pós graduação*, *Python* and tagged *detecção*, *opencv*, *python*, *veias*. Bookmark the *permalink*.

← Novos equipamentos Verificando se serviço esta online →

Pesquisar …

POSTS RECENTES

DOCTOR 0.2.12

MNote 2.17 Linux

MNote2.17 Changelog

Validação de Anuncio REVIVE

Implementações de 14/10

ARQUIVOS

novembro 2022

outubro 2022

setembro 2022

agosto 2022

julho 2022

junho 2022

maio 2022

abril 2022

março 2022

fevereiro 2022

janeiro 2022

dezembro 2021

novembro 2021

outubro 2021

setembro 2021

agosto 2021

julho 2021

junho 2021

maio 2021

abril 2021

março 2021

fevereiro 2021

janeiro 2021

dezembro 2020

novembro 2020

outubro 2020

setembro 2020

agosto 2020

março 2020

fevereiro 2020

novembro 2019

outubro 2019

setembro 2019

agosto 2019

julho 2019

junho 2019

maio 2019

abril 2019

março 2019

CATEGORIAS

arduino

Balança

Banco de dados

Biologia Celular

Blog

C/C++

casa

CUDA

Delphi

Dicas

DJANGO

docker

DOCTO

DOCTO

Eletricidade

ESCPOS

ETIQUETAS

Fila

Fisica I

GIT

IA

Java

Jira

Lazarus

mercurial

microbiologia

MNote2

MongoDB

MONGODB

Moodle

MYSQL

Mysql

OpenCV

Oracle

pacotes

PHP

pós graduação

Postgres

POSTGRES

Postgres

Programação

Projetos

Python

Química

R

Relógio

RFID

robotinics

SAAS

Sem categoria

Sharepoint

Shell Script

Sistemas Analogicos I

Sistemas Biomedicos

Solidworks

SQLite

SQLite

srvCP

srvmonitor2

srvOuve

SSC

Temperatura

treinamentos

Varejo

wordpress

Yocto Project

META

Cadastre-se

Acessar

Feed de posts

Feed de comentários

WordPress.org

MEU CV

VEJA NOSSOS PRODUTOS

Nossa lojinha esta começando agora, mas você já pode começar a comprar.

marcelomaurinmartins@gmail.com +55 16 981434112

Rua Jorge Felipe 231 Cidade
Nova Jardinopolis - SP