UNIVERSIDADE FEDERAL DE SERGIPE

MÁRCIO HENRIQUE MATOS DE FREITAS

DISCIPLINA: PROCESSAMENTO DE IMAGENS

São Cristóvão-SE
2023

MÁRCIO HENRIQUE MATOS DE FREITAS

DISCIPLINA: PROCESSAMENTO DE IMAGENS

Atividade pontuada para apresentar na disciplina de Processamentos de Imagens do Departamento de Computação da Universidade Federal de Sergipe.

Prof. Dr. Leonardo Nogueira Matos .

São Cristóvão-SE
2023

Sumário

1 Introdução

Na atividade pontuada, foi colocado um problema de filtro , logo foi necessário desenvolver em Python e Qt designer uma aplicação onde fosse possível obter uma imagem na webcam do computador / Notebook e aplicar os filtros gaussianos.

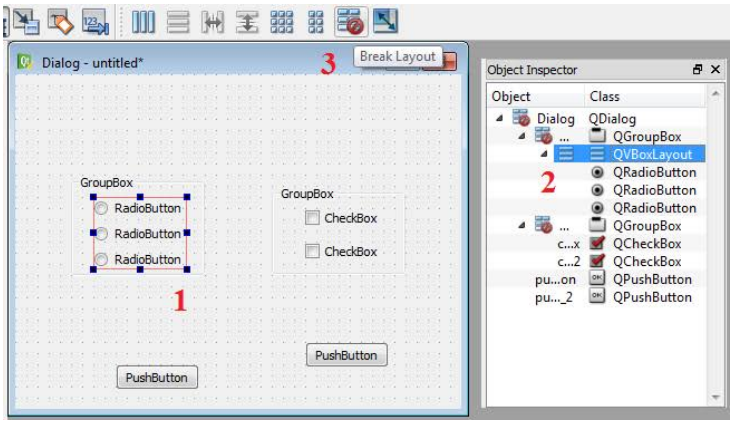
e realizar uma comparação entre as imagens sem filtro ,com filtro 1D e 2D.

2 Qt Designer

Criando em meados de 1995 o Qt designer, ou mais conhecidocomo Qt Creator é uma ambiente de desenvolvimento integrado, mais popularmente Ide. Permitindo que o usuário possa desenvolver aplicativos e outros recursos para plataformas múltiplas, oferecendo muita facilidade e robustez para o seu código. Muitos vêem o Qt designer como uma simples Ide para construções de tela principalmente quando se fala em design, porém essa software permite que, através das bibliotecas QT GUI e plugins, você possa programar em diversas linguagens como C, C++, Java, Ruby e Python.

A ideia do QT Creator é ser não só um compilador ou um debugger, mas ser uma plataforma completa para programação, criação e design

2.1 Interface



O Qt Quick é um arcabouço desenvolvido e mantido pelo Projeto Qt, que se tornou disponível a partir do Qt 4.7. O Qt Quick é baseado em uma linguagem declarativa que permite a construcão de interfaces dinâmicas, com transições fluidas e efeitos, comuns atualmente em dispositivos móveis. Essa linguagem é chamada de QML (Qt Modeling Language).

Dentre os tipos mais usuais podemos citar:
Elementos visuais: Rectangle, Gradient, animatedImage Elementos de interação:

MouseArea

Flickable
Elementos de estado:
PropertyChanges

StateChangeScript

Elementos de transição:
Transition
SequentialAnimation

Outros: manipulação de dados, vistas, posição, etc.
É possível criar as telas de interface, seja login, de Imagens etc , que é gerado um XML e é possível converter em código python,Java, C++.

3 Linguagem de Programação

Como já foi citado, o Qt designer pode utilizar algumas linguagens de Programação, entre elas Python. Que foi adotada como uma das linguagens na resolução de Problemas na disciplina.

A partir da documentação do Qt Creator foi possível criar telas direto na linguagem Python.

3.1 Pacotes

Alguns pacotes da linguagem de programação python foram utilizadas para construção e resolução do problema proposto.

3.1.1 Numpy e Opencv

Opencv é uma biblioteca de visão computacional e aprendizado de máquina de código aberto. Esta biblioteca é capaz de processar imagens e vídeos em tempo real, ao mesmo tempo em que possui recursos analíticos.

Já o Numpy , é uma biblioteca para a linguagem Python com funções para se trabalhar com computação numérica. Algumas das funções matemáticas utilizadas em processamento de

imagens, machine learning, inteligência artificial vêem prontas e isso já ajuda na construção do código.

4 Filtros

4.1 Filtro Gaussiano

O filtro gaussiano é um tipo de filtro linear utilizado em processamento de imagens digitais para suavizar a imagem e reduzir o ruído. Ele é chamado de gaussiano porque sua função de resposta ao impulso é uma distribuição normal, que também é conhecida como distribuição gaussiana.

A convolução da imagem com o filtro gaussiano pode ser realizada no domínio espacial ou no domínio da frequência. No domínio espacial, o filtro gaussiano é definido como:

$$g(x, y) = f(x, y) * G(x, y)$$

5 Código em python

```
1
2  def filtro_convolucao_gaussiana(self, imagem, indice_valor):
3          # Aqui voc  pode colocar seu c digo para aplicar o filtro
               2D
4          tamanho_mascara = 2*indice_valor + 3
5          mascara = np.zeros((tamanho_mascara, tamanho_mascara))
6          sigma = 1.0
7          for x in range(-tamanho_mascara//2, tamanho_mascara//2+1):
8              for y in range(-tamanho_mascara//2, tamanho_mascara
                   //2+1):
9                  r = np.sqrt(x**2 + y**2)
10                 mascara[x+tamanho_mascara//2, y+tamanho_mascara//2]
                       = (1.0 / (2.0 * np.pi * sigma**2)) * np.exp(-r
                       **2 / (2.0 * sigma**2))
11         mascara = mascara / mascara.sum()
```

```
12
13      # Aplica a convolu  o
14          imagem_convoluida = np.zeros_like(imagem)
15          imagem_padded = cv2.copyMakeBorder(imagem, tamanho_mascara
               //2, tamanho_mascara//2, tamanho_mascara//2,
               tamanho_mascara//2, cv2.BORDER_CONSTANT)
16          for i in range(tamanho_mascara//2, imagem_padded.shape[0]-
               tamanho_mascara//2):
17              for j in range(tamanho_mascara//2, imagem_padded.shape
                   [1]-tamanho_mascara//2):
18                  imagem_convoluida[i-tamanho_mascara//2, j-
                       tamanho_mascara//2] = np.sum(mascara*
                       imagem_padded[i-tamanho_mascara//2:i+
                       tamanho_mascara//2+1, j-tamanho_mascara//2:j+
                       tamanho_mascara//2+1])
19
20          # Retorna a imagem filtrada
21          return imagem_convoluida
```

```
1       def filtro_gaussiano_quadrado(self, imagem, indice_valor):
2           # Aqui voc  pode colocar seu c digo para aplicar o filtro
                1D
3           tamanho_mascara = 2*indice_valor + 3
4           mascara_sigma = 1.5
5           mascara = np.zeros((tamanho_mascara, tamanho_mascara))
6           for i in range(tamanho_mascara):
7               for j in range(tamanho_mascara):
8                   mascara[i, j] = np.exp(-((i - tamanho_mascara//2)
                       **2 + (j - tamanho_mascara//2)**2)/(2*
                       mascara_sigma**2))
9           mascara /= mascara.sum()
10
11          # Aplica a convolu  o do mascara com a imagem
12          filtrar_imagem = np.zeros_like(imagem)
13          imagem_padded = cv2.copyMakeBorder(imagem, tamanho_mascara
               //2, tamanho_mascara//2, tamanho_mascara//2,
               tamanho_mascara//2, cv2.BORDER_CONSTANT)
14          for i in range(imagem.shape[0]):
15              for j in range(imagem.shape[1]):
16                  filtrar_imagem[i, j] = (imagem_padded[i:i+
                       tamanho_mascara, j:j+tamanho_mascara]*mascara).
                       sum()
```

```
17
18          # Retorna a imagem filtrada
19          return filtrar_imagem
```

6 Conclusões

Tarefa foi realizada com êxito, foi desenvolvida a aplicação com Qt designer e Python para processamento de imagens em tons de cinza e utilizando filtro gaussiano. Um dos únicos problemas foi que não estavamos conseguindo manter a câmera do webcam capturando em tempo real.Porém, algo que não interferiu no processo de construção da aplicação.

Referências

WOODS, R. C. G. e R. E. **Digital Image Processing**. [S.l.: s.n.], 2008.

Woods (2008).