**MÔN HỌC: THIẾT KẾ ỨNG DỤNG**

**DỰA TRÊN MÃ NGUỒN MỞ**

Lớp ưu tiên: 20231FE6073001

GVHD: Cô Phạm Quỳnh Trang

Nhóm thực hiện: Nhóm 06

Thành viên: Vũ Duy Mạnh MSV:2020603281

Nguyễn Văn Tiến Mạnh MSV:2020601515

Nguyễn Hữu Tú MSV:2020601956

Lê Anh Vũ MSV:2020604102

**CHỦ ĐỀ: THIẾT KẾ ỨNG DỤNG CHỈNH SỬA ẢNH**

**Mã nguồn:**

import cv2  
import numpy as np  
  
img = cv2.imread('pic2.png')  
  
cv2.imshow('Original', img)  
  
#gennerating the kernels  
kernel\_sharpen\_1 = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])  
kernel\_sharpen\_2 = np.array([[1,1,1], [1,-7,1], [1,1,1]])  
kernel\_sharpen\_3 = np.array([[-1,-1,-1,-1,-1], [-1,2,2,2,-1], [-1,2,8,2,-1], [-1,2,2,2,-1], [-1,-1,-1,-1,-1]])/8.0  
  
#applying different kernels to the input image  
output\_1 = cv2.filter2D(img, -1, kernel\_sharpen\_1)  
output\_2 = cv2.filter2D(img, -1, kernel\_sharpen\_2)  
output\_3 = cv2.filter2D(img, -1, kernel\_sharpen\_3)  
  
cv2.imshow('Sharpening', output\_1)  
cv2.imshow('Excessive Sharpening', output\_2)  
cv2.imshow('Edge Enhancement', output\_3)  
  
cv2.waitKey(0)

*Nguồn: TX2 môn học thiết kế ứng dụng bằng mã nguồn mở*

Phân tích mã nguồn:

Đây là chương trình thực hiện làm mờ ảnh(sharpening) bằng các kennel khác nhau trong thư viện OpenCV. Sau đó hiển thị hình ảnh ra màn hình bao gồm ảnh gốc (Orignal) và các ảnh sau khi làm mờ(Sharpening, Excessive Sharpening, Edge Enhanment)

img = cv2.imread('pic2.png')  
  
cv2.imshow('Original', img)

2 dòng lệnh trên dùng để đọc ảnh có tên ‘pic2.png’. Sau đó hiển thị với tên ‘Original’

#gennerating the kernels  
kernel\_sharpen\_1 = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])  
kernel\_sharpen\_2 = np.array([[1,1,1], [1,-7,1], [1,1,1]])  
kernel\_sharpen\_3 = np.array([[-1,-1,-1,-1,-1], [-1,2,2,2,-1], [-1,2,8,2,-1], [-1,2,2,2,-1], [-1,-1,-1,-1,-1]])/8.0

Tạo các kennel khác nhau cho quá trình làm mờ

Kennel 1 và 2 là 2 kennel cơ bản cho việc làm mờ

Kennel 3 được dùng để tăng cường cạnh

#applying different kernels to the input image  
output\_1 = cv2.filter2D(img, -1, kernel\_sharpen\_1)  
output\_2 = cv2.filter2D(img, -1, kernel\_sharpen\_2)  
output\_3 = cv2.filter2D(img, -1, kernel\_sharpen\_3)

Áp dụng các kennel vào ảnh

cv2.imshow('Sharpening', output\_1)  
cv2.imshow('Excessive Sharpening', output\_2)  
cv2.imshow('Edge Enhancement', output\_3)

Hiển thị ảnh lên màn hình

**Ý tưởng thực hiện**:

Cải thiện chương trình, thêm giao diện người dùng, thêm các chức năng khác: làm mờ, làm mịn, cắt, xoay ảnh….

1. Tạo Giao diện GUI cho chương trình

Tạo các nút bấm cơ bản cho các chức năng: cắt ảnh, vẽ ảnh, bộ lọc,….

tạo các nút bấm khác: Lưu ảnh, Lưu thay đổi, xem ảnh gốc…

* 1. class FrontEnd:  
      def \_\_init\_\_(self, master):  
      self.master = master  
      self.draw\_ids = []  
      self.menu\_initialisation()  
      self.x = 0 # Giá trị mặc định cho self.x  
      self.y = 0 # Giá trị mặc định cho self.y  
       
      def menu\_initialisation(self):  
      self.master.geometry('750x630+250+10')  
      self.master.title('Photoshop')  
      self.frame\_header = ttk.Frame(self.master)  
      self.frame\_header.pack()  
       
      ttk.Label(self.frame\_header, text='\nChỉnh sửa ảnh\n').grid(row=0, column=2, columnspan=1)  
       
      self.frame\_menu = ttk.Frame(self.master)  
      self.frame\_menu.pack()  
      self.frame\_menu.config(relief=RIDGE, padding=(50, 15))  
       
      ttk.Button(self.frame\_menu, text="Chọn ảnh", command=self.upload\_action).grid(row=0, column=0, columnspan=2,  
      padx=5, pady=5, sticky='sw')  
       
      ttk.Button(self.frame\_menu, text="Cắt ảnh", command=self.crop\_action).grid(row=1, column=0, columnspan=2, padx=5,  
      pady=5, sticky='sw')  
       
      ttk.Button(self.frame\_menu, text="Vẽ", command=self.draw\_action).grid(row=2, column=0, columnspan=2,  
      padx=5, pady=5, sticky='sw')  
       
      ttk.Button(self.frame\_menu, text="Bộ lọc", command=self.filter\_action).grid(row=3, column=0, columnspan=2,  
      padx=5, pady=5, sticky='sw')  
       
      ttk.Button(self.frame\_menu, text="Làm mờ/mịn", command=self.blur\_action).grid(row=4, column=0, columnspan=2,  
      padx=5, pady=5, sticky='sw')  
       
      ttk.Button(self.frame\_menu, text="Xoay ảnh", command=self.rotate\_action).grid(row=5, column=0, columnspan=2, padx=5,  
      pady=5, sticky='sw')  
       
      ttk.Button(self.frame\_menu, text="Lật ảnh", command=self.flip\_action).grid(row=6, column=0, columnspan=2, padx=5,  
      pady=5, sticky='sw')  
       
      ttk.Button(self.frame\_menu, text="Lưu ảnh", command=self.save\_action).grid(row=7, column=0, columnspan=2, padx=5,  
      pady=5, sticky='sw')  
       
      ttk.Button(self.frame\_menu, text="Hình ảnh gốc", command=self.reset\_to\_original).grid(row=8, column=0, columnspan=2,  
      padx=5, pady=5, sticky='sw')  
       
      self.canvas = Canvas(self.frame\_menu, bg="gray", width=300, height=400)  
      self.canvas.grid(row=0, column=3, rowspan=10)  
       
      self.side\_frame = ttk.Frame(self.frame\_menu)  
      self.side\_frame.grid(row=0, column=4, rowspan=10)  
      self.side\_frame.config(relief=GROOVE, padding=(50, 15))  
       
      self.apply\_and\_cancel = ttk.Frame(self.master)  
      self.apply\_and\_cancel.pack()  
      self.apply = ttk.Button(self.apply\_and\_cancel, text="Lưu thay đổi", command=self.apply\_action).grid(row=0, column=0,  
      columnspan=1, padx=5, pady=5, sticky='sw')  
       
      ttk.Button(self.apply\_and\_cancel, text="Hủy thay đổi", command=self.cancel\_action).grid(row=0, column=1, columnspan=1,  
      padx=5, pady=5, sticky='sw')  
       
      self.show\_original = tk.BooleanVar() # Tạo một biến kiểu BooleanVar để theo dõi trạng thái của checkbutton  
      self.show\_original.set(False) # Ban đầu không xem ảnh gốc  
       
      check\_button = ttk.Checkbutton(self.apply\_and\_cancel, text="Xem ảnh gốc", variable=self.show\_original,  
      command=self.revert\_action)  
      check\_button.grid(row=0, column=2, columnspan=1, padx=5, pady=5, sticky='sw')  
       
       
      ttk.Button(self.master, text="Thoát", command=self.master.destroy).pack(padx=10, pady=10)

1. Tạo chức năng chọn ảnh

def upload\_action(self):  
 self.canvas.delete("all")  
 self.filename = filedialog.askopenfilename()  
  
 # Kiểm tra tệp có phải là tệp ảnh hay không  
 try:  
 img = Image.open(self.filename)  
 img.close()  
 # Nếu không có lỗi xảy ra khi mở tệp, đó là tệp hình ảnh  
 # Thực hiện việc đọc và hiển thị hình ảnh tại đây  
 self.original\_image = cv2.imread(self.filename)  
 self.edited\_image = cv2.imread(self.filename)  
 self.filtered\_image = cv2.imread(self.filename)  
 self.display\_image(self.edited\_image)  
 except (IOError, OSError):  
 # Nếu có lỗi xảy ra khi mở tệp, đó không phải là tệp hình ảnh  
 messagebox.showerror("Lỗi", "Tệp không phải là hình ảnh")

1. Tạo chức năng cắt ảnh:

def crop\_action(self):  
 if not hasattr(self, 'filename') or not self.filename:  
 messagebox.showwarning("Lỗi", "Vui lòng chọn ảnh trước khi thực hiện chức năng")  
 return  
 self.rectangle\_id = 0  
 # self.ratio = 0  
 self.crop\_start\_x = 0  
 self.crop\_start\_y = 0  
 self.crop\_end\_x = 0  
 self.crop\_end\_y = 0  
 self.canvas.bind("<ButtonPress>", self.start\_crop)  
 self.canvas.bind("<B1-Motion>", self.crop)  
 self.canvas.bind("<ButtonRelease>", self.end\_crop)  
  
def start\_crop(self, event):  
 self.crop\_start\_x = event.x  
 self.crop\_start\_y = event.y  
  
def crop(self, event):  
 if self.rectangle\_id:  
 self.canvas.delete(self.rectangle\_id)  
  
 self.crop\_end\_x = event.x  
 self.crop\_end\_y = event.y  
  
 self.rectangle\_id = self.canvas.create\_rectangle(self.crop\_start\_x, self.crop\_start\_y, self.crop\_end\_x,  
 self.crop\_end\_y, width=1)  
  
def end\_crop(self, event):  
 if self.crop\_start\_x <= self.crop\_end\_x and self.crop\_start\_y <= self.crop\_end\_y:  
 start\_x = int(self.crop\_start\_x \* self.ratio)  
 start\_y = int(self.crop\_start\_y \* self.ratio)  
 end\_x = int(self.crop\_end\_x \* self.ratio)  
 end\_y = int(self.crop\_end\_y \* self.ratio)  
 elif self.crop\_start\_x > self.crop\_end\_x and self.crop\_start\_y <= self.crop\_end\_y:  
 start\_x = int(self.crop\_end\_x \* self.ratio)  
 start\_y = int(self.crop\_start\_y \* self.ratio)  
 end\_x = int(self.crop\_start\_x \* self.ratio)  
 end\_y = int(self.crop\_end\_y \* self.ratio)  
 elif self.crop\_start\_x <= self.crop\_end\_x and self.crop\_start\_y > self.crop\_end\_y:  
 start\_x = int(self.crop\_start\_x \* self.ratio)  
 start\_y = int(self.crop\_end\_y \* self.ratio)  
 end\_x = int(self.crop\_end\_x \* self.ratio)  
 end\_y = int(self.crop\_start\_y \* self.ratio)  
 else:  
 start\_x = int(self.crop\_end\_x \* self.ratio)  
 start\_y = int(self.crop\_end\_y \* self.ratio)  
 end\_x = int(self.crop\_start\_x \* self.ratio)  
 end\_y = int(self.crop\_start\_y \* self.ratio)  
  
 x = slice(start\_x, end\_x, 1)  
 y = slice(start\_y, end\_y, 1)  
  
 self.filtered\_image = self.edited\_image[y, x]  
 self.display\_image(self.filtered\_image)  
  
def end\_text\_crop(self, event):  
 if self.crop\_start\_x <= self.crop\_end\_x and self.crop\_start\_y <= self.crop\_end\_y:  
 start\_x = int(self.crop\_start\_x \* self.ratio)  
 start\_y = int(self.crop\_start\_y \* self.ratio)  
 end\_x = int(self.crop\_end\_x \* self.ratio)  
 end\_y = int(self.crop\_end\_y \* self.ratio)  
 elif self.crop\_start\_x > self.crop\_end\_x and self.crop\_start\_y <= self.crop\_end\_y:  
 start\_x = int(self.crop\_end\_x \* self.ratio)  
 start\_y = int(self.crop\_start\_y \* self.ratio)  
 end\_x = int(self.crop\_start\_x \* self.ratio)  
 end\_y = int(self.crop\_end\_y \* self.ratio)  
 elif self.crop\_start\_x <= self.crop\_end\_x and self.crop\_start\_y > self.crop\_end\_y:  
 start\_x = int(self.crop\_start\_x \* self.ratio)  
 start\_y = int(self.crop\_end\_y \* self.ratio)  
 end\_x = int(self.crop\_end\_x \* self.ratio)  
 end\_y = int(self.crop\_start\_y \* self.ratio)  
 else:  
 start\_x = int(self.crop\_end\_x \* self.ratio)  
 start\_y = int(self.crop\_end\_y \* self.ratio)  
 end\_x = int(self.crop\_start\_x \* self.ratio)  
 end\_y = int(self.crop\_start\_y \* self.ratio)  
  
 if self.text\_on\_image.get():  
 self.text\_extracted = self.text\_on\_image.get()  
 start\_font = start\_x, start\_y  
 print(self.color\_code) # ((r,g,b),'#ff00000')  
 r, g, b = tuple(map(int, self.color\_code[0]))  
  
 self.filtered\_image = cv2.putText(self.edited\_image, self.text\_extracted, start\_font, cv2.FONT\_HERSHEY\_SIMPLEX, 2,  
 (b, g, r), 5)  
 self.display\_image(self.filtered\_image)

1. Tạo chức năng vẽ lên ảnh

def draw\_action(self):  
 if not hasattr(self, 'filename') or not self.filename:  
 messagebox.showwarning("Lỗi", "Vui lòng chọn ảnh trước khi thực hiện chức năng")  
 return  
  
 self.color\_code = ((255, 0, 0), '#ff0000')  
 self.refresh\_side\_frame()  
 self.canvas.bind("<ButtonPress>", self.start\_draw)  
 self.canvas.bind("<B1-Motion>", self.draw)  
 self.draw\_color\_button = ttk.Button(self.side\_frame, text="Chọn màu", command=self.choose\_color)  
 self.draw\_color\_button.grid(row=0, column=2, padx=5, pady=5, sticky='sw')  
  
def choose\_color(self):  
 color\_tuple = colorchooser.askcolor(title="Choose color")  
 if color\_tuple:  
 self.color\_code = color\_tuple  
 # Cập nhật self.x và self.y vào giá trị mặc định  
 self.x = 0  
 self.y = 0  
 self.draw(event=None) # Gọi lại hàm draw với màu mới đã chọn  
  
def start\_draw(self, event):  
 self.x = event.x  
 self.y = event.y  
 #self.draw\_ids = []  
  
def draw(self, event):  
 if event:  
 if hasattr(event, 'x') and hasattr(event, 'y'):  
 if self.x and self.y:  
 self.canvas.create\_line(self.x, self.y, event.x, event.y, width=2, fill=self.color\_code[-1], capstyle=ROUND, smooth=True)  
 r, g, b = self.color\_code[0]  
 cv2.line(self.filtered\_image, (int(self.x \* self.ratio), int(self.y \* self.ratio)),  
 (int(event.x \* self.ratio), int(event.y \* self.ratio)), (b, g, r), thickness=int(self.ratio \* 2), lineType=8)  
 self.x = event.x  
 self.y = event.y

1. Tạo chức năng bộ lọc

def filter\_action(self):  
 if not hasattr(self, 'filename') or not self.filename:  
 messagebox.showwarning("Lỗi", "Vui lòng chọn ảnh trước khi thực hiện chức năng")  
 return  
 self.refresh\_side\_frame()  
 ttk.Button(self.side\_frame, text="Negative", command=self.negative\_action).grid(row=0, column=2, padx=5, pady=5,  
 sticky='se')  
  
 ttk.Button(self.side\_frame, text="Black And white", command=self.bw\_action).grid(row=1, column=2, padx=5, pady=5,  
 sticky='se')  
  
 ttk.Button(self.side\_frame, text="Sharpening", command=self.Sharpening\_action).grid(row=2, column=2, padx=5, pady=5,  
 sticky='se')  
  
 ttk.Button(self.side\_frame, text="Excessive Sharpening", command=self.ExSharpening\_action).grid(row=3, column=2, padx=5, pady=5,  
 sticky='se')  
  
 ttk.Button(self.side\_frame, text="Edge Enhancement", command=self.EdEnhancement\_action).grid(row=4, column=2, padx=5, pady=5,  
 sticky='se')

1. Tạo chức năng làm mờ, mịn ảnh

def blur\_action(self):  
 if not hasattr(self, 'filename') or not self.filename:  
 messagebox.showwarning("Lỗi", "Vui lòng chọn ảnh trước khi thực hiện chức năng")  
 return  
 self.refresh\_side\_frame()  
  
 ttk.Label(self.side\_frame, text="Mức 1").grid(row=0, column=2, padx=5, sticky='sw')  
 self.gaussian\_slider = Scale(self.side\_frame, from\_=0, to=256, orient=HORIZONTAL, command=self.gaussian\_action)  
 self.gaussian\_slider.grid(row=1, column=2, padx=5, sticky='sw')  
  
 ttk.Label(self.side\_frame, text="Mức 2").grid(row=2, column=2, padx=5, sticky='sw')  
 self.average\_slider = Scale(self.side\_frame, from\_=0, to=256, orient=HORIZONTAL, command=self.averaging\_action)  
 self.average\_slider.grid(row=3, column=2, padx=5, sticky='sw')  
  
 ttk.Label(self.side\_frame, text="Mức 3").grid(row=4, column=2, padx=5, sticky='sw')  
 self.median\_slider = Scale(self.side\_frame, from\_=0, to=256, orient=HORIZONTAL, command=self.median\_action)  
 self.median\_slider.grid(row=5, column=2, padx=5, sticky='sw')

1. Tạo chức năng lật, xoay ảnh

def rotate\_action(self):  
 if not hasattr(self, 'filename') or not self.filename:  
 messagebox.showwarning("Lỗi", "Vui lòng chọn ảnh trước khi thực hiện chức năng")  
 return  
 self.refresh\_side\_frame()  
 ttk.Button(self.side\_frame, text="Xoay trái", command=self.rotate\_left\_action).grid(row=0, column=2, padx=5,  
 pady=5, sticky='sw')  
  
 ttk.Button(self.side\_frame, text="Xoay phải", command=self.rotate\_right\_action).grid(row=1, column=2, padx=5,  
 pady=5, sticky='sw')  
  
def flip\_action(self):  
 if not hasattr(self, 'filename') or not self.filename:  
 messagebox.showwarning("Lỗi", "Vui lòng chọn ảnh trước khi thực hiện chức năng")  
 return  
 self.refresh\_side\_frame()  
 ttk.Button(self.side\_frame, text="Lật dọc", command=self.vertical\_action).grid(row=0, column=2, padx=5,  
 pady=5, sticky='se')  
  
 ttk.Button(self.side\_frame, text="Lật ngang", command=self.horizontal\_action).grid(row=1, column=2, padx=5,  
 pady=5, sticky='se')