ДОДАТОК А

Програмний код, що реалізує алгоритм аналізу стилістичної подібності зображень з використанням MediaPipe KNIFT

```
import ttkbootstrap as tb
from ttkbootstrap.constants import *
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
import cv2
import numpy as np
import math


def load_image_unicode(path):
    try:
        pil_img = Image.open(path).convert('L')
        return np.array(pil_img)
    except Exception as e:
        print(f"Помилка: {e}")
        return None


def compare_features(detector_name, detector_func, img1, img2):
    kp1, des1 = detector_func(img1)
    kp2, des2 = detector_func(img2)

    if des1 is None or des2 is None:
        return None, "Ознак не виявлено."

    matcher_type = cv2.NORM_HAMMING if detector_name in ["ORB", "BRISK",
"AKAZE"] else cv2.NORM_L2
    bf = cv2.BFMatcher(matcher_type, crossCheck=True)
    matches = bf.match(des1, des2)
    matches = sorted(matches, key=lambda x: x.distance)
    img_match = cv2.drawMatches(img1, kp1, img2, kp2, matches[:30], None, flags=2)

    total_matches = len(matches)
    avg_distance = round(np.mean([m.distance for m in matches]), 2) if matches else 0

    if total_matches < 15:
        conclusion = "Схожість низька або відсутня."
    elif total_matches < 40:
        conclusion = "Можлива часткова стилістична подібність."
    else:
        conclusion = "Існує висока ймовірність стилістичної подібності."

    info = (
        f"Метод: {detector_name}\n"
        f"Ключові точки: {len(kp1)} (зображення 1) + {len(kp2)} (зображення 2)\n"
```

```
        f"Збіги: {total_matches}\n"
        f"Середня відстань: {avg_distance}\n\n"
        f"Висновок: {conclusion}"
    )

    return img_match, info


def get_detector_function(method):
    if method == 'orb':
        orb = cv2.ORB_create(nfeatures=1000)
        return "ORB", lambda img: orb.detectAndCompute(img, None)
    elif method == 'sift':
        sift = cv2.SIFT_create()
        return "SIFT", lambda img: sift.detectAndCompute(img, None)
    elif method == 'brisk':
        brisk = cv2.BRISK_create()
        return "BRISK", lambda img: brisk.detectAndCompute(img, None)
    elif method == 'akaze':
        akaze = cv2.AKAZE_create()
        return "AKAZE", lambda img: akaze.detectAndCompute(img, None)
    else:
        raise ValueError("Невідомий метод")


class StyleMatcherApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Style Matcher – Імітація MediaPipe KNIFT")
        self.root.geometry("1200x1000")
        self.root.resizable(False, False)

        self.img1_path = ""
        self.img2_path = ""
        self.img1_array = None
        self.img2_array = None
        self.tk_img = None

        self.setup_ui()

    def setup_ui(self):
        top_frame = tb.Frame(self.root)
        top_frame.pack(pady=10)

        tb.Label(top_frame, text="Зображення 1:").grid(row=0, column=0, padx=5)
        self.img1_label = tb.Label(top_frame, text="Не вибрано")
        self.img1_label.grid(row=0, column=1)
        tb.Button(top_frame, text="Обрати", bootstyle=PRIMARY,
command=self.load_img1).grid(row=0, column=2, padx=5)

        tb.Label(top_frame, text="Зображення 2:").grid(row=1, column=0, padx=5)
        self.img2_label = tb.Label(top_frame, text="Не вибрано")
```

```
        self.img2_label.grid(row=1, column=1)
        tb.Button(top_frame, text="Обрати", bootstyle=PRIMARY,
command=self.load_img2).grid(row=1, column=2, padx=5)

        button_frame = tb.Frame(self.root)
        button_frame.pack(pady=10)

        tb.Button(button_frame, text="ORB", bootstyle=SUCCESS, width=15,
command=lambda: self.compare('orb')).grid(row=0, column=0, padx=10)
        tb.Button(button_frame, text="SIFT", bootstyle=INFO, width=15, command=lambda:
self.compare('sift')).grid(row=0, column=1, padx=10)
        tb.Button(button_frame, text="BRISK", bootstyle=WARNING, width=15,
command=lambda: self.compare('brisk')).grid(row=0, column=2, padx=10)
        tb.Button(button_frame, text="AKAZE", bootstyle=DANGER, width=15,
command=lambda: self.compare('akaze')).grid(row=0, column=3, padx=10)

        tb.Label(self.root, text="Патчі ключових точок", font=("Arial", 10)).pack()
        self.patch_canvas = tb.Canvas(self.root, width=1100, height=150, bg="lightgray")
        self.patch_canvas.pack(pady=5)

        tb.Label(self.root, text="Візуалізація порівняння", font=("Arial", 10)).pack()
        self.canvas = tb.Canvas(self.root, width=1100, height=400, bg="white")
        self.canvas.pack(pady=5)

        self.text_info = tb.Label(self.root, text="Інформація про результат буде тут.",
wraplength=1100, justify="center", font=("Arial", 12))
        self.text_info.pack(pady=10)

    def load_img1(self):
        self.img1_path = filedialog.askopenfilename(filetypes=[("Зображення", "*.png *.jpg
*.jpeg")])
        if self.img1_path:
            self.img1_array = load_image_unicode(self.img1_path)
            self.img1_label.config(text=self.img1_path.split("/")[-1])

    def load_img2(self):
        self.img2_path = filedialog.askopenfilename(filetypes=[("Зображення", "*.png *.jpg
*.jpeg")])
        if self.img2_path:
            self.img2_array = load_image_unicode(self.img2_path)
            self.img2_label.config(text=self.img2_path.split("/")[-1])

    def compare(self, method):
        if self.img1_array is None or self.img2_array is None:
            messagebox.showerror("Помилка", "Оберіть обидва зображення.")
            return

        try:
            method_name, detector_func = get_detector_function(method)
            result, info = compare_features(method_name, detector_func, self.img1_array,
self.img2_array)
```

```
            if result is None:
                self.text_info.config(text="Недостатньо ознак для порівняння.")
                return

            result = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
            result_img = Image.fromarray(result).resize((1100, 400))
            self.tk_img = ImageTk.PhotoImage(result_img)

            self.canvas.delete("all")
            self.canvas.create_image(0, 0, anchor="nw", image=self.tk_img)
            self.text_info.config(text=info)

            self.show_patches()

        except Exception as e:
            messagebox.showerror("Помилка обробки", str(e))

    def show_patches(self):
        def extract_patches(image, num=15):
            gray = image
            kp = cv2.ORB_create(nfeatures=num).detect(gray, None)
            pyramid = [gray]
            for _ in range(3):
                gray = cv2.resize(gray, (round(gray.shape[1]/1.2), round(gray.shape[0]/1.2)))
                pyramid.append(gray)
            patches = []
            for pt in kp[:num]:
                octave = getattr(pt, 'octave', 0)
                if octave >= len(pyramid):
                    octave = 0
                center = pt.pt
                rot = cv2.getRotationMatrix2D(center, pt.angle, 1.0)
                rot[0][2] += 16 – center[0]
                rot[1][2] += 16 – center[1]
                patch = cv2.warpAffine(pyramid[octave], rot, (32, 32),
flags=cv2.INTER_LINEAR)
                patches.append(patch)
            return patches

        img1 = self.img1_array
        img2 = self.img2_array

        patches1 = extract_patches(img1, 15)
        patches2 = extract_patches(img2, 15)

        combined_img = Image.new("L", (1050, 70 * 2))
        for i, p in enumerate(patches1):
            im = Image.fromarray(p).resize((64, 64))
            combined_img.paste(im, (i * 70, 0))
        for i, p in enumerate(patches2):
            im = Image.fromarray(p).resize((64, 64))
            combined_img.paste(im, (i * 70, 70))
```

```
        tk_patch = ImageTk.PhotoImage(combined_img)
        self.patch_canvas.delete("all")
        self.patch_canvas.create_image(0, 0, anchor="nw", image=tk_patch)
        self.patch_canvas.image = tk_patch


if __name__ == "__main__":
    app = tb.Window(themename="superhero")
    StyleMatcherApp(app)
    app.mainloop()
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	78 с.
	Інші документи	
2.	Презентаційні матеріали	16 плакатів

.	.	Прізвище та ініціали.	Підп.	Дата	
Розробив		Матюшенко М.В.			Розробка алгоритму аналізу стилістичних особливостей графічного контенту на предмет порушення авторських прав з використанням алгоритму Media Pipe KNIFT
Перевірив		Селіванова К.Г.			
Н.контр.		Малєєва І.А.			
Затв.		Золотухін О.В.			

Шифр групи	Код напр./спец.
СШІмзд-23-1	122
ХНУРЕ кафедра ШІ	