Malek Soula L2CS03

Mini-Projet TP Technique D'indexation

1/ L'importation des bibliothèques nécessaires tkinter(l'interface graphique), PIL… :

```
import tkinter as tk
from tkinter import filedialog, ttk
from PIL import Image, ImageTk
import cv2
import numpy as np
from sklearn.neighbors import NearestNeighbors
from tensorflow.keras.applications.vgg16 import VGG16,preprocess_input
from tensorflow.keras.preprocessing import image
import os
```

2/ Développement de l'interface graphique avec tkinter

```
class ImageSearchApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Image Search Engine")
        self.root.geometry("1200x800")

        # Image database
        self.images = []
        self.filenames = []
        self.features = {}
        self.nbrs = {}

        # UI Setup
        self.setup_ui()

        # Load VGG16 model
        self.vgg_model = VGG16(weights='imagenet', include_top=False, pooling='avg')

    def setup_ui(self):
        # Left panel - Controls
        control_frame = tk.Frame(self.root, width=300, bg="#f0f0f0")
        control_frame.pack(side=tk.LEFT, fill=tk.Y, padx=10, pady=10)

        # Database controls
        db_frame = tk.LabelFrame(control_frame, text="Image Database", bg="#f0f0f0")
        db_frame.pack(fill=tk.X, pady=5)

        tk.Button(db_frame, text="Load Image Folder", command=self.load_image_folder).pack(fill=tk.X, pady=5)
        tk.Button(db_frame, text="Build Indexes", command=self.build_indexes).pack(fill=tk.X, pady=5)

    # Search controls
    search_frame = tk.LabelFrame(control_frame, text="Search", bg="#f0f0f0")
    search_frame.pack(fill=tk.X, pady=5)

    self.descriptor_var = tk.StringVar(value="color_histogram")
    ttk.Combobox(search_frame, textvariable=self.descriptor_var,
                 values=["color_histogram", "sift", "vgg16"]).pack(fill=tk.X, pady=5)

    tk.Button(search_frame, text="Select Query Image", command=self.select_query_image).pack(fill=tk.X, pady=5)

    # Status bar
    self.status_var = tk.StringVar()
    tk.Label(control_frame, textvariable=self.status_var, bg="#f0f0f0", anchor="w").pack(fill=tk.X, pady=10)

    # Right panel - Results
    result_frame = tk.Frame(self.root)
    result_frame.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True, padx=10, pady=10)

    # Query image display
    self.query_label = tk.Label(result_frame, text="Query Image", relief=tk.SUNKEN)
    self.query_label.pack(fill=tk.X, pady=5)

    # Results display
    results_title = tk.Label(result_frame, text="Top 5 Similar Images")
    results_title.pack(fill=tk.X)
```

3/ Les méthodes pour le chargement des images :

```python
def load_image_folder(self):
    folder_path = filedialog.askdirectory()
    if folder_path:
        self.images, self.filenames = self.load_images(folder_path)
        self.status_var.set(f"Loaded {len(self.images)} images from {folder_path}")

def load_images(self, directory, target_size=(224, 224)):
    images = []
    filenames = []
    for filename in os.listdir(directory):
        if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
            img_path = os.path.join(directory, filename)
            img = cv2.imread(img_path)
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            img = cv2.resize(img, target_size)
            images.append(img)
            filenames.append(filename)
    return np.array(images), filenames
```

4/ La Construction des indexes avec le dossier des images qu'on a sélectionné

```python
    def build_indexes(self):
        if len(self.images) == 0:
            self.status_var.set("Error: No images loaded!")
            return

        self.status_var.set("Building indexes...")
        self.root.update_idletasks()

        # Color Histogram
        self.features["color_histogram"] = self.color_histogram(self.images)
        self.nbrs["color_histogram"] = NearestNeighbors(n_neighbors=5, metric='cosine').fit(
            self.features["color_histogram"])

        # VGG16
        self.features["vgg16"] = self.extract_vgg_features(self.images)
        self.nbrs["vgg16"] = NearestNeighbors(n_neighbors=5, metric='cosine').fit(
            self.features["vgg16"])

        self.status_var.set("Indexes built successfully!")

    def color_histogram(self, images):
        histograms = []
        for img in images:
            hist = cv2.calcHist([img], [0, 1, 2], None, [8, 8, 8], [0, 256, 0, 256, 0, 256])
            hist = cv2.normalize(hist, hist).flatten()
            histograms.append(hist)
        return np.array(histograms)
```

5/ La Sélection de l’image à rechercher

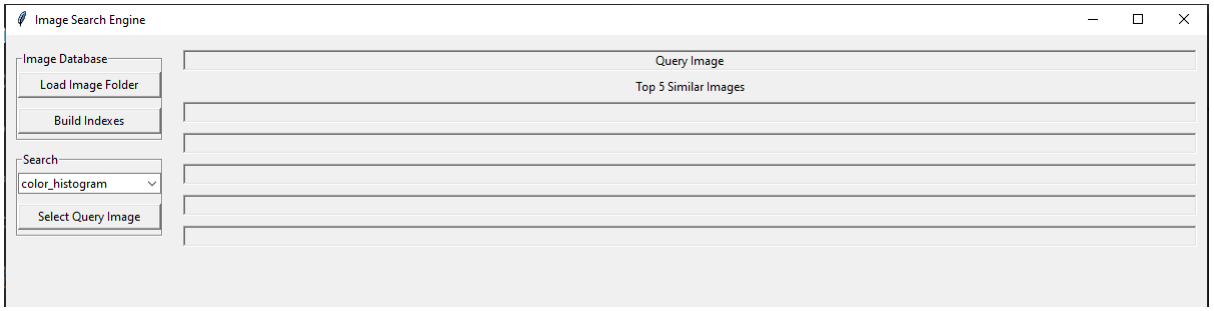
```
def select_query_image(self):
    if len(self.images) == 0:
        self.status_var.set("Error: Please load image database first!")
        return

    file_path = filedialog.askopenfilename(filetypes=[("Image files", "*.jpg *.jpeg *.png")])
    if file_path:
        self.search_similar_images(file_path)
```

6/ Recherche des deux images similaires :

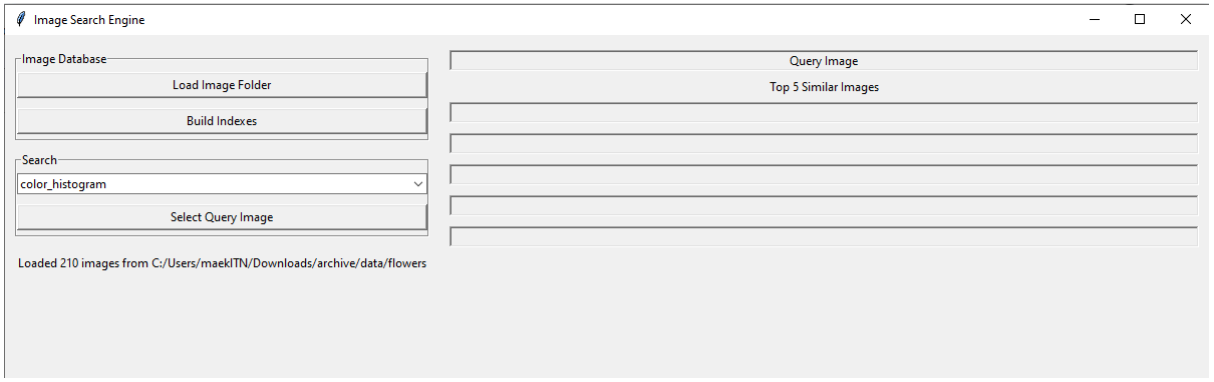
```
def search_similar_images(self, query_path):
    descriptor_type = self.descriptor_var.get()

    # Load and process query image
    query_img = cv2.imread(query_path)
    query_img = cv2.cvtColor(query_img, cv2.COLOR_BGR2RGB)
    query_img = cv2.resize(query_img, (224, 224))

    # Display query image
    self.display_image(query_img, self.query_label)

    # Extract features based on descriptor type
    if descriptor_type == "color_histogram":
        query_feat = self.color_histogram([query_img])[0]
    elif descriptor_type == "vgg16":
        query_feat = self.extract_vgg_features([query_img])[0]

    # Find similar images
    distances, indices = self.nbrs[descriptor_type].kneighbors([query_feat])

    # Display results
    for i, (idx, dist) in enumerate(zip(indices[0], distances[0])):
        if i < 5:  # Show top 5 results
            result_img = self.images[idx]
            self.display_image(result_img, self.result_labels[i],
                               f"Similarity: {1-dist:.2f}")
```

Malek Soula L2CS03

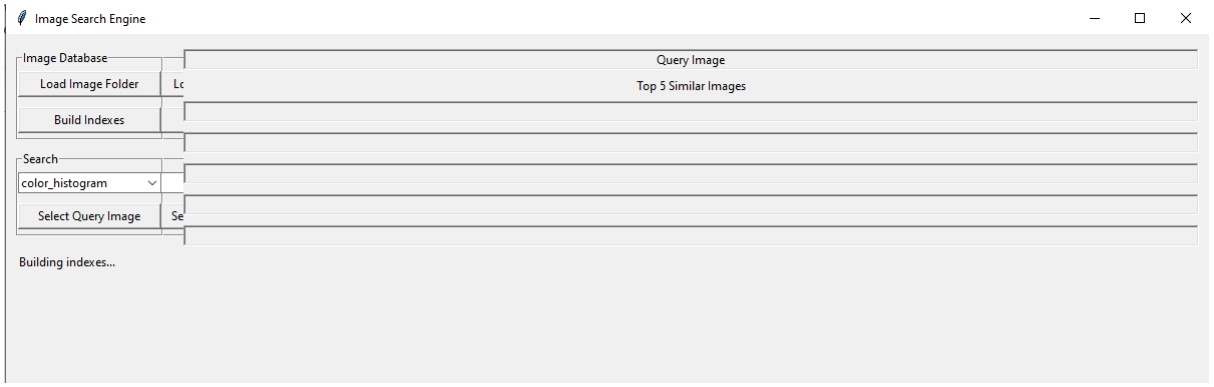
L’interface graphique principale de l’application :

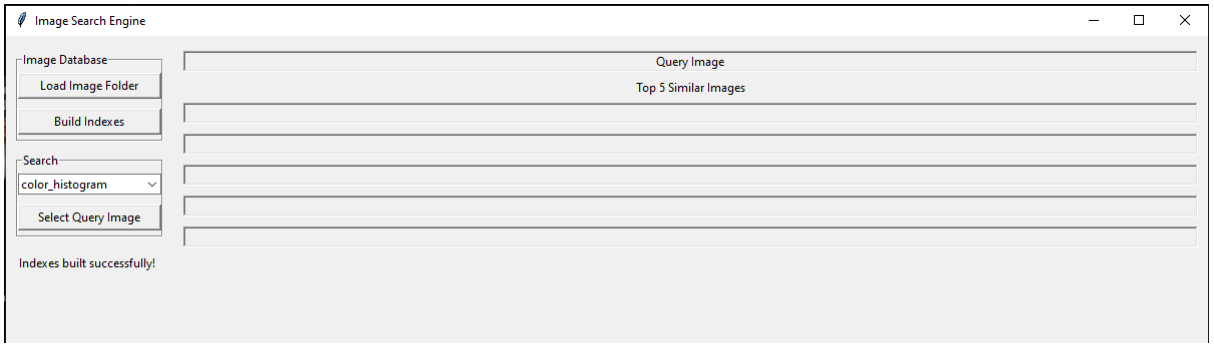


Le processus de chargement des images :



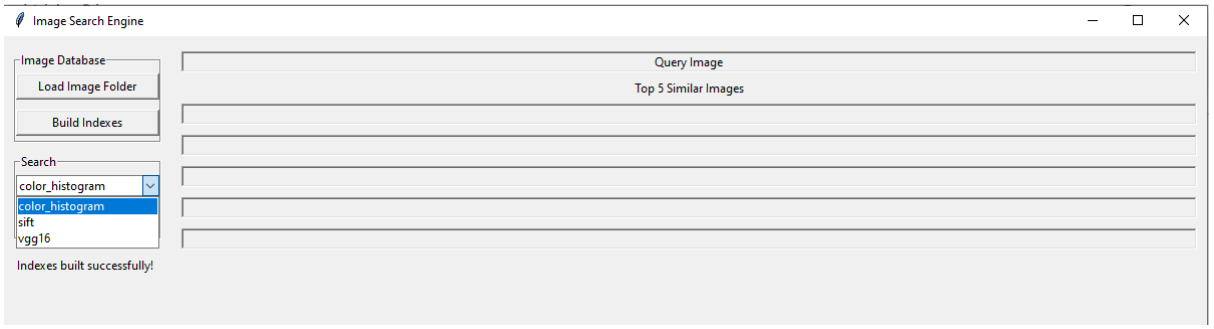
La Construction des indices à partir du dossier des images sélectionné :



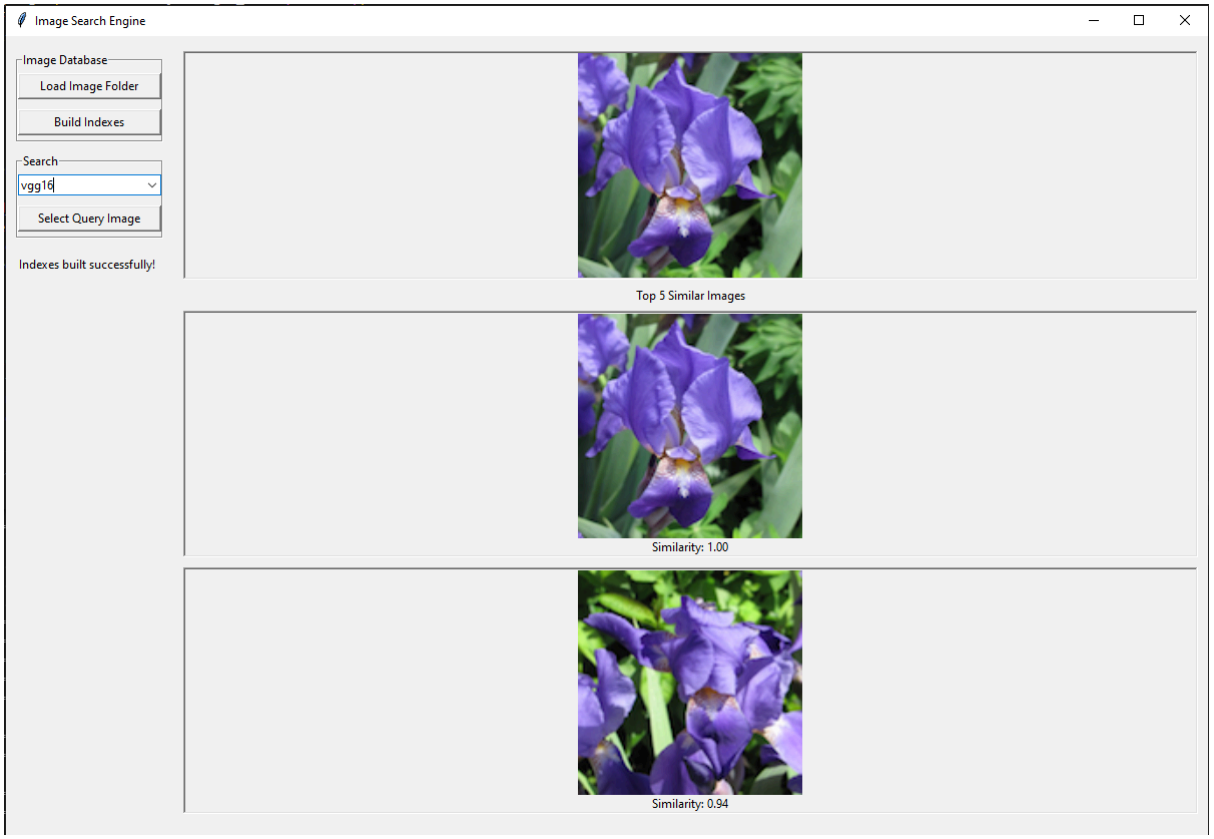


Malek Soula L2CS03

Le choix du modèle de recherche soit par l'historgramme de couleur soit par le modele VGG16



Les résultats de la recherche avec le modèle VGG16



Malek Soula L2CS03

Les resultats de recherche avec l’histogramme de couleur des images