Universidad Tecnológica de Honduras

Manual

Tercer Parcial

Inteligencia Artificial

Catedrático:

Norman Alberto Cubilla Rivera

Integrantes:

Owen Javier Hernández (202010011089)

Fecha:

8 de agosto del 2025

1. Archivo: menu_distancia.py

Este archivo es el menú principal del sistema. Permite al usuario interactuar con el programa mediante una interfaz gráfica.

```python
import tkinter as tk
from tkinter import messagebox
import subprocess
import os
```

- import tkinter as tk: Importa la biblioteca gráfica tkinter, que se usa para crear ventanas y botones.
- from tkinter import messagebox: Importa el módulo messagebox para mostrar ventanas emergentes (errores, advertencias).
- import subprocess: Permite ejecutar otros programas o scripts de Python desde este archivo.
- import os: Se usa para verificar si los archivos necesarios existen en el sistema.

```python
ARCHIVO_MANUAL = "Medir_Distancia/distancia_manual.py"
ARCHIVO_AUTOMATICO = "Medir_Distancia/distancia_automatica.py"
ENVIO_CORREO = "Medir_Distancia/enviar_distancia.py"
```

ARCHIVO_MANUAL = "Medir_Distancia/distancia_manual.py"

ARCHIVO_AUTOMATICO = "Medir_Distancia/distancia_automatica.py"

ENVIO_CORREO = "Medir_Distancia/enviar_distancia.py"

- Define las rutas de los tres scripts principales. Asegúrate de que estas rutas coincidan con la estructura de carpetas de tu proyecto.

```python
def verificar_archivos():
    faltantes = []
    if not os.path.exists(ARCHIVO_MANUAL):
        faltantes.append(ARCHIVO_MANUAL)
    if not os.path.exists(ARCHIVO_AUTOMATICO):
        faltantes.append(ARCHIVO_AUTOMATICO)
    if not os.path.exists(ENVIO_CORREO):
        faltantes.append(ENVIO_CORREO)
    return faltantes
```

- Esta función comprueba si los tres archivos necesarios existen.
- Si falta alguno, lo agrega a la lista faltantes y la devuelve.
- Ayuda a evitar errores al ejecutar el programa.

```python
def ejecutar_manual():
    faltantes = verificar_archivos()
    if faltantes:
        messagebox.showerror("Error", f"No se encontraron los archivos: {', '.join(faltantes)}")
        return
    try:
        subprocess.run(["python3", ARCHIVO_MANUAL], check=True)
    except subprocess.CalledProcessError:
        messagebox.showerror("Error", "Hubo un error al ejecutar el modo manual.")
    except FileNotFoundError:
        messagebox.showerror("Error", "No se encontró 'python3'. Asegúrate de tener Python instalado.")
```

- ejecutar_manual(): Esta función se activa cuando el usuario hace clic en el botón "Modo Manual".
- Primero verifica si los archivos existen.
- Luego ejecuta el script distancia_manual.py usando subprocess.run().
- Si hay un error (por ejemplo, el script falla o no se encuentra Python), muestra un mensaje de error.

```python
def ejecutar_automatico():
    faltantes = verificar_archivos()
    if faltantes:
        messagebox.showerror("Error", f"No se encontraron los archivos: {', '.join(faltantes)}")
        return
    try:
        subprocess.run(["python3", ARCHIVO_AUTOMATICO], check=True)
    except subprocess.CalledProcessError:
        messagebox.showerror("Error", "Hubo un error al ejecutar el modo automático.")
    except FileNotFoundError:
        messagebox.showerror("Error", "No se encontró 'python3'. Asegúrate de tener Python instalado.")
```

- ejecutar_automatico(): Similar a la anterior, pero ejecuta el script automático.
- Ideal para usuarios que no quieran hacer clic manualmente.

```python
def enviar_resultados():
    try:
        subprocess.run(["python3", ENVIO_CORREO], check=True)
        print("Correo electrónico enviado correctamente.")
    except Exception as e:
        messagebox.showerror("Error", f"Error al enviar el correo electrónico: {e}")
```

- enviar_resultados(): Ejecuta el script enviar_distancia.py.
- Asume que ya se generaron los archivos resultado_con_puntos.jpg y resultados.txt.

- Si ocurre un error (por ejemplo, sin conexión a internet), muestra una ventana de error.

```python
def menu_principal():
    faltantes = verificar_archivos()
    if faltantes:
        messagebox.showwarning("Advertencia", f"No se encontraron: {', '.join(faltantes)}\nAsegúrate de que estén en la
```

- Crea la ventana principal del menú.
- Muestra una advertencia si faltan archivos, antes de abrir la interfaz.

```python
    root = tk.Tk()
    root.title("📏 Calculadora de Distancia")
    root.geometry("400x300")
    root.resizable(False, False)
```

- tk.Tk(): Crea la ventana principal.
- .title(): Pone un título a la ventana.
- .geometry(): Define el tamaño de la ventana (400px de ancho, 300px de alto).
- .resizable(False, False): Evita que el usuario cambie el tamaño de la ventana.

```python
    tk.Label(root, text="Selecciona una opción", font=("Arial", 18, "bold")).pack(pady=20)
```

- Crea un texto centrado que dice "Selecciona una opción".

```python
    tk.Button(
        root,
        text="Modo Manual",
        font=("Arial", 14),
        width=20,
        bg="#2196F3",
        fg="white",
        command=ejecutar_manual
    ).pack(pady=10)
```

- Crea un botón azul que dice "Modo Manual".
- Cuando se hace clic, ejecuta la función ejecutar_manual.

```python
    tk.Button(
        root,
        text="Modo Automático",
        font=("Arial", 14),
        width=20,
        bg="#4CAF50",
        fg="white",
        command=ejecutar_automatico
    ).pack(pady=10)
```

- Botón verde para el modo automático.

```python
    tk.Button(
        root,
        text="Enviar Resultados",
        font=("Arial", 14),
        width=20,
        bg="#FF9800",
        fg="white",
        command=enviar_resultados
    ).pack(pady=10)
```

- Botón naranja para enviar los resultados por correo.

```python
    tk.Label(
        root,
        text="Los scripts deben estar en la misma carpeta",
        font=("Arial", 9),
        fg="gray"
    ).pack(pady=10)
```

- Mensaje pequeño en la parte inferior como recordatorio.

```python
    root.mainloop()
```

- Inicia el bucle principal de la interfaz gráfica. Sin esto, la ventana no se mostraría.

```python
if __name__ == "__main__":
    menu_principal()
```

- Este bloque asegura que menu_principal() solo se ejecute si el archivo es ejecutado directamente (no si es importado).

---

2. Archivo: distancia_manual.py

Este script permite al usuario hacer clic en dos puntos de una imagen y calcular la distancia entre ellos.

```python
import cv2
import numpy as np
import math
```

- cv2: Biblioteca OpenCV para procesamiento de imágenes.
- numpy: Necesario para manejar matrices (imágenes).
- math: Para cálculos matemáticos (raíz cuadrada).

```python
puntos = []
```

- Lista vacía para almacenar las coordenadas (x, y) de los puntos que el usuario seleccione.

```python
def click_event(event, x, y, flags, param):
    global puntos
    if event == cv2.EVENT_LBUTTONDOWN:
        if len(puntos) < 2:
            puntos.append((x, y))
            print(f"Punto seleccionado: ({x}, {y})")
            cv2.circle(img_copy, (x, y), 5, (0, 0, 255), -1)
            cv2.imshow("Selecciona dos puntos", img_copy)
```

- Esta función se activa cuando el usuario hace clic con el botón izquierdo del mouse.
- Si aún no se han seleccionado dos puntos, guarda las coordenadas.

- Dibuja un círculo rojo en el punto seleccionado.
- Actualiza la imagen mostrada.

```python
            if len(puntos) == 2:
                x1, y1 = puntos[0]
                x2, y2 = puntos[1]
                distancia_pixeles = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
                escala_cm_por_px = 0.05
                distancia_cm = distancia_pixeles * escala_cm_por_px
```

- Cuando se tienen dos puntos, calcula la distancia en píxeles usando la fórmula de distancia euclidiana.
- Luego la convierte a centímetros multiplicando por una escala (ajustable).

```python
                cv2.line(img_copy, puntos[0], puntos[1], (255, 0, 0), 2)
                mid_x = (x1 + x2) // 2
                mid_y = (y1 + y2) // 2
                cv2.putText(img_copy, f"{distancia_cm:.2f} cm", (mid_x, mid_y),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 0), 2)
```

- Dibuja una línea azul entre los dos puntos.
- Coloca un texto amarillo con la distancia en el centro.

```python
                cv2.imwrite("resultado_con_puntos.jpg", img_copy)
                with open("resultados.txt", "w") as f:
                    f.write(f"Distancia en píxeles: {distancia_pixeles:.2f}\n")
                    f.write(f"Distancia en centímetros: {distancia_cm:.2f} cm\n")
```

- Guarda la imagen con los puntos y la línea.
- Guarda los resultados en un archivo de texto.

```python
img = cv2.imread("circulos1.jpeg")
if img is None:
    print("Error: No se pudo cargar la imagen.")
    exit()
```

- Carga la imagen. Si no existe, muestra un error y termina.

```python
img_copy = img.copy()
cv2.imshow("Selecciona dos puntos", img_copy)
cv2.setMouseCallback("Selecciona dos puntos", click_event)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Crea una copia de la imagen para dibujar.
- Muestra la imagen y configura el evento del mouse.
- Espera a que el usuario cierre la ventana.

---

3. Archivo: distancia_automatica.py

Este script detecta círculos automáticamente usando el algoritmo de Hough.

```python
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray, (9, 9), 2)
```

- Convierte la imagen a escala de grises.
- Aplica un desenfoque gaussiano para reducir el ruido.

```python
circles = cv2.HoughCircles(
    gray, cv2.HOUGH_GRADIENT, dp=1.2, minDist=50,
    param1=50, param2=30, minRadius=10, maxRadius=100
)
```

- Usa HoughCircles para detectar círculos.
- dp=1.2: Resolución del acumulador.
- minDist=50: Distancia mínima entre centros.
- param1=50: Umbral para el detector Canny.
- param2=30: Umbral para la detección de círculos (mientras más bajo, más círculos detecta).
- minRadius y maxRadius: Rango de tamaño de círculos.

```python
if circles is not None:
    circles = np.round(circles[0, :]).astype("int")
    if len(circles) >= 2:
        (x1, y1, r1) = circles[0]
        (x2, y2, r2) = circles[1]
```

- Si se detectan círculos, redondea sus coordenadas.
- Toma los dos primeros círculos.

```python
        cv2.circle(img_copy, (x1, y1), r1, (0, 255, 0), 2)
        cv2.circle(img_copy, (x1, y1), 3, (0, 0, 255), -1)
```

- Dibuja un círculo verde alrededor del perímetro.
- Dibuja un punto rojo en el centro.

```python
        distancia_pixeles = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
        distancia_cm = distancia_pixeles * escala_cm_por_px
```

- Calcula la distancia entre centros.

```python
        cv2.imwrite("resultado_con_puntos.jpg", img_copy)
        with open("resultados.txt", "w") as f:
            f.write(f"Distancia en píxeles: {distancia_pixeles:.2f}\n")
            f.write(f"Distancia en centímetros: {distancia_cm:.2f} cm\n")
```

- Guarda los resultados.

---

4. Archivo: enviar_distancia.py

Este script envía los resultados por correo usando Gmail.

```python
remitente = "tu_correo@gmail.com"
destinatario = "destinatario@gmail.com"
password = "contraseña_de_aplicacion"
```

- Datos del correo. Usa una contraseña de aplicación de Google (no tu contraseña normal).

```python
mensaje = MIMEMultipart()
mensaje['Subject'] = "Resultados de Medición"
mensaje['From'] = remitente
mensaje['To'] = destinatario
```

- Crea un mensaje con asunto, remitente y destinatario.

```python
cuerpo = "Se han calculado los resultados..."
mensaje.attach(MIMEText(cuerpo, "plain"))
```

- Agrega el cuerpo del mensaje.

```python
with open("resultado_con_puntos.jpg", "rb") as img:
    imagen_adjunto = MIMEImage(img.read())
    imagen_adjunto.add_header('Content-Disposition', 'attachment', filename="resultado_con_puntos.jpg")
    mensaje.attach(imagen_adjunto)
```

- Adjunta la imagen.

```python
with open("resultados.txt", "rb") as txt_file:
    parte_txt = MIMEBase("application", "octet-stream")
    parte_txt.set_payload(txt_file.read())
    encoders.encode_base64(parte_txt)
    parte_txt.add_header("Content-Disposition", "attachment; filename=resultados.txt")
    mensaje.attach(parte_txt)
```

- Adjunta el archivo de texto.

```python
servidor = smtplib.SMTP_SSL('smtp.gmail.com', 465)
servidor.login(remitente, password)
servidor.send_message(mensaje)
servidor.quit()
```

- Conecta a Gmail, inicia sesión y envía el correo.