Water Body (Rivers, Lakes, Reservoirs) Detection & Mapping

Muhammad Aarish Mughal
Reg Id. **223570**
Department of Computer Science
Multan, Punjab, Pakistan.
aarishmughal21@gmail.com

Aymen Majid
Reg Id. **223574**
Department of Computer Science
Multan, Punjab, Pakistan.
aymenmajid440@gmail.com

Implementation

We have developed a test prototype of our intended application, structured into three abstract-level Python modules. We have implemented a few features related to our project for now. Those features are as follows:

- Colour Channel Histograms
- Contour Detection
- Histogram Analysis Panel
- Zoom and Pan
- Save Filtered Image
- Dynamic Blue Range Tuning

Each module has a distinct role in the architecture of the application.

- Main.py
- Processing.py
- Utils.py

These files contain different pieces of python code that work together to make this application work. These files are also attached separately with the submission. To provide a preview of the implementation, small snippets of the code from each module are shown below.

Main.py

```
import tkinter as tk
from tkinter import filedialog, Label,
Button, Scale, HORIZONTAL, Frame
import cv2
import numpy as np
import matplotlib.pyplot as plt
from processing import
detect_water_and_contours, save_image
from utils import convert_cv_to_tk

class WaterApp:
   def load_image(self):
        path =
filedialog.askopenfilename()
        if not path:
            return

        self.image = cv2.imread(path)
        self.image =
cv2.resize(self.image, (800, 600))
        self.update_detection()

    def update_sliders(self, _=None):
        if self.image is not None:
            self.update_detection()
```

Processing.py

```
import tkinter as tk
from tkinter import filedialog, Label,
Button, Scale, HORIZONTAL, Frame
import cv2
import numpy as np
import matplotlib.pyplot as plt
from processing import
detect_water_and_contours, save_image
from utils import convert_cv_to_tk

class WaterApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Water Body
Detection")
        self.root.geometry("1800x800")
        self.root.configure(bg='#f0f0f0'
)

        self.image = None
```

```
        self.water_only = None
        self.contour_image = None
        self.mask = None

        tk.Label(root, text="Water Body
Detector", font=("Arial", 24, "bold"),
bg='#f0f0f0').pack(pady=10)

        top_frame = Frame(root,
bg='#f0f0f0')
        top_frame.pack(pady=10)

        self.panel_orig =
Label(top_frame)
        self.panel_orig.pack(side="left"
, padx=10)

        self.panel_result =
Label(top_frame)
        self.panel_result.pack(side="lef
t", padx=10)
```

Utils.py

```
import cv2
from PIL import Image, ImageTk

def convert_cv_to_tk(cv_img):
    img_rgb = cv2.cvtColor(cv_img,
cv2.COLOR_BGR2RGB)
    pil_img = Image.fromarray(img_rgb)
    return ImageTk.PhotoImage(pil_img)
```

Debugging-Test-run

At first, the first problem we faced was that the input image could not be opened. After a few trial runs with the input image being in the parent directory of the file **main.py,** we settled on allowing the user to choose the input image file via the Windows Explorer which proved to be very convenient.

Second issue was that the output image would not be shown. We fixed this issue by showing the output image file (the input image after processing is performed), also allowing the user to save this output image file for later analysis.

One of the few other errors included the sliders not working as expected but we fixed those with the help of our AI helper: **ChatGPT**. *(This tool is a life saver when the bugs you face aren't usual or can be fixed via **Stack overflow, etc.**)*

Results analysis

Since this prototype is an initial phase into our project, it obviously isn't the prettiest in terms of design and UI. But the working of the prototype seemed smooth and flawless. A small analysis with application screenshots is attached with this report.

Conclusion

Looking at this project, we believe that this would be impossible without the core concepts that we were taught in the **Digital Image Processing** class. Different concepts like Colour Space Conversion, Binary Masking, Edge Detection, Contour Detection, etc., were all implemented in this project which further improved our concepts.

Future Improvements

Our future intentions with this project are to improve its user interface and improve its UI. Another option we are looking at is the use of **Django** – the python framework for making quick web applications.

Bibliography

- **OpenCV Documentation. (n.d.).** *Open-Source Computer Vision Library*. https://docs.opencv.org/
- **Gonzalez, R. C., & Woods, R. E. (2018).** *Digital Image Processing (4th ed.). Pearson.*

- **Shapiro, L. G., & Stockman, G. C. (2001).**
  Computer Vision. Prentice Hall.

Result Analysis Screenshots