import cv2

import argparse

import numpy as np

import matplotlib.pyplot as plt

parser = argparse.ArgumentParser(description='How many images do you want to process? and the starting index of the images? and the type of images?')

parser.add\_argument('--amount', type=int, default=1, help='amount of images to process', required=True)

parser.add\_argument('--start', type=int, default=1, help='starting index', required=True)

parser.add\_argument('--type', type=str, default="train", help='train or val', required=True)

args = parser.parse\_args()

# Load the image

test = args.amount

test2 = args.start

typerunning = args.type

x = range(int(test))

contain\_Green = 0

contain\_Sea = 0

contain\_Urban = 0

contain\_Highway = 0

contain\_Houses = 0

contain\_Green\_Sea = 0

contain\_Green\_Urban = 0

contain\_Green\_Highway = 0

contain\_Green\_Houses = 0

contain\_Sea\_Urban = 0

contain\_Sea\_Highway = 0

contain\_Sea\_Houses = 0

contain\_Urban\_Highway = 0

contain\_Urban\_Houses = 0

contain\_Highway\_Houses = 0

for n in x:

    index:int = int(test2)+n

    image = cv2.imread("maps\\maps\\"+typerunning+"\_processed\\"+str(index)+".jpg")

    # Get image dimensions

    height, width, \_ = image.shape

    # Define the region of interest (right half of the image)

    roi = image[:, width // 2:]

    # Define the target colors with threshold values

    target\_colors = {

        "Green": (np.array([0, 255, 0]), 20),      # BGR format with threshold

        "Sea": (np.array([255, 0, 0]), 20),        # BGR format with threshold

        "Urban": (np.array([0, 0, 255]), 20),      # BGR format with threshold

        "Highway": (np.array([0, 255, 255]), 20),  # BGR format with threshold

        "Houses": (np.array([255, 255, 0]), 20)    # BGR format with threshold

    }

    # Check if the target colors exist in the right half of the image

    contains\_colors = {key: np.any(np.linalg.norm(roi - color[0], axis=-1) <= color[1]) for key, color in target\_colors.items()}

    count\_colors = {key: sum(contains\_colors[color] for color in contains\_colors if color == key) for key in contains\_colors}

    # Add the counts to the variables

    contain\_Green += count\_colors.get("Green", 0)

    contain\_Sea += count\_colors.get("Sea", 0)

    contain\_Urban += count\_colors.get("Urban", 0)

    contain\_Highway += count\_colors.get("Highway", 0)

    contain\_Houses += count\_colors.get("Houses", 0)

    # Check if two colors are present in the same image

    if contains\_colors["Green"] and contains\_colors["Sea"]:

        contain\_Green\_Sea += 1

    if contains\_colors["Green"] and contains\_colors["Urban"]:

        contain\_Green\_Urban += 1

    if contains\_colors["Green"] and contains\_colors["Highway"]:

        contain\_Green\_Highway += 1

    if contains\_colors["Green"] and contains\_colors["Houses"]:

        contain\_Green\_Houses += 1

    if contains\_colors["Sea"] and contains\_colors["Urban"]:

        contain\_Sea\_Urban += 1

    if contains\_colors["Sea"] and contains\_colors["Highway"]:

        contain\_Sea\_Highway += 1

    if contains\_colors["Sea"] and contains\_colors["Houses"]:

        contain\_Sea\_Houses += 1

    if contains\_colors["Urban"] and contains\_colors["Highway"]:

        contain\_Urban\_Highway += 1

    if contains\_colors["Urban"] and contains\_colors["Houses"]:

        contain\_Urban\_Houses += 1

    if contains\_colors["Highway"] and contains\_colors["Houses"]:

        contain\_Highway\_Houses += 1

print(f"contain\_Green: {contain\_Green}")

print(f"contain\_Sea: {contain\_Sea}")

print(f"contain\_Urban: {contain\_Urban}")

print(f"contain\_Highway: {contain\_Highway}")

print(f"contain\_Houses: {contain\_Houses}")

# Plot the outputs

colors = ['Green', 'Sea', 'Urban', 'Highway', 'Houses']

counts = [contain\_Green, contain\_Sea, contain\_Urban, contain\_Highway, contain\_Houses]

contains\_both = ['G+S', 'G+U', 'G+Hi', 'G+Ho', 'S+U', 'S+Hi', 'S+Ho', 'U+Hi', 'U+Ho', 'Hi+Ho']

counts\_both = [contain\_Green\_Sea, contain\_Green\_Urban, contain\_Green\_Highway, contain\_Green\_Houses, contain\_Sea\_Urban, contain\_Sea\_Highway, contain\_Sea\_Houses, contain\_Urban\_Highway, contain\_Urban\_Houses, contain\_Highway\_Houses]

plt.bar(colors, counts)

plt.xlabel('Colors')

plt.ylabel('Counts')

plt.title('Occurrences of Classes in the Dataset')

plt.show()

plt.bar(contains\_both, counts\_both)

plt.xlabel('Colors')

plt.ylabel('Counts')

plt.title('Occurrences of the same Classes in the Dataset')

plt.show()

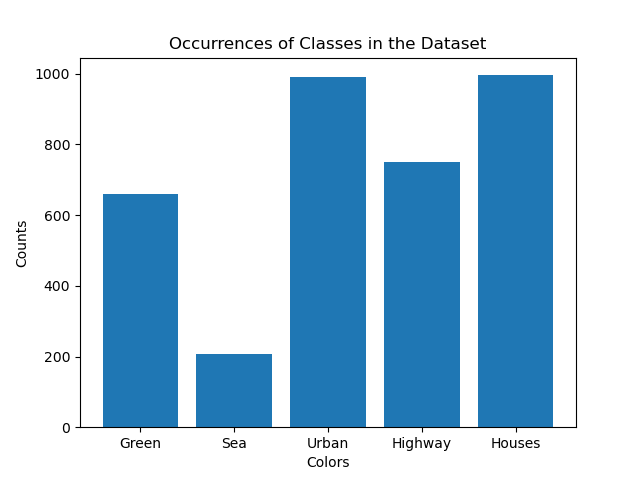
Output Data Validation dataAfbeelding met tekst, schermopname, diagram, Lettertype

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, diagram, Rechthoek

Automatisch gegenereerde beschrijvingAfbeelding met tekst, Lettertype, schermopname

Automatisch gegenereerde beschrijving

Output of Training setAfbeelding met tekst, schermopname, diagram, Lettertype

Automatisch gegenereerde beschrijvingAfbeelding met tekst, Lettertype, schermopname

Automatisch gegenereerde beschrijving