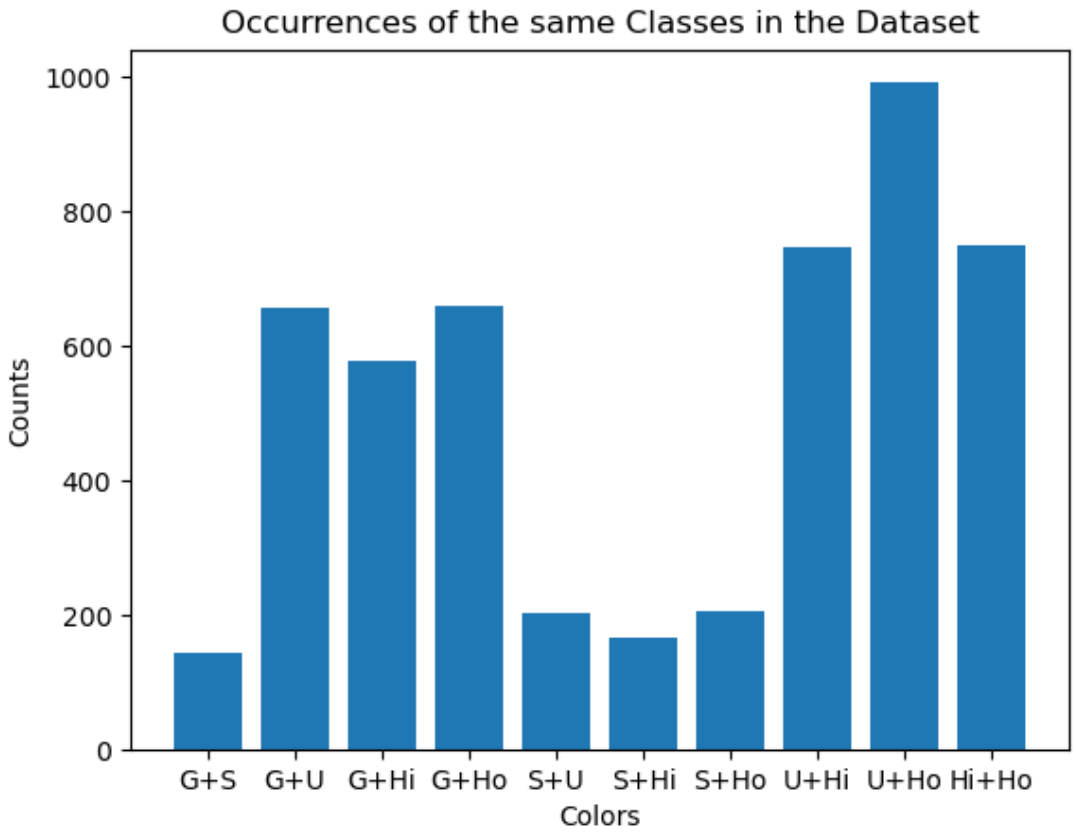
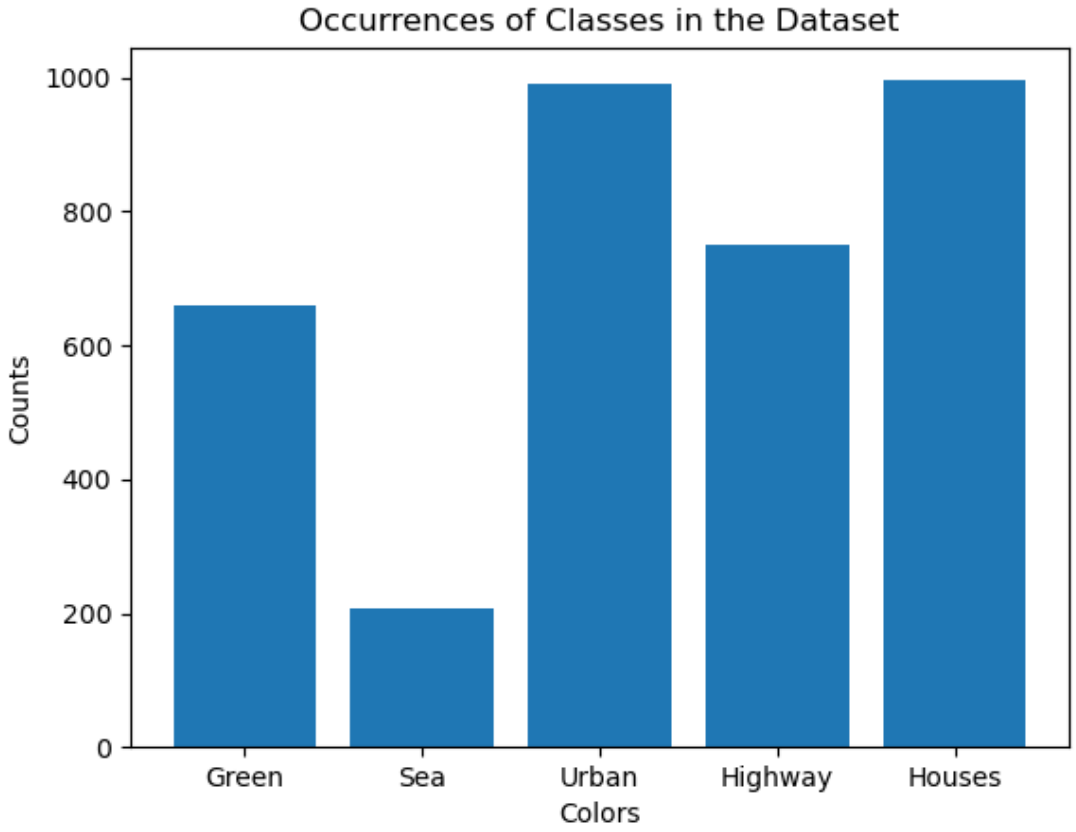
```
import cv2
import argparse
import numpy as np
import matplotlib.pyplot as plt

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()
```

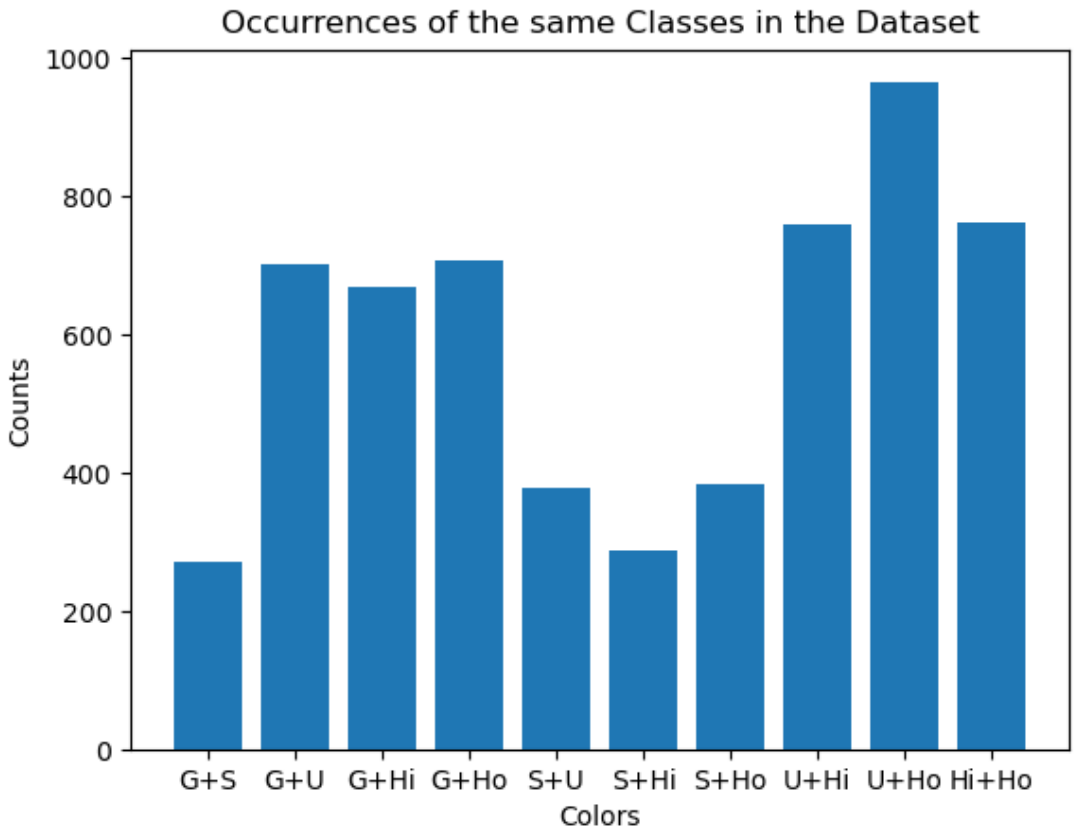
Output Data Validation data

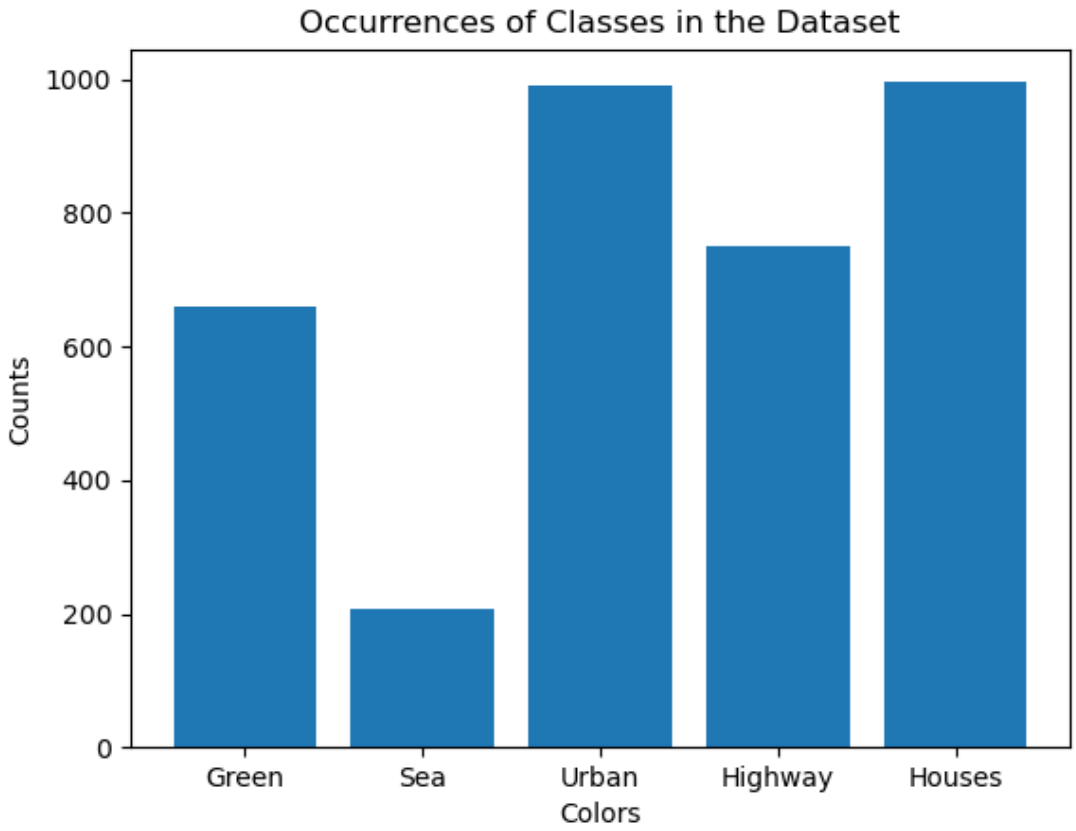





```
contain_Green: 708
contain_Sea: 412
contain_Urban: 964
contain_Highway: 764
contain_Houses: 970
```

Output of Training set






```
contain_Green: 660
contain_Sea: 207
contain_Urban: 992
contain_Highway: 750
contain_Houses: 996
```