the-depths-3d-vision-quest-2024-1

March 28, 2024

1 Image Matching Challenge - Exploratory Data Analysis

Hoş geldiniz! Bu heyecan verici yarışmada, 2D görüntülerden 3D sahneleri oluşturma yeteneğinizi test edeceğiz. Ancak önce bazı önemli bilgilere göz atalım:

2 Eğitim ve Test Veri Kümesi İncelemesi

Hoş geldiniz! Bu veri kümesi, eşsiz mekanlarda çekilmiş bir dizi görüntü içerir. Bazı eğitim veri setleri, "images_full" adlı bir klasörde ek görüntüler içerebilir. Yayınlanan test klasörü ise eğitimdeki kilise sahnesinin bir alt kümesini içerir ve yalnızca örnek amaçlar için sağlanmıştır. Eğitim verisi genellikle ardışık bir sırayla çekilen ve önemli ölçüde içerik örtüşmesine sahip görüntülerden oluşurken, test seti sınırlı görüntü örtüşmesine ve rastgele bir sıraya sahiptir.

2.1 Eğitim Veri Kümesi Klasörleri:

- **images:** Aynı konumun yakınında çekilmiş bir dizi görüntü.
- **smf:** Bu görüntü grubu için bir 3D rekonstrüksiyon, bu yarışma ile birlikte paketlenmiş olan 3D hareket yapısından-colmap adlı kütüphane ile açılabilir.
- **LICENSE.txt:** Bu veri kümesinin lisansı.

2.2 Eğitim Etiketleri CSV Dosyası (`train_labels.csv`):

2.2.1 1 Eğitim Etiketleri İncelemesi:

- **dataset:** Veri kümesi için benzersiz bir tanımlayıcı.
- **scene:** Sahne için benzersiz bir tanımlayıcı.
- **image_path:** Dosya adı ve yolu dahil görüntü dosya adı.
- **rotation_matrix:** İlk hedef sütunu. Satır majörü kurallarına göre düzleştirilmiş, noktalı virgül ile ayrılmış değerler içeren bir 3x3 matris vektörü.
- **translation_vector:** İkinci hedef sütunu. Noktalı virgül ile ayrılmış değerlere sahip 3 boyutlu bir vektör.

2.3 Hedefler:

1. **Hassas 3D Haritalar Oluşturma:** Farklı senaryo ve ortamlardan gelen görüntü setlerinden doğru mekansal temsiller oluşturmak için bir model geliştirmek.
2. **Yapıdan Hareketle (SfM):** Çeşitli görüntülerin bir koleksiyonundan bir ortamın 3D modelini yeniden oluşturma süreci.

3. **Çeşitli Görüntü Kaynaklarını Keşfetme:** Dronlar , yoğun ormanlar ve gece vakti gibi gerçekçi ve uygulanabilir senaryolardan görüntülerle çalışma.

Organizatörler, bu yarışma için farklı zorluklar içeren 6 kategori belirlediler: 1. **Foto Turizmi ve Tarihi Koruma:** Farklı bakış açıları, sensör tipleri, günün/zamanın saati ve örtüler. Antik tarihi siteler eşsiz zorluklar ekler. 2. **Gece ve Gündüz ve Zamansal Değişimler:** Gündüz ve gece fotoğraflarının birleşimi, kötü aydınlatma veya aylar/yıllar arasında farklı hava koşullarında çekilmiş fotoğraflar. 3. **Hava ve Karışık Hava-Yer:** Dronlardan gelen görüntüler, rastgele düzlem dışı dönüşler, benzer görüntülerle eşleştirilmiş ve ayrıca yerden çekilmiş görüntüler. 4. **Tekrarlanan Yapılar:** Simetrik nesneler, perspektifi ayırt etmek için ayrıntılara ihtiyaç duyar. 5. **Doğal Ortamlar:** Ağaçlar ve bitki örtüsü gibi düzensiz yapılar. 6. **Şeffaflıklar ve Yansımalar:** Cam eşyalar gibi doku eksikliği ve farklı bir dizi sorun oluşturan yansımalar ve yansımalar.

Hazır mısınız? Görüntüleri eşleştirmek için hazırlanın ve 3D dünyayı yeniden keşfedin!

3 Install & Import dependencies

[1]:

```
!pip install -q mediapy
```

[2]:

```
%cd /kaggle/working/
!rm -rf /kaggle/working/Hierarchical-Localization
!git clone --quiet --recursive https://github.com/cvg/Hierarchical-Localization/
%cd /kaggle/working/Hierarchical-Localization
!pip install -e .

from hloc import extract_features, match_features, reconstruction,␣
 ↪visualization, pairs_from_exhaustive
from hloc.visualization import plot_images, read_image
from hloc.utils import viz_3d

%cd /kaggle/working/
```

```
/kaggle/working
/kaggle/working/Hierarchical-Localization
Obtaining file:///kaggle/working/Hierarchical-Localization
  Preparing metadata (setup.py) … done
Collecting lightglue@ git+https://github.com/cvg/LightGlue (from
hloc==1.5)
  Cloning https://github.com/cvg/LightGlue to /tmp/pip-install-
irz4p48j/lightglue_5897a12b5eae41f3948b2721381a21ff
  Running command git clone --filter=blob:none --quiet
https://github.com/cvg/LightGlue /tmp/pip-install-
irz4p48j/lightglue_5897a12b5eae41f3948b2721381a21ff
  Resolved https://github.com/cvg/LightGlue to commit
075ae4260fbcb9f0f98cd1743ae72cb3f90a9dae
  Installing build dependencies … done
  Getting requirements to build wheel … done
  Preparing metadata (pyproject.toml) … done
```

```
Requirement already satisfied: torch>=1.1 in
/opt/conda/lib/python3.10/site-packages (from hloc==1.5) (2.1.2+cpu)
Requirement already satisfied: torchvision>=0.3 in
/opt/conda/lib/python3.10/site-packages (from hloc==1.5) (0.16.2+cpu)
Requirement already satisfied: numpy in /opt/conda/lib/python3.10/site-packages
(from hloc==1.5) (1.26.4)
Requirement already satisfied: opencv-python in /opt/conda/lib/python3.10/site-
packages (from hloc==1.5) (4.9.0.80)
Requirement already satisfied: tqdm>=4.36.0 in /opt/conda/lib/python3.10/site-
packages (from hloc==1.5) (4.66.1)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.10/site-
packages (from hloc==1.5) (3.7.5)
Requirement already satisfied: plotly in /opt/conda/lib/python3.10/site-packages
(from hloc==1.5) (5.18.0)
Requirement already satisfied: scipy in /opt/conda/lib/python3.10/site-packages
(from hloc==1.5) (1.11.4)
Requirement already satisfied: h5py in /opt/conda/lib/python3.10/site-packages
(from hloc==1.5) (3.10.0)
Collecting pycolmap>=0.6.0 (from hloc==1.5)
  Downloading pycolmap-0.6.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86
_64.whl.metadata (12 kB)
Requirement already satisfied: kornia>=0.6.11 in /opt/conda/lib/python3.10/site-
packages (from hloc==1.5) (0.7.2)
Collecting gdown (from hloc==1.5)
  Downloading gdown-5.1.0-py3-none-any.whl.metadata (5.7 kB)
Requirement already satisfied: kornia-rs>=0.1.0 in
/opt/conda/lib/python3.10/site-packages (from kornia>=0.6.11->hloc==1.5) (0.1.2)
Requirement already satisfied: packaging in /opt/conda/lib/python3.10/site-
packages (from kornia>=0.6.11->hloc==1.5) (21.3)
Requirement already satisfied: filelock in /opt/conda/lib/python3.10/site-
packages (from torch>=1.1->hloc==1.5) (3.13.1)
Requirement already satisfied: typing-extensions in
/opt/conda/lib/python3.10/site-packages (from torch>=1.1->hloc==1.5) (4.9.0)
Requirement already satisfied: sympy in /opt/conda/lib/python3.10/site-packages
(from torch>=1.1->hloc==1.5) (1.12)
Requirement already satisfied: networkx in /opt/conda/lib/python3.10/site-
packages (from torch>=1.1->hloc==1.5) (3.2.1)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.10/site-packages
(from torch>=1.1->hloc==1.5) (3.1.2)
Requirement already satisfied: fsspec in /opt/conda/lib/python3.10/site-packages
(from torch>=1.1->hloc==1.5) (2024.3.0)
Requirement already satisfied: requests in /opt/conda/lib/python3.10/site-
packages (from torchvision>=0.3->hloc==1.5) (2.31.0)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in
/opt/conda/lib/python3.10/site-packages (from torchvision>=0.3->hloc==1.5)
(9.5.0)
Requirement already satisfied: beautifulsoup4 in /opt/conda/lib/python3.10/site-
packages (from gdown->hloc==1.5) (4.12.2)
```

```
Requirement already satisfied: contourpy>=1.0.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib->hloc==1.5) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.10/site-
packages (from matplotlib->hloc==1.5) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/conda/lib/python3.10/site-packages (from matplotlib->hloc==1.5) (4.47.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib->hloc==1.5) (1.4.5)
Requirement already satisfied: pyparsing>=2.3.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib->hloc==1.5) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/lib/python3.10/site-packages (from matplotlib->hloc==1.5)
(2.9.0.post0)
Requirement already satisfied: tenacity>=6.2.0 in
/opt/conda/lib/python3.10/site-packages (from plotly->hloc==1.5) (8.2.3)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.10/site-
packages (from python-dateutil>=2.7->matplotlib->hloc==1.5) (1.16.0)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.10/site-
packages (from beautifulsoup4->gdown->hloc==1.5) (2.5)
Requirement already satisfied: MarkupSafe>=2.0 in
/opt/conda/lib/python3.10/site-packages (from jinja2->torch>=1.1->hloc==1.5)
(2.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/conda/lib/python3.10/site-packages (from
requests->torchvision>=0.3->hloc==1.5) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/site-
packages (from requests->torchvision>=0.3->hloc==1.5) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/conda/lib/python3.10/site-packages (from
requests->torchvision>=0.3->hloc==1.5) (1.26.18)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.10/site-packages (from
requests->torchvision>=0.3->hloc==1.5) (2024.2.2)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in
/opt/conda/lib/python3.10/site-packages (from requests[socks]->gdown->hloc==1.5)
(1.7.1)
Requirement already satisfied: mpmath>=0.19 in /opt/conda/lib/python3.10/site-
packages (from sympy->torch>=1.1->hloc==1.5) (1.3.0)
Downloading
pycolmap-0.6.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.9
MB)
                         11.9/11.9 MB
59.1 MB/s eta 0:00:00:00:01:01
Downloading gdown-5.1.0-py3-none-any.whl (17 kB)
Building wheels for collected packages: lightglue
  Building wheel for lightglue (pyproject.toml) … done
  Created wheel for lightglue: filename=lightglue-0.0-py3-none-any.whl
size=39472
```

```
sha256=ed94267cde1318124167d3adaff5a65310892f2d8f5f3cf223a31ec096b2d8e0
  Stored in directory: /tmp/pip-ephem-wheel-cache-
py78n76c/wheels/30/34/06/6b38022b3f1bd6489c3cd65367c6a4dddf487443dd2b85ec8e
Successfully built lightglue
Installing collected packages: pycolmap, gdown, lightglue, hloc
  Running setup.py develop for hloc
Successfully installed gdown-5.1.0 hloc-1.5 lightglue-0.0 pycolmap-0.6.1
/kaggle/working
```

[3]:
```python
from pathlib import Path

import cv2
import mediapy
import pandas as pd
import plotly.express as px
import pycolmap
```

4 Advanced Dataset Exploration with Emojis

[4]:
```python
# Path to the train_labels.csv file
train_labels_path = "/kaggle/input/image-matching-challenge-2024/train/
↪train_labels.csv"
```

[5]:
```python
# Load the CSV file into a DataFrame
train_labels_df = pd.read_csv(train_labels_path)
```

[6]:
```python
# Display the first few rows of the DataFrame
print("First few rows of train_labels.csv:")
print(train_labels_df.head())
```

```
First few rows of train_labels.csv:
  image_name                                    rotation_matrix  \
0     00.png  0.999017467386748;-0.01951432487219089;0.03979…
1     01.png  0.999147719991382;-0.021624129414769648;0.0351…
2     02.png  0.9992527616183833;-0.02402019259931326;0.0302…
3     03.png  0.9993946226667176;-0.023560629216676 25;0.0255…
4     04.png  0.9995276708105233;-0.022568162677423 56;0.0208…

                                  translation_vector  \
0  -0.011700149127917355;0.018812528601332625;0.3…
1  -0.011610785964818585;0.016710808069866724;0.3…
2  -0.011589797430545654;0.014113680489915202;0.3…
3  -0.011471598819000773;0.011325953000912126;0.3…
4  -0.011389007765655301;0.008237801582322509;0.3…

                                  calibration_matrix               dataset  \
0  5809.066058292364;0.0;2496.9582994472266;0.0;5…  transp_obj_glass_cup
```

```
1  5809.066058292364;0.0;2496.9582994472266;0.0;5…  transp_obj_glass_cup
2  5809.066058292364;0.0;2496.9582994472266;0.0;5…  transp_obj_glass_cup
3  5809.066058292364;0.0;2496.9582994472266;0.0;5…  transp_obj_glass_cup
4  5809.066058292364;0.0;2496.9582994472266;0.0;5…  transp_obj_glass_cup

                  scene
0  transp_obj_glass_cup
1  transp_obj_glass_cup
2  transp_obj_glass_cup
3  transp_obj_glass_cup
4  transp_obj_glass_cup
```

5 2 Exploring the Interplay Between Datasets and Scenes

[7]:

```
# Calculate the number of unique scenes within each dataset
scenes_per_dataset = train_labels_df.groupby('dataset')['scene'].nunique()
```

[8]:

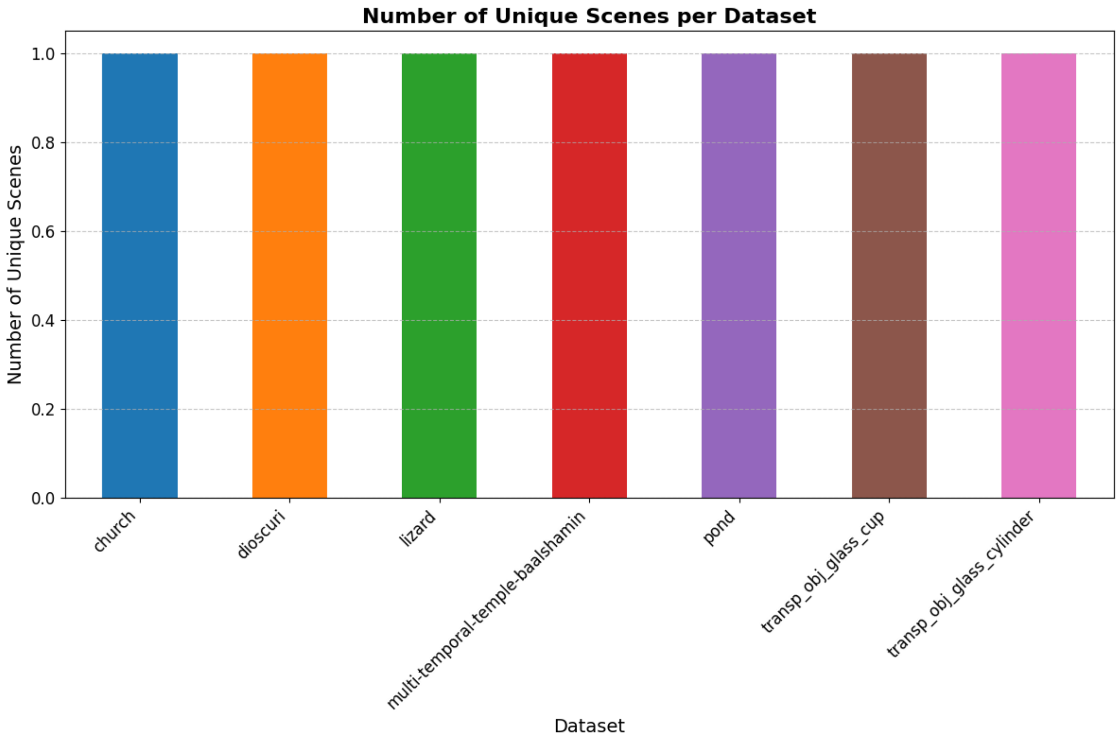
```
train_labels_df.groupby("dataset")["scene"].nunique()
```

[8]:

```
dataset
church                                1
dioscuri                              1
lizard                                1
multi-temporal-temple-baalshamin      1
pond                                  1
transp_obj_glass_cup                  1
transp_obj_glass_cylinder             1
Name: scene, dtype: int64
```

[9]:

```
# Import necessary library for plotting
import matplotlib.pyplot as plt

# Define a custom color palette
colors = plt.cm.tab10.colors

# Visualize the relationship between datasets and scenes
plt.figure(figsize=(12, 8))
scenes_per_dataset.sort_values(ascending=False).plot(kind='bar', color=colors)
plt.title('Number of Unique Scenes per Dataset', fontsize=16, fontweight='bold')
plt.xlabel('Dataset', fontsize=14)
plt.ylabel('Number of Unique Scenes', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



6 Delving into Train Dataset Categories

[10]:
```
train_categories_df = pd.read_csv("/kaggle/input/image-matching-challenge-2024/
 ↪train/categories.csv")
train_categories_df
```

[10]:
```
                              scene                              categories
0                            church                  symmetries-and-repeats
1                          dioscuri  historical_preservation;air-to-ground
2                            lizard                       day-night;temporal
3  multi-temporal-temple-baalshamin       historical_preservation;temporal
4                              pond                day-night;temporal;nature
5               transp_obj_glass_cup     symmetries-and-repeats;transparent
6          transp_obj_glass_cylinder     symmetries-and-repeats;transparent
```

[11]:
```
# Check for missing or corrupted data
train_categories_df.isnull().sum()
```

[11]:
```
scene         0
categories    0
dtype: int64
```

[12]:

```
# Handle missing data appropriately
train_categories_df.dropna(inplace=True)  # We can directly drop the missing␣
 ↪values
```

[13]:

```
# Check the first few observations in the dataset
print(train_categories_df.head())
```

```
                             scene                          categories
0                           church              symmetries-and-repeats
1                         dioscuri  historical_preservation;air-to-ground
2                           lizard                   day-night;temporal
3  multi-temporal-temple-baalshamin     historical_preservation;temporal
4                             pond            day-night;temporal;nature
```

[14]:

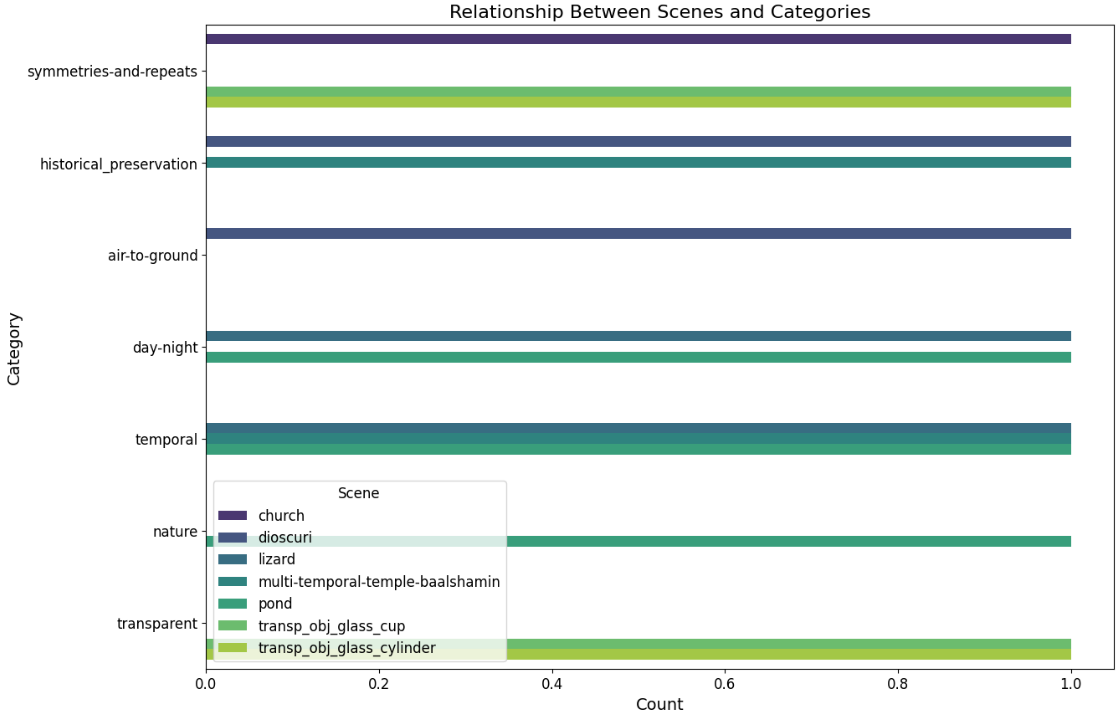
```
# Split the categories column by semicolon and explode it into separate rows
train_categories_df['category'] = train_categories_df['categories'].str.split(';
 ↪')
train_categories_df = train_categories_df.explode('category')

# Drop the original categories column since it's no longer needed
train_categories_df.drop(columns=['categories'], inplace=True)

# Display the updated DataFrame
train_categories_df.head()
```

[14]:

```
      scene                 category
0    church   symmetries-and-repeats
1  dioscuri  historical_preservation
1  dioscuri            air-to-ground
2    lizard                day-night
2    lizard                 temporal
```

[15]:

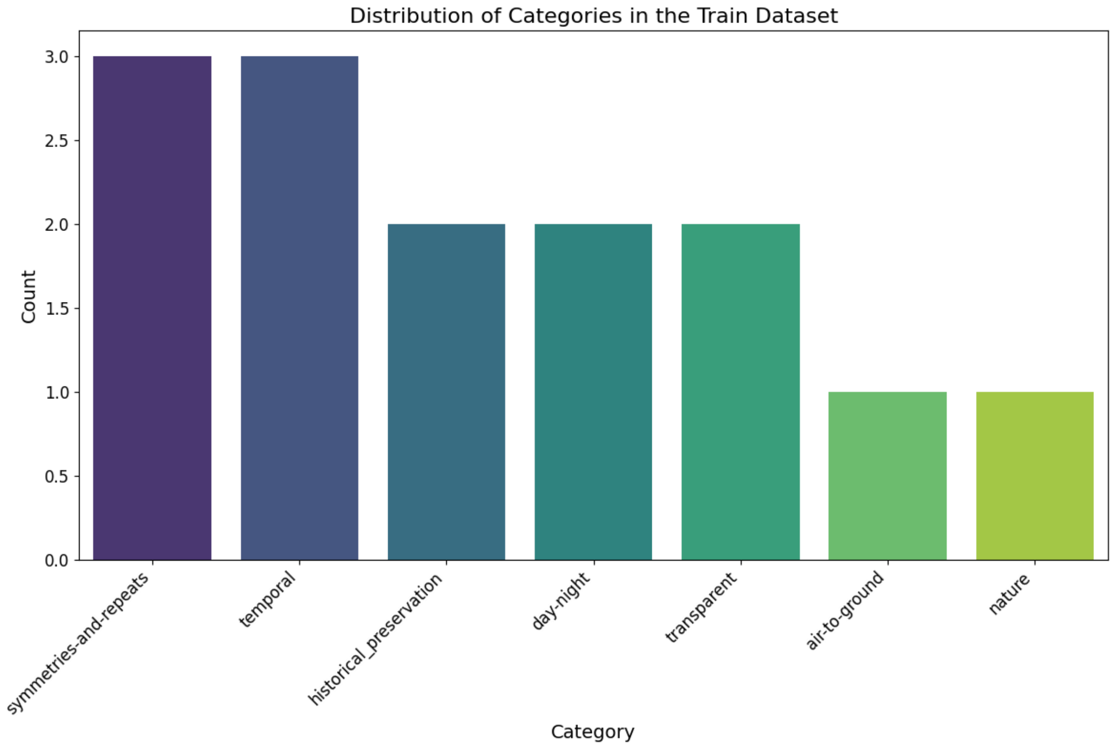
```
import seaborn as sns

# Determine the number of categories
num_categories = len(train_categories_df['category'].unique())

# Visualize the relationship between scenes and categories
plt.figure(figsize=(14, 10))
sns.countplot(y='category', hue='scene', data=train_categories_df,␣
 ↪palette='viridis')
plt.title('Relationship Between Scenes and Categories', fontsize=16)
plt.xlabel('Count', fontsize=14)
plt.ylabel('Category', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.legend(title='Scene', fontsize=12, title_fontsize=12)
plt.show()
```



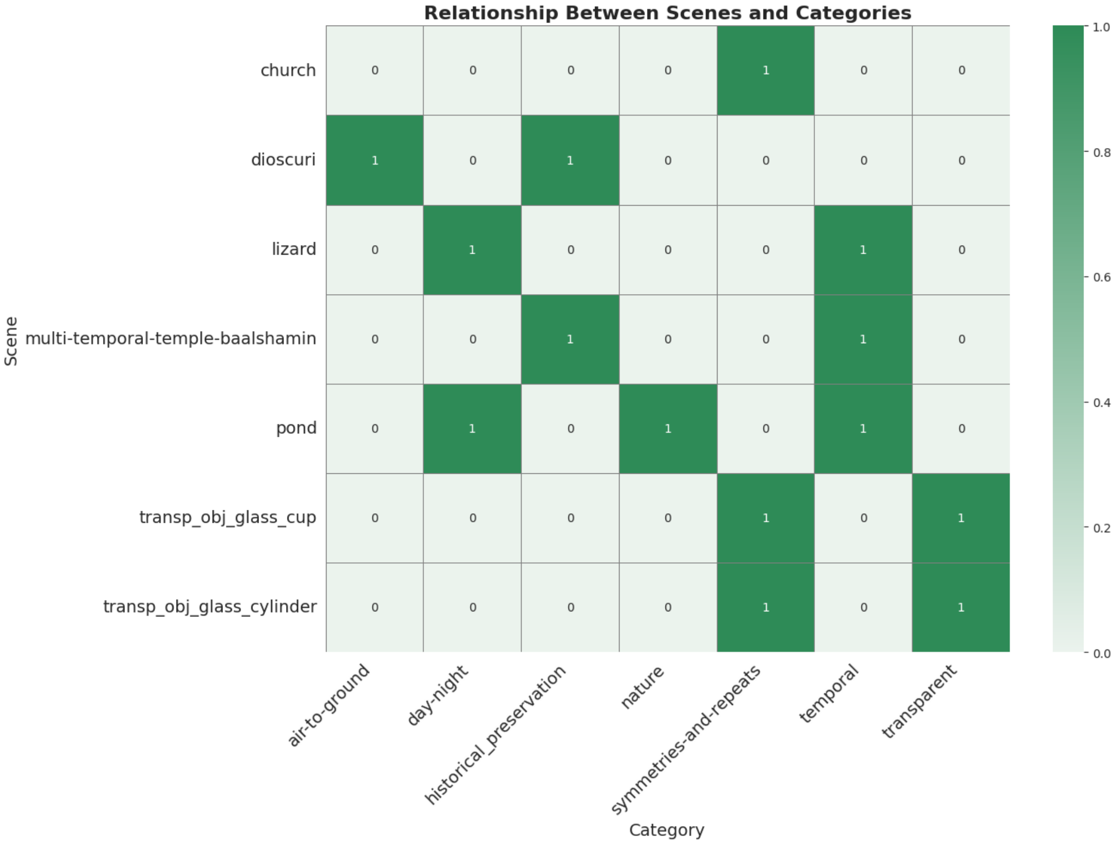
7 Visualizing Train Dataset Categories

[16]:

```
# Count the occurrences of each category
category_counts = train_categories_df['category'].value_counts()

# Create a bar plot
plt.figure(figsize=(12, 8))
sns.barplot(x=category_counts.index, y=category_counts.values,␣
 ↪palette='viridis')
plt.title('Distribution of Categories in the Train Dataset', fontsize=16)
plt.xlabel('Category', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.yticks(fontsize=12)
plt.tight_layout()
plt.show()
```



[17]:
```
import matplotlib.font_manager as fm

# Create a pivot table to count the occurrences of each category in each scene
scene_category_counts = train_categories_df.pivot_table(index='scene',␣
 ↪columns='category', aggfunc='size', fill_value=0)

# Define a custom color palette
custom_palette = sns.light_palette("seagreen", as_cmap=True)

# Set custom font styles
title_font = {'fontname': 'Times New Roman', 'fontsize': 16, 'fontweight':␣
 ↪'bold'}
label_font = {'fontname': 'Arial', 'fontsize': 14}

# Set the figure style
sns.set_style("white")

# Plot the heatmap with improved aesthetics
plt.figure(figsize=(14, 10))
sns.heatmap(scene_category_counts, cmap=custom_palette, cbar=True, linewidths=0.
 ↪5, linecolor='gray', annot=True, fmt='d', annot_kws={'fontsize': 10})
plt.title('Relationship Between Scenes and Categories', **title_font)
plt.xlabel('Category', **label_font)
plt.ylabel('Scene', **label_font)
plt.xticks(rotation=45, ha='right', **label_font)
plt.yticks(**label_font)
plt.gca().patch.set_facecolor('lightgray')  # Set background color
plt.tight_layout()
plt.show()
```



Scene	air-to-ground	day-night	historical_preservation	nature	symmetries-and-repeats	temporal	transparent
church	0	0	0	0	1	0	0
dioscuri	1	0	1	0	0	0	0
lizard	0	1	0	0	0	1	0
multi-temporal-temple-baalshamin	0	0	1	0	0	1	0
pond	0	1	0	1	0	1	0
transp_obj_glass_cup	0	0	0	0	1	0	1
transp_obj_glass_cylinder	0	0	0	0	1	0	1

7.1 Dive into each dataset!

[18]:
```
def explore(split: str, dataset: str, plot_image_limit: int = 12) -> None:
    # Define the path to the dataset
    path = Path("/kaggle/input/image-matching-challenge-2024") / split / dataset
    images_path = path / "images"
    smf_path = path / "smf"

    # Load and display images
    images = [cv2.cvtColor(cv2.imread(str(p)), cv2.COLOR_BGR2RGB) for p in␣
  ↪list(images_path.glob("*"))[:plot_image_limit]]
    mediapy.show_images(images, height=300, columns=3)
    # If not the test split, visualize 3D reconstruction
    if split != "test":
        rec_gt = pycolmap.Reconstruction(smf_path)
        fig = viz_3d.init_figure()
        viz_3d.plot_reconstruction(fig, rec_gt, cameras=False,␣
 ↪color='rgba(227,168,30,0.5)', name="Ground Truth", cs=5)
        fig.show()
```

[19]:
```
explore(split="train", dataset="church")
```

```
<IPython.core.display.HTML object>
```

[20]:
```
explore(split="test", dataset="church")
```

```
<IPython.core.display.HTML object>
```

[21]:
```
explore(split="train", dataset="dioscuri")
```

```
<IPython.core.display.HTML object>
```

[22]:
```
explore(split="train", dataset="lizard")
```

```
<IPython.core.display.HTML object>
```

[23]:
```
explore(split="test", dataset="lizard")
```

```
<IPython.core.display.HTML object>
```

[24]:
```
explore(split="train", dataset="pond")
```

```
<IPython.core.display.HTML object>
```

[25]:
```
explore(split="train", dataset="transp_obj_glass_cup")
```

```
<IPython.core.display.HTML object>
```

[26]:
```
explore(split="train", dataset="transp_obj_glass_cylinder")
```

```
<IPython.core.display.HTML object>
```