SCHOOL OF COMPUTER AND COMMUNICATION SCIENCES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Computer Vision Laboratory
Unseen Spacecraft Pose Estimation

Baseline solution by implementing a machine learning framework with target models included

Bachelor's Thesis in Computer Science

Author:	Jérémy Chaverot
Supervisor:	Prof. Dr. Mathieu Salzmann
Advisor:	Dr. Andrew Price, PhD. Chen Zhao
Semester:	Fall 2023

Summary

This project falls within an Unseen 6 Degrees of Freedom (DoF) competition, organized in collaboration with the European Space Agency (ESA) Advanced Concept Team. Essentially, we are dealing with space objects that are unfamiliar to us, and our objective is to accurately predict their 6DoF poses. The action of the Computer Vision Laboratory (CVLab) team is twofold: firstly, we are tasked with creating a challenging dataset featuring multi-object, unseen, and occluded spacecraft scenarios. This involves ensuring a high degree of rendering realism. Secondly, we are focused on developing a baseline solution, which entails implementing a pose estimation model and conducting thorough training and testing on our dataset. My role this semester was primarily concentrated on the latter aspect, specifically on a track that incorporated target models.

Foremost, we started with a literature search by reading recent papers about generalizable 6DoF object pose estimation[1]. Among the various models we explored, one that particularly captured our attention was the "Generalizable Model-Free 6-DoF Object Pose Estimation from RGB Images," commonly referred to as Gen6D [1]. We have compiled a table summarizing the advantages and disadvantages of this architecture for a clearer understanding:

Table 1: Summary of Gen6D

Pros	Cons
Generalizability	Limited by Reference Image Quality
Model-Free	Everyday Life Objects Training Data
Simple Input Requirements	Difficulty with Symmetric Objects
Robustness to Background Clutter	Dependence on Initial Detection and Selection
Effective in Diverse Environments	Potential Challenges with Severe Occlusions
Competitive Performance	Computationally Intensive

We need to point out that Gen6D does not need a model because it uses a Structure From Motion (SFM) software called *Colmap*. As we are on the track with the target models included, we simply skip this reconstruction step.

After selecting the model, we needed to establish a suitable environment for executing the code: EPFL Scitas Izar servers. Equipped with two NVIDIA V100 PCIe 32 GB GPUs, these servers are ideally configured for our Machine Learning (ML) task. However, this stage proved to be more time-consuming than expected. It entailed various technical challenges, including setting up the virtual environment, installing the necessary dependencies, composing the bash execution script, and, fundamentally, learning the correct way to utilize the server.

After resolving the engineering aspects, we were able to delve into the practical side of the project, working directly with the code to adapt it to our dataset. After understanding the code, the initial task involved developing a data loader specifically for the Spacecraft dataset. This was accompanied by various minor modifications within the code, such as updating names and adjusting data paths. To align with the data format used by Gen6D for poses, it was necessary to write a

[1] `https://github.com/liuyuan-pal/Awesome-generalizable-6D-object-pose`

Python script. This was because our dataset utilized quaternions combined with a translation vector in one text file, whereas Gen6D's format employed a rotation matrix augmented by the translation vector formatted as a separate numpy array for each image.

Next, we tackled the debugging phase, aimed at addressing the model's accuracy issues. We encountered several problems. For instance, we needed to rectify the inverted masks for the objects using a Python script with the OpenCV library. Additionally, the model processes the reference images by cropping them into 128x128 pixel dimensions, while the detector is designed to identify objects in the query images that range from half to double the size of the reference images. Consequently, we resized the query images to reduce the scale difference relative to the reference images. This resizing was accompanied by necessary adjustments to the intrinsic matrix. Also we had to enhance the selection process of the reference images to ensure a more uniform distribution around the objects: contrary to everyday objects, spacecrafts can be observed from any angle, this should be considered in our approach.

One should bear in mind that in this model implementation, we decided to not retrain Gen6D. As a result, the outcomes are now more reliant than ever on the quality of the reference images. Given that we have access to highly realistic renderings of the spacecrafts and precise ground truth poses provided by the dataset team, we aim to assess Gen6D's effectiveness across a range of objects. Below are the visual results for the Hubble object:

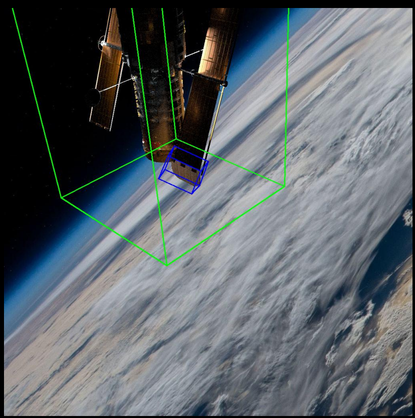

Figure 1: Hubble Space Telescope with earth rendered background, 1024x1024 first query image

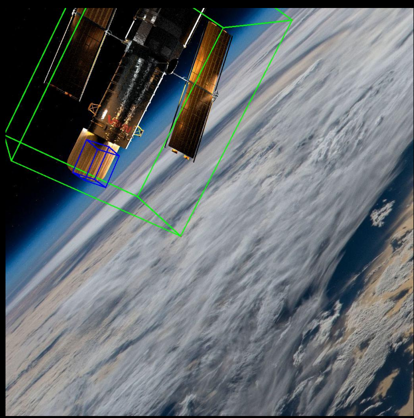

Figure 2: Hubble Space Telescope with earth rendered background, 1024x1024 second query image

Acknowledgments

Before delving into the topic, I'd like to express some acknowledgments. First and foremost I must thank my advisors Andrew Price and Chen Zhao for accepting my

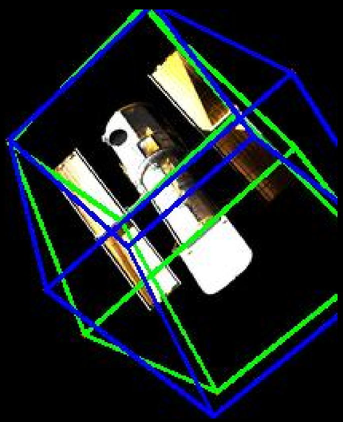

Figure 3: Hubble Space Telescope, no background, 256x256 query image, $e_{\text{ADD}} = 2.925$, $e_{\text{ADD-S}} = 1.183$

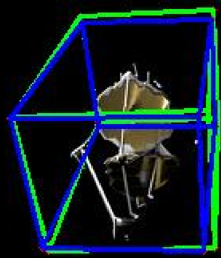

Figure 4: James Webb Space Telescope, no background, 256x256 query image, $e_{\text{ADD}} = 1.415$, $e_{\text{ADD-S}} = 0.808$

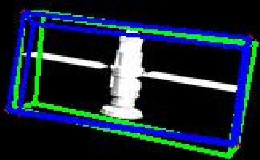

Figure 5: Cosmos Link, no background, 256x256 query image, $e_{\text{ADD}} = 1.718$, $e_{\text{ADD-S}} = 0.383$

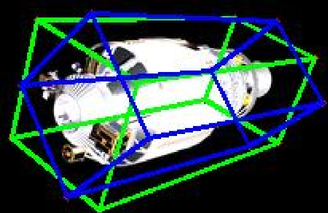

Figure 6: Rocket Body, no background, 256x256 query image, $e_{\text{ADD}} = 1.713$, $e_{\text{ADD-S}} = 0.252$

request to take part in a semester project under their guidance. I am grateful to have been able to practice my skills with them, and can only hope that the feeling is mutual. Moreover I would also like to thoroughly thank my friends and family for supporting me in my academic journey, despite a rather unstable start in my studies.

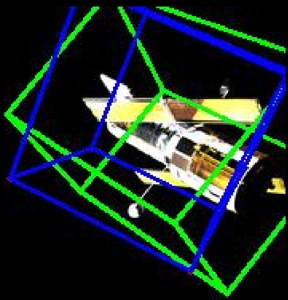

Figure 7: Hubble Space Telescope, no background, 256x256 query image, $e_{\text{ADD}} = 6.514$, $e_{\text{ADD-S}} = 1.571$

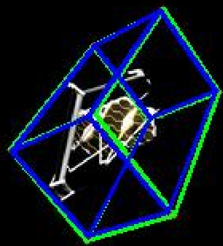

Figure 8: James Webb Space Telescope, no background, 256x256 query image, $e_{\text{ADD}} = 2.224$, $e_{\text{ADD-S}} = 1.261$

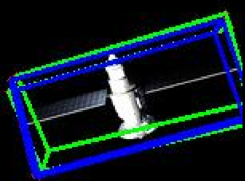

Figure 9: Cosmos Link, no background, 256x256 query image, $e_{\text{ADD}} = 1.925$, $e_{\text{ADD-S}} = 0.377$

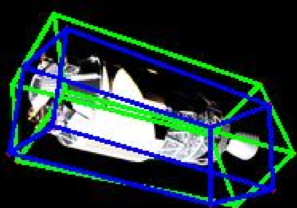

Figure 10: Rocket Body, no background, 256x256 query image, $e_{\text{ADD}} = 1.982$, $e_{\text{ADD-S}} = 0.501$

Figure 11: Hubble Space Telescope, no background, intermediary result, $e_{\text{ADD}} = 9.577$, $e_{\text{ADD-S}} = 5.196$

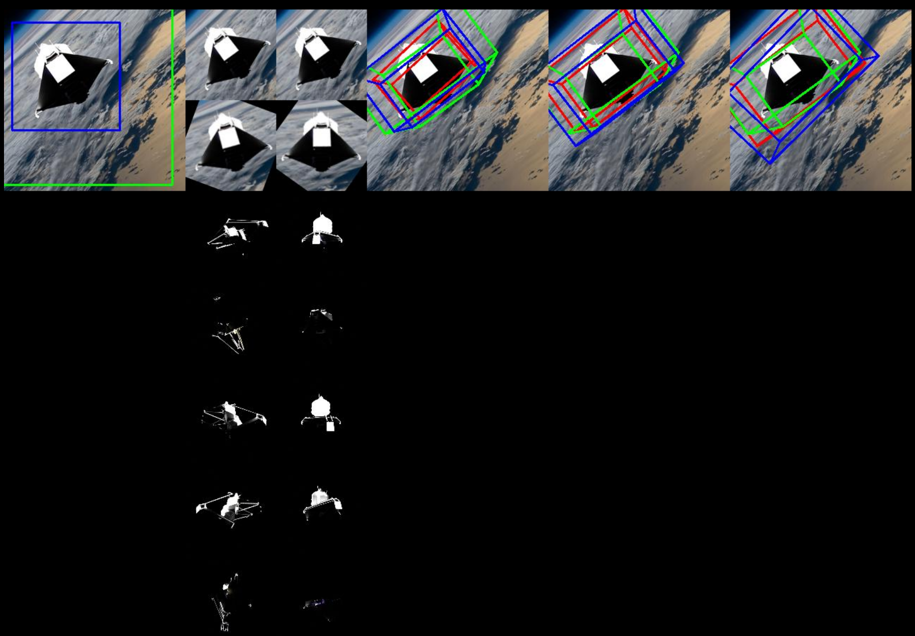

Figure 12: James Webb Space Telescope, with earth rendered background, intermediary result, $e_{\text{ADD}} = 10.934$, $e_{\text{ADD-S}} = 4.317$

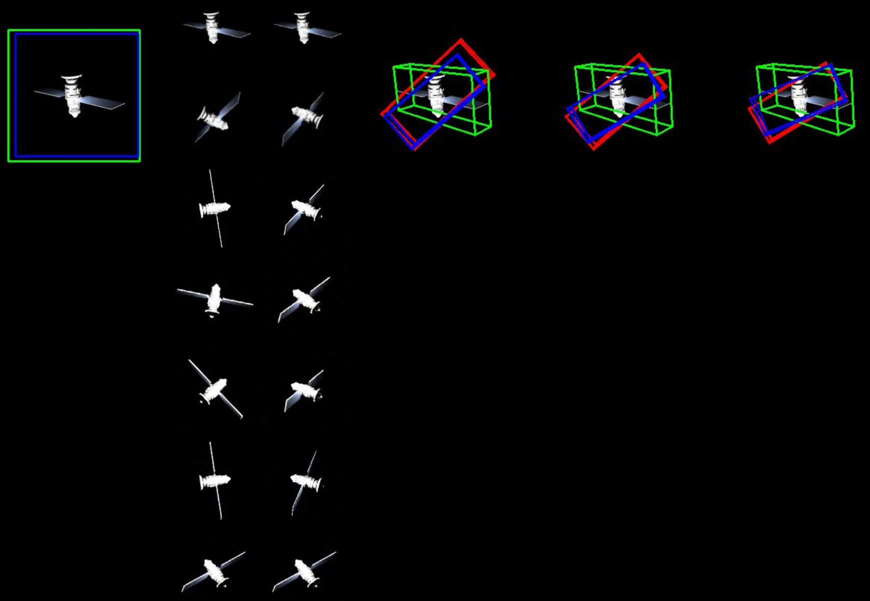

Figure 13: Cosmos Link, no background, intermediary result, $e_{\text{ADD}} = 11.094$, $e_{\text{ADD-S}} = 6.127$

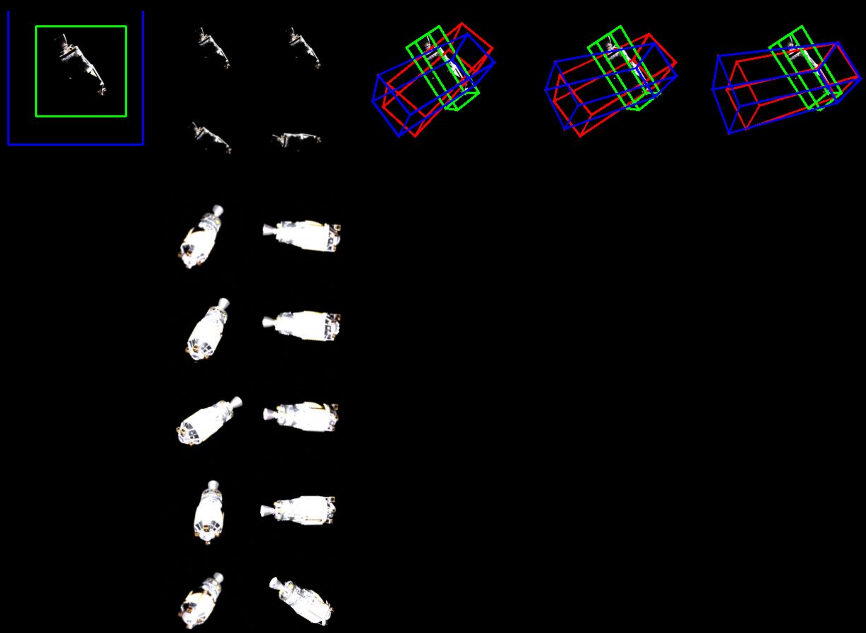

Figure 14: Rocket Body, no background, intermediary result, $e_{\text{ADD}} = 29.335$, $e_{\text{ADD-S}} = 17.743$

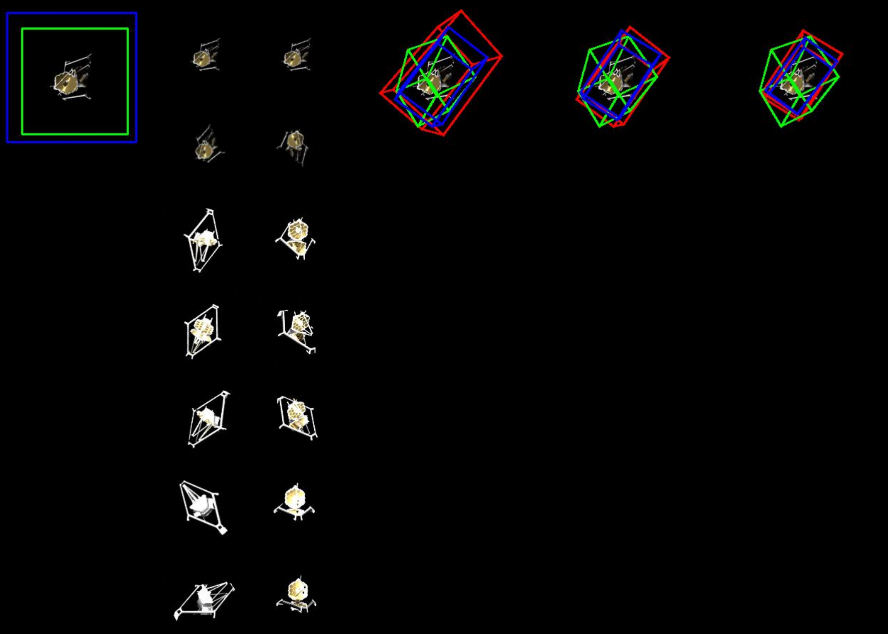

Figure 15: James Webb Space Telescope, with no background, intermediary result, $e_{\text{ADD}} = 21.983$, $e_{\text{ADD-S}} = 12.358$

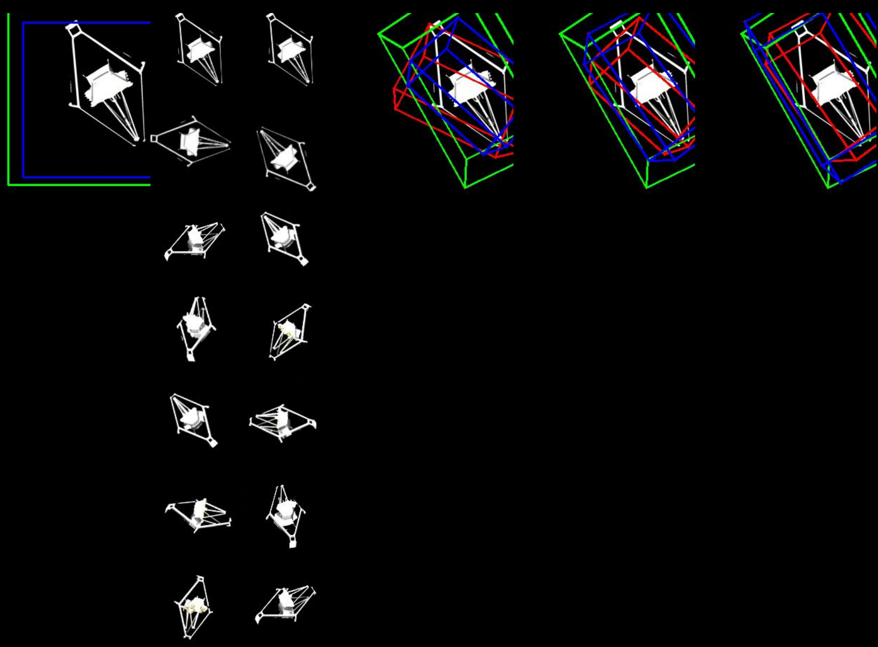

Figure 16: James Webb Space Telescope, with no background, intermediary result, $e_{\text{ADD}} = 1.060$, $e_{\text{ADD-S}} = 0.556$

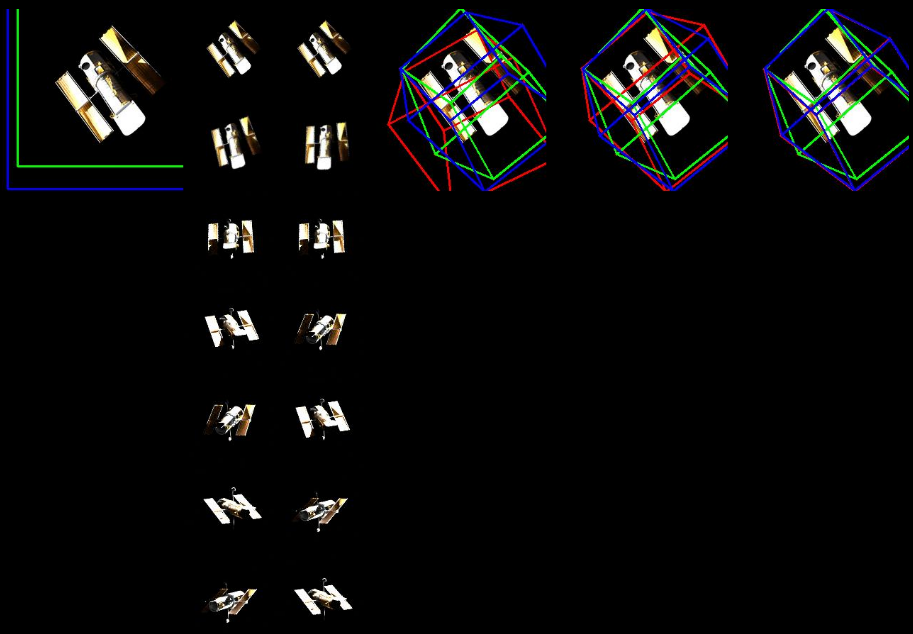

Figure 17: An Estimation by Gen6D (Highlighted in Blue) of the Hubble Object, Presented Without a Background.

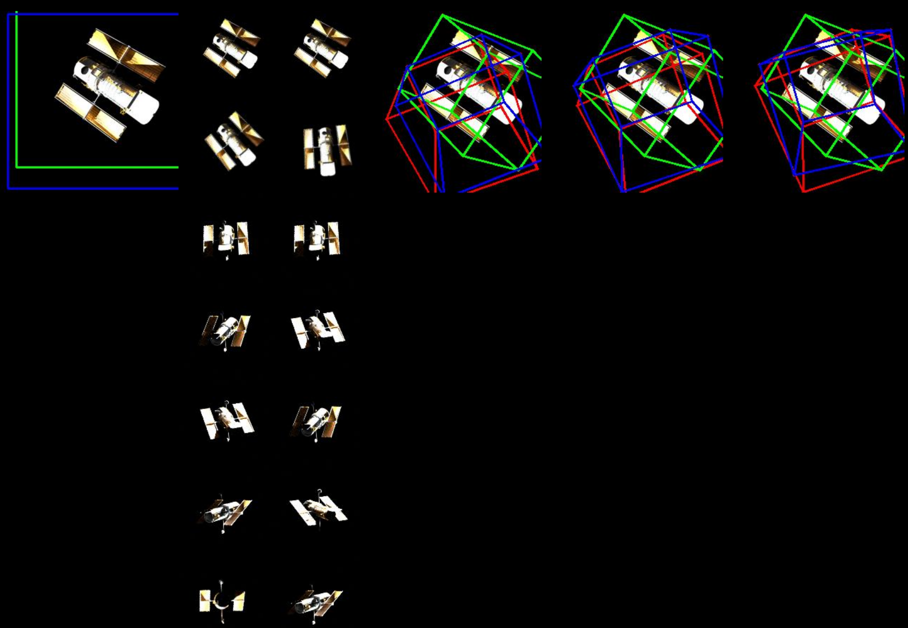

Figure 18: An Estimation by Gen6D (Highlighted in Blue) of the Hubble Object, Presented Without a Background. Gen6D happens to lack of accuracy.

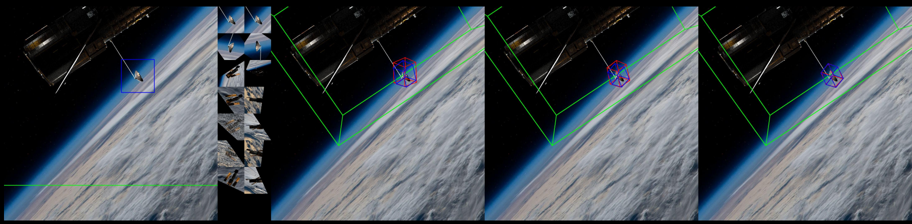

Figure 19: An Estimation by Gen6D (Highlighted in Blue) of the Hubble Object, Presented With a Background. Here the query object size is too big for the detector.

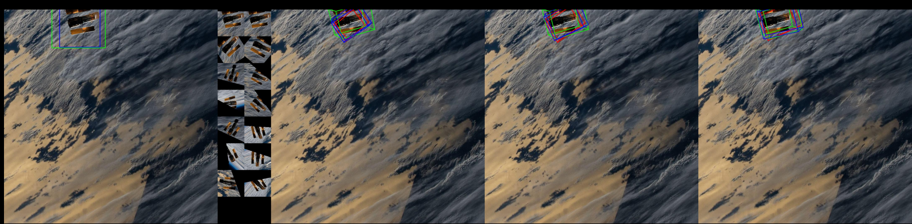

Figure 20: An Estimation by Gen6D (Highlighted in Blue) of the Hubble Object, Presented With a Background.

Abstract

Contents

1 Introduction

Test ref to Listing A.1. Test ref to Listing A.3 Test ref to Gen6D [1]

1.1 Problem statement

1.1.1 The settings

1.1.2 The goal

1.2 The work environment: Scitas Izar

Test to refer to the video from 3B1B: [2].

```
#!/bin/bash
#SBATCH --chdir /scratch/izar/jchavero
#SBATCH --partition=gpu
#SBATCH --qos=gpu_free
#SBATCH --gres=gpu:2
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --mem 16G

echo STARTING AT `date`

echo "Loading modules"
module load gcc openmpi py-torch py-torchvision cuda

echo "Launching the virtual environment"
source ~/opt/izar1/venv-gcc/bin/activate

echo "Navigating to the directory and executing the task"
cd ~/Gen6D
python eval.py --cfg configs/gen6d_pretrain.yaml --object_name
    spacecraft/hubble

echo FINISHED AT `date`
```

Listing 1.1: Bash script `execute.sh` to run a machine learning model on Scitas Izar EPFL. While the overall structure remains consistent, this script is specific to Gen6D's architecture, further discussed later.

Then to run the script we use the following command:

```
$ sbatch execute.sh
```

Listing 1.2: Linux command to run the bash script.

2 Scientific papers review

2.1 Some ML models

2.2 Gen6D: Pros and cons

3 Gen6D: formal description

3.1 Overview of the network

3.2 Detection

3.3 Viewpoint selection

3.4 Pose refinement

3.5 Results on LINEMOD

4 Implementation of the model

4.1 Data loader

abstract base classes (ABC) each and every abstract method

4.2 Issues and proposed solutions

4.2.1 Issues No. 1

4.2.2 Issues No. 2

5 Experimental results and analysis

5.1 Spacecraft dataset characteristics

5.2 Vizualisation of results

5.3 Evaluation metrics

To appreciate the quality of the estimations, the most widely used pose error functions are the Average Distance of Model Points (ADD) and the Average Closest Point Distance (ADD-S) metrics, both introduced by Hinterstoisser et al. [3]. For an object model $\mathcal{M}$, we compute the average distance to the corresponding model point. Therefore the error of an estimated pose $\hat{\mathbf{P}} = (\hat{\mathbf{R}}, \hat{\mathbf{T}})$ w.r.t. the ground truth pose $\bar{\mathbf{P}} = (\bar{\mathbf{R}}, \bar{\mathbf{T}})$ is calculated as follows:

$$e_{\text{ADD}}(\hat{\mathbf{P}}, \bar{\mathbf{P}}, \mathcal{M}) = \underset{\mathbf{x}\in\mathcal{M}}{\text{avg}} \left\| \bar{\mathbf{P}}\mathbf{x}^{\star} - \hat{\mathbf{P}}\mathbf{x}^{\star} \right\|_2 \quad ^{1} \tag{5.1}$$

$$= \underset{\mathbf{x}\in\mathcal{M}}{\text{avg}} \left\| (\bar{\mathbf{R}}\mathbf{x} + \bar{\mathbf{T}}) - (\hat{\mathbf{R}}\mathbf{x} + \hat{\mathbf{T}}) \right\|_2 \tag{5.2}$$

When the model $\mathcal{M}$ has symmetries that leads to no indistinguishable views, the error is computed as the average distance to the closest model point:

$$e_{\text{ADD-S}}(\hat{\mathbf{P}}, \bar{\mathbf{P}}, \mathcal{M}) = \underset{\mathbf{x}_1\in\mathcal{M}}{\text{avg}} \; \underset{\mathbf{x}_2\in\mathcal{M}}{\min} \left\| \bar{\mathbf{P}}\mathbf{x}_1^{\star} - \hat{\mathbf{P}}\mathbf{x}_2^{\star} \right\|_2 \tag{5.3}$$

$$= \underset{\mathbf{x}_1\in\mathcal{M}}{\text{avg}} \; \underset{\mathbf{x}_2\in\mathcal{M}}{\min} \left\| (\bar{\mathbf{R}}\mathbf{x}_1 + \bar{\mathbf{T}}) - (\hat{\mathbf{R}}\mathbf{x}_2 + \hat{\mathbf{T}}) \right\|_2 \tag{5.4}$$

It's important to point out that $e_{\text{ADD-S}}$ is more lenient compared to e_{ADD}, and should only be applied in cases where there is a definite presence of symmetry in the object and the estimated pose is already notably precise. Otherwise, using $e_{\text{ADD-S}}$ becomes irrelevant since the estimation is advantaged.

5.4 Quantitative evaluation

[1] In this context, the vector $\mathbf{x}^{\star}$ represents a vector that has been extended by appending a 1, specifically for the purpose of matrix multiplication.

6 Ways of improvements

6.1 Specialized spacecraft training set

6.2 Improved object detection algorithms

Rely more on the 3D model (for now only the size) and the segmented images, would optimize for symmetric and irregular shaped spacecrafts

6.3 Robustness to occlusion

7 Conclusion

Limitations Acknowledgments My personal contribution

Abbreviations

ESA European Space Agency

DoF Degrees of Freedom

CVLab Computer Vision Laboratory

SFM Structure From Motion

ML Machine Learning

ADD Average Distance of Model Points

ADD-S Average Closest Point Distance

Appendix

```
"""
Author:      Jeremy Chaverot
Date:        November 29, 2023
Description: Create the files val.txt, train.txt and test.txt
    according to a test percentage
"""

import os
import sys
import random


if __name__ == "__main__":

  # Check if the correct number of arguments is provided
    if len(sys.argv) != 3:
        print("Usage: python format.py <object_name> <
    test_percentage>")
        sys.exit(1)

    object = sys.argv[1]
    test_percentage = float(sys.argv[2])

    if (test_percentage < 0 or 1 < test_percentage):
        print("Wrong value for the variable <test_percentage>.
    Should be between 0 and 1 included.")
        sys.exit(1)

    # Get a list of all files in the folder
    all_files = os.listdir(f'data/SpaceCraft/{object}/images')

    # Filter the list to include only image files and exclude
    MacOS temporary files
    image_files = [file for file in all_files if file.lower().
    endswith(('.jpg')) and not file.startswith('._')]

    # Get the number of images in the folder
    num_images = len(image_files)

    # Iterate through each image and apply the transformation
```

```
    with open(f'data/SpaceCraft/{object}/train.txt', 'w') as
    train, open(f'data/SpaceCraft/{object}/test.txt', 'w') as
    test:
        for image_file in image_files:
            rand = random.random()
            image_path = 'SpaceCraft/hubble/images/' + image_file
            if (rand < test_percentage):
                test.write(image_path + '\n')
            else: train.write(image_path + '\n')

    print(f"Done splitting {num_images} images in train.txt and
    test.txt")
```

Listing A.1: Python script `format.py` to randomly generate the training set and the test set based on a specified probability. Should be run from Gen6D's root folder.

```
"""
Author:      Jeremy Chaverot
Date:        November 20, 2023
Description: Transform every images of a folder into jpg format.
"""

import os
import sys
from PIL import Image


def transform_image(image_path):
    img = Image.open(image_path)
    new_image_path = image_path.split('.')[0] + '.jpg'
    img.save(new_image_path)


if __name__ == "__main__":

  # Check if the correct number of arguments is provided
    if len(sys.argv) != 2:
        print("Usage: python to_jpg.py </path/to/your/images>")
        sys.exit(1)

    folder_path = sys.argv[1]

    # Get a list of all files in the folder
    all_files = os.listdir(folder_path)

    # Filter the list to include only image files and exclude
    MacOS temporary files
    image_files = [file for file in all_files if file.lower().
    endswith(('.png', '.jpg', '.jpeg', '.gif', '.bmp')) and not
```

```
    file.startswith('._')]

    # Get the number of images in the folder
    num_images = len(image_files)

    # Iterate through each image and apply the transformation
    for image_file in image_files:
        image_path = os.path.join(folder_path, image_file)
        transform_image(image_path)
        os.remove(image_path)

    print(f"Number of images transformed into .jpg: {num_images}")
```

Listing A.2: Python script `to_jpg.py` to transform every images of a specified folder into `jpg` format.

```
"""
Author:      Jeremy Chaverot
Date:        November 20, 2023
Description: Transform a txt file with quaternions and the translation vector into multiple npy files containing the rotation matrix augmented with the translation vector.
"""

import numpy as np
import sys
import os


def quaternion_to_matrix(Q, translation):
    """
        Covert a quaternion and translation into a full three-dimensional augmented rotation matrix.

        Input
        :param Q: A 4 element array representing the quaternion (qw, qx, qy, qz).
        :param translation: A 3 element array representing the translation (x, y, z).

        Output
        :return: A 3x4 element matrix representing the full 3D rotation matrix with
                 translation. This rotation matrix converts a point in the local
                 reference frame to a point in the global reference frame.
    """
```

```

    # Extract the values from Q
    qw = Q[0]
    qx = Q[1]
    qy = Q[2]
    qz = Q[3]

    # Extract the values from the translation vector
    x = translation[0]
    y = translation[1]
    z = translation[2]

    # First row of the rotation matrix
    r00 = 2 * (qw * qw + qx * qx) - 1
    r01 = 2 * (qx * qy - qw * qz)
    r02 = 2 * (qx * qz + qw * qy)

    # Second row of the rotation matrix
    r10 = 2 * (qx * qy + qw * qz)
    r11 = 2 * (qw * qw + qy * qy) - 1
    r12 = 2 * (qy * qz - qw * qx)

    # Third row of the rotation matrix
    r20 = 2 * (qx * qz - qw * qy)
    r21 = 2 * (qy * qz + qw * qx)
    r22 = 2 * (qw * qw + qz * qz) - 1

    # 3x3 rotation matrix
    rot_matrix_augm = np.array([[r00, r01, r02, x],
                                [r10, r11, r12, y],
                                [r20, r21, r22, z]])

    return rot_matrix_augm


if __name__ == "__main__":

    # Check if the correct number of arguments is provided
    if len(sys.argv) != 3:
        print("Usage: python quaternion_to_matrix.py </path/to/
    your/text/file> </path/to/the/pose/folder>")
        sys.exit(1)

    file_path = sys.argv[1]
    pose_folder_path = sys.argv[2]
    file_content = None

    try:
        with open(file_path, 'r') as file:
```

```python
            file_content = file.read()
    except FileNotFoundError:
        print(f"The file {file_path} was not found.")
        sys.exit(1)
    except Exception as e:
        print(f"An error occurred: {e}")
        sys.exit(1)

    poses = file_content.split('\n')[:-1]

    # Iterate through each pose and apply the transformation
    for pose in poses:
        image_id, obj_id, qw, qx, qy, qz, x, y, z = pose.split(',')
        Q = np.array([qw, qx, qy, qz], dtype=np.float32)
        translation = np.array([x, y, z], dtype=np.float32)
        matrix = quaternion_to_matrix(Q, translation)
        np.save(pose_folder_path + '/pose' + str(int(image_id)), matrix)

    print(f"Number of transformation processed: {len(poses)}")
```

Listing A.3: Python script `quaternion_to_matrix.py` to transform a txt file with quaternions and the translation vector into multiple npy files containing the rotation matrix augmented with the translation vector.

```python
"""
Author:      Jeremy Chaverot
Date:        December 10, 2023
Description: Invert the masks from a given folder.
"""

import cv2
import os
import sys


def inverse_masks_in_folder(folder_path):
  # Iterate through the list of files at the specified path
    for filename in os.listdir(folder_path):
      # Filter to include only png image files and exclude MacOS temporary files
        if filename.endswith(".png") and not filename.startswith('._'):
            mask_path = os.path.join(folder_path, filename)
            try:
                # Read the mask image
                mask = cv2.imread(mask_path, cv2.IMREAD_GRAYSCALE)
```

```python
                if mask is None:
                    print(f"Failed to read image: {mask_path}")
                    continue

                # Invert the mask
                inverted_mask = cv2.bitwise_not(mask)

                # Save the inverted mask with a temporary name
                temp_path = os.path.join(folder_path, "temp_" + filename)
                cv2.imwrite(temp_path, inverted_mask)

                # Delete the original mask
                os.remove(mask_path)

                # Rename the inverted mask to the original filename
                os.rename(temp_path, mask_path)
                print(f"Inverted and replaced mask for: {mask_path}")
            except Exception as e:
                print(f"Error processing {mask_path}: {e}")


if __name__ == "__main__":

  # Check if the correct number of arguments is provided
    if len(sys.argv) != 2:
        print("Usage: python invert_mask.py <folder_path>")
        sys.exit(1)

    folder_path = sys.argv[1]
    inverse_masks_in_folder(folder_path)
```

Listing A.4: Python script `invert_mask.py` to invert the masks from a specified folder. We aim to have a black object set against a white background.

```python
"""
Author:      Jeremy Chaverot
Date:        January 01, 2024
Description: Resize the images from a given folder.
"""

import os
import sys
from PIL import Image


def resize_images(folder_path, resize_factor):
```

```
  # Iterate through the list of files at the specified path
    for filename in os.listdir(folder_path):
      # Filter to include only png image files and exclude MacOS temporary files
        if filename.endswith(".png") and not filename.startswith('._'):
            img_path = os.path.join(folder_path, filename)
            with Image.open(img_path) as img:
                # Calculate new size
                new_size = tuple([int(dim / resize_factor) for dim in img.size])
                # Resize the image
                resized_img = img.resize(new_size, Image.ANTIALIAS)
                # Save the resized image with a different name temporarily
                temp_path = os.path.join(folder_path, "temp_" + filename)
                resized_img.save(temp_path)

            # Delete the original image
            os.remove(img_path)

            # Rename the resized image to the original filename
            os.rename(temp_path, img_path)


if __name__ == "__main__":

  # Check if the correct number of arguments is provided
    if len(sys.argv) != 3:
        print("Usage: resize.py <folder_path> <resize_factor>")
        sys.exit(1)

    folder_path = sys.argv[1]
    factor = int(sys.argv[2])

    resize_images(folder_path, factor)
```

Listing A.5: Python script `resize.py` designed to alter an image's size with respect to a specified resize factor.

Bibliography

[1] Y. Liu, Y. Wen, S. Peng, C. Lin, X. Long, T. Komura, and W. Wang. *Gen6D: Generalizable Model-Free 6-DoF Object Pose Estimation from RGB Images*. 2023. arXiv: 2204.10776.

[2] G. S. 3Blue1Brown. *Quaternions and 3d rotation, explained interactively*. Youtube. 2018. URL: https://www.youtube.com/watch?v=zjMuIxRvygQ.

[3] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. "Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes." In: *Computer Vision – ACCV 2012*. Springer Berlin Heidelberg, 2013, pp. 548–562.