



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

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the advisors.

Lausanne, Switzerland, 05.01.24

Jérémy Chaverot

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 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.

Abstract

Contents

Acknowledgments	ii
Abstract	iii
1 Introduction	1
1.1 Problem statement	1
1.1.1 The settings	1
1.1.2 The goal	1
1.2 The work environment: Scitas Izar	1
2 Scientific papers review	2
2.1 Some ML models	2
2.2 Gen6D: Pros and cons	2
3 Gen6D: formal description	3
3.1 Overview of the network	3
3.2 Detection	3
3.3 Viewpoint selection	3
3.4 Pose refinement	3
3.5 Results on <i>LINEMOD</i>	3
4 Implementation of the model	4
4.1 Data loader	4
4.2 Issues and proposed solutions	4
4.2.1 Issues No. 1	4
4.2.2 Issues No. 2	4
5 Experimental results and analysis	5
5.1 Spacecraft dataset characteristics	5
5.2 Vizualisation of results	5
5.3 Evaluation metrics	5
5.4 Quantitative evaluation	5
6 Ways of improvements	6
6.1 Specialized spacecraft training set	6
6.2 Improved object detection algorithms	6
6.3 Robustness to occlusion	6
7 Conclusion	7
Abbreviations	iv
Appendix	v

Bibliography

xii

1 Introduction

Test ref to Listing A.1. Test ref to Listing A.3

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].

```
1 #!/bin/bash
2 #SBATCH --chdir /scratch/izar/jchavero
3 #SBATCH --partition=gpu
4 #SBATCH --qos=gpu_free
5 #SBATCH --gres=gpu:2
6 #SBATCH --nodes=1
7 #SBATCH --ntasks-per-node=1
8 #SBATCH --cpus-per-task=1
9 #SBATCH --mem 16G
10
11 echo STARTING AT `date`
12
13 echo "Loading modules"
14 module load gcc openmpi py-torch py-torchvision cuda
15
16 echo "Launching the virtual environment"
17 source ~/opt/izar1/venv-gcc/bin/activate
18
19 echo "Navigating to the directory and executing the task"
20 cd ~/Gen6D
21 python eval.py --cfg configs/gen6d_pretrain.yaml --object_name
    spacecraft/hubble
22
23 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:

```
1 $ 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:

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

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

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:

$$e_{\text{ADD-S}}(\hat{\mathbf{P}}, \bar{\mathbf{P}}, \mathcal{M}) = \text{avg}_{\mathbf{x}_1 \in \mathcal{M}} \min_{\mathbf{x}_2 \in \mathcal{M}} \left\| \bar{\mathbf{P}}\mathbf{x}_1^* - \hat{\mathbf{P}}\mathbf{x}_2^* \right\|_2 \quad (5.3)$$

$$= \text{avg}_{\mathbf{x}_1 \in \mathcal{M}} \min_{\mathbf{x}_2 \in \mathcal{M}} \left\| (\bar{\mathbf{R}}\mathbf{x}_1 + \bar{\mathbf{T}}) - (\hat{\mathbf{R}}\mathbf{x}_2 + \hat{\mathbf{T}}) \right\|_2 \quad (5.4)$$

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

¹In this context, the vector \mathbf{x}^* 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

ADD Average Distance of Model Points

ADD-S Average Closest Point Distance

Appendix

```
1  """
2  Author:      Jeremy Chaverot
3  Date:        November 29, 2023
4  Description: Create the files val.txt, train.txt and test.txt
               according to a test percentage
5  """
6
7  import os
8  import sys
9  import random
10
11
12  if __name__ == "__main__":
13
14      # Check if the correct number of arguments is provided
15      if len(sys.argv) != 3:
16          print("Usage: python format.py <object_name> <
17               test_percentage>")
18          sys.exit(1)
19
20      object = sys.argv[1]
21      test_percentage = float(sys.argv[2])
22
23      if (test_percentage < 0 or 1 < test_percentage):
24          print("Wrong value for the variable <test_percentage>.
25               Should be between 0 and 1 included.")
26          sys.exit(1)
27
28      # Get a list of all files in the folder
29      all_files = os.listdir(f'data/SpaceCraft/{object}/images')
30
31      # Filter the list to include only image files and exclude
32      # MacOS temporary files
33      image_files = [file for file in all_files if file.lower().
34                     endswith(('.jpg')) and not file.startswith('.') ]
35
36      # Get the number of images in the folder
37      num_images = len(image_files)
38
39      # Iterate through each image and apply the transformation
```

```

36 with open(f'data/SpaceCraft/{object}/train.txt', 'w') as
    train, open(f'data/SpaceCraft/{object}/test.txt', 'w') as
    test:
37     for image_file in image_files:
38         rand = random.random()
39         image_path = 'SpaceCraft/hubble/images/' + image_file
40         if (rand < test_percentage):
41             test.write(image_path + '\n')
42         else: train.write(image_path + '\n')
43
44 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.

```

1  """
2  Author:      Jeremy Chaverot
3  Date:        November 20, 2023
4  Description: Transform every images of a folder into jpg format.
5  """
6
7  import os
8  import sys
9  from PIL import Image
10
11
12 def transform_image(image_path):
13     img = Image.open(image_path)
14     new_image_path = image_path.split('.')[0] + '.jpg'
15     img.save(new_image_path)
16
17
18 if __name__ == "__main__":
19
20     # Check if the correct number of arguments is provided
21     if len(sys.argv) != 2:
22         print("Usage: python to_jpg.py </path/to/your/images>")
23         sys.exit(1)
24
25     folder_path = sys.argv[1]
26
27     # Get a list of all files in the folder
28     all_files = os.listdir(folder_path)
29
30     # Filter the list to include only image files and exclude
31     # MacOS temporary files
32     image_files = [file for file in all_files if file.lower().
33                     endswith(('.png', '.jpg', '.jpeg', '.gif', '.bmp')) and not

```



```

32     file.startswith('.')])
33
34     # Get the number of images in the folder
35     num_images = len(image_files)
36
37     # Iterate through each image and apply the transformation
38     for image_file in image_files:
39         image_path = os.path.join(folder_path, image_file)
40         transform_image(image_path)
41         os.remove(image_path)
42
43     print(f"Number of images transformed into .jpg: {num_images}")
44 )

```

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

```

1  """
2  Author:      Jeremy Chaverot
3  Date:        November 20, 2023
4  Description: Transform a txt file with quaternions and the
5                translation vector into multiple npy files containing the
6                rotation matrix augmented with the translation vector.
7  """
8
9  import numpy as np
10 import sys
11 import os
12
13 def quaternion_to_matrix(Q, translation):
14     """
15     Covert a quaternion and translation into a full three-
16     dimensional augmented rotation matrix.
17
18     Input
19     :param Q: A 4 element array representing the quaternion (
20               qw, qx, qy, qz).
21     :param translation: A 3 element array representing the
22                       translation (x, y, z).
23
24     Output
25     :return: A 3x4 element matrix representing the full 3D
26             rotation matrix with
27             translation. This rotation matrix converts a
28             point in the local
29             reference frame to a point in the global
30             reference frame.
31     """

```

```

25
26     # Extract the values from Q
27     qw = Q[0]
28     qx = Q[1]
29     qy = Q[2]
30     qz = Q[3]
31
32     # Extract the values from the translation vector
33     x = translation[0]
34     y = translation[1]
35     z = translation[2]
36
37     # First row of the rotation matrix
38     r00 = 2 * (qw * qw + qx * qx) - 1
39     r01 = 2 * (qx * qy - qw * qz)
40     r02 = 2 * (qx * qz + qw * qy)
41
42     # Second row of the rotation matrix
43     r10 = 2 * (qx * qy + qw * qz)
44     r11 = 2 * (qw * qw + qy * qy) - 1
45     r12 = 2 * (qy * qz - qw * qx)
46
47     # Third row of the rotation matrix
48     r20 = 2 * (qx * qz - qw * qy)
49     r21 = 2 * (qy * qz + qw * qx)
50     r22 = 2 * (qw * qw + qz * qz) - 1
51
52     # 3x3 rotation matrix
53     rot_matrix_augm = np.array([[r00, r01, r02, x],
54                                [r10, r11, r12, y],
55                                [r20, r21, r22, z]])
56
57     return rot_matrix_augm
58
59
60 if __name__ == "__main__":
61
62     # Check if the correct number of arguments is provided
63     if len(sys.argv) != 3:
64         print("Usage: python quaternion_to_matrix.py </path/to/your/text/file> </path/to/the/pose/folder>")
65         sys.exit(1)
66
67     file_path = sys.argv[1]
68     pose_folder_path = sys.argv[2]
69     file_content = None
70
71     try:
72         with open(file_path, 'r') as file:

```

```

73         file_content = file.read()
74     except FileNotFoundError:
75         print(f"The file {file_path} was not found.")
76         sys.exit(1)
77     except Exception as e:
78         print(f"An error occurred: {e}")
79         sys.exit(1)
80
81     poses = file_content.split('\n')[:-1]
82
83     # Iterate through each pose and apply the transformation
84     for pose in poses:
85         image_id, obj_id, qw, qx, qy, qz, x, y, z = pose.split(',')
86
87         Q = np.array([qw, qx, qy, qz], dtype=np.float32)
88         translation = np.array([x, y, z], dtype=np.float32)
89         matrix = quaternion_to_matrix(Q, translation)
90         np.save(pose_folder_path + '/pose' + str(int(image_id)),
91               matrix)
92
93     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.

```

1  """
2  Author:      Jeremy Chaverot
3  Date:        December 10, 2023
4  Description: Invert the masks from a given folder.
5  """
6
7  import cv2
8  import os
9  import sys
10
11
12  def inverse_masks_in_folder(folder_path):
13      # Iterate through the list of files at the specified path
14      for filename in os.listdir(folder_path):
15          # Filter to include only png image files and exclude MacOS
16          # temporary files
17          if filename.endswith(".png") and not filename.startswith(
18              '._'):
19              mask_path = os.path.join(folder_path, filename)
20              try:
21                  # Read the mask image
22                  mask = cv2.imread(mask_path, cv2.IMREAD_GRAYSCALE)
23              except:
24                  pass
25      )

```

```

21         if mask is None:
22             print(f"Failed to read image: {mask_path}")
23             continue
24
25         # Invert the mask
26         inverted_mask = cv2.bitwise_not(mask)
27
28         # Save the inverted mask with a temporary name
29         temp_path = os.path.join(folder_path, "temp_" +
filename)
30         cv2.imwrite(temp_path, inverted_mask)
31
32         # Delete the original mask
33         os.remove(mask_path)
34
35         # Rename the inverted mask to the original
filename
36         os.rename(temp_path, mask_path)
37         print(f"Inverted and replaced mask for: {
mask_path}")
38     except Exception as e:
39         print(f"Error processing {mask_path}: {e}")
40
41
42 if __name__ == "__main__":
43
44     # Check if the correct number of arguments is provided
45     if len(sys.argv) != 2:
46         print("Usage: python invert_mask.py <folder_path>")
47         sys.exit(1)
48
49     folder_path = sys.argv[1]
50     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.

```

1  """
2  Author:      Jeremy Chaverot
3  Date:       January 01, 2024
4  Description: Resize the images from a given folder.
5  """
6
7  import os
8  import sys
9  from PIL import Image
10
11
12 def resize_images(folder_path, resize_factor):

```

```

13 # Iterate through the list of files at the specified path
14 for filename in os.listdir(folder_path):
15     # Filter to include only png image files and exclude MacOS
    temporary files
16     if filename.endswith(".png") and not filename.startswith(
        '._'):
17         img_path = os.path.join(folder_path, filename)
18         with Image.open(img_path) as img:
19             # Calculate new size
20             new_size = tuple([int(dim / resize_factor) for
    dim in img.size])
21             # Resize the image
22             resized_img = img.resize(new_size, Image.
    ANTIALIAS)
23             # Save the resized image with a different name
    temporarily
24             temp_path = os.path.join(folder_path, "temp_" +
    filename)
25             resized_img.save(temp_path)
26
27             # Delete the original image
28             os.remove(img_path)
29
30             # Rename the resized image to the original filename
31             os.rename(temp_path, img_path)
32
33
34 if __name__ == "__main__":
35
36     # Check if the correct number of arguments is provided
37     if len(sys.argv) != 3:
38         print("Usage: resize.py <folder_path> <resize_factor>")
39         sys.exit(1)
40
41     folder_path = sys.argv[1]
42     factor = int(sys.argv[2])
43
44     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

- [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.