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: PhD. 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 we dive into the real subject, I would like to make a few acknowledgements. 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

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

```
#!/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. This script is specified to Gen6D's architecture, further discussed later.

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

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

5.4 Quantitative evaluation

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

Appendix

```python
"""
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.
    """
```

```
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:
```

```
            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.

```
"""
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
    )
```

```
                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.

```
"""
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.