Alma Mater Studiorum - University of Bologna

COMPUTER SCIENCE AND ENGINEERING - DISI

ARTIFICIAL INTELLIGENCE

A study on tackling visual odometry by a transformer architecture

Master degree thesis

Supervisor

Prof. Luigi Di Stefano

Co-supervisor

Luca De Luigi

Candidate

Xiaowei Wen

ACADEMIC YEAR 2021-2022- SECOND SESSION

Dedicato a nonna Giovanna e a tutte le persone che mi hanno aiutato a essere qui.

Summary

This dissertation describes a deepening study about Visual Odometry problem tackled with transformer architectures. The initial objectives were: create a synthetic dataset using BlenderProc2 framework, try different versions of transformer architectures which includes: ResNet feature-extractor with encoder, ResNet feature-extractor with encoder-decoder, ResNet-feature extractor with encoder-decoder and pose Auto-encoder.

"Dio benedica quelle persone che quando incroci il loro sguardo per sbaglio, sorridono."

Thanks

Bologna, 06 October 2022 Xiaowei Wen

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this section will be summarized the content of the whole thesis.

1.1 Background

In the recent years, the field of computer vision has been growing in complexity and the number of applications has been increasing, main applications are: image classification, object detection, face recognition, image segmentation, Simultaneous Localization and Mapping (SLAM) and visual odometry which is a task in which the robot is able to understand where it is and how it is oriented.

The development of computer vision has been a long process, the growth is favoured by the development of new hardware components and new challenges, about the latters, we have CIFAR-10 (Doon et al. [2]), Fashion-MNIST(Xiao et al. [12]), MS-Coco (Lin et al. [7]) and ImageNet (Deng et al. [1]). These datasets are often used as benchmark for novel models.

Starting from AlexNet (Krizhevsky et al. [6]), then VGG (Simonyan et al. [8]), Inception-V1 (Szegedy et al. [9]), Inception-V2(Szegedy et al. [10]), ResNet (He et al. [5]), etc., the complexity of the models has increased enormously, but in the same time improving the state-of-the-art. Each of these models introduced some innovations and improved the performance on the benchmarks, for example:

- AlexNet introduced the concept of the *convolutional neural network* (CNN) and use of the separation of the models into two different GPUs.
- VGG introduced the concept of stages, which repeated more times, composes the model.
- Inception-V1 and Inception-V2 which are based on the concept of *inception*

module which was composed by 4 different paths that the input has to go through to reach the output.

- ResNet is a model that is based on the concept of *residual network* which is composed by several blocks of the same type with the skip connections:

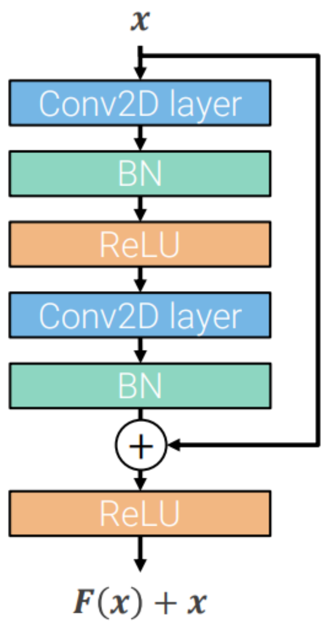

Figure 1.1: Skip connection

Basically, the input of the block is added to the output before feeding it to the next block, in this way, we can avoid the vanishing gradient problem making easier the training process.

After this, the computer-visionist lend the Transformer architecture (Vaswani et al. [11]) to the field of vision, bringing up Vit (Dosovitskiy et al. [3]) which is based on the Multi-Head Attention (MHA) mechanism.

1.2 Problem

The term "odometry" originated from two Greek works *hodos* (meaning "journery" or "travel") and *metron* (meaning "measure"). This derivation is related to the estimation of the change in a robot's pose (translation and rotation) over time. Mobile robot use data from motion sensor to estimate their position relative to their initial location, this is called odometry. VO is a technique used to localize a robot by using only a stream of images acquired from a single or multiple camera. There are different ways to classify the typology of Visual Odometry:

- based on the camera setup:
  - Monocular VO: using only one camera;
  - Stereo VO: using two cameras;
- based on the information:
  - Feature based method: which extracts the image feature points and tracks them in the image sequence;
  - Direct method: a novel method which uses the pixel intensity in the image sequence directly as visual input.
  - Hybrid method: which combines the two methods.
- Visual inertial odometry: if a Inertial measurement unit (IMU) is used within the VO system, it is commonly referred to as Visual inertial odometry.

We can represent the pose in different ways, for example: **euler angles**, **quaternions**, **SO(3)**, **rotation matrices** combined with **translation vectors**.

The goal is to create a Neural network (NN), based on the transformer presented by Vaswani et al. [11], which is able to estimate the camera's pose relative to the initial one, in this project given sequence of images.

1.3 Solution

To solve the problem of visual odometry, we tried different approaches to feed the data into the model, and to construct the model itself. We tried following approaches to feed the data:

- Feeding the sequence into the model directly and presenting the pose as *euler angles*.
- Feeding the sequence into the model directly and presenting the pose as *rotation matrix* and *translation vector*.
- Feeding the sequence into the model where the first frame is the origin of the reference frame and presenting the pose as *euler angles*.
- Feeding the sequence into the model where the first frame is the origin of the reference frame and presenting the pose as *rotation matrix* and *translation vector*.

We used these input strategies to feed the sequence, we tried different variants of models:

- Using the different version of ResNet (producing embedding of size 512 and 2048) model as feature extractor attached to the transformer with only encoder part.
- Using a Multi Layer Perceptron (MLP) for the images divided into patches to extract the features, and concatenating all the features as a single embedding then feed it into the only-encoder version of transformer.
- Using a MLP for the images divided into patches to extract the features, and concatenating all the features as a single embedding then feed it into the encoder-decoder version of transformer.
- Using the different version of ResNet model as feature extractor attached to the transformer with encoder and decoder.
- The same model as the previous but implementing it in auto-regressive mode.

1.4 Results

1.4.1 Full sequence prediction

The different models we achieved different results, but the most important result is that we settled down an important baseline for the future development of this system.

The encoder-only version of transformer, and feeding the *sequence 3* of Kitti (for more details § 3.1) where the first image is considered as origin, we showed that the model is able to learn a single sequence, in overfitting, and the model fails when trying to overfit a more complex sequence.

The encoder-decoder version we achieved the same results as the previous version, but the model was able to learn the also the *sequence 7* of Kitti, but it fails when we train the model with both *sequence 3* and *sequence 7*.

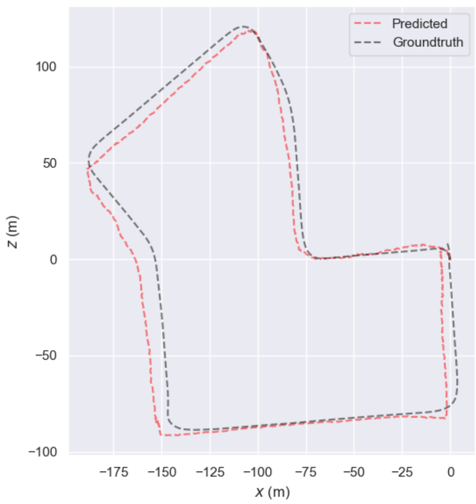

Figure 1.2: Good prediction sequence 7

1.4.2 Autoregressive models

We implemented only the encoder-decoder version of the transformer in the autoregressive way, and the results are similar to those of full-sequence predictions.

1.5 Thesis Organization

First chapter introduces the general content about thesis and gives a short presentation of the project and the results;

Second chapter a deepening about the theoretical foundations used during the stage and the project;

Third chapter presents the datasets used during for the training and the testing of the model;

Fourth chapter presents the state-of-the-art of Visual Odometry;

Fifth chapter presents the experiments did during to develop the system;

Sixth chapter presents the different implementations of the system;

Seventh chapter discusses about the results and possible future developments.

During the drafting of the essay, following typography conventions are considered:

- the acronyms, abbreviations, ambiguous terms or terms not in common use are defined in the glossary, in the end of the present document;

- the first occurrences of the terms in the glossary are highlighted like this: word;
- the terms from the foreign language or jargon are highlighted like this: *italics*.

Chapter 2

Theoretical foundations

In this chapter will be presented the main theoretical knowledge useful to understand the content from successive chapters.

2.1 Deep Learning

2.1.1 Convolutional Neural Network

2.1.2 Transformer

2.2 Visual Odometry

Visual Odometry is an important task in robotics' computer vision field, because it allows the robot to understand where it is and how it is oriented.

Chapter 3

Datasets

In this chapter will be presented the datasets created and used for the visual odometry.

3.1 Kitti

The odometry benchmark consists of 22 stereo sequences, saved in loss less png format: 11 sequences are provided with ground truth trajectories for training and 11 sequences (11-21) without ground truth trajectories for evaluation.

This odometry benchmark is a subset of KITTI Vision Benchmark suite.

3.1.1 Scene

The images represents a various of scenes from mid-sized city, rural areas and on highways.

Figure 3.1: KITTI - example of scene

3.1.2 Image generation

Each sequence of the KITTI dataset is composed of by four sequences of images: left-coloured, right-coloured, left-grey and right-grey. Each one is captured by a camera mounted on the top of vehicle. They calibrated the four video cameras intrinsically and extrinsically and rectified the input image. Then they computed the 3D rigid motion parameters which relate the coordinate system of the laser scanner.

Meanwhile, the ground-truth is directly given by the output of the GPS/IMU localization unit projected into the coordinate system of the left camera after rectification.

3.1.3 Dataset statistics

The dataset consists of 22 stereo sequences, with a total length of 39.2 km, which was the longest in the time of the publication of the paper. In the dataset, there are no specifically indicate which sequence is used for training, validation or testing, but in this work, the dataset is split as this:

Sets	N. of Sequence	N. Image
Training set	8	20.098
Validation set	2	1.902
Test set	1	1.201
Total	11	23.201

Table 3.1: KITTI - dataset statistics

The images dimensions about 1240x370 are slightly different, generally varying for few pixels.

3.1.4 Usage

This dataset is the one mainly used, as it is the one of the most famous and most used in the literature.

The sequence **3** and **7** are used for evaluation and testing, because they are the easier ones.

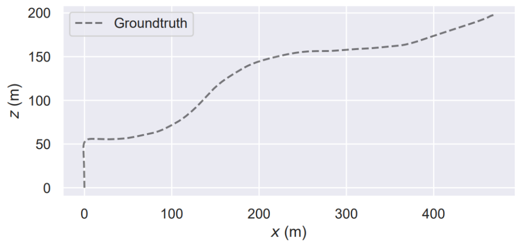

Figure 3.2: KITTI - sequence 3

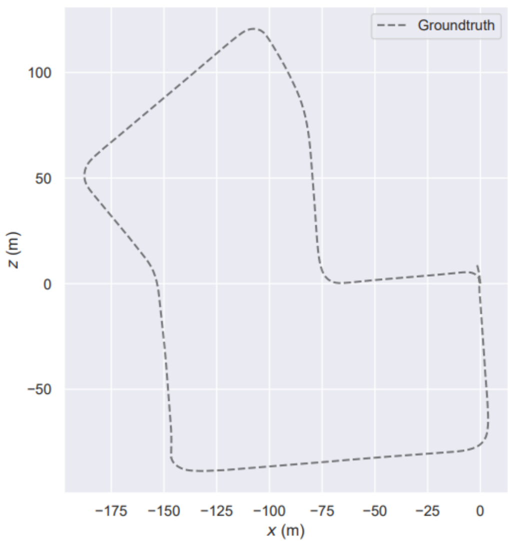

Figure 3.3: KITTI - sequence 7

Initially, to test the model's capacity, the model was trained and evaluated on the same sequence, to see if it's able to reproduce the ground truth. Then, the model was fed with more complex sequences.

3.2 Synthetic

As in there are few real-life datasets for visual odometry, we decided to create a synthetic dataset by using BlenderProc2 framework, which is a procedural photo-realistic rendering framework, and it allows to:

- **Loading**: **.obj*, **.ply*, **.blend*, *BOP*, *ShapeNet* etc.
- **Objects**: set or sample objects poses, apply physics and collision checking.
- **Materials**: set or sample physically-based materials and textures.
- **Lighting**: set or sample lights, automatic lighting of 3D-front scenes.

- **Cameras**: set, sample or load camera poses from file.
- **Rendering**: RGB, stereo, depth, normal and segmentation images/sequences.
- **Writing**: *.hdf5 containers, *COCO* and *BOP* annotations.

3.2.1 Scene

To create the synthetic dataset, the first thing is to create a scene with customized objects, material and textures.

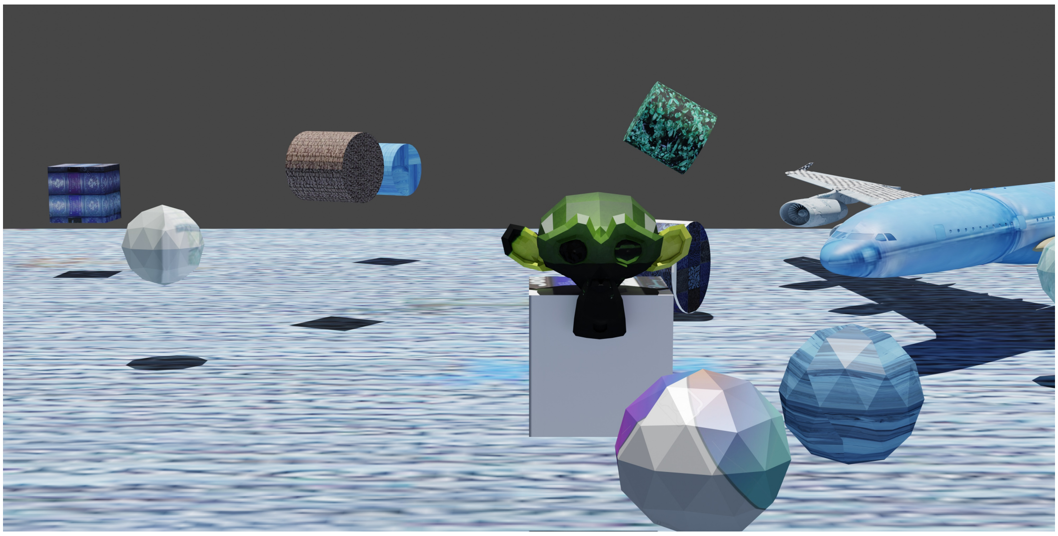

Figure 3.4: Example of scene

The scene is composed by a set of objects, more precisely:

- **a monkey**: which is at the centre of the scene over a cube.
- **a plane**: which is at left corner of the scene.
- **a set of cubes and spheres**: which are placed randomly in the scene.

When rendering the scene, the textures are loaded *randomly*, in the way that in different sequences the textures are different.

3.2.2 Image generation

To generate the sequences, we need to choose the camera position in the scene to do so, we choose randomly a position sampler from the following set for each new pose:

- **disk**: samples a point on a circle or on a 2-ball or on an arc/sector with an inner angle less or equal to 180 degrees.
- **sphere**: samples a point from the surface or from the interior of a solid sphere.
- **part-sphere**: samples a point from the surface or from the interior of a solid sphere which is split in two parts.
- **shell**: samples a point from the volume between two spheres (with radius of the spheres given as parameters).

once we have the next position of the camera, we compute the rotation matrix to be applied to the camera in the way that the camera is always looking at the POI (Point Of Interest) which corresponds to the centre of the scene.

```
rotation = bproc.camera.rotation_from_forward_vec(poi - new_
    position)
```

Code 3.1: Computes the rotatition matrix for the camera.

Then, we apply the rotation matrix to the camera and we generate the image, and by setting a certain number of frames between two poses, the framework renders a sequence of images with relative intermediate poses.

But sometime, it happens that the new camera pose is too close to an object of the scene, so we set two conditions that need to be satisfied, otherwise the sampled camera pose is skipped. The first condition checks if there are obstacles in front of the camera which are too far or too close based on the given *proximity_ checks*, while the second evaluates the interestingness or coverage of the scene.

```
def check_pose(c2w_m, special_obj, bvh_tree):
    if not bproc.camera.perform_obstacle_in_view_check(c2w_m,
        {"min": 5.0}, bvh_tree):
         return False
    if bproc.camera.scene_coverage_score(c2w_m, special_
        objects=special_obj) < 0.7:
         return False
    return True
```

Code 3.2: Checks whether the camera pose satisfies the conditions.

But when the new position is to far away from the old position, the rotation of the camera assumes a wrong value during the transition, because it rotates counter-clockwise instead of clockwise, or vice-versa. For example: If we sample the camera position from a disk at 0, 90, 180, 270 degrees, the rotations should be as in the figure 3.1:

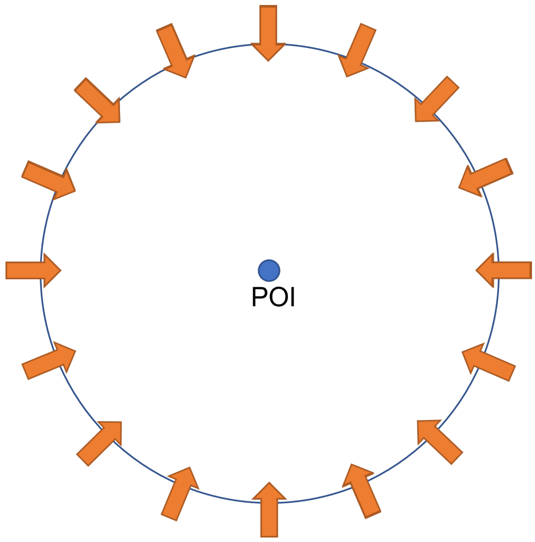

Figure 3.5: Correct transition on the disk

But, the transition from 180 to 270 degrees we obtain is a wrong rotation which is like in figure 3.2:

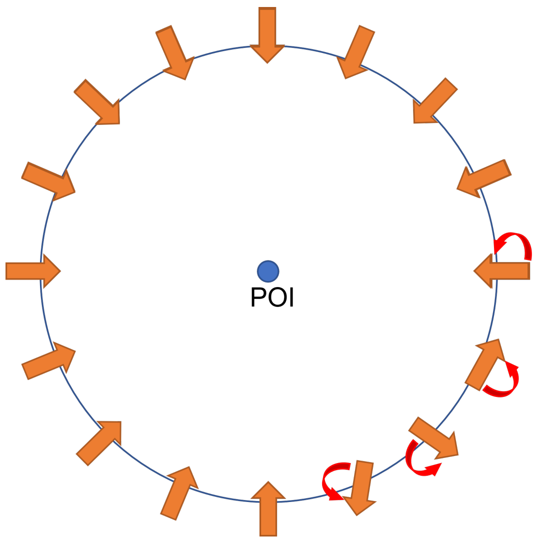

Figure 3.6: Wrong transition on the disk

To solve this problem, we tried different solutions: as first, we sample the next pose near to the previous one, but this solution sometime still fails. The final solution was to sample the new position as previously defined with samplers, but instead of letting the framework to compute the intermediate poses, we manually interpolate them, and by setting frame number to one, we obtain a sequence of correctly rotating images.

3.2.3 Dataset statistics

In total, we have generated 14 sequences, which are divided as follow:

Sets	**N. of Sequence**	**N. Image**
Training set	12	29.100
Validation set	1	1.002
Test set	1	1.003
Total	14	31.105

Table 3.2: Synthetic dataset statistics

Each image has dimension of 1024x308 pixels with 3 RGB channels. The whole dateset has dimension **1.69 GB**.

3.2.4 Usage

By using the dataset at training time the loss function is highly variable reaching values of **thousands**, also because the **Kitti** dataset is much fluid as the trajectory and the camera rotation angles are very small, so, the sequences generated are not similar to the real dataset.

Chapter 4

The State of the art

Chapter 5

Experiments

5.1 Encoder

5.2 Encoder-decoder

5.3 Encoder-Decoder with Auto-encoder

5.4 Prediction Strategies

Chapter 6

Implementations

Chapter 7

Final discussions

In this chapter will be discussed the results achieved.

7.1 Result Achieved

7.2 Knowledge Acquired

7.3 Future Developments

7.4 Personal Evaluation

Glossary

Auto-regressive A model is defined as auto-regressive when the next predictions depends on the previous ones.. 4, 25

IMU Inertial measurement unit is a device that measures the motion of an object in space.. 27

MHA Multi head attention is .. 27

MLP Multi layer perceptron is 27

NN Neural network models are a subset of Machine Learning models. NN is a network based on the concept of artificial neuron and neurons are organized in layers. The aim of the network is to mimic the data that is used in the training time. . 27

SLAM Slam is a computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it.. 27

Vanishing gradient problem When there are more layers in the network, the value of the product of derivative decreases until at some point the partial derivative of the loss functions approaches a value close to zero, and the partial derivative vanishes.. 2, 25

Word Example of a term in the glossary. 6, 25

Acronyms

IMU Inertial measurement unit. 3

MHA Multi-Head attention. 2

MLP Multi layer perceptron. 4

NN Neural network. 3

SLAM Simultaneous Localization and Mapping. 1

Bibliopraphy

Paper references

[5] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: (2015). DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385 (cit. on p. 1).

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: 25 (2012). URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (cit. on p. 1).

[8] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: (2014). URL: https://arxiv.org/abs/1409.1556 (cit. on p. 1).

[9] Christian Szegedy et al. "Going Deeper with Convolutions". In: (2014). URL: https://arxiv.org/abs/1409.4842 (cit. on p. 1).

[10] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: (2015). URL: https://arxiv.org/abs/1512.00567 (cit. on p. 1).