Általános információk, a diplomaterv szerkezete

A diplomaterv szerkezete a BME Villamosmérnöki és Informatikai Karán:

1. Diplomaterv feladatkiírás
2. Címoldal
3. Tartalomjegyzék
4. A diplomatervező nyilatkozata az önálló munkáról és az elektronikus adatok kezeléséről
5. Tartalmi összefoglaló magyarul és angolul
6. Bevezetés: a feladat értelmezése, a tervezés célja, a feladat indokoltsága, a diplomaterv felépítésének rövid összefoglalása
7. A feladatkiírás pontosítása és részletes elemzése
8. Előzmények (irodalomkutatás, hasonló alkotások), az ezekből levonható következtetések
9. A tervezés részletes leírása, a döntési lehetőségek értékelése és a választott megoldások indoklása
10. A megtervezett műszaki alkotás értékelése, kritikai elemzése, továbbfejlesztési lehetőségek
11. Esetleges köszönetnyilvánítások
12. Részletes és pontos irodalomjegyzék
13. Függelék(ek)

Felhasználható a következő oldaltól kezdődő LaTeXdiplomatervsablon dokumentum tartalma.

A diplomaterv szabványos méretű A4-es lapokra kerüljön. Az oldalak tükörmargóval készüljenek (mindenhol 2,5 cm, baloldalon 1 cm-es kötéssel). Az alapértelmezett betűkészlet a 12 pontos Times New Roman, másfeles sorközzel, de ettől kismértékben el lehet térni, ill. más betűtípus használata is megengedett.

Minden oldalon – az első négy szerkezeti elem kivételével – szerepelnie kell az oldalszámnak.

A fejezeteket decimális beosztással kell ellátni. Az ábrákat a megfelelő helyre be kell illeszteni, fejezetenként decimális számmal és kifejező címmel kell ellátni. A fejezeteket decimális aláosztással számozzuk, maximálisan 3 aláosztás mélységben (pl. 2.3.4.1.). Az ábrákat, táblázatokat és képleteket célszerű fejezetenként külön számozni (pl. 2.4. ábra, 4.2. táblázat vagy képletnél (3.2)). A fejezetcímeket igazítsuk balra, a normál szövegnél viszont használjunk sorkiegyenlítést. Az ábrákat, táblázatokat és a hozzájuk tartozó címet igazítsuk középre. A cím a jelölt rész alatt helyezkedjen el.

A képeket lehetőleg rajzoló programmal készítsék el, az egyenleteket egyenlet-szerkesztő segítségével írják le (A LaTeX ehhez kézenfekvő megoldásokat nyújt).

Az irodalomjegyzék szövegközi hivatkozása történhet sorszámozva (ez a preferált megoldás) vagy a Harvard-rendszerben (a szerző és az évszám megadásával). A teljes lista névsor szerinti sorrendben a szöveg végén szerepeljen (sorszámozott irodalmi hivatkozások esetén hivatkozási sorrendben). A szakirodalmi források címeit azonban mindig az eredeti nyelven kell megadni, esetleg zárójelben a fordítással. A listában szereplő valamennyi publikációra hivatkozni kell a szövegben (a LaTeX-sablon a BibTeX segítségével mindezt automatikusan kezeli). Minden publikáció a szerzők után a következő adatok szerepelnek: folyóirat cikkeknél a pontos cím, a folyóirat címe, évfolyam, szám, oldalszám tól-ig. A folyóiratok címét csak akkor rövidítsük, ha azok nagyon közismertek vagy nagyon hosszúak. Internetes hivatkozások megadásakor fontos, hogy az elérési út előtt megadjuk az oldal tulajdonosát és tartalmát (mivel a link egy idő után akár elérhetetlenné is válhat), valamint az elérés időpontját.

Fontos:

- A szakdolgozatkészítő / diplomatervező nyilatkozata (a jelen sablonban szereplő szövegtartalommal) kötelező előírás, Karunkon ennek hiányában a szakdolgozat/diplomaterv nem bírálható és nem védhető!
- Mind a dolgozat, mind a melléklet maximálisan 15 MB méretű lehet!

Jó munkát, sikeres szakdolgozatkészítést, ill. diplomatervezést kívánunk!

FELADATKIÍRÁS

A feladatkiírást a tanszéki adminisztrációban lehet átvenni, és a leadott munkába eredeti, tanszéki pecséttel ellátott és a tanszékvezető által aláírt lapot kell belefűzni (ezen oldal *helyett*, ez az oldal csak útmutatás). Az elektronikusan feltöltött dolgozatban már nem kell beleszerkeszteni ezt a feladatkiírást.

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Control Engineering and Information Technology

Comparison of convolution and transformer-based image processing neural networks

Bachelor's Thesis

Author
István Péter

Advisor
dr. Bálint Kiss
Ádám Gyula Nemes

December 6, 2022

Contents

HALLGATÓI NYILATKOZAT

Alulírott *Péter István*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2022. december 6.

Péter István
hallgató

Kivonat

Jelen dokumentum egy diplomaterv sablon, amely formai keretet ad a BME Villamosmérnöki és Informatikai Karán végző hallgatók által elkészítendő szakdolgozatnak és diplomatervnek. A sablon használata opcionális. Ez a sablon LaTeX alapú, a *TeXLive* TeX-implementációval és a PDF-LaTeX fordítóval működőképes.

Abstract

This document is a LaTeX-based skeleton for BSc/MSc theses of students at the Electrical Engineering and Informatics Faculty, Budapest University of Technology and Economics. The usage of this skeleton is optional. It has been tested with the *TeXLive* TeX implementation, and it requires the PDF-LaTeX compiler.

Chapter 1

Introduction

A bevezető tartalmazza a diplomaterv-kiírás elemzését, történelmi előzményeit, a feladat indokoltságát (a motiváció leírását), az eddigi megoldásokat, és ennek tükrében a hallgató megoldásának összefoglalását.

A bevezető szokás szerint a diplomaterv felépítésével záródik, azaz annak rövid leírásával, hogy melyik fejezet mivel foglalkozik.

1.1 The field of Computer Vision

Computer Vision is a branch of Artificial Intelligence aimed at deducting higher-order information from visual input like images, videos, or more specialised sensor data like LIDAR point clouds etc. Some individual tasks in Computer Vision are image classification, object detection, segmentation, pose estimation of specific entities etc.

1.2 Object Detection

Blah Blah

1.3 My Goal

My goal in this thesis is to give an overview of the differences between already estabilished Fully Convolutional Neural Networks (FCN) compared to upcoming Transformer-based architectures in the task of Object Detection. In the former, i restrict myself to single-stage detectors, namely the YOLO architecture.

Chapter 2

Overview of the Literature

In this chapter I am going to review the theoretical background for the two competing paradigms I cover: the fully convolutional, one-stage detector, whose most prominent variant is the You Only Look Once (YOLO) architecture, and the Transformer-based Detectection Transformer (DETR). For the former, I will explain in some detail chosing it over its competitors of the same kind, for example the Single Shot Detector (SSD).

In the case of the Tranformer-based category, I chose, for the sake of simplicity, the DETR architecture over its later successors, like DINO or Deformable DETR. The changes introduced in **its paper (insert citation)** are important enough to be discussed on their own, but I will mention the improvements achieved by the successors whenever the state-of-the-art is concerned.

Likewise, I have chosen the YOLOv5 for in-depth comparison as the DETR's counterpart, mainly because it is a contemporary of the latter (both being introduced in 2020), but mentioning the latest improvements introduced by YOLOv7 as well.

2.1 YOLO: You Only Look Once

2.1.1 Implementation details

2.2 DETR: The Detection Transformer

2.2.1 The Transformer in Natural Language Processing

The Transformer architecture has been introduced in the *Attention is All You Need* [5] paper in 2017, originally intended for Natural Language Processing (NLP) tasks, more specifically sequence transduction problems, like translation.

At the time, the attention mechanism and some variants of the encoder-decoder architecture was already widely used in the state-of-the-art, along with convolutional layers, Long Short Term Memory (LSTM) cells or Gated Recurrent Units (GRU). The Transformer was a successful attempt at replacing the latter three with trainable versions of the attention mechanisms called *Multi-Head Attention*.

In the Transformer model, the bulk of the learning happens at the weights of the linear transformations that estabilish the **heads** of the Attention layers, as the Attention layer itself does only mathematical operations on its input.

The article mentions that attention mechanisms and encoder-decoder based architectures have already been used at the time in the state-of-the art models. The novelty of the Transformer was getting rid of the convolutional, or traditionally recurrent components, and relying almost solely on the attention mechanism, namely a slightly modified version of it: the *multi-head self-attention*.

2.2.1.1 Multi

2.2.1.2 What is the difference between a neural layer and the attention?

2.2.2 DETR

The main advantage of the Detection Transformer is its capacity for every region to attend to every other region. In the fully convolutional case, this is done with hierarchical convolutions that together define large *receptive fields*.

2.2.3 Explainability

2.3 Comparison

Chapter 3

Practical Applications: Training on a Specific Task

3.1 Datasets

The Microsoft COCO: Common Objects in Context is an immensely popular dataset for numerous computer vision tasks (classification, object detection, instance segmentation, collectively called object recognition). It was introduced in 2014 in the paper *Microsoft COCO: Common Objects in Context* [3], aiming to improve upon already existent visual datasets like ImageNet and PASCAL, and striving to be a benchmark of scene understanding. It contains labeled data for 91 object classes, captured in their natural habitat (hence *context*). It was later updated in 2017, a notable change being the introduction of *stuff* labels (among the already existent *thing* labels), for objects with no clear boundaries, like sky and grass. These wer introduced as panoptic segmentation labels.

The official site of the dataset[1] mentions that the creators of COCO have partnered with the developers of open source tool FiftyOne in providing a software to facilitate downloading, visualizing, and evaluation on COCO, so I will be using it for intial data exploration. Another tool endorsed by the creators is the COCO API.

I conducted initial data exploration on the COCO dataset with the FiftyOne tool (launching an application session from a Python shell as described in their tutorials), as seen in figure 3.1. The success of the COCO dataset lies in its richness and difficulty: some training images, as seen in figure 3.2, are caught in so called *non-iconic views*, where visual features can be ambiguous and they should be interpreted in context to recognise the object.

3.1.1 Preparing the filtered COCO dataset

[1] `https://cocodataset.org`

Figure 3.1: Exploring the train split of the MS COCO 2017 dataset in the FiftyOne tool. There are 43867 instances of labeled cars and 9973 instances of labeled trucks, with 14714 images containing either of them.

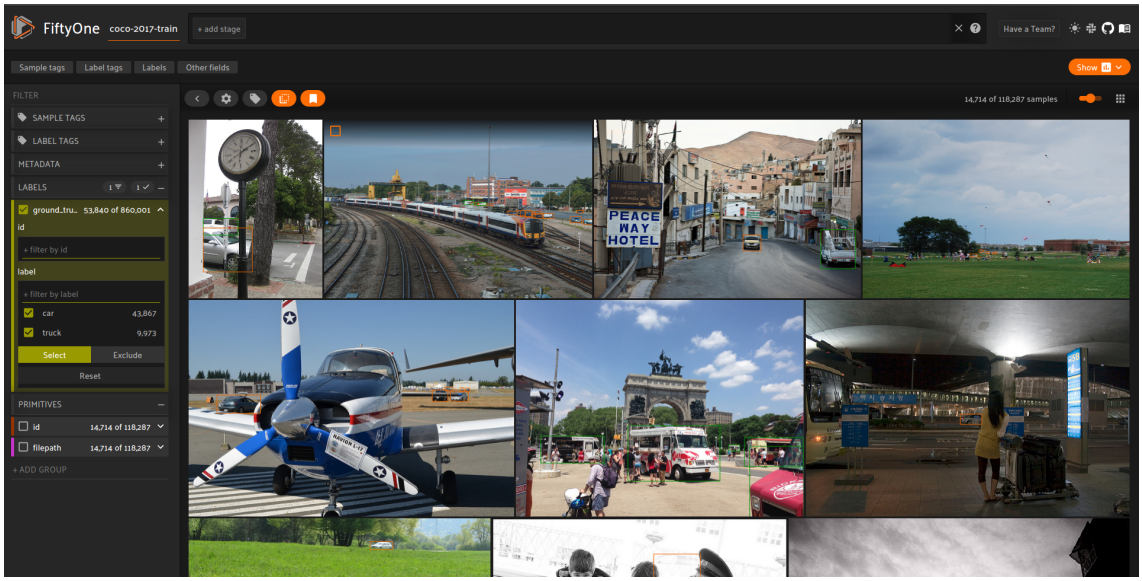

Figure 3.2: Some difficult examples: objects in overexposed regions of the image, crowd labels consising of very small (e.g aerial) images of cars (orange) and trucks(green), detection based on a skewed reflection of the onject, small parts of it.

Chapter 4

Higher Order Applications: Multiple Object Tracking

As the final aim of my thesis, I am going to inspect the performance of the chosen object detectors on a higher-order problem, namely multiple object tracking. First I am going to review the problem itself, and the common metrics used to measure its performance.

After this, I will elaborate on the object tracking method I chose as the basis, showing how it incorporates the aforementioned detectors, and the way the performance of the latter is expected to influence tracking performance.

4.1 The problem and possible approaches

Multiple object tracking (MOT) consists of determining the *trajectory* of given kinds of objects in a video stream. In the current context, the trajectory of a single, unique object is a series of bounding boxes with an identifier, one box for every frame in which the object is visible. The rectangular bounding boxes are expected to fit the object shilouette as tightly as possible.

Along the years, several approaches have been developed for the problem, but with the advent of highly accurate object detectors (most notably the breakthrough of convolutional neural networks from the mid-2010s and onward), the detection-based paradigm has become one of the most popular.

The approach consists of detecting objects on each new frame, then assigning them to previously found trajectories, if the object has been seen before, or registering a new trajectory otherwise. This is generally called the *association step*. In some scenarios, when the objects of interest are densely packed, this can pose a considerable difficulty.

Among some popular solutions to this association problem are the *proximity-based* (the Simple Online and Realtime Tracking (SORT) architecture [2] is one example) and the *feature-based* (the DeepSORT architecture [7] incorporates such methods using *deep appearance descriptors*) assignments. The former considers each already existent trajectory, takes their previous location, or the predicted location for this time step, and does the assignment based on some spatial distance between these and current detections. The latter does overarching associations based on visual appearance features, not necessarily restricted to the appearance observed in the last few frames.

The weakness of the proximity-based approach is when the density of objects is high, their movement highly dynamic and their trajectories intertwined. In this case, feature-based methods might excel. The weakness of the feature-based methods might show when they confuse similar, but distinct objects, or when they cannot account for some drastic, fast changes in appearance (like changes in lighting, orientation, deforming, or partial occlusion). In these situations, assignment to last known locations can help. Thus, these two approaches are not, and should not be exclusive.

The introduction above focuses on paradigms and aspects mostly relevant for my thesis, and narrows down the field somewhat. For a comprehensive, recent overview of the problem, see the literature review at [4]. It goes beyond just the detection-based approach, and also formalises the general problem.

Finally, it is worth mentioning that the most common multi-object tracking targets are pedestrians, faces and vehicles, and thus most popular MOT datasets consist of these. In this thesis, I am going to tackle tracking vehicles in road traffic footages.

4.2 Metrics

The MOT challenge [1] is one of the most popular currently used multiple object tracking benchmarks. Its most recent version is the MOT20 benchmark.

4.3 SORT

TODO: explain SORT

4.4 My measurement

4.4.1 Data

I will evaluate the model's performance for multi-object tracking on the UA-DETRAC dataset[2]. The dataset contains 100 videos (60 for training, 40 for testing) of road traffic captured at different locations in China. The total length of the video footage is around 10 hours, stored frame by frame (as separate 960 pixel by 540 pixel JPEG images), at the rate of 25 frames per second.

The annotations contain information about vehicle type, illumination, scale (proportional to the square root of the bounding box area), occlusion ratio (the measure by which other objects occlude the vehicle) and truncation ratio (the degree of the bounding box lying outside the frame). Information about weather conditions e.g. rainy, cloudy, sunny etc. is also included.

4.4.2 Benchmark

The dataset and the benchmark is described in [6]. The article also proposes an evaluation protocol for multi-object tracking. A key point is the joint analysis of detection and

[1] `https://motchallenge.net/`
[2] `https://detrac-db.rit.albany.edu/`

tracking performance, analysing the effects of the chosen model's precision/recall values (and the underlying confidence threshold setting that influences both) in relation with the tracking performance, as reflected by the MOTA and MOTP score. These relationships are visualized on the PR-MOTA and PR-MOTP curves (See figure 4.1).

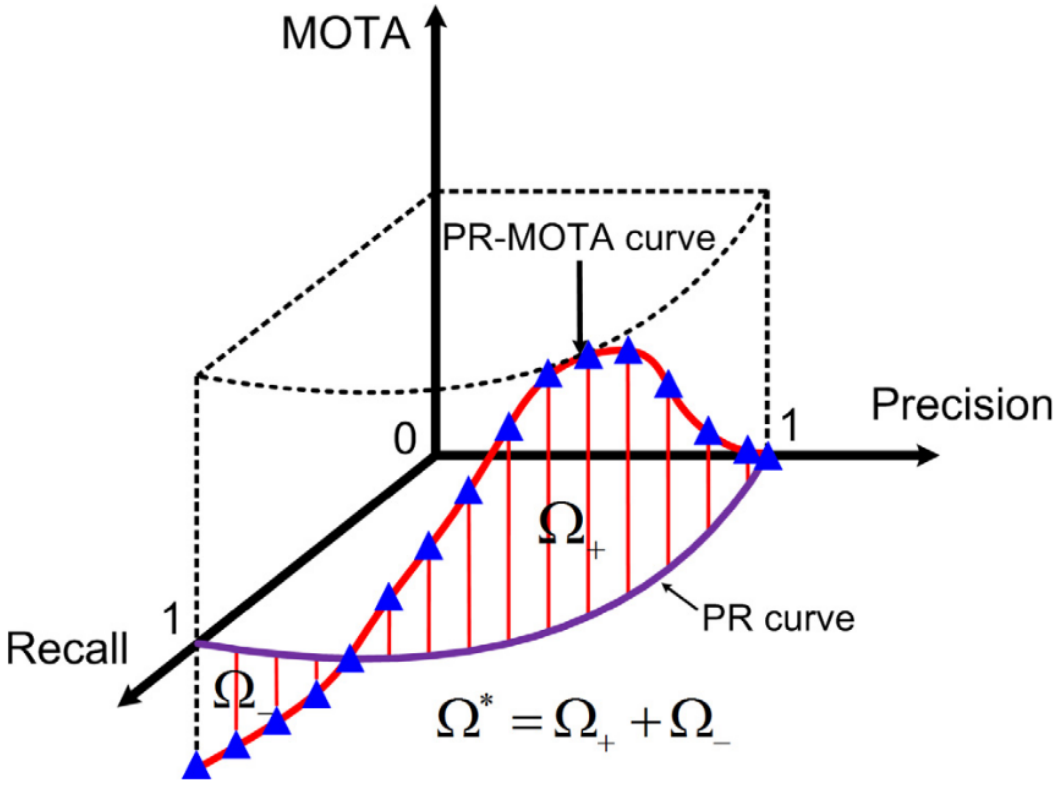

Figure 4.1: Visualization of the PR-MOTA curve. Image taken from [6]

The authors argue that, as it is not fair, nor enough to compare the performance of two object detectors based on different points on the PR curve, it is also not enough to determine the maximum point on the PR-MOTA curve, as a good tracker must produce good scores in a wider range of settings. The whole range of the curve must be taken into account in some form, thus the need for a new metric, Ω^*, or the *PR-MOTA score*:

$$\Omega^* = \frac{1}{2} \int_{\mathcal{C}} \Psi(p, r)\, d\mathbf{s}$$

where Ψ is the MOTA score across the whole dataset at precision p and recall r, and we calculate the (approximate value of the) signed area under the PR-MOTA curve as an integral along the PR curve $\mathcal{C}$ (for every $(p, r) \in \mathcal{C}$). Dividing by 2 ensures that the score stays in the interval $(-\infty, 100\%]$. Similar metrics can be defined for the MOTP, FP, FN, IDS, MT and ML scores.

TODO: explain MOTA, MOTP, FP, FN, IDS, MT, ML.

4.4.3 Evaluation

The tracking evaluation toolkit on the official DETRAC website is not available[3], because the login feature does not work. Thus, I had to write my own implementation for the evaluation based on the CLEAR MOT metrics [1].

4.4.4 Data Exploration

At the time of writing this thesis, the test and train images, grouped into sequences that form videos, can be accessed through the Download page of the official site[4] as `DETRAC-train-data.zip`. However, the tracking annotations for the train and test sets cannot be downloaded, as clicking on the links triggers a popup prompting to log in first. As the login functionality currently does not work, I had to look for alternative ways to access the data.

Fortunately, after a short search I have found a GitHub repository owned by the Georgia Tech Database Group. Their Exploratory Video Analytics System (EVA) repository contains, among others, a guide on how to download the UA-DETRAC dataset[5], along with a bash script serving the same purpose.

Through those links, I could download the training annotations. Sadly, the test annotations, claimed on the official UA-DETRAC website to be released, were still nowhere to be found, but I figured the 60 sequences (or even a subset of them) should be enough to evaluate tracking performance. The integrity of the measurement wouldn't have been compromised either, as I wasn't planning on doing detector training or hyperparameter tuning on the train set. There were two kinds of training annotation formats provided: XML and MAT.

The XML annotations are meant for detection training, and contain additional information like vehicle category, weather conditions during filming and bounding box scale. I did not use this data directly, as I used the models pre-trained on the COCO dataset, but inspecting this data confirms the vehicle categories supported by the DETRAC dataset: *car*, *van*, *bus*, *other*. The corresponding COCO object categories are *car*, *truck*, *bus*, so that is what I will be looking for when runnning detection on the images, and ignoring all other classes.

The MAT annotations are files in MATLAB serialization format, containing trajectory and position information for all tracked entities. For every image sequence, there is a `MVI_NNNNN.MAT` file. The image sequences themselves are consecutive frames under `Insight-MVT_Annotation_Train/MVI_NNNNN`, after unzipping `DETRAC-train-data.zip`.

Initially I inspected the MAT files' inner structure (see figure 4.2) in GNU Octave, MATLAB's open-source and somewhat compatible counterpart. I found that each file contained 5 matrices:

1. X: An $N \times T$ matrix, where T is the number of trajectories, and N is the number of frames. Given the frame i and trajectory j, $x_{i,j}$ denotes the x coordinate of the bottom center of the bounding box[6], or 0 if trajectory j is not present in that frame.

[3] `https://detrac-db.rit.albany.edu/Tracking`, under DETRAC-toolkit (Windows beta)

[4] `https://detrac-db.rit.albany.edu/download`

[5] `https://github.com/georgia-tech-db/eva/tree/master/data/ua_detrac`

[6] I found this out initially through trial and error, when trying to visualize bounding boxes in Python, because I did not know this to be a common format for specifying bounding boxes. Later, I found it mentioned in `https://detrac-db.rit.albany.edu/FAQ` as *foot position*.

2. Y: An $N \times T$ matrix, similar to X, but it contains the y coordinates of the bounding boxes' foot position.

3. W: contains the width of the boxes.

4. H: contains the height of the boxes.

5. $frameNums$: A row vector of length N, containing the 1-based indices of the frames.

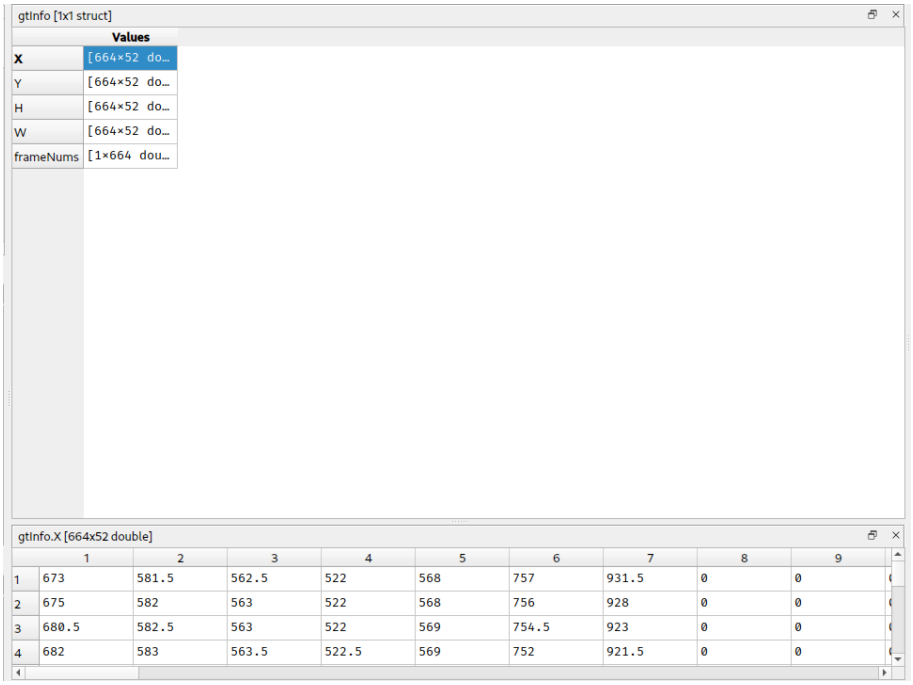

Figure 4.2: Label data exploration in octave

4.4.5 Data Processing

...

Bibliography

[1] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008(1):246309, May 2008. ISSN 1687-5281. DOI: 10.1155/2008/246309. URL https://doi.org/10.1155/2008/246309.

[2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, sep 2016. DOI: 10.1109/icip.2016.7533003. URL https://doi.org/10.1109%2Ficip.2016.7533003.

[3] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014. URL https://arxiv.org/abs/1405.0312.

[4] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. Multiple object tracking: A literature review. *Artificial Intelligence*, 293:103448, apr 2021. DOI: 10.1016/j.artint.2020.103448. URL https://doi.org/10.1016%2Fj.artint.2020.103448.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[6] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *Computer Vision and Image Understanding*, 2020.

[7] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649, 2017. DOI: 10.1109/ICIP.2017.8296962.