

# Analysis of CCTV data in order to characterize behaviors

BSc Computer Science

Dimo Dimchev(180322572)

Project Supervisor Prof. Paul Watson

## Declaration

"I declare that this dissertation represents my own work except where otherwise stated"

## Acknowledgements

First, I want to thank my project supervisor prof. Paul Watson for the guidance and support he gave me throughout the project. I would like to thank Peter Michalak and Tom Cooper for the helping me with the stream processing system environment setup and Nik Aznan for the advice about analysis of Kitti files and giving me access to the raw CCTV video tapes.

Lastly, I would like to thank my friends and family for the constant support throughout the way.

## Abstract

This dissertation looks at analyzing CCTV data in order to detect objects and measure social distancing between them in real time. It also looks at how 3D location of the object can be recovered from 2D image. The motivation for this project comes from the regulations of the well-known high transmission disease COVID-19 and intends to shed some light on how well they are observed in the public transport.

The script produced by this project analyzes Kitti files in order to generate alerts when social distancing is not being observed and generates relevant statistics in real time. Producing alerts when social distance is not being observed allows better compliance with the measures and the generated statistics can give feedback to the authorities on how well the measures for Covid-19 are being observed in the public transport.

# Table of Contents

DECLARATION.....	2
ACKNOWLEDGEMENTS .....	3
ABSTRACT.....	4
<b>TABLE OF CONTENTS.....</b>	<b>5</b>
<b>1 INTRODUCTION .....</b>	<b>7</b>
1.1 PURPOSE.....	7
1.2 AIMS AND OBJECTIVES:.....	8
1.3 DISSERTATION OUTLINE .....	9
<i>Introduction</i> .....	9
<i>Background</i> .....	9
<i>Implementation</i> .....	10
<i>Results and Evaluation</i> .....	10
<i>Conclusions</i> .....	10
<b>2 BACKGROUND: .....</b>	<b>11</b>
2.1 STREAM PROCESSING.....	11
2.1.1 <i>Publish/subscribe messaging systems.</i> .....	11
2.1.2 <i>The role of stream processing in my project</i> .....	12
2.1.3 <i>The uses of stream processing</i> .....	12
2.2 KAFKA.....	14
2.2.1 <i>Introduction</i> .....	14
2.2.2 <i>Streams in Kafka</i> .....	15
2.2.3 <i>Kafka Streams API</i> .....	15
2.3 FAUST.....	15
2.3.1 <i>The advantages of Faust</i> .....	16
2.3.2 <i>Design considerations:</i> .....	16
2.3.3 <i>The uses of Faust</i> .....	16
2.4 KAFKA / FAUST BASICS AND SIMILARITIES. ....	17
2.4.1 <i>Topics</i> .....	17
2.4.2 <i>Partitions</i> .....	17
2.4.3 <i>Fault Tolerance</i> .....	18
2.4.4 <i>Distribution of work/load</i> .....	18
2.4.5 <i>Offsets</i> .....	18
2.4.6 <i>Log Compaction</i> .....	19
2.5 ZOOKEEPER.....	19
2.5.1 <i>Advantages of Zookeeper</i> .....	19
2.5.2 <i>The importance of Zookeeper to Kafka and Faust</i> .....	19
2.6 FAIRMOT – FAIR MULTI-OBJECT TRACKING.....	21
2.7 KITTI FILES .....	22
2.8 KITTI FILES ANALYSIS.....	23
2.8.1 <i>Programmatic Position Estimation</i> .....	23
2.9 PYTHON .....	24

2.9.1	History.....	24
2.9.2	Python the programming language.....	24
2.10	UBUNTU.....	25
2.10.1	The uses of Ubuntu .....	25
2.11	DISTANCE BETWEEN TWO POINTS IN 2D AND 3D SPACE.....	26
2.11.1	The difference between 3D and 2D space .....	26
2.11.2	The distance between two points in 2D space.....	26
2.11.3	The distance between two points in 3D space.....	28
<b>3</b>	<b>IMPLEMENTATION .....</b>	<b>29</b>
3.1	SYSTEM DESIGN.....	29
3.1.1	CCTV data .....	29
3.1.2	Object-Tracking System (FairMOT).....	29
3.1.3	Kitti Files.....	29
3.1.4	Stream-Processing system .....	30
3.2	DEVELOPMENT ENVIRONMENT.....	30
3.2.1	Operating system.....	30
3.2.2	Object-Tracking system environment .....	31
3.2.2.1	Installation .....	31
3.2.2.1.1	Anaconda .....	31
3.2.2.1.2	FFMPEG .....	31
3.2.3	Stream Processing system environment .....	32
3.2.3.1	Faust.....	32
3.2.3.2	Kafka .....	33
3.2.3.3	Zookeeper .....	33
3.2.3.4	Running Faust scripts in the environment. ....	33
3.3	DEVELOPMENT .....	35
3.3.1	Feeding data into the Stream Processing System.....	35
3.3.2	Representing Kitti files as custom Records in a stream .....	37
3.3.3	Calculating Center Point and BBOX height .....	39
3.3.3.1	Calculating Object Height in Pixels in Agent 2.....	40
3.3.3.2	Calculating Center Point in Agent 2 .....	41
3.3.4	Calculating Depth with pedestrian height estimation .....	43
3.3.5	Calculating object location in the real world .....	45
3.3.6	Social Distancing.....	46
3.3.7	Headcounts .....	51
3.3.8	Statistics.....	51
<b>4</b>	<b>RESULTS AND EVALUATION .....</b>	<b>52</b>
4.1	FEEDING DATA INTO THE SYSTEM(PRODUCER PROGRAM) .....	52
4.2	REPRESENTING KITTI FILES AS CUSTOM RECORDS IN A STREAM .....	54
4.3	CALCULATING CENTER POINT AND BBOX HEIGHT.....	54
4.4	RECOVERING OBJECT DEPTH AND LOCATION WITH PEDESTRIAN HEIGHT ESTIMATION. ....	55
4.5	MEASURING OF SOCIAL DISTANCE .....	55
4.6	HEADCOUNTS.....	57
4.7	STATISTICS.....	57
4.8	SOCIAL DISTANCE, HEADCOUNTS, STATISTICS RESULTS COMPARED TO THE RAW VIDEO DATA .....	57
4.9	OVERALL EVALUATION.....	68
<b>5</b>	<b>CONCLUSIONS .....</b>	<b>69</b>

5.1	FULFILMENT OF OBJECTIVES.....	69
5.2	PERSONAL DEVELOPMENT .....	70
5.2.1	<i>Time management</i> .....	70
5.2.2	<i>Object-tracking Systems</i> .....	70
5.2.3	<i>Stream-processing systems</i> .....	70
5.2.4	<i>Python</i> .....	70
5.2.5	<i>Faust</i> .....	70
5.2.6	<i>Ubuntu</i> .....	70
5.2.7	<i>Problem solving</i> .....	70
5.3	FUTURE WORK .....	71
5.3.1	<i>Analysis of parameters and how they affect the results of the algorithm</i> .....	71
5.3.2	<i>Recovering object depth and location</i> .....	72
5.3.2.1	Using multiple cameras from different angles .....	72
5.3.2.2	Knowing the metrics of the vehicle.....	73
<b>REFERENCES:.....</b>		<b>74</b>
	<i>Stream Processing</i> .....	74
	<i>Zookeeper:</i> .....	75
	<i>FairMOT:</i> .....	76
	<i>Ubuntu:</i> .....	76
	<i>Python:</i> .....	76
	<i>Kafka + Faust</i> .....	77
	<i>Kitti Files</i> .....	77
	<i>Distance between two points in 2D and 3D space</i> .....	77

# 1 Introduction

This chapter seeks to introduce the project to the reader and provide the structure of the dissertation.

## 1.1 Purpose

It is widely known that Coronavirus disease(COVID-19) is an infection disease caused by a newly discovered coronavirus that could cause difficulty breathing, chest pain and loss of speech or movement. Due to the high transmission rate of Covid safety measures must be followed in order to prevent Covid transmission. Some of the measures include staying 1 meter apart and limiting social gatherings.

The purpose of this project is to design a software that would analyze data from CCTV cameras inside public transport in order to measure social distancing between passengers to check in real time if the safety measures are being observed or violated. By acquiring this information statistics could be computed in order to help the government with information about how well the measures are being observed in the public transport.

## 1.2 Aims and objectives:

The original aim of the project was to implement a machine learning software for object tracking that can run on an Edge device inside a bus or a train in order to generate alerts when social distancing is not being observed and produce relevant statistics in real time.

The main original objectives of the project were:

- **Use a machine learning object detection algorithm that takes CCTV data and outputs Kitti files.**

This objective included:

- Researching FairMOT
- Installing FairMOT software
- Using FairMOT to produce Kitti files from CCTV video tapes.

However, due to Covid-19 restrictions I was not able to gain access to the needed hardware – an Nvidia Jetson board. This led to me not being able to generate my own Kitti files, so I had to use already generated ones provided to me by my supervisor Paul Watson.

- **Design and Implement a stream processing system that takes the Kitti files and measures social distancing.**

This objective included:

- Research stream processing
- Learn Python
- Create stream processing system that takes the Kitti files as and processes them in order to measure social distance between objects.
- Produce alerts when social distancing is not being observed.
- Produce statistics in real time.

Over the course of the project, I realized that this objective included recovering 3D object location relative to the camera in order to measure social distance between objects. This was because there was no 3D location provided in the original Kitti files. So, in order to measure social distance two more sub objectives were added:

- Recover the depth of an object detected from a 2D video tape with the provided information from the Kitti files.
- Recover 3D position of an object relative to the CCTV camera.

I also had to choose a stream processing library in order to create the stream processing system, so I chose Faust. This added the following objectives to my list:

- Researching and learning about Faust

Also, Kafka and Zookeeper had to be researched as the Faust library was built on them and they had to run in the background in order to use Faust.



- Researching Kafka streams
  - Research Zookeeper
  - Researching Faust
- **Implement the systems on an Nvidia Jetson Edge device.**
  - Combine the object detection algorithm and stream processing system.
  - Implement them on Nvidia Jetson Edge device.

This objective turned out to be impossible to complete as I did not have access to the needed hardware – an Nvidia Jetson board .

## 1.3 Dissertation Outline

### Introduction

An introduction to the dissertation stating the project's aim and objectives and presenting the outline of the dissertation.

- Purpose
- Aims and Objectives
- Dissertation outline

### Background

This chapter contains a review of the background research done about the technologies used in this dissertation as well as research about mathematical formulas used in the dissertation. It also contains an explanation of why these technologies are needed in the project.

- Stream Processing Research
- Kafka
- Faust
- Zookeeper
- Fair Multi-object Tracking
- Kitti Files
- Python
- Ubuntu
- Calculating distance

## Implementation

This chapter explains the design of the system and exposes how was that design implemented.

- System Design
  - This section contains a high-level explanation of how the system works.
- Development environment
  - This section exposes how the development environment was setup
- Development
  - This section contains explanation of the development done in this project

## Results and Evaluation

Evaluation of the results produced by the system created in the Implementation chapter.

## Conclusions

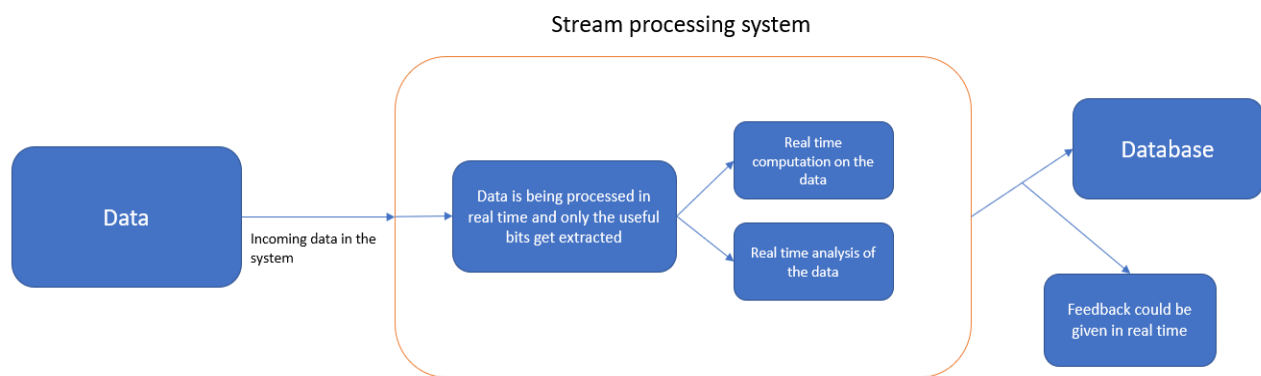
- Fulfilment of objectives
- Personal Development
- Future work

## 2 Background:

This section is a review of the technologies that are going to be used in the project.

### 2.1 Stream Processing

Stream processing is a computer programming architecture in which data is computed directly as it is produced or received.



#### Stream Processing system diagram

The job of a stream processing system is to take Data (Look at the diagram above) that is a never-ending stream of events, process that data to extract only the useful bits from it and do some computation on them. The Stream processing system could be used also to analyze events that happen in real time. This is the case in this project in which the data from the CCTV cameras is analyzed by an object tracking algorithm and the output from that algorithm is imported into the stream processing system, where data is analyzed to produce social distancing alerts and count detected people. After computation is done on the data the results could be stored in a database for more in-depth analysis later or they could also be inputted into another system that will use them in real-time. [1]

Stream processing is technology that focuses on the real-time processing of continuous streams of big data in motion like live CCTV data coming from a camera and Internet of things. One type of stream processing tools are the distributed publish-subscribe messaging systems such as Kafka. [2] [3]

#### 2.1.1 Publish/subscribe messaging systems.

Publish/subscribe or pub/sub is a communication model that directs the flow of messages from senders to receivers based on the data interests of the receivers. This is a many to many

communication model which means that each sender can send to many receivers and each receiver can receive from many senders. [4][5]

In this model the senders also called publishers generate messages without knowing their receivers, and receivers also known as subscribers express their data interests and are subsequently notified of the messages from a variety of senders that match their interests.

Stream processing also known as real-time analytics is used best to query continuous data stream and detect the conditions and patterns that you would like within a small time period from the time data was received. In my case I use it to quickly detect if social distancing is present. And if it's not it returns an alert.

### 2.1.2 The role of stream processing in my project

The data from the CCTV cameras comes as a never-ending stream of events. This stream of events needs to be processed in real time, because the application is time sensitive and generates alerts in real time if social distancing is not observed.

With stream processing you can detect patterns, inspect results, and combine and analyze data from multiple streams. This makes it the perfect Big data technology for real time and never-ending streams.

### 2.1.3 The uses of stream processing

Stream processing is used in variety of cases. Some of them are:

- Internet of things
  - Internet of things is the concept of computer network of physical objects having built-in electronic devices for intersection with each other or with external environment. The concept considers the organization of such networks as a phenomenon capable of reshaping economic and social processes so as to execute the need for human participation in some actions and operations. [6]
- Algorithmic trading
  - Algorithmic trading is a process for executing orders utilizing automated and preprogramed trading instructions to account for variable such as price, timing and volume. Algorithmic trading makes use of mathematical formulas and models and human oversight to make decisions to buy/sell Shares. Algorithmic traders can trade tens of thousands of times per second, this is often achieved by a stream processing real time system. [7]
- Smart Care

- Also known as home care or home health software applies real time information processing involving both computer hardware and software that deals with storage, retrieval , sharing, and use of health care information. Smart care software is designed specifically for companies employing home health providers as well as government entities who track payments to home health care providers.  
Home health care software allows health care providers to obtain and transmit information like health status, functional status, support system information in real-time while on location with a patient.[8]
- Production line monitoring
  - A Production line is a line of machines in a factory that a product moves along while it is being produced. In the production line each machine or worker performs a particular job that must be finished before the product moves on to the next position. A production line monitoring system monitors the whole process and enables the staff to maximize efficiency and improve management.[9]
- Intrusion
  - Intrusion detection systems are devices or software applications that monitor a network or systems for malicious activity. Intrusion detection systems or for short IDSs are based on patterns. They detect attacks by looking for specific patterns like byte sequences in network traffic or known malicious instruction sequences used by malware. This is done in by using a stream processing system to process data and find patters in real time.
- Surveillance
  - Surveillance is a monitoring of behavior, places, activities for the purpose of information gathering. This includes closed-circuit television(CCTV) that performs observation from a distance or interception of electronically transmitted information such as Internet traffic.  
Surveillance is used for crime prevention, protection of a process, person or an object. [11]
- Fraud Detection
  - Fraud detection is a set of processes and analyses that allow businesses to identify and prevent unauthorized financial activity. Since fraud is often committed through patterns, fraud detection uses artificial intelligence to look for these types of patterns and sends an alert when one is detected. This is all done in rea time so an alert of a fraud could be produced instantly.[10]
- Real-time systems
  - Real time operating systems or (RTOS) are operating systems intended to serve real-time applications that process data as it comes in. A real-time system is a time bound system which has well-defined, fixed time constraints. In such systems processing must be done within the defined constraints or the system will fail.  
Real-time systems are either Event-driven or Time sharing.  
The difference is that in event driven real time systems tasks are switched only when an event of higher priority needs servicing, where in time-sharing real time systems tasks are switched in a specific time period that is assigned by a scheduling algorithm like round robin. [12]
- Machine Learning techniques for predictive maintenance

- Predictive maintenance predicts failure in a system, and the actions could include corrective actions, the replacement of system or a planned failure. Predictive maintenance is used to detect the anomalies and failure patterns and provide early warnings in real time. Here a stream processing system is also used to process the data in real time, detect patterns and produce warnings and possible solutions to the problems.[13]

## 2.2 Kafka

### 2.2.1 Introduction

Apache Kafka is a distributed data store optimized for processing streaming data in real-time. It is used to build real-time streaming data pipelines and real-time streaming applications.[22]

Kafka's three main functions are:

- To Publish and subscribe to streams of records.
- To Effectively store streams of records in the order in which data is generated.
- Real-time processing of data.

Kafka uses a partitioned log model that is a combination between queuing, which allows the data processing to be distributed across many consumer instances and the publish-subscribe model.

[23]

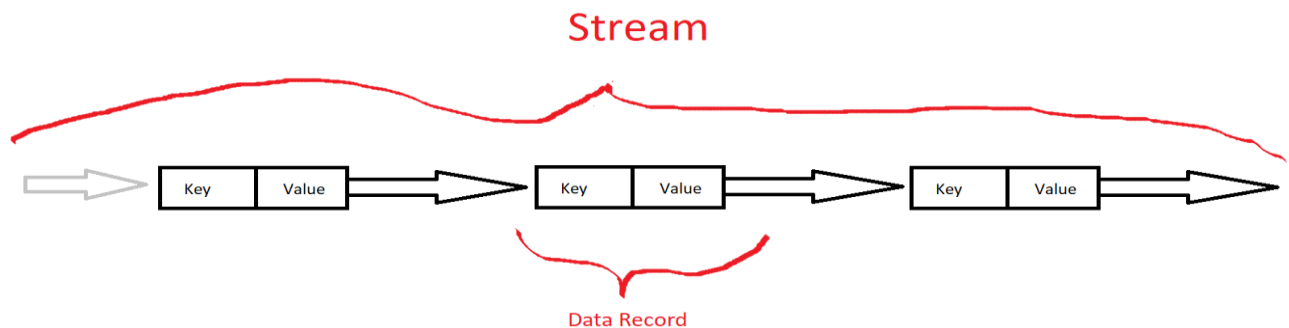
Each log is an ordered sequence of records and every log is broken up into partitions that correspond to different subscribers.

### 2.2.2 Streams in Kafka

Streams are unbounded, continuous real time flow of data (records).

-You don't need to explicitly request new records; you just receive them.

The records are key – value pairs.



### Streams in Kafka diagram

### 2.2.3 Kafka Streams API

- This is an API that transforms and enriches data.
  - Supports per-record stream processing with millisecond latency.
  - Supports stateless processing(transformation)- Filtering and mapping.
  - Supports stateful processing(transformation) – Joins and aggregations.
  - Supports windowing operations.
- It is used to write standard java applications and microservices to process your data in real time.
  - Allows development on Mac, Linux, Windows.
  - Kafka is equally viable for small, medium, and large use cases.
  - Provides Kafka security mechanisms.
  - Supports exactly-once semantics
- The Kafka Streams API is part of the open-source Apache Kafka project.

## 2.3 Faust

The Kafka alternative in python.

Faust is a distributed stream processing library built to handle large amounts of data in real time.

It is used to build high performance distributed systems and real-time pipelines that process billions of events every day. It provides both stream and event processing and is similar to tools such as Kafka Streams, Apache Spark/Samza/Flink.

Faust is a python 3 library, taking advantage of performance improvements in the language like the AsyncIO module for high performance asynchronous inputs and outputs. We can say that Faust is a software that is inspired by the Kafka Streams ideas but takes a slightly different approach to the processing of streams. The source code of Faust is short enough to be easy to understand and is an excellent resource to learn more about how stream processing applications and frameworks like Kafka Streams work in general.

### 2.3.1 The advantages of Faust

Unlike most of the other stream processing frameworks, Faust does not use domain-specific language. Instead, it provides stream processing as a Python 3 library that lets you reuse the tools you already might be familiar with when stream processing. This enables you to combine the stream processing library Faust with python libraries such as NumPy, PyTorch, Pandas, NLTK, Django, Flask, and many others. The idea is to make Faust intuitive to use for anyone already familiar with Python programming.

Python was built as a library that you can implement into any existing Python code with support for all available python libraries.[35]

### 2.3.2 Design considerations:

- Modern python
  - It uses async/await features from Python 3 .
- Library
  - It is designed as a python library, so its easier to embed it into an existing Python program.
- Supervised
  - The Faust worker is built up by services that that start and stop in a certain order.  
Example – Zookeeper -> Kafka -> Faust.  
Those services are managed by supervisors, but if an irrecoverable error such as not recovering from lost Kafka connection occurs, Faust is designed to crash.
- Extensible
  - Faust abstracts away storages, serializers and message transports. This makes Faust easy to extend and integrate into different systems.

### 2.3.3 The uses of Faust



Faust has the idea of making the design of traditionally complex streaming architectures simpler. Some of the ways to use Faust are:

- Risk and fraud detection
- Order execution quality monitoring
- Event logging pipelines
- News aggregation and tagging
- Analysis of Kitti files

## 2.4 Kafka / Faust basics and similarities.

As Faust is built and operates on top of Kafka it is important to look at what both platforms have in common and how they work together.

### 2.4.1 Topics

In Kafka Topic is basically the stream name to which records are being published. They are the base abstraction of where data lives within Kafka. Each topic is backed by logs which are partitioned and distributed.

In Faust we have an abstraction of the Kafka topic to both consume data from a stream as well as publish data to more streams represented by Kafka topics. Every Faust application needs to be consuming from at least one topic and may create many intermediate topics as a byproduct of the streaming applications.[25]

### 2.4.2 Partitions

In Kafka partitions are the fundamental unit within Kafka where data lives. Every Kafka topic has one or more partitions. Those partitions are represented internally as logs and messages always make their way to one partition (log). Each partition is replicated and has one leader at any point in time.

Faust uses the notion of a partition to maintain order amongst messages and as a way to split the processing of data. A Faust application uses the notion of a “key” to make sure the messages that are supposed to be processed together are processed together.[25]

### 2.4.3 Fault Tolerance

In Kafka, every partition is replicated with the total copies represented by the In Sync Replicas (ISR). Every ISR is a candidate to take over as the new leader if the current leader fails. The maximum number of faulty Kafka brokers that can be tolerated is the number of ISR - 1. This means that if every partition has three replicas all fault tolerance guarantees hold as long as at least one replica is functional.

Faust has the same guarantees that Kafka offers with regards to fault tolerance of data.[25]

### 2.4.4 Distribution of work/load

For every partition, all reads, and writes are routed to the leader of the partition. For a specific topic, the load is as distributed as the number of partitions. The number of partitions control how much many parallel instances of the consumers can operate on messages.

Faust Uses Parallel consumers and is also limited by the number of partitions to dictate how many concurrent Faust application instances can run to distribute work. Faust application instances more than the source topic partition count will be idle and not improve message processing rates.[25]

### 2.4.5 Offsets

For every topic-partition combination Kafka Keeps track of the offset of messages in the log to know where new messages should be appended. On a consumer level, offsets are maintained at the group - topic -partition level for consumers to know where to continue consuming for a given "group". The group acts as a namespace for consumers to register when multiple consumers want to share a single topic.

Kafka maintains processing guarantees of at least once by committing offsets after message consumption. Once an offset has been committed at the consumer level the message at that offset for the group-topic-partition will not be reread.

Faust uses a group to maintain a namespace within an app. Faust commits offsets after when a message is processed through all of its operations. Faust allows a configurable commit interval which makes sure that all messages that have been processed completely since the last interval will be committed. [25]

#### 2.4.6 Log Compaction

Log compaction is used by Kafka to make sure that as the data for a key changes it does not affect the size of the log such that every state change is maintained for all time. Only the most recent value is guaranteed to be available. Periodic compaction removes all values for a key except the last one.

Tables in Faust use log compaction to ensure table state can be recovered. [25]

### 2.5 Zookeeper

Zookeeper is an open-source coordination service for distributed application. Zookeeper offers a simple set of primitives that the distributed applications can build up on in order to implement higher level services for synchronization, maintenance of the configuration, naming and groups.

Zookeeper is also distributed service that is designed to be easy to program and uses a data model styled after the familiar directory tree structure of file systems. It runs in Java has bindings for both Java and C.

Coordination services are prone to errors such as race conditions and deadlock. The motivation behind ZooKeeper is to relieve distributed applications the responsibility of implementing coordination services from scratch.

Zookeeper is a service that maintains configuration information, naming, providing distributed synchronization, and provides group services. This is provided in a simple interface, so you don't have to write your services from scratch. Zookeeper also be used to implement consensus, group management, leader election and presence protocols. And you can build on it for your own specific needs.[14]

#### 2.5.1 Advantages of Zookeeper

Zookeeper is known for its Reliability, because it keeps working even if a node fails, its Simplicity due to its simple architecture, Its Speed due to its fast processing for workloads, its Scalability due to Zookeeper being scaled by simply adding additional nodes.

#### 2.5.2 The importance of Zookeeper to Kafka and Faust

In order to use Faust, you must have a Kafka server running in the background. And in order to have a Kafka server running in the background you must have ZooKeeper installed.

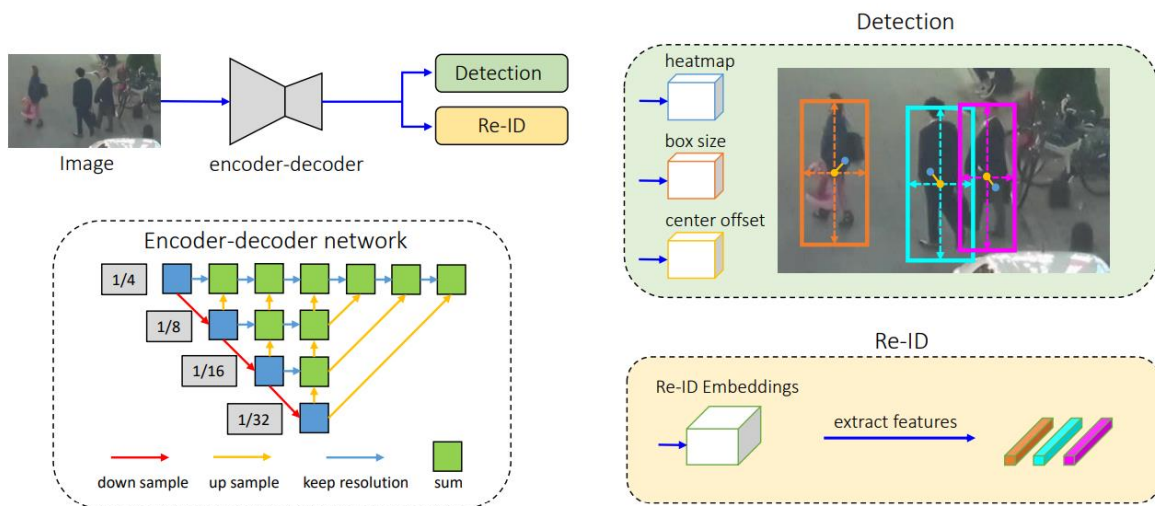
This is because Kafka is a distributed system built to use ZooKeeper as a way of coordinating tasks.

Zookeeper has five primary functions in Kafka.[15]

- Used for controller election.  
The broker responsible for maintaining the Leader - Follower relationship for all partitions is called "the controller". Zookeeper ensures that replicas take the role of "the controller" if the node of the current leader is shut down.
- Cluster membership  
List of all available brokers in the cluster.
- Topic Configuration  
Maintains all existing topics. Also maintains number of partitions for each topic, location of the replicas, configuration overrides for topics, preferred leader node.
- Access control lists  
Who is allowed to read, write or both to each topic, list of consumer groups, members of the groups, and the most recent offset that each consumer received from each partition.
- Quotas
  - How much data each client is allowed to read or write or both.

## 2.6 FairMOT – Fair Multi-object Tracking

FairMOT is a software that is used for object detection and object tracking. The differences between FairMOT and other object tracking systems is that it treats object detection and object re-identification equally, whereas in other object tracking systems the detection has a higher priority than the re-identification.[17]



**Diagram for Fair Multi-Object tracking [17]**

FairMOT adopts a network structure consisting of two homogeneous branches for detecting objects and extracting re-ID features.

The detection branch is implemented in an anchor-free style which assesses object centers and sizes represented as position-aware measurement maps.

The detection branch as the name suggests specialises in object detection.

The re-ID branch estimates a re-ID feature for each pixel to characterize the object centered at the pixel.

This branch specialises in re-identification of objects and aims to generate features that can distinguish objects. This is achieved by a convolution layer with 128 kernels on top of backbone features to extract re-ID features for each location.

The two branches are completely homogeneous which essentially differs from the previous methods which perform detection and re-identification in cascaded style.

When FairMOT is evaluated on the Multi-Object Tracking Challenge benchmark it ranks first among all other trackers on the following datasets: 2DMOT15, MOT16, MOT17, MOT20.

It is said that the models can give even better results if they are further pre-trained with FairMOT's proposed self-supervised learning method.[17]

## 2.7 Kitti Files

Kitti files are mostly used as a way of representing data produced by a machine learning algorithm and making that data ready to be ingested into a data set where it will be analyzed.

Kitti is the name of the format for object detection data. It is used when preparing your own data for ingestion into dataset. Or simply said these are text files formatted with values that are space separated into the following fields:

{frame\_id},{track\_id},{Type},{truncated}, {occluded}, {alpha}, {bbox\_left} {bbox\_top} {bbox\_right} {bbox\_bottom}, {dimensions - 3 value}, {location - 3 value}, {rotation-y}, {score}

Where frame\_id is the id of the frame where an object appears.

Track\_id is the unique tracking id of an object.

Type describes the type of object that could be: 'Car', 'Van', 'Truck', 'Pedestrian', 'Person\_sitting', 'Cyclist', 'Tram', 'Misc' or 'DontCare'.

In our case this will only be a 'Pedestrian' object.

Truncated is an integer value that indicates the level of truncation.

-The principal of truncation means that the object partially lies outside of the image area.

Occluded is an integer value that indicates the occlusion state(0 is full visible, 1 is partly ocuded,2 is largely occluded, 3 is unknown.)

Bbox coordinates are the coordinates of the Top-left and bottom-right angles of the 2D bounding box around the object.

Dimensions are the 3D object dimensions height, width, length.

Score – is the score of confidence of detection, which basically indicates how sure the algorithm is if the detection is correct. The higher this value the better.

Rotation\_y is the rotation ry around the Y-axis in camera coordinates.

Alpha is the observation angle of the object.

-The difference between Alpha and Rotation y is that rotation is directly given in camera coordinates, while alpha also considers the vector from the camera center to the object center to compute the relative orientation of the object with respect to the camera. Rotation-y doesn't consider X-Z coordinates of the object it is more like bird's eye view, while for alpha the observation angle will change with the when an object is moving along the Z or X axis. [28]

## 2.8 Kitti Files Analysis

### 2.8.1 Programmatic Position Estimation

Programmatic Position Estimation[34] is the method that is going to be used for finding out the X,Y,Z coordinate location of an object in the real-world scene from the 2D image taken by CCTV.

Programmatic Position Estimation[34] is a method that uses the principles of photogrammetry to recover 3D position of an object within the scene based on a known camera projection model and an assumption made on the real-world size of the object width or height.

Target position is initially known within the “sensor space” or the pixel position of the image. Consequently, the position of the target is estimated based on the photogrammetry principles combined with knowledge of the perspective under which targets are imaged.

All the targets are imaged under a standard perspective projection:

$$x = f \cdot X/Z \text{ and } y = f \cdot Y/Z \quad (\text{Equation 1})$$

Where X,Y and Z are the object's real-world position in 3D co-ordinate scene is presented in the 2D image as (x,y), in pixel-coordinate space for a given camera focal length f. Both positions are assumed to be centroid of the object with (x,y) being the center of the bounding box of the image sub region where the object is detected in the scene.

With knowledge of the camera's focal length f the original object position (X,Y,Z) can be recovered based on knowledge(assumed) of either object width or object height.

Based on this knowledge the depth of the object position could be recovered as follows.

$$Z = f' \cdot (\text{assumed object height 3D}) / (\text{object height of bounding box 2D image}) \quad (\text{Equation 2})$$

Where  $f'$  represents focal length, f translated from mm to focal length in pixels.

This is done with this equation:

$$f' = (\text{width of the image in pixels}) \cdot f / (\text{camera CCD sensor width(mm)}) \quad (\text{Equation 3})$$

Assumed object height in Equation 2 is going to be the average pedestrian height based on available medical statistics.

When Z is known from substituting in equations 2 and 3, we can substitute in Equation 1 and find X and Y and recover the (X,Y,Z) coordinates in meters in the real world.

## 2.9 Python

### 2.9.1 History

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0.

Python 2.0 was released in 2000 and introduced features such as list comprehensions and garbage collection system.

Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Much Python 2 code does not run unmodified on Python 3.[19]

### 2.9.2 Python the programming language

Python is high-level, general-purpose programming language. Its design philosophy emphasizes code readability with its notable use of significant identification. The python programming language construct and object-oriented approach aim to help programmers write clear, logical code for their projects no matter if they are small or large in scale.

Python is dynamically typed. This means that variables must be defined before being used. It also implies that the dynamically typed languages do not require explicit declaration of variables. This could make coding much easier and at the same time if the programmer is not careful it could lead to mistakes that don't even give error messages, so they are even harder to find. For example, if the programmer tries to use a defined variable and misspells it the misspelled variable will be treated as a new variable and this will not even be an error.

Python is garbage-collected. This means that Python deletes unwanted objects automatically to free memory. This garbage collector runs during program execution and uses object's reference count. When the value of an object is used (increased, decreased, assigned, placed in a container) the reference count increases and when an object is deleted with `del` the reference count decreases.

And if an object's reference count reaches zero Python automatically collects it and frees the memory used by it.

What are the advantages of python?

Python supports multiple programming paradigms, including structured, O-O programming and functional programming.

Python's large standard library is one of its biggest strengths. It provides tools suited to many tasks. For internet-facing applications, many standard formats, and protocols such as MIME and HTTP are supported. The standard library also includes modules for making graphical user interfaces,



connecting to relational databases, generating random numbers, arithmetic with arbitrary-precision decimals, reg-expressions, unit testing and many more.

As of March 2021, the Python Package Index (PyPI), the official repository for third-party Python software contains over 290,000 packages that provide a wide range of functionality. Those packages include everything from creating Mobile apps and Machine learning to web scraping and stream processing.[19][20][21]

## 2.10 Ubuntu

Ubuntu is a Linux operating system freely available with both community and professional support. The Ubuntu community is built on the ideas that software should be free of charge and software tools should be usable by people in their local language. Also, another of the ideas of the software is to give the people the freedom to customize and after their software in whatever way they see fit.

Those ideas make ubuntu one of the best environments for programing in every language and working with software in general.

Ubuntu has tree editions : Desktop, Server, and Core. All editions can run on the computer alone or in a virtual machine.

Ubuntu includes a lot of different pieces of software starting with the Linux kernel version 5.4 and GNOME 3.28. It also contains every standard desktop application such as word processing, spreadsheets apps, browser web server software, email. Ubuntu also includes a lot of different programming languages and tools that make every programmers life easier. [18]

### 2.10.1 The uses of Ubuntu

In this project Ubuntu is used as an operating system to support the development process and facilitates programing.

More precisely it supports the development process by:

- Making Zookeeper installation easier
  - Making FairMOT installation easier
  - Making Kafka installation easier
  - Running Zookeeper/Kafka server is easier.
- Doe to Linux's powerful terminal.

It facilitates programming because Linux enables you to run multiple scripts directly from the terminal. And in case of an error the error messages in Linux are much more precise compared to Windows for example, which helps with finding and resolving the problems much faster and easier.

## 2.11 Distance between two points in 2D and 3D space

### 2.11.1 The difference between 3D and 2D space

The difference between a 2-dimensional and 3-dimensional space is that 2d space also known as “flat” is using only two dimensions while 3d space uses 3 dimensions.

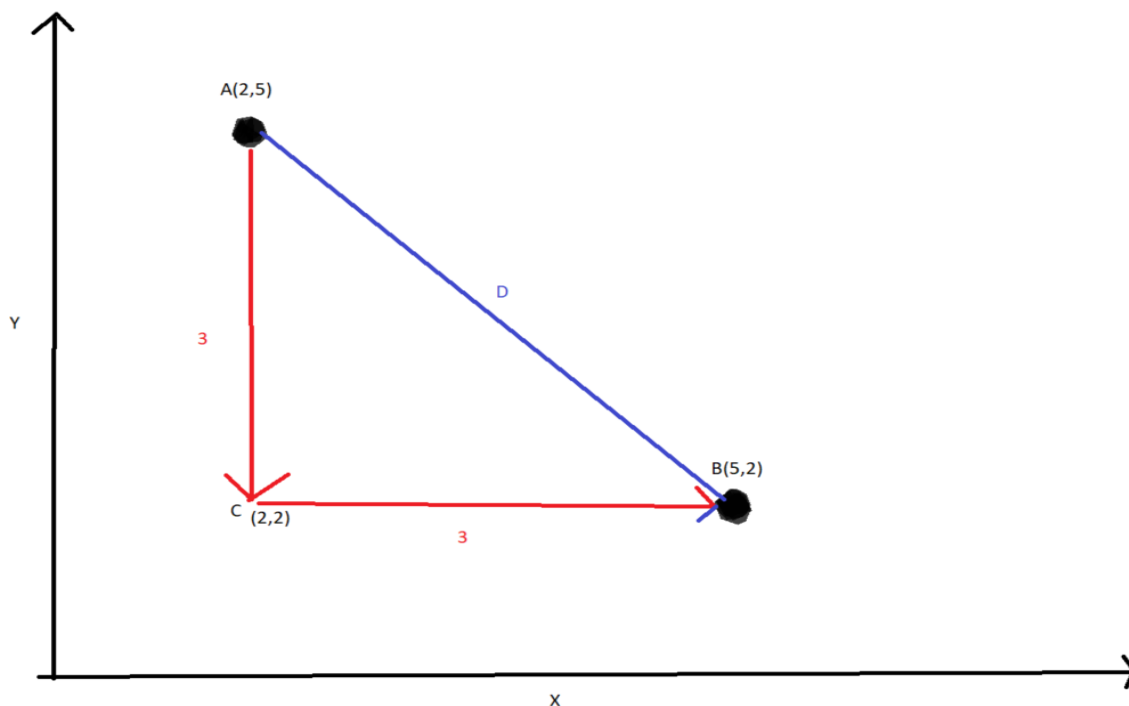
For example, a picture has only two dimensions and if you turn it to the side, it becomes a line.

This is because a two-dimensional space has an infinite number of lines. A 3-dimensional space adds another dimension – depth(Z) that allows rotation and visualization of objects from multiple perspectives. We can also say that 3D space has an infinite amount of 2d spaces within.

An example of the difference between 2d and 3d space is a picture of a person and the person. If we look at the picture of the person from the side it looks like a straight line, but if we look at the real person from the side, we are just looking at him from different perspective.

### 2.11.2 The distance between two points in 2D space

In two-dimensional space we have only 2 dimensions (x, y) so if we look at the picture below:



Distance between two points in 2D space

In order to measure the distance between A and B we need to know how we can go from A to B.

So, if A has a Y value of 5 and b has a Y value of 2 the difference between those values is 3.

So, if we travel 3 units down from A, we arrive at point C with coordinates (2, 2).

(1) AC is parallel to the Y axis.

Now the only difference between C and B is the X value.

The difference between the X values of C and B is 3. We can draw that as a line parallel to the X axis (2) with length 3.

From (1) and (2) we get a right-angled triangle and we can calculate the distance from A to B with the formula:

$$D = \sqrt{AC * AC + BC * BC} \quad \text{(from right-angled triangle)[29]}$$

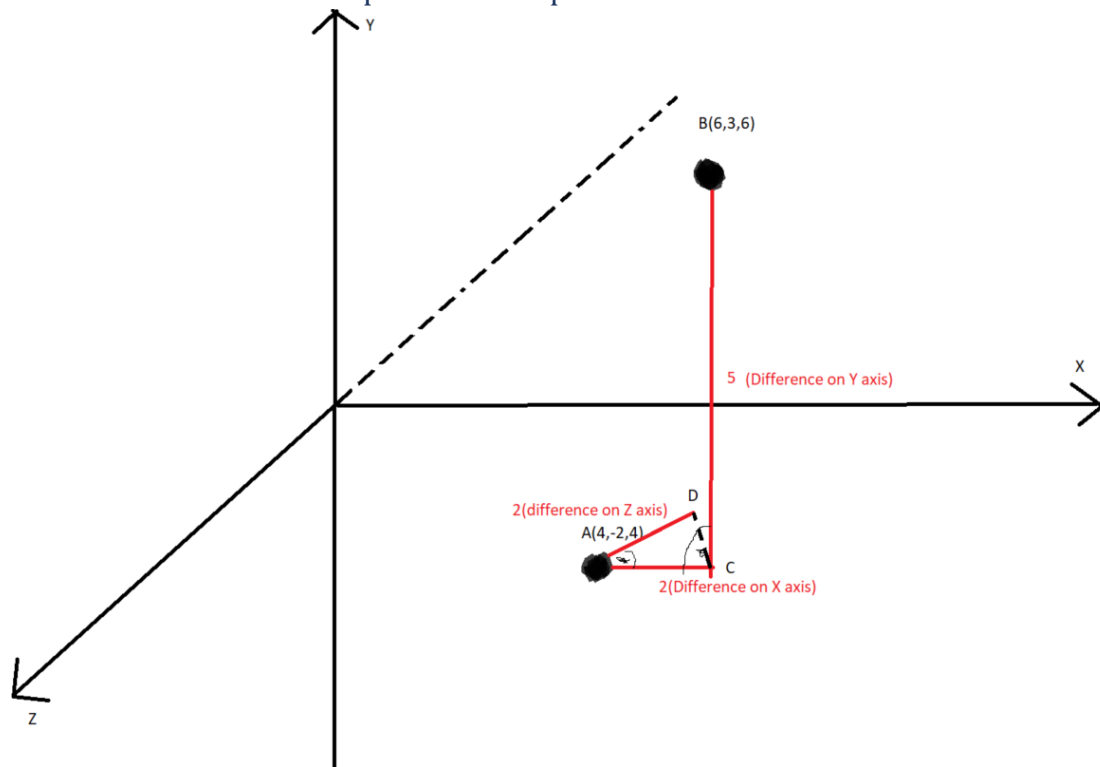
In this case it is equal to  $3\sqrt{2}$

This is also widely known as the Pythagorean theorem as the angle between AC and BC is 90 degrees, due to AC being parallel to Y axis and BC being parallel to X axis.

So, to summarize:

The distance between two points in 2D space is the square root of the total of the difference between their coordinates.

### 2.11.3 The distance between two points in 3D space



#### Distance between two points in 3D space

The way we find the distance between two points in 3D space is really similar to the way we do it in 2D space, with the difference that in 3D space we have another dimension depth(Z).

So, if we want to find the distance between two points – A and B, we need to first find what are the differences of the two points by each dimension or what is the distance from at each dimension from one point to the other.

In our example the two points have the following coordinates:  $A(4, -2, 4)$ ,  $B(6, 3, 6)$ , where  $P(x, y, z)$  is the point at 3D space.

First the Y axis: The Y coordinate of A is -2 and the Y coordinate of B is 3, so the overall distance between them is 5 units.

The X axis: The X coordinate of A is 4 and the X coordinate of B is 6, so the overall distance between them is 2 units.

The Z axis: The Z coordinate of A is 4 and the Z coordinate of B is 6, so the overall distance between them is 2 units.

On the picture you can see the lines AD, which is parallel to the Z axis and equals to 2, AC, which is parallel to the X axis and is equal to 2 and BC, which is parallel to the Y axis.

Now we can find the distance between the A and B by this formula:

$$D = \sqrt[2]{AC * AC + BC * BC + AD * AD} \quad [29]$$

Where AC, BC and AD are the distance at each individual axis x, y, and z.

We can also look at this as calculating the distance between two points at two different 2D spaces.

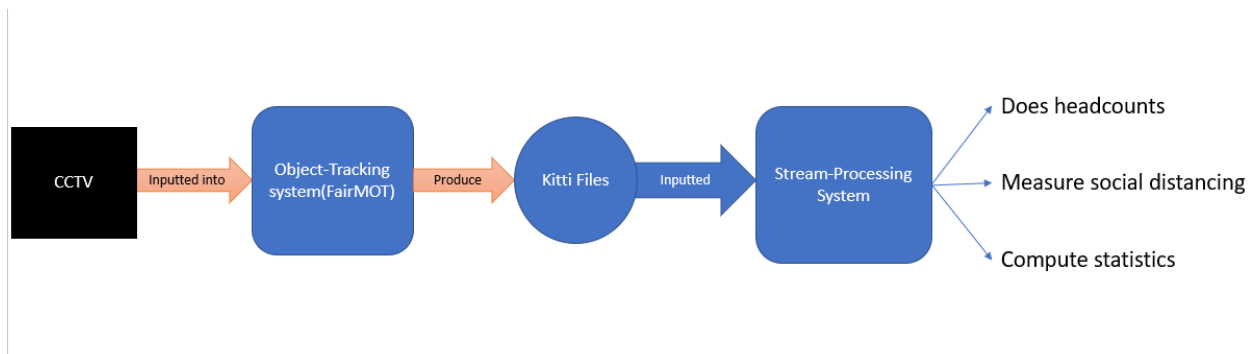
## 3 Implementation

This chapter represents the implementation of the system design.

### 3.1 System Design

This chapter will describe the design of the system created in this project in order to give a high-level explanation of its components and why they are needed.

The system consists of two main components or two smaller sub-systems – object-tracking system that consumes raw CCTV data and produces Kitti files, and a stream processing system that processes these Kitti files to produce headcounts, measure social distancing, create statistics in real time. The system is represented in the diagram below.



#### System Design diagram

##### 3.1.1 CCTV data

CCTV data is the raw video data coming from a CCTV camera. This is the starting point of the object tracking system as this raw video data is about to be processed so, computation can be done on it.

##### 3.1.2 Object-Tracking System (FairMOT)

FairMOT is the object-tracking , machine learning based software that is used to identify objects and track them. This software is used to detect objects in each frame ,draw bounding boxes around them and produce object-detection data in Kitti format.

##### 3.1.3 Kitti Files

This is the data format that holds all of the detection data as space separated values.

### 3.1.4 Stream-Processing system

The stream processing system is used for the analysis of the produced Kitti files in real-time in order to compute headcounts, measure social distancing and produce alerts when social distancing is violated. It could also be used to produce statistics about how often social distancing is violated in real-time.

## 3.2 Development environment

This section of the implementation chapter I will show how the programming environment was set up in order to work on the project. It will also discuss the operating system in which the programming was done.

### 3.2.1 Operating system

At first the operating system in which the project would be created was Windows. This was because all of the software that was going to be used in the project(Listed in the background section.) was compatible with windows and I had more experience with programming on windows, so It looked like the safer option. However, during the environment setup of the project a lot of windows specific issues occurred.

Some of them were:

- All of the installation guidelines and examples on the repositories and documentation of FairMOT and Faust were written for Linux type console, so converting Linux console commands to windows console commands had to be done every time.
- Another drawback was that windows error messages give far less information about the error than Linux shell messages.

These problems were solved by installing a Linux shell for windows from the windows store. However more issues occurred like:

- Zookeeper wouldn't start because of windows permission settings.
- Kafka cluster won't find the running instance of zookeeper.
- Running a script that is not on the same disk as the zookeeper/ Kafka would not work.
- FairMOT installation fails because it's backbone framework DCNv2 was not able to be installed on windows therefore it is incompatible with windows.

Those problems led to having the worry about how to make it happen in Windows rather than how to make it make it happen at all. More often the problem I found myself in was looking for how to do something on windows rather than how am I supposed to do it at all.

So, a decision was made to change the operation system from windows to Linux.

Ubuntu Linux was installed on a separate machine solving all of the problems listed above.

Some of the pros of Ubuntu are:

1. Ubuntu's command line  
Ubuntu's command line is better than windows. It is much easier to use and errors that occur have more detailed feedback than in windows.
2. Python 3

Ubuntu Linux comes with already installed python 3 that is needed in the project. The natural way of running python scripts is to do it directly from the shell, which is what was needed.

3. Ubuntu is free.

Ubuntu free, so if environments setup were unsuccessful, it would not cost anything to revert back to windows.

### 3.2.2 Object-Tracking system environment

In order to use the FairMOT object-tracking software it first needs to be properly installed on your system. In this section I will go through the installation of FairMOT on Ubuntu. I must also mention that this installation was performed on Windows as well, but it was unsuccessful, due to compatibility problem of the backbone framework of FairMOT with Windows.

#### 3.2.2.1 Installation

In other to install the FairMOT software I first installed Anaconda. It is used to manage programing environments and to install data science and machine learning packages from a cloud-based repository.

##### 3.2.2.1.1 Anaconda

Anaconda is a distribution of the Python and R programing languages for scientific computing like machine learning applications, data science applications, large-scale data processing and predictive analytics. Anaconda aims to simplify packet management and deployment. It is suitable for windows, Linux and macOS.[30]

I used anaconda to create a programing environment in which FairMOT's requirements would be installed.

This was done with the command:

Conda create -n FairMOT

And the

Conda activate FairMOT

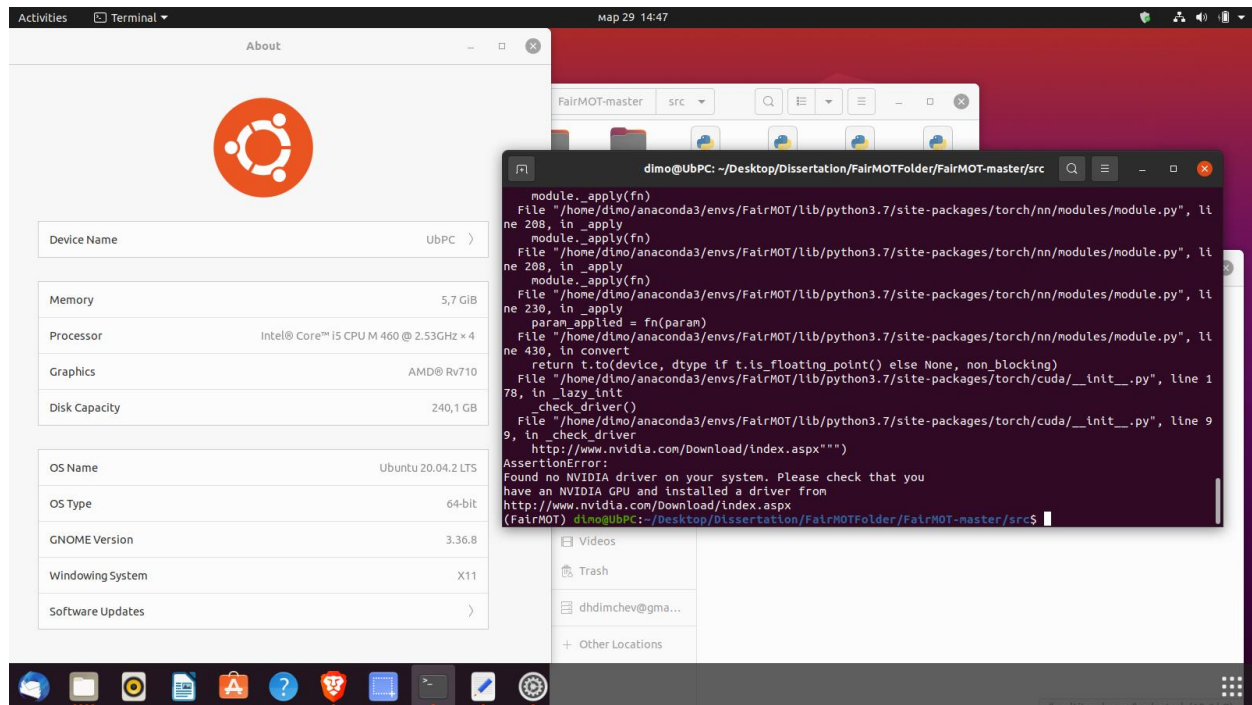
To activate the environment and use the requirements that are installed in it.

##### 3.2.2.1.2 FFMPEG

FFMPEG is the leading multimedia framework. It is able to decode, encode, transcode, mux, demux, stream, filter, and play anything that humans and machines have created.[31]

After the requirements were installed and the backbone of FairMOT was cloned(DCNv2) I installed FFMPEG, and everything was ready to run.

However due to Covid restriction I was not able to continue any further, because I didn't have the access to the hardware I needed – a NVIDIA Jetson board. This was in the original project plan, but it proved impossible to gain access due to the continuation of Covid restrictions throughout the project.



Ubuntu screenshot

The screenshot shows that the FairMOT software will not run unless there is an Nvidia video card, which I didn't have access to.

### 3.2.3 Stream Processing system environment

In order to create stream processing pipelines, we need to have Faust installed.

#### 3.2.3.1 [Faust](#)

First in order to install Faust PIP was installed.

Pip is the standard package manager for Python that allows you to install and manage additional packages that are not part of the standard python library. In this case pip for python 3 is install as we need python 3 async/await functionality ,which is a requirement by the Faust library.[32]

Pip was installed with the command:

```
sudo apt install python3-pip
```



Then Faust was installed with the command:

Pip install -U faust

### 3.2.3.2 [Kafka](#)

The Faust library uses Kafka as backend, so in order to use faust we need to have a Kafka server up and running. This means that we need to install Kafka as well.

This is done by first installing a java development kit on ubuntu and then downloading apache Kafka from the Kafka website and extracting it in a directory of choice.

### 3.2.3.3 [Zookeeper](#)

Zookeeper is the software that Kafka uses to coordinate tasks.

It is installed by:

1. Creating a new superuser in Ubuntu that is going to be used for zookeeper and giving it admin privileges.

This is done with the commands:

```
usermod -aG sudo zookeeper  
sudo getent group sudo
```

2. Creating a Zookeeper data directory

This is the place where zookeeper will be installed .This is also the place where zookeeper would be run from.

This is done with the commands:

```
sudo mkdir -p /data/zookeeper  
chown -R zookeeper: zookeeper /data/ zookeeper
```

With those two commands zookeeper directory is created and then this directory's privileges are given to the newly made superuser zookeeper.

3. Downloading and installing Zookeeper in the created data directory
4. Creating a zoo.cfg file in the zookeeper directory and configuring it to standalone mode.

This was done in order to set up zookeeper's configuration in standalone mode also known as developing/testing mode. The following .cfg configuration is mostly for developing environments.

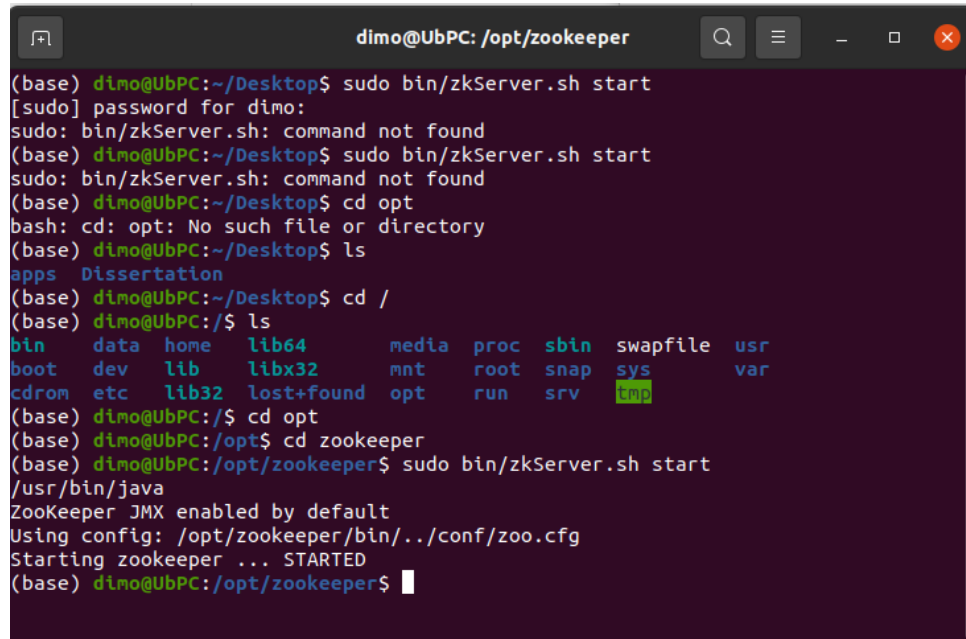
### 3.2.3.4 [Running Faust scripts in the environment.](#)

Once Kafka and Zookeeper are installed Faust apps are ready to be run.

This is done by :

1. Starting Zookeeper

Sudo bin/zkServer.sh start

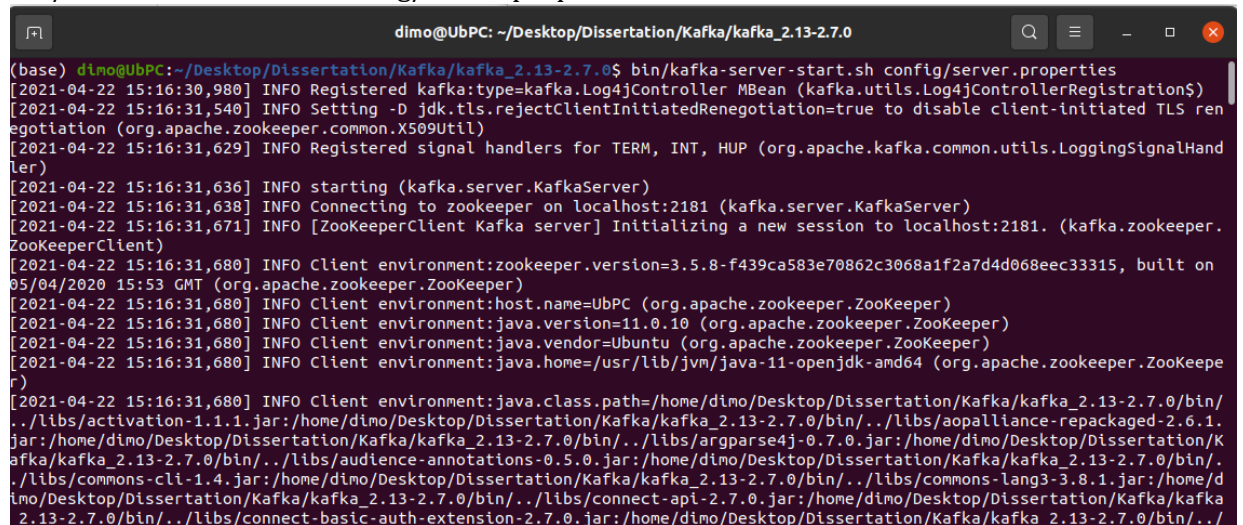


```
(base) dimo@UbPC:~/Desktop$ sudo bin/zkServer.sh start
[sudo] password for dimo:
sudo: bin/zkServer.sh: command not found
(base) dimo@UbPC:~/Desktop$ sudo bin/zkServer.sh start
sudo: bin/zkServer.sh: command not found
(base) dimo@UbPC:~/Desktop$ cd opt
bash: cd: opt: No such file or directory
(base) dimo@UbPC:~/Desktop$ ls
apps  Dissertation
(base) dimo@UbPC:~/Desktop$ cd /
(base) dimo@UbPC:/$ ls
bin    data  home  lib64  media  proc  sbin  swapfile  usr
boot  dev   lib   libx32  mnt    root  snap  sys       var
cdrom  etc   lib32  lost+found  opt    run   srv   tmp
(base) dimo@UbPC:/$ cd opt
(base) dimo@UbPC:/opt$ cd zookeeper
(base) dimo@UbPC:/opt/zookeeper$ sudo bin/zkServer.sh start
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /opt/zookeeper/bin/./conf/zoo.cfg
Starting zookeeper ... STARTED
(base) dimo@UbPC:/opt/zookeeper$
```

Console screenshot 1

## 2. Starting Kafka server

Bin/kafka-server-start.sh config/server.properties



```
(base) dimo@UbPC:~/Desktop/Dissertation/Kafka/kafka_2.13-2.7.0$ bin/kafka-server-start.sh config/server.properties
[2021-04-22 15:16:30,980] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2021-04-22 15:16:31,540] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Util)
[2021-04-22 15:16:31,629] INFO Registered signal handlers for TERM, INT, HUP (org.apache.kafka.common.utils.LoggingSignalHandler)
[2021-04-22 15:16:31,636] INFO starting (kafka.server.KafkaServer)
[2021-04-22 15:16:31,638] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2021-04-22 15:16:31,671] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2021-04-22 15:16:31,680] INFO Client environment:zookeeper.version=3.5.8-f439ca583e70862c3068a1f2a7d4d068eec33315, built on 05/04/2020 15:53 GMT (org.apache.zookeeper.ZooKeeper)
[2021-04-22 15:16:31,680] INFO Client environment:host.name=UbPC (org.apache.zookeeper.ZooKeeper)
[2021-04-22 15:16:31,680] INFO Client environment:java.version=11.0.10 (org.apache.zookeeper.ZooKeeper)
[2021-04-22 15:16:31,680] INFO Client environment:java.vendor=Ubuntu (org.apache.zookeeper.ZooKeeper)
[2021-04-22 15:16:31,680] INFO Client environment:java.home=/usr/lib/jvm/java-11-openjdk-amd64 (org.apache.zookeeper.ZooKeeper)
[2021-04-22 15:16:31,680] INFO Client environment:java.class.path=/home/dimo/Desktop/Dissertation/Kafka/kafka_2.13-2.7.0/bin/./libs/activation-1.1.1.jar:/home/dimo/Desktop/Dissertation/Kafka/kafka_2.13-2.7.0/bin/./libs/aopalliance-repackaged-2.6.1.jar:/home/dimo/Desktop/Dissertation/Kafka/kafka_2.13-2.7.0/bin/./libs/argparse4j-0.7.0.jar:/home/dimo/Desktop/Dissertation/Kafka/kafka_2.13-2.7.0/bin/./libs/audience-annotations-0.5.0.jar:/home/dimo/Desktop/Dissertation/Kafka/kafka_2.13-2.7.0/bin/./libs/commons-cli-1.4.jar:/home/dimo/Desktop/Dissertation/Kafka/kafka_2.13-2.7.0/bin/./libs/commons-lang3-3.8.1.jar:/home/dimo/Desktop/Dissertation/Kafka/kafka_2.13-2.7.0/bin/./libs/connect-api-2.7.0.jar:/home/dimo/Desktop/Dissertation/Kafka/kafka_2.13-2.7.0/bin/./libs/connect-basic-auth-extension-2.7.0.jar:/home/dimo/Desktop/Dissertation/Kafka/kafka_2.13-2.7.0/bin/./
```

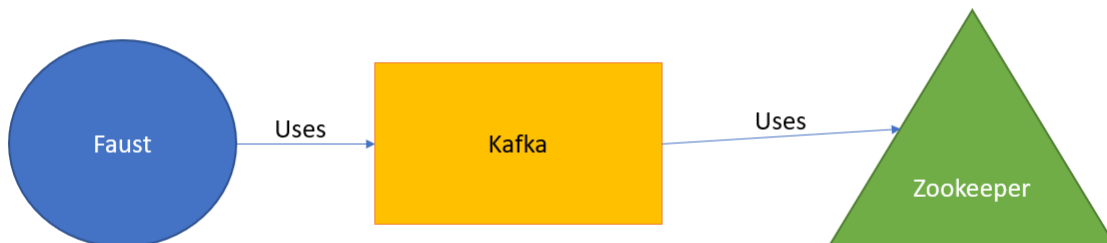
Console screenshot 2

## 3. Running Faust applications.

Faust -A example worker

Faust applications will not work unless there is a running instance of zookeeper and Kafka server on the system. This is because Faust uses Kafka and Kafka uses Zookeeper in order to function.

So, Zookeeper and Kafka are Faust's dependencies that have to run in the background when Faust is used.



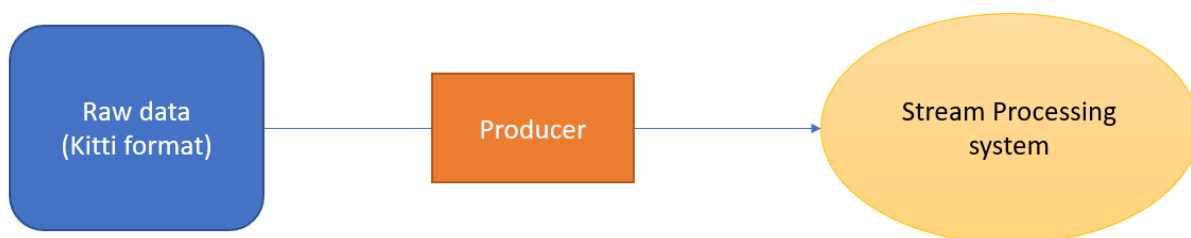
**Faust/Kafka/Zookeeper diagram**

### 3.3 Development

This section will show the development done in this project as well as explain the thought process behind it.

#### 3.3.1 Feeding data into the Stream Processing System

In order to do stream processing first data must enter the stream.



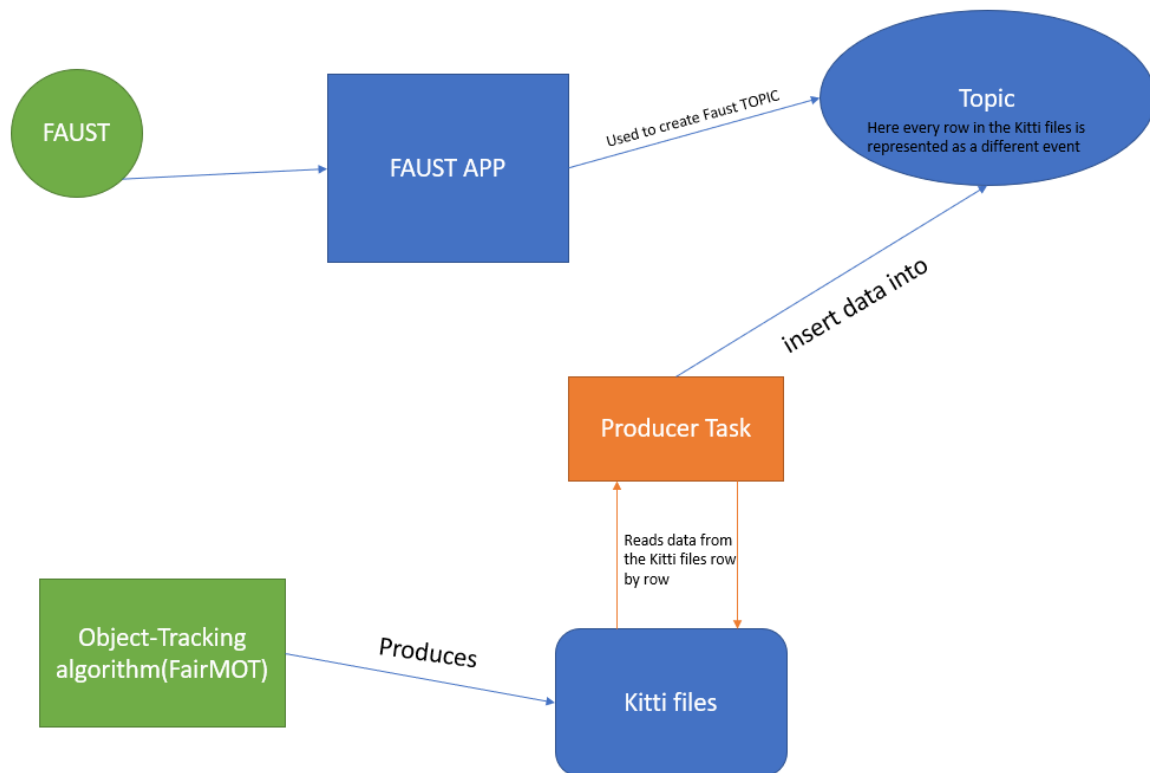
**producer diagram**

In the diagram above the “Producer” rectangle represents the program that connects the stream processing system with the object detection/identification system. This “Producer” program is responsible for feeding data into the stream processing system by sending Kitti files as events into a topic.

In other to create the producer a Faust application was created called “Kitti\_producer”. An application in Faust is an instance of the Faust library that provides the core API of Faust.

Then a new Topic is created, which is a stream name where data would be published.

Next the task of the producer program was created, which is to read data line by line from the Kitti files and send that data to the newly created topic.



### producer diagram 1

After creating the producer this is what it looks like in python code.

```
1 import faust
2 from aiofile import LineReader, AIOFile
3 from time import sleep
4
5 app = faust.App('Kitti_producer', broker='kafka://localhost:9092')
6 topic = app.topic('kitti_files', value_serializer='raw')
7
8
9 # noinspection PyTypeChecker
10 @app.task
11 async def load_in_stream():
12     async with AIOFile('/home/dimo/PycharmProjects/pythonProject/inputs/Kitti Files/Buses/test.txt', 'r',
13                       encoding="utf-8") as afp:
14         async for line in LineReader(afp):
15             await topic.send(value=line, value_serializer='raw')
16             print('success')
17             print(line)
18             sleep(5)
19
20
21 if __name__ == '__main__':
22     app.main()
```

### producer in python

Note: The print() and sleep() statements were added for better debugging and readability in the console.

This is what the console output looks like:

**producer console**

### 3.3.2 Representing Kitti files as custom Records in a stream

First a Faust Record was created. A Faust record is a model of a dictionary type in python that holds data as keys and values of a certain type. This Faust record would hold all of the fields contained in the Kitti file format as well as some extra fields that would be used for later computation.

37

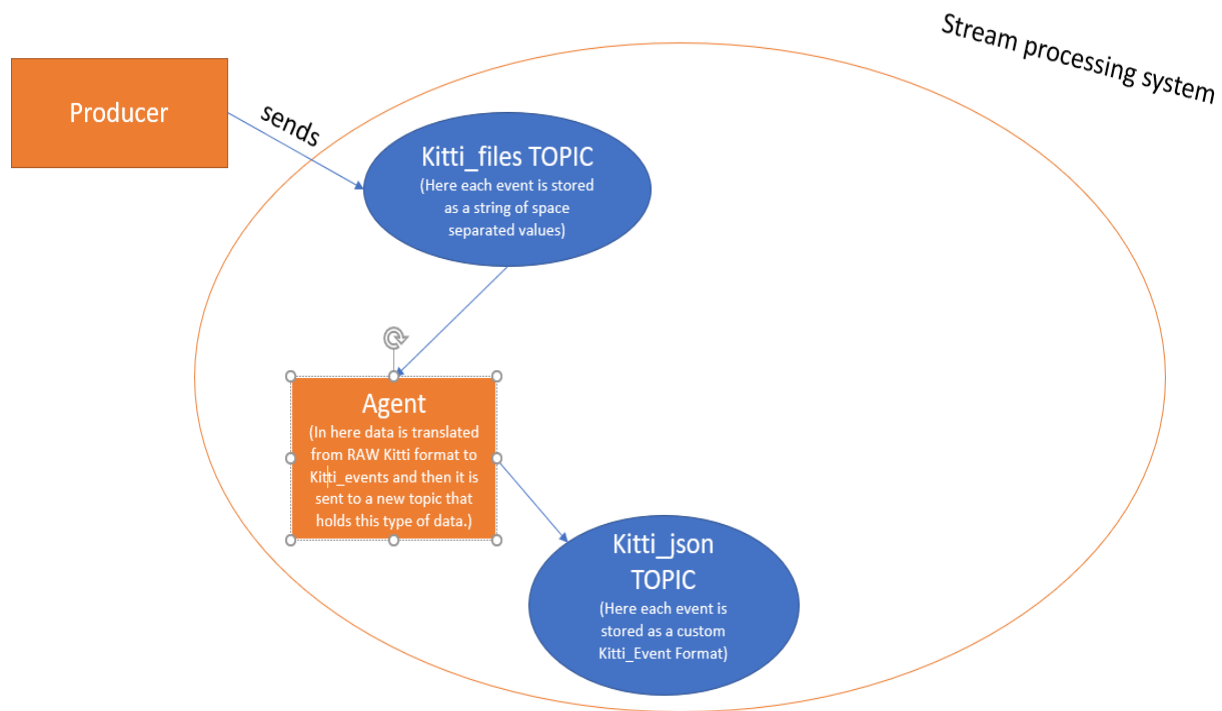
```

4
5
6 # Here create a record that will hold kitti events as Faust records.
7 class KittEvent(faust.Record, ABC, serializer='json'):
8     frame_id: int # The number of the current frame of the camera
9     track_id: int # The unique tracking id of an object
10    type: str # Type of the object(pedestrian)
11    truncated: int
12    occluded: int
13    Alpha: int
14    top_left_x: float # The coordinates of the top-left and bottom right pixels of the bounding box around the object.
15    top_left_y: float
16    bottom_right_x: float
17    bottom_right_y: float
18    # The following fields are negative in the kitti files
19    # ( This means that they currently don't exist in the kitti files.),
20    # but I still assign them, so if in the future they are added they can be used.
21    dimension_height_m: float
22    dimension_width_m: float
23    dimension_length_m: float
24    location_x_m: float
25    location_y_m: float
26    location_z_m: float
27    score: int
28    obj_height_pix: float = 0
29    obj_center_pix_x: float = 0
30    obj_center_pix_y: float = 0
31

```

### Kitti\_event record in python code 1

A stream processor also known as agent was created to consume the raw Kitti files from the "kitti\_files" topic and turn them into Kitti\_events. These Kitti events would then be sent to a new topic called "kitti\_json" for later processing.



### Stream processing system implementation 1

This is how an agent looks in python code:

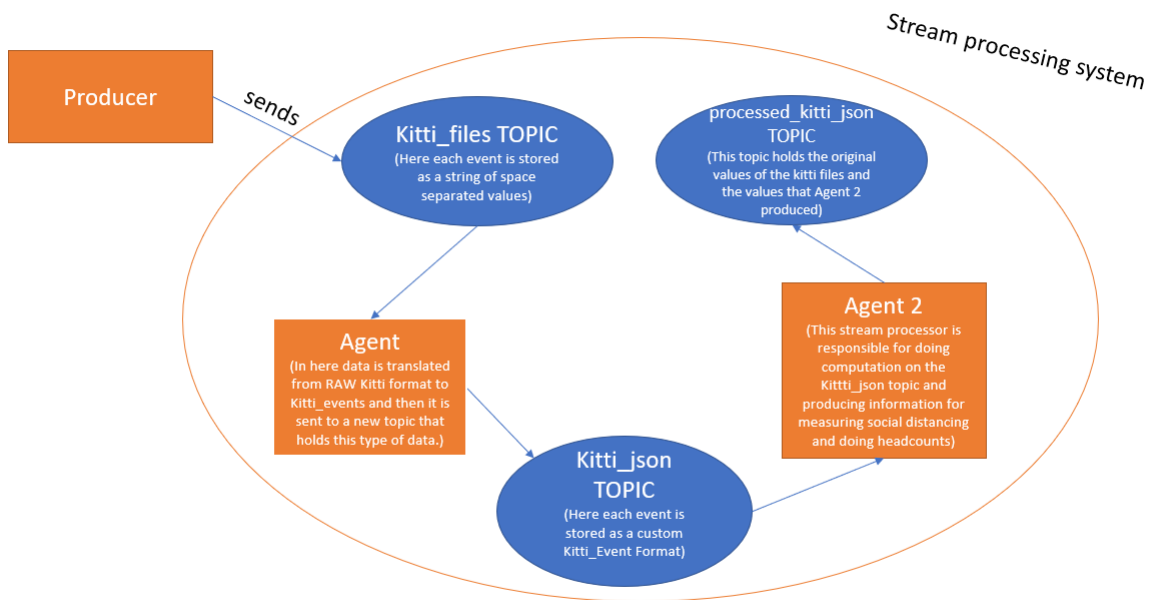
```
19 @app.agent(consume_topic)
20 async def convert_json_kitti(stream):
21     async for kitti in stream:
22         # print(kitti.decode('utf-8'))
23         split_list = (kitti.decode('utf-8')).split()
24         '''
25         This prints the kitti event before it is sent to its topic (Used for testing if the values are correct)
26         kit_prt = Kitti_Event(split_list[0], split_list[1], split_list[2], split_list[3], split_list[4], split_list[5],
27                               split_list[6], split_list[7], split_list[8], split_list[9], split_list[10],
28                               split_list[11],
29                               split_list[12], split_list[13], split_list[14], split_list[15], split_list[16])
30         print(kit_prt)
31         '''
32         await json_event_values_topic.send(
33             value=Records.Kitti_Event(split_list[0], split_list[1], split_list[2], split_list[3], split_list[4],
34                                       split_list[5],
35                                       split_list[6], split_list[7], split_list[8], split_list[9], split_list[10],
36                                       split_list[11],
37                                       split_list[12], split_list[13], split_list[14], split_list[15], split_list[16]),
38             value_serializer='json')
39     ''
```

### Agent 1

What this agent does is it takes each event from the kitti\_files topic and creates an array with the string split python command. Every value of the array is assigned to the corresponding field in the Kitti event and is sent to the new topic.

### 3.3.3 Calculating Center Point and BBOX height

After the data is stored in the kitti\_json topic it goes through another stream processor (Agent 2) that computes some useful information about the objects like height of the object in pixels and center point coordinates of the object. This information is again stored as a custom Kitti\_Event and sent to another topic called processed\_kitti\_json. This is done because later this information would be used for recovering depth of the picture and measuring social distancing.

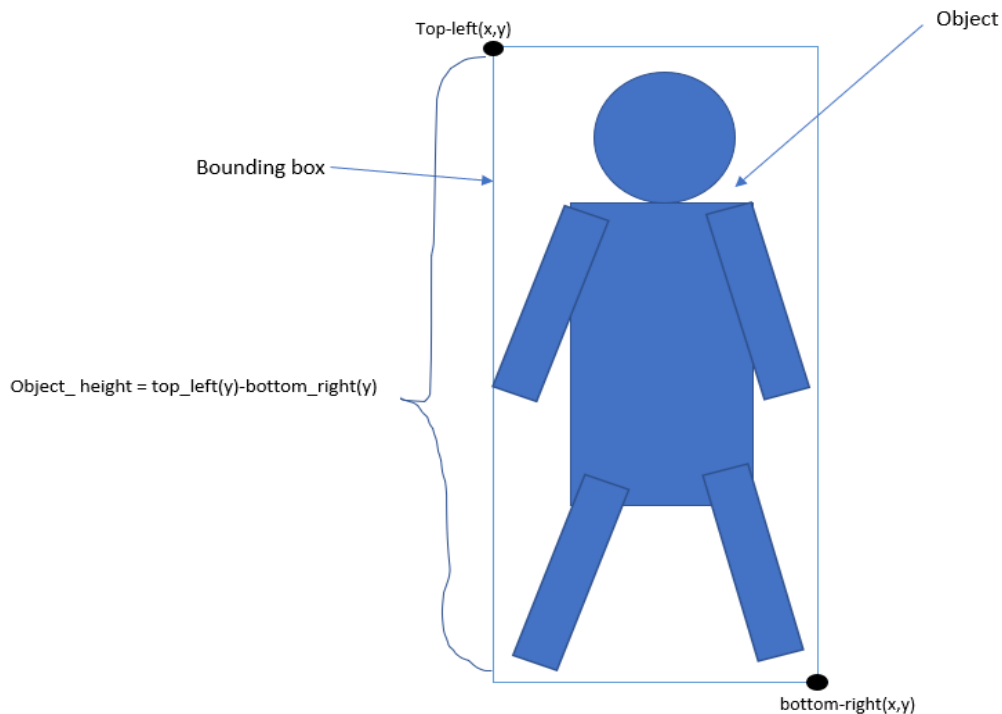


### Stream processing system implementation 2

### 3.3.3.1 Calculating Object Height in Pixels in Agent 2

Calculating object's pixel height must be done in order to later calculate the depth of the image and eventually measure social distance.

This is done by computing the absolute difference between the Y values of the top\_right and bottom\_left point of the object.



#### Object height diagram

This is how this looks like in python code.

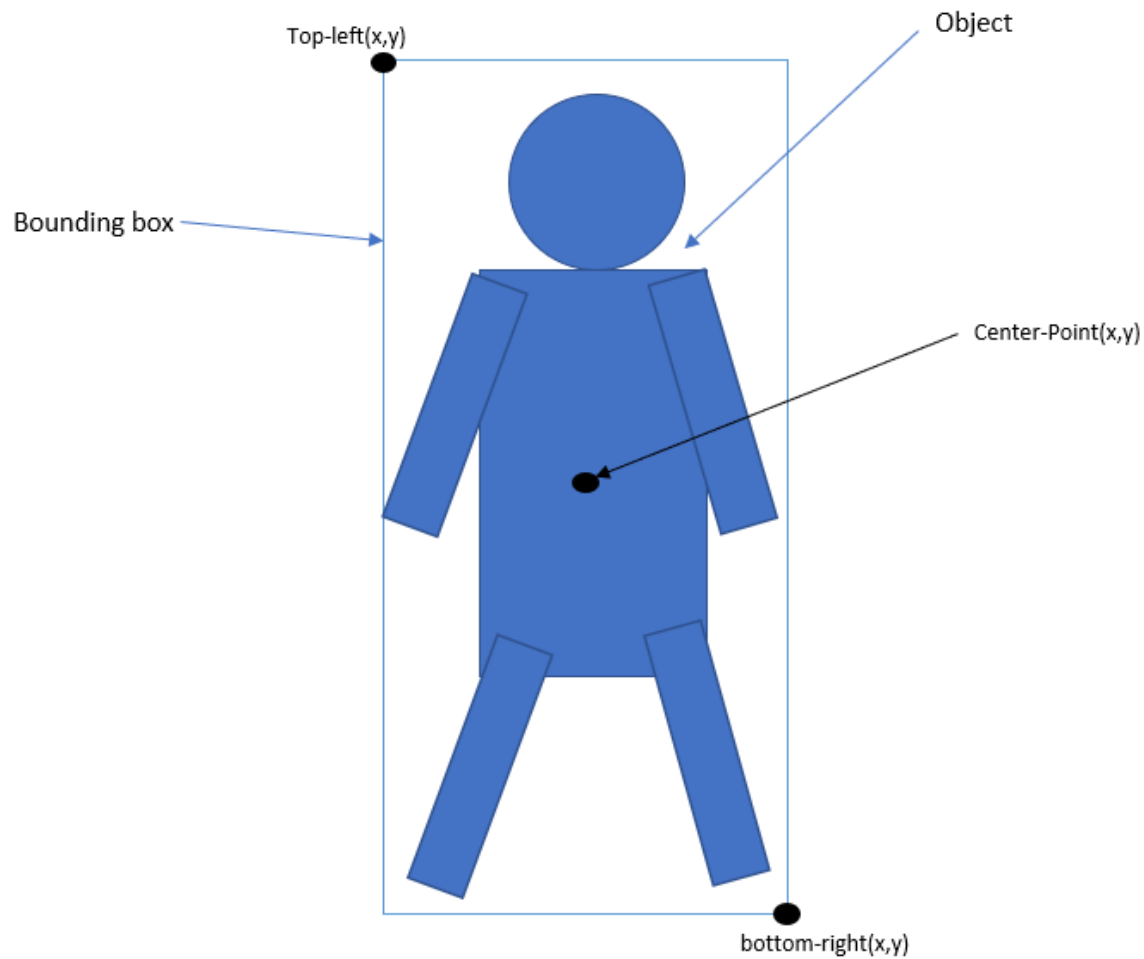
```
# returns an absolute value of the height of an object in pixels
def obj_height_pixels(y1, y2):
    return abs(y1 - y2)
```

#### Object height function in python

Then this information is saved on the custom Kitti\_Event in order to be used again later.



### 3.3.3.2 Calculating Center Point in Agent 2



**Object center point figure 1**

In order to have a better understanding of where the object is in the picture, we must compute its Center Point coordinates. Center point coordinates are a pair of pixel coordinates  $p(x,y)$  that represent the center of the bounding box surrounding the object. This is done by extracting the coordinates of the top – left and bottom right point from the Kitti files and finding the center of the object in pixel coordinates in the following way:

1. Computing the absolute difference between bottom-right and top-left point  
This is done by calculating the absolute difference in pixels of bottom-right and top-left X values and Y values. The absolute X difference stands for the width of the object and the absolute Y difference stands for the height of the object.

$$height = TopLeft(Y) - BottomRight(Y)$$

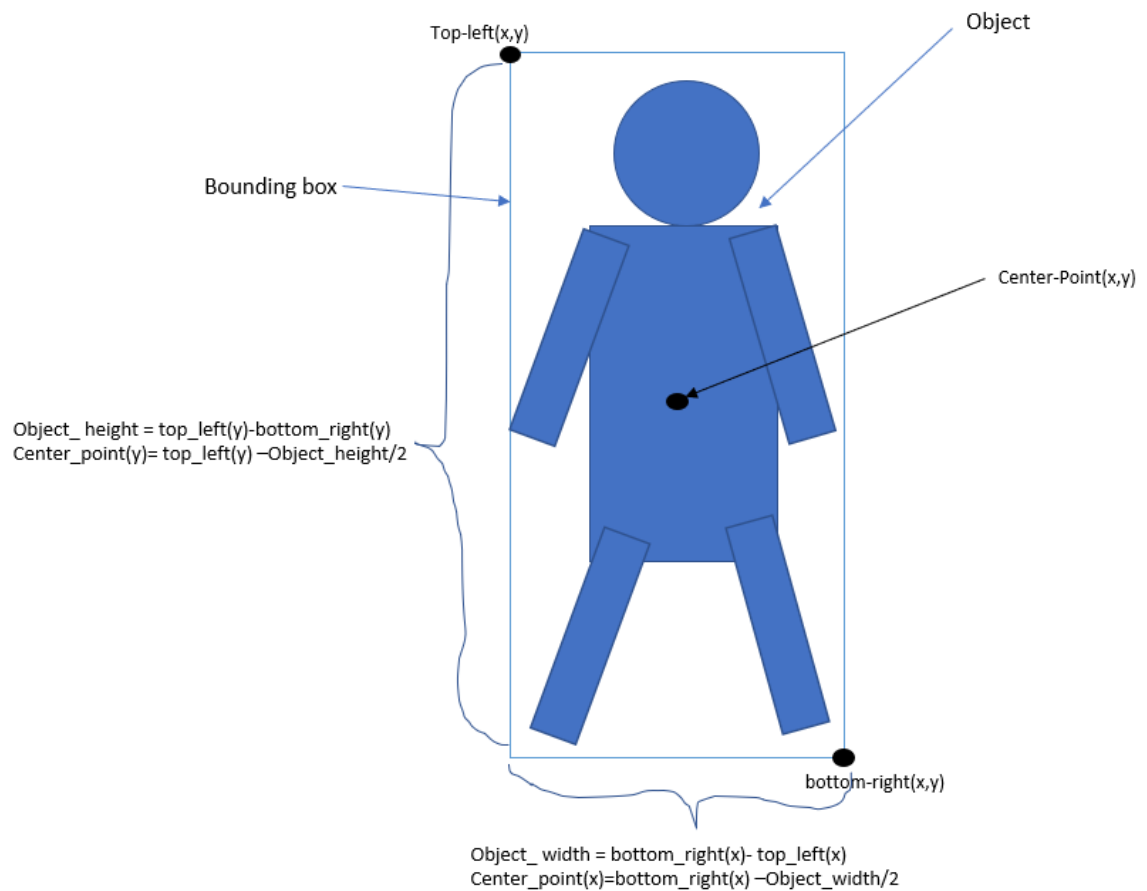
$$width = BottomRight(X) - TopLeft(X)$$

2. Finding center-point X value

Center-point x value is found by first dividing the full object width by 2 to find half of the object's width. It is assumed that bottom-right X always would have a higher value than the top-left X due to how a basic coordinate system works. Subtracting half of the object width from the bottom-right X coordinate of the object gives us the X value of the center point of the object.

3. Finding center-point Y value

Center-point Y value is found by first dividing the full object height by 2 to find half of the object's height. It is assumed that Top-left Y always would have a higher value than the bottom-right Y due to how a basic coordinate system works. Subtracting half of the object height from the Top-left Y coordinate of the object gives us the Y value of the center point of the object.



Object center point figure 2

#### 4. Saving the (X,Y) center point

After the X,Y center point is found it is saved as a record called Center\_Point. Then this record is used to hold the data until it is eventually saved in the Kitti\_Event record, where the center point coordinates would later be used.

This is how this looks like in python code.

On the first code screenshot you can see how the idea described above is implemented in python by first taking the x and y values of the top left and bottom right point and calculating the center point of the object in pixels.

```
def find_object_center_pixels(x1, y1, x2, y2):
    dif_x = abs(x1 - x2)
    dif_y = abs(y1 - y2)
    x: float
    y: float
    if x1 > x2:
        x = x1 - (dif_x / 2)
    else:
        x = x2 - (dif_x / 2)
    if y1 > y2:
        y = y1 - (dif_y / 2)
    else:
        y = y2 - (dif_y / 2)
    return Records.center_point(x, y)
```

#### Object center point in python

After calculating the center point in pixels, it is returned as a center\_point(x,y) record. You can see how this record looks in Python on the screenshot below.

```
class center_point(faust.Record, ABC):
    x: float
    y: float
```

#### Object center point Record in python

### 3.3.4 Calculating Depth with pedestrian height estimation

In order to calculate Social distancing, the depth of the image must be recovered. There are several ways to do that. Some of them are to use multiple cameras or to know the metrics of a vehicle and

translate the real-world coordinate into pixel coordinates. I will talk about these Ideas in the future proposed work part of the conclusions chapter.

In this chapter I will focus on the method I used for recovering depth, which is a photogrammetric position estimation method. This method was shown to me by Nik Khadijah Nik Aznan who recommended me a paper called “A programmatic approach for real-time 3D localization and tracking of pedestrians in monocular infrared imagery.”[34]

This paper suggested a way of recovering real world coordinates(X,Y,Z) from knowing object's height in pixels(on the picture),image dimensions, the focal length of the camera and estimation on the object height in meters. The method is displayed in more detail in the background section, here I will describe the implementation of this method in my program.

The formula used for recovering depth is:

$$Z = \frac{f * Y}{y}$$

Where Z is dept of the picture, f is the focal length (m),Y(m) is the estimated height of an object and y(pix) is the height in pixels of the object.

The problem with this equation is that it gives us the depth in *meter<sup>2</sup>*/pix format. To compare social distancing the depth needs to be in meters. So focal length of the camera has to be converted from meters to pixels in order to have the answer of the formula in meters.

(1)

$$Z = \frac{f(m)*Y(m)}{y(pix)} \longrightarrow \text{We get Z in } \frac{meters^2}{pixels}$$

By converting focal length from meters to pixels we get Z in meters!

$$Z = \frac{f(\cancel{pix})*Y(m)}{Y(\cancel{pix})} \longrightarrow \text{We get Z in meters!}$$

Focal length was converted to pixels by this formula:  
(2)

$$f' = \frac{\text{image width}(\text{pix}) * \text{focal length}(\text{mm})}{\text{sensor width}(\text{mm})}$$

This is the formula used to convert focal length from mm to pixels!

The formula used takes the width of the image(in pixels), the focal length of the camera (in millimeters) and the sensor width of the camera (in millimeters).

After converting focal length from mm to pixels we can simply substitute in the formula (1) and find recover the depth of the image.

This is how this was executed in Python code.

```
def convert_focal_length_mm_to_pixels(image_width_pix, focal_length_mm, sensor_width_mm):  
    return image_width_pix * focal_length_mm / sensor_width_mm
```

#### Focal length to pixels python 1

The code snippet above demonstrates how focal length is converted from mm to pixels.

# Returns Z or the object depth of an object

```
def object_depth_Z(obj_height_meters, obj_height_pix, focal_length_pixels):  
    return focal_length_pixels * obj_height_meters / obj_height_pix
```

#### Recovering object depth python 1

The code snippet above demonstrates how object depth is recovered in meters.

#### 3.3.5 Calculating object location in the real world

We know that all objects are imaged under a standard perspective projection[34]:

$$x = \frac{f * X}{Z} \quad y = \frac{f * Y}{Z}$$

Where X(m),Y(m) and Z(m) are the real-world object position in 3D space, x(pix) and y(pix) are center point of the bounding box of the object and f is focal length(mm).

We have already calculated object depth  $Z$  in the last step so if we substitute in these formulas, we will recover the  $(X,Y,Z)$  position of the object in the real world relative to the camera.

This is how this was done in python.

```
def find_object_center_meters(x_center_pix, y_center_pix, focal_length_pixels, obj_depth):
    x_m = x_center_pix * obj_depth / focal_length_pixels
    y_m = y_center_pix * obj_depth / focal_length_pixels

    return Records.position_3d(x_m, y_m, obj_depth)
```

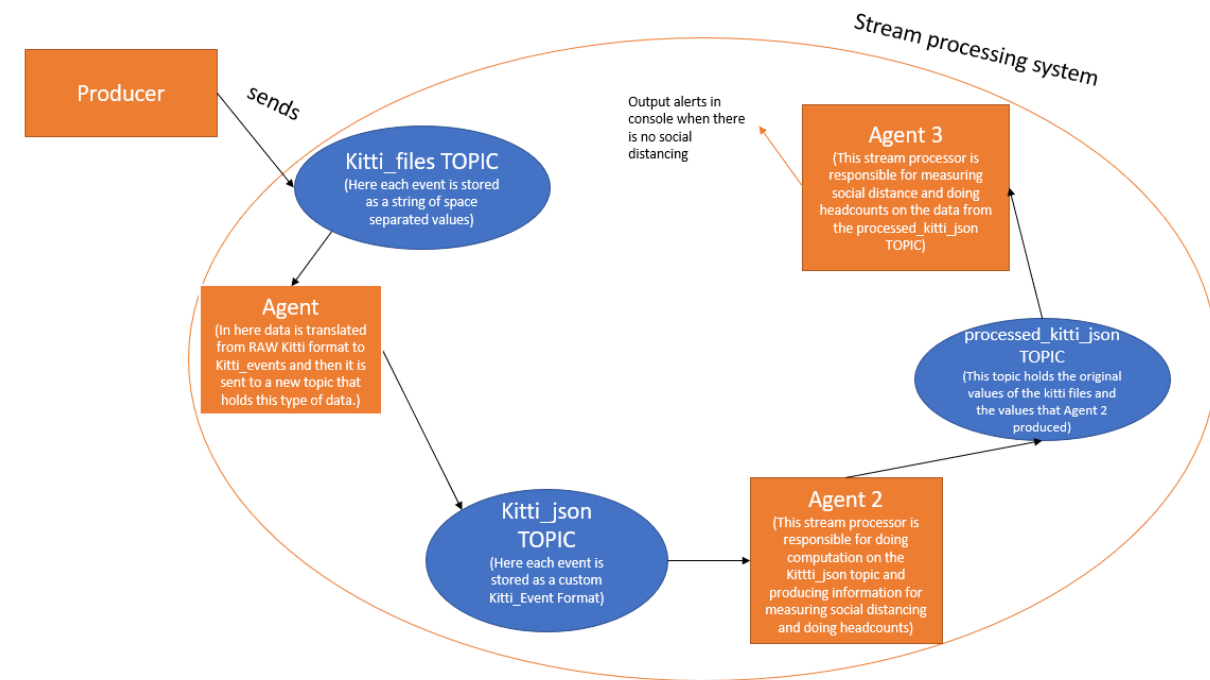
### Recovering location python

This screenshot shows how the above formulas are applied and are returned as a Faust record that holds the  $p(X,Y,Z)$  representing the position of an object in 3D space.

Later this record is used to add these  $(X,Y,Z)$  coordinates to the Kitti files.

### 3.3.6 Social Distancing

Once depth is recovered and  $(X,Y,Z)$  coordinates of an object in the real world relative to the camera are calculated and sent to the processed\_kitti\_json topic, we can start measuring social distancing.



### Stream processing system implementation 3

A third agent is created that processes the processed\_kitti\_json topic in order to measure social distance and produce alerts when social distance is not present. This is achieved through

windowing. Windowing is when state is preserved over a defined window of time. In Faust, this concept exists as a windowing table that preserves state over a defined window of time. This allows us to measure social distancing between events that are in the same time window.

However, there were problems with using Faust's windowing table:

- No custom time window.  
In Faust windowing tables were using real time(seconds, minutes, hours), but for this project seconds would not do the trick, because Kitti Files are already ordered by frames(frame\_id) and it will be more convenient to use frames for timestamp.
- Problems with partitioning  
There were problems with creating a windowed table in Faust, where a table would be created with for example 1 partition and the topic for this table would be with 1 partition, but the Kafka log to that topic would be created with 8 partitions(by default). This is a Faust issue that a lot of people have and there is still no proper solution it.

Because of these problems with Faust's windowed table, I had to write my own windowed table. The windowed table was created as a python dictionary that holds all the events as values and the keys were the tracking identities'(track\_id) of each event.

The following pseudocode describes how this windowing table works.

- 1.Event5 is processed in stream processor(Agent 3)
2. For each event in window:  
    if Event5.frame\_id – event.frame\_id >= frameWindow  
        remove event  
    else  
        measure\_social\_distancing(Event5,event)
3. Add Event5 to window

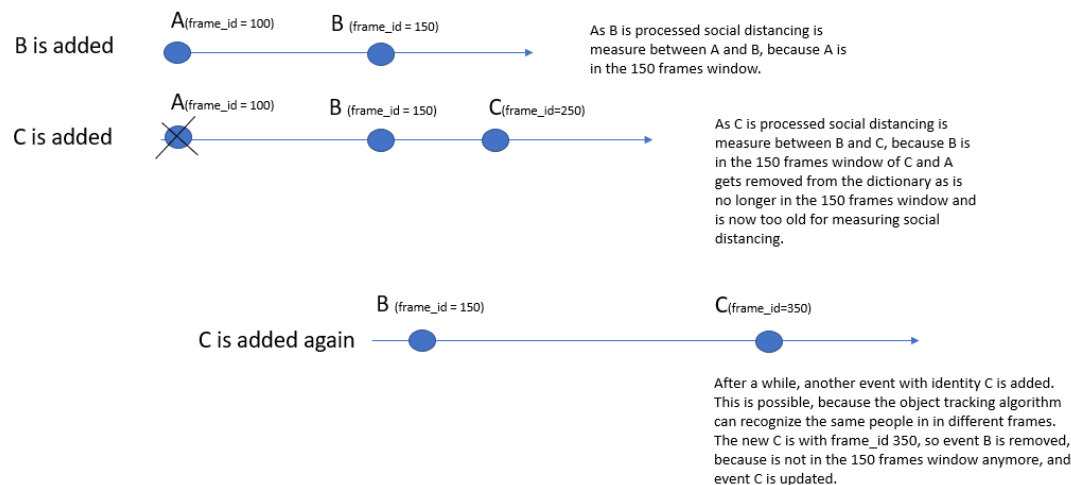
### Windowing table pseudocode

Pseudocode explanation:

Events come in the stream ordered by frame id from the Kitti files, and also the stream is grouped by frame id again before being processed in agent 3. So, each subsequent event would have a larger frame id that the previous ones. So Event5 from the pseudocode should have the biggest frame id out of every event in the window(the window is the dictionary that events are stored). And in step (2) if the difference between the newly added Event5 and another event in the dictionary is larger

or equal to frameWindow(by default frameWindow is 150)frames then the event is removed from the dictionary, because it is no longer in the desired frame window. Else if the difference is less than 150 social distancing is measured.

And last(step 3) the event is added to the window. I must mention that If there exist an event with the same key(tracking identity) in the window(dictionary) it just gets updated with the new value, this is how dictionaries work in python.



## Windowing table explanation

The diagram above shows when are removed, added, and updated in the dictionary. It also shows when social distancing is measured.

Next, I will describe how social distancing is measured.

When two events enter the `measure_social_distancing()` function in the above pseudocode., social distancing is calculated by calculating the distance between two points in 3D space. And if the distance is below 1 meter an alert is printed in the console.

The code snippets bellow demonstrates how windowing was implemented in python. Comments that explain the code are in green color and print statements are in light blue color. You can also see that in the end all the cases that violate social distancing are sent in pairs to a new topic called `social_distancing_violated_topic`, so if needed this data could be used to produce statistics.



```
# Stream processor is initialised and the topic is inputted
```

```
@app.agent(processed_events_json_topic)
async def process_kitti_events_in_table(stream):
```

```
#For each event in the stream processor grouped by frame.id
```

```
    async for event in stream.group_by(Records.Kitti_Event.frame_id):
```

```
        # if an entry with that key does not exist in the dictionary
```

```
        if event.track_id not in list(config.dictionary_ev):
```

```
            config.overall_headcounts += 1 # Increments the counter of the overall distinct object
```

```
headcounts
```

```
        for key in list(config.dictionary_ev):
```

```
            #if there is an item in the dictionary that is within the frameWindow – measure social distancing.
```

```
            #IF there is an item that is not in the frame window remove it from the dictionary.
```

```
            #frame_Distance is set to 150 at the moment
```

```
            if abs(int(config.dictionary_ev[key].frame_id) - int(event.frame_id)) <= config.frame_Distance:
```

```
                config.measure_social_distance_count += 1
```

```
                print('VinIF Social Distancing will be measured and the item will be added to the dictionary')
```

```
#Calculating the distance between two points in 3D space
```

```
    distance = functions.distance_between_2points_in_3d_space(float(event.location_x_m),
```

```
                                                                float(event.location_y_m),
```

```
                                                                float(event.location_z_m),
```

```
                                                                float(config.dictionary_ev[
```

```
                                                                    key].location_x_m),
```

```
                                                                float(config.dictionary_ev[
```

```
                                                                    key].location_y_m),
```

```
                                                                float(config.dictionary_ev[
```

```
                                                                    key].location_z_m))
```

```
    if distance >= 1: #if distance between the two objects is more or equal to 1 meter
```

```
        print('There is social distancing')
```

```
    else:
```

```
        config.violated_social_distance_count += 1
```

```
        print(f'\n\t'
```

```
            f'Social Distancing Violated in frame {event.frame_id},\n\t'
```

```
            f'No social distancing between object id{event.track_id}'
```

```
            f'and object id {config.dictionary_ev[key].track_id}')
```

```
        holder = Records.violate_s_distance_holder(event.frame_id,
config.dictionary_ev[key].frame_id,
```

```
                                                    event.track_id, config.dictionary_ev[key].track_id,
```

```
                                                    config.frame_Distance
```

```
        )
```

```
        #The frame ids of both objects that violated social distancing are sent to a new topic
where they can be processed again
```

**Windowing table implementation in python 1**

```

        config.dictionary_ev.pop(key) #Remove event from dictionary
        print('1 Item was removed from the dict 1')
    else: # If even.track_id is a key in events
        for key in list(config.dictionary_ev):
            #if there is an item in the dictionary that is within the frameWindow – measure social distancing.

            #IF there is an item that is not in the frame window remove it from the dictionary.

            #config.frame_Distance is set to 150 at the moment

            if abs(int(config.dictionary_ev[key].frame_id) - int(
                event.frame_id)) <= config.frame_Distance and key != event.track_id:
                print('VinEl Social distancing will be measured')

            #Calculating the distance between two points in 3D space
            distance = functions.distance_between_2points_in_3d_space(float(event.location_x_m),
                                                                    float(event.location_y_m),
                                                                    float(event.location_z_m),
                                                                    float(config.dictionary_ev[
                                                                    key].location_x_m),
                                                                    float(config.dictionary_ev[
                                                                    key].location_y_m),
                                                                    float(config.dictionary_ev[
                                                                    key].location_z_m))

            if distance >= 1: #if distance between the two objects is more or equal to 1 meter
                print('There is social distancing')

            else:
                print(f'\n\t'
                    f'Social Distancing Violated in frame {event.frame_id},\n\t'
                    f'No social distancing between object id {event.track_id} \t'
                    f'and object id {config.dictionary_ev[key].track_id}')
                holder = Records.violate_s_distance_holder(event.frame_id,
                    config.dictionary_ev[key].frame_id,
                    event.track_id, config.dictionary_ev[key].track_id,
                    config.frame_Distance
                )

            #The frame Ids of both objects that violated social distancing are sent to a new topic where they can be
            processed again

            await social_distancing_violated_topic.send(value=holder, value_serializer='json')
        else:
            config.dictionary_ev.pop(key)
            print('2 Item was removed from the dict 2')
        config.dictionary_ev[event.track_id] = event
        # print(dictionary_ev) – this is used for debugging of the dictionary.

```

## Windowing table implementation in python 2

```
def distance_between_2points_in_3d_space(x1, y1, z1, x2, y2, z2):
    return math.sqrt((x1 - x2) ** 2 + (y1 - y2) ** 2 + (z1 - z2) ** 2)
```

### Distance between two points in 3D space

This is the formula for calculating the distance between 2 points in 3D space implemented in python.

### 3.3.7 Headcounts

Headcounts are computed along with social distancing. This is done by creating a variable that gets updated every time a new tracking identity occurs. Also, a timer was created that periodically executes the tasks of printing the current headcounts (different tracking ids in the frame window) and overall tracking ids detected by the program.

This is how this was done in python.

```
@app.timer(interval=10)
async def print_headcounts():
    functions.headcounts()
```

### headcounts timer

```
def headcounts():
    print(f" \r\n The headcounts for the last {config.frame_Distance} frames: \n\t {len(list(config.dictionary_ev))}"
          f" \n And the overall headcounts since the consumer is running are: \n\t {config.overall_headcounts}"
    )
```

### headcounts function

This code would periodically print the overall headcounts and the headcounts in the currently in the dictionary.

### 3.3.8 Statistics

Once social distancing and headcounts are computed multiple statistics could be produced.

The statistics I produced compare how often social distance is violated. This is achieved with counters that increment themselves every time social distance is measured and every time that is violated.

Then another timer task was written that would compute the statistics and print them out every couple of seconds.

```
@app.timer(interval=30)
async def print_stats():
    print(f'Social distancing is violated {statsistics.how_often_violated() * 100} % of the time! ')
```

### Statistics timer

```
# Returns the floating point percentage of how many violations of social distancing are happening compared to the
# number of measuring
def how_often_violated():
    violated_measure: int = config.violated_social_distance_count
    overall_measure: int = config.measure_social_distance_count
    return violated_measure / overall_measure
```

### Statistics function

The above screenshots show how this was achieved in python.

Violated social distance count and measure social distance count are counters in a separate config file that increment every time social distance is violated and or measured.

## 4 Results and Evaluation

In this section I will look at the results generated by each part of the stream processing system and evaluate their quality.

### 4.1 Feeding data into the system(Producer program)

Here I will evaluate if my producer program works properly.

Expected behavior:

Read data from the Kitti files and send that exact data to a topic in the stream processing system.

Actual behavior:

As expected, it reads the data from the Kitti files and sends it to a topic.

This was tested by first adding a print statement that prints the data being send in the console and then when it is received in the topic an agent prints the value received.

This behavior can be seen in the following screenshots.

```
(base) dlnogubPC:~/PycharmProjects/pythonProject$ python3 produce_Kitti_To_Stream.py --datadir=/home/dlno/PycharmProjects/pythonProject/WorkerDir1 worker --web-port=6066
faust v1.10.4
id      Kittl_producer
transport [URL('kafka://%SClocalhost:9092')]
store
web      http://ubpc:6066
log      -stderr- (warn)
pid      3939
hostname UbPC
platform CPython 3.8.5 (Linux x86_64)
drivers
transport aiotkafka=1.1.6
web        aiohttp=3.7.4.post0
datadir   /home/dlno/PycharmProjects/pythonProject/WorkerDir1
appdir     /home/dlno/PycharmProjects/pythonProject/WorkerDir1/v1

starting>
[2021-04-27 15:46:47,242] [3939] [WARNING] success
[2021-04-27 15:46:47,243] [3939] [WARNING] 70 6 pedestrian 0 0 -10 836.452432999081 88.85084075927733 1014.2818443446688 546.2047225952149 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:46:52,253] [3939] [WARNING] success
[2021-04-27 15:46:52,253] [3939] [WARNING] 71 6 pedestrian 0 0 -10 816.6752235204365 112.36906880842673 996.1938279114722 573.5971044903963 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:46:57,263] [3939] [WARNING] success
[2021-04-27 15:46:57,264] [3939] [WARNING] 72 6 pedestrian 0 0 -10 778.3599327527724 126.9615438945603 987.9703014180704 663.9205017211443 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:47:02,270] [3939] [WARNING] success
[2021-04-27 15:47:02,270] [3939] [WARNING] 73 6 pedestrian 0 0 -10 756.2677686012873 154.56383797158782 985.3336792258187 738.5964965024739 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:47:07,281] [3939] [WARNING] success
[2021-04-27 15:47:07,281] [3939] [WARNING] 74 6 pedestrian 0 0 -10 723.7545057768492 176.32206403623326 980.9077988413818 829.689323955751 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:47:12,291] [3939] [WARNING] success
[2021-04-27 15:47:12,292] [3939] [WARNING] 75 6 pedestrian 0 0 -10 787.5442176727507 228.5718143913124 974.499807665451 900.9234504177772 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:47:17,299] [3939] [WARNING] success
[2021-04-27 15:47:17,300] [3939] [WARNING] 76 6 pedestrian 0 0 -10 780.0455805671052 279.0680007324873 1050.6052339399673 953.0235612187585 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:47:22,313] [3939] [WARNING] success
[2021-04-27 15:47:22,313] [3939] [WARNING] 998 10 pedestrian 0 0 -10 27.364667717334584 804.6689208984375 160.57146967280212 1079.672705078125 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:47:27,323] [3939] [WARNING] success
[2021-04-27 15:47:27,324] [3939] [WARNING] 999 10 pedestrian 0 0 -10 31.622910642967547 841.6307467964103 139.22257765876577 1063.940789719804 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:47:32,335] [3939] [WARNING] success
[2021-04-27 15:47:32,335] [3939] [WARNING] 1090 12 pedestrian 0 0 -10 24.70303550355483 833.7164428710937 128.50345711119127 1073.4385498046875 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:47:37,344] [3939] [WARNING] success
```

## producer program

In this screenshot you can see the producer program that feeds data into the system prints in the console every message that is being send to the consumer topic.

```
(base) dlnogubPC:~/PycharmProjects/pythonProject$ python3 consumer_Kitti_stream.py --datadir=/home/dlno/PycharmProjects/pythonProject/WorkerDir2 worker --web-port=6067
faust v1.10.4
id      faust_consumer
transport [URL('kafka://%SClocalhost:9092')]
store
web      http://ubpc:6067
log      -stderr- (warn)
pid      4045
hostname UbPC
platform CPython 3.8.5 (Linux x86_64)
drivers
transport aiotkafka=1.1.6
web        aiohttp=3.7.4.post0
datadir   /home/dlno/PycharmProjects/pythonProject/WorkerDir2
appdir     /home/dlno/PycharmProjects/pythonProject/WorkerDir2/v1

starting>
[2021-04-27 15:49:44,751] [4045] [WARNING] 70 6 pedestrian 0 0 -10 836.452432999081 88.85084075927733 1014.2818443446688 546.2047225952149 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:49:44,754] [4045] [WARNING] 71 6 pedestrian 0 0 -10 816.6752235204365 112.36906880842673 996.1938279114722 573.5971044903963 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:49:44,757] [4045] [WARNING] 72 6 pedestrian 0 0 -10 778.3599327527724 126.9615438945603 987.9703014180704 663.9205017211443 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:49:44,759] [4045] [WARNING] 73 6 pedestrian 0 0 -10 756.2677686012873 154.56383797158782 985.3336792258187 738.5964965024739 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:49:44,760] [4045] [WARNING] 74 6 pedestrian 0 0 -10 723.7545057768492 176.32206403623326 980.9077988413818 829.689323955751 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:49:44,762] [4045] [WARNING] 75 6 pedestrian 0 0 -10 787.5442176727507 228.5718143913124 974.499807665451 900.9234504177772 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:49:44,774] [4045] [WARNING] 76 6 pedestrian 0 0 -10 780.0455805671052 279.0680007324873 1050.6052339399673 953.0235612187585 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:49:44,780] [4045] [WARNING] 998 10 pedestrian 0 0 -10 27.364667717334584 804.6689208984375 160.57146967280212 1079.672705078125 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:49:44,783] [4045] [WARNING] 999 10 pedestrian 0 0 -10 31.622910642967547 841.6307467964103 139.22257765876577 1063.940789719804 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:49:44,785] [4045] [WARNING] 1090 12 pedestrian 0 0 -10 24.70303550355483 833.7164428710937 128.50345711119127 1073.4385498046875 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:49:44,786] [4045] [WARNING] 1101 14 pedestrian 0 0 -10 1653.973703174445 224.31219482421875 1740.2758329583671 379.1311950683594 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:49:44,795] [4045] [WARNING] 1103 15 pedestrian 0 0 -10 31.2003622199479 833.7779052734375 133.72630098805993 1064.0388671875 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:49:44,797] [4045] [WARNING] 1104 15 pedestrian 0 0 -10 34.33907569214363 841.1485011164176 130.09679931267655 1056.1882775381866 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:49:44,803] [4045] [WARNING] 1105 15 pedestrian 0 0 -10 32.906563394002475 843.03013289896 131.9913709032508 1065.3440131279517 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:49:44,804] [4045] [WARNING] 1106 15 pedestrian 0 0 -10 32.96655841785937 845.1717472541944 131.9667795226874 1066.9017161858612 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:49:44,806] [4045] [WARNING] 1110 15 pedestrian 0 0 -10 33.152182178744745 852.7197295452077 131.6083429821555 1072.4447125978566 -10 -10 -10 -1000 -1000 -1000 -10
[2021-04-27 15:49:44,810] [4045] [WARNING] 1319 21 pedestrian 0 0 -10 30.209898852741432 840.7407836914062 148.25939588602807 1071.8337646484374 -10 -10 -10 -1000 -1000 -1000 -10
```

## consumer program

In this screenshot we see the consumer. The consumer just prints all the messages(events) received from the producer.

From both outputs we can see that all the data sent arrived unchanged and in the same order that is being sent.

## 4.2 Representing Kitti files as custom Records in a stream

The raw data coming from the producer comes as a space separated string. Then this data is presented as a custom Record. In this chapter I check the consistency of this representation by checking if all the fields are assigned properly.

Expected behavior: All the fields are assigned properly and a Kitti\_Event Record is created.

Actual behavior: As expected.

In order to evaluate this behavior a print statement was added that prints the Kitti\_Event when it is created in order for me to check if every field is assigned as expected.

These were the raw Kitti files sent to the topic.

```
2022 49 pedestrian 0 0 -10 718.6226241971669 132.4845731068983 838.9900368731464 477.3429651044531 -10 -10 -10 -1000 -1000 -10
2023 49 pedestrian 0 0 -10 720.7171654910911 132.4478600233409 836.4601146654594 463.1891592862529 -10 -10 -10 -1000 -1000 -10
2024 49 pedestrian 0 0 -10 715.9867039419223 133.973000179992 831.959885210513 466.16374151875016 -10 -10 -10 -1000 -1000 -10
```

And when they were represented as Kitti events every field of the space separated values got assigned to a specific variable in the Kitti\_Events that has a name. This is how they look like:

```
[2021-04-27 16:11:57,187] [6349] [WARNING] <Kitti_Event: frame_id='2022', track_id='49', type='pedestrian', truncated='0', occluded='0', Alpha='-10', top_left_x='718.6226241971669', top_left_y='132.4845731068983', bottom_right_x='838.9900368731464', bottom_right_y='477.3429651044531', dimension_height_m='-10', dimension_width_m='-10', dimension_length_m='-10', location_x_m='-1000', location_y_m='-1000', location_z_m='-1000', score='-10', obj_height_pix=0, obj_center_pix_x=0, obj_center_pix_y=0>
[2021-04-27 16:11:57,190] [6349] [WARNING] <Kitti_Event: frame_id='2023', track_id='49', type='pedestrian', truncated='0', occluded='0', Alpha='-10', top_left_x='720.7171654910911', top_left_y='132.4478600233409', bottom_right_x='836.4601146654594', bottom_right_y='463.1891592862529', dimension_height_m='-10', dimension_width_m='-10', dimension_length_m='-10', location_x_m='-1000', location_y_m='-1000', location_z_m='-1000', score='-10', obj_height_pix=0, obj_center_pix_x=0, obj_center_pix_y=0>
[2021-04-27 16:11:57,192] [6349] [WARNING] <Kitti_Event: frame_id='2024', track_id='49', type='pedestrian', truncated='0', occluded='0', Alpha='-10', top_left_x='715.9867039419223', top_left_y='133.973000179992', bottom_right_x='831.959885210513', bottom_right_y='466.16374151875016', dimension_height_m='-10', dimension_width_m='-10', dimension_length_m='-10', location_x_m='-1000', location_y_m='-1000', location_z_m='-1000', score='-10', obj_height_pix=0, obj_center_pix_x=0, obj_center_pix_y=0>
```

### custom Kitti\_Event Records

## 4.3 Calculating Center Point and BBOX height

Here I evaluate if the program correctly calculates the height of the BBOX and the center point of an object.

Expected behavior: The program calculates Center Point and BBOX height correctly.

Actual behavior: As expected.

In order to check if the program is doing the calculations properly Center Point and BBOX height were calculated by hand and then compared to the results in the program.

```
[2021-04-27 16:59:23,047] [7346] [WARNING] <Kitti_Event: frame_id='70', track_id='6', type='pedestrian', truncated='0', occluded='0', Alpha='-10', top_left_x='836.452432999081', top_left_y='88.85084075927733', bottom_right_x='1014.2818443446688', bottom_right_y='546.2047225952149', dimension_height_m='-10', dimension_width_m='-10', dimension_length_m='-10', location_x_m='-1000', location_y_m='-1000', location_z_m='-1000', score='-10', obj_height_pix=0, obj_center_pix_x=0, obj_center_pix_y=0>
[2021-04-27 16:59:23,048] [7346] [WARNING] 457.35388183593756
[2021-04-27 16:59:23,048] [7346] [WARNING] <center_point: x=925.367138671875, y=317.52778167724614>
```

### Calculating center point and bbox height

From this screenshot we see the Kitti\_Event and its fields. The next message is the calculated height of the BBOX, and the last message is the center point's coordinates p(x,y).

BBOX height was calculated getting top\_left\_y and bottom\_right\_y and finding the absolute difference between them. As I look at the values, I notice that This looks like this:

546,2047225952149 - 88,85084075927733 = 457,3538818359376

Which is the exact same number for BBOX height.

Also, center point was calculated this way:

Center X = bottom\_right\_X - absolute(top\_left\_X - bottom\_right\_X)/2

Center Y = top\_left\_Y - absolute(top\_left\_Y - bottom\_right\_Y)/2

If we substitute with the values from the screenshot we get:

Center X = 925,367138671875

Center Y = 317,527781677246

Which are the values calculated by the program. So, behavior is as expected.

#### 4.4 Recovering object depth and location with pedestrian height estimation.

In this chapter I check if the calculations for object depth and 3D location are recovered correctly.

```
[2021-04-28 16:33:52,128] [5493] [WARNING] <Kitti_Event: frame_id='70', track_id='6', type='pedestrian', truncated='0', occluded='0', Alpha='-10', top_left_x='836.452432999081', top_left_y='88.85084075927733', bottom_right_x='1014.2818443446688', bottom_right_y='546.2047225952149', dimension_height_m='-10', dimension_width_m='-10', dimension_length_m='-10', location_x_m='-1000', location_y_m='-1000', location_z_m='-1000', score='-10', obj_height_pix=457.35388183593756, obj_center_pix_x=925.367138671875, obj_center_pix_y=317.52778167724614, dimension height=1.76>
[2021-04-28 16:33:52,129] [5493] [WARNING] 0.8679366954840141
[2021-04-28 16:33:52,129] [5493] [WARNING] <position_3d: x=3.5610196583981977, y=1.2219179019725128, z=0.8679366954840141>
```

#### Recovering object depth and location

The first message from the screenshot above is the Kitti\_event with all the information in it. The second message is the recovered value of object depth object depth. And the third message is the 3D position in the real world relative to the camera.

After substituting in the formulas for object depth and 3D position estimation I got the same results that got as output in in the screenshot above, so depth and location were recovered correctly.

#### 4.5 Measuring of social distance

After recovering object location correctly, social distance is calculated as the distance between 2 points in 3D space. If this distance between two objects is less than 1 meter and these objects were



detected within 150 frames window , then social distancing is violated.

```
[2021-04-28 17:04:55,557] [6511] [WARNING] VinIF Social Distancing will be measured and the item will be added to the dictionary
[2021-04-28 17:04:55,558] [6511] [WARNING] There is social distancing
[2021-04-28 17:04:55,558] [6511] [WARNING] VinIF Social Distancing will be measured and the item will be added to the dictionary
[2021-04-28 17:04:55,559] [6511] [WARNING] There is social distancing
[2021-04-28 17:04:55,559] [6511] [WARNING] VinIF Social Distancing will be measured and the item will be added to the dictionary
[2021-04-28 17:04:55,559] [6511] [WARNING] There is social distancing
[2021-04-28 17:04:55,559] [6511] [WARNING] VinIF Social Distancing will be measured and the item will be added to the dictionary
[2021-04-28 17:04:55,560] [6511] [WARNING] There is social distancing
[2021-04-28 17:04:55,560] [6511] [WARNING] VinIF Social Distancing will be measured and the item will be added to the dictionary
[2021-04-28 17:04:55,560] [6511] [WARNING] There is social distancing
[2021-04-28 17:05:00,550] [6511] [WARNING] <Kitti Event: frame_id='3873', track_id='177', type='pedestrian', truncated='0', occluded='0', Alpha='-10', top_left_x='710.7463045921339', top_left_y='159.09437255859373', bottom_right_x='819.6610074195046', bottom_right_y='427.4045471191406', dimension_height_m='-10', dimension_width_m='-10', dimension_length_m='-10', location_x_m='-1000', location_y_m='-1000', location_z_m='-1000', score='-10', obj_height_pix=267.7101745605469, obj_center_pix_x=765.2036560058593, obj_center_pix_y=293.54945983886716, dimension_height=1.76>
[2021-04-28 17:05:00,551] [6511] [WARNING] 1.482775981596816
[2021-04-28 17:05:00,551] [6511] [WARNING] <position_3d: x=5.030658385625616, y=1.9298745375086905, z=1.482775981596816>
[2021-04-28 17:05:00,567] [6511] [WARNING] VinIF Social Distancing will be measured and the item will be added to the dictionary
[2021-04-28 17:05:00,568] [6511] [WARNING] Social Distancing Violated in frame 3873,
No social distancing between object id177and object id 168
[2021-04-28 17:05:00,569] [6511] [WARNING] VinIF Social Distancing will be measured and the item will be added to the dictionary
[2021-04-28 17:05:00,570] [6511] [WARNING] There is social distancing
[2021-04-28 17:05:00,570] [6511] [WARNING] VinIF Social Distancing will be measured and the item will be added to the dictionary
[2021-04-28 17:05:00,570] [6511] [WARNING] There is social distancing
[2021-04-28 17:05:00,570] [6511] [WARNING] VinIF Social Distancing will be measured and the item will be added to the dictionary
[2021-04-28 17:05:00,571] [6511] [WARNING] Social Distancing Violated in frame 3873,
No social distancing between object id177and object id 172
[2021-04-28 17:05:00,573] [6511] [WARNING] VinIF Social Distancing will be measured and the item will be added to the dictionary
[2021-04-28 17:05:00,573] [6511] [WARNING] Social Distancing Violated in frame 3873,
No social distancing between object id177and object id 174
[2021-04-28 17:05:00,576] [6511] [WARNING] VinIF Social Distancing will be measured and the item will be added to the dictionary
[2021-04-28 17:05:00,576] [6511] [WARNING] There is social distancing
[2021-04-28 17:05:01,497] [6511] [WARNING] Social distancing is violated 55.95854922279793 % of the time!
```

I wrote some print statements for debugging that would print messages when social distancing is measured, not violated, violated and in which frame is violated. You can see them in the screenshot above.

However, I tried testing if these results were true the following way:

When there is a social distance violated message, I looked at the frame Ids of the objects that violated the social distance and checked:

- If these objects were in the N-frame window

The N-frame window is the maximum difference in frame ids that two objects can have in order measure social distancing between them. By default, the N-frame window is set to 150 frames, however It is a variable that could be changed. For example, if one object is last detected on frame 1 and another object is detected after 300 frames and the N-frame window is 150 frames social distancing would not be measured between them, because the object detected in frame 1 would be too old and would not exist in the picture when the object in frame 300 gets detected.

If social distance is measured between two objects, then they must be detected in a N-frame window from each other. If they are not, then the windowing table is not working correctly.

This worked as expected. All the frames that violated social distancing were in 150 frame windows from each other, so windowing table is working correctly.

- If the social distance between the two objects is calculated correctly.

I checked if the social distance is calculated correctly by taking the values for 3D position from the objects and calculating the distance between these 2 points in 3D space by substituting in the formula and solving it with calculator.

The program worked as expected as every time I checked social distance was calculated correctly.



## 4.6 Headcounts

Headcounts are two types – overall headcounts ( all the headcounts that were detected by the system ) and headcounts currently in the window(all the headcounts in the last 150 frames ).

Headcounts are just a counter that gets increased every time an event with new tracking id gets added. So, I checked If the headcounts matched the number of events added with unique tracking Id.

Overall headcounts worked as expected.

Headcounts currently in the window also worked as expected, however there is a potential problem, because all headcounts are stored in memory and not on Kafka. If the python script crashes all of the headcounts would be lost and the counter would be set to zero. This potential problem occurs, because the windowing table was created by me and is not from the functionality of the Faust library. This potential problem could also be solved by storing the headcounts data in a Kafka topic that has Kafka log , so they can be recovered later if crashes occur.

## 4.7 Statistics

The only thing that was not working correctly here was that statistics task would sometimes execute too early when there is not any social distance measuring done and when it tries to produce the statistic (violated social distance/ all measured cases )% it would crash the program, because there would be zero measuring of social distance done and you can't divide by zero.

This was solved by adding a statement that checks if the overall measuring of social distance was done more than 0 times and if it was not it returns.

Other than that, the statistics behave as expected when the function for statistics was run with random values and would work the same way with the actual counters from the Kitti files.

Statistics said that social distancing was observed around 50-60 percent of the time. This percentage would change as more data is added, but for the most part it remains around 50 to 60 percent.

## 4.8 Social Distance, headcounts, statistics results compared to the raw video data

In this part of the Results and evaluation chapter I will compare the data produced by the stream processing system by processing the Kitti Files to what can be visually seen on the raw video data tapes that the Kitti files were generated from.

Some important points about this result evaluation:

- Video tapes were recorded at 4 frames per second instead of the assumed 30

When designing the windowing table that would determine if two objects are detected close enough in time to measure social distancing between them, it was assumed that the camera creating the videos works at 30 frames per second.

However, after receiving the original tapes, I noticed that they were recorded by the cameras at 4 frames per second. This was not that big of an issue as I was able to change the **frameWindow** in the windowing table to 4 frames, which meant that in order to measure social distance between two objects in the image they must be last detected within the same second. This is probably not optimal, so in the future it might be useful to further adjust the parameters of the algorithm and analyze for what variable value best results are achieved.

- Assumptions of **focal length** and **sensor width**.

In order to recover object depth and 3D location focal length and CCD sensor width of the CCTV camera are needed. As I couldn't get access to the original CCTV specification of the videos used to produce the Kitti files, I assumed that focal length of the CCTV camera that is used to shoot the videos is 3.6mm and the sensor width of the camera is 0.83 mm.

However, for different cameras the focal lengths and sensor widths vary and if the cameras used to generate these Kitti files have different focal length and sensor width than the ones assumed the program will recover depth incorrectly, resulting in recovering 3D location incorrectly.

To compare visually the results of the stream processing system with the raw video data the stream processing system was run with the parameters and Kitti Files listed below. After the stream processing system processed the Kitti File the frames in which social distancing was measured were extracted from the raw videos. The frames were extracted from the videos using FFMPEG with the following command:

```
ffmpeg -i <input> -vf "select=eq(n\,%Frame Number%)" -vframes 1 out.png
```

This command above extracts a specific frame of the video by its number and returns it as a portable network graphics.

The output of the stream processing system below was generated with frameWindow = 4 , focal length = 3.6 mm and sensor width = 0.83 mm.

Text that presents frames in which social distancing is violated would be painted in Red and frames that observe social distance are painted in green. Statistics are colored in yellow. Headcounts are colored in blue.

[2021-05-04 12:44:46,346] [11269] [WARNING] Social Distancing Violated in frame 1665,

No social distancing between object id25and object id 23

[2021-05-04 12:44:46,443] [11269] [WARNING] Social Distancing Violated in frame 1944,

No social distancing between object id35and object id 34

[2021-05-04 12:44:50,522] [11269] [WARNING] There is social distancing in frame 2189

[2021-05-04 12:44:50,523] [11269] [WARNING] There is social distancing in frame 2189 and there are 1 headcounts detected in the last 4

[2021-05-04 12:44:50,712] [11269] [WARNING] There is social distancing in frame 2190 and there are 1 headcounts detected in the last 4

[2021-05-04 12:44:50,825] [11269] [WARNING] There is social distancing in frame 2191 and there are 1 headcounts detected in the last 4

[2021-05-04 12:44:50,835] [11269] [WARNING] There is social distancing in frame 2192 and there are 2 headcounts detected in the last 4

[2021-05-04 12:44:55,874] [11269] [WARNING] The headcounts for the last 4 frames:

1

[2021-05-04 12:45:04,864] [11269] [WARNING] There is social distancing in frame 4337

[2021-05-04 12:45:04,983] [11269] [WARNING] There is social distancing in frame 4339 and there are 2 headcounts detected in the last 4

[2021-05-04 12:45:05,881] [11269] [WARNING] The headcounts for the last 4 frames:

1

[2021-05-04 12:45:06,838] [11269] [WARNING] Social Distancing Violated in frame 4722,

No social distancing between object id128and object id 126

[2021-05-04 12:45:06,967] [11269] [WARNING] Social Distancing Violated in frame 4723,

No social distancing between object id 128 and object id 126there are 2 headcounts detected in the last 4

[2021-05-04 12:45:15,876] [11269] [WARNING] Social distancing is observed 60.0 % of the time!



**frame 1665**

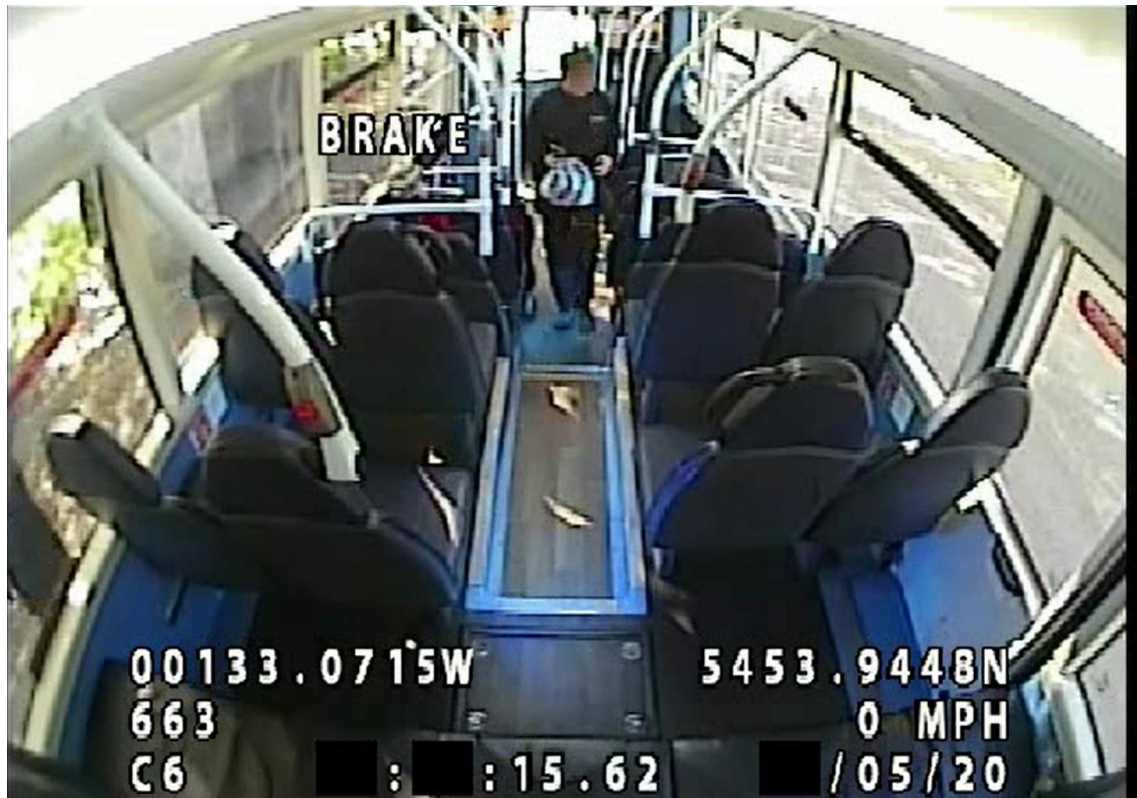
In the picture above you see frame 1665.

The analysis of the Kitti file calculated that social distance was violated in this frame. This was because the women sitting at the front were too close to each other during the frameWindow. There is another screenshot that from a couple frames back that shows more clearly where the women are and how close they were.



**screenshot from video 1**

However the above frame in which also social distance gets violated was not detected by the stream processing system. This could be happening because the algorithm needs to be tuned and analysed in order to find out the perfect config settings to get the best results.



frame 1944

In the picture above you see frame 1944.

In frame 1944 was violation of social distancing , this was because the detected objects with id 34 and 35 were to close at this point.





#### Frame 2189

In the picture above you see frame 2189, where social distancing is observed.

A couple of frames after frame 1944 the passenger moves through the bus and sits in the back resulting in measuring social distance again. The program this time calculates that there is more than 1 meter apart from the objects in the frameWindow and signals that social distance is observed.

The stream processing system also detects 2 headcounts at the moment, which is the number of objects detected in the last second.



frame 4339

In the picture above you see frame 4339, where social distancing is observed.

The stream processing system also detects 2 headcounts at the moment, which is the number of objects detected in the last second.



frame 4722

In the picture above you see frame 4722, where social distancing got violated.

In this picture the passenger in the front walked too close to another passenger sitting in the front of the bus. It could be hard to see the passenger sitting, because images are blurred due to privacy reasons, that is why I will show you a couple of frames that show the passenger(sitting) as he stands up and gets off the bus.



screenshot from video 2



In the screenshot above you see the passenger standing up.



screenshot from video 3

At screenshot above you see passenger moving towards the bus exit.



**screenshot from video 4**

In this screenshot you can see one of the passenger already left the bus and the second one is going towards the exit.



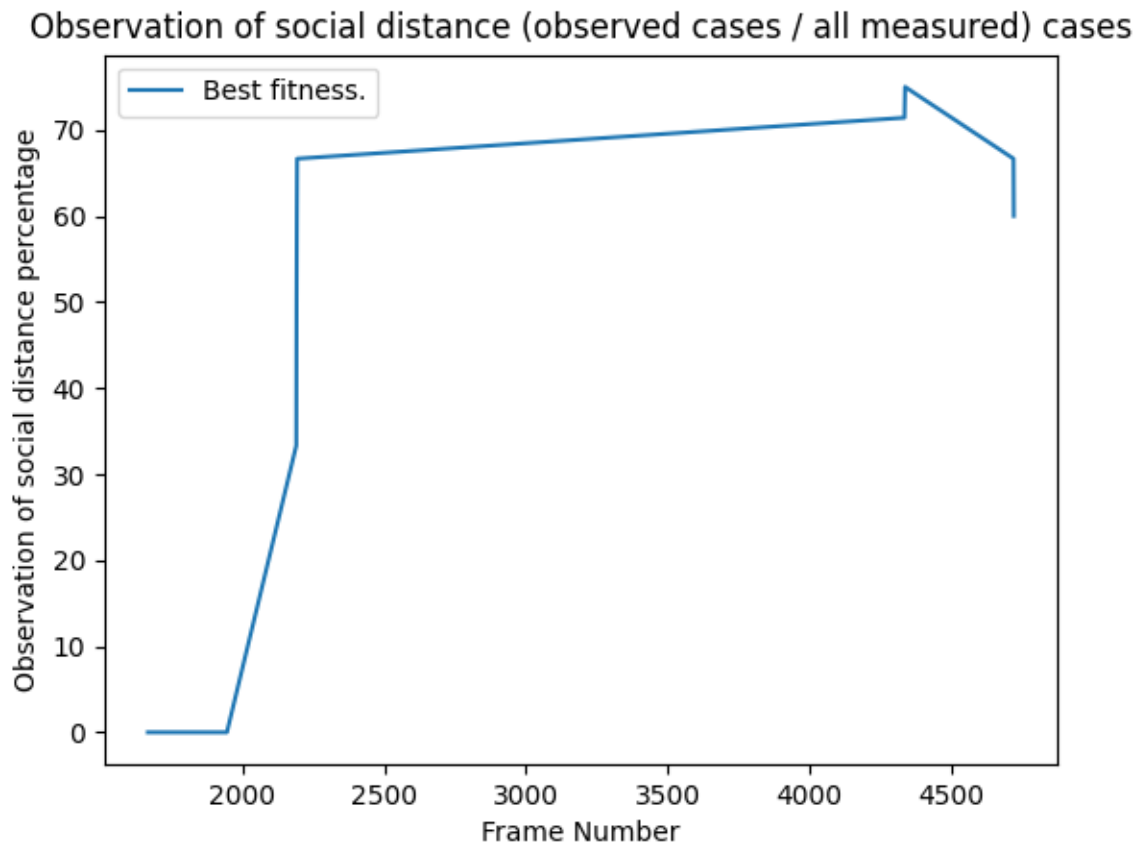
**frame 4723**

In the picture above you see frame 4723, where social distancing got violated.

This frame again shows the violation between the same passengers as in frame 4722.

The stream processing system also detects 2 headcounts at the moment, which is the number of objects detected in the last second.

Statistics:



#### Statistics figure

In the figure above you can see how the percentage of observation of social distancing changes throughout different frames from the output of the program.

At the end it goes down to 60%, which is the output generated by the stream processing system.

#### 4.9 Overall evaluation

Since I was not able to create the Videos and Kitti files myself due to not having access to the hardware needed I was only able to judge on the correctness of the results visually by watching the raw CCTV videos and extracting frames from them in order to check if the stream processing system judged the situation correctly.



Overall, the stream processing system successfully processed data in order to produce headcounts and measure social distance between objects. Statistics about the percentage of (observed social distance cases / all cases) are produced and were accurate.

However, the algorithm needs proper tuning and analysis of how changes to different parameters effect the results given. For example, the parameter for height assumption is set to 1meter and 75 centimeters, but this needs to be tested if for example making the height assumption a higher number would make the algorithm produce better results. Also, the focal length and sensor width were assumed, and these parameters should be changed according to the specification of the CCTV camera in order to achieve the best results.

This would go into future work of the project as I did not have the time to analyze and tune the algorithm during the development period.

## 5 Conclusions

### 5.1 Fulfilment of objectives

The project aim was split into 3 main objectives.

- Implement a machine learning algorithm that takes CCTV data and outputs Kitti files(FairMOT)

This objective was unfulfilled.

The original plan was to generate the Kitti files myself, but due to Covid restrictions I did not have access to the needed hardware – a NVIDIA Jetson board. Not having the hardware made this objective impossible and prevented me from fulfilling it.

- Design and implement a stream processing system that takes the Kitti files and produces alerts when social distancing is not being observed.

This objective was fulfilled by creating the stream processing system with python and Faust and using height estimation to recover the depth of the 2D image. Then 3D location of the objects in the real world was recovered and used to measure social distancing between objects and print an alert message when social distancing is violated.

- Implement the created systems on a Nvidia Jetson Device

This objective was unfulfilled.

This objective was impossible to fulfill, because due to Covid restrictions I was not able to gain access to the needed software – a NVIDIA Jetson board.

## 5.2 Personal Development

In this section I talk about what I learned throughout this project and what I could have done better.

### 5.2.1 Time management

Throughout this project I learned that time management and planning are crucial to the success of any project. I learned that sticking to the plans you made is as important as making them in the first place. A great lesson for the future will be to have a realistic plan ahead of time and actually follow it.

It would have been better if I had more soft deadlines in my project planning instead of only hard deadlines. I think that this would have led to better time management of the project.

### 5.2.2 Object-tracking Systems

By researching how object-tracking systems work I have furthered my knowledge in this area. I also gained experience in setting up such system on ubuntu and on windows.

However, due to Covid I didn't have access to the needed hardware to Implement an object tracking system and generate Kitti files with it.

Looking back, I should have considered the risks of not acquiring the needed hardware and what should I do accordingly.

### 5.2.3 Stream-processing systems

Throughout this project I learned a lot about how stream-processing systems work. I have gained experience in working with stream processors, timers, topics, and tables. I learned how I can implement a stream processing system in my python applications using Faust.

### 5.2.4 Python

Processing streams in order to calculate social distancing and headcount furthered my knowledge of Python Scripting and Dynamically typed languages in general. Learning Python Scripting was a great addition to my language repertoire as It is one of the best languages for data-science and machine learning.

### 5.2.5 Faust

Working on this project allowed me to learn Faust, which is a python framework for stream processing. Learning Faust also introduced me to Stream-processing and its concepts.

### 5.2.6 Ubuntu

Due to problems with setting up the implementation environment on windows I had to switch to Linux and learn ubuntu. This gave me a great insight in how Linux systems work and improved my knowledge of working with Linux console.

### 5.2.7 Problem solving

I have no doubt that this project made me a better problem solver. This is because I faced a lot of difficulties and things were not going my way most of the time. For example, at the beginning of the project I had trouble with the setup of the implementation environment, but I solved it by switching my operating system from Windows to Linux. I also had problems with the windowing table provided from the Faust library, so I created my own windowing table using python.

Over the course of the project, I realized that in practice not everything will be as expected and in order to succeed I must be able to break down problems and solve them my way.

## 5.3 Future Work

In this section I will propose some work that could be done in the future.

### 5.3.1 Analysis of parameters and how they affect the results of the algorithm

In the future it might a good Idea to analyze how changing parameters of the algorithm affects performance

Some of the parameters that can be tuned are:

- assumed height of a person

The assumed height of a person is needed, when recovering depth.

During the development period I did not have the time to experiment with different height estimation values and evaluate what would be the best value for the job.

Analysis could be done in order to find the best value for height that would be assumed from the algorithm.

- frameWindow

The frameWindow variable is responsible for controlling the size of the windowing table.

This value represents the time in which two objects need to be detected in order to have the stream processing system measuring the social distance between them.

- Focal length of CCTV camera

Focal length should be set according to the CCTV camera specification.

During my work on the project, I did not have access to the CCTV camera specification, so I had to assume it's 3.6 mm

- Sensor width of CCTV camera

Sensor width should be set according to the CCTV camera specification.

During my work on the project, I did not have access to the CCTV camera specification, so I had to assume it's 0.86 mm

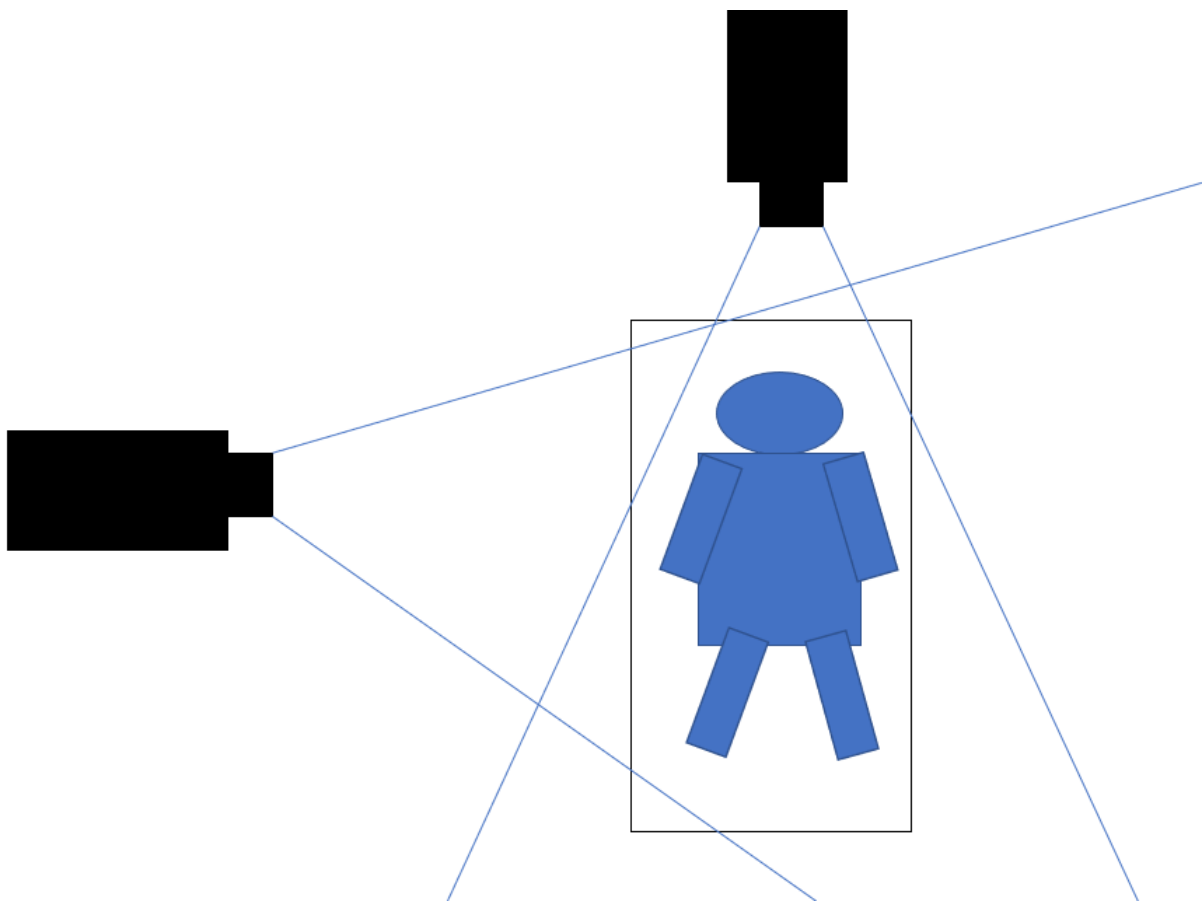
### 5.3.2 Recovering object depth and location

Recovering object depth and location in the real world relative to the camera plays a crucial role to measuring the distance between objects. In this project location was recovered through height estimation, however there are many possible ways to do that. In here I will share some creative Ideas that can be used to recover object location.

#### 5.3.2.1 Using multiple cameras from different angles

While researching how to recover the location of an object I had the Idea of using multiple cameras. However, I had to work with the Kitti files provided to me, which were generated by only one camera.

The Idea was that Object location could be recovered if we have multiple cameras that look at the object (pedestrian )from different angles. If the angles of the cameras at which they capture the object are known, then it should be possible to recover the 3D location of an object.



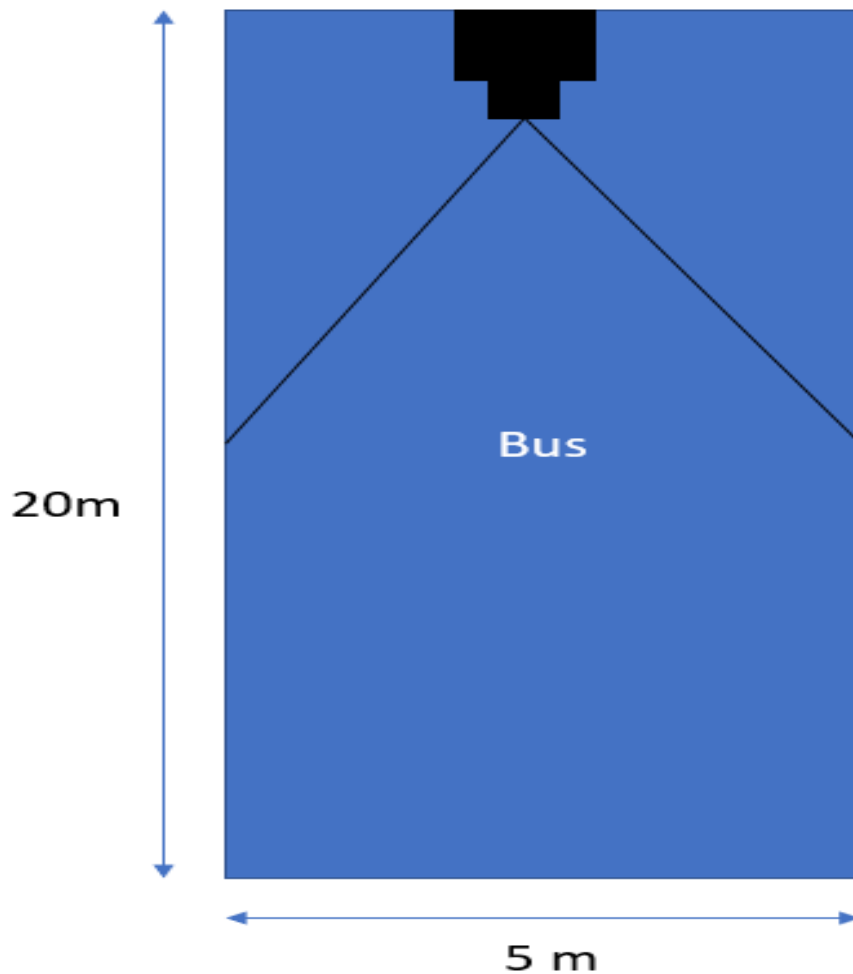
#### **multiple cams. from different angles**

As you can see in the picture above there are 2 cameras looking at the same object from different angles. If one is placed behind or in front of the object and the other is placed to the side at known angle this can be used to recover 3D location of the object relative to these cameras.



### 5.3.2.2 Knowing the metrics of the vehicle

Another Idea for recovering depth is to know the width and length of the vehicle you are measuring social distancing in and use these metrics to recover social distancing by calculating the pixel / meter ratio and applying it to calculate the location of an object.



#### Metrics of vehicle

In the screenshot above the blue rectangle represents a bus with length 20m and width 5 m. The width of a buss corresponds to the front and back of the vehicle and the length to the sides. If the width is known and the camera sees the back of the buss, then this distance(5m) could be translated to pixels and find the pixel/ meter ration this way. Then by assuming that the bus is one big rectangle depth and 3D location can be recovered by using the Pythagorean theorem.

## References:

### Stream Processing

[1] Hazelcast. (n.d.). *What Is Stream Processing? A Layman's Overview*. [online] Available at: <https://hazelcast.com/glossary/stream-processing/>

[2] GmbH (2019). *What is Stream Processing? - Ververica*. [online] Ververica.com. Available at: <https://www.ververica.com/what-is-stream-processing>.

[3] Wikipedia. (2021). Stream processing. [online] Available at: [https://en.wikipedia.org/wiki/Stream\\_processing](https://en.wikipedia.org/wiki/Stream_processing)

[4] Wikipedia Contributors (2018). *Event stream processing*. [online] Wikipedia. Available at: [https://en.wikipedia.org/wiki/Event\\_stream\\_processing](https://en.wikipedia.org/wiki/Event_stream_processing)

[5] Diao, Y. and J. Franklin, M. (n.d.). *PUBLISH/SUBSCRIBE OVER STREAMS*. [online] *polytechnique*, p.6. Available at: <http://www.lix.polytechnique.fr/~yanlei.diao/publications/StreamPubSub.pdf> [Accessed 3 Mar. 2021].

[6] Contributors to Wikimedia projects (2015). *Internet of things*. [online] Wikipedia.org. Available at: [https://bg.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82\\_%D0%BD%D0%B0\\_%D0%BD%D0%B5%D1%89%D0%B0%D1%82%D0%B0](https://bg.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82_%D0%BD%D0%B0_%D0%BD%D0%B5%D1%89%D0%B0%D1%82%D0%B0) [Accessed 3 Mar. 2021].

[7]chen (2019). *Algorithmic Trading Definition*. [online] Investopedia. Available at: <https://www.investopedia.com/terms/a/algorithmictrading.asp>.

[8]Wikipedia. (2020). *Home health care software*. [online] Available at: [https://en.wikipedia.org/wiki/Home\\_health\\_care\\_software](https://en.wikipedia.org/wiki/Home_health_care_software) [Accessed 3 April 2021].

[9]dictionary.cambridge.org. (n.d.). *PRODUCTION LINE | meaning in the Cambridge English Dictionary*. [online] Available at: <https://dictionary.cambridge.org/dictionary/english/production-line>.

[10]mitel. (n.d.). *WHAT IS FRAUD DETECTION?* [online] Available at: <https://www.mitel.com/features-benefits/fraud-detection>.

[11]Wikipedia Contributors (2019). *Surveillance*. [online] Wikipedia. Available at: <https://en.wikipedia.org/wiki/Surveillance>.

[12]Wikipedia Contributors (2019). *Real-time operating system*. [online] Wikipedia. Available at: [https://en.wikipedia.org/wiki/Real-time\\_operating\\_system](https://en.wikipedia.org/wiki/Real-time_operating_system).

[13]InfoQ. (n.d.). *Machine Learning Techniques for Predictive Maintenance*. [online] Available at: <https://www.infoq.com/articles/machine-learning-techniques-predictive-maintenance/> [Accessed 3 May 2021].

#### Zookeeper:

[14]zookeeper.apache.org. (n.d.). *ZooKeeper: Because Coordinating Distributed Systems is a Zoo*. [online] Available at: <https://zookeeper.apache.org/doc/current/zookeeperOver.html> [Accessed 3 April 2021].

[15]Dattell. (2019). *What is ZooKeeper & How Does it Support Kafka?* [online] Available at: <https://dattell.com/data-architecture-blog/what-is-zookeeper-how-does-it-support-kafka/> [Accessed 3 May 2021].

[16]Knowledge Base by phoenixNAP. (2020). *Install Apache ZooKeeper on Ubuntu {Standalone & Replicated}*. [online] Available at: <https://phoenixnap.com/kb/install-apache-zookeeper> [Accessed 3 May 2021].

#### FairMOT:

[17]Zhang, Y., Wang, C., Wang, X., Zeng, W. and Liu, W. (n.d.). *FairMOT : On the Fairness of Detection and Re-Identification in Multiple Object Tracking*. [online] . Available at: <https://arxiv.org/pdf/2004.01888.pdf> [Accessed 12 Jan. 2021].

#### Ubuntu:

[18]Ubuntu.com. (2019). *1.1. What is Ubuntu?* [online] Available at: <https://help.ubuntu.com/lts/installation-guide/s390x/ch01s01.html>.

#### Python:

[19]Wikipedia Contributors (2019). *Python (programming language)*. [online] Wikipedia. Available at: [https://en.wikipedia.org/wiki/Python\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Python_(programming_language)).

[20]Stross-Radschinski, A.C. (n.d.). *Get the Python Brochure Vol.1 as download!* —. [online] brochure.getpython.info. Available at: <https://brochure.getpython.info/> .

[21]Python (2019). *Welcome to Python.org*. [online] Python.org. Available at: <https://www.python.org/>.

### Kafka + Faust

[22]Amazon Web Services, Inc. (n.d.). *What is Apache Kafka? | AWS*. [online] Available at: <https://aws.amazon.com/msk/what-is-kafka/>.

[23]Confluent. (n.d.). *What is Apache Kafka? | Confluent*. [online] Available at: <https://www.confluent.io/what-is-apache-kafka/> [Accessed 3 May 2021].

[24]Apache Kafka. (n.d.). *Apache Kafka*. [online] Available at: <https://kafka.apache.org/documentation/streams/> [Accessed 3 May 2021].

[25]faust.readthedocs.io. (n.d.). *Kafka - The basics you need to know — Faust 1.9.0 documentation*. [online] Available at: <https://faust.readthedocs.io/en/latest/userguide/kafka.html> [Accessed 3 May 2021].

[26]Ask Solem (2018). Faust: Stream Processing for Python. *Medium*. [online] 31 Jul. Available at: <https://robinhood.engineering/faust-stream-processing-for-python-a66d3a51212d>.

[27]Stack Overflow. (n.d.). *“faust” tag wiki*. [online] Available at: <https://stackoverflow.com/tags/faust/info> [Accessed 3 May 2021].

[35]faust.readthedocs.io. (n.d.). *Faust - Python Stream Processing — Faust 1.9.0 documentation*. [online] Available at: <https://faust.readthedocs.io>

### Kitti Files

[28]GitHub. (n.d.). *NVIDIA/DIGITS*. [online] Available at: <https://github.com/NVIDIA/DIGITS/blob/v4.0.0-rc.3/digits/extensions/data/objectDetection/README.md> [Accessed 3 May 2021].

### Distance between two points in 2D and 3D space

[29]Three-Dimensional Coordinate Systems. (n.d.). [online] . Available at: <https://www.math.usm.edu/lambers/mat169/fall09/lecture17.pdf>.

## ANACONDA

[30]Wikipedia Contributors (2019). *Anaconda (Python distribution)*. [online] Wikipedia. Available at: [https://en.wikipedia.org/wiki/Anaconda\\_\(Python\\_distribution\)](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)).

## FFMPEG

[31]www.ffmpeg.org. (n.d.). *About FFmpeg*. [online] Available at: <https://www.ffmpeg.org/about.html>.

## PIP

[32]Python, R. (n.d.). *What Is Pip? A Guide for New Pythonistas – Real Python*. [online] realpython.com. Available at: <https://realpython.com/what-is-pip/>.

[33]www.cctv42.co.uk. (n.d.). *Understanding the range of a CCTV camera. A guide from CCTV42*. [online] Available at: <https://www.cctv42.co.uk/help-advice/faqs/can-you-explain-about-the-range-of-a-cctv-camera/>.

[34]Kundegorski, M. and Breckon, T. (2014). *A photogrammetric approach for real-time 3D localization and tracking of pedestrians in monocular infrared imagery*.