Benchmark

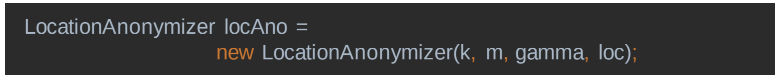
	APACHE STORM	APACHE Spark	Flink	APACHE nifi
General Properties				
Latency	Very Low	Medium	Low	Medium
Maximum Throughput (Event per Sec.)	Low(High?)	High	High	High
Fault Tolerance	At least once	Exactly once	Exactly once	Exactly once
Salability	High	High	High	normal
Project specified Properties				
State Management	Achieved by checkpoint	yes	yes	maybe
Dynamic deployment	yes(using Topology)	yes(by adjusting program)	maybe	yes
Jar package Compatible	Need test(depend on dependency)	Need test(depend on dependency)	yes	maybe
	1. real-time stream processing2. provides guaranteed data processing	1. Powerful framework 2. Good for batch processing	1. Powerful framework 2. Easy to use	1. Designed for dynamic env 2. Template

Evaluation

Attribute	Describe	Test
Flow Latency	How long does it take, for a single message to go through the Stream	Direct time calculation in Programming
Dynamic Latency	How long does it take to change the PET after the user specified to do so	Compare timestamp of request of change and timestamp of policy change
Redeployment Latency	How long does it take to attach a new PET to the system	Measure Time to reconfigure the whole system to the new setting
Maximum Throughput	How large amount of message can the system handle	Overflow the pipeline with huge among of messages
Fault tolerance	Validity, Integrity of the message and operation	Documentation and research
Deployment effort	How easy to implement all this	Objective

BA implementation

Location k-Anonymity

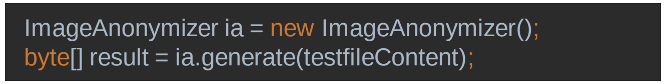


```java
LocationAnonymizer locAno =
                      new LocationAnonymizer(k, m, gamma, loc);
```

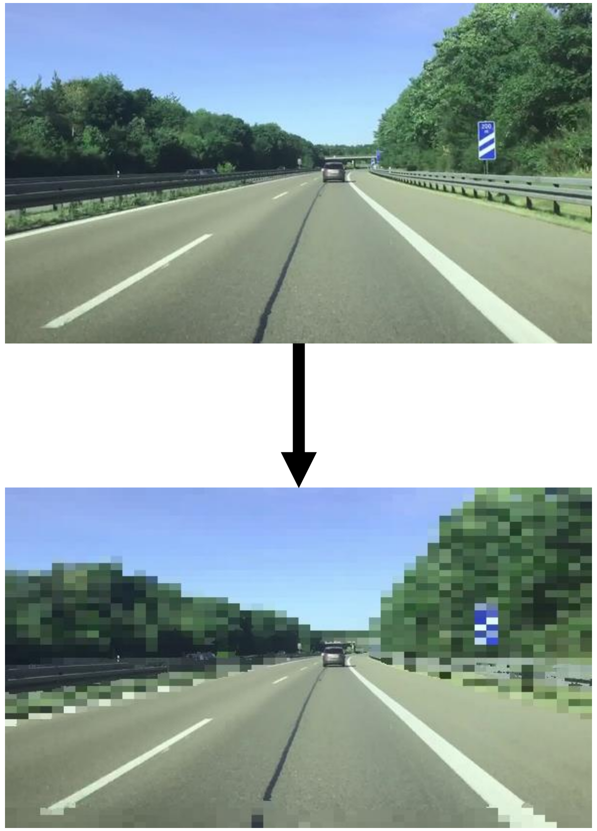
```
Input:
real location
Point2D.Double[25.0, 25.0]


Outcome:
[Point2D.Double[25.0,
25.0], Point2D.Double[21.0,
21.0], Point2D.Double[27.0,
27.0]]
Entropy:
0.031221052440928272
```

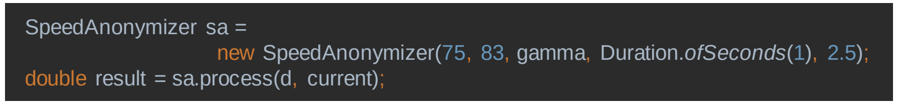
Camera Data Anonymization



```java
ImageAnonymizer ia = new ImageAnonymizer();
byte[] result = ia.generate(testfileContent);
```



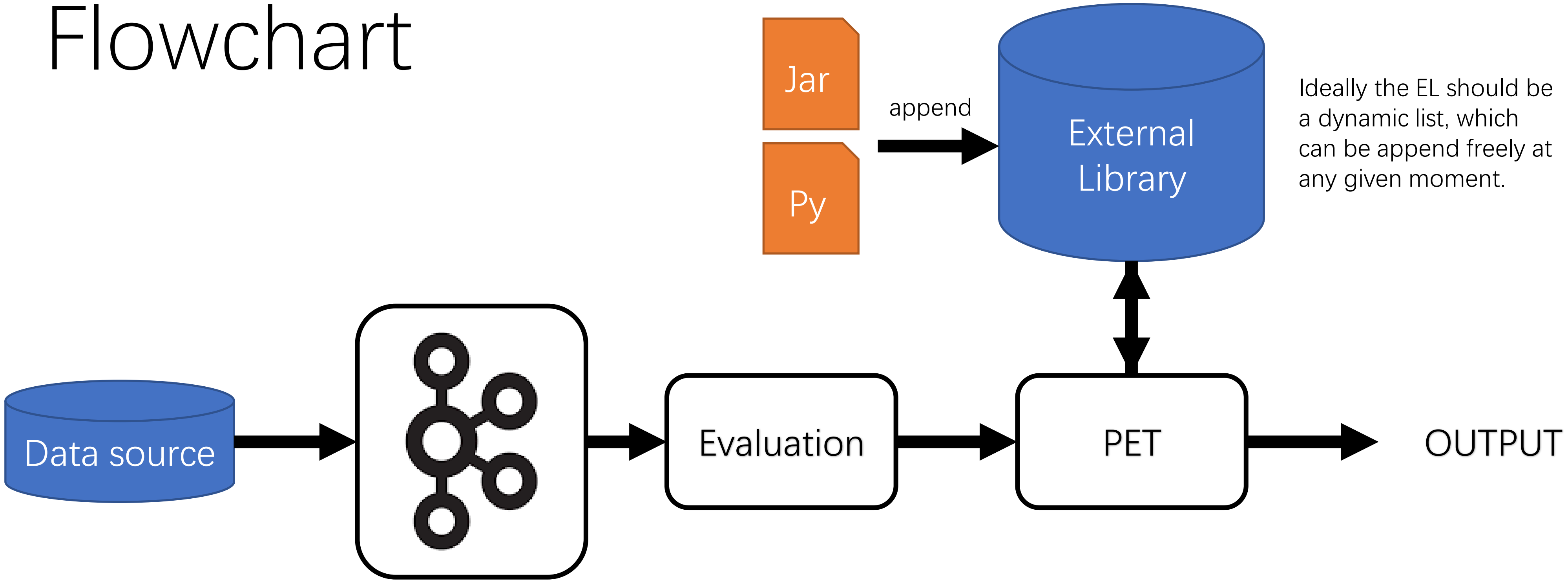
Speed Anonymization



```java
SpeedAnonymizer sa =
                      new SpeedAnonymizer(75, 83, gamma, Duration.ofSeconds(1), 2.5);
double result = sa.process(d, current);
```

Input	Outcome
75.0	75.0
78.0	77.36
81.0	79.84
85.0	81.77
87.0	82.43
89.0	82.79
90.0	82.89
91.0	82.79
91.0	82.73
92.0	82.97

Flowchart



Jar

Py

append

External Library

Data source

Evaluation

PET

OUTPUT

Ideally the EL should be a dynamic list, which can be append freely at any given moment.

Two types of data:

1. Driving data
2. User config

- Transform input to POJO
- Evaluate PL via Driving data
- Determine PET via User config (validity check)
- Output PET ID

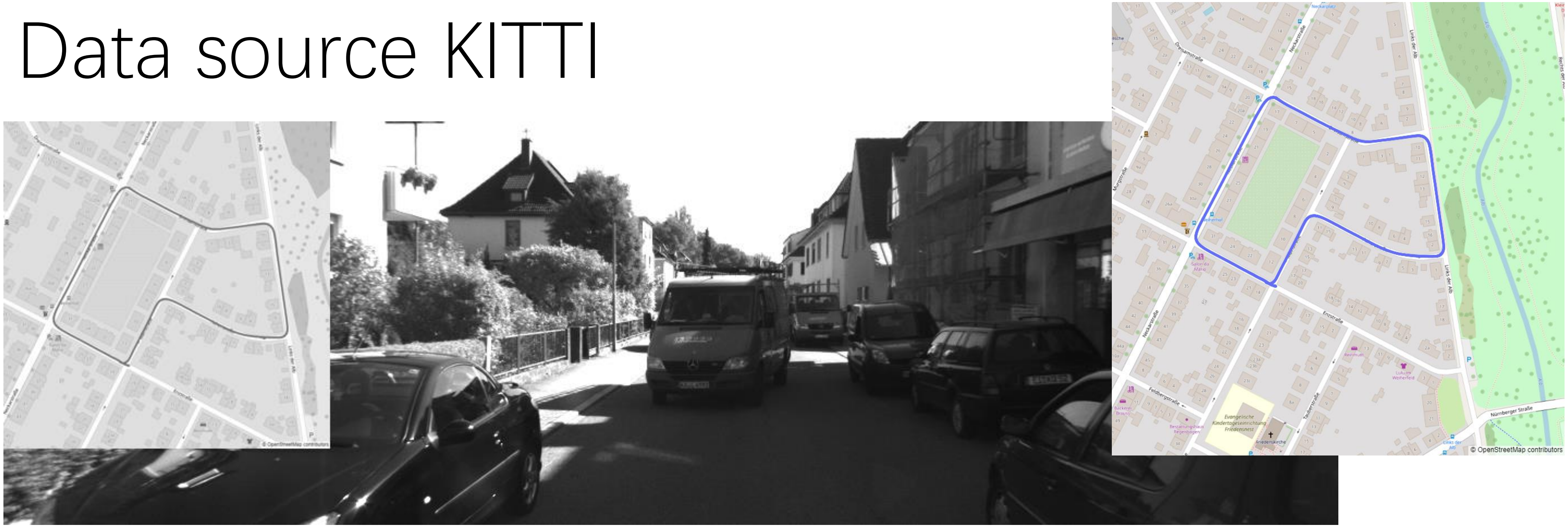
- Use PET ID to index the PET Object from External Library
- Instantiation and apply algorithm
- Out put modified data

*** If the technique don’t support dynamic adding of new package, we can think of something fix in size and enlarge if necessary.**

PET Indexing

- For non-dynamic Array:
  - Indexing like user privilege setting in linux.
  - Input: Integer number_PL, ArrayList<Boolean> activated situation
  - Output: ArrayList<Integer> List_activate_PET
  - Example: (23, [True, False, True]) -> ([3, 0, 1])
- For dynamic Array:
  - Indexing list, which PET for which situation.
  - Input: ArrayList<Integer> Indexing, ArrayList<Boolean> activated situation
  - Example: ([3, 2, 2], [True, False, True]) -> ([3, 0, 2])

Data source KITTI



- Sample Points: 1107
- Length: 1min51s
- Data: Camera, GPS, IMU

https://lukasyeah.ddns.net/index.php/s/mXPzxPS6SM2an3X

Problem

- Potential bottleneck:
  - Neural network for image segmentation.
  - Maintenance of location frequency list
- Location frequency is recorded in discrete fashion. Which in this case should be continuous.
- Python code for image segmentation, potential conflicting