VQPy: An Object-Oriented Approach to Modern Video Analytics

Shan Yu, Zhenting Zhu, Yu Chen, Hanchen Xu,
Pengzhan Zhao, Yang Wang, Arthi Padmanabhan, Hugo Latapie, Harry Xu

The Surge of Video Data

Surveillance Cameras

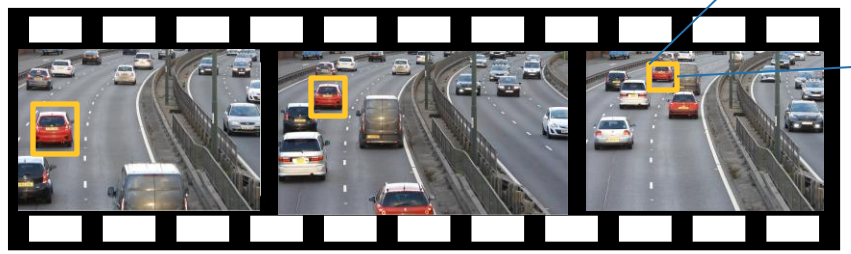
1 billion installed in the world in 2021

Autonomous Driving

A self-driving test vehicle generates **20 to 40 TB** of data per day.

Video Query

Amber alert: The license plate of a red Honda heading north.



Response: “FX66 UUW”

Intersection incident type: A turning right vehicle collides with a going straight vehicle.

Video queries center on video objects and their spatial/temporal relationships.

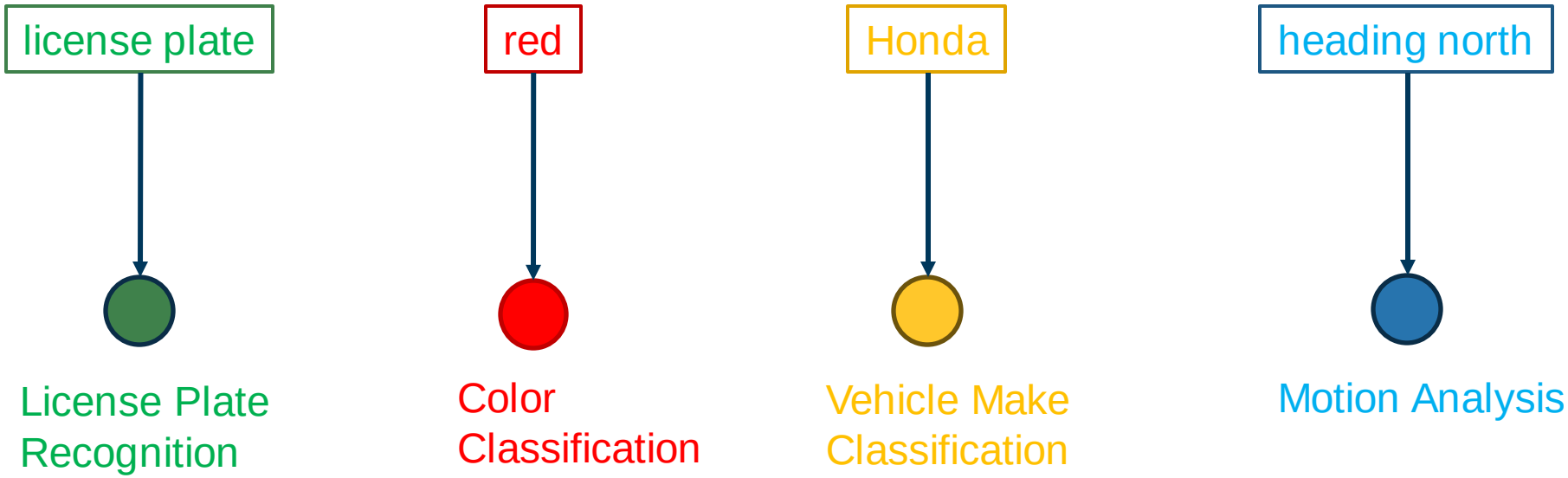
Response: “frame [i, i + k]”

Complexity of Video Queries: Multiple CV Tasks Required

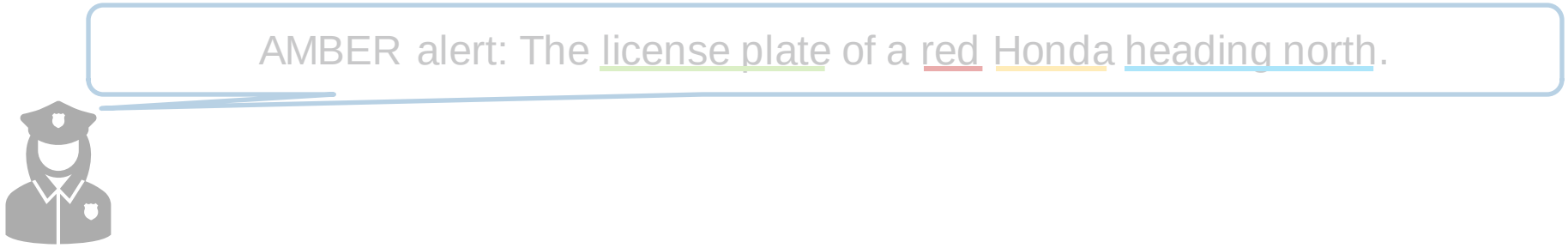
AMBER alert: The license plate of a red Honda heading north.

Complexity of Video Queries: Multiple CV Tasks Required

AMBER alert: The license plate of a red Honda heading north.



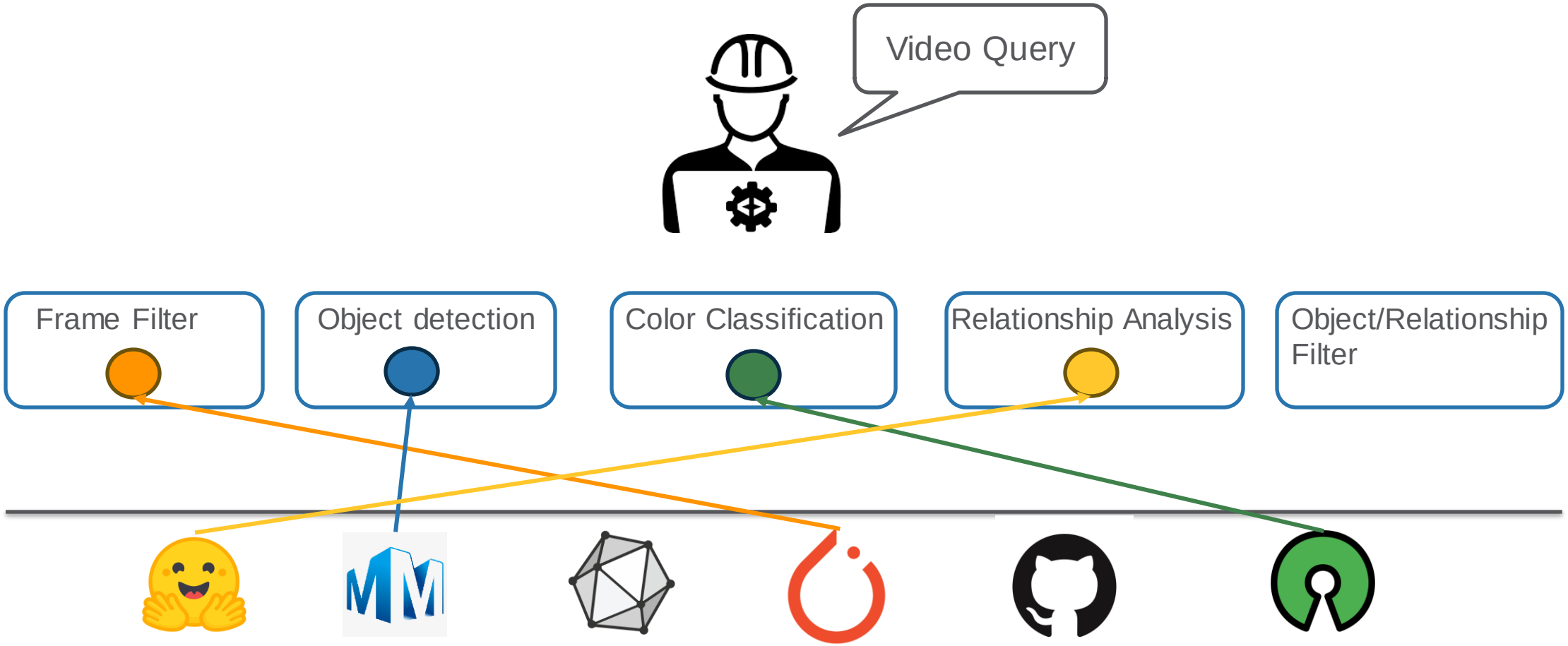
Complexity of Video Queries: Multiple CV Tasks Required



How to efficiently answer specific video queries?

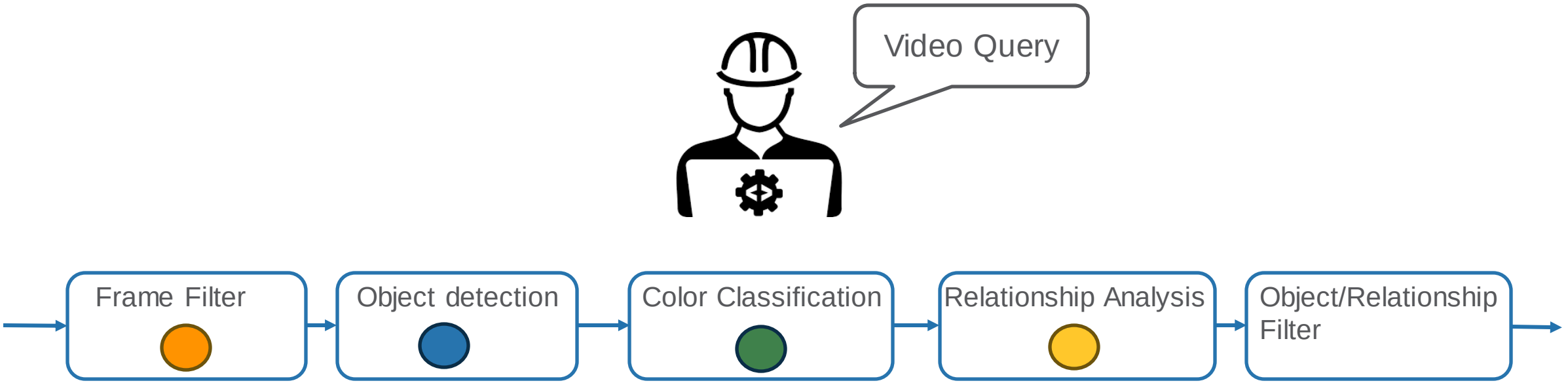
- License Plate Recognition
- Color Classification
- Car Make Classification
- Motion Analysis

Solution 1: Handcrafting Pipelines



Problem 1: Pipelines need to be manually constructed.

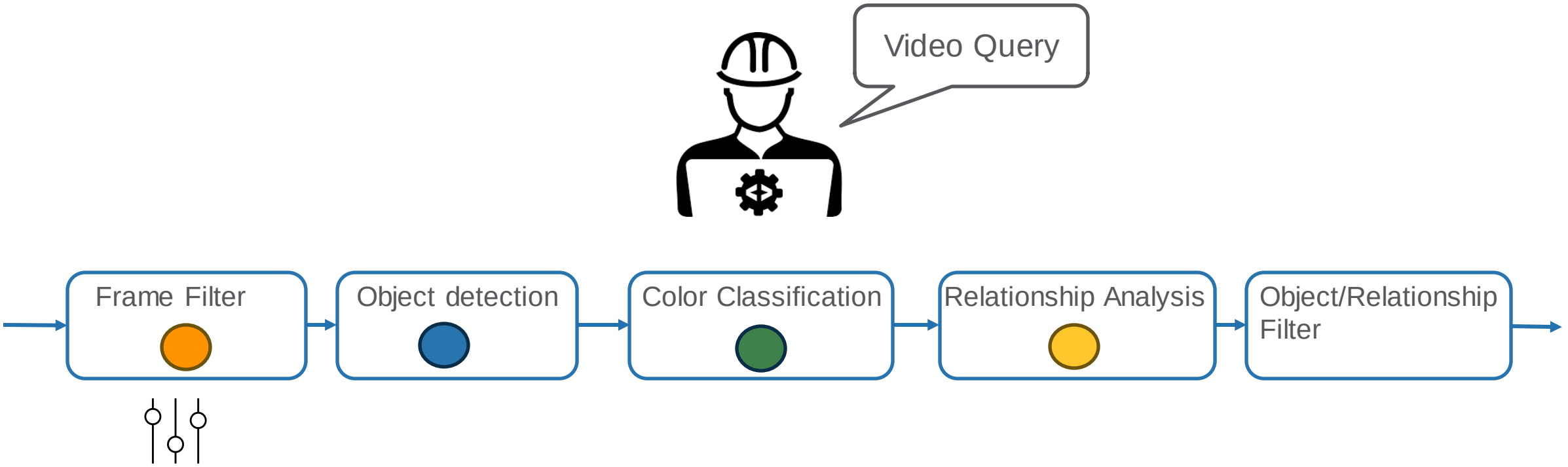
Solution 1: Handcrafting Pipelines



Writing glue code to connect the tasks together

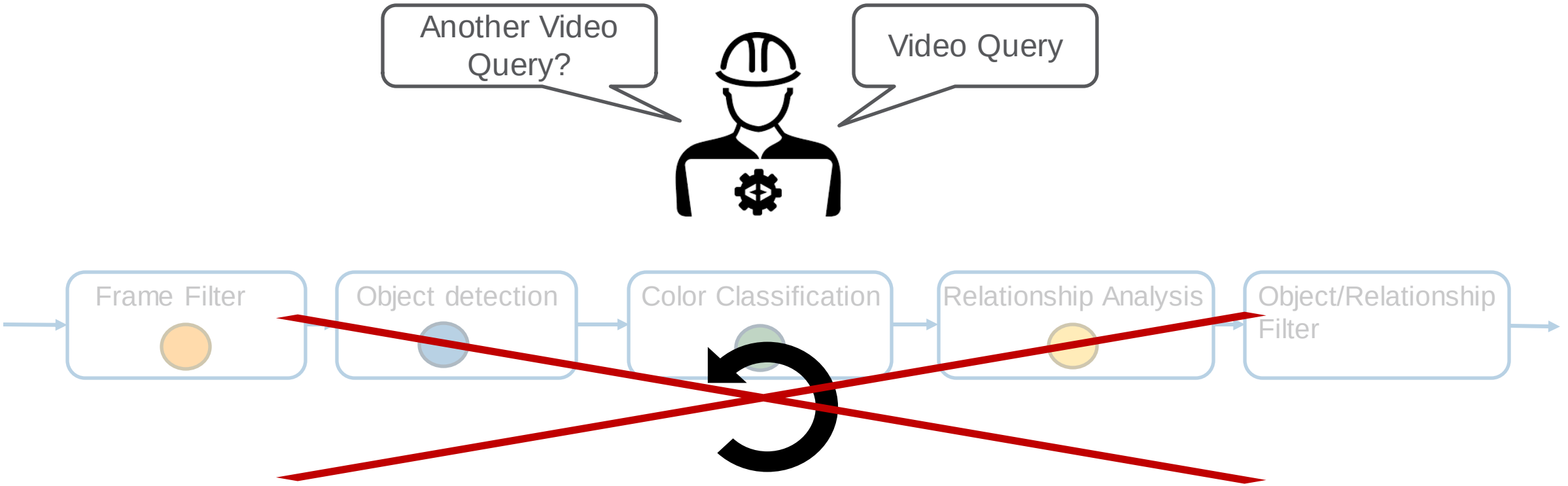
Problem 1: Pipelines need to be manually constructed.

Solution 1: Handcrafting Pipelines



Problem 2: Pipeline need to be manually optimized.

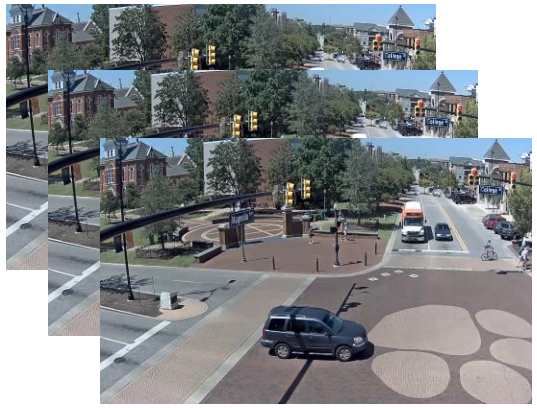
Solution 1: Handcrafting Pipelines



Problem 3: Each pipeline is subject to answer one query.

Solution 2: SQL-based Video Query Frameworks

A SQL-like query expression

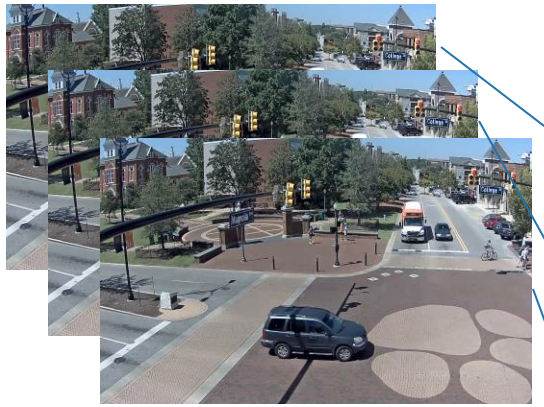


Solution 2: SQL-based Video Query Frameworks

Think like a table

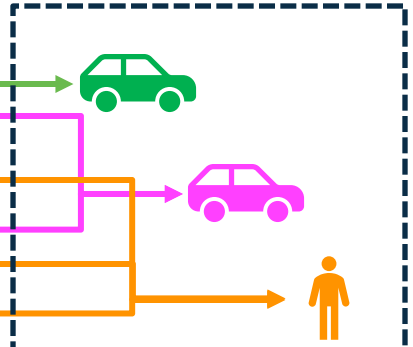
Lack of a video object abstraction

Tabular-based data model



frame_id	bboxes	labels	scores
0	[[1 2, 4, 5], [2, 3, 8, 9], …, [7, 6, 5, 3]]	[car, car, bus, person]	[0.5, 0.9, 0.8, 0.6]
1	[[4, 6, 7,8], [6, 7, 4, 5]]	[car, person]	[0.9. 0.8]
2	[[6, 5, 7, 6], …]	[person, …]	[0.9, …]

Video objects



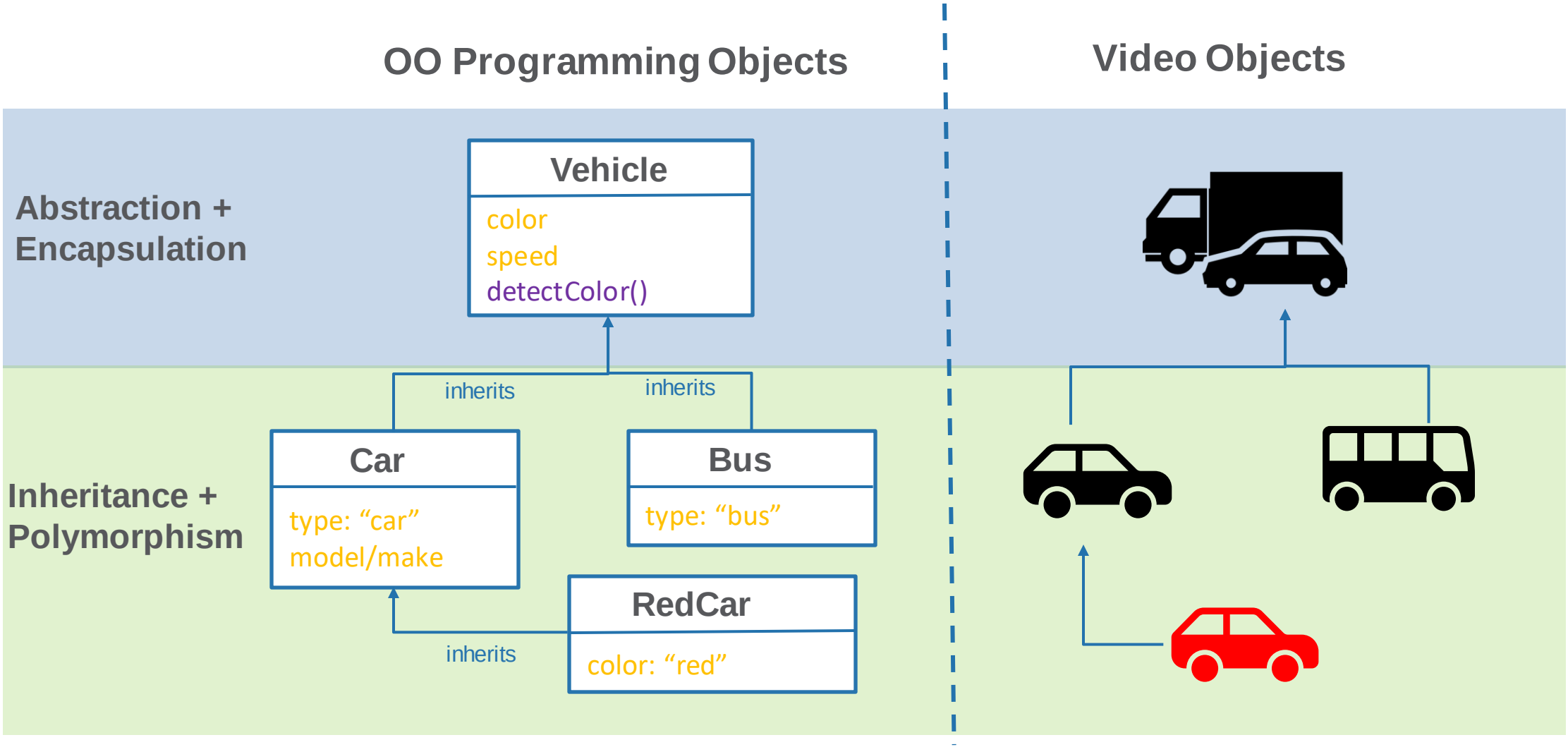
A SQL-like query expression

```
LOAD VIDEO 'video.mp4'
INTO MyVideo;

CREATE FUNCTION Add1
IMPL './add1.py';

CREATE FUNCTION Velocity
IMPL './velocity.py';

CREATE FUNCTION Color
IMPL './color.py';

CREATE TABLE
    TrackResult
AS
SELECT
        id, Color(Crop(data, bbox)), T.iid,
            T.bbox, T.score, T.label
FROM MyVideo
    JOIN LATERAL
        UNNEST(EXTRACT_OBJECT(
                data, Yolo, NorFairTracker))
        T(iid, label, bbox, score);

CREATE TABLE
    TrackResultAdd1
AS
SELECT
        Add1(id, iid, bbox)
    FROM
TrackResult

CREATE TABLE
    TrackResultJoin
AS
SELECT
        trackresult.id, trackresult.iid,
        trackresult.color, trackresult.bbox,
        trackresult.label, trackresult.score,
        trackresultadd1.last_bbox
    FROM
        TrackResult JOIN TrackResultAdd1
    ON
        trackresult.id = trackresultadd1.added_id
    AND
        trackresult.iid = trackresultadd1.cur_iid;

SELECT
    id, iid, bbox
FROM
    TrackResultJoin
WHERE
    Velocity(bbox, last_bbox) > 1
AND
    color = 'red' AND label = 'car';

DROP TABLE IF EXISTS MyVideo;
DROP TABLE IF EXISTS TrackResult;
DROP TABLE IF EXISTS TrackResultAdd1;
```

A red speeding car query

Can we have a video object abstraction for video queries?

Insight: Video Objects Are Similar to Objects in OO Programming



VQPy – An Object-Oriented Approach to Video Analytics

Frontend: A Python library for video-object-oriented programming

Express a Red Speeding Car Query

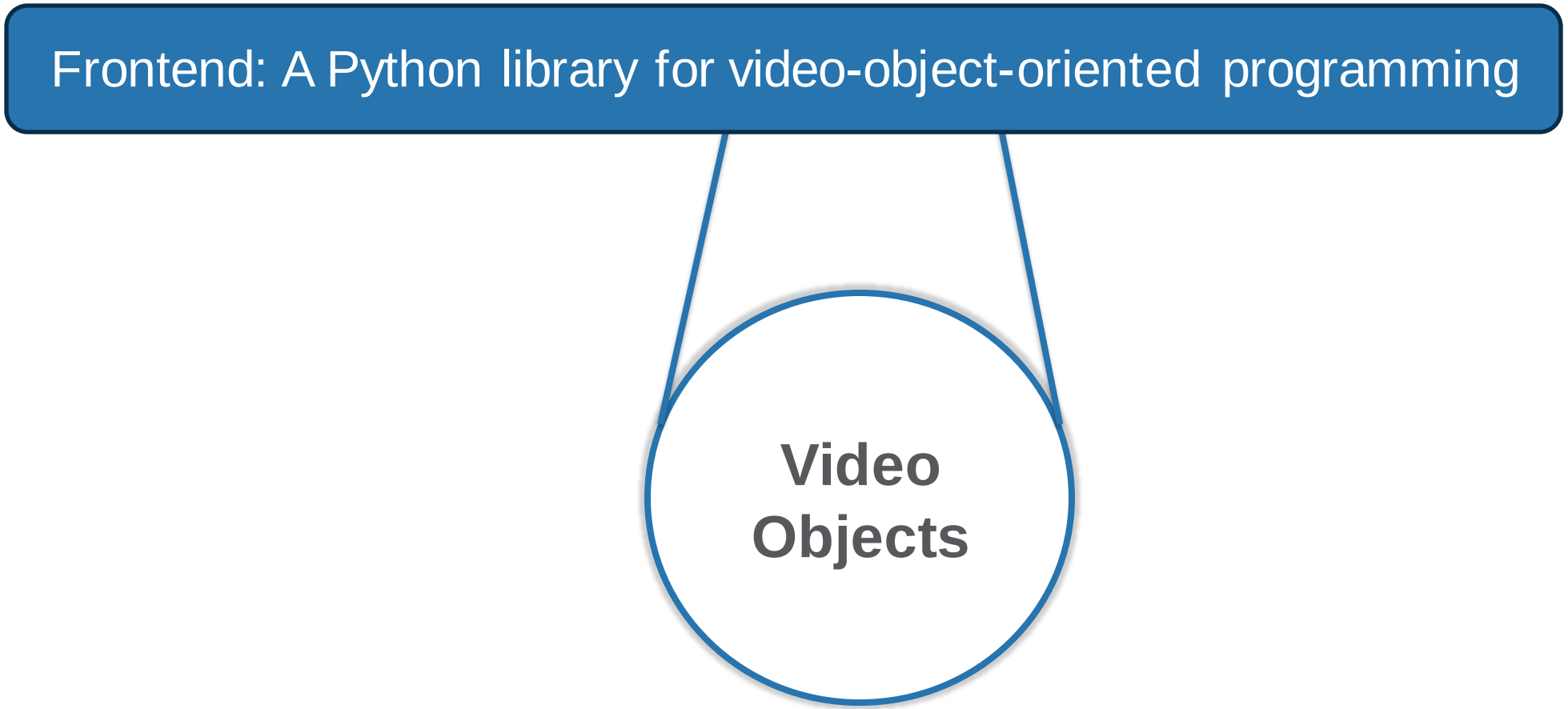
With a SQL-like language

```
LOAD VIDEO 'video.mp4'
INTO MyVideo;

CREATE FUNCTION Add1
IMPL './add1.py';

CREATE FUNCTION Velocity
IMPL './velocity.py';

CREATE FUNCTION Color
IMPL './color.py';

CREATE TABLE
    TrackResult
AS
SELECT
        id, Color(Crop(data, bbox)), T.iid,
            T.bbox, T.score, T.label
FROM MyVideo
    JOIN LATERAL
        UNNEST(EXTRACT_OBJECT(
                data, Yolo, NorFairTracker))
        T(iid, label, bbox, score);

CREATE TABLE
    TrackResultAdd1
AS
SELECT
        Add1(id, iid, bbox)
    FROM
TrackResult

CREATE TABLE
    TrackResultJoin
AS
SELECT
        trackresult.id, trackresult.iid,
        trackresult.color, trackresult.bbox,
        trackresult.label, trackresult.score,
        trackresultadd1.last_bbox
    FROM
        TrackResult JOIN TrackResultAdd1
    ON
        trackresult.id = trackresultadd1.added_id
    AND
        trackresult.iid = trackresultadd1.cur_iid;

SELECT
    id, iid, bbox
FROM
    TrackResultJoin
WHERE
    Velocity(bbox, last_bbox) > 1
AND
    color = 'red' AND label = 'car';

DROP TABLE IF EXISTS MyVideo;
DROP TABLE IF EXISTS TrackResult;
DROP TABLE IF EXISTS TrackResultAdd1;
```

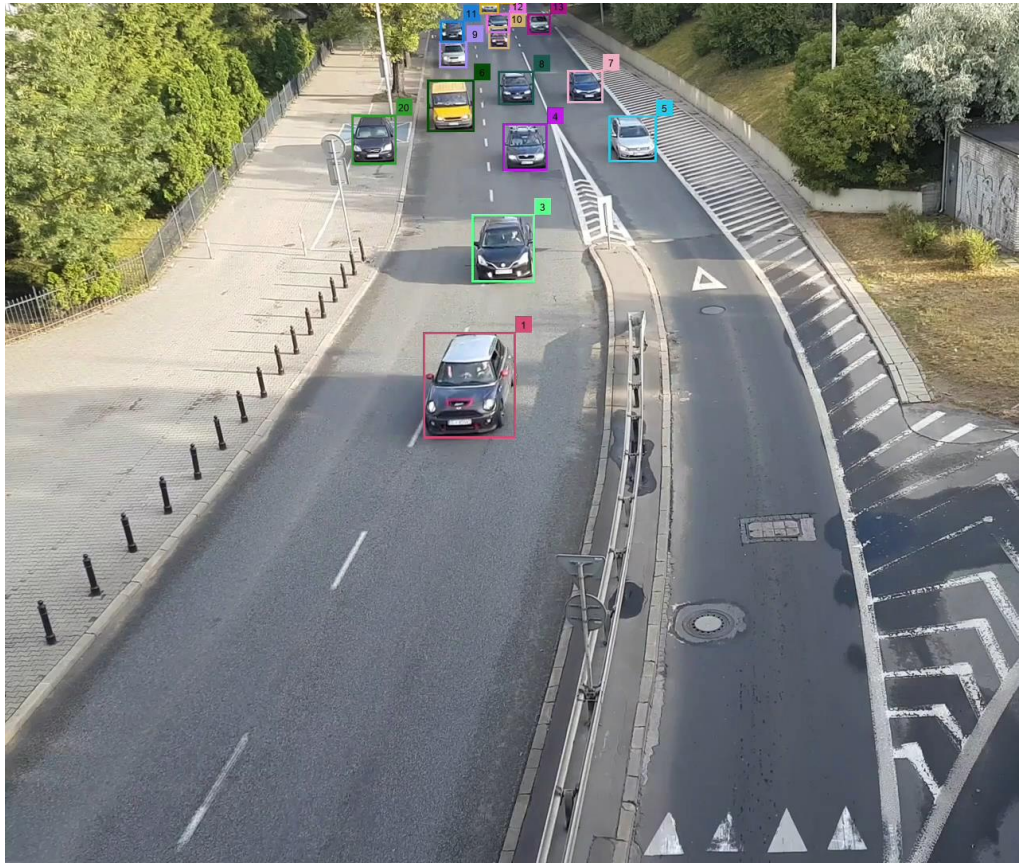
With VQPy

```
vobj Car(vqpy.Vehicle):
    def __init__(self):
        self.model = "yolov8m"
    # inherit color and velocity properties from Vehicle

query RedSpeedCar(vqpy.Query):
    def __init__(self):
        self.car = Car()

    def frame_constraint(self):
        return self.car.color == "red" and self.car.velocity > 100

    query frame_output(self):
        return self.car.id
```

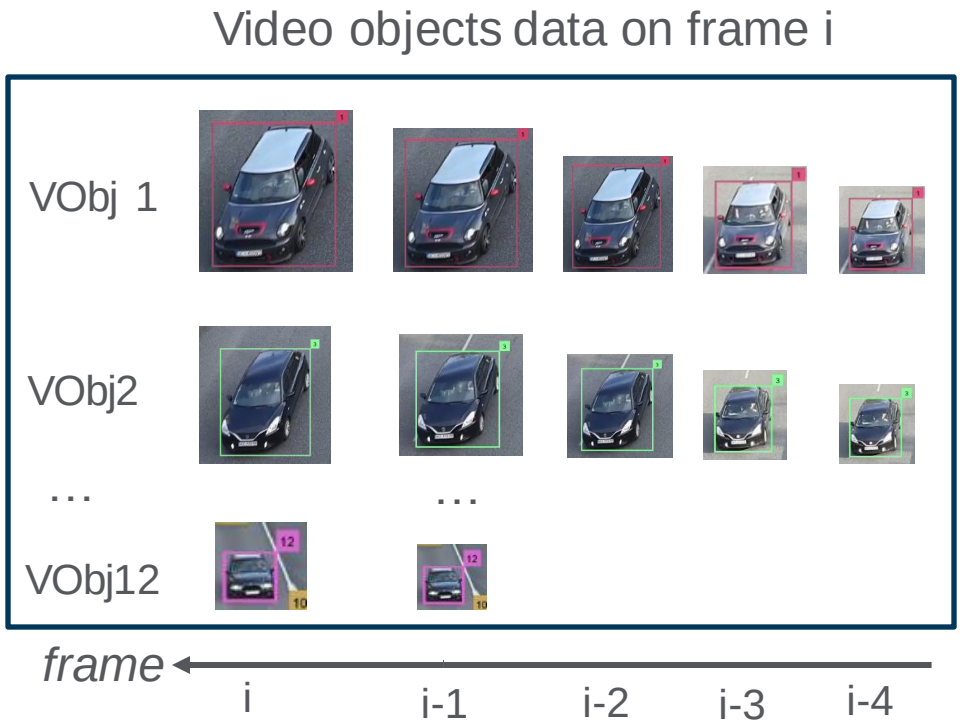
VQPy – An Object-Oriented Approach to Video Analytics



Video Object Abstraction in VQPy



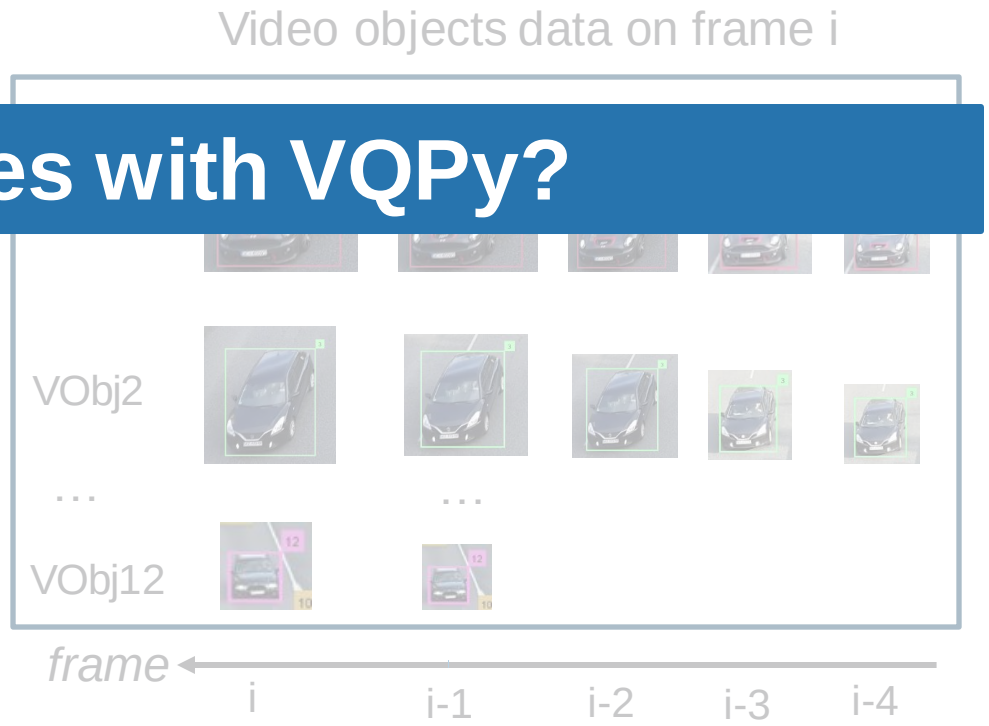
A video object represents a unique entity that may appear across multiple frames.



https://www.kaggle.com/datasets/trainingdatapro/cars-video-object-tracking

Video Object Abstraction in VQPy

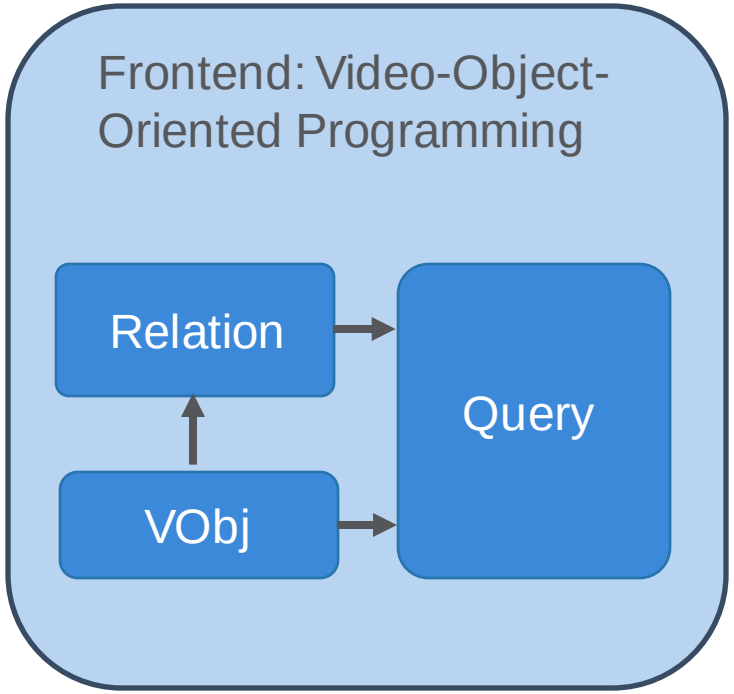
A video object represents a unique entity that may appear across multiple frames.

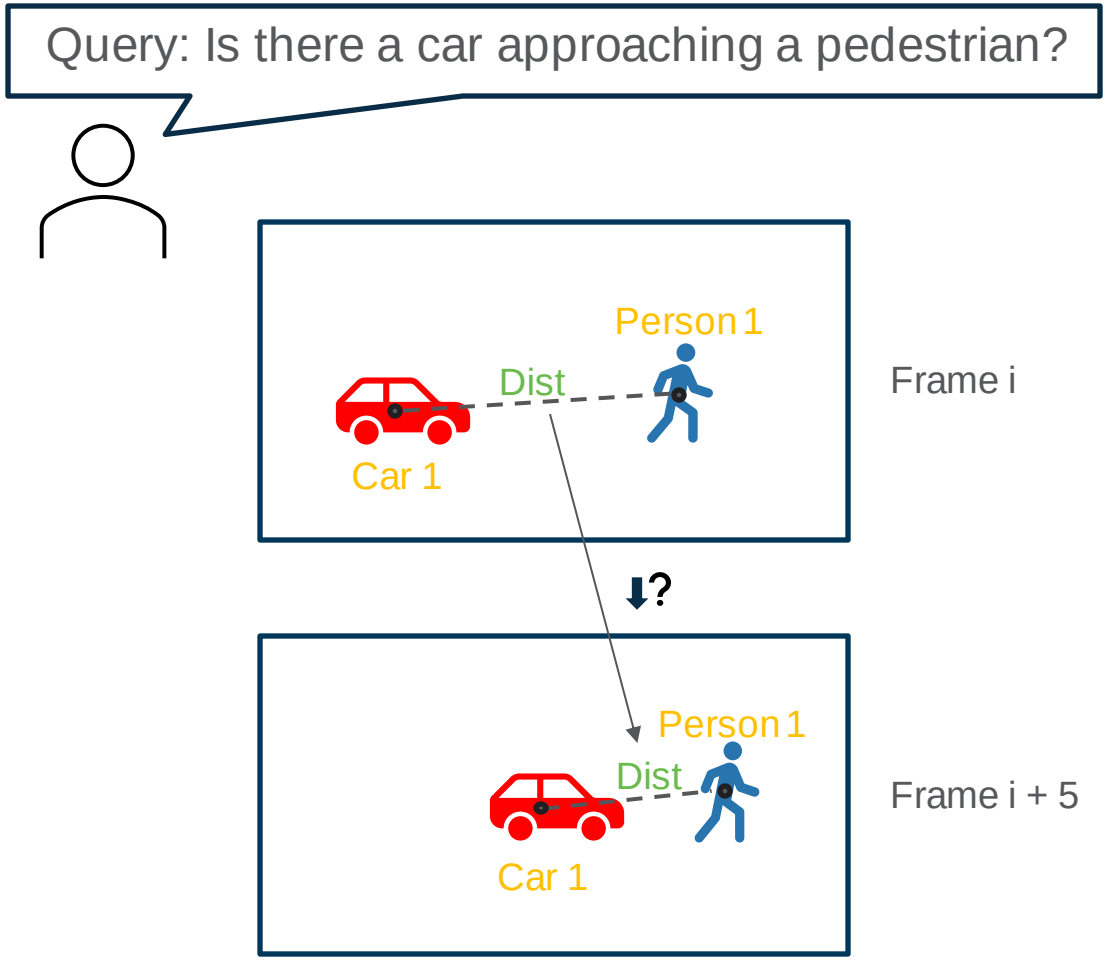


How to program video queries with VQPy?

https://www.kaggle.com/datasets/trainingdatapro/cars-video-object-tracking

Frontend Overview





Query on Video Objects

Retrieve the license plates of vehicles traveling at speeds exceeding 80 mph.

Frame 1

{'Vehicle': [{'license_plate': 'FX66 UUW'}]}

Frame 2

{'Vehicle': [{'license_plate': 'FX66 UUW'},
{'license_plate': 'HX14 UHH'}]}

Frame 3

{'Vehicle': [{'license_plate': 'FX66 UUW'},
{'license_plate': 'HX14 UHH'}]}

```
vobj Vehicle:

  def __init__(self):
    self.model = “yolox”

  @stateful(input="center", history_len=2)
  def speed(self, hist_centers):
    ...

  @stateless(input=”image")
  def license_plate(self, vehicle_image):
    ...
```

Construct a Vehicle VObj

```
query SpeedTicketing:

  def __init__(self):
    self.vehicle = Vehicle()

  def frame_constraint(self):
    return self.vehicle.speed > 80

  def frame_output(self):
    return self.vehicle.license_plate
```

Construct a SpeedTicketing Query

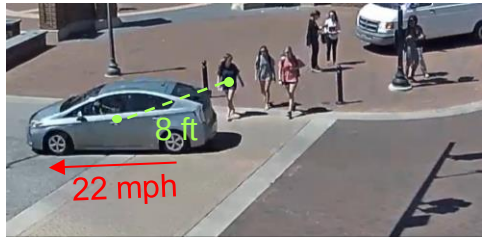
Query on VObjs and Relations

Retrieve the event that a car speeds past a person.

{'frame_id': [5,6]}

Frame 5

Frame 6



Frame 7

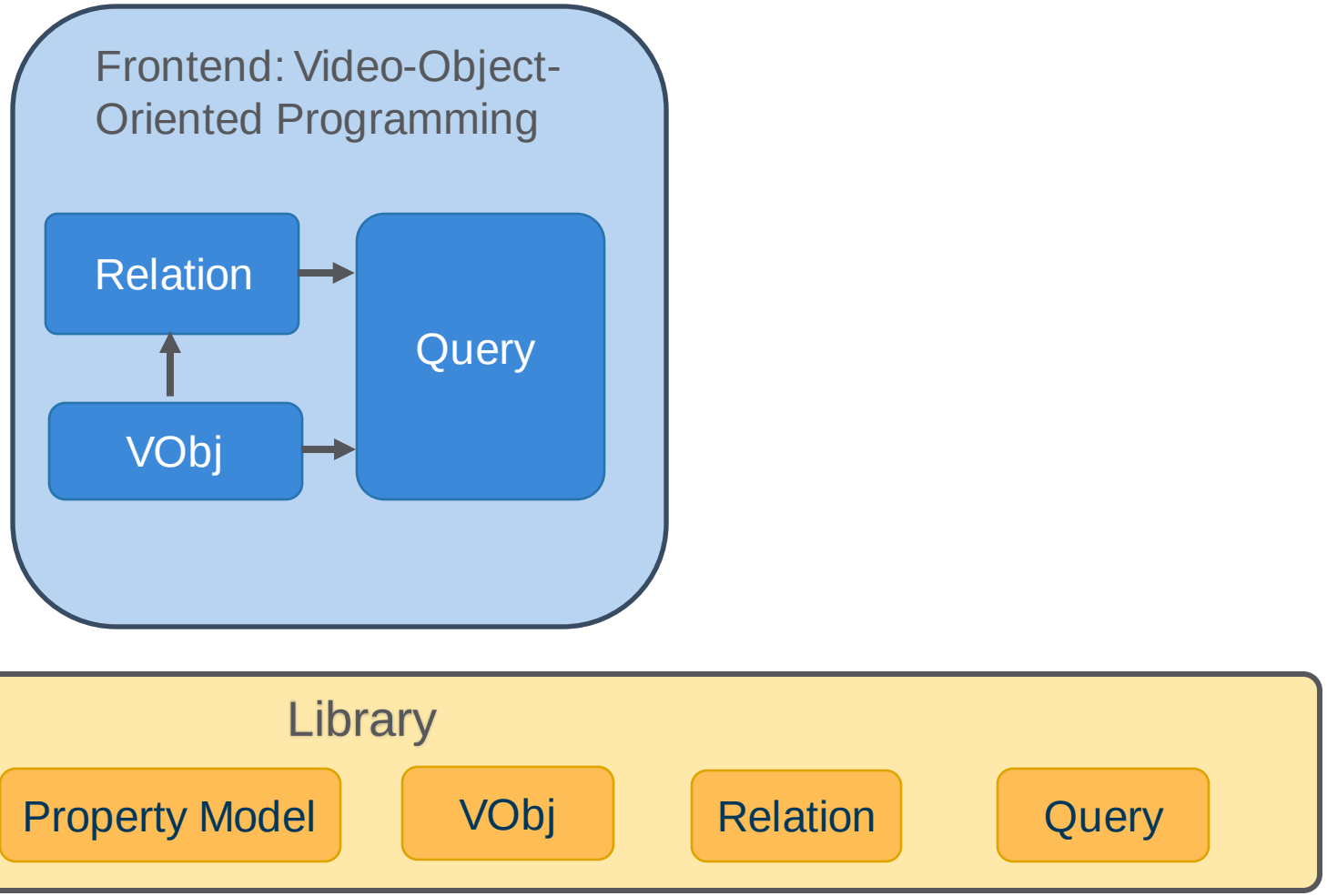
```
relation SpatialRelation(vqpy.Relation):

  def __init__(self, vobj1, vobj2):
    self.vobj1 = vobj1
    self.vobj2 = vobj2

  @stateless(input1="center", input2="center")
  def distance(self, centers):
    return math.dist(centers[0], centers[1])
```

VQPy Spatial Relation

```
query TrafficHazards(vqpy.Query):

  def __init__(self):
    self.car = Car()
    self.person = Person()
    self.relation = \
      SpatialRelation(self.car, self.person)

  def frame_constraint():
    return (self.relation.distance < 6)
                & (self.car.speed > 15)
```

A car speeds past a person Query

Frontend Summary



VQPy's Backend

Backend: An Object-Based Optimization Framework

- How to choose the right data model?
- How to automatically construct a pipeline for frontend queries?
- How to optimize the pipeline based on video objects?
- How to build the backend to be easily extensible with custom optimizations?

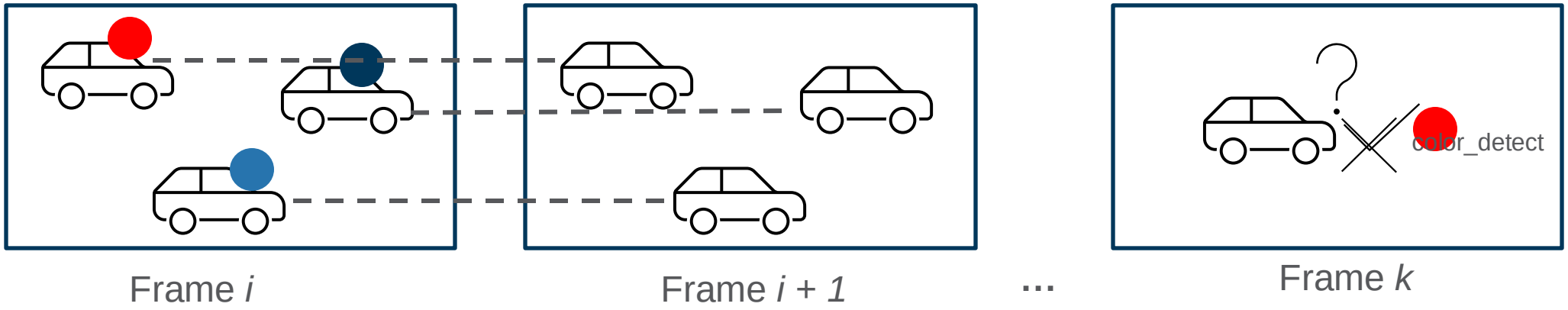
VQPy's Backend

Backend: An Object-Based Optimization Framework

- How to choose the right data model?
- How to automatically construct a pipeline for frontend queries?
- How to optimize the pipeline based on video objects?
- How to build the backend to be easily extensible with custom optimizations?

Object-level Computation Reuse

```
vobj Vehicle(vqpy.VObj):

 @stateless(model="color_detect", intrinsic=True)
 def color(self, images):
  # built-in color_detect model
  ...
```



Evaluation

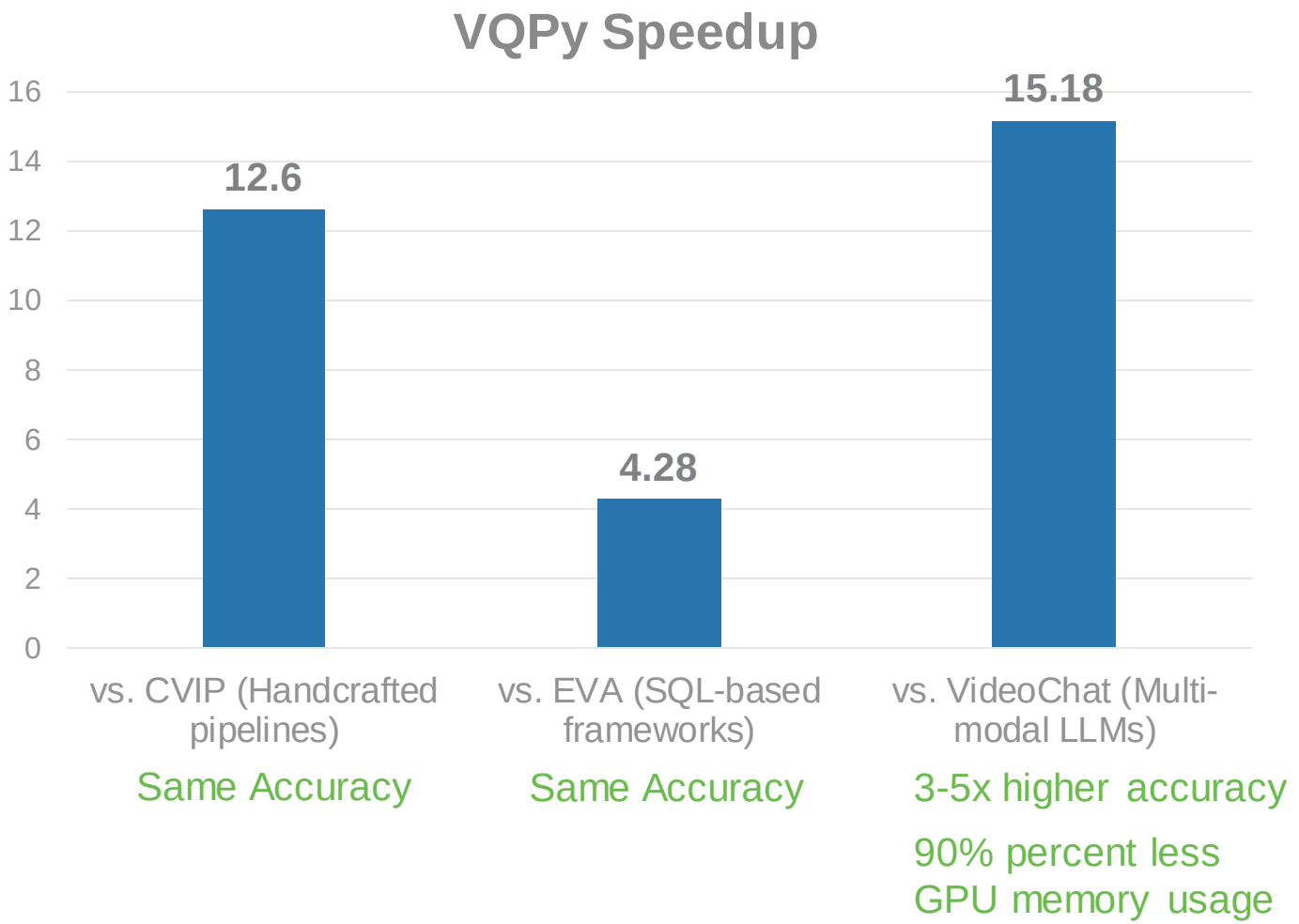
Evaluated with 14 video queries on 5 datasets from real-world surveillance video streams.

- Handcrafted pipelines
  - Complex vehicle retrieval queries
  - Baseline: CVIP[1]
- SQL-based frameworks
  - Video-object-based queries
  - Baseline: EVA [2]
- Multi-modal LLMs
  - Diverse types of video queries
  - Baseline: VideoChat [3]

[1] Le, H. D.-A., Nguyen, Q. Q.-V., Luu, D. T., Chau, T. T.-T., Chung, N. M., and Ha, S. V.-U. Tracked-vehicle retrieval by natural language descriptions with multi-contextual adaptive knowledge. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pp. 5510–5518, June 2023.
[2] Xu, Z., Kakkar, G. T., Arulraj, J., and Ramachandran, U. Eva: A symbolic approach to accelerating exploratory video analytics with materialized views. In Proceedings of the 2022 International Conference on Management of Data, SIGMOD '22 URL https://doi.org/ 10.1145/3514221.3526142.
[3] Li, K., He, Y., Wang, Y., Li, Y., Wang, W., Luo, P., Wang, Y., Wang, L., and Qiao, Y. Videochat: Chat- centric video understanding. arXiv preprint arXiv:2305.06355, 2023.

Evaluation

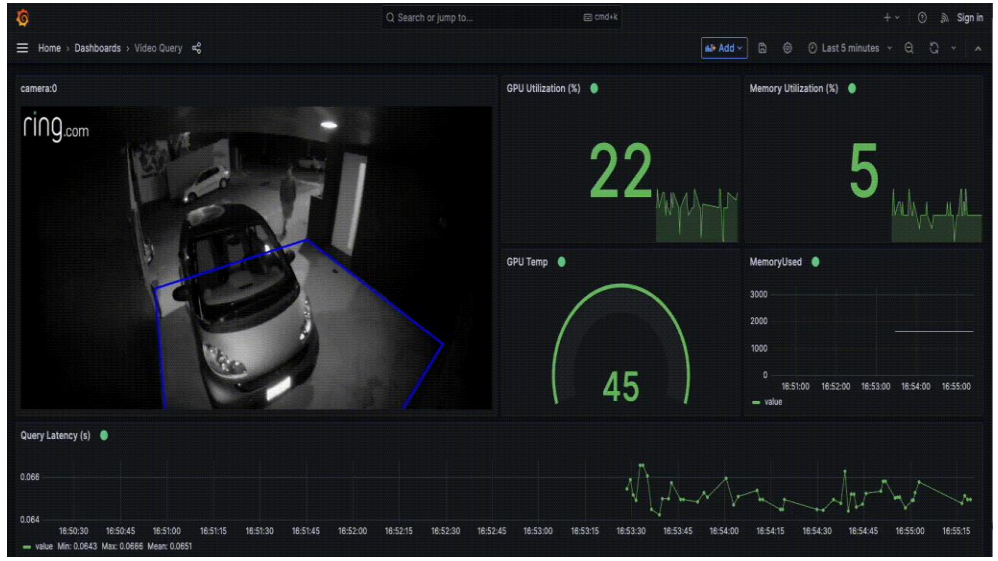
Evaluated with 14 video queries on 5 datasets from real-world surveillance video streams.



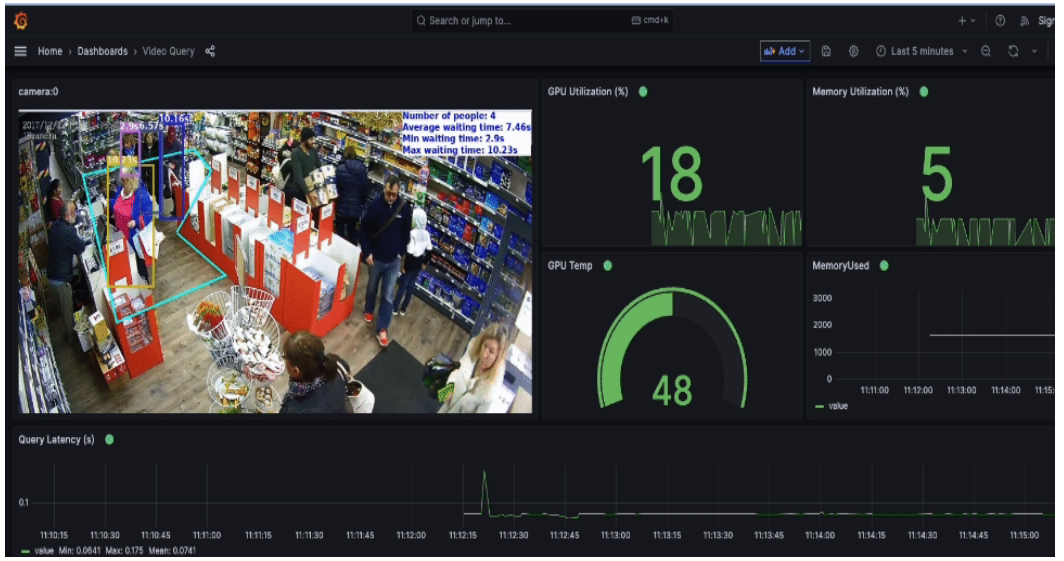
Industrial Adoption

VQPy has been integrated into Cisco as a query development/execution layer in its DeepVision framework.

Loitering Alert



Queue Analysis



Conclusion

VQPy: a video-object-oriented approach towards video analytics

- Expressiveness: Easily express video queries on video objects and their interactions, with an object-oriented programming frontend.
- Efficiency: Streamlines the efficient execution of video-object-oriented queries, with an object-based optimization backend.

https://github.com/vqpy/vqpy

Thank you