# BYTE-TRACK WITH HUNGARIAN MATCHING ALGORITHM AND ENHANCED KALMAN FILTER

GOKTUG CAN SIMAY

ALI DOGAN

EMRE EMIN OSMAN

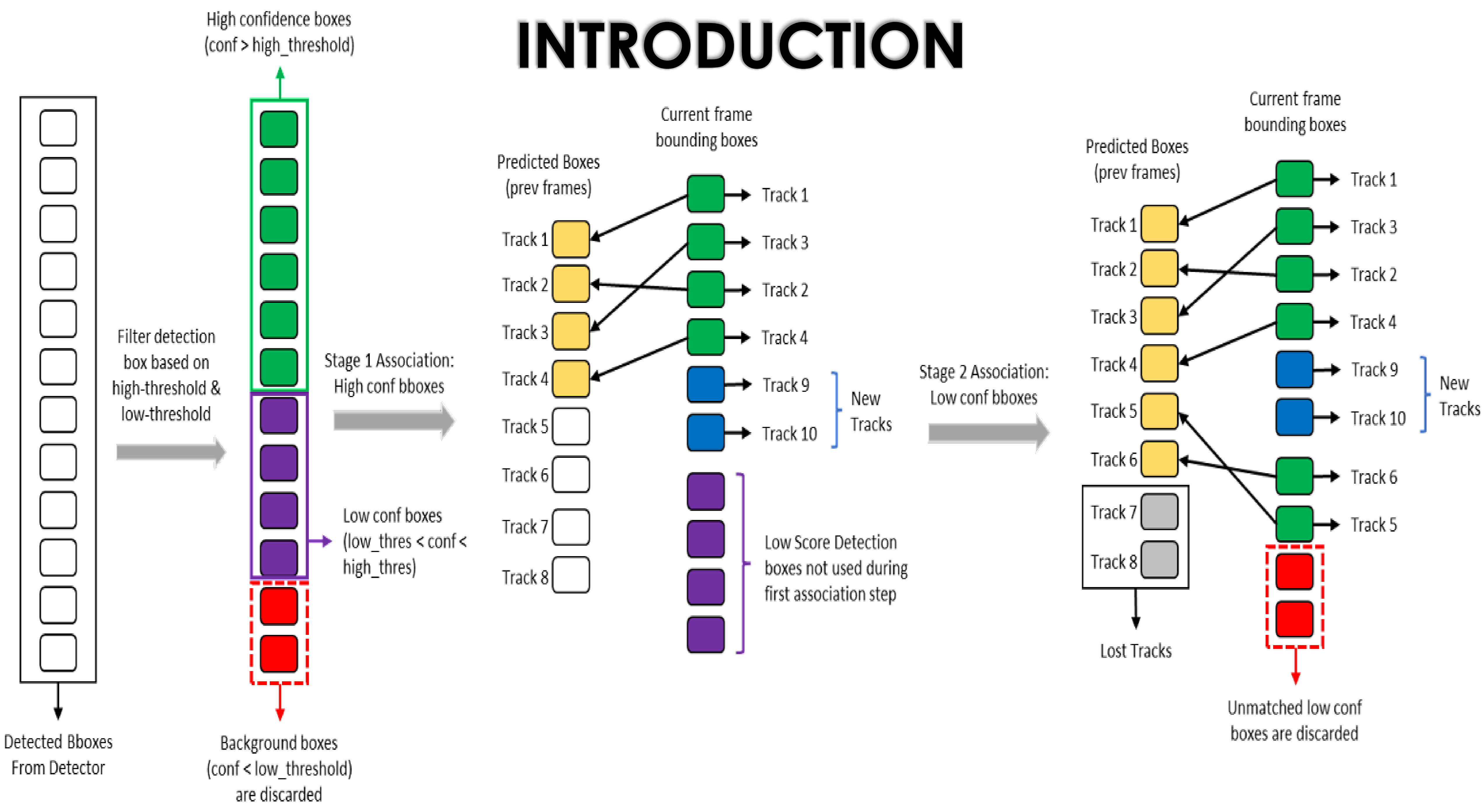
colab: https://colab.re-search.google.com/drive/1UYf_YsqSb9bBBeWsVuMT9Wj9vLTUIORH

: https://github.com/simaygoktug/MOT

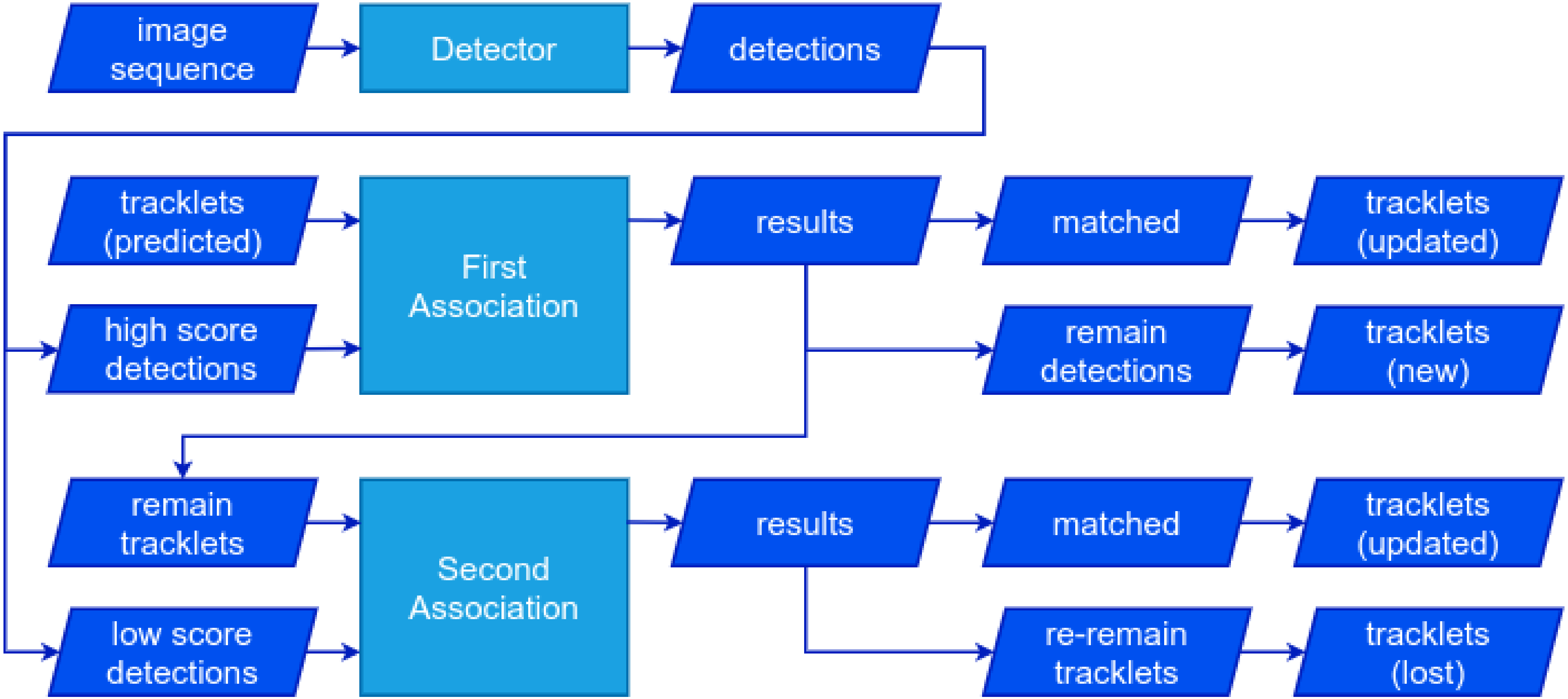
# CONTENT

- INTRODUCTION
- RELATED PROJECTS
- DATA
- APPROACH (METHODS AND EXPERIMENTS)
- RESULTS AND CONCLUSION
- REFERENCES

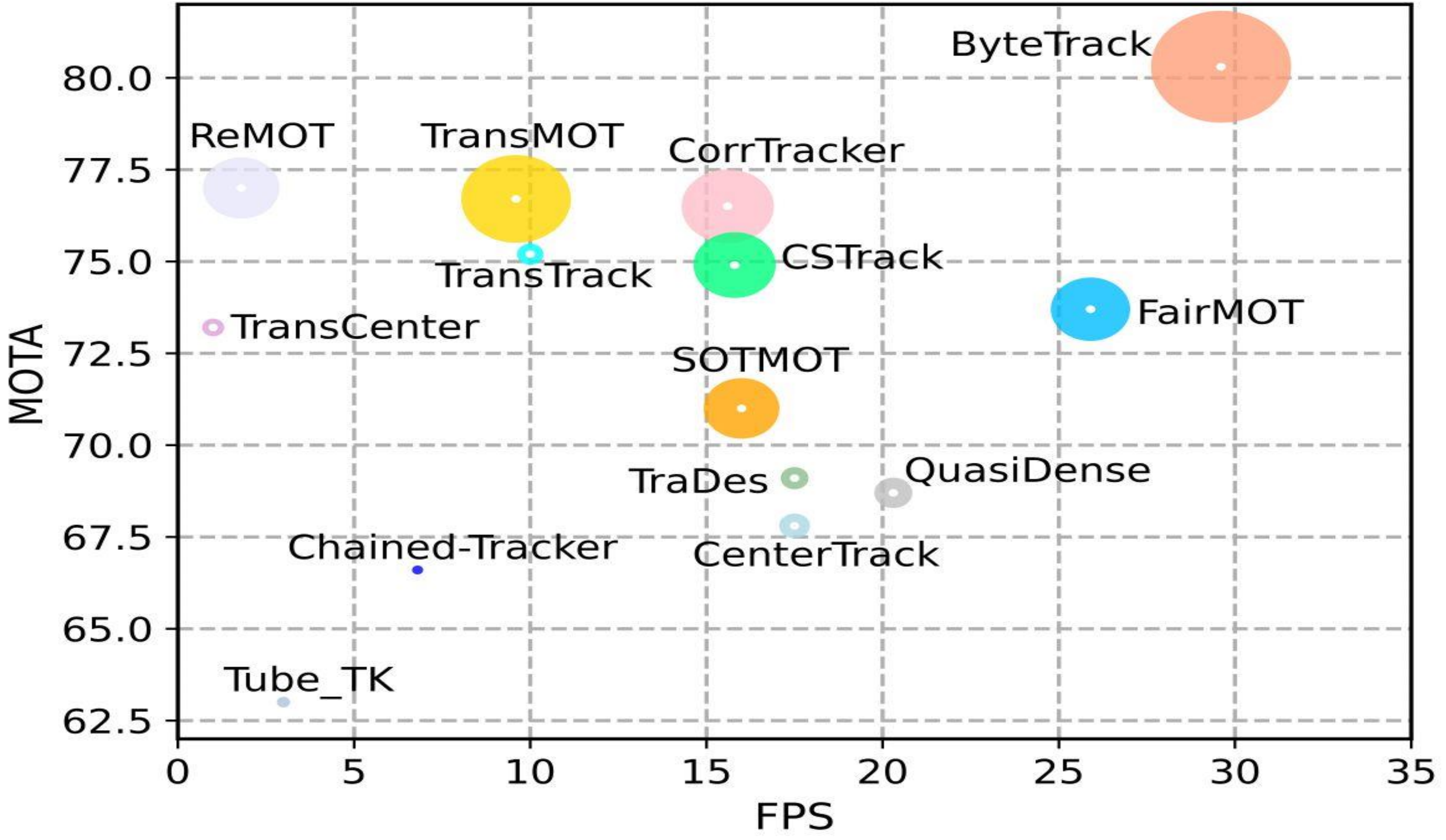
# INTRODUCTION



ByteTrack: Multi-Object Tracking by Associating Every Detection Box



# RELATED PROJECTS



# DATA

kaggle

# METHODS

**Particle Filter:** Particle filters are a well-established method for tracking objects in video. They are typically more accurate than Kalman filters however the performance gains would not be worth the additional computational cost in this situation.

**Ensemble Random Forest Filter:** ERF filters [5] are a type of Bayesian filter that can be used for tracking objects in video. They are typically more accurate than Kalman filters, but they can also be slower when the background is overly complex.

**Optimizing Kalman Filter:** In ByteTrack's current Kalman filter, in the predict() and update() methods, the _motion_mat and _update_mat matrices are recalculated each time. This causes unnecessary calculations. Optimization could be done by calculating and storing these matrices once. Calculation of the covariance matrix in predict() and update() methods; It is performed using the np.linalg.multi_dot() function. This function is designed to efficiently calculate matrix multiplications. However, this function may still involve some unnecessary calculations. The np.linalg.block_diag() function could be used to optimize these calculations.

The Gaussian Kalman Filter is a popular filter for linear systems. However, for nonlinear systems it has to perform linearization. This process may introduce linearization error and reduce prediction accuracy.

Unscented Kalman Filter does not perform linearization for nonlinear systems. Instead, it uses a set of points that accurately represent the behavior of nonlinear systems. These spots are created using a process called unscented conversion. The unscented transformation distributes random points in the state space, and these points consider the dynamics and measurements of the system.

The matching algorithms used by ByteTrack are responsible for associating object detections from different frames. The current implementation of ByteTrack uses the Lucas-Kanade algorithm. We considered using the Hungarian algorithm as an alternative. We also considered using the Delaunay triangulation algorithm. It is a geometric algorithm that can be used to find the best possible matches between object detections.

# METHODS

mot / yolox / tracker /

Add file

simaygoktug Update byte_tracker.py

aadb8e8 · 20 hours ago History

| Name | Last commit message | Last commit date |
|---|---|---|
| .. | | |
| basetrack.py | Update basetrack.py | last year |
| byte_tracker.py | Update byte_tracker.py | 20 hours ago |
| byte_tracker_improved_hungarian.py | Add files via upload | 2 weeks ago |
| ensemble_random_forest_filter.py | Add files via upload | 2 weeks ago |
| kalman_filter.py | Update kalman_filter.py | 5 days ago |
| kalman_filter_improved_different_calculation.py | Add files via upload | 2 weeks ago |
| kalman_filter_improved_less_calculation.py | Rename kalman_filter_improved.py to kalman_filter_improved_less_calcu... | 2 weeks ago |
| matching.py | Add files via upload | 2 years ago |

MOT_Goktug.ipynb

Dosya Düzenle Göster Ekle Çalışma zamanı Araçlar Yardım Son düzenleme tarihi: 31 Ara 2023

+ Kod + Metin

```
#ByteTrack indir. (MAIN)

%cd {HOME}
!git clone https://github.com/ifzhang/ByteTrack.git
%cd {HOME}/ByteTrack

# workaround related to https://github.com/roboflow/notebooks/issues/80
!sed -i 's/onnx==1.8.1/onnx==1.9.0/g' requirements.txt

!pip3 install -q -r requirements.txt
!python3 setup.py -q develop
!pip install -q cython_bbox
!pip install -q onemetric
# workaround related to https://github.com/roboflow/notebooks/issues/112 and https://github.com/roboflow/notebooks/issues/106
!pip install -q loguru lap thop

from IPython import display
display.clear_output()


import sys
sys.path.append(f"{HOME}/ByteTrack")


import yolox
print("yolox.__version__:", yolox.__version__)
```

```
[ ] #ByteTrack indir. (IMPROVED)

%cd {HOME}
!git clone https://github.com/simaygoktug/mot.git
%cd {HOME}/mot

# workaround related to https://github.com/roboflow/notebooks/issues/80
!sed -i 's/onnx==1.8.1/onnx==1.9.0/g' requirements.txt

!pip3 install -q -r requirements.txt
!python3 setup.py -q develop
```

# EXPERIMENTS

We got these results when we took any frame from our road_traffic_clip video and ran the model on it before the changes were made:

*384x640 5 persons, 7 cars, 4 motorcycles, 1 bus, 1 traffic light, 1 bench, 1 backpack, 253.7ms Speed: 4.5ms preprocess, 253.7ms inference, 928.0ms postprocess per image at shape (1, 3, 384, 640)*

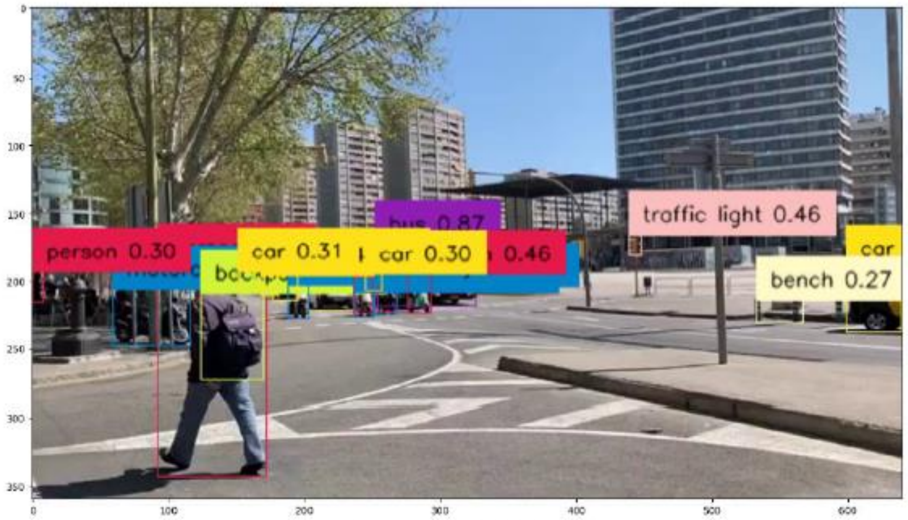


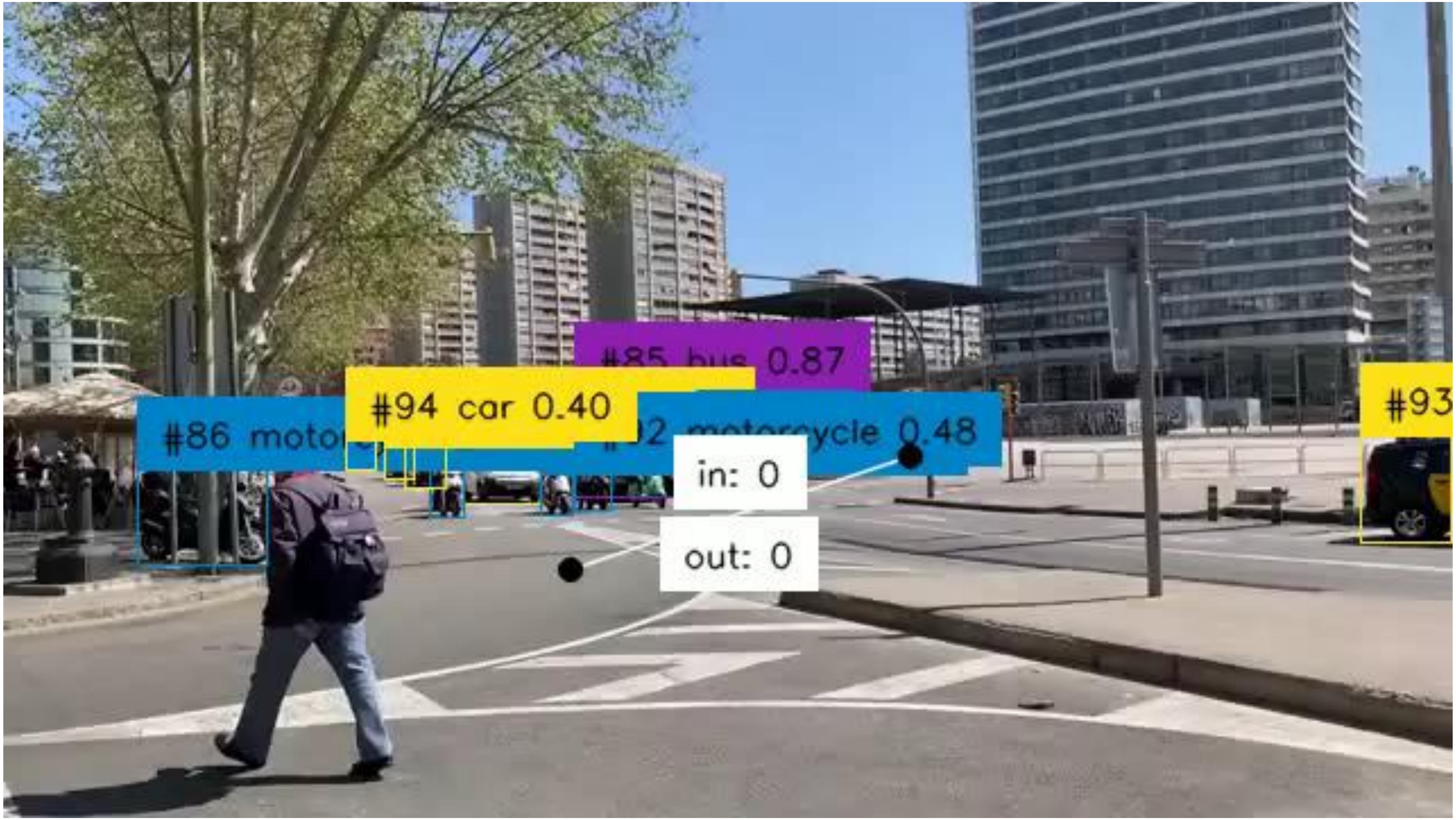

**Image 3.2.1**

The output after the improvements were made:

*384x640 5 persons, 7 cars, 4 motorcycles, 1 bus, 1 traffic light, 1 bench, 1 backpack, 63.1ms Speed: 2.6ms preprocess, 63.1ms inference, 2.7ms postprocess per image at shape (1, 3, 384, 640)*


**Image 3.2.2**

# RESULTS AND CONCLUSION



# RESULTS AND CONCLUSION

$$MOTA = 1 - (FP + FN + IDsw) / GT$$

| for MOT17 | without improvements | with improvements |
|---|---|---|
| FP | **25491** | **25480** |
| FN | **83721** | **83683** |
| IDs | **2196** | **2237** |
| MOTA | **80,3** | **78,89** |


# RESULTS AND CONCLUSION

| for road_traffic_clip | without improvements | with improvements | gain in speed (%) |
| --- | --- | --- | --- |
| Speed (ms) | **253,7** | **63,1** | **402.06** |
| Preprocessing (ms) | **4,5** | **2,6** | |
| Postprocessing (ms) | **928,0** | **2,7** | |

# REFERENCES


- ZHANG, Z., ET AL. : FAIRMOT: TOWARDS A FAIR DETECTION AND TRACKING BENCHMARK. ARXIV:2004.07755 (2020).
- PENG, Z., ET AL. : SIAMMOT: SIAMESE MULTIPLE OBJECT TRACKING. PRO-CEEDINGS OF THE AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE. VOL. 34. NO. 7. 2020.
- BEWLEY, A., ET AL. : SIMPLE ONLINE AND REAL-TIME MULTIPLE OBJECT TRACKING USING KALMAN FILTER AND HUNGARIAN ALGORITHM. PROCEEDINGS OF THE ASIAN CONFERENCE ON COMPUTER VISION. SPRINGER, CHAM, 2016.
- HE, T., ET AL. : TRANSMOT: A TRANSFORMER-BASED METHOD FOR MULTIPLE OBJECT TRACKING. PROCEEDINGS OF THE IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION. IEEE, 2021.
- GODOY, V., NAPA-GARCIA, G., GOMEZ-HERNANDEZ, J. : ENSEMBLE RANDOM FOREST FILTER: AN ALTERNATIVE TO THE KALMAN FILTER FOR INVERSE MODELING. 10.48550/ARXIV.2207.03909 (2022).