Video Surveillance for Road Traffic Monitoring

Project Presentation

Group 1
Marc Pérez, Pau Vallespí, Adrià Subirana
Anna Domenech, Goio Garcia

INDEX

Speed Estimation Using Visual Cues

Introduction
Assumptions
View Transformer
Speed Value Estimation
Results (Given data)
Results (Our data)
Conclusions

Multi-camera Tracking

Introduction
Algorithm
Overview
Temporal Compatibility and Similarity
Association of tracks
Using GPS data
Qualitative results
Quantitative results
Conclusions

SPEED ESTIMATION USING VISUAL CUES

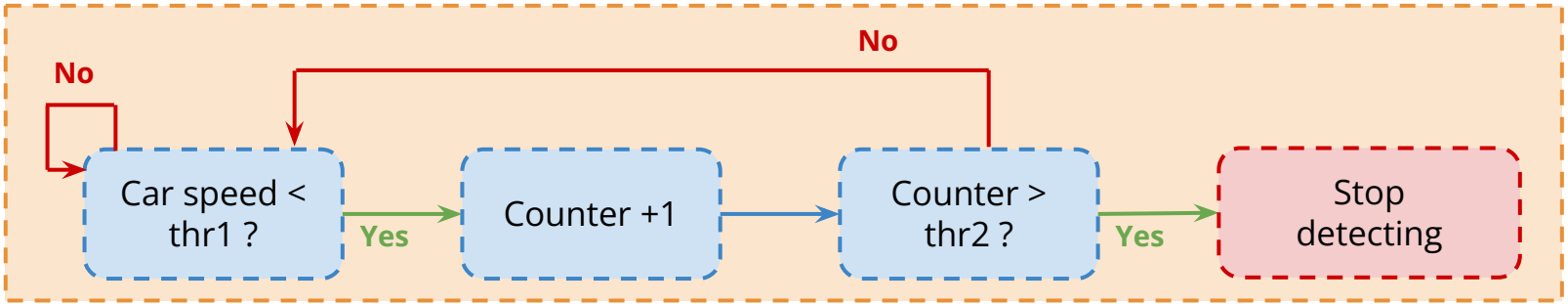
Introduction

- Detection
  - Model → **YOLO v8**
  - Confidence → **50%**
- Path tracking → **Kalman Filter**
- Main *Python Libraries* used:
  - numpy
  - opencv
  - supervision
  - ultralytics
  - scikit-learn

SPEED ESTIMATION USING VISUAL CUES

Assumptions

- Different **height** and **width** for each road
- Bicycles detections are **deleted**
- For **parked** or **stopped** cars we designed a simple pipeline

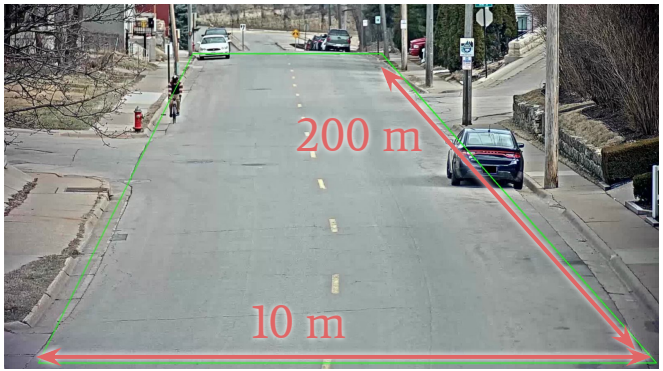


SPEED ESTIMATION USING VISUAL CUES

View Transformer

Provide **source** dimensions (points in the image) and **target** dimensions (estimated distance equivalent distance in the real world)

`cv2.getPerspectiveTransform(source, target)`



SPEED ESTIMATION USING VISUAL CUES

Speed Value Estimation

1. Check Data Availability

Enough centroid coordinates > 10

2. Calculate Distance Traveled

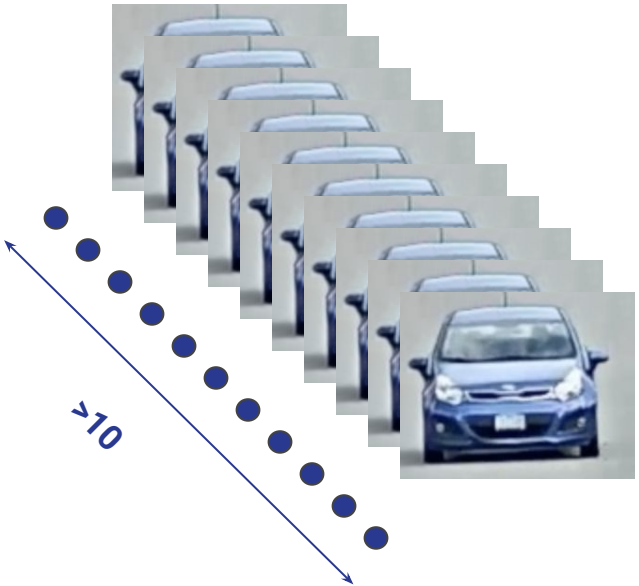
$$\text{Distance} = \sqrt{(x_{\text{end}} - x_{\text{start}})^2 + (y_{\text{end}} - y_{\text{start}})^2}$$

3. Determine Time Interval

4. Calculate Speed

$$\text{Speed} = \frac{\text{Distance}}{\text{Time Interval}}$$

5. Apply Thresholds



SPEED ESTIMATION USING VISUAL CUES

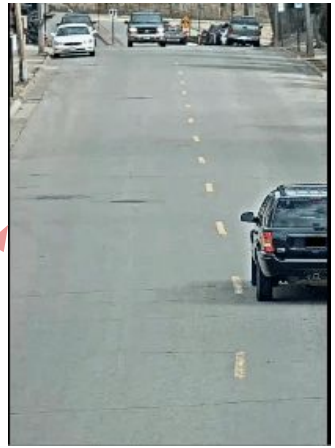
Results (Given data)

SPEED ESTIMATION USING VISUAL CUES

Results (Given data)

- Some **outliers**.
- The more a car resembles a **van**, the **less accurate** its estimation tends to be.
- Speed **decreases** as cars move further away and **increases** as they approach closer.

ID	Average speed (km/h)	Maximum Speed (km/h)
1	29.96	42.69
2	27.75	38.92
3	30.59	41.40
4	27.21	40.94
5	37.52	87.78
6	17.76	19.62
7	38.02	50.49
8	28.01	65.69
9	27.75	38.66
10	45.27	124.83
11	31.17	183.73
12	32.21	45.12
13	26.54	35.57

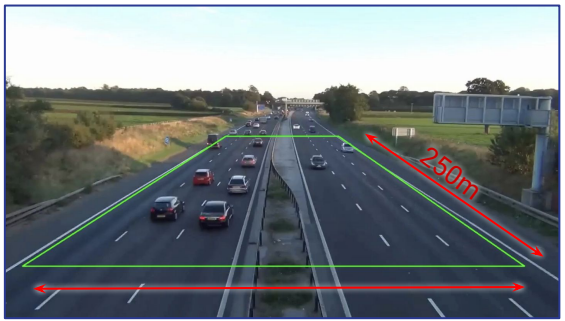


SPEED ESTIMATION USING VISUAL CUES

Results (Our data)

SPEED ESTIMATION USING VISUAL CUES

Results (Our data)



Original video

(Pau & Marc)

SPEED ESTIMATION USING VISUAL CUES

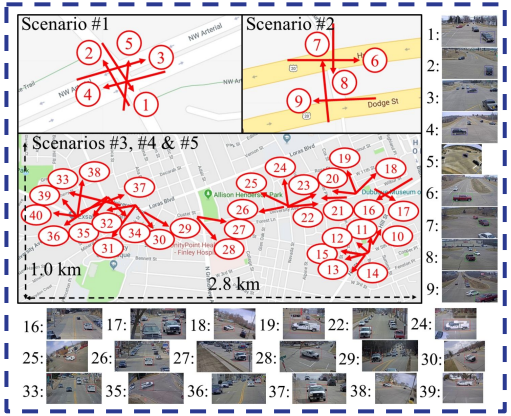
Conclusions

- Leveraged **Kalman Filter** for precise path tracking in detection tasks.
- Utilized **YOLO v8** Model with a confidence threshold of **50%** for robust object detection.
- Integrated essential **Python libraries** such as **numpy**, **opencv**, **supervisely**, **ultralytics**, and **scikit-learn** for efficient processing.
- Engineered a pipeline to identify and manage **stopped** or **parked** cars thus removing them from the output.
- Some insights from the analysis reveal **problems** in the detections influenced by vehicle **characteristics** and **proximity**.
- Appearance of **some outliers** in speed measures, but **consistent results** in average
- A good **estimation of the real distances** is a key step for an accurate result

MULTI CAMERA TRACKING

Introduction

- To perform multi camera tracking, we need **high quality multi-object tracking**.
- To associate data across cameras we need **higher level features**.
  - We use a ResNet50-IBN Re-ID Network(*) trained on vehicles. The following features are extracted:
    - Mean Feature.
    - From pre-trained SVM → extract categorical attributes: Color and Type of car.
  - Matching based on Hungarian Algorithm (**).



Dataset used (***)

ByteSort (**)

- Uses both high and low confidence detections.
- Kalman filter.
- Data Association is performed using IoU and visual features.

(*)Joint discriminative and generative learning for person re-identification. Git Hub Repository.
(**) Szűcs, G., Borsodi, R., Papp, D. (2023). Multi-Camera Trajectory Matching based on Hierarchical Clustering and Constraints. Multimedia Tools and Applications. Github Repository. Paper.

MULTI CAMERA TRACKING

Algorithm: *Overview*

Step 1. Initialization.

1. **Camera synchronization.** Convert the offset of each camera into frames and obtain the global start and end for each camera.
2. Create a list with all the tracks and assign a **global id**.

Step 2. Compute tracks **temporal compatibility** and **similarity** to infer **pairs of candidates for a match.**

Step 3. Associate tracks. All candidates are ordered by **similarity** and processed in this order.

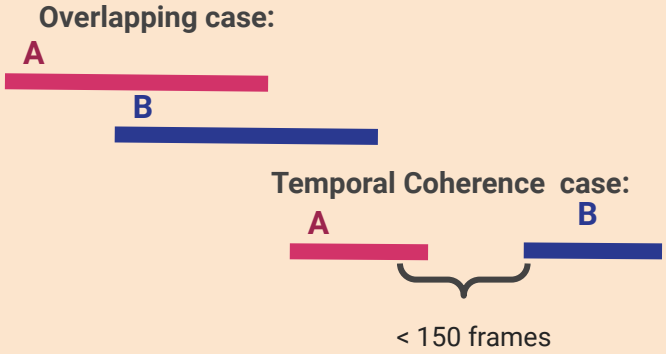
1. If any of the tracks of a candidate (i-th,j-th) has not been previously used:
    a. Merge the i-th and j-th candidate. Recompute Mean Feature.
    b. Remove j-th candidate.
    c. If the new visual descriptor is similar enough to any track and shares temporal compatibility. Add this multitrack to the list of candidates.

MULTI CAMERA TRACKING

Algorithm: *Temporal Compatibility and Similarity*

Temporal compatibility Matrix.

- NxN matrix where N is the total number of tracks across all the cameras.
- The i-th and j-th track are compatible if they overlap in time or have a temporal coherence transition.



Similarity Metrics:

The features obtained with ResNet for all the tracks are concatenated to form F. Then, the similarity matrix is computed as: $sim = FF^T$

Categorical Attributes are used to filter out candidates.

MULTI CAMERA TRACKING

Algorithm: *Association of tracks*

Candidates [sim, iter*, i, j]

All pairs of tracks which have temporal compatibility and:

- Similarity > 0.5
- Same color vehicles.

**Iter* is used to mark the iteration in which a candidate has been proposed.*

Init. MutlicameraTrack object for each track.
[MulltiCameraTrack(),..]

Order by similarity to process high confidence information first.

For each Pair of (i-th,j-th) MCT in candidates:

If i-th and j-th have not been used:

- Add j-th to i-th MCT.
- Mark j-th MCT as used.
- Recompute i-th MCT mean feature, and add to candidates iif is similar enough to other existing MCT.

Re-assign the ids only considering MultiCameraTracks with more than one camera assigned.

MULTI CAMERA TRACKING

Algorithm: *Using GPS data (I)*

- The dataset provides us with **calibration matrices** for each camera.

Calibration matrix

The 3x3 homography that converts **2D GPS** coordinates to **2D screen** coordinates

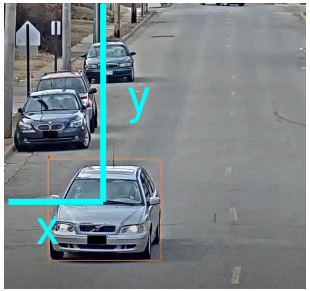
- By taking the **inverse** of this matrix, it is possible to obtain the **GPS coordinate** of objects in the screen, assuming they are on the ground plane.
- **GPS coordinates** give us a camera-independent feature that can be compared across cameras.

$$\text{Homogeneous screen coords} \rightarrow K^{-1} \rightarrow \text{Homogeneous GPS coords}$$

MULTI CAMERA TRACKING

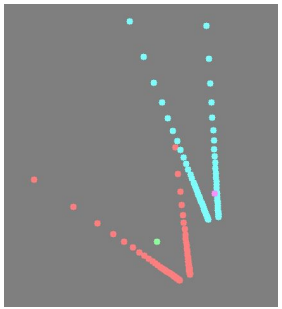
Algorithm: *Using GPS data (II)*

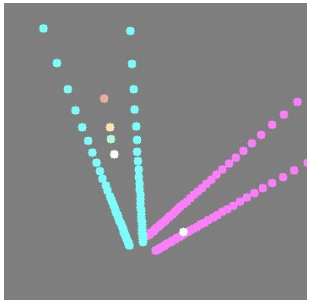
Track merging implementation:



Project to GPS coordinates

After completing the steps of camera synchronization and given temporally compatible candidates:





Extract screen coordinates of the object in both tracks

The homography maps points located inside the ground plane. To approximate the point of contact with the ground, the center of the bounding box is taken.

If tracks temporally intersect:

Same object is being observed from 2 cameras at the same time.

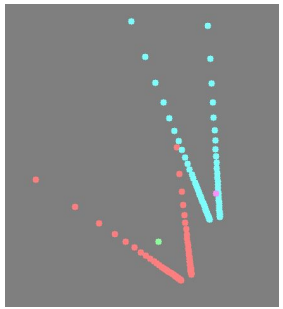
If tracks do NOT temporally intersect:

There exists a transition from the first track to the second one.

MULTI CAMERA TRACKING

Algorithm: *Using GPS data (III)*

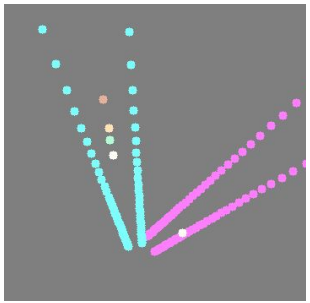
Track merging implementation:



If tracks temporally intersect:

Same object is being observed from 2 cameras at the same time.

Merged if:
The position is the same (under a certain threshold) the majority of the intersection



If tracks do NOT temporally intersect:

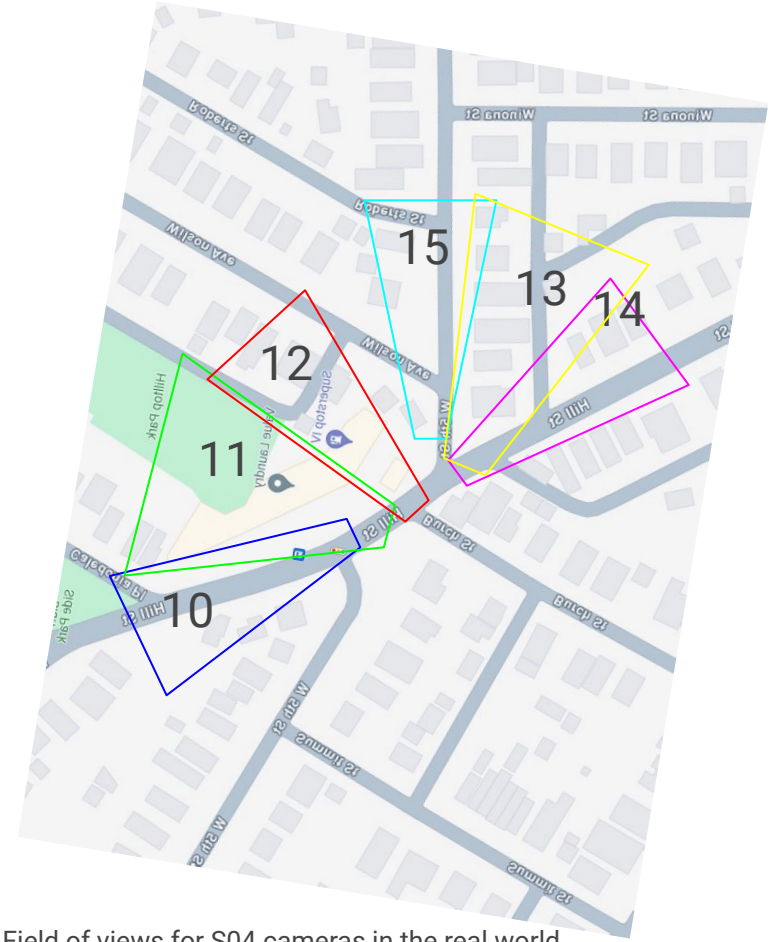
There exists a transition from the first track to the second one.

Merged if:
The transition is plausible. Use extrapolated speed from last 5 frames.

MULTI CAMERA TRACKING

Algorithm: *Using GPS data (IV)*

What you order:

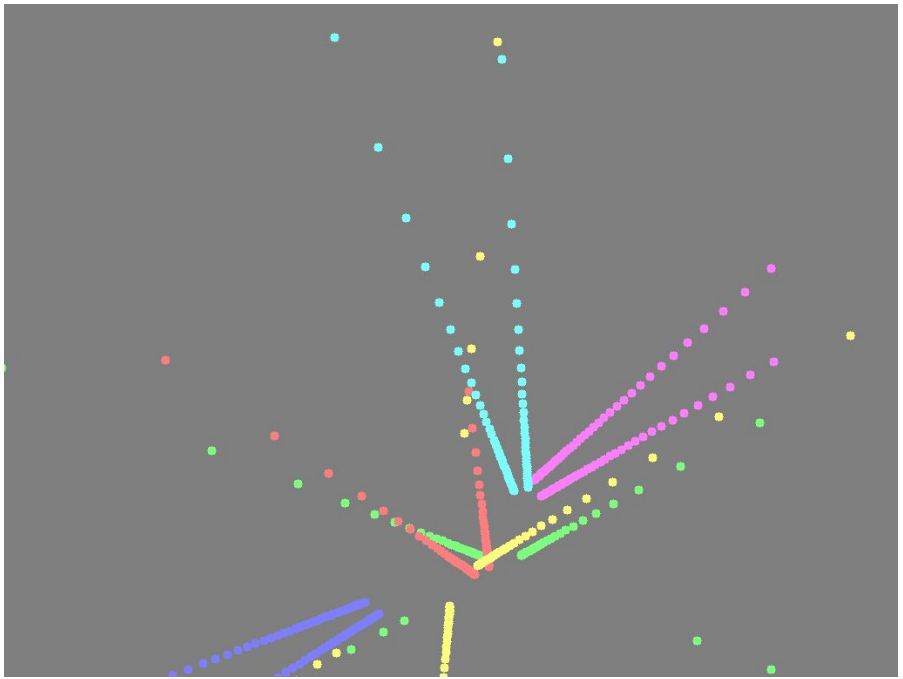


Field of views for S04 cameras in the real world

MULTI CAMERA TRACKING

Algorithm: *Using GPS data (V)*

What you get:

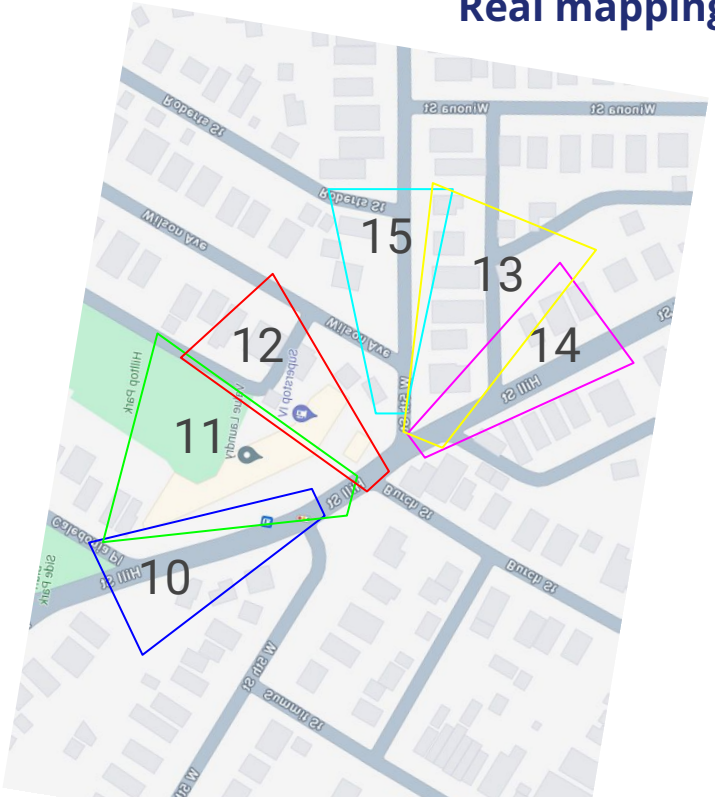


Field of views for S04 cameras given by calibration matrix

MULTI CAMERA TRACKING

Algorithm: *Using GPS data (VI)*

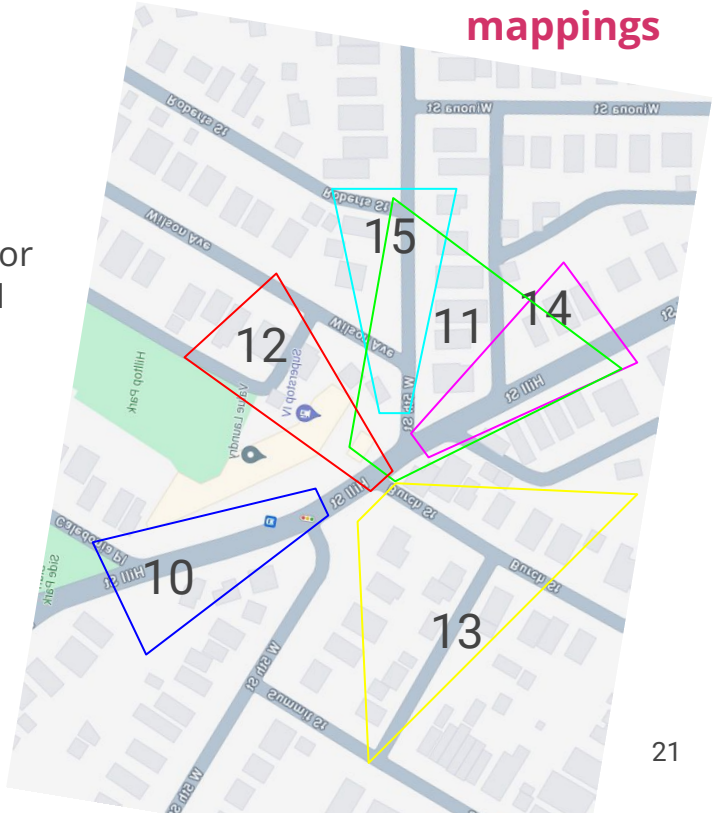
Real mappings



The calibration matrices for camera 13 (yellow) and 11 (green) produce incorrect results.

Projection from screen coordinates to GPS coordinates is also unreliable.

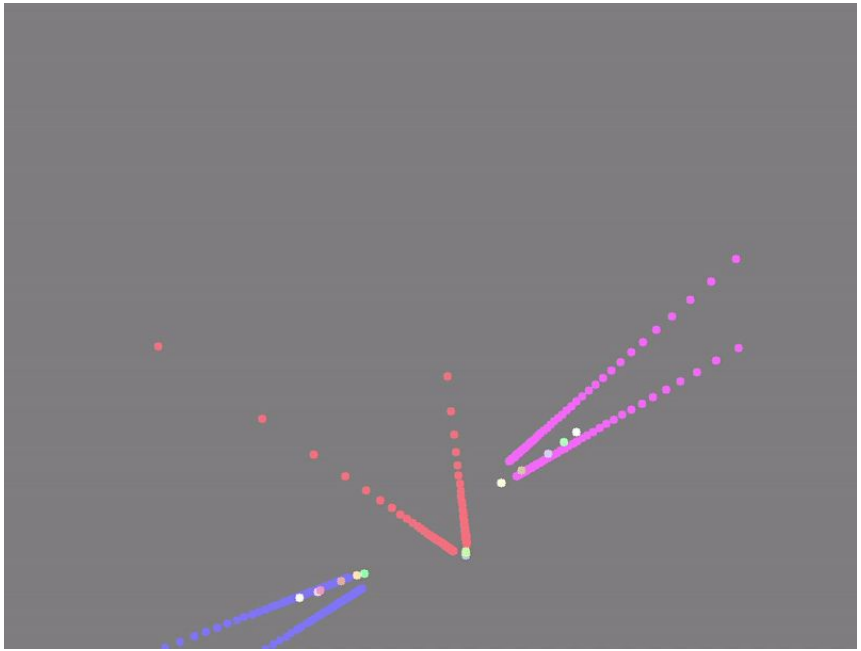
Calibration mappings



MULTI CAMERA TRACKING

Algorithm: *Using GPS data (VII)*

Visualization of Camera 10, Camera 12 and Camera 14 projections



Main problems:

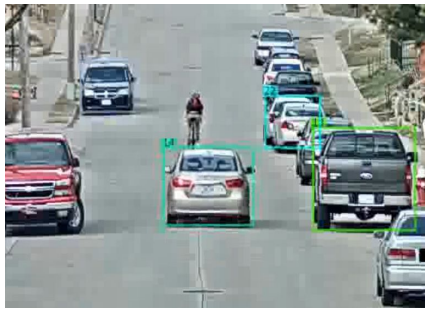
- Points near the top of the image are projected too far away.
- Points near the bottom of image are too close to differentiate independent objects.
- Difficult to account for fisheye lens.

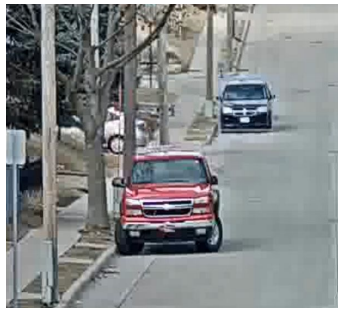
Because of these problems, using GPS coordinates has not been possible.

MULTI CAMERA TRACKING

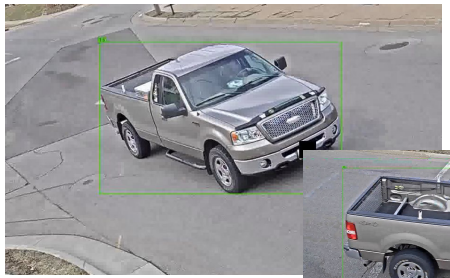
Qualitative results (I)

C015





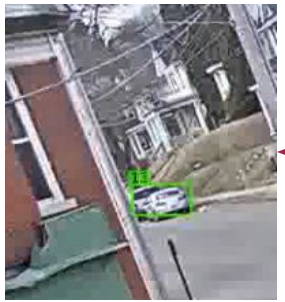
Residential Scenarios have more complex interactions. Change of Label ID, illumination may affect quality of extracted features.



C013

C012

C014



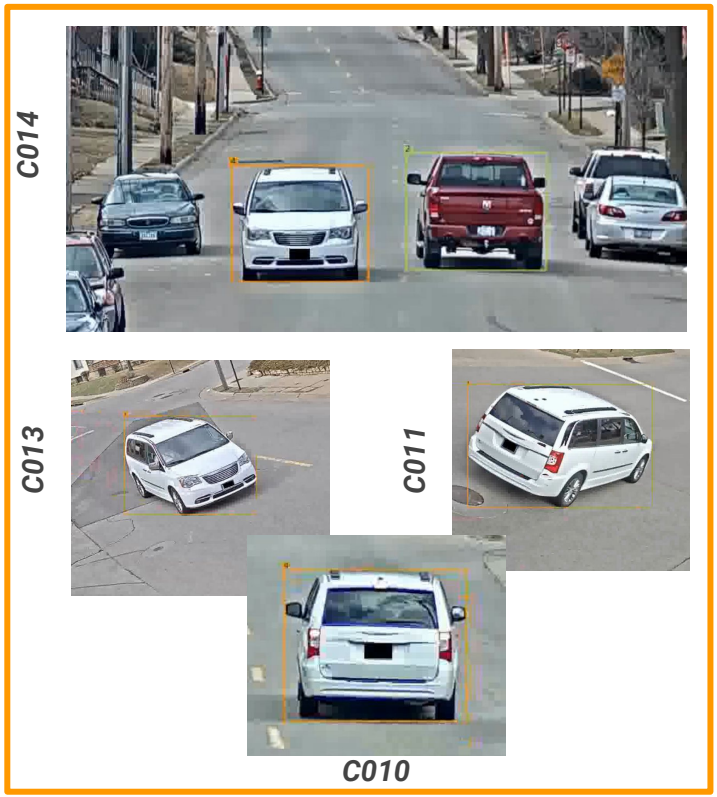
C013

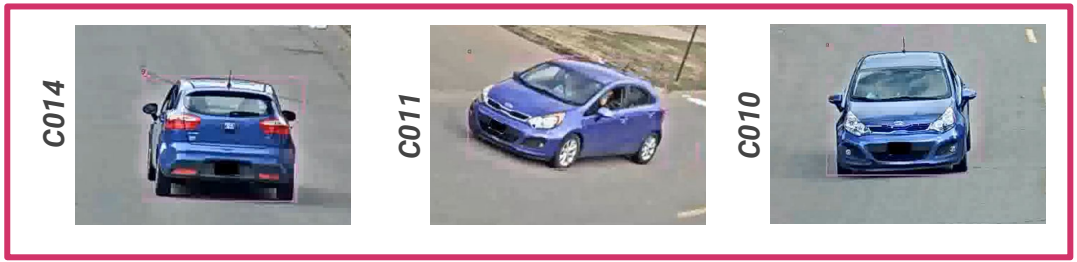
Label 13 wrongly matched.

Label 13 correctly matched.

MULTI CAMERA TRACKING

Qualitative results (II)





MULTI CAMERA TRACKING

Quantitative results

IDF1: 24.2 **S03**

C011

C010

C014

* This sequences are not shown temporarily synchronized

MULTI CAMERA TRACKING

Conclusions

- MTMC performance **highly depends on tracking and detection** accuracy. Errors in previous steps will directly affect.
- Use of **Re-ID network** for feature extraction and categorical attributes help on data association. But limitations can be observed, such as susceptibility to changes in illumination.
- The use of **GPS coordinates** could help to improve the system performance, giving spatial correlation.

Video Surveillance for Road Traffic Monitoring

Project Presentation

Group 1
Marc Pérez, Pau Vallespí, Adrià Subirana
Anna Domenech, Goio Garcia