A multi-camera solution for counting vehicles on the edge

Luca Ciampi, Claudio Gennaro, Fabio Carrara, Fabrizio Falchi, Claudio Vairo, Giuseppe Amato

luca.ciampi@isti.cnr.it

I-CITIES 2021

Outline

1. **Introduction and Motivations**
   Smart Cities need Smart Parking

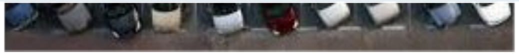
2. **A multi-camera solution for counting vehicles on the edge**
   A deep learning-based detector together with a decentralized technique that exploits the geometry of the captured images, running on the edge of the network

3. **Experimental evaluation**
   Results and Conclusion

INTRODUCTION

"30% OF TRAFFIC CONGESTION WITHIN CITIES IS ATTRIBUTABLE TO DRIVERS TRYING TO FIND AVAILABLE PARKING."

Source: "Cruising for parking" by Donald C. Shoup



CRUCIAL TO IMPROVE URBAN ENVIRONMENT AND LIFE OF CITIZENS

- CITY MOBILITY
- POLLUTION MONITORING
- INFRASTRUCTURE MANAGEMENT

Barrier + Infrared Sensors

Not feasible in every scenario!

- Vehicles entering and not parking
  - Express couriers
  - Provisioning trucks
- Bicycles, motorcycles, pedestrians

Ground Sensors

- One sensor for each parking space
- Very expensive (~80€ each)
- Installation costs
- Maintenance cost

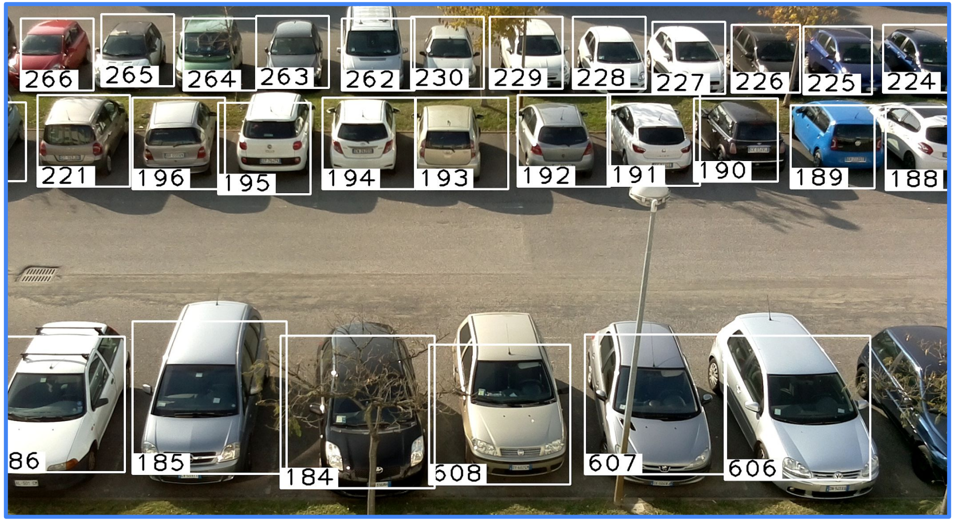
PROPOSED SOLUTION

Why Visual Monitoring?

- **Cheaper**: one camera can monitor up to 50 cars
- **Simple Infrastructure**: possible reuse of available surveillance infrastructure
- **Versatile**: Smart video surveillance (useful for other tasks)
- **Expandable**: ready-to-use solution, simple "plug-and-play" insertion of new cameras into the system

Detection-based approach: localize and count

➡ Lot Occupancy Detection



➡ Counting Vehicles

Multi-Camera Scenario

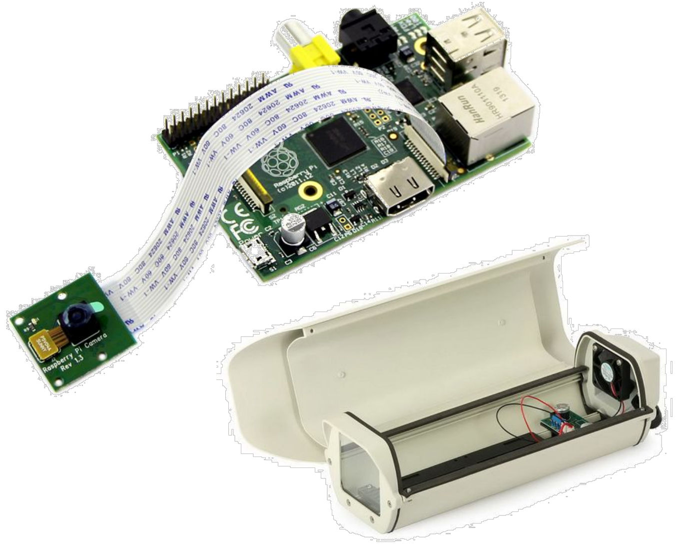
- robustness: monitor the same parking lot from different perspectives and viewpoints
    - redundancy provides robustness and fault-tolerance
- expandability: cover a wider area
- problem of aggregating data from individual cameras (partially overlapped FOVs)

Multi-camera system to automatically estimate the number of cars present in the entire monitored parking area

- It combines a deep learning-based detector together with a decentralized technique that exploits the geometry of the captured images
- It runs directly on the edge devices (i.e., smart-cameras)

Smart Cameras

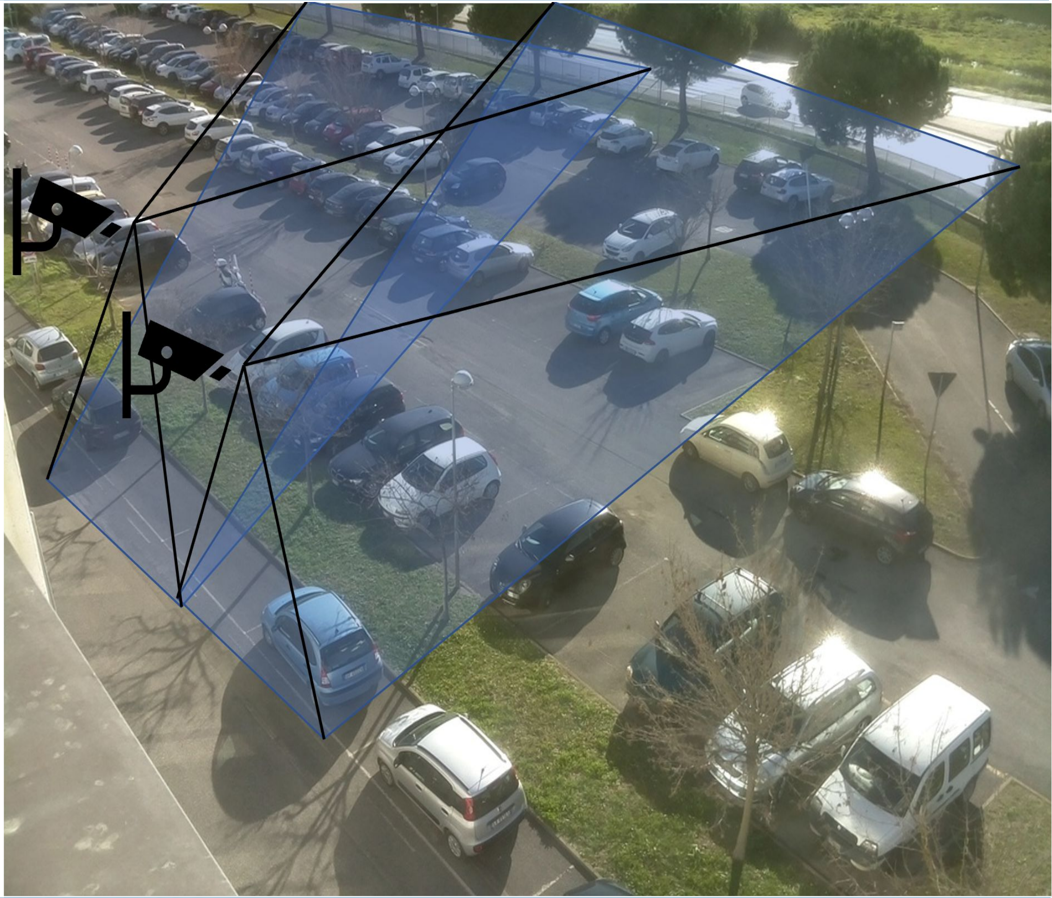
- Device able to capture images
- Computational Capabilities: it analyzes images and takes decision directly onboard
- Networking: it transmits elaborated results rather than video streams



The simulated scenario

9 Cameras

- various perspectives
- partially overlapped FOVs (Multi-Camera Scenario)
- many illuminations, weather conditions
- partial occlusion patterns obstacles

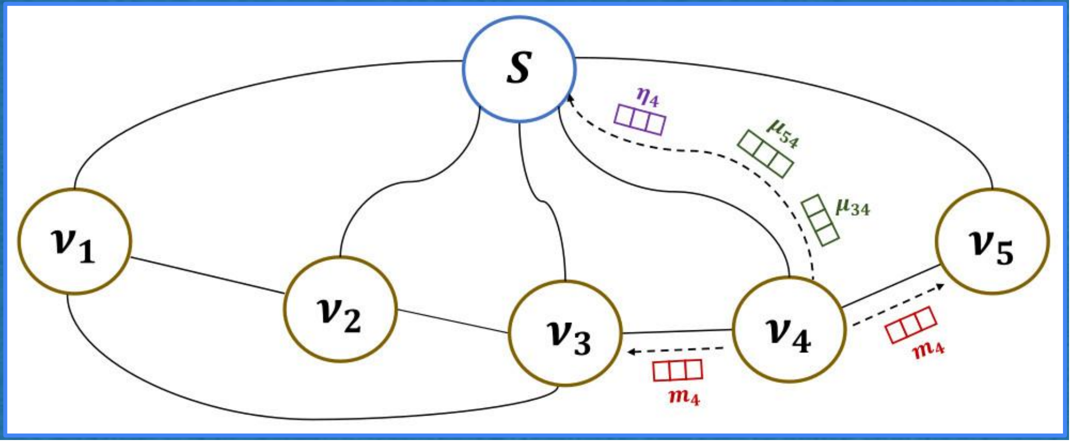


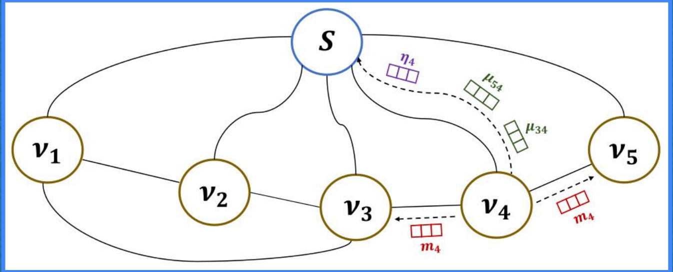
System overview and modelization

Example: system with n=5 cameras

- We model the system as a graph
  - n nodes vi, one for each camera
  - a sink node S
  - nodes can elaborate data and communicate
  - edges connect neighboring nodes (having shared FOV)

Modelization





Algorithm 1 : Initialization

At each Initialization Signal by S, each node ν_i performs the following steps:

```
1: ReceiveInitSignal()                         ▷ waits the initialization signal from S
2: image_i ← CameraCapture()
3: for each j ∈ J do                           ▷ J is the set of neighboring nodes of node ν_i
4:     SendImage(image_i, ν_j)                 ▷ sends image_i to node ν_j
5:     image_j ← ReceiveImage()                ▷ receives image_j from node ν_j
6:     H_{j,i} = ComputeHomography(image_j, image_i)
```

Homography Matrix → it maps points from a 2D image to its projection on a second 2D image having a shared area

Performed automatically! Given a pair of neighboring cameras:

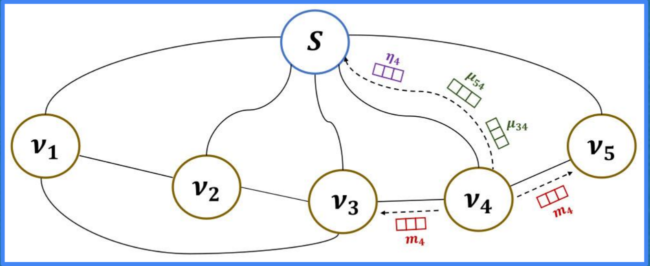
- Find SIFT keypoints and feature descriptors of the two images
- Filter matched feature descriptors using Euclidean distance
- Apply RANSAC and compute Homography

Stitching of two images coming from two neighboring cameras. We exploited the computed homography matrix.

Algorithm 2 : Local Counting

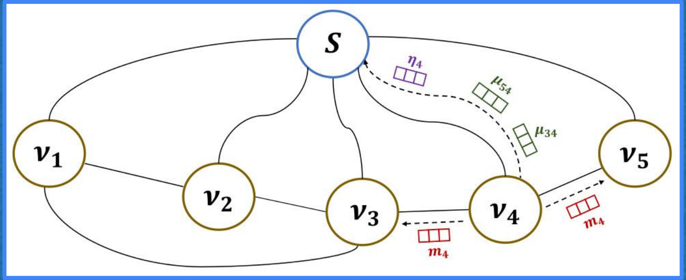
At each Computational Signal by S, each node ν_i performs the following steps:

```
ReceiveComputSignal()                          ▷ waits the computational signal from S
image_i ← CameraCapture()
masks_i ← MaskRCNN(image_i)
η_i ← |masks_i|
SendMessage(η_i, S)                            ▷ sends η_i to Sink node S
m_i ← PackMessage(masks_i)                     ▷ builds message m_i containing masks_i
for each j ∈ J do                              ▷ J is the set of neighboring nodes of node ν_i
    SendMessage(m_i, ν_j)                      ▷ sends m_i to node ν_j
    m_j ← ReceiveMessage()                     ▷ receives message m_j from node ν_j
    masks_j ← UnpackMessage(m_j)               ▷ unpacks m_j containing masks_j
    μ_{j,i} ← compute_μ(masks_i, masks_j, H_{j,i})
    SendMessage(μ_{j,i}, S)                    ▷ sends μ_{j,i} to Sink node S
```



$\mu \rightarrow$ represents the num of cars detected by ν_j and already detected by ν_i

Global Counting



Algorithm 4 : Global Counting

The Sink node S performs the following steps:

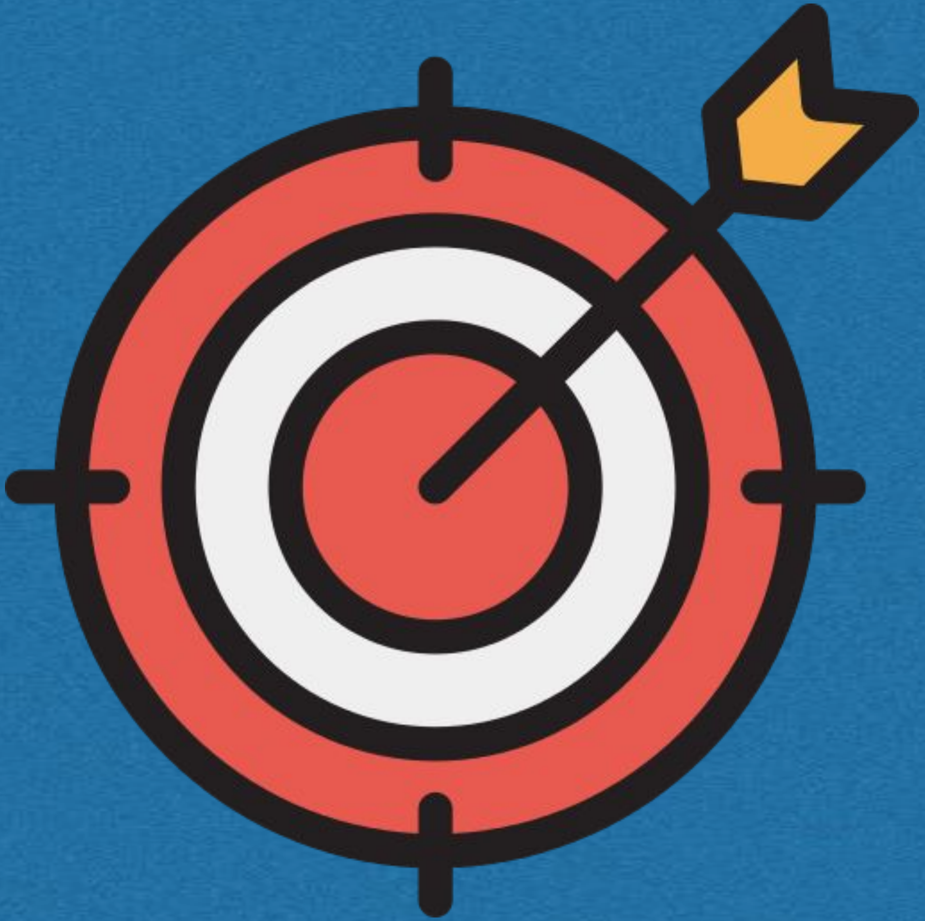
1: **for each** $(\mu_{i,j}, \mu_{j,i})$ **do**

2: $\quad \overline{\mu_k} \leftarrow \textsc{Aggregate}(\mu_{i,j}, \mu_{j,i})$

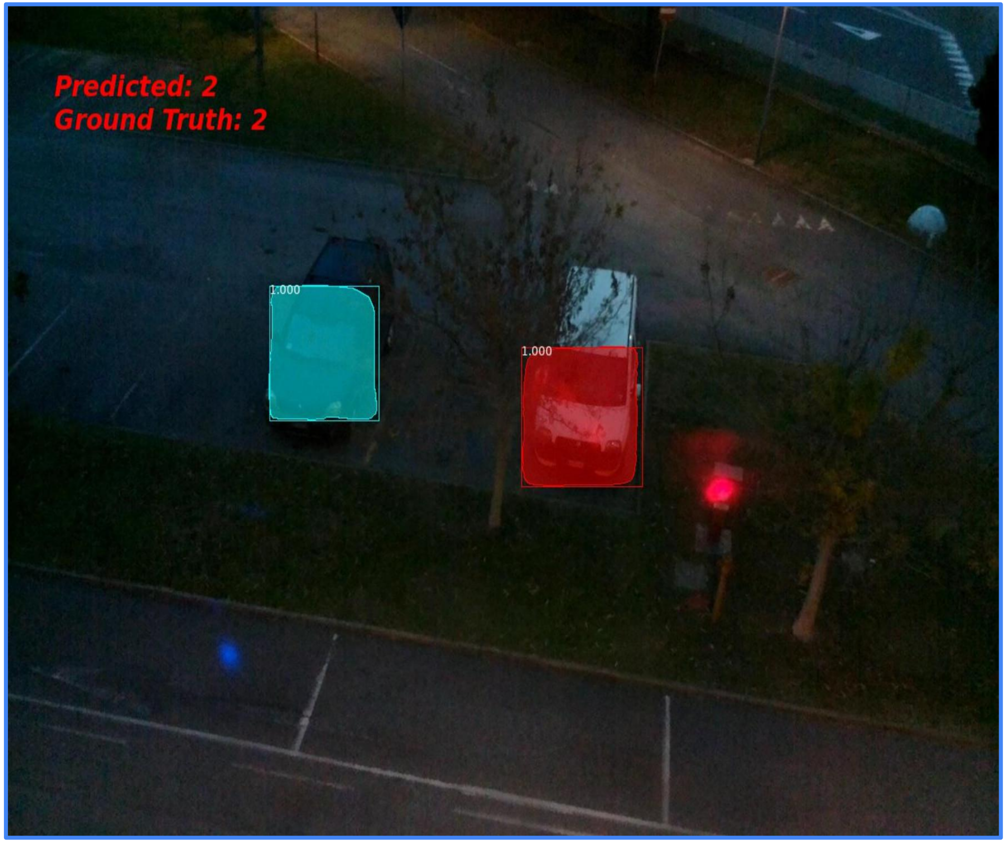
3: global_cars_count $\leftarrow \sum_{n=1}^{N} \eta_n - \sum_{k=1}^{K} \overline{\mu_k}$

$\triangleright$ N is the set of nodes, K is the set of aggregations

RESULTS



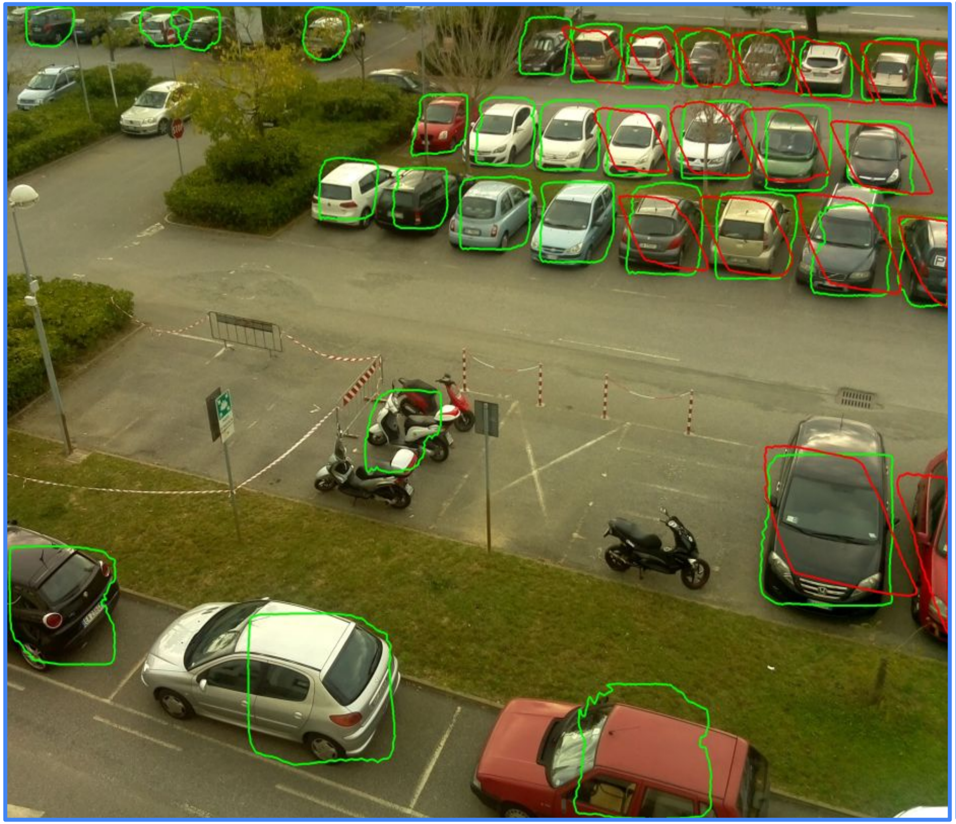
Single Camera Detections and Counting



Projection of detections between two adjacent cameras (using homography)

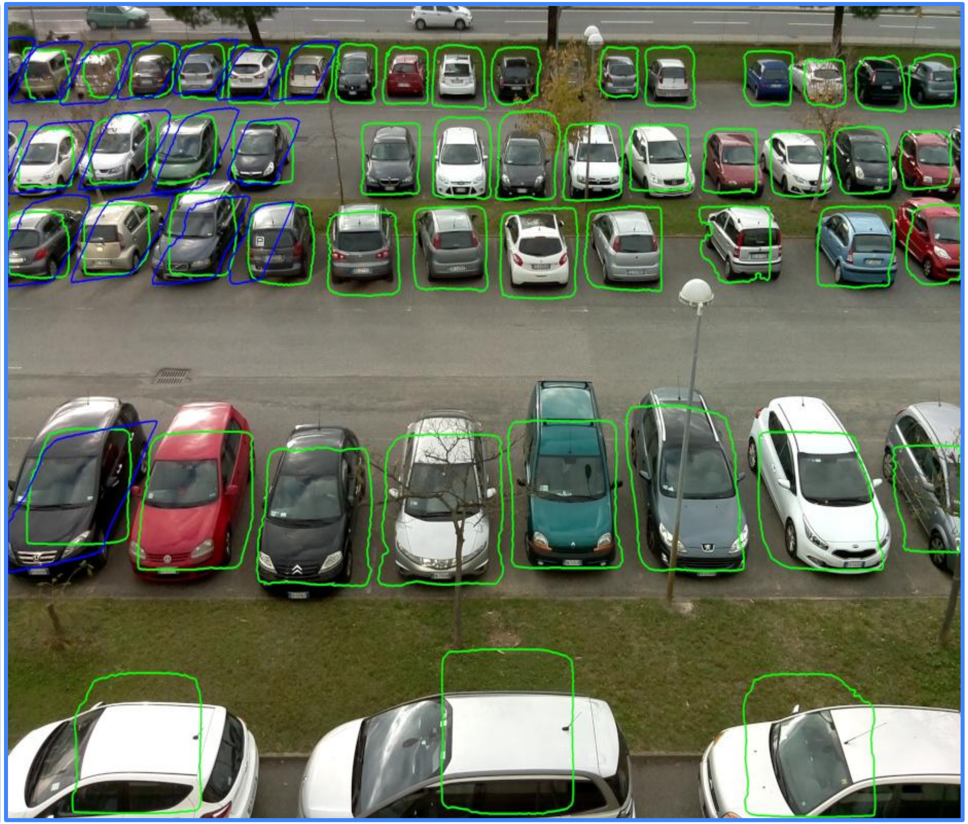
CAMERA 9

In **Red** → From Camera 8 to Camera 9



CAMERA 8

In **Blue** → From Camera 9 to Camera 8



L. Ciampi, C. Gennaro, F. Carrara, F. Falchi, C. Vairo, G. Amato

Metrics

MAE: Mean Absolute Error

$$MAE = \frac{1}{N}\sum_{n=1}^{N} |c_n^{gt} - c_n^{pred}|$$

MSE: Mean Squared Error

$$MSE = \frac{1}{N}\sum_{n=1}^{N} (c_n^{gt} - c_n^{pred})^2$$

MRE: Mean Relative Error

$$ARE = \frac{1}{N}\sum_{n=1}^{N} \frac{|c_n^{gt} - c_n^{pred}|}{num\_vehicles}$$

Results - Local Counting single images

	Sunny			Overcast			Rainy		
Train Set	MAE	MSE	MRE	MAE	MSE	MRE	MAE	MSE	MRE
Sunny	-	-	-	0.29	0.34	0.009	0.96	2.78	0.02
Overcast	0.62	1.09	0.02	-	-	-	0.56	1.26	0.01
Rainy	0.84	1.65	0.02	0.49	0.65	0.01	-	-	-

		Test Set								
Metric	Train Set	C1	C2	C3	C4	C5	C6	C7	C8	C9
MAE	C1	-	0.77	1.21	2.53	3.26	2.57	2.88	2.88	1.54
	C8	3.87	0.85	0.76	0.45	0.48	0.71	1.07	-	0.41
MRE	C1	-	0.08	0.05	0.06	0.07	0.05	0.06	0.05	0.05
	C8	0.11	0.09	0.03	0.01	0.01	0.01	0.02	-	0.01
MSE	C1	-	1.48	2.91	10.61	20.24	13.50	19.82	17.30	7.19
	C8	22.60	1.78	1.36	0.57	0.74	0.95	4.97	-	2.13

Metrics

MAE:
Mean Absolute Error

$$MAE = \frac{1}{N}\sum_{n=1}^{N} |c_n^{gt} - c_n^{pred}|$$

MSE:
Mean Squared Error

$$MSE = \frac{1}{N}\sum_{n=1}^{N} (c_n^{gt} - c_n^{pred})^2$$

MRE:
Mean Relative Error

$$ARE = \frac{1}{N}\sum_{n=1}^{N} \frac{|c_n^{gt} - c_n^{pred}|}{num\_vehicles}$$

Results - Global Counting entire car park

	Error			Absolute Err.			Squared Err.			Relative Err. (%)		
	N	**M**	**O**	**N**	**M**	**O**	**N**	**M**	**O**	**N**	**M**	**O**
Overcast-1	124	-33	**2**	124	33	**2**	15,376	1,089	**4**	71.6	19.0	**1.2**
Overcast-2	131	-26	**1**	131	26	**1**	17,161	676	**1**	76.1	15.1	**0.6**
Rainy-1	80	-39	**-5**	80	39	**5**	6,400	1,521	**25**	47.6	23.2	**2.9**
Rainy-2	105	-44	**-5**	105	44	**5**	11,025	1,936	**25**	54.4	22.8	**2.6**
Sunny-1	117	-38	**2**	117	38	**2**	13,689	1,444	**4**	68.0	22.1	**1.2**
Sunny-2	113	-37	**2**	113	38	**2**	12,769	1,444	**4**	66.1	22.2	**1.2**
Mean	111.6	-36.1	**-0.5**	111.6	36.3	**2.8**	12,736.6	1,351.6	**10.5**	63.9	20.7	**1.6**

N: Naïve Counting; **M**: Overlap Masking; **O**: Ours (mean aggr., IoU Threshold $\tau = 0.2$)

Conclusion

- We presented a distributed artificial intelligence-based system that automatically counts the vehicles present in an *entire* parking lot using images taken by multiple smart cameras.
- All the computation is performed in a distributed manner at the edge of the network
- No need for any extra information of the monitored parking area, such as the location of the parking spaces
- We modeled our system as a graph, where the nodes, i.e., the smart cameras, are responsible for estimating the number of cars present in their view and merging data from nearby devices that have an overlapping field of view. Our solution is simple but effective, combining a deep-learning technique with a distributed geometry-based approach.

Thanks for your attention!

Questions?

luca.ciampi@isti.cnr.it