Event-Driven Dynamic Platform Selection for Power-Aware Real-Time Anomaly Detection in Video

C. G. Blair [1] N. M. Robertson[2]

[1]Institute for Digital Communications
University of Edinburgh

[2]Visionlab, Institute for Sensors, Signals and Systems
Heriot-Watt University

Intl. Conf. on Computer Vision Theory and Applications. Lisbon, January 2014

Outline

Defence Applications

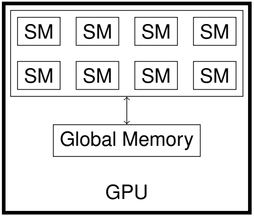
- Increased situational awareness & surveillance requirements.
  - Human vigilance decays over time.
- Increasing processing power in vehicles and autonomous sensors.
  - Engine on/off $\rightarrow$ power available changes; balance power and desire for fast, accurate detections?

Architectures

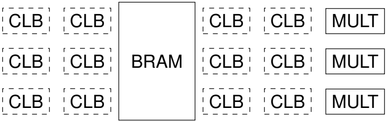
Given a complex, demanding algorithm, choose:

- **GPU:** parallelise/ accelerate by mapping algorithm onto existing architecture. High power, accuracy.
- **FPGA:** accelerate by instantiating architecture to match algorithm. Lower power, harder to write.
- Combinations?

GPU:



FPGA:



Algorithm Mapping to Hardware

- How to select architecture for given algorithm? Both fine-grained and coarse-grained.
- Design Space Exploration $\rightarrow$ Multidimensional space (power, latency, chip area, accuracy, ...)
- Large search space: exhaustive search $\rightarrow$ dynamic, local+taboo, genetic algorithm search.
- Weighting & constraints depend on specific application, *but may change over time*. Consider vehicle example.

Anomaly Detection: Vehicle parking

- Anomaly detection categories (A/B/C): “very different from training set“ / ambiguous / weak visual evidence [1]
- i-LIDS “Parked Vehicle” dataset.
- Real (messy) surveillance data.

[1]: Loy *et al.*, Detecting and discriminating behavioural anomalies. Pattern Recognition, 2011.

Related Work

- Manually select yellow line regions and note obstructions: sensitive to camera changes, detects non-vehicles [1].
- Real-time blob detection (no class information) [2].
- *Different problem: power-aware platform selection at runtime?*

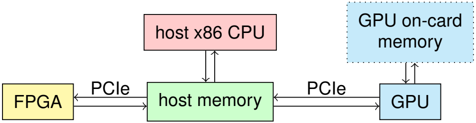
[1]: Albiol *et al.*, Detection of Parked Vehicles Using Spatiotemporal Maps, IEEE J. Intelligent Transportation Systems, 2011.

[2]: Bevilacqua & Vaccari, Real time detection of stopped vehicles in traffic scenes. AVSS 2007.

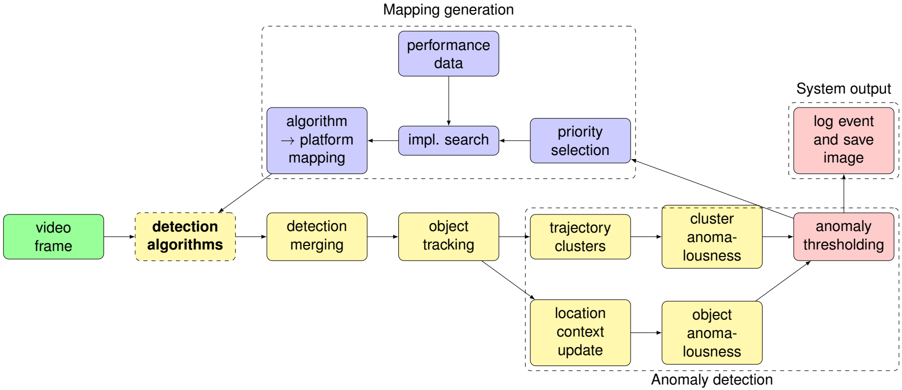
Problem Statement

- Surveillance Application:
  - Real-time detection of people and vehicles $\rightarrow$ parked vehicles.
  - Awareness of system power consumption.
  - *Re-map (trade-off) processing between architectures on-the-fly if we see potentially anomalous behaviour.*

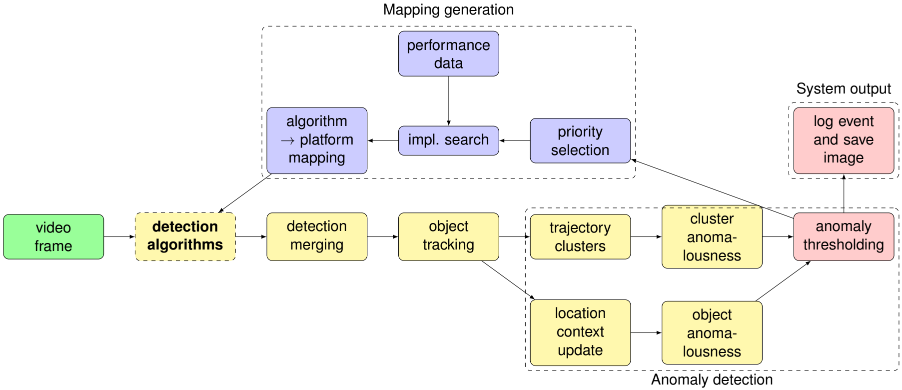
System



FPGA: Xilinx Virtex-6 VLX240.
GPU: nVidia GTX560, 384 CUDA cores.
CPU: Intel Xeon dual-core.
Transfers use DMA but no direct path between accelerators.



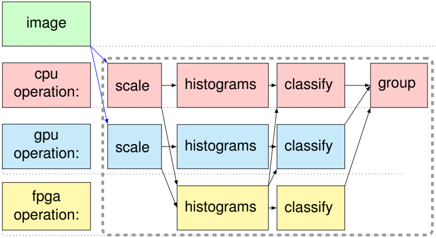
Only “detection algorithms” stage is computationally expensive



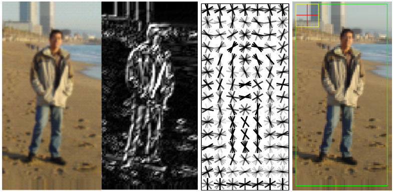
Only “detection algorithms” stage is computationally expensive

Computationally Expensive Detectors

- Histogram of Oriented Gradients:
  - Sliding-window classifier at multiple scales.
  - Local dense features extraction
  - Linear SVM classifier
- Label each as *scale-histogram-classify* (*ccc* or *gfg*).
- Measure time, power, accuracy of every version. [1]

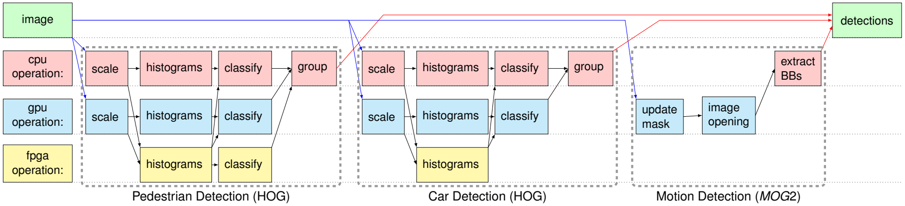


Pedestrian Detection (HOG)

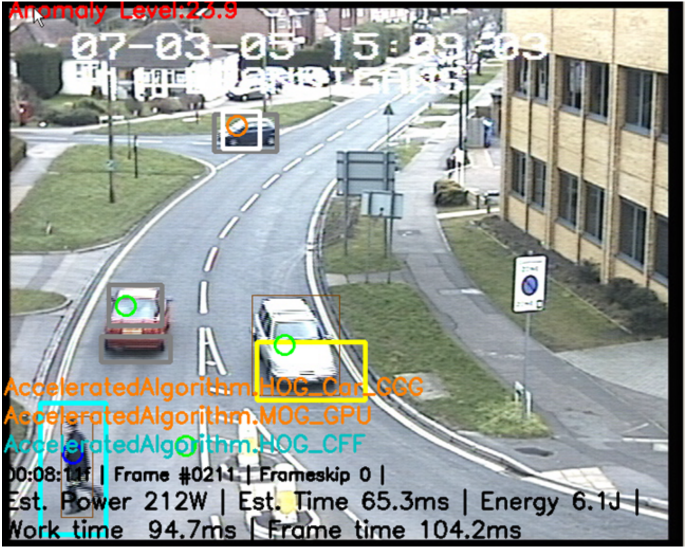


[1] Blair, C., Robertson, N.M. & Hume, D., Characterising a Heterogeneous System for Person Detection in Video using Histograms of Oriented Gradients: Power vs. Speed vs. Accuracy. IEEE J. Emerging and Selected Topics in Circuits and Systems, 2013

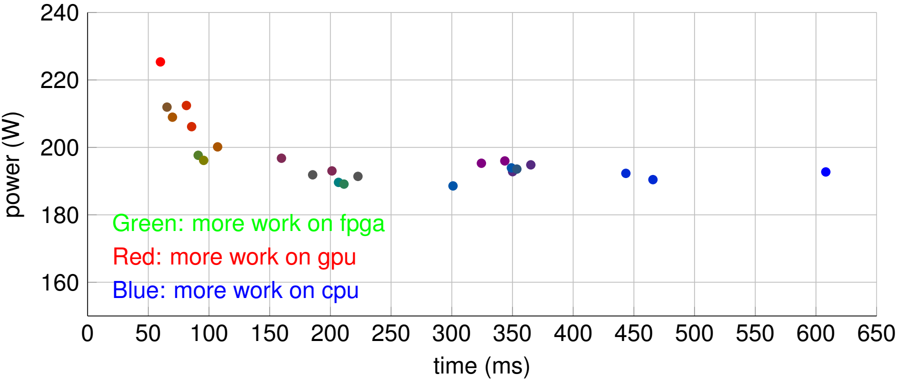
Car and motion detection



Detector outputs



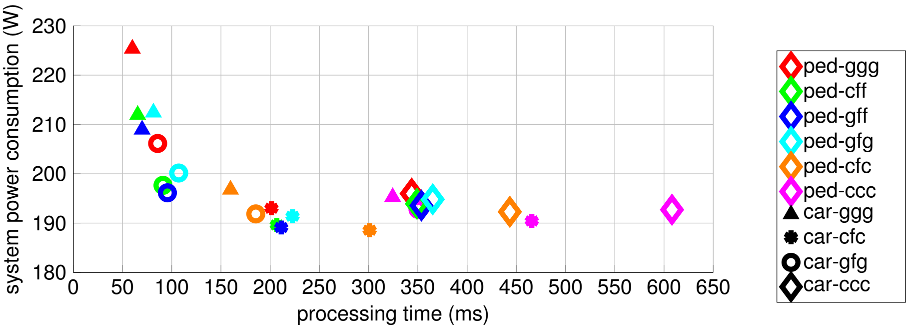
Detector performance: Power vs. Runtime



Idle baseline: 150W

1 point = ped + car + motion solution

Power vs. Runtime (detailed)



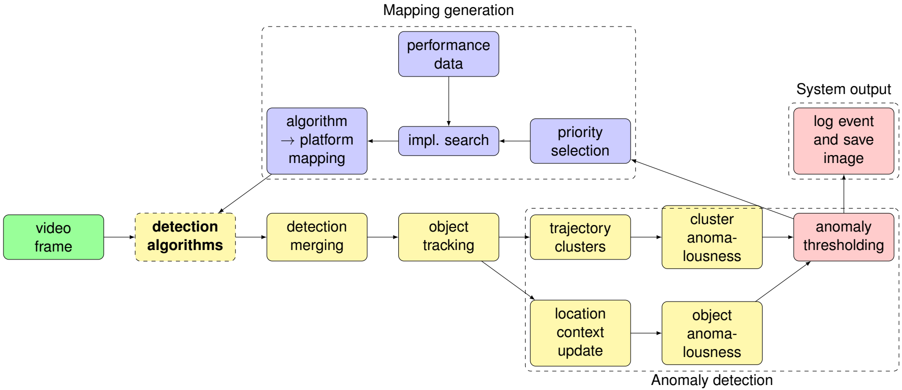
Idle baseline: 150W

Legend: 1 point = ped + car + motion solution
colour: pedestrian detector type
shape: car detector type

e.g. car-gfg is car detector using scale (GPU)

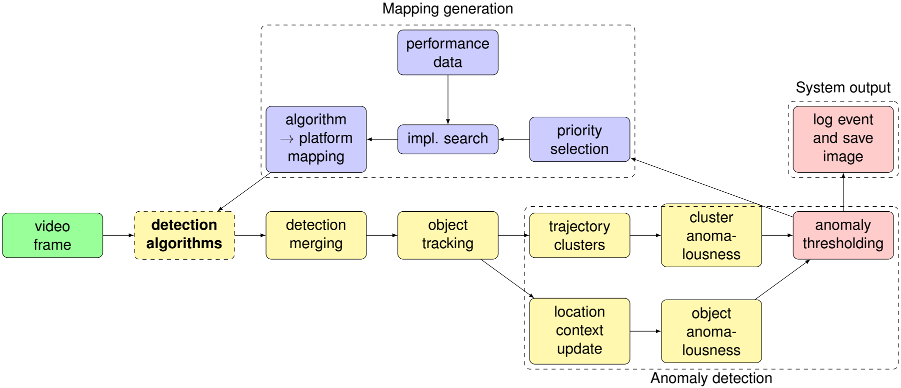
→ histogram (FPGA) → classify (GPU)

High-Level Anomaly Detection



Transform detections to ground plane and match to Kalman-filtered tracks

High-Level Anomaly Detection



Transform detections to ground plane and match to Kalman-filtered tracks

Anomaly Detection via Clustering

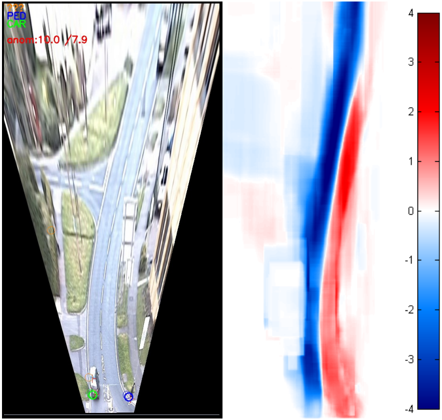
- Cluster tracks into trees of trajectories (Piciarelli & Foresti).
  - Trajectories which only one object travels along are unusual.
  - Trajectories that split from their frequently-travelled siblings are unusual.

- Define cluster anomaly measure U_C:

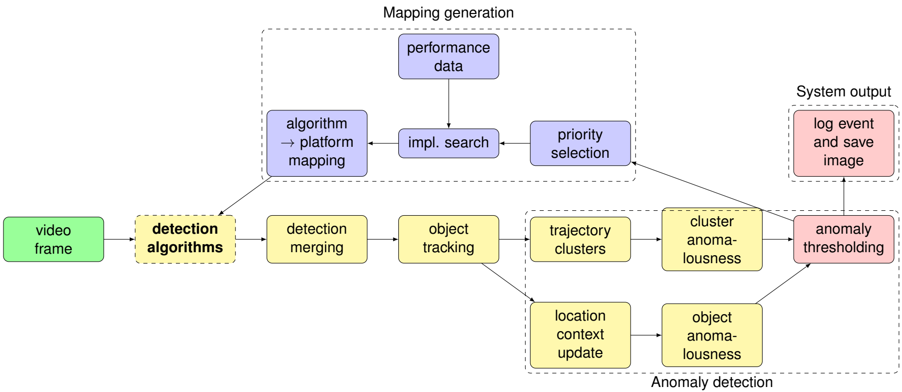
$$U_C(C_i) = \begin{cases} \dfrac{1}{1 + \mathit{transits}(C_i)}, & \text{for root node } C_i, \\ 1 - \dfrac{\mathit{transitions}(C_p \to C_i)}{\Sigma(\mathit{transitions}(C_p \to \text{all children of } C_p))}, & \text{for child node } C_i \text{ of parent } C_p. \end{cases}$$

Contextual Anomaly Detection

- “Events or movements not present in training data”.
- Learn object presence & mean velocity per-pixel $\bar{v}$ in x & y.
- $U_x \propto p(A|D) = \frac{p(D|A)p(A)}{p(D|A)p(A)+p(D|\bar{A})p(\bar{A})}$
- No info about $p(D|A)$ so set to constant.
- For x: $p(D|\bar{A}) = f(v_x, \bar{v_x})$
- Object anomaly $U_o = U_c + U_x + U_y$
- $U_{max} = max(all\ U_o)$

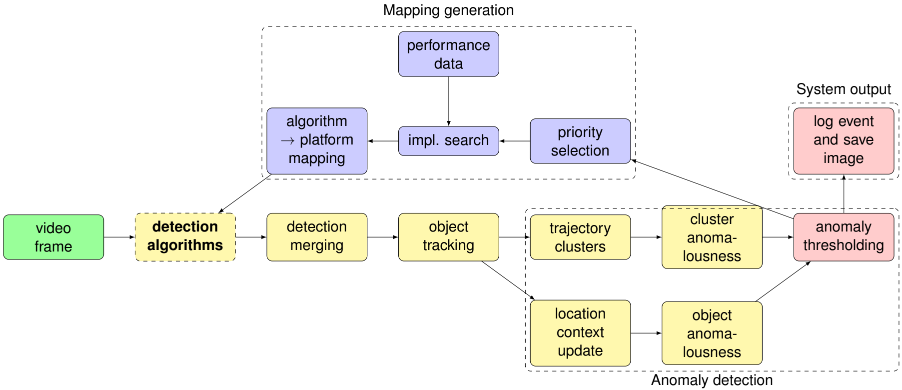


Anomaly Measure



Single number defines overall frame anomaly level; controls priority selection.

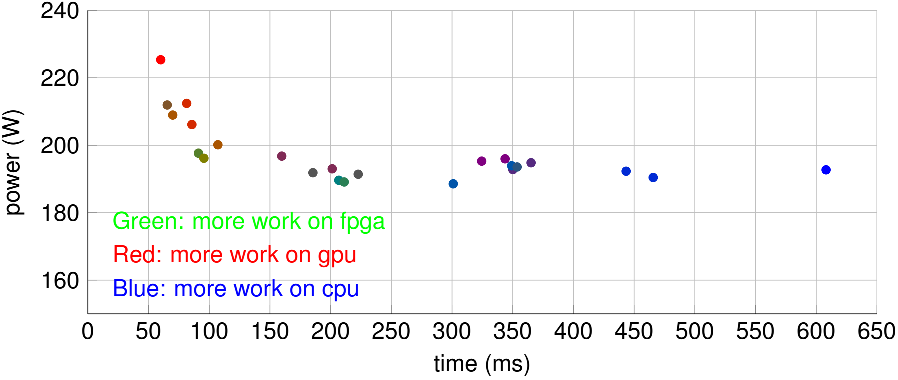
Anomaly Measure



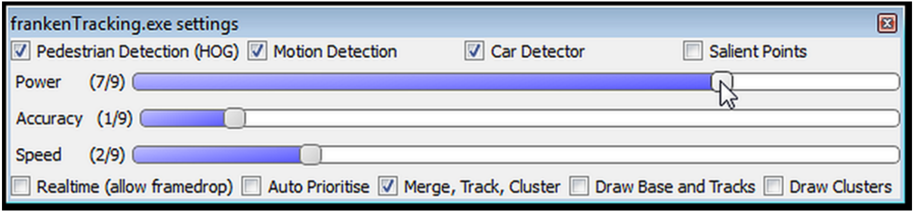
Single number defines overall frame anomaly level; controls priority selection.

Implementation Search

- Exhaustive search (ped, car, motion) = $6 \times 4 \times 1$ combinations
- Cost $C = w_p P + w_t t + w_\epsilon \epsilon$.
- P, t, ϵ known; set all w using anomaly level



Evaluation



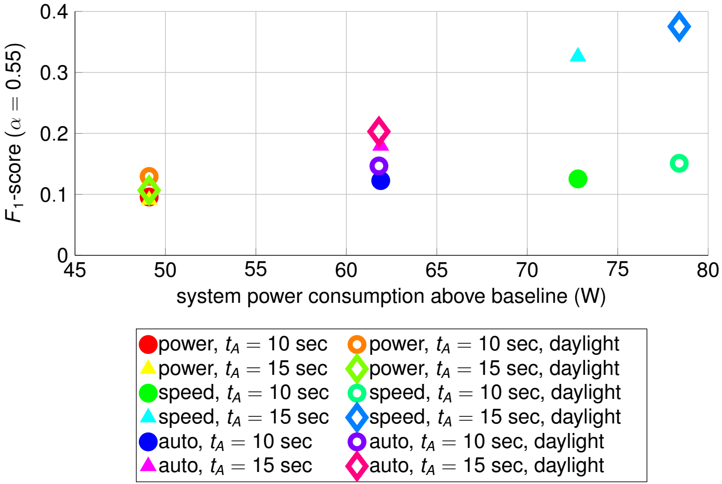
- Run next frame using chosen implementations
  - Choice of FPGA/ GPU/ CPU now task-driven, dynamic.
  - Skip frames ($\sim 50 - 75\%$) to keep realtime.
  - Processing time, system power (est.), log events.
  - Evaluate task-driven (auto) vs. fixed power or speed-optimised version.

Anomaly Detection Results

Prioritisation	True positives	False positives	False negatives	$p(\%)$	$r(\%)$
for $t_A = 10$ *seconds*					
power	4	29	23	12.1	14.8
speed	6	40	22	13.0	21.4
auto	6	42	22	12.5	21.4
for $t_A = 15$ *seconds*					
power	2	10	29	16.7	6.5
speed	8	8	23	**50.0**	**25.8**
auto	4	10	26	28.6	13.3

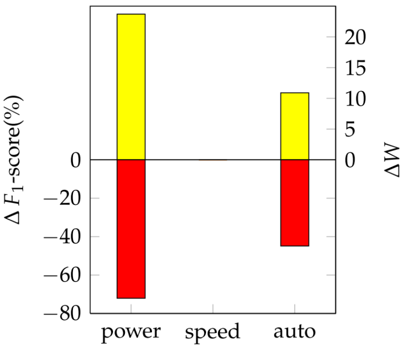
- Event detection relatively poor. Causes?
  - Poor detectors (high false negative/positive), occlusion, slow-moving traffic, sudden image gain changes, camera shake, anomaly detectors too simple to capture multi-vehicle events. . .

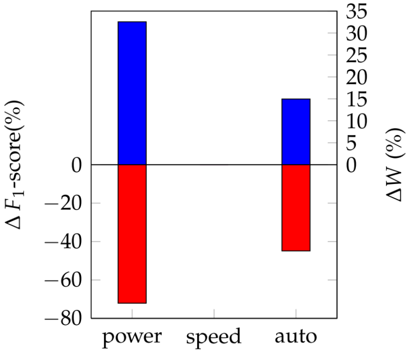
Accuracy vs. Power



$$F_1 = (\alpha + 1)rp/(r + \alpha p)$$

Relative tradeoffs





decrease in accuracy from *speed* (%)

decrease in P from *speed* (W)

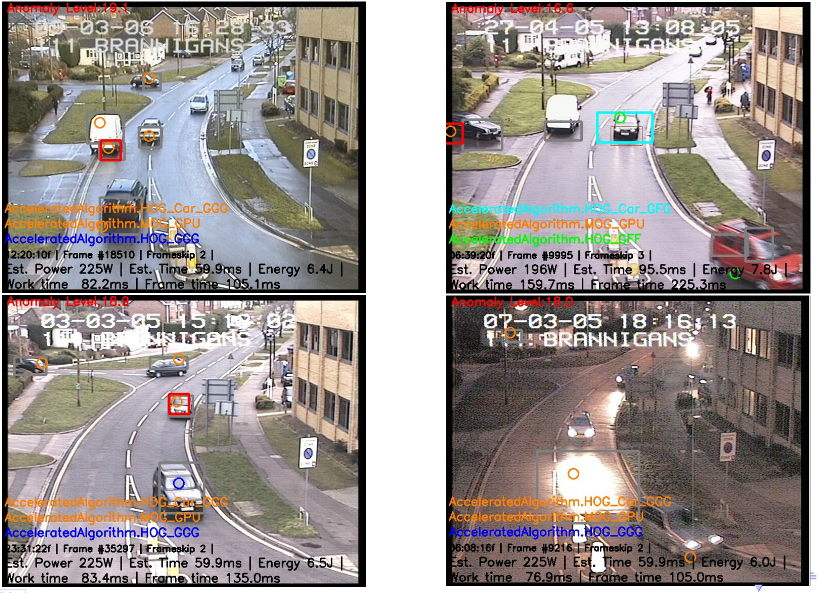
% *decrease* in accuracy from *speed*

% *decrease* in P from *speed*

Video


Detections
Anomaly Level:14.8
AcceleratedAlgorithm.HOG_Car_GGG
AcceleratedAlgorithm.MOG_CPU
AcceleratedAlgorithm.HOG_GGG
0:24:01f | Frame #0601 | Frameskip 2 |
Est. Power 225W | Est. Time 59.9ms | Energy 6.4J |
Work time 81.8ms | Frame time 104.7ms
(x=556, y=18)
frankenTracking.exe settings
Pedestrian Detection (HOG)
Motion Detection
Car Detector
Salient Points
Power (1/9)
Accuracy (0/9)
Speed (9/9)
Realtime (allow framedrop)
Auto Prioritise
Merge, Track, Cluster
Draw Base and Tracks
Draw Clusters

Hits and Misses

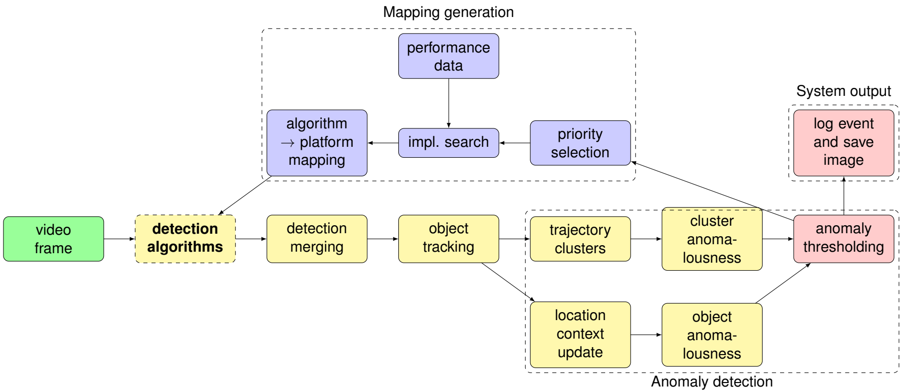


Summary

- Dynamic selection of implementations between different hardware platforms (FPGA, GPU, CPU) is possible, in response to changing user requests or scene conditions.
- Scene-controlled mapping selection offers reduced power consumption at some cost in accuracy.

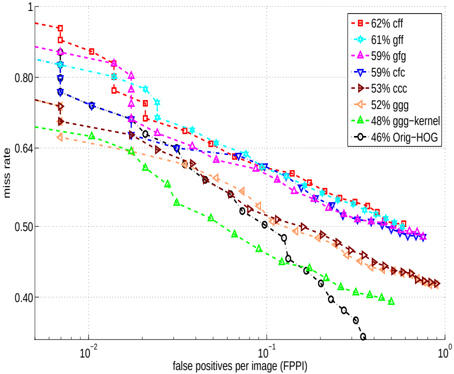
- Future work
  - Mobile chips (lower power)
  - Improved detector algorithms

Questions?


Mapping generation
performance data
algorithm → platform mapping
impl. search
priority selection
System output
log event and save image
video frame
detection algorithms
detection merging
object tracking
trajectory clusters
cluster anomalousness
anomaly thresholding
location context update
object anomalousness
Anomaly detection

Detector performance: Accuracy

Pedestrian:



Car: