Hardware Accelerated Vehicle Detection Using Computer Vision for a Dynamic Traffic System

Nestor Michael C. Tiglao, PhD

Joint work with Patrick Celon, Timothy Chua, Paul Ilaga, and Jethro Limjoco

Objectives

- To develop and implement a **dynamic closed-loop traffic management system covering two intersections**
- Measure the **throughput of the implemented system** and **compare it** against the **throughput of a static scheduling system**
- Implement the Computer Vision learning algorithm on **RasPi** and **FPGA** for **benchmarking**

Computer Vision

Implement a hardware-accelerated Computer Vision algorithm to detect vehicles in traffic

+

Computer Networks

A network based approach to communication between intersections

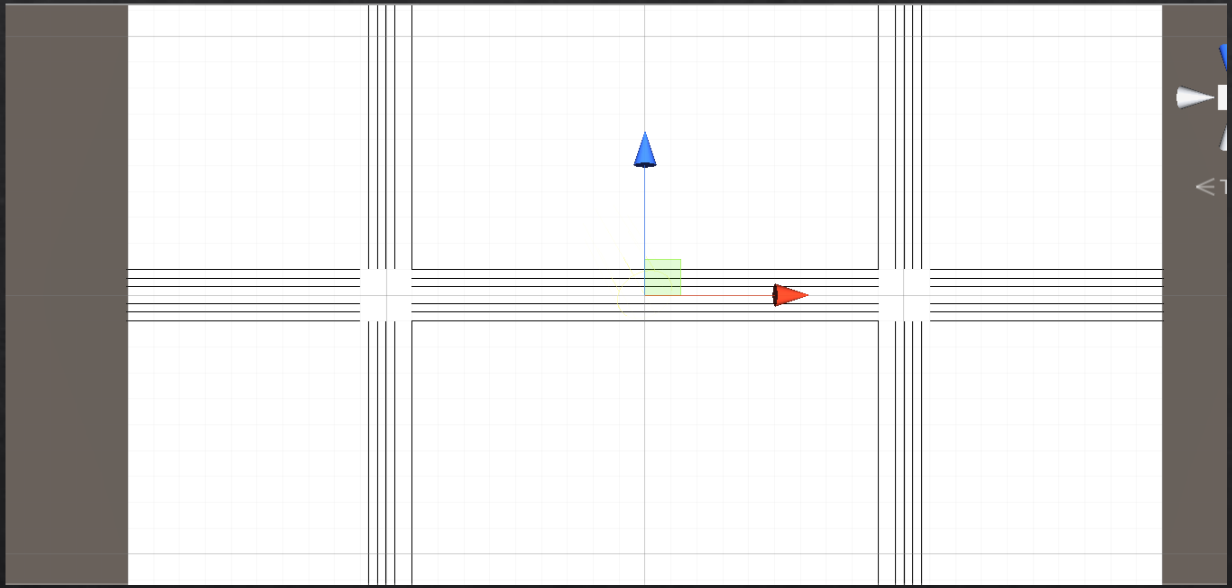
+

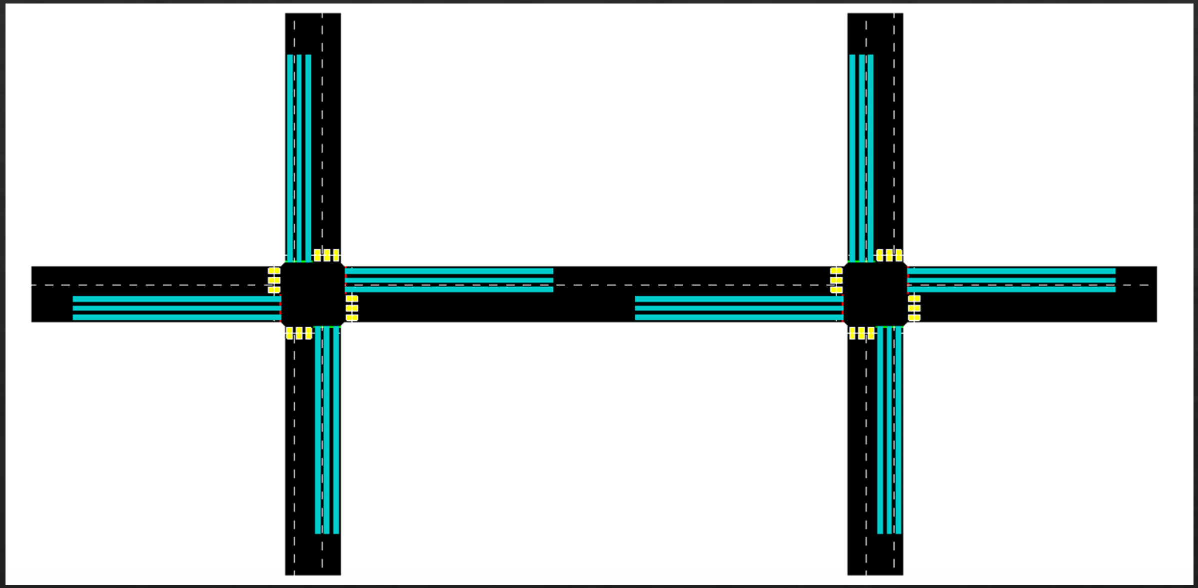
Dynamic Traffic Control

A demand-based green light allocation algorithm (ITLC)

Our System

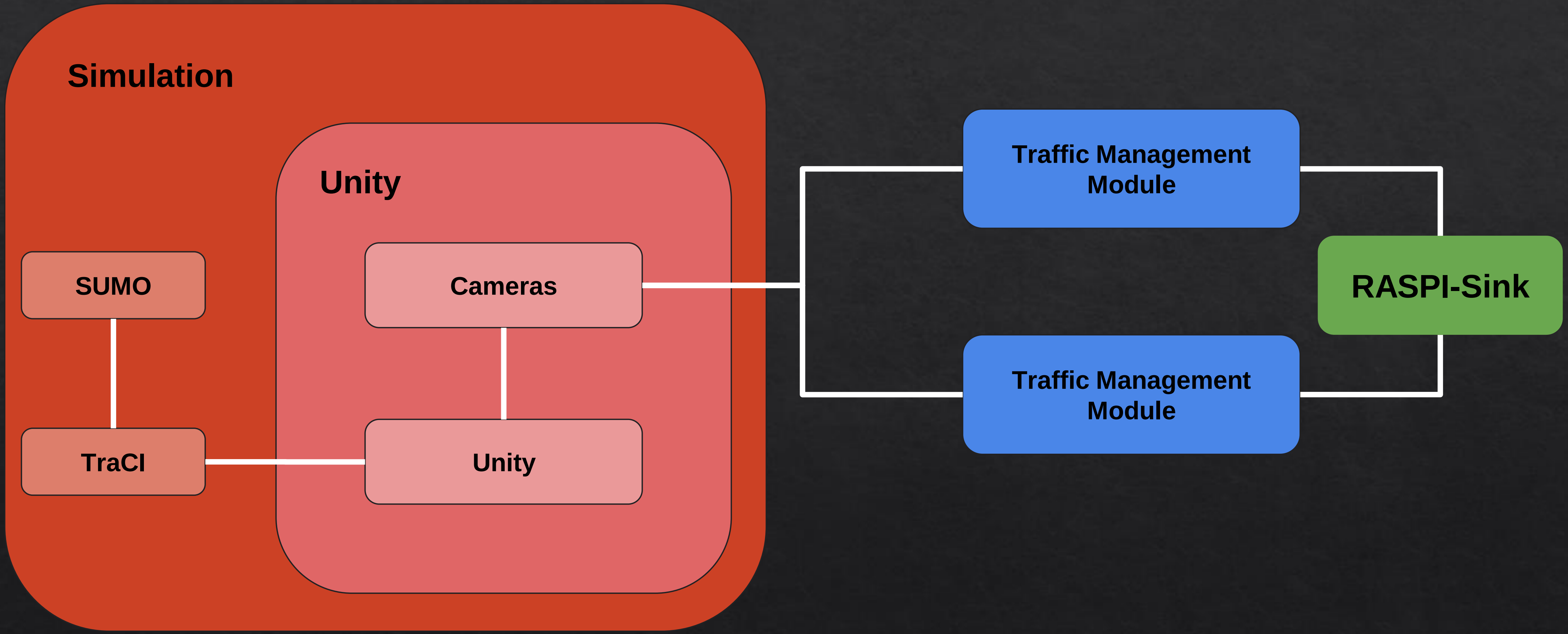
Simulation Setup



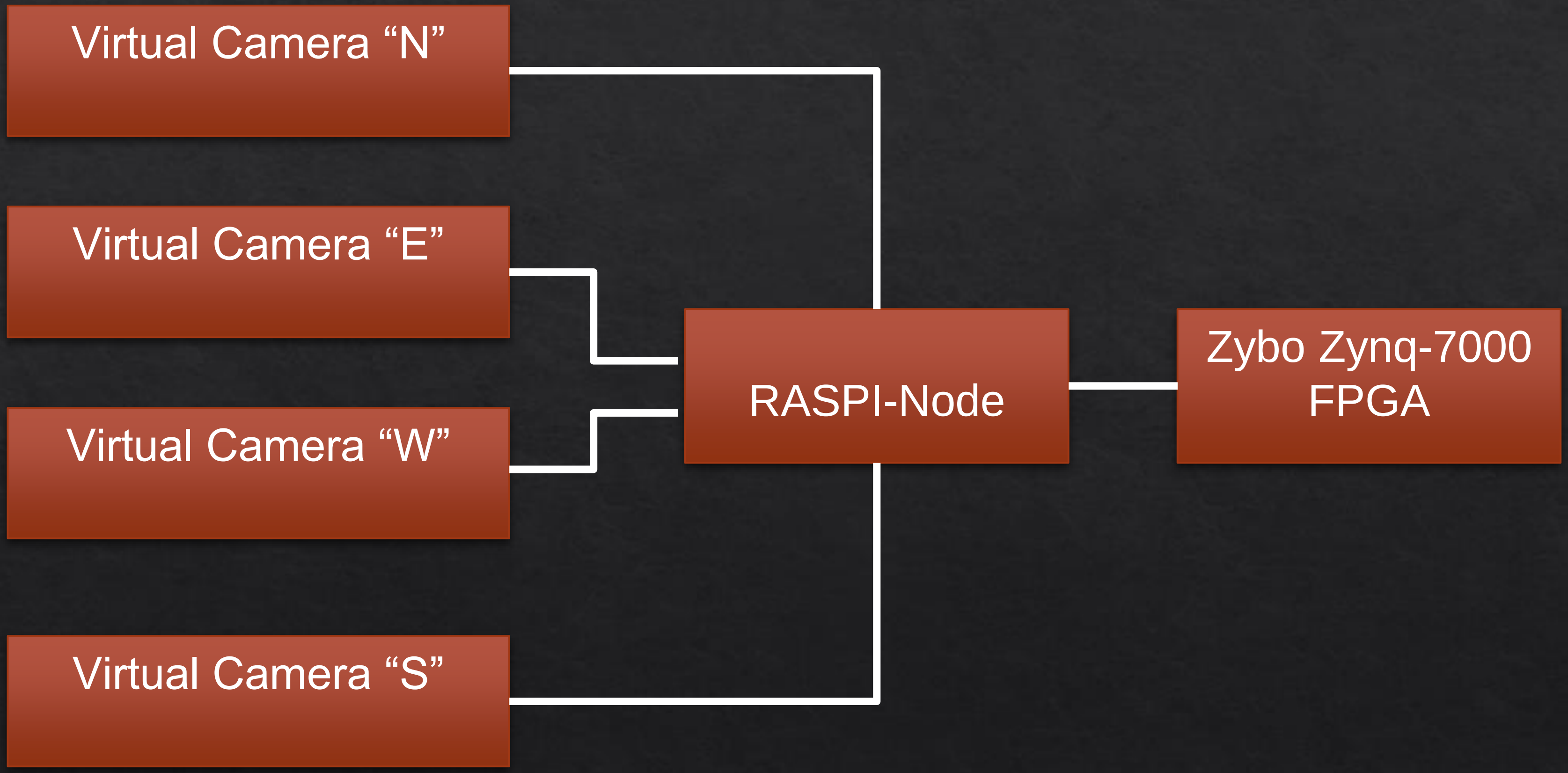


SUMO is an open source, highly portable, microscopic and continuous traffic simulation package designed to handle large road networks.
http://sumo.sourceforge.net/

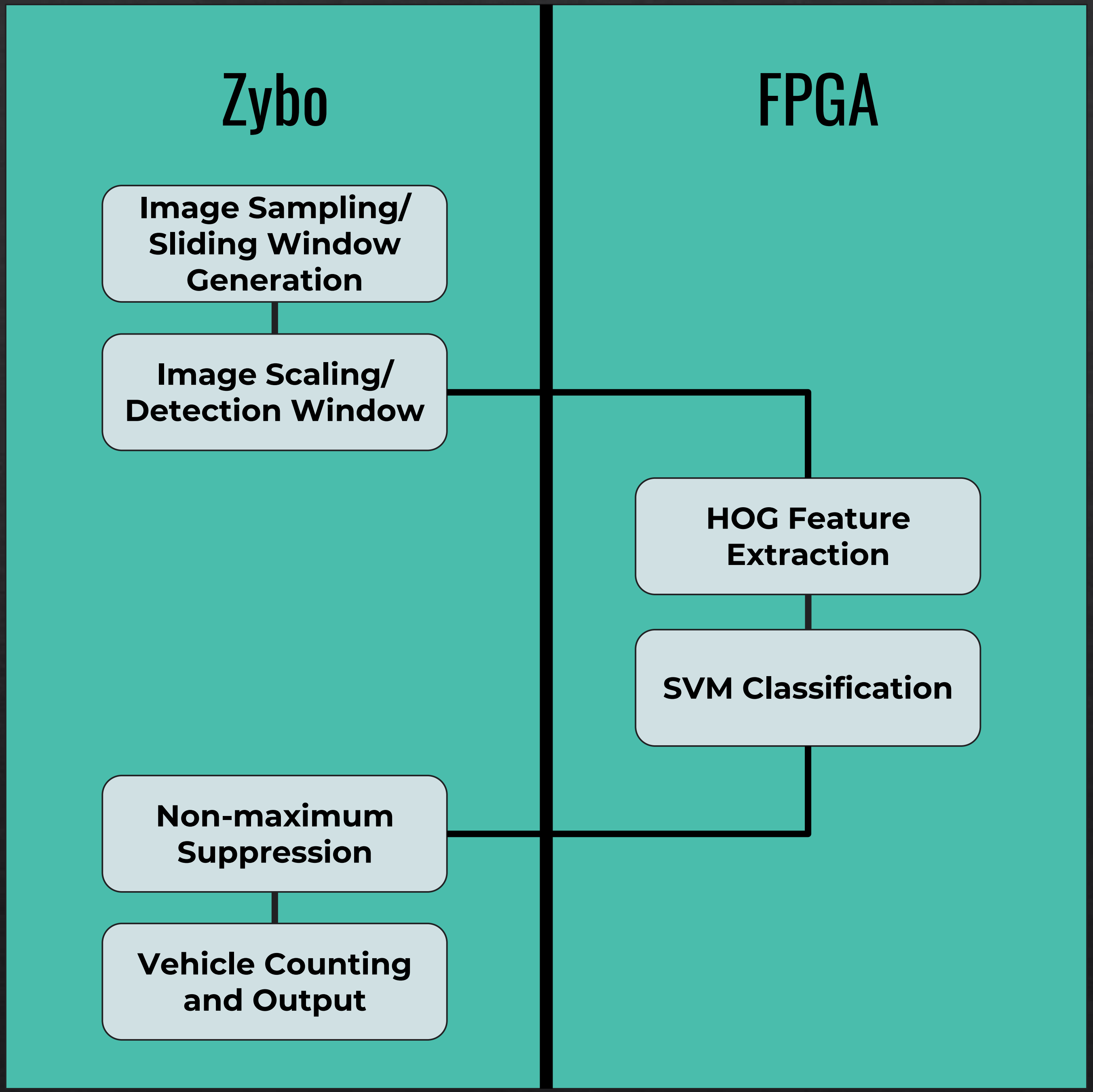
Representation of the Complete System



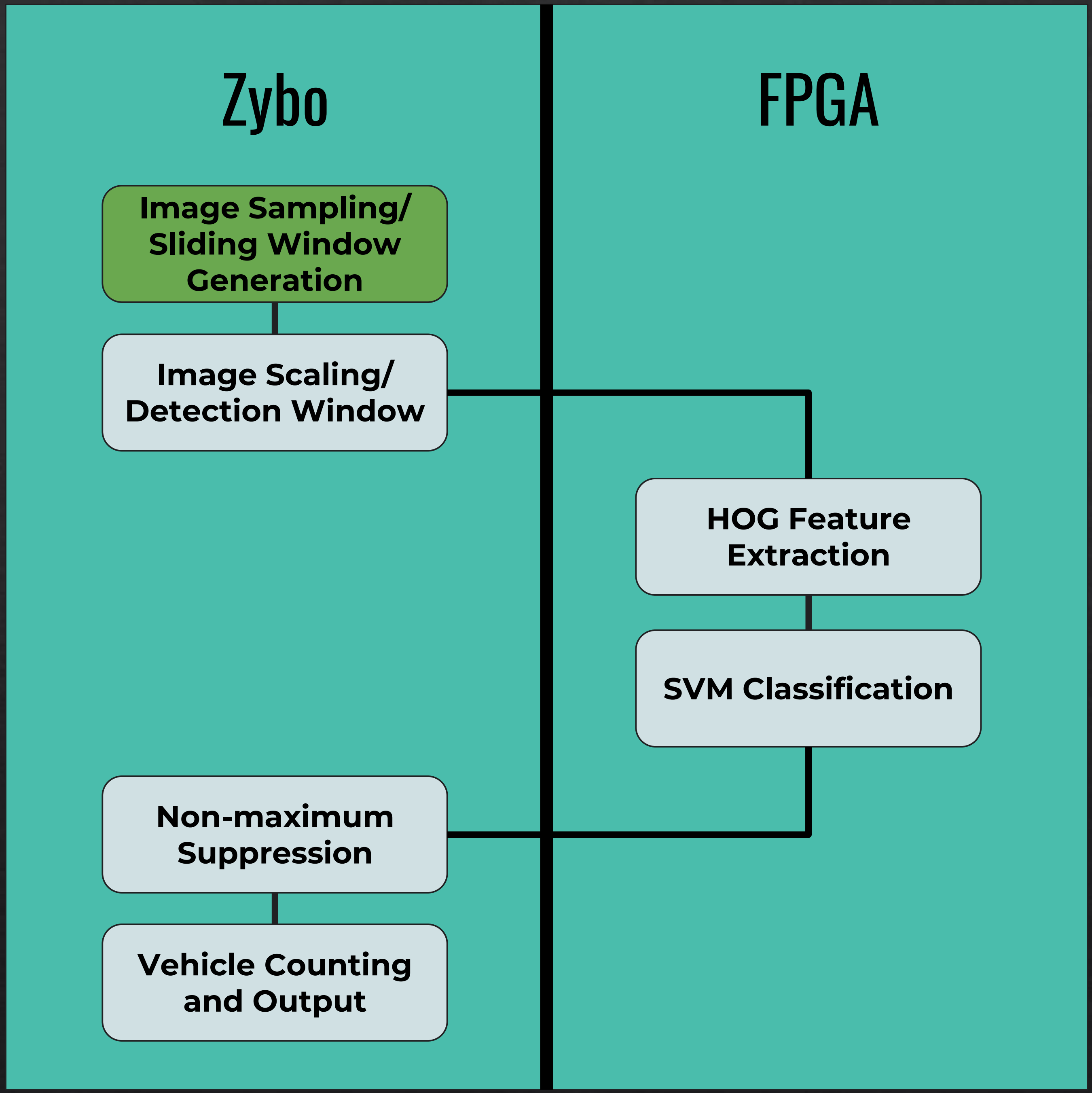
Traffic Management Module



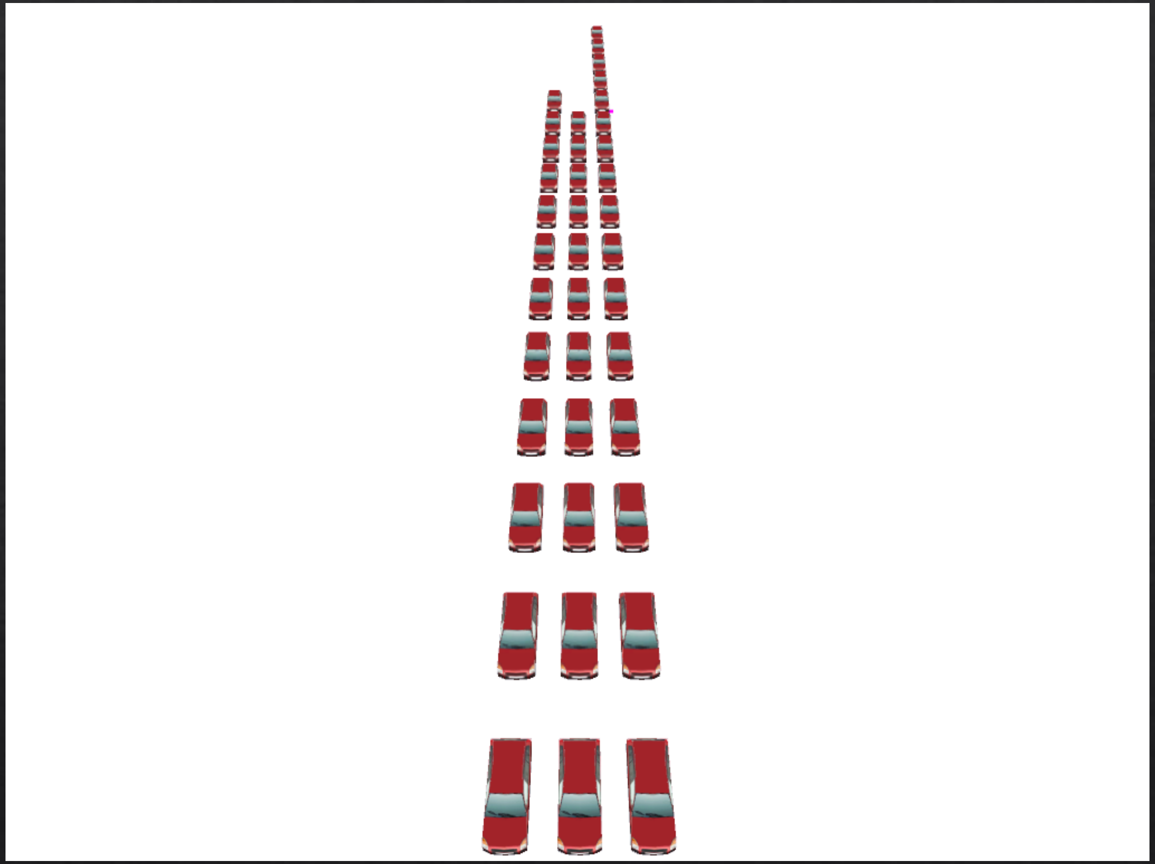
Interplay of the FPGA and the on-board Zybo SoC/SBC



Interplay of the FPGA and the on-board Zybo SoC/SBC

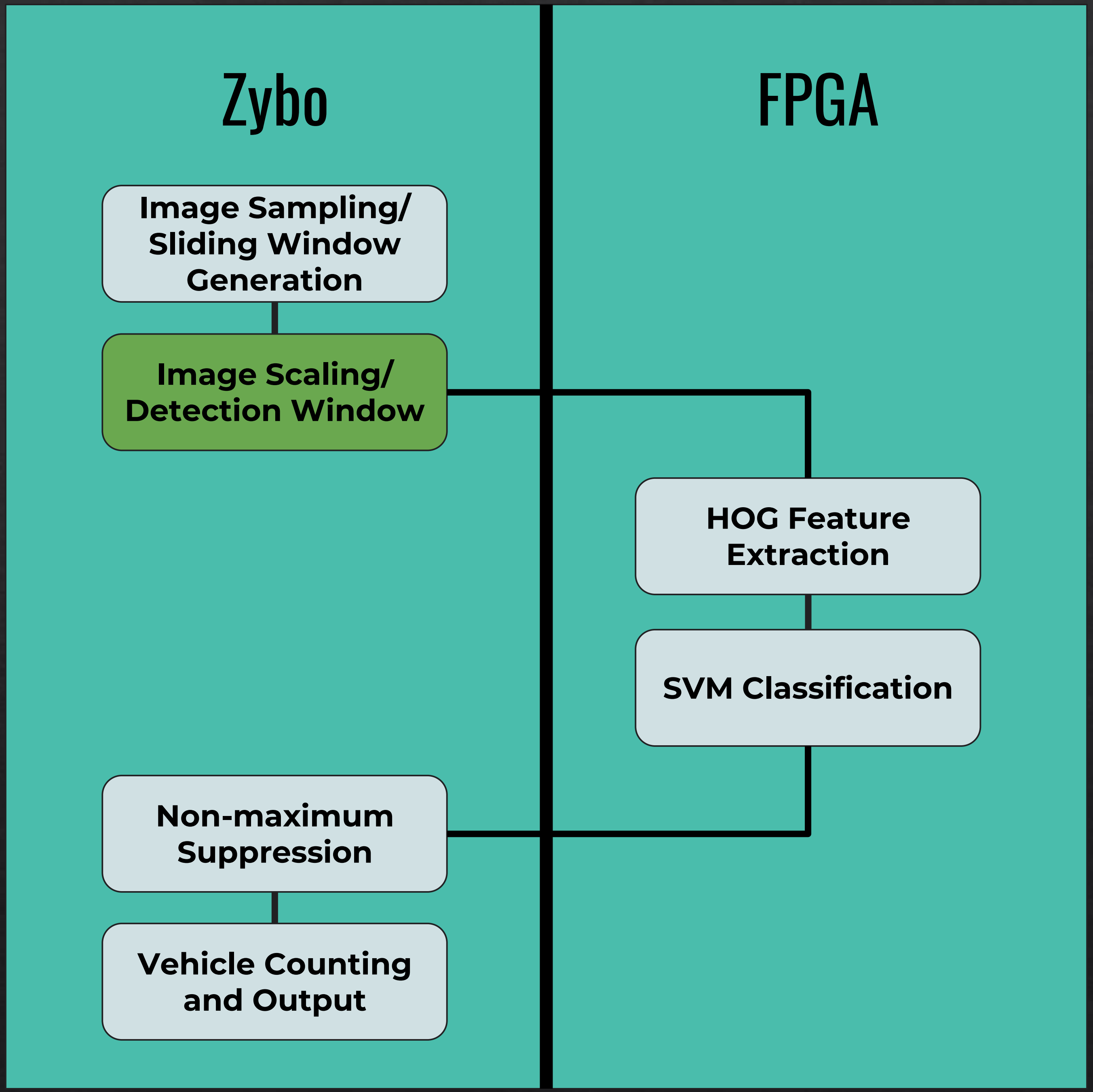


An in-game view of a virtual camera



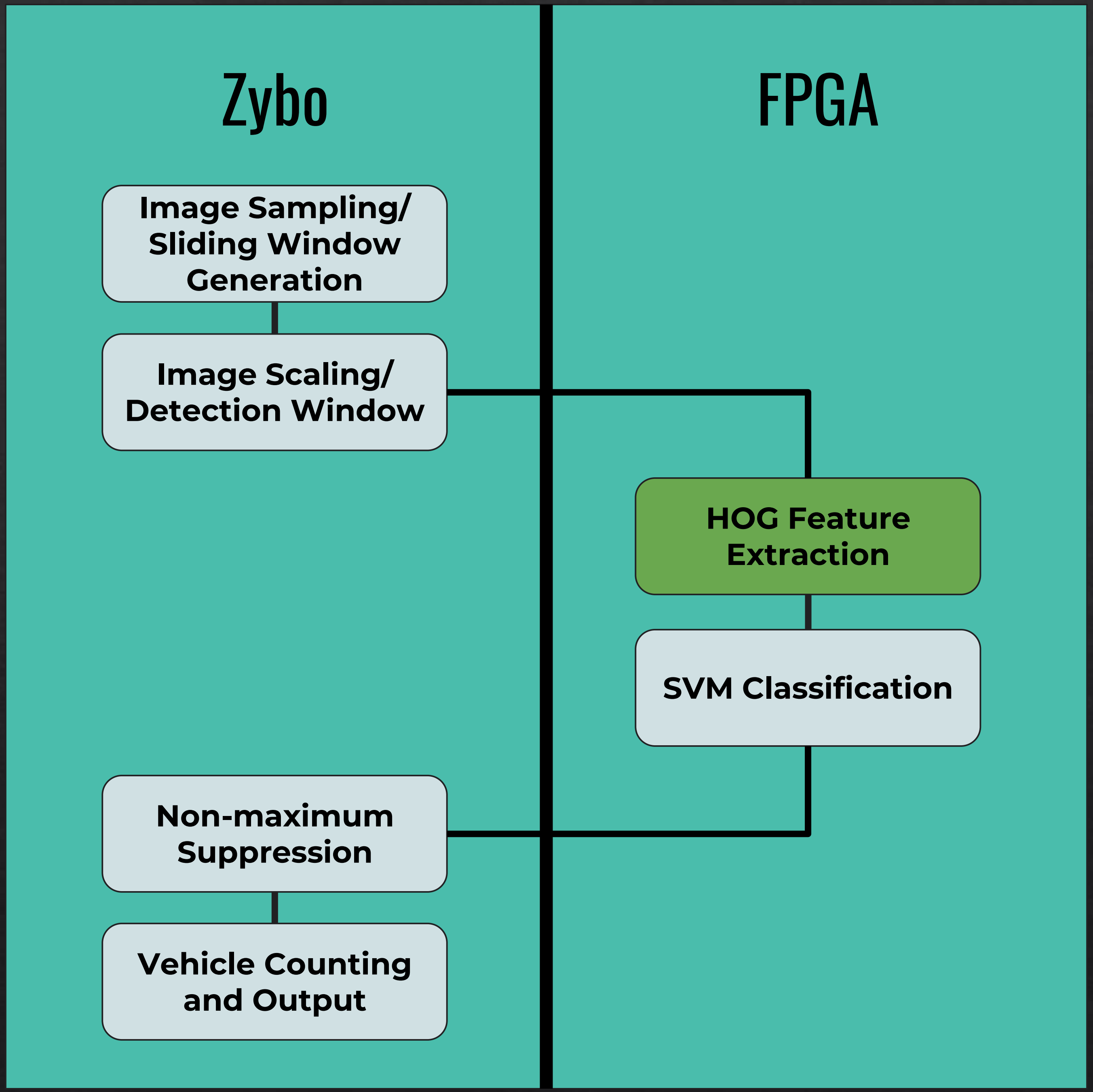
Camera view rendered in Unity

Interplay of the FPGA and the on-board Zybo SoC/SBC

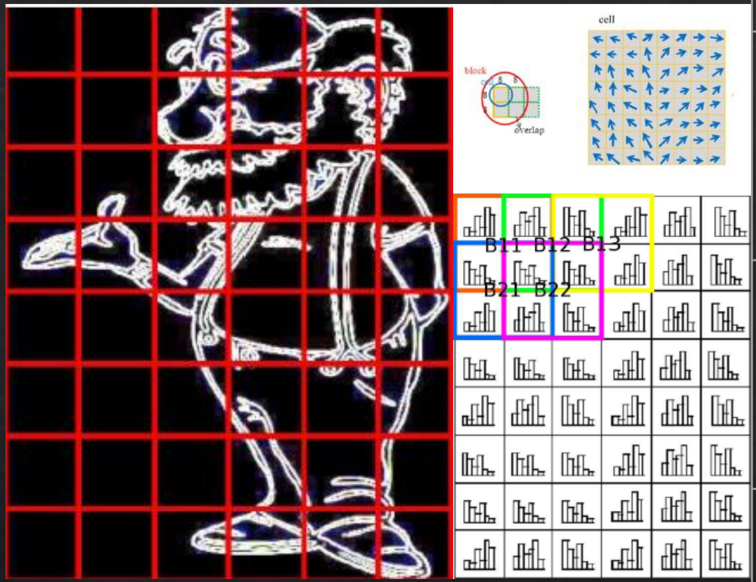


Sliding Window Optimization

Interplay of the FPGA and the on-board Zybo SoC/SBC



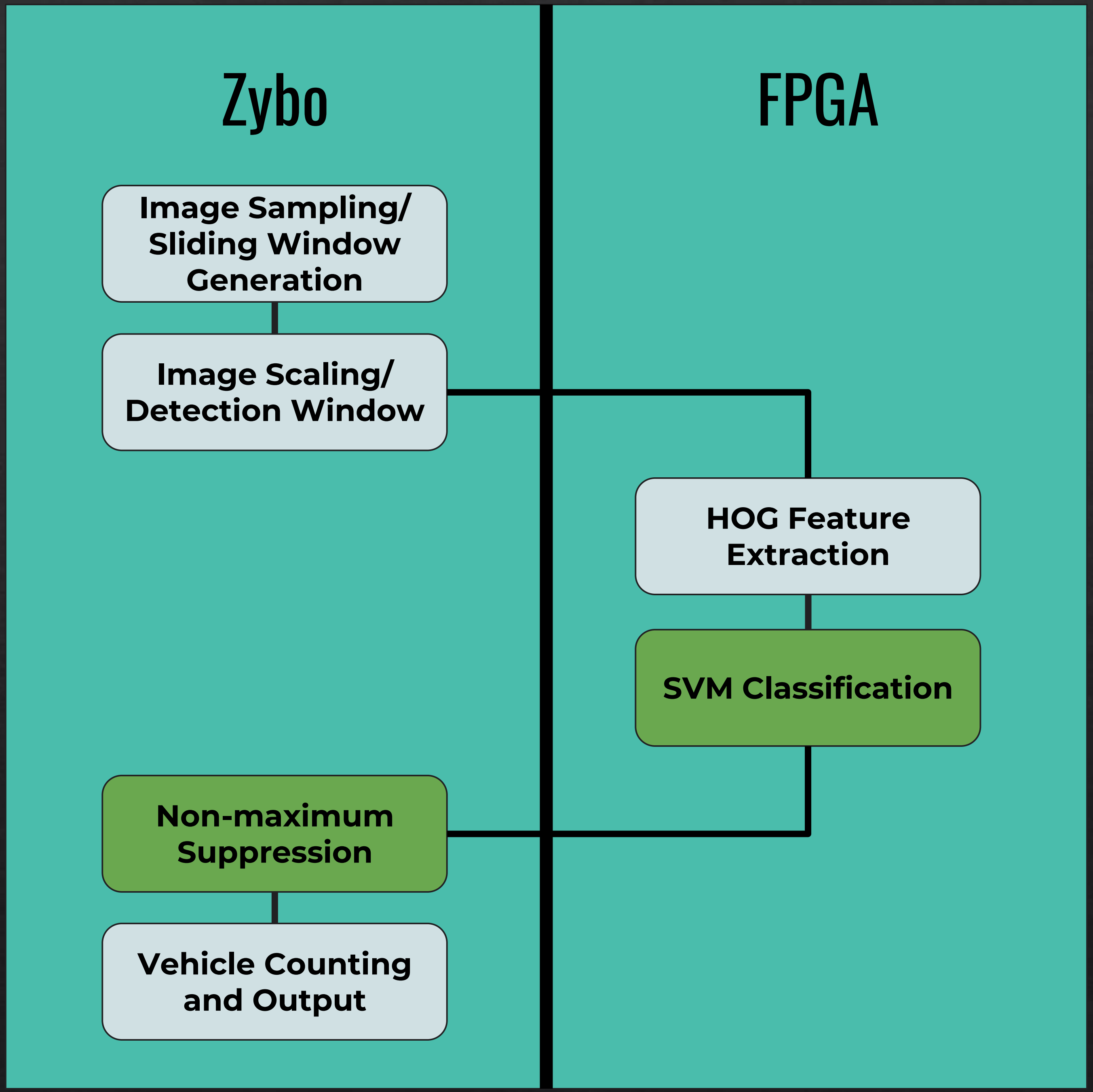
Hardware Approximation of HOG



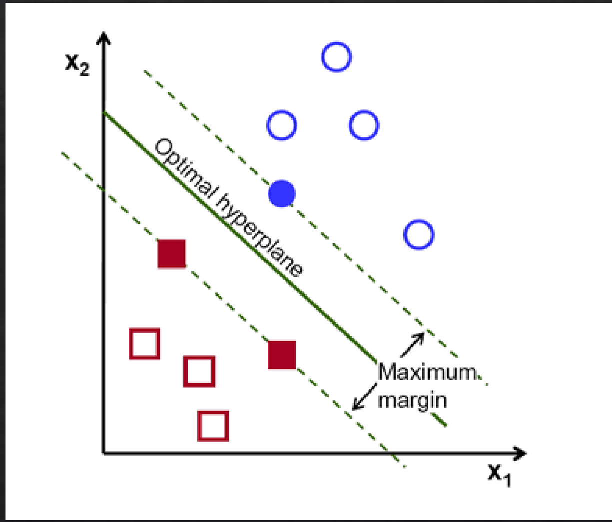
Operation	Software Implementation	Hardware Implementation[1]
Gradient Magnitude	Root Mean Square $mag_{x,y} = \sqrt{dX_{x,y}^2 + dY_{x,y}^2}$	$max\left(a - (a >> 3) + (b >> 1),\ a\right)$ $a = max(dx, dy),\ b = min(dx, dy)$
Orientation Binning	Arctangent $angle_{x,y} = atan\left(\frac{dY_{x,y}}{dX_{x,y}}\right)$	$dx \times \tan\theta_{\mathrm{i}} \leq dy \leq dx \times \tan\theta_{\mathrm{i+1}}$
Block Normalization	Inverse square root $\frac{1}{\sqrt{x}}$	$y_d\left(\frac{3 - x\ {y_d}^2}{2}\right)$ $y_d = Decimal\{\ (x\_{\mathrm{IEEE754}} >> 1) - 0x5F3759DF\ \}$

[1] P. Y. Chen, C. C. Huang, C. Y. Lien and Y. H. Tsai, "An efficient hardware implementation of HOG feature extraction for human detection," in IEEE Transactions on Intelligent Transportation Systems 15.2, 2014, pp. 656-662.

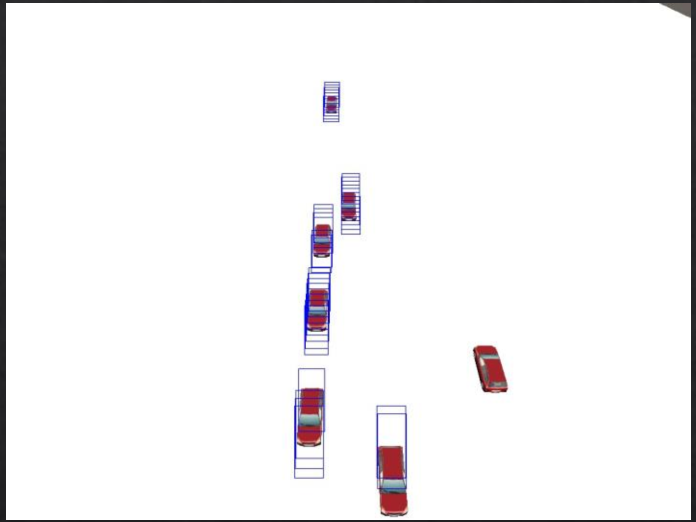
Interplay of the FPGA and the on-board Zybo SoC/SBC

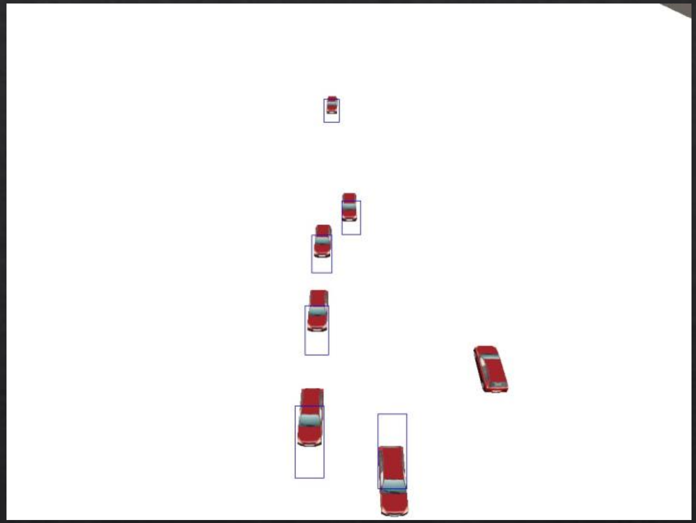


Classification to Suppression



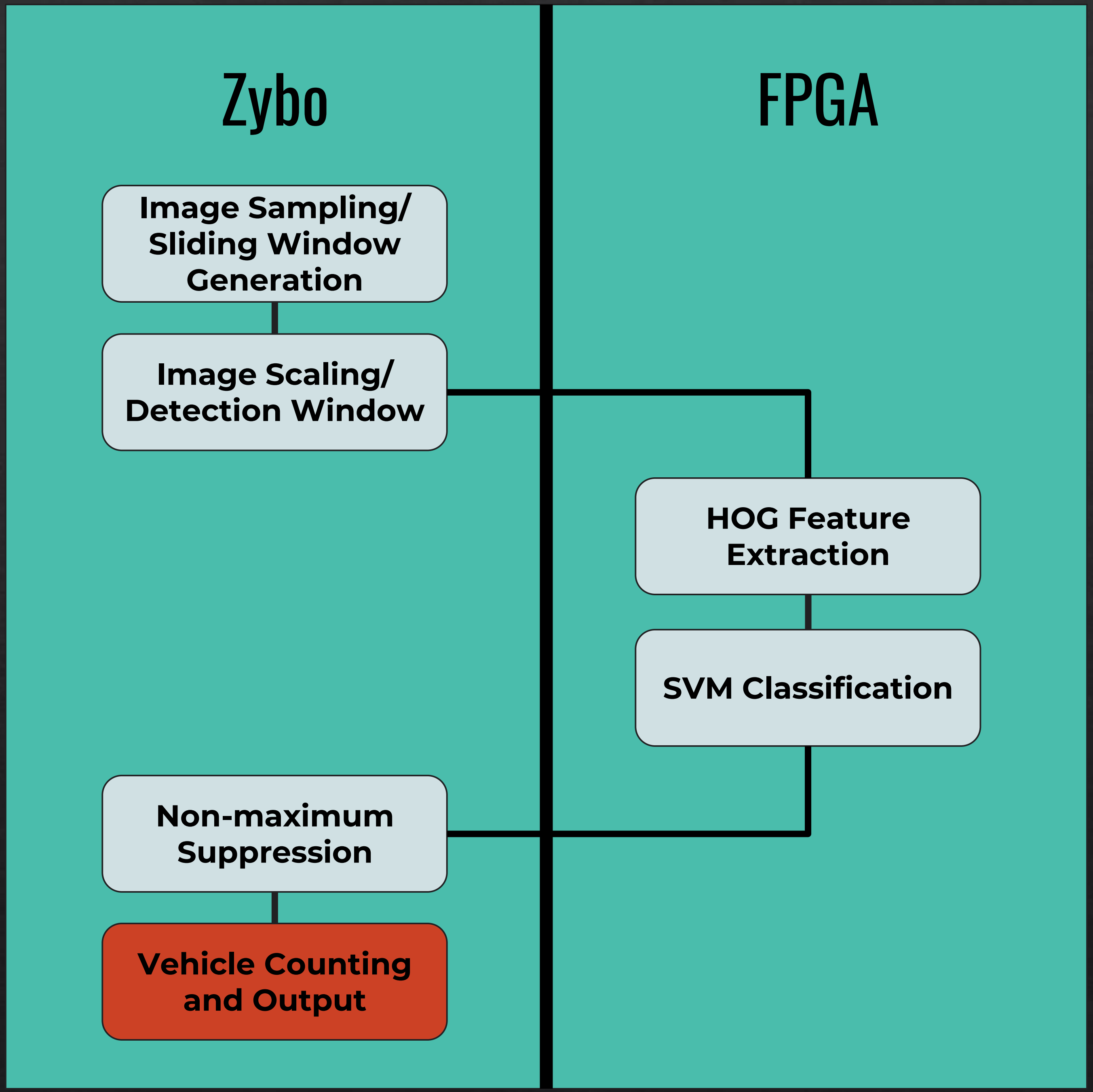
SVM



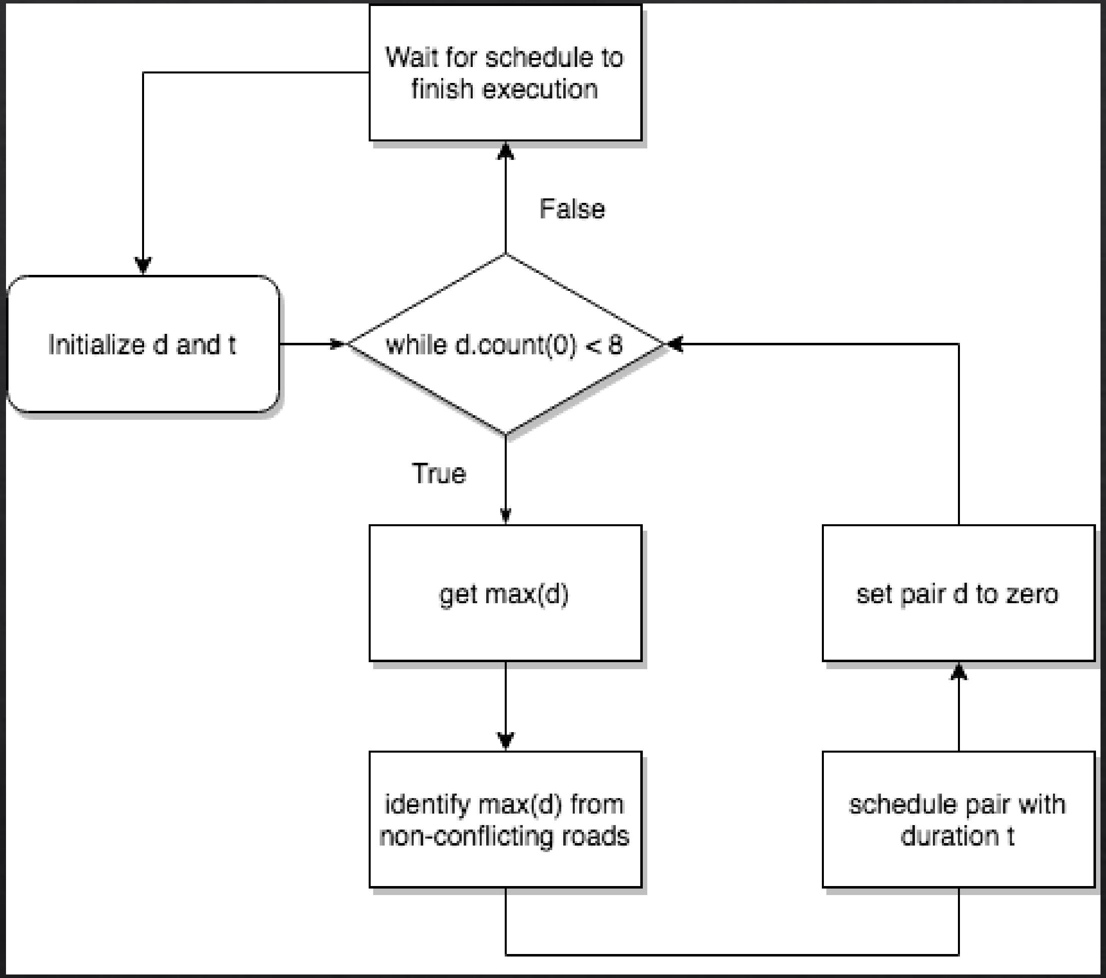


Non-Maximum Suppression

Interplay of the FPGA and the on-board Zybo SoC/SBC



Intelligent Traffic Light Scheduling Algorithm



Reference: M. B. Younes and A. Boukerche, “An intelligent traffic light scheduling algorithm through vanets,” in 39th Annual IEEE Conference on Local Computer Networks Workshops, pp. 637– 642, Sept 2014.

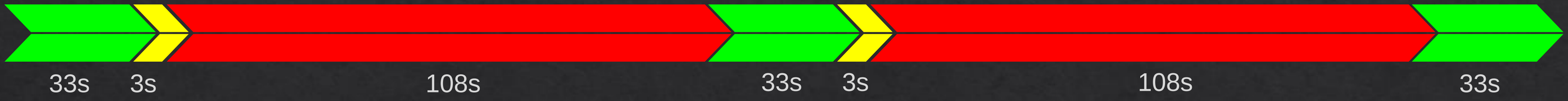
Timeline of Signals on a Single Road
Terminal Output

```
('rrrrrGGGGGrrrrrrrrrr', 1100.0, 3, 2)
('rrrrryyyyyrrrrrrrrrr', 1112.0, 3, 2)
('rrrrGGrrrrrrrrGGrrrr', 1115.0, 1, 5)
('rrrryyrrrrrrrryyrrrr', 1124.0, 1, 5)
('rrrrrrrrrrGGGGGrrrrr', 1127.0, 4, 5)
('rrrrrrrrrryyyyyrrrrr', 1136.0, 4, 5)
('rrrrrrrrrrrrrrrGGGGG', 1139.0, 7, 6)
('rrrrrrrrrrrrrrryyyyy', 1145.0, 7, 6)
('rrrrrGGGrrrrrrrGGGrr', 1149.0, 2, 6)
('rrrrryyyrrrrrrryyyrr', 1161.0, 2, 6)
('rrrrGGrrrrrrrrGGrrrr', 1164.0, 1, 5)
('rrrryyrrrrrrrryyrrrr', 1176.0, 1, 5)
('rrrrrrrrrrGGGGGrrrrr', 1179.0, 4, 5)
('rrrrrrrrrryyyyyrrrrr', 1188.0, 4, 5)
('rrrrrGGGGGrrrrrrrrrr', 1191.0, 3, 2)
('rrrrryyyyyrrrrrrrrrr', 1203.0, 3, 2)
('rrrrrrrrrrGGGGGrrrrr', 1207.0, 4, 5)
('rrrrrrrrrryyyyyrrrrr', 1222.0, 4, 5)
('rrrrrrrrrrrrrrrGGGGG', 1225.0, 6, 7)
('rrrrrrrrrrrrrrryyyyy', 1231.0, 6, 7)
('GGGGGrrrrrrrrrrrrrrr', 1234.0, 1, 0)
('yyyyyrrrrrrrrrrrrrrr', 1240.0, 1, 0)
('rrrrrGGGGGrrrrrrrrrr', 1243.0, 3, 2)
('rrrrryyyyyrrrrrrrrrr', 1249.0, 3, 2)
```

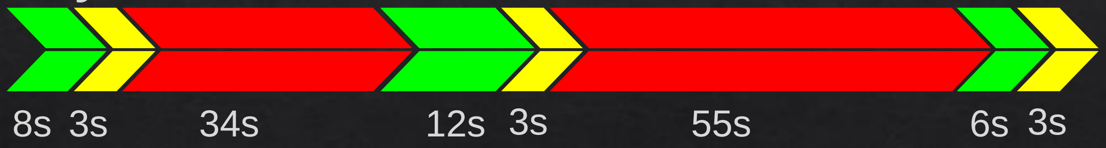
Reference: P. S. Chakraborty, P. R. Sinha, and A. Tiwari, “Real time optimized traffic management algorithm for intelligent transportation systems,” in 2015 IEEE International Conference on Computational Intelligence Communication Technology, pp. 744–749, Feb 2015.

Timeline of Signals on a Single Road
Graphical Representation (3 green lights)

Static



Dynamic



NOT TO SCALE

RasPi is slightly more robust than FPGA

		FPGA			RASPI		
Video Name	Description	Accuracy	F1-score	MCC	Accuracy	F1-score	MCC
MVI_39031	Direct front view, daytime, low occlusion	94.5%	0.97	0.946	96.8%	0.98	0.96
MVI_39211		88.3%	0.84	0.84	95.3%	0.85	0.85
MVI_39311		88.5%	0.802	0.795	91%	0.88	0.87
MMDA_3017	EDSA - Aurora Intersection, high occlusion	68.7%	0.51	0.52	71.8%	0.59	0.60
MMDA_3100	Roxas - EDSA Intersection, medium occlusion	80.5%	0.598	0.60	78.7%	0.62	0.62
MMDA_4079	Roxas Boulevard - Quirino Avenue intersection, medium Occlusion	68.99%	0.656	0.65	64.4%	0.66	0.65

Camera placement matters!

DETRAC Dataset[1]
94% Accuracy

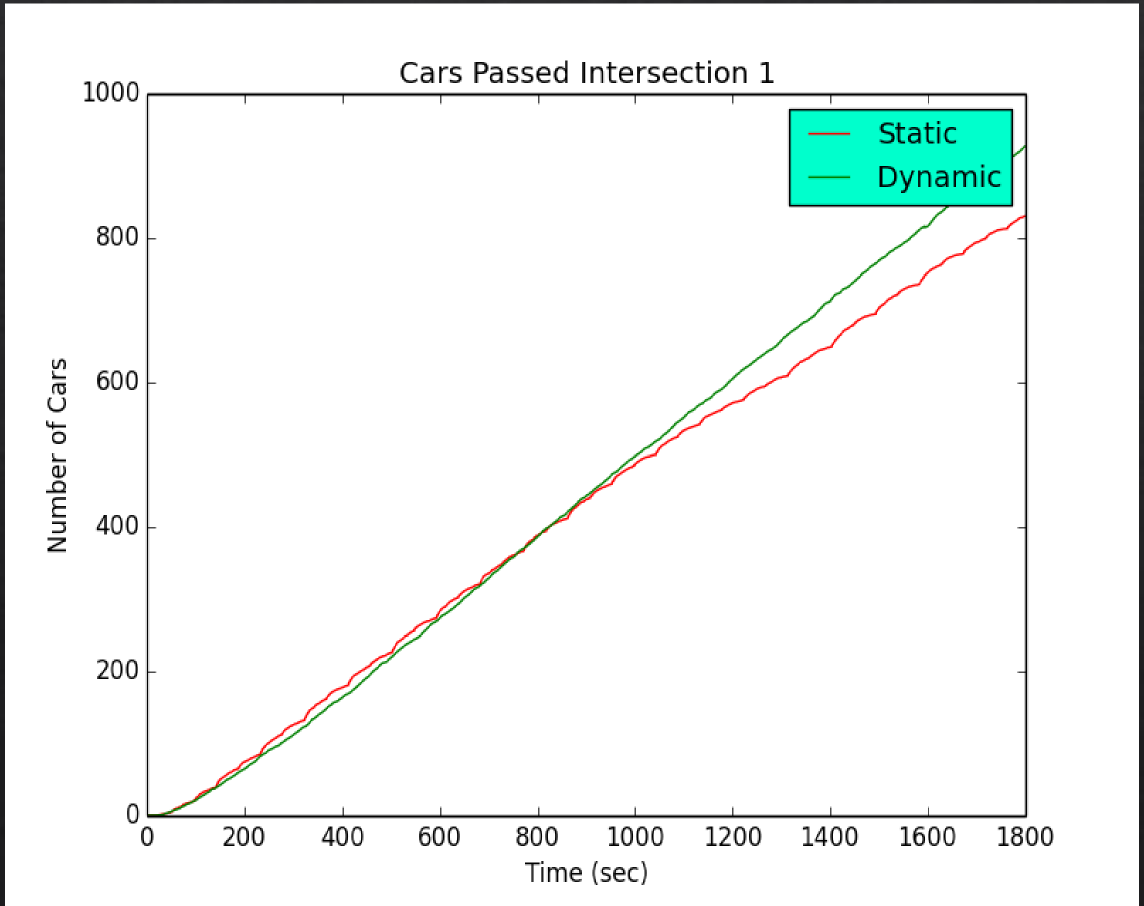
Unusable MMDA Dataset
41% Accuracy

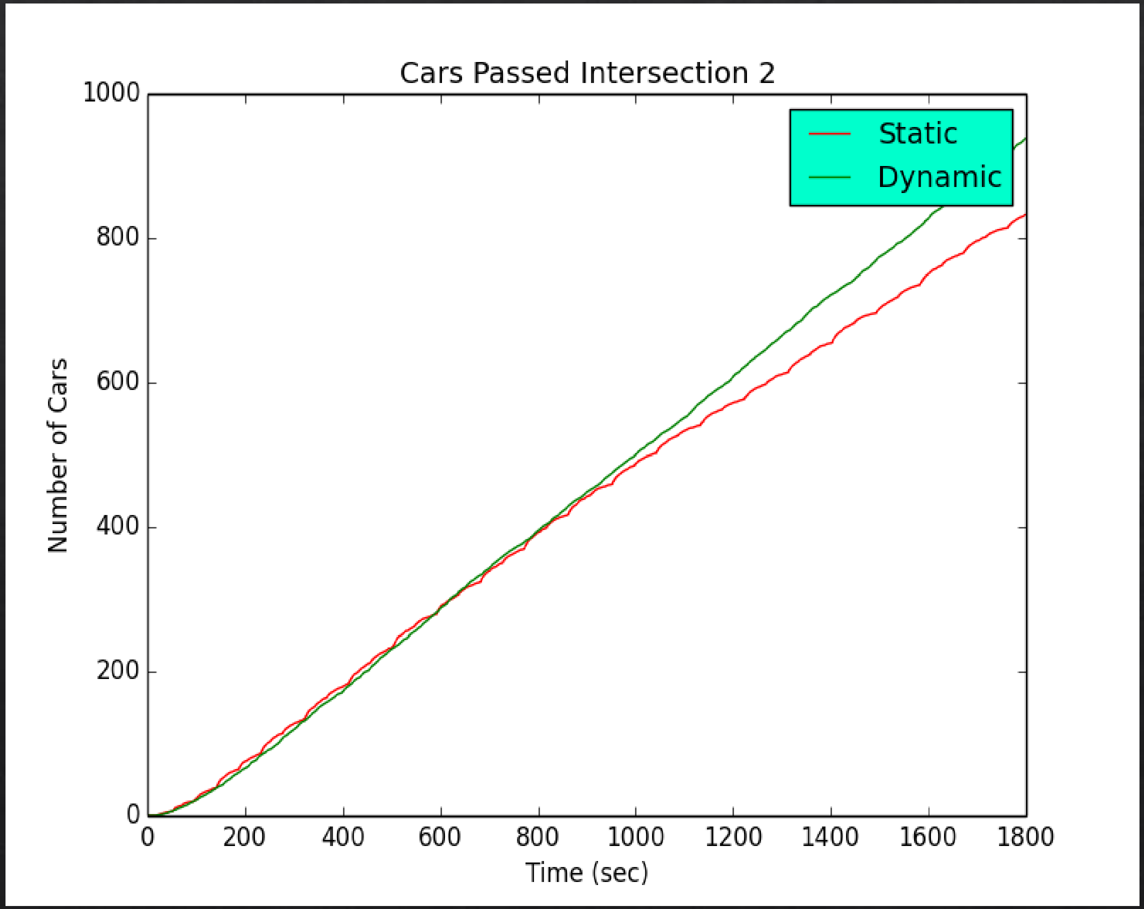
Reference: L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi, J. Lim, M. Yang and S. Lyu, "UA-DETRAC: A New Benchmark and Protocol for Multi-Object Detection and Tracking", Arxiv.org, 2018. [Online]. Available: https://arxiv.org/abs/1511.04136. [Accessed: 23- May- 2018].

FPGA performs 13x faster than the RasPi

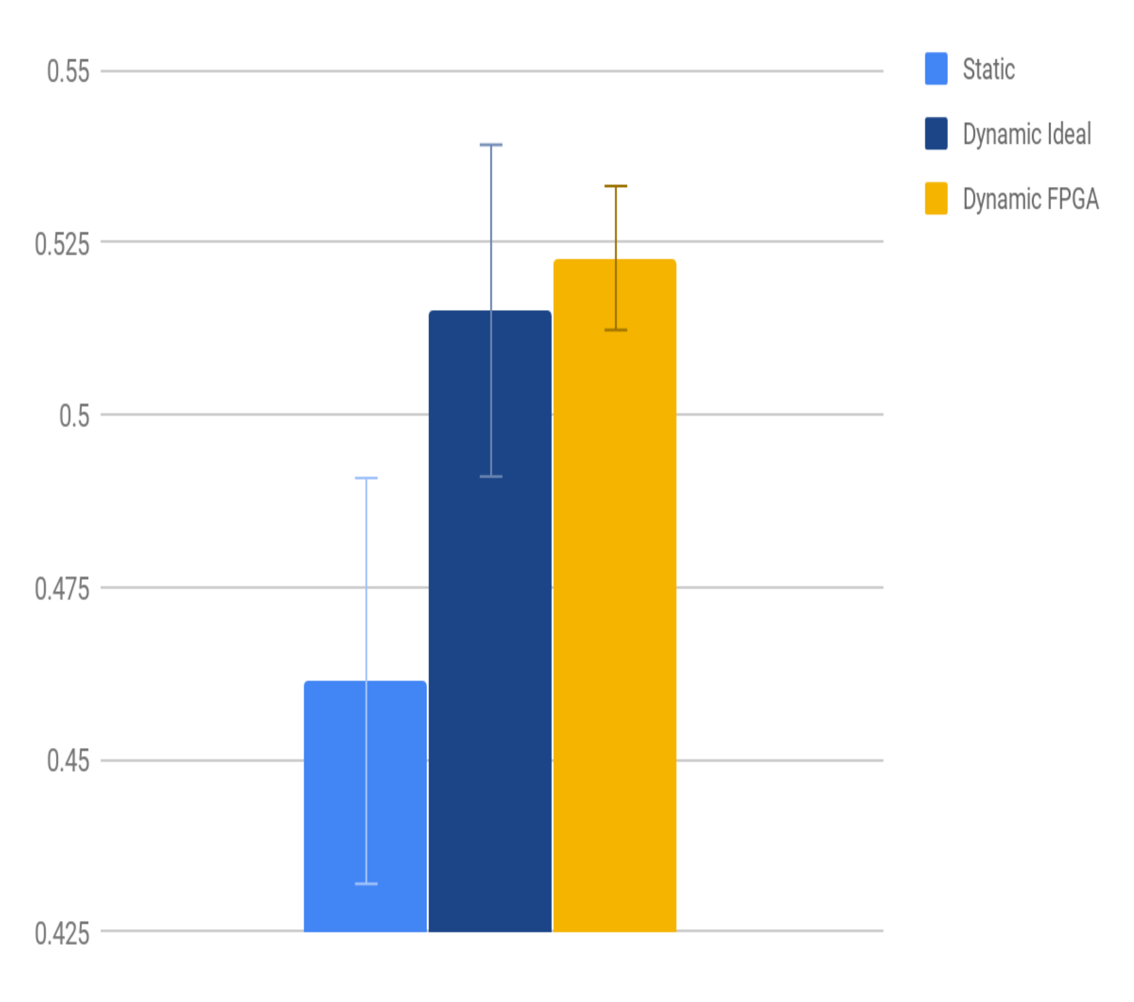
Video Name	Number of Pictures	FPGA Proc Time (sec)	RASPI Proc Time (sec)	Speed Up
MVI_39031	2568	0.26615258	3.6345863	13.65602469
MVI_39211	1703	0.18806675	2.362984754	12.56460667
MVI_39311	1583	0.16284416	2.243037446	13.77413501
MMDA_3017	13366	1.33831585	19.16865903	14.32297084
MMDA_3100	10502	1.05670986	15.10151064	14.29106627
MMDA_4079	11492	1.15280146	16.48849331	14.30297744

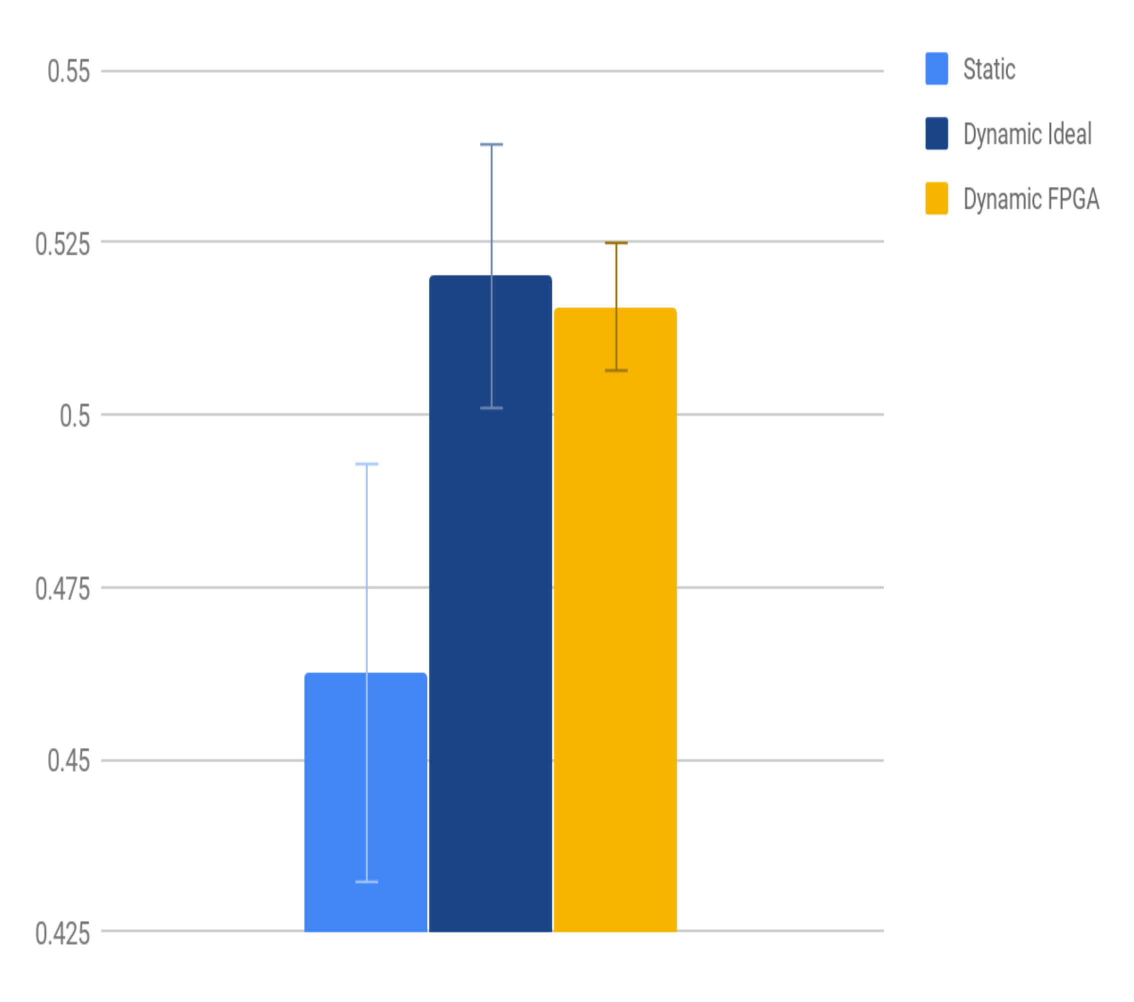
Traffic System Output
Static vs Dynamic





Average Throughput Improvements





Average Throughput Improvements

	Intersection 1	Intersection 2
An Ideal System	11.63%	12.43%
Our System	13.29%	11.48%

Conclusion

- The dynamic scheduling system was able to adjust to heavy traffic and **provide an increase in throughput** in comparison to the static scheduling system.

- Furthermore, the FPGA was shown to provide a **significant speedup** in performing the HOG + SVM algorithm in comparison to the purely software implementation from OpenCV.

Future Work

- This system can be **expanded** to cover more than **two intersections.** Also, this system should be tested through **real life deployment** to increase system robustness on different scenarios.

- To further increase speed up, **sliding window generation can also be implemented in the FPGA**. Also, implementing partial reconfiguration on the FPGA would enable the FPGA to adapt to various lane orientations.