GPU-based pedestrian detection for autonomous driving

V. Campmany[1,2,3], S. Silva[1,2,3], J.C. Moure[1,2], T. Espinosa[1,2], D. Vázquez[1,3], A. M. López[1,3]

[1]Autonomous University of Barcelona   [2]Computer Architecture & Operating Systems
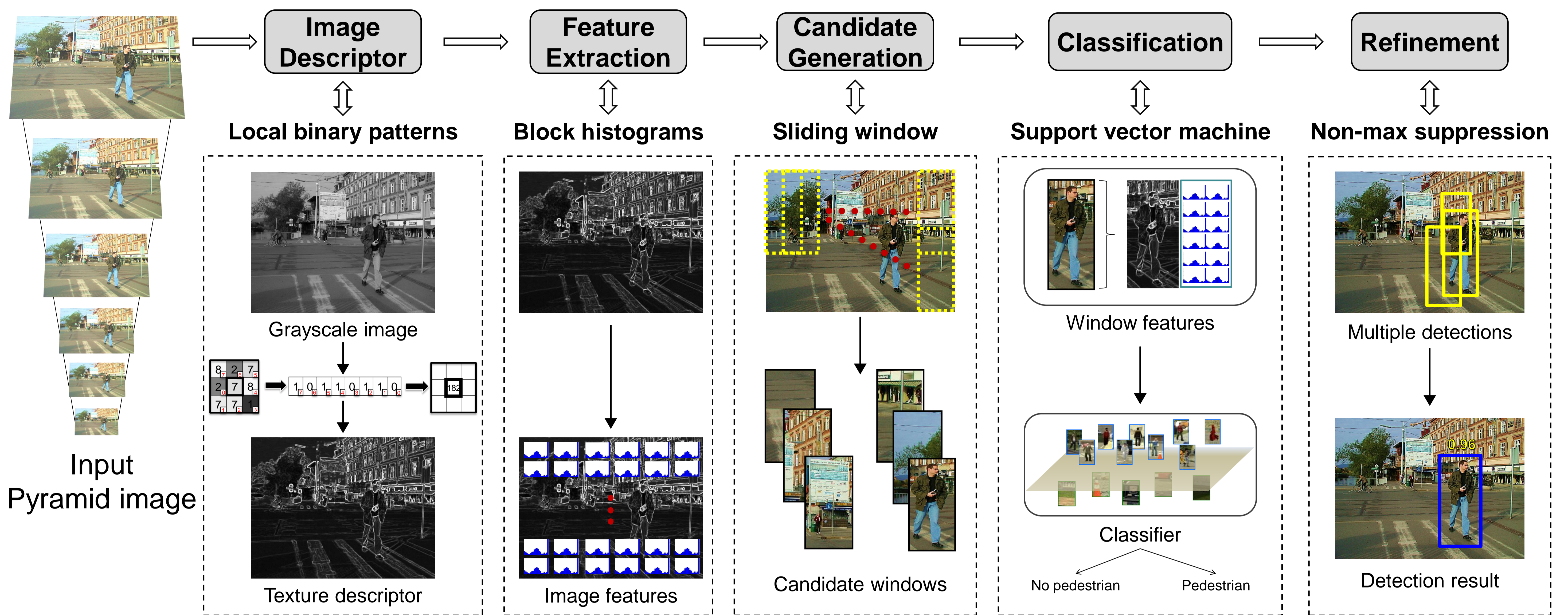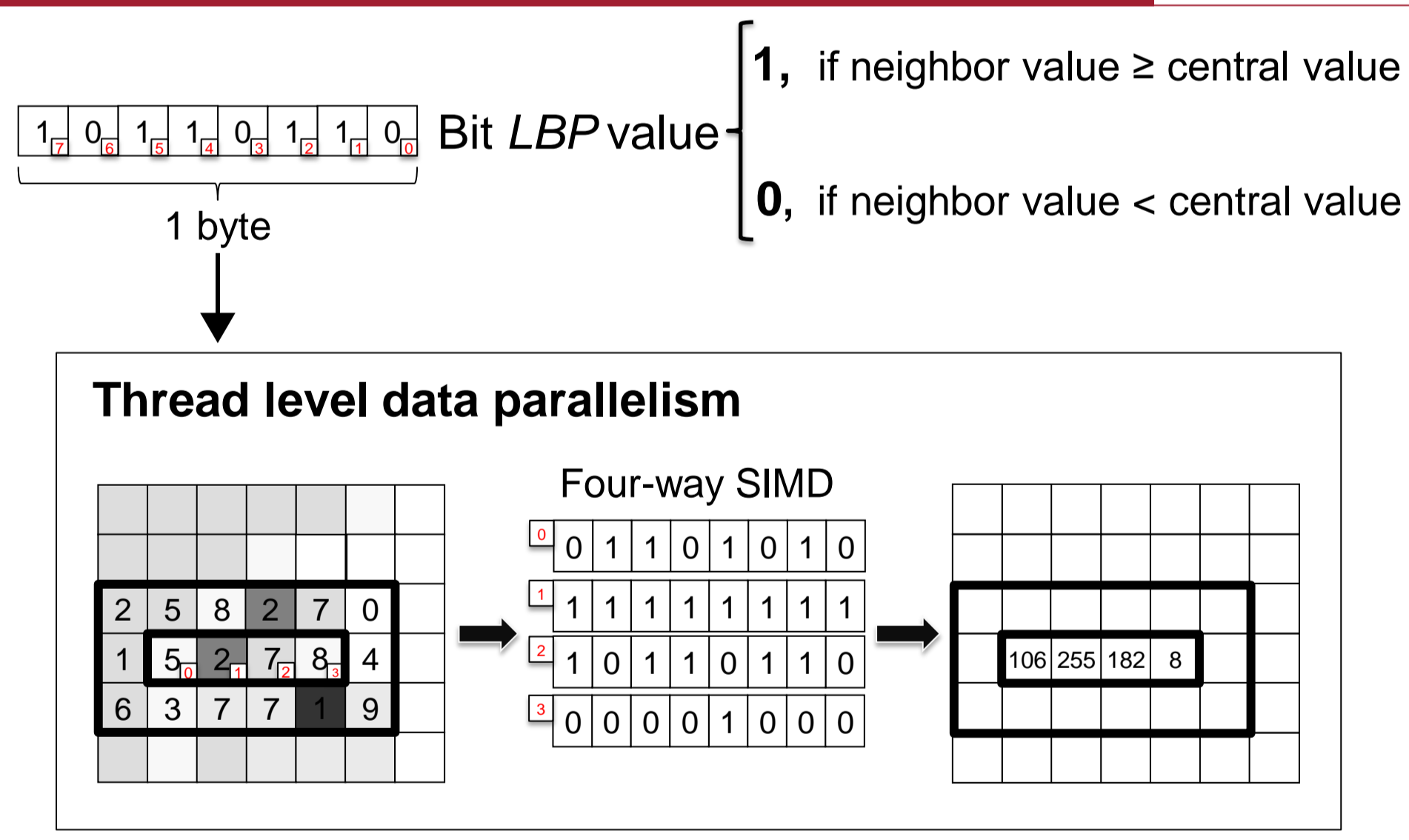[3]Computer Vision Center

Abstract

Pedestrian detection for autonomous driving has gained a lot of prominence during the last few years. Besides the fact that it is one of the hardest tasks within computer vision, it involves huge computational costs. The real-time constraints in the field are tight, and regular processors are not able to handle the workload obtaining an acceptable ratio of frames per second (fps). Moreover, multiple cameras are required to obtain accurate results, so the need to speed up the process is even higher. Taking the work in [1] as our baseline, we propose a CUDA implementation of a pedestrian detection system. Further, we introduce significant algorithmic adjustments and optimizations to adapt the problem to the GPU architecture. The aim is to provide a system capable of running in real-time obtaining reliable results.
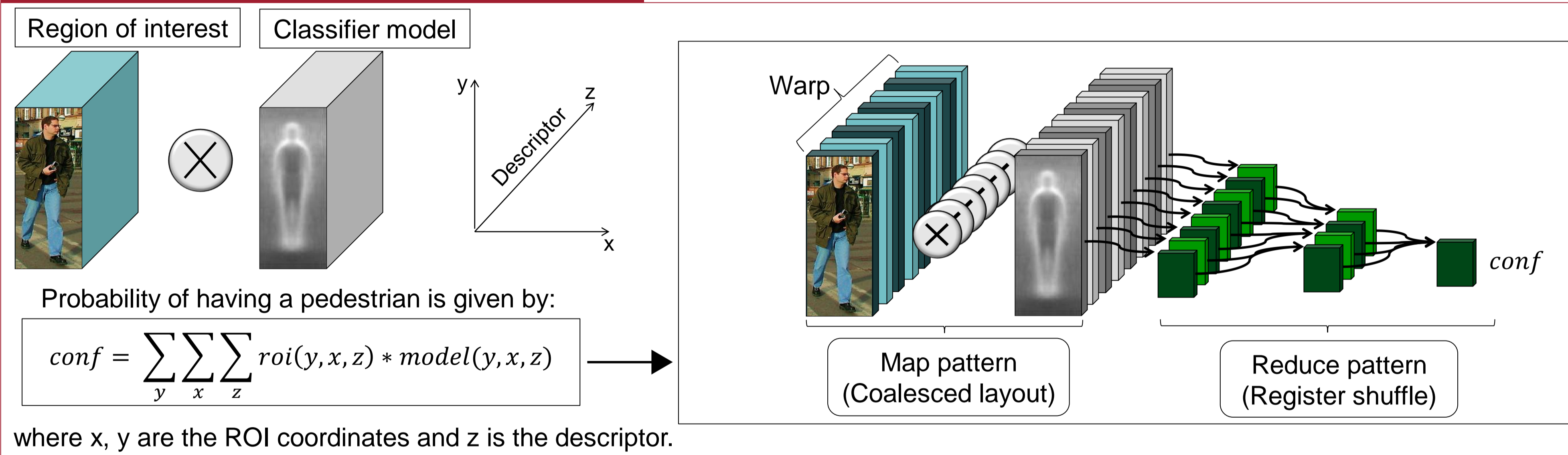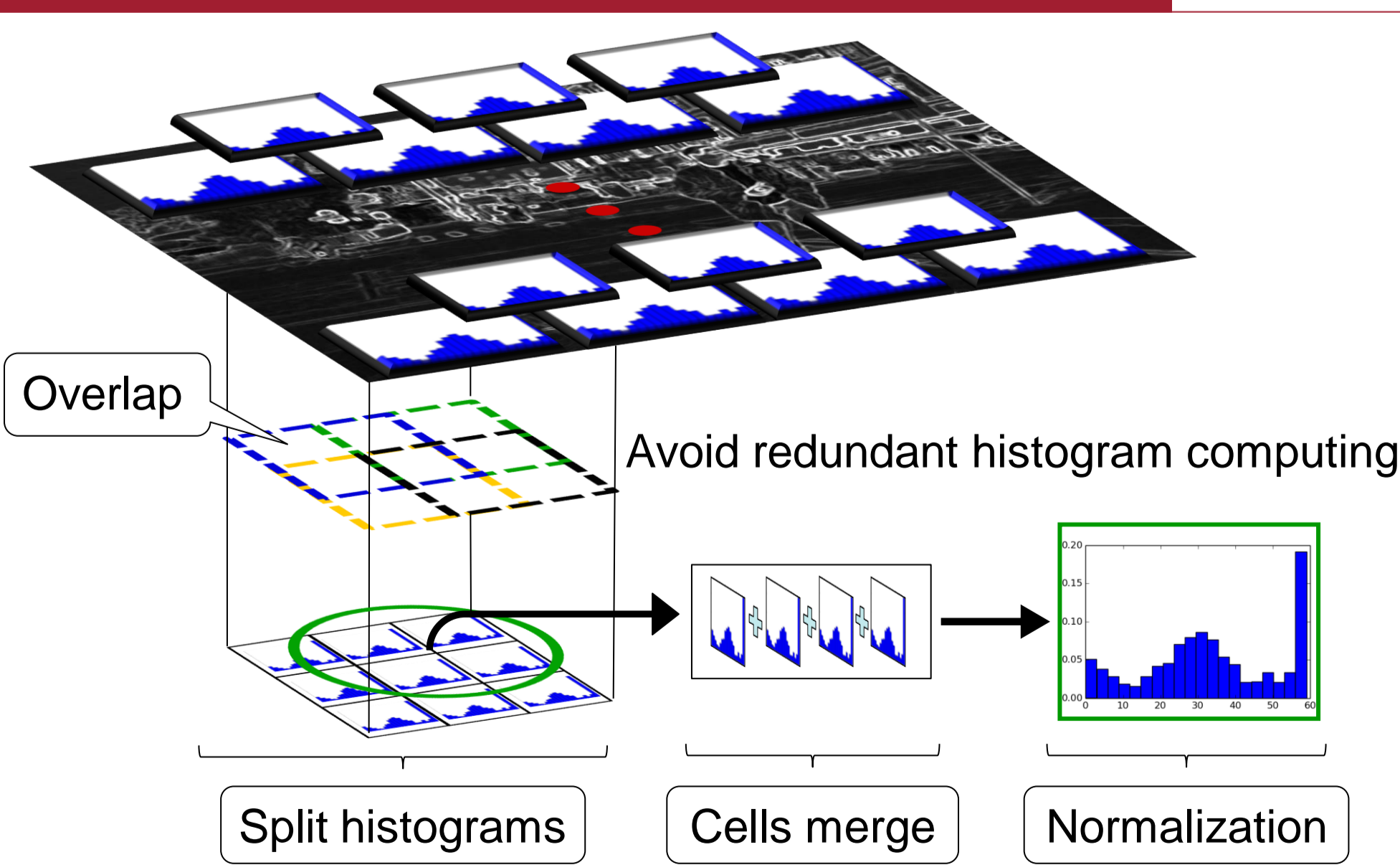
Object Detection Pipeline

Image Descriptor — Local binary patterns — Grayscale image / Texture descriptor

Feature Extraction — Block histograms — Image features

Candidate Generation — Sliding window — Candidate windows

Classification — Support vector machine — Window features / Classifier — No pedestrian / Pedestrian

Refinement — Non-max suppression — Multiple detections / Detection result

Input Pyramid image

Local Binary Patterns

Bit LBP value
1, if neighbor value ≥ central value
0, if neighbor value < central value

1 byte

Thread level data parallelism

Four-way SIMD

Support Vector Machine

Region of interest   Classifier model

Warp

Probability of having a pedestrian is given by:

$$conf = \sum_{y} \sum_{x} \sum_{z} roi(y,x,z) * model(y,x,z)$$

where x, y are the ROI coordinates and z is the descriptor.

Map pattern (Coalesced layout)   Reduce pattern (Register shuffle)   conf

Block Histograms

Overlap

Avoid redundant histogram computing

Split histograms   Cells merge   Normalization

Results

Performance comparison versus CPU
GPU-Naive
GPU-Optimized

Benchmark
GPU: GTX Titan
CPU: Intel Xeon ES-1650

Speed-Up — LBP, Histograms, SVM — Kernel

KITTI, Pedestrian
miss rate (%) vs false positives per image
HOG - LinSVM (50.29)
LBP - LinSVM (48.91)
HOGLBP - LinSVM (38.66)
HOGLBP - RF (30.05)

Future work
HOG – LinSVM
HOGLBP – LinSVM
HOGLBP – Random Forest

References:
[1] Marin, J., Vázquez, D., López, A. M., Amores, J., & Leibe, B. (2013, December). Random forests of local experts for pedestrian detection. In *Computer Vision (ICCV), 2013 IEEE International Conference on* (pp.2592-2599).