

Towards robust automatic detection of vulnerable road users: monocular pedestrian tracking from a moving vehicle

Kristof Van Beeck, **Floris De Smedt**, Sander Beckers, Lars
Struyf, Joost Vennekens, Gorik De Samblanx, Toon
Goedemé, Tinne Tuytelaars

June 13, 2011

Context

Introduction
Problem
situation

Algorithms

HOG
Kalman
approach
Probabilistic
approach
Experiments &
results

Implementation

Hardware
Implementation
details
Implementation
results

Conclusion & Future work

Conclusion
Future work

Context

Introduction

Problem situation

Algorithms

HOG

Kalman approach

Probabilistic approach

Experiments & results

Implementation

Hardware

Implementation details

Implementation results

Conclusion & Future work

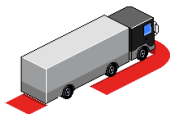
Conclusion

Future work

Research projects

1. PhD project Kristof Van Beeck
 - ▶ Automatic detection of vulnerable road users in blind spot systems
2. S.O.S. OpenCL: Multicore cooking
 - ▶ Improving performance of applications by using GPU-hardware

Existing systems



Passive: Need action of the driver

- ▶ Blind spot mirror
- ▶ Passive camera system

Active: A lot of false detections. Alarm is not reliable.

- ▶ Ultrasonic systems
- ▶ IR systems

Context

Introduction

Problem
situation

Algorithms

HOG

Kalman
approachProbabilistic
approachExperiments &
results

Implementation

Hardware

Implementation
detailsImplementation
results

Conclusion &

Future work

Conclusion

Future work

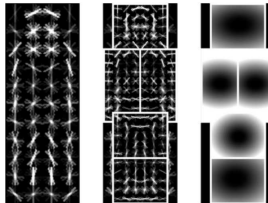
Proposed solution



- ▶ Active camera system
 - ▶ Automatic detection of vulnerable road users
 - ▶ Automatic warning on monitor
 - ▶ Easy to integrate in existing camera systems
- ▶ Multi-cue approach
 - ▶ **Appearance-based information**
 - ▶ Motion-based information
 - ▶ Efficient integration

Star cascade HOG object detector

- ▶ Appearance-based method
- ▶ We use a person-model for pedestrian detection
- ▶ Cascade object detection with deformable part models (P. Felzenszwalb, 2010)
- ▶ Multi-part model
- ▶ High accuracy at the cost of high calculation cost



Context

Introduction
Problem
situation

Algorithms

HOG

Kalman
approach
Probabilistic
approach
Experiments &
results

Implementation

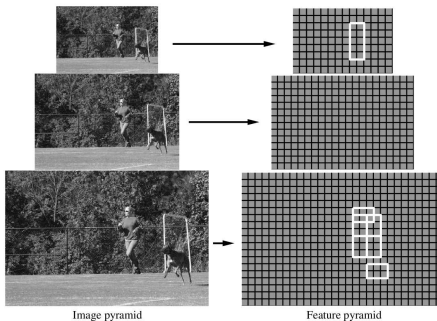
Hardware
Implementation
details
Implementation
results

Conclusion &
Future work

Conclusion
Future work

Procedure

- ▶ Building scale-space pyramid
 - ▶ Resize images
 - ▶ Calculate HOG-features for each layer
- ▶ Calculate the model response



Context

Introduction
Problem
situation

Algorithms

HOG
**Kalman
approach**
Probabilistic
approach
Experiments &
results

Implementation

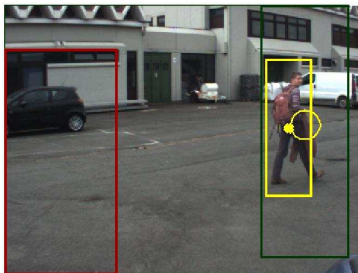
Hardware
Implementation
details
Implementation
results

Conclusion &
Future work

Conclusion
Future work

Kalman

- ▶ Tracking-by-detection
- ▶ Initial search region defined
- ▶ On detection
 - ▶ Initiate Kalman-model
 - ▶ Define circular region, radius is scale dependent
 - ▶ Define new search region
- ▶ Search space is extended with search regions of detections



Context

Introduction
Problem
situation

Algorithms

HOG

**Kalman
approach**

Probabilistic
approach
Experiments &
results

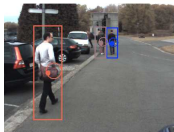
Implementation

Hardware
Implementation
details
Implementation
results

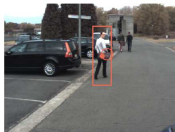
**Conclusion &
Future work**

Conclusion
Future work

Kalman



Frame 60



Frame 70

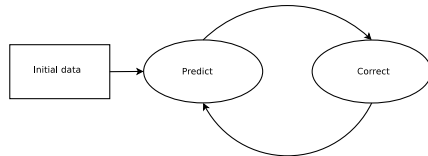


Frame 80



Frame 90

$$\begin{bmatrix} x_t \\ y_t \\ vx_t \\ vy_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ vx_{t-1} \\ vy_{t-1} \end{bmatrix}$$



- ▶ Constant velocity model
- ▶ Initialisation speed vector is opposite to camera-motion
- ▶ Speed of pedestrian is negligible compared to speed of the truck

Context

Introduction
Problem
situation

Algorithms

HOG
Kalman
approach
**Probabilistic
approach**
Experiments &
results

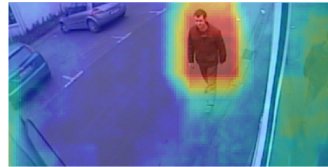
Implementation

Hardware
Implementation
details
Implementation
results

Conclusion &
Future work

Conclusion
Future work

Particle filter



- ▶ Multi-hypothesis framework
- ▶ Using the matching score of the HOG-model
- ▶ Increased accuracy over Kalman
- ▶ Same motion model as Kalman



Frame 20



Frame 40



Frame 60



Frame 80

Datasets

- ▶ 2000 frames in 12 sequences with a standard camera
- ▶ 10000 frames in 7 sequences with a blind spot camera
- ▶ Experiments are done with the first dataset
- ▶ Implementation in Matlab and partially in C (no OpenCL optimisation)



Results

Table: Accuracy results for Kalman Filter approach.

	avg. fps	precision	recall
seq. 1	9.57	0.76	0.92
seq. 2	8.97	0.95	0.93
seq. 3	9.47	0.95	0.8
average	9.34	0.89	0.88

Table: Speed Results for both our algorithm implementations

		Dataset 1
Kalman Filter Approach	Average max. FPS	12.6 FPS
	Average min. FPS	2.8 FPS
	Average FPS	8.6 FPS
Particle Filter Approach	Average max. FPS	0.25 FPS
	Average min. FPS	0.11 FPS
	Average FPS	0.22 FPS

GPU implementation

- ▶ Parts of the scale-space pyramid construction on GPU
 - ▶ Image rescaling
 - ▶ Part of the HOG feature calculation
- ▶ OpenCL: A framework for writing programs that execute across heterogenous platforms consisting of CPUs, GPUs and other processors.
- ▶ Dedicated Nvidia GTX295 card as GPU
- ▶ Intel i7 870 processor (2.93GHz) as CPU

OpenCL version	1.0
CUDA cores	480 (240 per GPU)
compute units	30
Clock frequency	1242MHz
Global memory	1792MB (896MB per GPU)

Context

Introduction
Problem
situation

Algorithms

HOG
Kalman
approach
Probabilistic
approach
Experiments &
results

Implementation

Hardware
**Implementation
details**
Implementation
results

Conclusion &
Future work

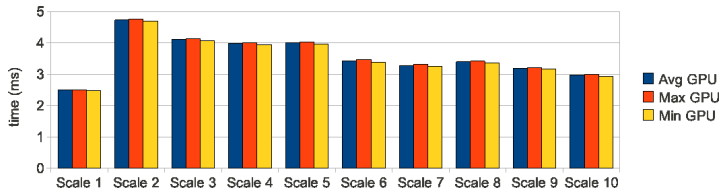
Conclusion
Future work

Image rescale

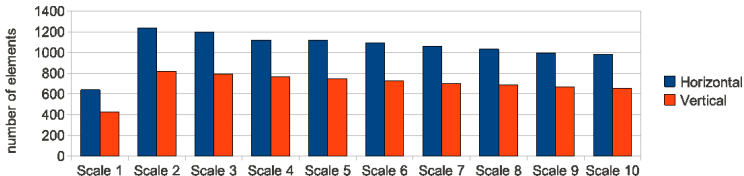
- ▶ Rescale in 2 steps
 - ▶ Horizontally
 - ▶ Vertically
- ▶ Source image in texture memory
- ▶ Double precision output
- ▶ 2 approaches



Image rescale first approach



- ▶ Each row/column resized in the same way
- ▶ Resize information reused
- ▶ 1 thread per row/column



Context

Introduction
Problem
situation

Algorithms

HOG
Kalman
approach
Probabilistic
approach
Experiments &
results

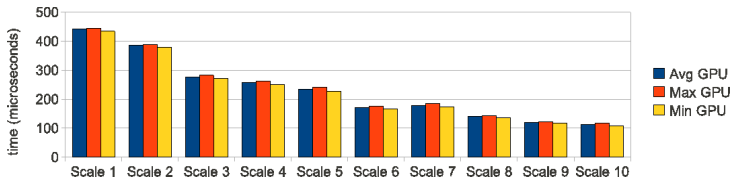
Implementation

Hardware
**Implementation
details**
Implementation
results

Conclusion &
Future work

Conclusion
Future work

Image rescale second approach



- ▶ 1 thread per pixel of resized image
- ▶ A lot more threads than first approach (more parallel)
- ▶ Higher overhead cost
- ▶ Big time performance improvement

Context

Introduction
Problem
situation

Algorithms

HOG
Kalman
approach
Probabilistic
approach
Experiments &
results

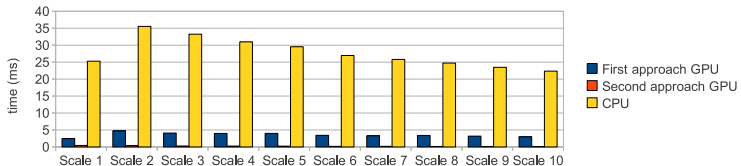
Implementation

Hardware
**Implementation
details**
Implementation
results

Conclusion &
Future work

Conclusion
Future work

Image rescale comparison



- ▶ Huge improvement over CPU
- ▶ Second approach is a lot faster at the cost of more overhead
- ▶ GPU bottleneck is memory access

Context

Introduction
Problem
situation

Algorithms

HOG
Kalman
approach
Probabilistic
approach
Experiments &
results

Implementation

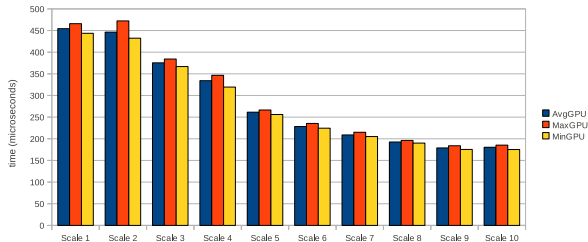
Hardware
Implementation
details
Implementation
results

Conclusion &

Future work

Conclusion
Future work

Feature calculation



- ▶ Part of the HOG feature calculation on GPU
 - ▶ Creating histogram of gradients
 - ▶ **Calculate features from HOG**
- ▶ Each thread calculates 1 value of each layer (32 layers)
- ▶ Decrease in calculation time is directly proportional to image size

Context

Introduction
Problem
situation

Algorithms

HOG
Kalman
approach
Probabilistic
approach
Experiments &
results

Implementation

Hardware
Implementation
details
Implementation
results

Conclusion &

Future work

Conclusion
Future work

Comparison GPU-CPU

Table: Timing results for sequential calculation of 10 scales

	CPU (ms)	GPU (ms)
Rescale	277.93	35.519
		2.316
Features	70.89	2.86

Context

Introduction
Problem
situation

Algorithms

HOG
Kalman
approach
Probabilistic
approach
Experiments &
results

Implementation

Hardware
Implementation
details
Implementation
results

Conclusion &
Future work

Conclusion
Future work

Conclusion

- ▶ Very robust appearance-based detector
- ▶ 2 tracking frameworks
 - ▶ Kalman: near real-time performance but only one hypothesis
 - ▶ Particle filter: multi-hypothesis but large calculation time
- ▶ GPU implementation
 - ▶ Part of the Felzenszwalb implementation
 - ▶ Large improvement over CPU performance

Context

Introduction
Problem
situation

Algorithms

HOG
Kalman
approach
Probabilistic
approach
Experiments &
results

Implementation

Hardware
Implementation
details
Implementation
results

Conclusion &
Future work

Conclusion
Future work

Future work

- ▶ Implementation of remaining parts on GPU
- ▶ Integrate this GPU-implementation with the tracking frameworks
- ▶ Speed-up the Particle approach

Context

- Introduction
- Problem situation

Algorithms

- HOG
- Kalman approach
- Probabilistic approach
- Experiments & results

Implementation

- Hardware
- Implementation details
- Implementation results

Conclusion & Future work

- Conclusion
- Future work**

Thank you for your attention.
Are there any questions?