Improved Robustness and Efficiency for Automatic Visual Site Monitoring

Gerald Dalley
Thesis Defense
9 June 2009

Committee: Eric Grimson, Trevor Darrell, and Bill Freeman
Collaborators: Xiaogang Wang, Josh Migdal, Kinh Tieu, Lily Lee, Tomáš Ižo,...
Commercial and Transportation Applications

- **Efficiency**
  - What are the traffic bottlenecks?
  - How can we coordinate arrival schedules to minimize congestion?

- **Marketing**
  - How do in-store marketing campaigns effect behavior?
  - Are shoppers stopping at the sales booth?

- **Loss prevention**
  - How can we detect customer theft?
  - How can we detect employee theft?
Security Applications

• Threat detection
  – Unauthorized access
  – Violence
  – Theft
  – Tailing
  – Loitering
  – Sudden widespread panic

• Recognition
  – Is this person authorized?
  – Is this a “wanted” person?

• Activity understanding
  – *What are the common traffic patterns?*
  – How can we deploy security resources more effectively?
Applications & Typical Scenes

Identifying individuals

Event detection

General tracking

Mixed: boats, cars, people

Activity modeling in large public spaces
Automatic Site Monitoring Pipeline

- Detection
- Tracking
- Analysis
Automatic Site Monitoring Pipeline

- Detection
- Tracking
- Analysis

- Background Subtraction
- Feature Points
- Strong Model (HOG)
Automatic Site Monitoring Pipeline

- **Detection**
  - Background subtraction

- **Tracking**
  - Feature points

- **Analysis**
  - Strong models
Automatic Site Monitoring Pipeline

- Kalman filter
- Meanshift
- ...

Detection

Tracking

Analysis

Time windowing: for rendering purposes only
Automatic Site Monitoring Pipeline

Detection

• Identifying individual people

Tracking

• Recognize events (loitering, theft, etc.)
  – Ivanonv and Bobick, PAMI 2000.
  – Vu, Bremond, and Thonnat, ECAI 2002.
  – PETS 2006 and PETS 2007 workshops (many papers)

Analysis

• Model flow patterns and site usage
  – Wang et al., CVPR 2008.
Thesis Contributions

• Background subtraction
  – Waving trees, rippling water  \textit{5.5\% drop in false positive rate}

• Large-scale monitoring
  – Clustering of path segments
  – Dalal and Triggs on a GPU  \textit{Up to 76x faster than CPU}

• Gait recognition
  – Model-based silhouettes  \textit{6\%—44\% boost in recognition rates}

• Event detection
  – Integrated detection and tracking  \textit{Only system to complete the PETS 2007 challenge}
This talk...
Outline

- Motivation
- Activity model overview
- Weak model detectors
- Strong model detector
- Data parallel implementation
- Summary
Outline

• Activity model overview
• Weak model detectors
• Strong model detector
• Data parallel implementation
High Level

• Goal
  – Cluster trajectories to find common paths

• Approach
  – Infinite mixture model
Hierarchical Dirichlet Processes (HDPs)

- HDPs: Teh JASA 2006
- w/ trajectories: Wang CVPR 2008

$$\gamma$$

$$\beta$$

$$\alpha$$

$$\pi_j$$

$$z_{ji}$$

$$w_{ji}$$

$$I_j$$

$$J$$

$$\varphi_c$$

$$\lambda$$

$$\phi_{17}$$

Mixing Weight

Cluster Index
Outline

• Activity model overview
• Weak model detectors
  – Background subtraction
  – Feature point detection
• Strong model detector
• Data parallel implementation
Background Subtraction
Background Subtraction:

Precision-Recall

Recall

Precision

Ideal Corner

background subtraction
Background Subtraction:

Problems

Split blobs, missing people: 
*glare, too frequent foreground*

Merged blobs: 
*shadows, crowds*
Alternative:
Shi & Tomasi Feature Point Detection

- True positives
- False positives
- False negatives
- True negatives
- Don’t care
Improved Recall, but Low Precision

![Graph showing recall vs. precision with background subtraction and feature points]

Ideal Corner

- • • • • • feature points
- • • • • • background subtraction
Clustering Feature Point Trajectories
Perplexity
(cluster uncertainty given observed location)
Crowded bidirectional traffic
Most Tracks Just Going East and West
A Few Bad Tracks Couple East and West

- Normal westbound track
- Bad track #1
- Bad track #2
- Westbound & #2 associated
- Eastbound & #1 associated
- #1 & #2 associated
- Normal eastbound track
- Track starting point
Outline

• Activity model overview
• Weak model detectors
• Strong model detector
  – Dalal and Triggs' HOG detector
  – Classification results
  – Activity modeling results
• Data parallel implementation
Dalal & Triggs HOG Features

At every possible pedestrian location

Input Window → Gamma Correct → Block Descr. Locs. → Gradients → Block Descr. → Voting Stencil → Window Descriptor

At every block descriptor location
Sufficient Precision and Recall

- Ideal Corner
- Lower precision, but fewer systematic errors

Graph showing precision vs. recall for different methods:
- HOG detector
- Feature points
- Background subtraction

Note: Sufficient precision and recall.
Better Perplexity

**Point Tracking**
- mean = 1.5
- median = 1.1

**Pedestrian Detector Tracks**
- mean = 2.6
- median = 2.4
Selected Clusters

SE ➔ Escalator

SE ➔ NW

SE ➔ SW

SW ➔ SE
Breaking up Merged Paths

More permissive priors →
*Can separate the 6 paths from west to escalators*
Some Directional Degeneracies Remain

Cause: tracking errors

Cause: loitering and meandering
Outline

• Activity model overview
• Weak model detectors
• Strong model detector
• Data parallel implementation
  – Motivation
  – GPU Intuition
  – Our design
  – Speedups
Good Results, but Too Slow

...a little faster would be nice.

Our data:
• 40 hours
• 1920×1080 frames
  • 6.75× the pixels/frame w.r.t. 640×480
  • 27× the pixels/frame w.r.t. 320×240
• progressive scan
Title: CPU Characteristics

- One thing fast
  - High clock speed
  - Pipelining

- Complex control flow
  - Cache
  - Branch prediction
  - Speculation
  - ...

- Task parallel: a few different things fast
  - Multicore
  - Hyperthreading
  - Sophisticated caches

- Data parallel: Same instruction on a few data items
  - MMX, SSE, etc.
GPU Characteristics

• *Same* instruction, many data items
  – 240 “cores” or more

• Very high memory *bandwidth*
  – 10× a CPU’s

• Typical speedups:
  – 10×—100×

• Programming
  – Style: C/C++
  – Optimization effort ≈ C++ & assembly mix

• Slow if...
  – Insufficiently parallel code
  – Random memory access
  – Branching
Intuition: What Works Well on a GPU

• In general
  – $10^{<<<\text{MANY>>>}$ independent inputs and/or outputs
  – Localized memory access

• Typical applications
  – Filterbanks
  – Sliding window algorithms
  – Code that’s easy to vectorize in Matlab
Dalal & Triggs HOG Features

At every possible pedestrian location

Input Window → Gamma Correct → Block Descr. Locs. → Gradients → Voting Stencil → Block Descr. → Window Descriptor

At every block descriptor location
Our CUDA Pipeline

1. **Read Frame**
2. **Decode Frame**
3. **memcpy to GPU**
4. **Data Reordering**
5. **Crop and Resample**
6. **Gamma Correction and Gradients**
7. **Normalized Block Descriptors**
8. **Linear SVM Classification**
9. **memcpy to CPU**
10. **Meanshift**
11. **Non-maxima Suppression**
12. **Kalman Tracking**
13. **Output**

**For each frame:**

- **Runs on CPU (1 thread)**
- **Runs on GPU**
- **CPU ↔ GPU transfers**
- **Possible on GPU**
### CPU vs. GPU Times:
Results from a Simplified Profiling Application

<table>
<thead>
<tr>
<th>Processing Step</th>
<th>CPU Implementation</th>
<th>GPU Implementation</th>
<th>GPU Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read input (CPU)</td>
<td>0%</td>
<td>17%</td>
<td></td>
</tr>
<tr>
<td>GPU resizer setup</td>
<td></td>
<td>5%</td>
<td></td>
</tr>
<tr>
<td>Resize</td>
<td>4%</td>
<td>11%</td>
<td>24.3×</td>
</tr>
<tr>
<td>Gradients</td>
<td>24%</td>
<td>9%</td>
<td>164.0×</td>
</tr>
<tr>
<td>Normalized block descriptors</td>
<td>57%</td>
<td>35%</td>
<td>97.7×</td>
</tr>
<tr>
<td>Window classification</td>
<td>14%</td>
<td>8%</td>
<td>100.6×</td>
</tr>
<tr>
<td>Cleanup</td>
<td>0%</td>
<td>12%</td>
<td>0.5×</td>
</tr>
<tr>
<td>Detection (CPU)</td>
<td>0%</td>
<td>4%</td>
<td>1.1×</td>
</tr>
<tr>
<td>TOTAL</td>
<td>23 seconds</td>
<td>0.4 seconds</td>
<td>58.8×</td>
</tr>
</tbody>
</table>
GPU Speedup Results

• Our Implementation
  – 58.8× to 76× speedup (vs. optimized CPU-only)
  – Current bottlenecks
    • Video decoding on the CPU (17%)
    • Block descriptors (35%)
    • Bookkeeping & memory transfers (17%)

• Wojek, Dorkó, Schulz, Schiele [DAGM 2008]
  – 30× speedup
  – Optimized for the previous GPU architecture
  – Less efficient usage of memory bandwidth
Summary

• Fast HOG implementation
  – 58.8× to 76× speedup

• Better clustering of trajectory flows
  – Qualitative improvements
  – Perplexity
Future Work

• Scale to true HD real-time
  – Multithreaded CPU
  – Multiple GPUs
  – Asynchronous data transfers
  – More computation to GPUs

• Better HOG training
  – Explicit occlusion handling
  – Add video features
    (a la Dalal and Triggs 2006)

• Alternative detectors
  – Boosted cascade on GPUs
    (CPU: Avidan; Viola & Jones)

• Activity modeling
  – Learn long-term flow trends
  – Temporal dependencies
    (via HMMs)

• Integrate with other technologies in this thesis...
Other Potential Applications for Fast and Robust Pedestrian Detection

Appearance models for recognition

Multimodal tracking for event detection

Abandoned Luggage!
Acknowledgments

• Thesis Committee
  – Eric Grimson
  – Bill Freeman
  – Trevor Darrell

• Too many friends and fellow students here at MIT to list individually...

• Collaborators
  – Xiaogang Wang
  – Josh Migdal
  – Kinh Tieu
  – Lily Lee
  – Tomáš Ižo
  – Jim Sukha, Krista Ehinger, and Geza Kovacs

• Funders
  – DARPA
  – MIT
  – Shell
  – Singapore
  – ...

• Work Experiences
  – Microsoft Research
  – MERL
  – BAE Systems
  – D.E. Shaw

• My wife, Dianna
Automatic Site Monitoring Pipeline

Detection

- Background subtraction

- Feature points

- Strong models

Tracking

Analysis
Automatic Site Monitoring Pipeline

• Identifying individual people

• Recognize events (loitering, theft, etc.)
  – PETS 2006 and PETS 2007 workshops (many papers)

• Model flow patterns and site usage
Same Quality
(minor differences due to training tweaks)

Due to some minor training artifacts
A Learned Classification Boundary (rotated)
Detections on One Frame
ROC Curves
Training Set Influence
MRF equations [MG05]

Grid of observed pixel colors

Grid of unknown FG/BG labels

\[
\Psi_{\{s,r\}}(X_s^t, X_r^t) = \begin{cases} 
\psi_1, & \text{if } X_s^t = X_r^t = 1 \\
\psi_2, & \text{if } X_s^t = X_r^t = 0 \\
\psi_3, & \text{if } X_s^t \neq X_r^t 
\end{cases}
\]

\[
\Phi_{\{s\}}(X_s^t, D_s^t) = \begin{cases} 
\delta(d_s^t), & \text{if } X_s^t = 0 \\
\ln 2^{24}, & \text{if } X_s^t = 1 
\end{cases}
\]

\[
\Theta_{\{s,t,s'\}}(X_s^t, X_s^{t'}) = \begin{cases} 
\theta_1, & \text{if } X_s^t = X_s^{t'} = 1 \\
\theta_2, & \text{if } X_s^t = X_s^{t'} = 0 \\
\theta_3, & \text{if } X_s^t \neq X_s^{t'} 
\end{cases}
\]
Detections before Global Optimization
Sample Marginal of Observations
Our Data
Foreground Likelihood
Training Data
Our CUDA Pipeline: Percent Time per Module

- Read Frame: 0% / 17%
- Decode Frame: 4% / 11%
- memcpy to GPU: - / 5%
- Data Reordering: 24% / 9%
- Crop and Resample: 4% / 11%
- Gamma Correction and Gradients: 57% / 35%
- Normalized Block Descriptors: 4% / 11%
- Linear SVM Classification: 14% / 8%
- memcpy to CPU: 0% / 12%
- Meanshift: 0% / 4%
- Non-maxima Suppression: 0% / 4%
- Kalman Tracking: 0% / 4%
- Output: 0% / 4%
# CPU vs. GPU Times:
Results from a Simplified Profiling Application

<table>
<thead>
<tr>
<th>Processing Step</th>
<th>Time (CPU Impl.)</th>
<th>Time (GPU Impl.)</th>
<th>GPU Impl. Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read input (CPU)</td>
<td>68.0 ms</td>
<td>68.0 ms</td>
<td>(0%) (17%)</td>
</tr>
<tr>
<td>GPU resizer setup</td>
<td></td>
<td>18.1 ms</td>
<td>(5%)</td>
</tr>
<tr>
<td>Resize</td>
<td>1,045.8 ms</td>
<td>43.0 ms</td>
<td>(4%) (11%)</td>
</tr>
<tr>
<td>Gradients</td>
<td>5,636.2 ms</td>
<td>34.4 ms</td>
<td>(24%) (9%)</td>
</tr>
<tr>
<td>Normalized block descriptors</td>
<td>13,412.3 ms</td>
<td>137.2 ms</td>
<td>(57%) (35%)</td>
</tr>
<tr>
<td>Window classification</td>
<td>3,159.1 ms</td>
<td>31.4 ms</td>
<td>(14%) (8%)</td>
</tr>
<tr>
<td>Cleanup</td>
<td>23.8 ms</td>
<td>45.5 ms</td>
<td>(0%) (12%)</td>
</tr>
<tr>
<td>Detection (CPU)</td>
<td>16.2 ms</td>
<td>15.0 ms</td>
<td>(0%) (4%)</td>
</tr>
<tr>
<td>TOTAL</td>
<td>23,082.4 ms</td>
<td>392.3 ms</td>
<td>(4%) (17%)</td>
</tr>
</tbody>
</table>
bgsub
Suppressing Spurious Detections
Results with Mittal & Paragios Traffic Clip

Key:
- **bboxes from ground truth**
  - True positive
  - False positive
  - False negative

Ground truth, frame 150

Mittal and Paragios: 1 TP, 0 FP, 2 FN

MoG: 3 TP, 14 FP, 0 FN

Ours: 3 TP, 0 FP, 0 FN
Ground truth

Mittal & Paragios (2 TP, 0 FP, 1 FN)

MoG (3 TP, 3 FP, 0 FN)

Ours (3 TP, 0 FP, 0 FN)
Our Model

- Mixture of Gaussians (MoG)

\[ p(c_i | \Phi) \propto \sum_{j \in N_i} w_j \mathcal{N}(c_i; \mu_j, \Sigma_j) \]

- \( c_i \): the observed color at pixel \( i \)
- \( \Phi \): the model \( \{w_j, \mu_j, \Sigma_j\}_j \)
- \( N_i \): neighborhood of pixel \( i \)

Pixels Affected by a Given Mixture Component

- Standard MoG
- Ours
Foreground/Background Classification

- Find best matching background Gaussian, $j$
  - Use neighborhood
- Squared Mahalanobis Distance

$$d_{ij} = (c_i - \mu_j)^T \Sigma_j^{-1} (c_i - \mu_j)$$
Model Update Options

Soft Updates

Regional
update all matches

Local
soft Stauffer-Grimson

Hard Updates

best match only

hard Stauffer-Grimson
ROC (Wallflower)
Parameter Sensitivity

**traffic**

- Frequency vs. max number of mistakes
- Different line colors represent different window sizes: ws=7, ws=5, ws=3, ws=1

**wallflower**

- Frequency vs. max number of mistakes
- Different line colors represent different window sizes: ws=7, ws=5, ws=3, ws=1