Cascade Object Detection with Deformable Part Models

Pedro Felzenszwalb    Ross Girshick
University of Chicago

David McAllester
TTI at Chicago

THE UNIVERSITY OF
CHICAGO
COMPUTER SCIENCE

TOYOTA TECHNOLOGICAL INSTITUTE
What we do

more than one order of magnitude speedup

We build *fast cascade detectors* from *state-of-the-art* deformable part models

UofC-TTI object detection system
## Speedup examples

<table>
<thead>
<tr>
<th></th>
<th>baseline</th>
<th>cascade</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>bicycle</td>
<td>14.7 sec/image</td>
<td>0.6 sec/image</td>
<td>24x</td>
</tr>
<tr>
<td>bus</td>
<td>14.5 sec/image</td>
<td>0.7 sec/image</td>
<td>21x</td>
</tr>
<tr>
<td>car</td>
<td>11.9 sec/image</td>
<td>0.9 sec/image</td>
<td>13x</td>
</tr>
<tr>
<td>person</td>
<td>12.8 sec/image</td>
<td>1.9 sec/image</td>
<td>7x</td>
</tr>
<tr>
<td><strong>PASCAL 2007</strong></td>
<td><strong>average</strong></td>
<td></td>
<td><strong>14.5x</strong></td>
</tr>
</tbody>
</table>

*Single-threaded* implementations

Cascade thresholds set for full recall (i.e., “slow mode”)

Average image size: 382 x 471 pixels
Star models

test image  part-based deformable model  detection
Object hypothesis score

\[ \Omega \] set of \((x, y, \text{scale})\) part locations

\[ m_i(\omega) \] score of \(i\)-th part at \(\omega \in \Omega\)

\[ \Delta \] set of \((dx, dy)\) part displacements

\[ d_i(\delta) \] cost of moving \(i\)-th part by \(\delta \in \Delta\)

\[
\text{score}(\omega, \delta_1, \ldots, \delta_n) = m_0(\omega) + \sum_{i=1}^{n} m_i(a_i(\omega) + \delta_i) - d_i(\delta_i)
\]
Object hypothesis score

\[ \Omega \] set of \((x, y, \text{scale})\) part locations

\[ m_i(\omega) \] score of \(i\)-th part at \(\omega \in \Omega\)

\[ \Delta \] set of \((dx, dy)\) part displacements

\[ d_i(\delta) \] cost of moving \(i\)-th part by \(\delta \in \Delta\)

\[
\text{score}(\omega, \delta_1, \ldots, \delta_n) = m_0(\omega) + \sum_{i=1}^{n} m_i(a_i(\omega) + \delta_i) - d_i(\delta_i)
\]
Object hypothesis score

\[ \text{score}(\omega, \delta_1, \ldots, \delta_n) = 
\]

\[ m_0(\omega) + \sum_{i=1}^{n} m_i(a_i(\omega) + \delta_i) - d_i(\delta_i) \]

\( \Omega \) set of \((x, y, scale)\) part locations

\( m_i(\omega) \) score of \(i\)-th part at \(\omega \in \Omega\)

\( \Delta \) set of \((dx, dy)\) part displacements

\( d_i(\delta) \) cost of moving \(i\)-th part by \(\delta \in \Delta\)
Object hypothesis score

\[ \text{score}(\omega, \delta_1, \ldots, \delta_n) = m_0(\omega) + \sum_{i=1}^{n} m_i(a_i(\omega) + \delta_i) - d_i(\delta_i) \]

- \( \Omega \) set of \((x, y, scale)\) part locations
- \( m_i(\omega) \) score of \(i\)-th part at \( \omega \in \Omega \)
- \( \Delta \) set of \((dx, dy)\) part displacements
- \( d_i(\delta) \) cost of moving \(i\)-th part by \( \delta \in \Delta \)
Object hypothesis score

\( \Omega \) set of \((x, y, \text{scale})\) part locations

\( m_i(\omega) \) score of \(i\)-th part at \(\omega \in \Omega\)

\( \Delta \) set of \((dx, dy)\) part displacements

\( d_i(\delta) \) cost of moving \(i\)-th part by \(\delta \in \Delta\)

score\((\omega, \delta_1, \ldots, \delta_n)\) =

\[ m_0(\omega) + \sum_{i=1}^{n} m_i(a_i(\omega) + \delta_i) - d_i(\delta_i) \]

score of root
Object hypothesis score

\[
\begin{align*}
\text{score}(\omega, \delta_1, \ldots, \delta_n) = \\
m_0(\omega) + \sum_{i=1}^{n} m_i(a_i(\omega) + \delta_i) - d_i(\delta_i)
\end{align*}
\]

sum over non-root parts

\(\Omega\) set of \((x, y, \text{scale})\) part locations

\(\Delta\) set of \((dx, dy)\) part displacements

\(m_i(\omega)\) score of \(i\)-th part at \(\omega \in \Omega\)

\(d_i(\delta)\) cost of moving \(i\)-th part by \(\delta \in \Delta\)
Object hypothesis score

\[ \text{score}(\omega, \delta_1, \ldots, \delta_n) = m_0(\omega) + \sum_{i=1}^{n} m_i(a_i(\omega) + \delta_i) - d_i(\delta_i) \]

- \( \Omega \): set of \((x, y, scale)\) part locations
- \( m_i(\omega) \): score of \(i\)-th part at \(\omega \in \Omega\)
- \( \Delta \): set of \((dx, dy)\) part displacements
- \( d_i(\delta) \): cost of moving \(i\)-th part by \(\delta \in \Delta\)

score of \(i\)-th part at displaced location
Object hypothesis score

\[ \text{score}(\omega, \delta_1, \ldots, \delta_n) = m_0(\omega) + \sum_{i=1}^{n} m_i(a_i(\omega) + \delta_i) - d_i(\delta_i) \]

- \( m_0(\omega) \): score of the hypothesis at \( \omega \in \Omega \)
- \( m_i(\omega) \): score of the \( i \)-th part at \( \omega \in \Omega \)
- \( \delta_i \): part displacement
- \( d_i(\delta) \): cost of moving the \( i \)-th part by \( \delta \in \Delta \)
- \( \Omega \): set of \((x, y, scale)\) part locations
- \( \Delta \): set of \((dx, dy)\) part displacements
Root location score

\[
\text{score}(\omega) = m_0(\omega) + \sum_{i=1}^{n} \text{score}_i(a_i(\omega))
\]

\[
\text{score}_i(\eta) = \max_{\delta_i \in \Delta} (m_i(\eta + \delta_i) - d_i(\delta_i))
\]

Maximize over part displacements
Root location score

\[
\text{score}(\omega) = m_0(\omega) + \sum_{i=1}^{n} \text{score}_i(a_i(\omega))
\]

\[
\text{score}_i(\eta) = \max_{\delta_i \in \Delta} (m_i(\eta + \delta_i) - d_i(\delta_i))
\]

anchor position of \(i\)-th part

Maximize over part displacements
Root location score

\[
\text{score}(\omega) = m_0(\omega) + \sum_{i=1}^{n} \text{score}_i(a_i(\omega))
\]

\[
\text{score}_i(\eta) = \max_{\delta_i \in \Delta} (m_i(\eta + \delta_i) - d_i(\delta_i))
\]

optimal appearance/displacement tradeoff

Maximize over part displacements
Object detection

Detection by thresholding score(ω)

Baseline algorithm: \( O(pn|Ω|) \)

Using fast distance transforms + dynamic programming

\[ |Ω| \text{ is huge} \]
\[ p, \text{ cost to compute } m_i(ω), \text{ is expensive} \]

Bottleneck in practice

Use a cascade to compute \( m_i(ω) \) in fewer locations
Our object models

- root filters
- 8 part filters
- deformation costs

comp. 1

comp. 2

comp. 3

mixture of 3 left-right asymmetric star models
Star-cascade ingredients

1. A hierarchy of models defined by a part ordering

2. A sequence of thresholds: $t = ((t'_1, t_1), \ldots, (t'_n, t_n))$

\[
\forall \delta_1 : m_0(\omega) - d_1(a_1(\omega) \oplus \delta_1) \leq t'_1 \quad \rightarrow \text{prune } \omega
\]

\[
m_0(\omega) - d_1(a_1(\omega) \oplus \delta_1^*) + m_1(a_1(\omega) \oplus \delta_1^*) \leq t_2 \quad \rightarrow \text{prune } \omega
\]

\[
\forall \delta_2 : m_0(\omega) - d_1(a_1(\omega) \oplus \delta_1^*) + m_1(a_1(\omega) \oplus \delta_1^*) - d_2(a_2(\omega) \oplus \delta_2) \leq t'_2 \quad \rightarrow \text{prune } \delta_2
\]

\vdots
Star-cascade algorithm

test image

object model
+ part ordering
+ thresholds
Star-cascade algorithm

HOG pyramid from test image

object model + part ordering + thresholds
Star-cascade algorithm

HOG pyramid from test image

object model + part order + thresholds
Star-cascade algorithm

filter score tables

Root
$m_0(\omega)$

Part 1
$m_1(\omega)$

Part 2
$m_2(\omega)$

cascade test:

model:

operation:
Star-cascade algorithm

filter score tables

Root
\[ m_0(\omega) \]

Part 1
\[ m_1(\omega) \]

Part 2
\[ m_2(\omega) \]

cascade test:

model:

operation:
Star-cascade algorithm

filter score tables

Root
$m_0(\omega)$

Part 1
$m_1(\omega)$

Part 2
$m_2(\omega)$

cascade test:

model:

operation: test root locations
Star-cascade algorithm

filter score tables

Root

\( m_0(\omega) \)

Part 1

\( m_1(\omega) \)

Part 2

\( m_2(\omega) \)

cascade test:

model:

operation: test root locations
Star-cascade algorithm

filter score tables

Root

\[ m_0(\omega) \]

Part 1

\[ m_1(\omega) \]

cascade test: \[ m_0(\omega) \geq t_1 \]

Part 2

\[ m_2(\omega) \]

model: filter score tables

operation: test root locations

result: fail
Star-cascade algorithm

filter score tables

Root
\( m_0(\omega) \)

Part 1
\( m_1(\omega) \)

cascade test: \( m_0(\omega) \geq t_1 \)

Part 2
\( m_2(\omega) \)

model: 
operation: test root locations
result: fail
Star-cascade algorithm

filter score tables

<table>
<thead>
<tr>
<th>Root</th>
<th>Part 1</th>
<th>Part 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_0(\omega)$</td>
<td>$m_1(\omega)$</td>
<td>$m_2(\omega)$</td>
</tr>
</tbody>
</table>

cascade test: $m_0(\omega) \geq t_1$

model:

operation: test root locations  result: fail
# Star-cascade algorithm

<table>
<thead>
<tr>
<th>Root</th>
<th>( m_0(\omega) )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>filter score tables</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part 1</th>
<th>( m_1(\omega) )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part 2</th>
<th>( m_2(\omega) )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**cascade test:** \( m_0(\omega) \geq t_1 \)

**model:**

**operation:** test root locations

**result:** fail
Star-cascade algorithm

filter score tables

Root
\( m_0(\omega) \)

Part 1
\( m_1(\omega) \)

Part 2
\( m_2(\omega) \)

cascade test: \( m_0(\omega) \geq t_1 \)

model: [images]

operation: test root locations

result: fail
Star-cascade algorithm

filter score tables

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Root
$m_0(\omega)$

Part 1
$m_1(\omega)$

Part 2
$m_2(\omega)$

cascade test: $m_0(\omega) \geq t_1$

model:

operation: test root locations

result: fail
Star-cascade algorithm

filter score tables

Root
$m_0(\omega)$

Part 1
$m_1(\omega)$

Part 2
$m_2(\omega)$

cascade test: $m_0(\omega) \geq t_1$

model:

operation: test root locations

result: fail
Star-cascade algorithm

filter score tables

Root
\( m_0(\omega) \)

Part 1
\( m_1(\omega) \)

Part 2
\( m_2(\omega) \)

cascade test: \( m_0(\omega) \geq t_1 \)

model:

operation: test root locations

result: fail
Star-cascade algorithm

- **Root**
  - $m_0(\omega) \geq t_1$

- **Part 1**
  - $m_1(\omega)$

- **Part 2**
  - $m_2(\omega)$

**filter score tables**

**cascade test:** $m_0(\omega) \geq t_1$

**model:**
- [filtered image]

**operation:** test root locations

**result:** pass
Star-cascade algorithm

filter score tables

**Root**

\[ m_0(\omega) \]

**Part 1**

\[ m_1(\omega) \]

**Part 2**

\[ m_2(\omega) \]

cascade test: \[ m_0(\omega) - d_1(\delta_1) \geq t'_1 \]

model:

operation: displacement search
**Star-cascade algorithm**

**Filter score tables**

- **Root**
  \[ m_0(\omega) \]

- **Part 1**
  \[ m_1(\omega) \]

- **Part 2**
  \[ m_2(\omega) \]

**Cascade test:**
\[ m_0(\omega) - d_1(\delta_1) \geq t'_1 \]

**Model:**

**Operation:** displacement search
Star-cascade algorithm

filter score tables

Root
\[ m_0(\omega) \]

Part 1
\[ m_1(\omega) \]

cascade test:
\[ m_0(\omega) - d_1(\delta_1) \geq t_1' \]

model:

operation: displacement search
Star-cascade algorithm

Filter score tables

Root
\[ m_0(\omega) \]

Part 1
\[ m_1(\omega) \]

Part 2
\[ m_2(\omega) \]

cascade test: \[ m_0(\omega) - d_1(\delta_1) \geq t'_1 \]

model:

operation: displacement search

result: pass
Star-cascade algorithm

filter score tables

Root
$m_0(\omega)$

Part 1
$m_1(\omega)$

cascade test: $m_0(\omega) - d_1(\delta_1^*) + m_1(\omega \oplus \delta_1^*) \geq t_2$

Part 2
$m_2(\omega)$

model:

operation: test partial score

result: fail
Star-cascade algorithm

filter score tables

Root
$m_0(\omega)$

Part 1
$m_1(\omega)$

Part 2
$m_2(\omega)$

cascade test: $m_0(\omega) \geq t_1$

model:

operation: test root locations

result: pass
Star-cascade algorithm

filter score tables

Root

\[ m_0(\omega) \]

Part 1

\[ m_1(\omega) \]

cascade test: \[ m_0(\omega) \geq t_1 \]

Part 2

\[ m_2(\omega) \]

model: 

operation: test root locations

result: pass
Star-cascade algorithm

Filter score tables

Root
\[ m_0(\omega) \]

Part 1
\[ m_1(\omega) \]

Part 2
\[ m_2(\omega) \]

cascade test: \[ m_0(\omega) - d_1(\delta_1) \geq t'_1 \]

model:

operation: displacement search
Star-cascade algorithm

filter score tables

Root
\[ m_0(\omega) \]

Part 1
\[ m_1(\omega) \]

cascade test: \[ m_0(\omega) - d_1(\delta_1) \geq t'_1 \]

Part 2
\[ m_2(\omega) \]

model:

operation: displacement search
Star-cascade algorithm

filter score tables

Root
\(m_0(\omega)\)

Part 1
\(m_1(\omega)\)
cached!

Part 2
\(m_2(\omega)\)

cascade test: \(m_0(\omega) - d_1(\delta_1) \geq t'_1\)

model:

operation: displacement search

result: pass
Star-cascade algorithm

- Filter score tables
- Root: \( m_0(\omega) \)
- Part 1: \( m_1(\omega) \)
- Part 2: \( m_2(\omega) \)

Cascade test: 
\[
m_0(\omega) - d_1(\delta_1^*) + m_1(\omega \oplus \delta_1^*) \geq t_2
\]

Model: 
- operation: test partial score
- result: pass
Star-cascade algorithm

**Root**

\[ m_0(\omega) \]

**Part 1**

\[ m_1(\omega) \]

**Part 2**

\[ m_2(\omega) \]

**Filter score tables**

**Cascade test:**

\[ m_0(\omega) - d_1(\delta_1^*) + m_1(\omega \oplus \delta_1^*) - d_2(\delta_2) \geq t_3 \]

**Model:**

**Operation:** displacement search
Star-cascade algorithm

Filter score tables

Root

\[ m_0(\omega) \]

Part 1

\[ m_1(\omega) \]

Part 2

\[ m_2(\omega) \]

cascade test:

\[ m_0(\omega) - d_1(\delta^*_1) + m_1(\omega \oplus \delta^*_1) - d_2(\delta_2) \geq t_3 \]

Model:

Operation: displacement search
Result: pass
Star-cascade algorithm

filter score tables

Root
\[ m_0(\omega) \]

Part 1
\[ m_1(\omega) \]

Part 2
\[ m_2(\omega) \]

cascade test:
\[ m_0(\omega) - d_1(\delta_1^*) + m_1(\omega \oplus \delta_1^*) - d_2(\delta_2) \geq t_3 \]

model:

operation: displacement search
result: pass
Star-cascade algorithm

filter score tables

Root
\[ m_0(\omega) \]

Part 1
\[ m_1(\omega) \]

Part 2
\[ m_2(\omega) \]

cascade test:
\[ m_0(\omega) - d_1(\delta_1^*) + m_1(\omega \oplus \delta_1^*) - d_2(\delta_2^*) + m_2(\omega \oplus \delta_2^*) \geq t_3 \]

model:

operation: test partial score
result: pass
Star-cascade algorithm

filter score tables

Root
$m_0(\omega)$

Part 1
$m_1(\omega)$

Part 2
$m_2(\omega)$

cascade test: ...

model:

operation: continue testing remaining parts
Star-cascade algorithm

filter score tables

Root
$m_0(\omega)$

Part 1
$m_1(\omega)$

Part 2
$m_2(\omega)$

cascade test: all tests passed $\Rightarrow$ detection!

model:

operation: report object hypothesis
Star-cascade algorithm

filter score tables

Root
\[ m_0(\omega) \]

Part 1
\[ m_1(\omega) \]

Part 2
\[ m_2(\omega) \]

cascade test:
model:
operation: continue with root locations...
Threshold selection

We want *safe* and *effective* thresholds

---

don’t prune many true positives

but do prune lots of true negatives
PAA thresholds

\[ X = \text{IID set of positive examples } \sim D \]

error(\( t \)) = \( P_{x \sim D}(\text{cascade-score}(t, \omega) \neq \text{score}(\omega)) \)

**Probably Approximately Admissible thresholds**

provably safe \( \rightarrow \) \( P(\text{error}(t) > \epsilon) \leq \delta \) empirically effective

min of partial scores over examples in \( X \)

Theorem: \(|X| \geq 2n/\epsilon \ln(2n/\delta) \implies (\epsilon, \delta)-\text{PAA thresholds}\)
Example results

- **High recall**
  - PASCAL 2007 comp3 class: motorbike
  - Baseline (AP 48.7)
  - Cascade (AP 48.9)

- **Less recall ⇒ faster**
  - PASCAL 2007 comp3 class: motorbike
  - Baseline (AP 48.7)
  - Cascade (AP 41.8)

**23.2x faster**
(618ms per/image)

**31.6x faster**
(454ms per/image)
Simplified part models

- PCA of HOG features
- Project filters and features onto top 5 PCs (top 5 PCs account for ~ 90% of variance)
- Double number of cascade stages
  - 1st half: place PCA filters
  - 2nd half: replace PCA filters with full filters
- ~ 3x speedup (included in previous numbers)
Grammar models

- We focus on star models
  - simple algorithm & good PASCAL results
- We give a cascade algorithm for a general class of grammar models
  - trees with variable structure
  - but no shared parts
- future work: empirical evaluation
Conclusion

- A simple cascade algorithm for star models
  - ~ 15x speedup with no loss in AP scores
  - > 15x speedup with controlled recall sacrifice
  - parallel implementation ⇒ several frames per second

- Cascade for a general class of grammar models

- Detection is cheaper than scoring parts everywhere

- Get the source code from:
  http://www.cs.uchicago.edu/~rbg/cascade