Object detection and algorithms for efficient inference

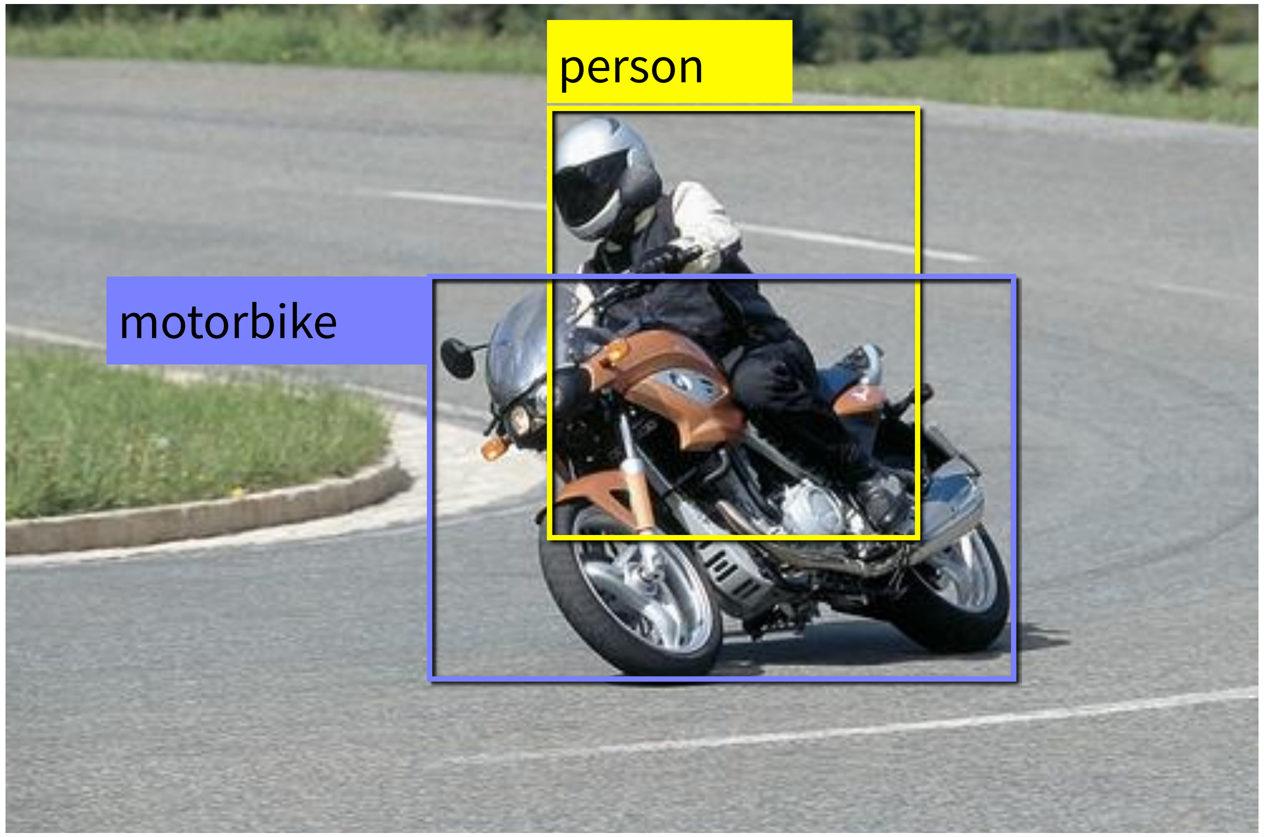
Hyun Oh Song
CS231M - Mobile computer vision
April 29, 2015

Contents

- Sliding window object detection
- Deformable part models
- Cascade DPM
- Sparselets
- Hashing based

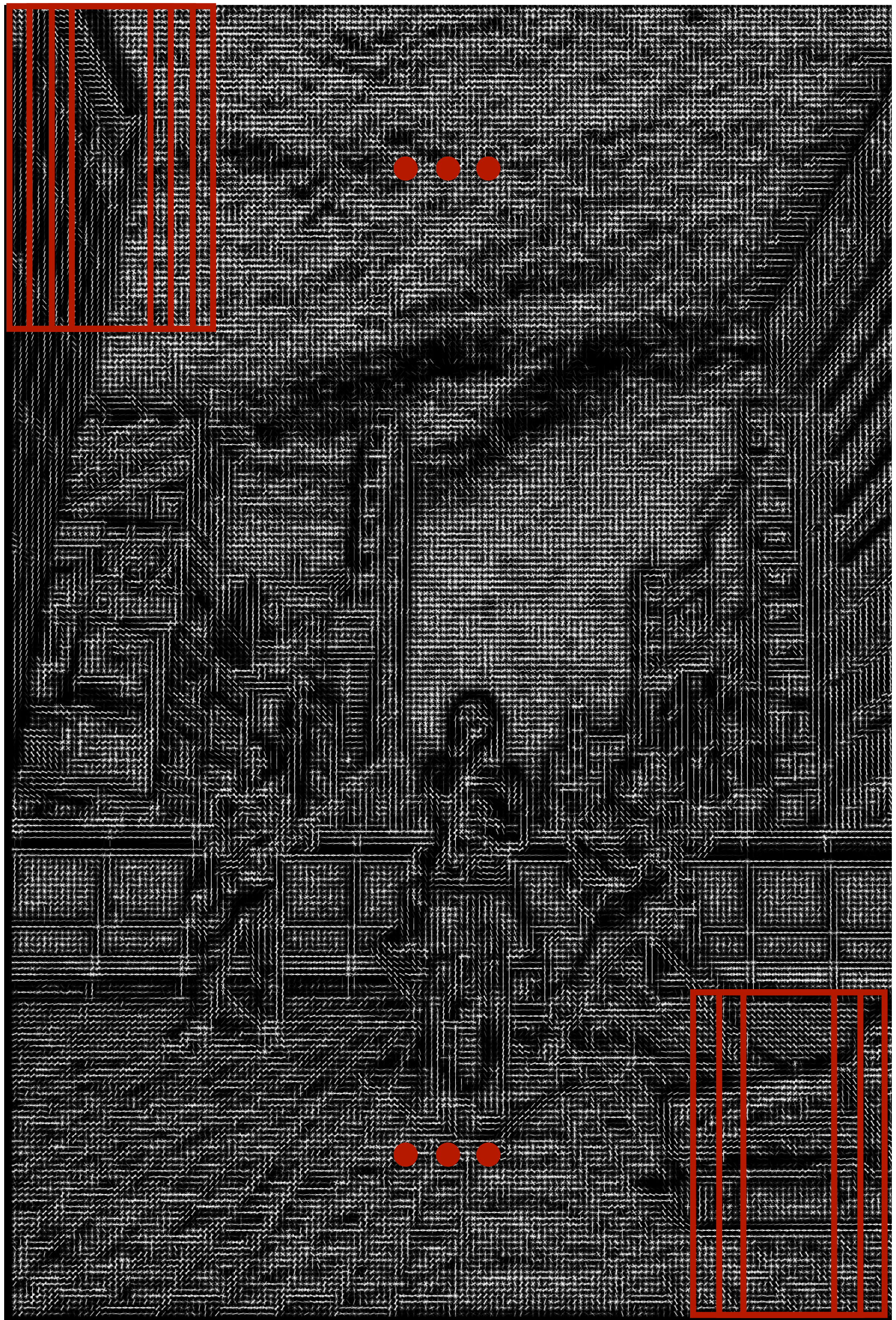
Background: Object Detection

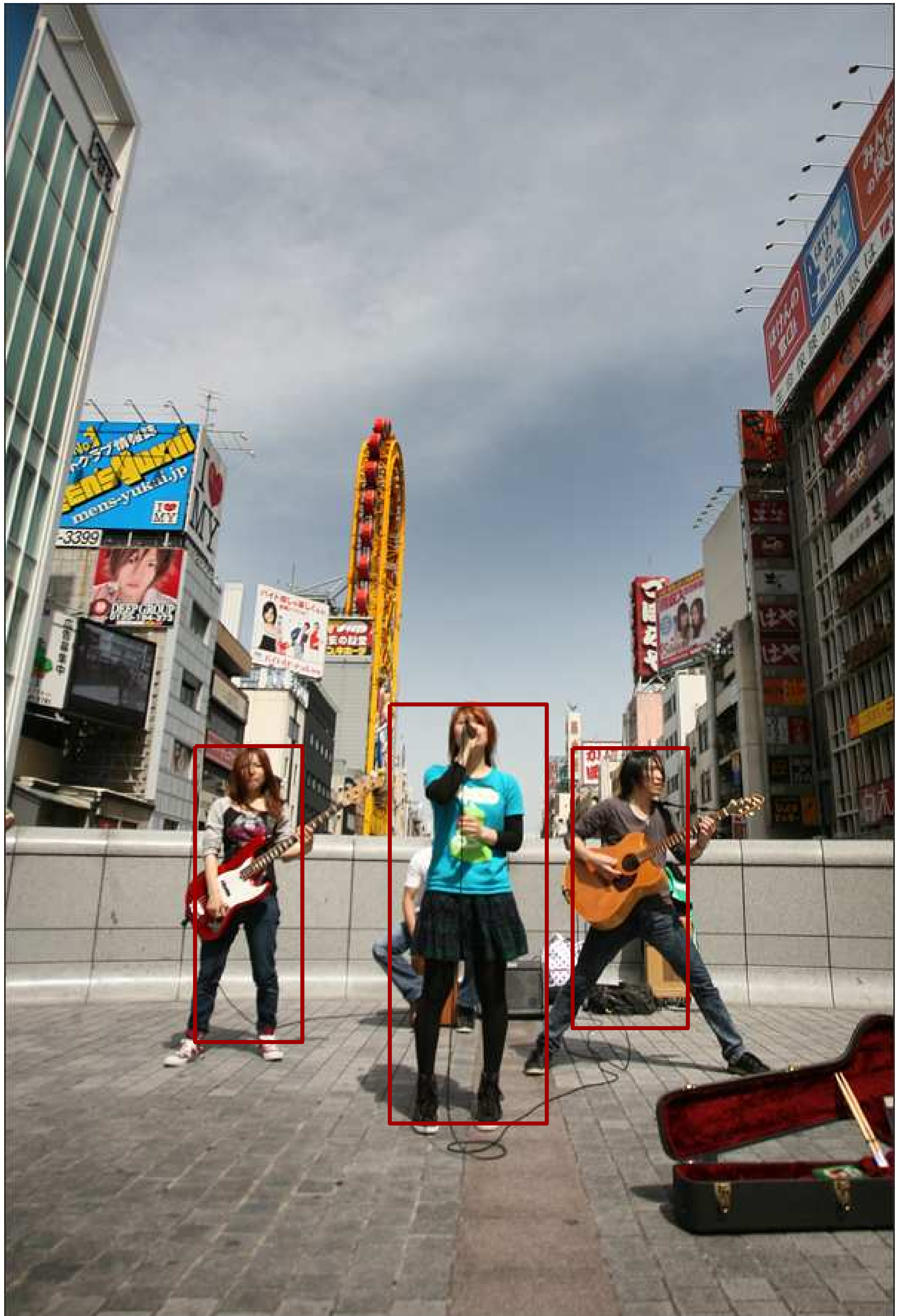
Input



Desired output

Sliding window classification

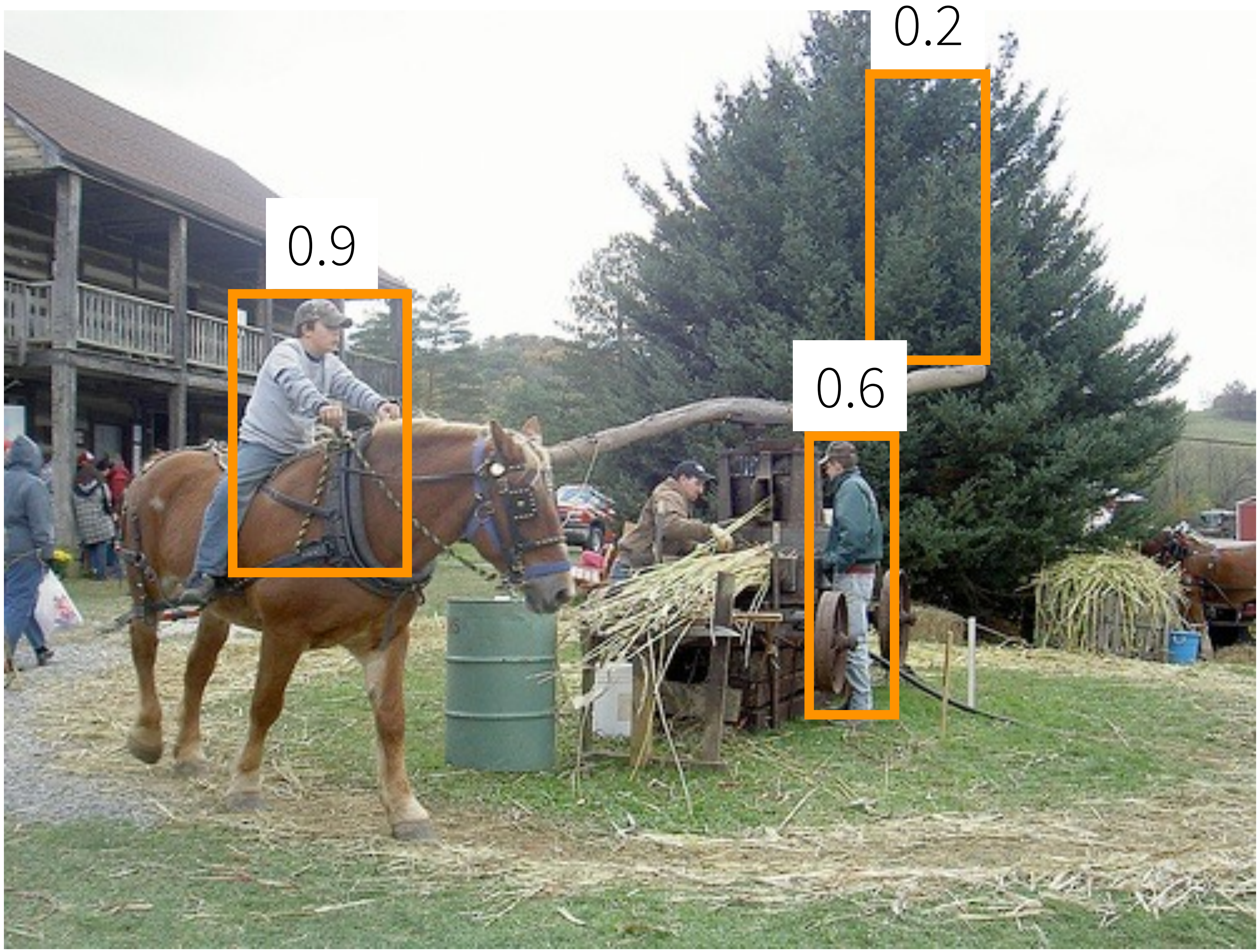




Evaluating a detector

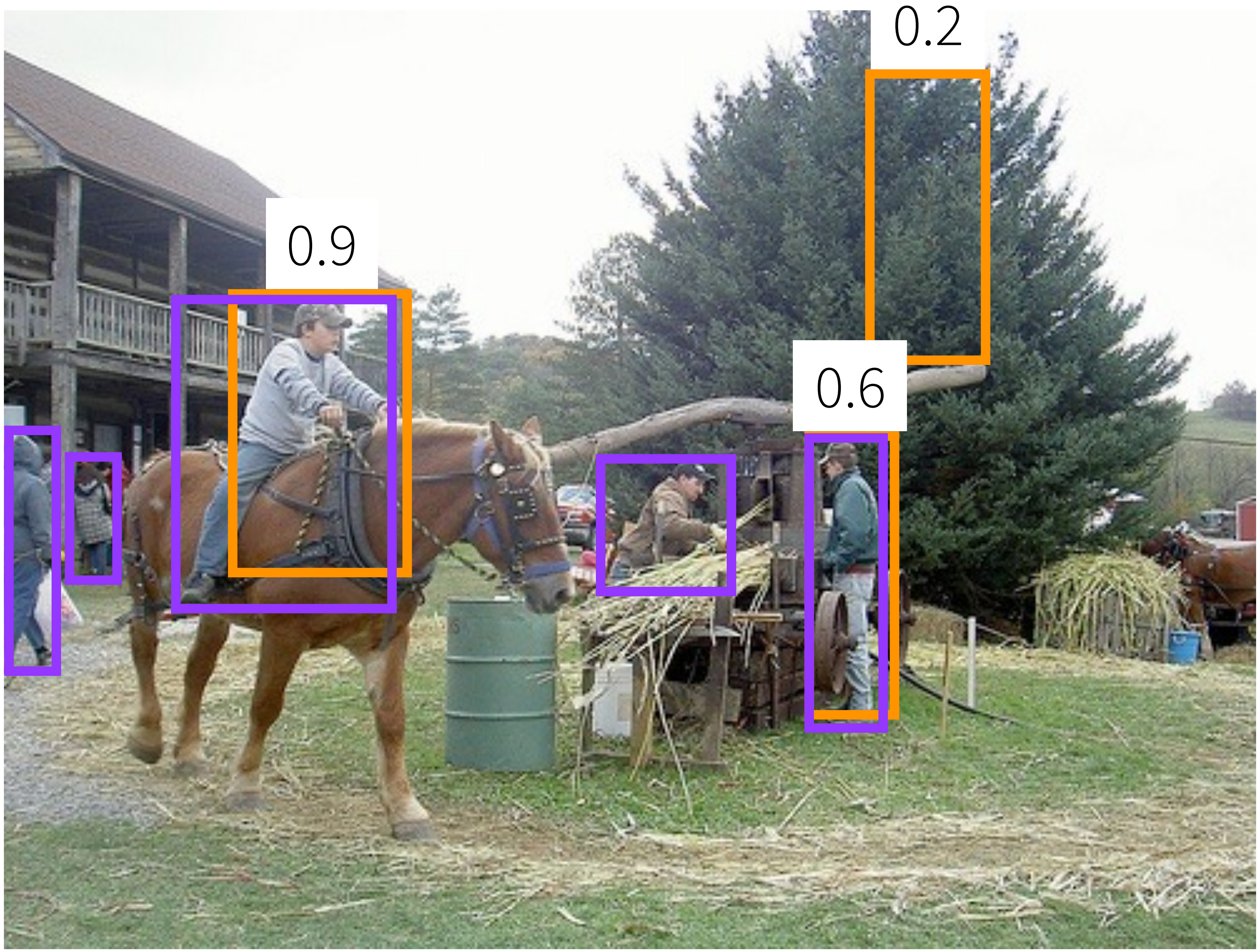
Test image (previously unseen)

Detections



□ ‘person’ detector predictions

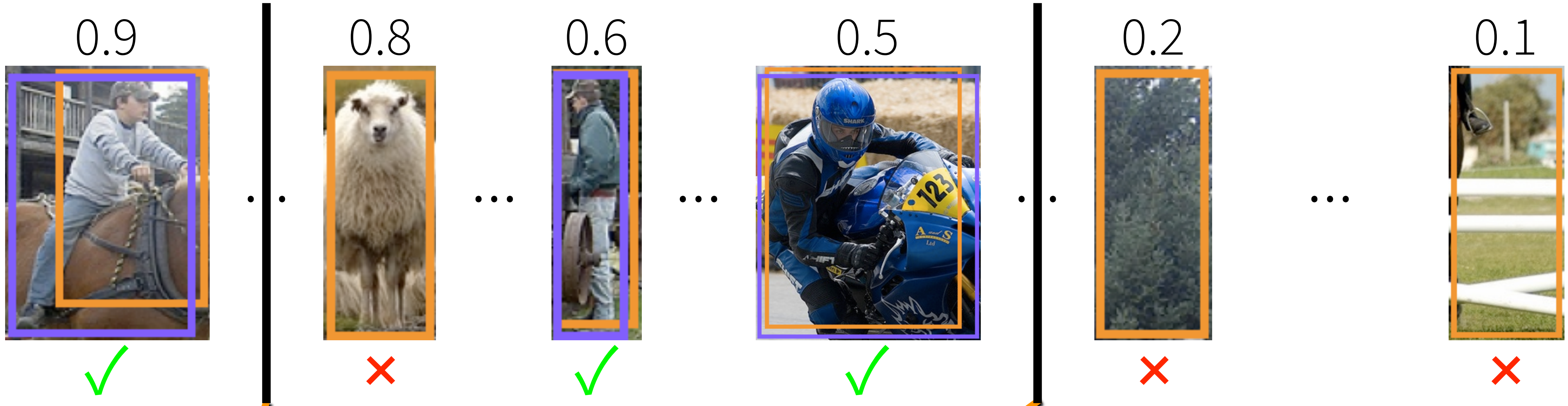
Compared to ground truth

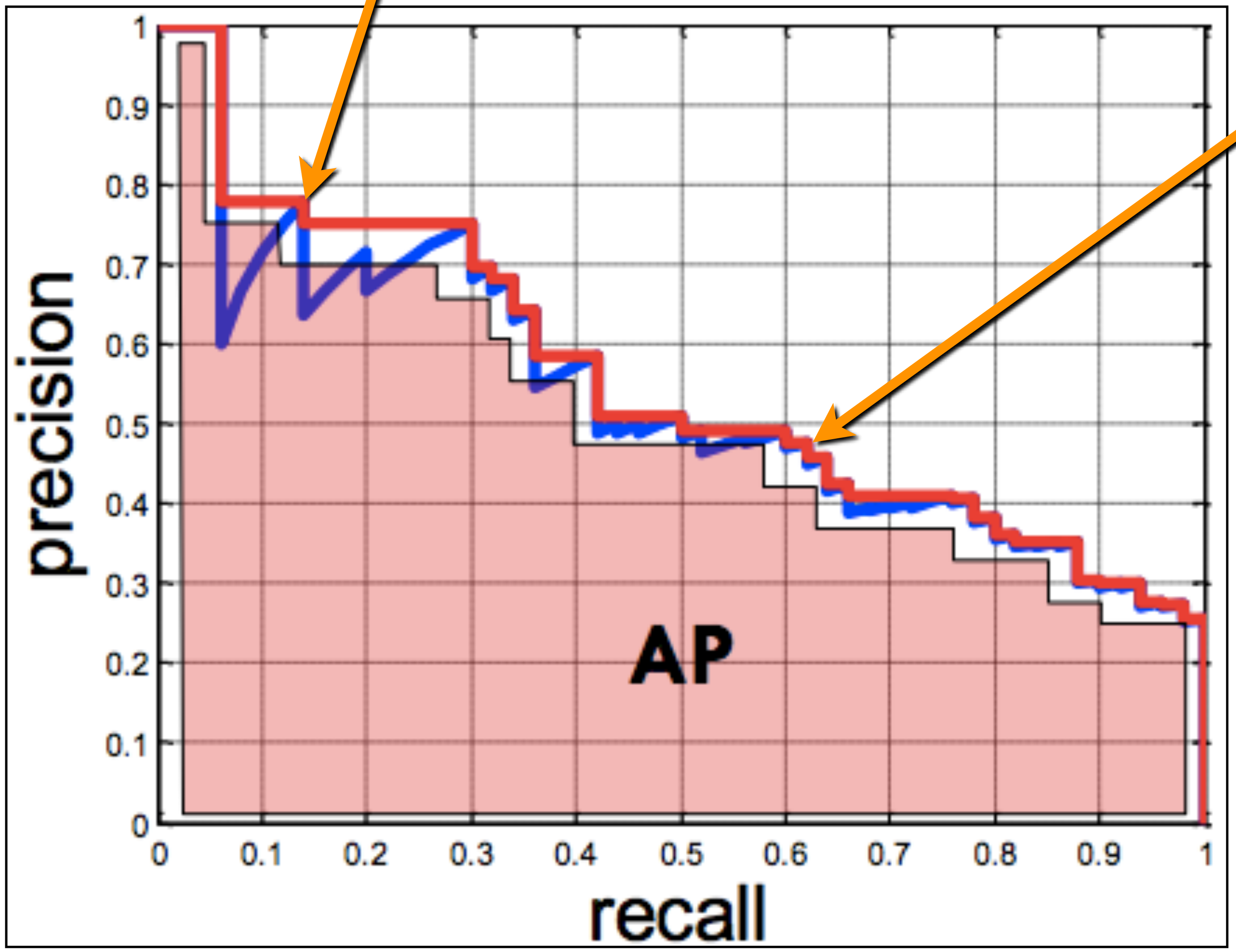


□ ‘person’ detector predictions

□ ground truth ‘person’ boxes

Evaluation metric = AP





Average Precision (**AP**)

0% is worst

100% is best

mean AP over classes (**mAP**)

PASCAL VOC Challenge

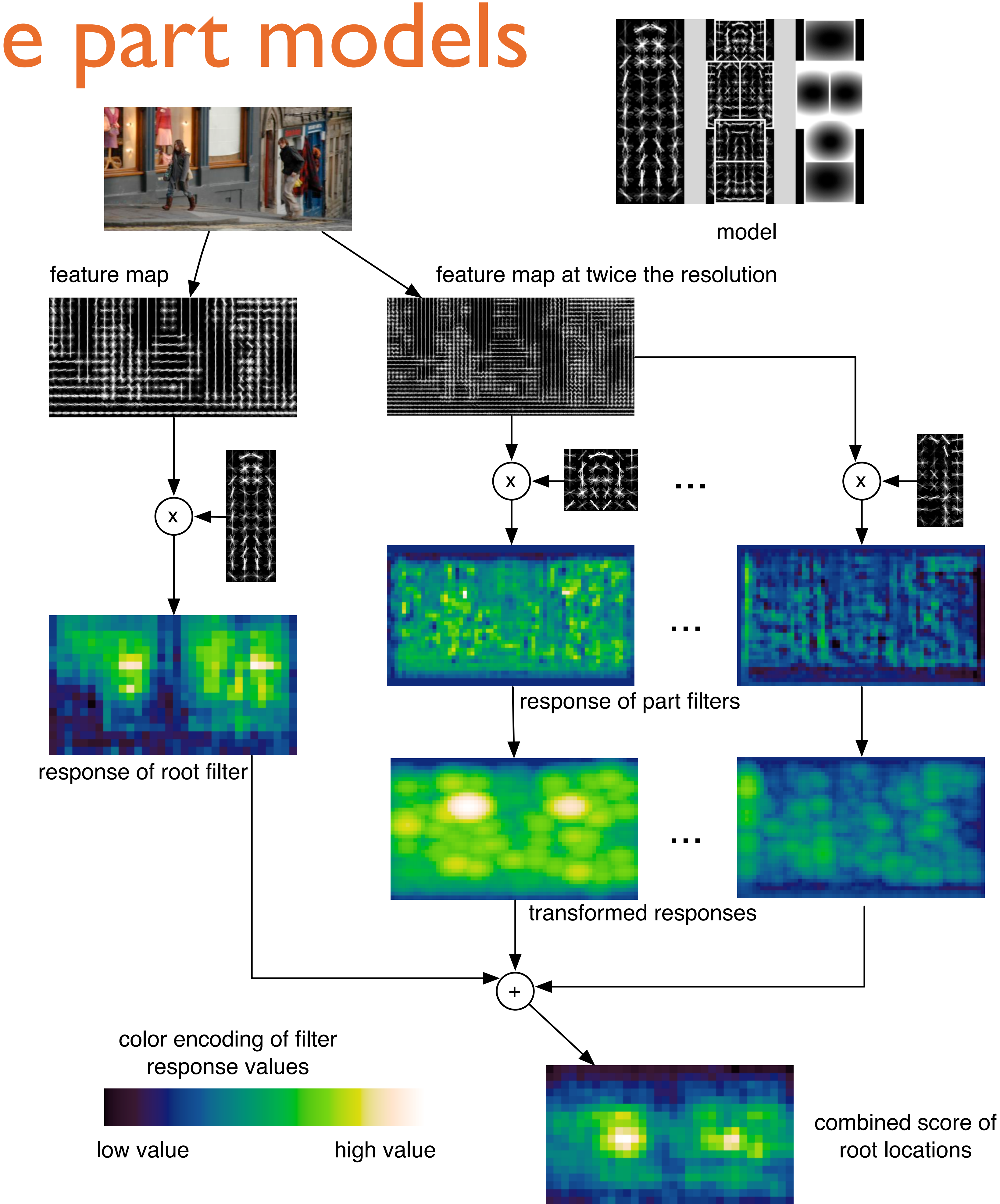
Dataset: 22k images, 50k objects, 20 classes

Detect: people, horses, sofas, bicycles, pottedplants, ...

Contents

- Sliding window object detection
- Deformable part models
- Cascade DPM
- Sparselets
- Hashing based

Deformable part models

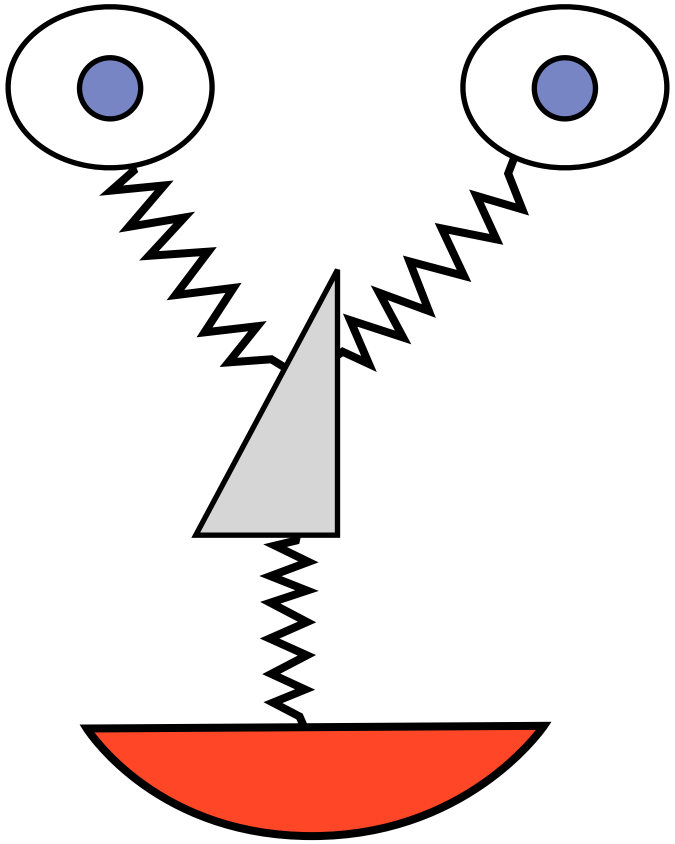


Felzenszwalb et al, PAMI 2010

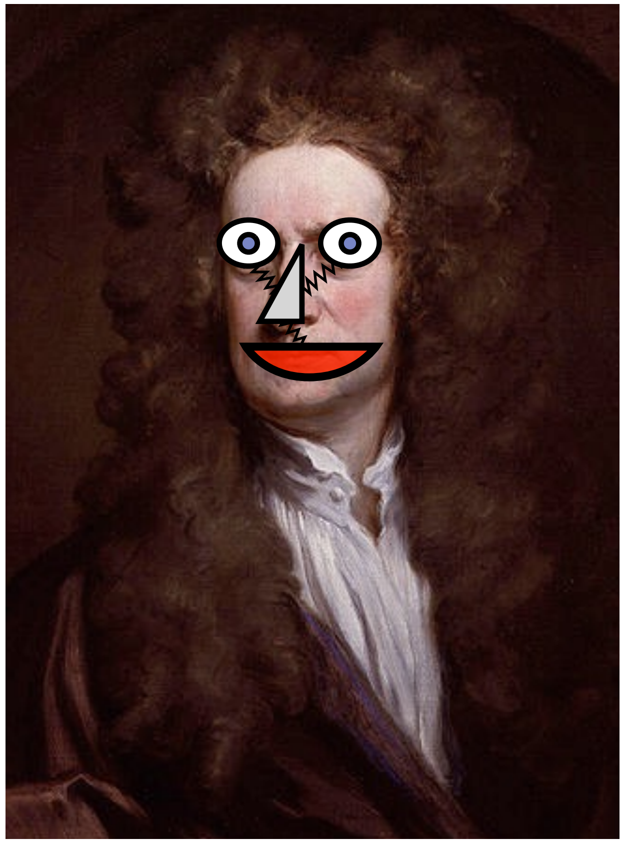
slide credit: Girshick et al

Star models

test image



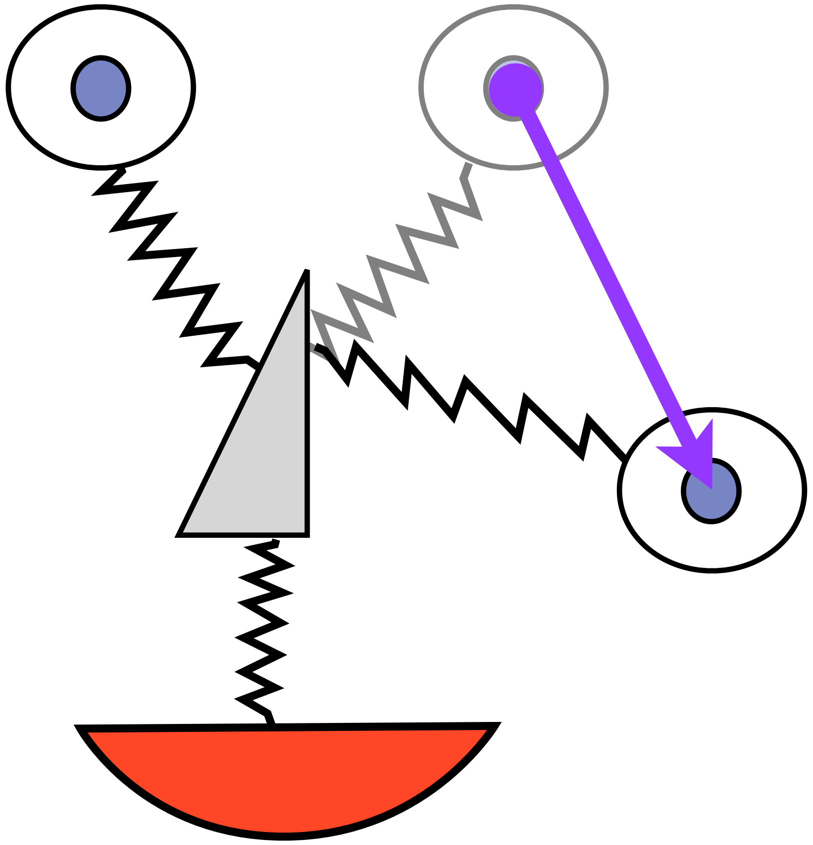
part-based deformable model



detection

slide credit: Girshick et al

Object hypothesis score



Ω set of $(x, y, scale)$ part locations

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

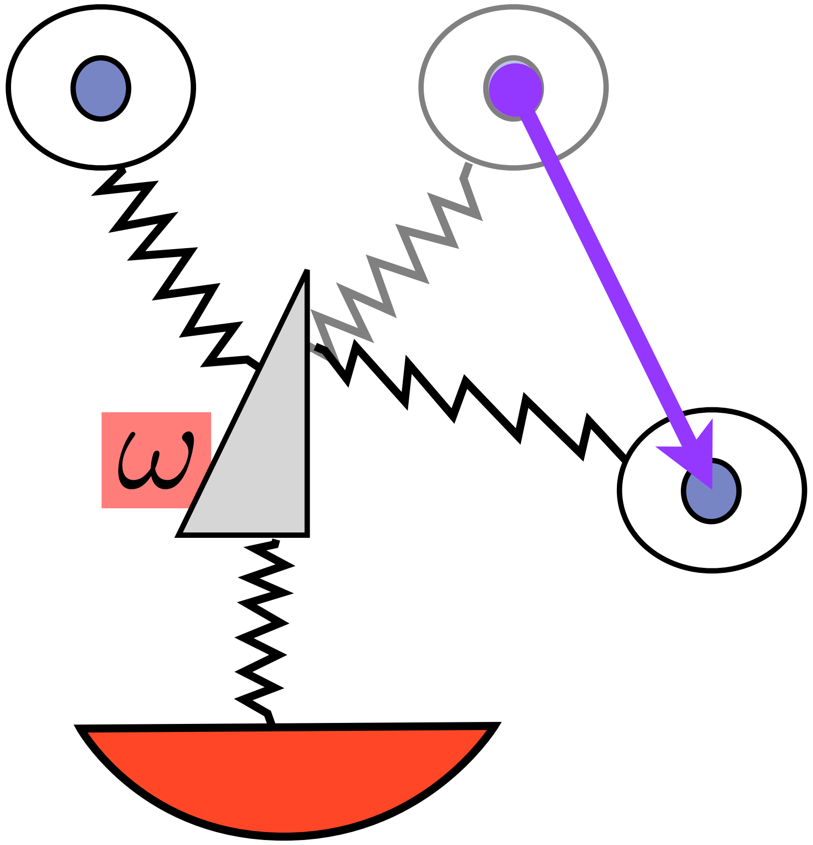
Δ set of (dx, dy) part displacements

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

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

slide credit: Girshick et al

Object hypothesis score



Ω set of $(x, y, scale)$ part locations

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

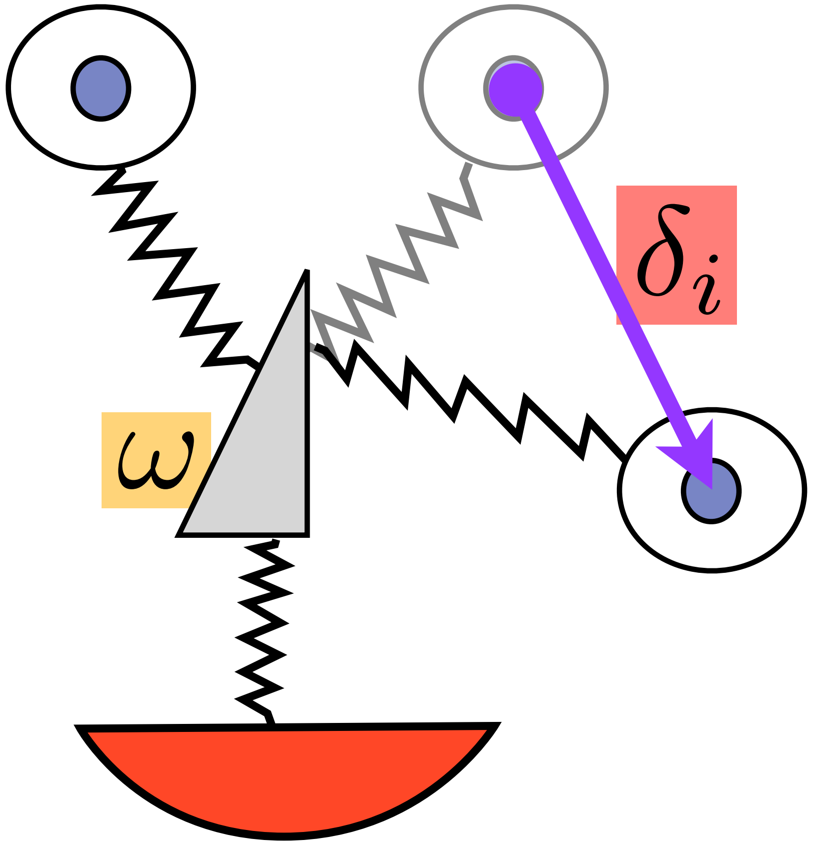
Δ set of (dx, dy) part displacements

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

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

slide credit: Girshick et al

Object hypothesis score



Ω set of $(x, y, scale)$ part locations

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

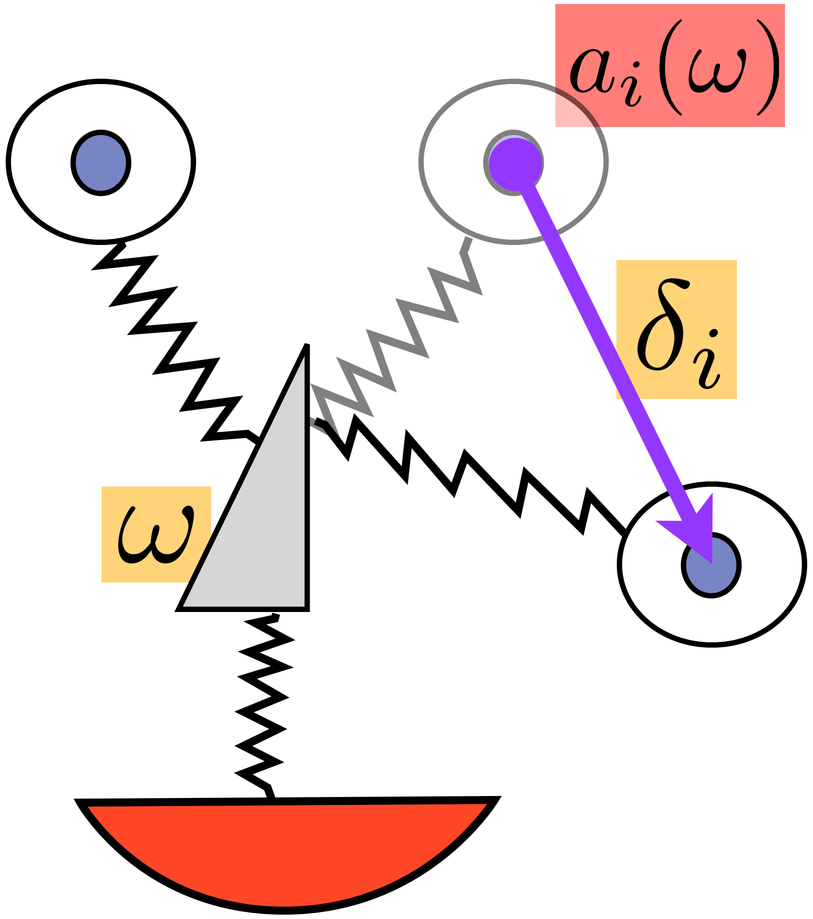
Δ set of (dx, dy) part displacements

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

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

slide credit: Girshick et al

Object hypothesis score



Ω set of $(x, y, scale)$ part locations

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

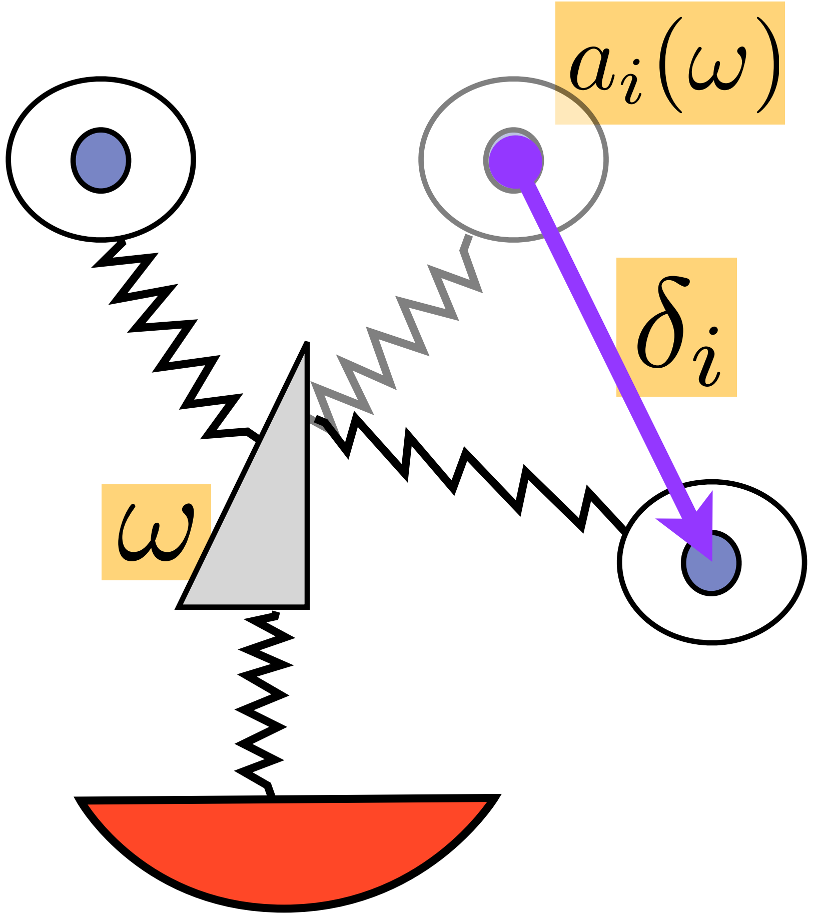
Δ set of (dx, dy) part displacements

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

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

slide credit: Girshick et al

Object hypothesis score



Ω set of $(x, y, scale)$ part locations

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

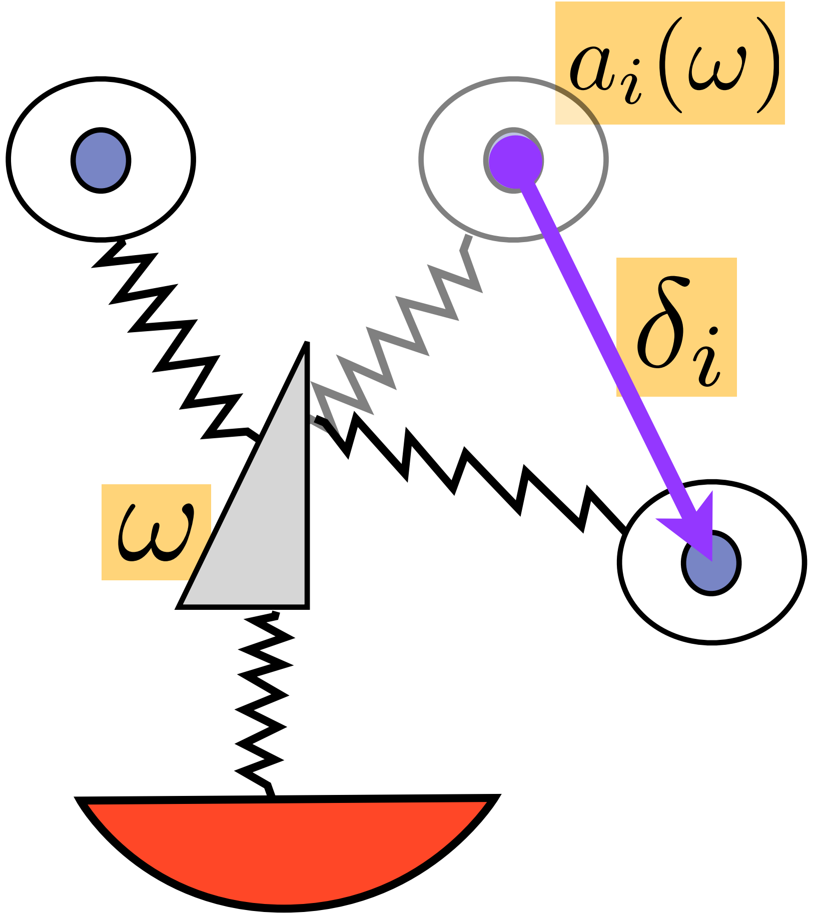
Δ set of (dx, dy) part displacements

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

$$\text{score}(\omega, \delta_1, \ldots, \delta_n) = \underbrace{m_0(\omega)}_{\text{score of root}} + \sum_{i=1}^{n} m_i(a_i(\omega) + \delta_i) - d_i(\delta_i)$$

slide credit: Girshick et al

Object hypothesis score



Ω set of $(x, y, scale)$ part locations

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

Δ set of (dx, dy) part displacements

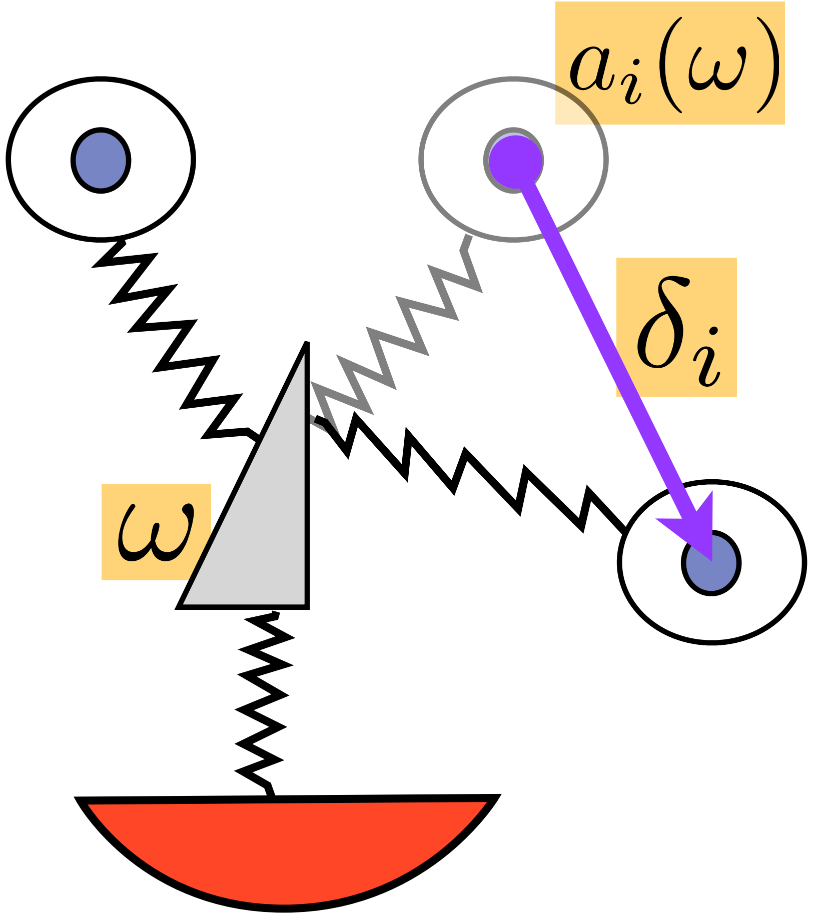
$d_i(\delta)$ cost of moving i-th part by $\delta \in \Delta$

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

sum over non-root parts

slide credit: Girshick et al

Object hypothesis score



Ω set of $(x, y, scale)$ part locations

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

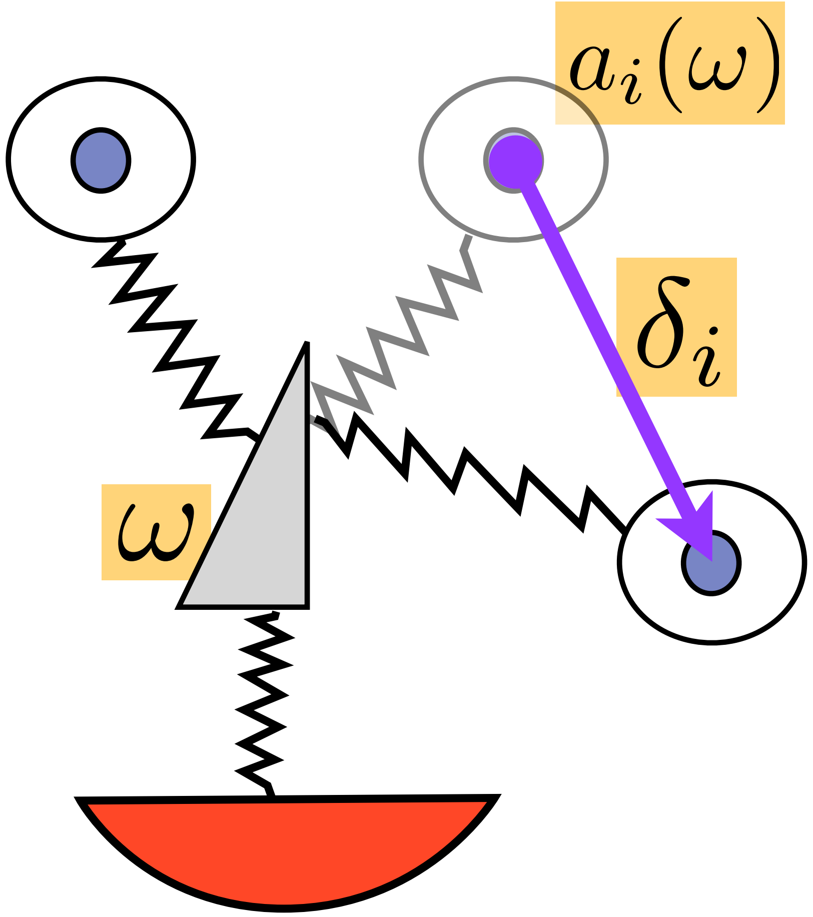
Δ set of (dx, dy) part displacements

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

$$\text{score}(\omega, \delta_1, \ldots, \delta_n) = m_0(\omega) + \sum_{i=1}^{n} \underbrace{m_i(a_i(\omega) + \delta_i)}_{\text{score of } i\text{-th part at displaced location}} - d_i(\delta_i)$$

slide credit: Girshick et al

Object hypothesis score



Ω set of $(x, y, scale)$ part locations

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

Δ set of (dx, dy) part displacements

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

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

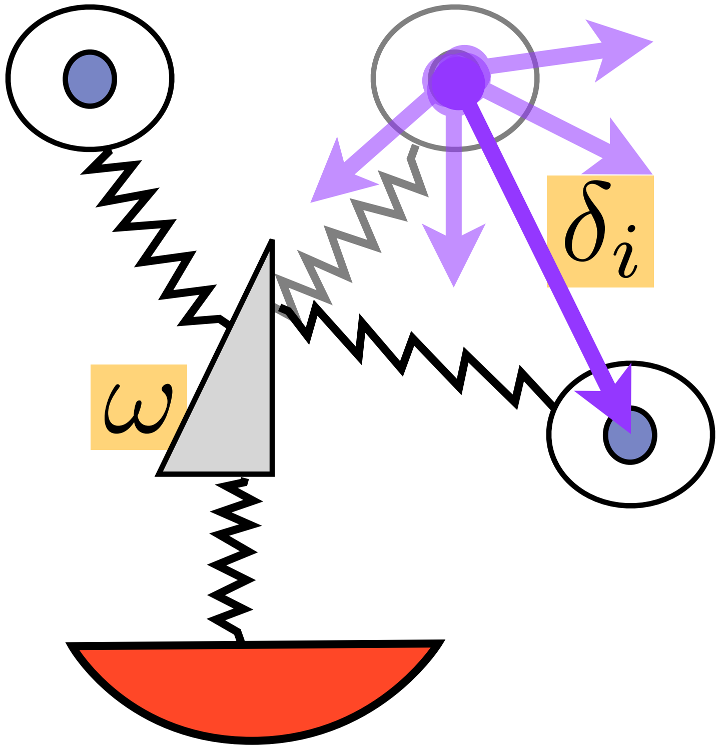
minus cost of i-th displacement

slide credit: Girshick et al

Object hypothesis score

$$\mathrm{score}(\omega) = m_0(\omega) + \sum_{i=1}^{n} \boxed{\mathrm{score}_i(a_i(\omega))}$$

$$\mathrm{score}_i(\eta) = \max_{\delta_i \in \Delta}(m_i(\eta + \delta_i) - d_i(\delta_i))$$



Maximize over part displacements

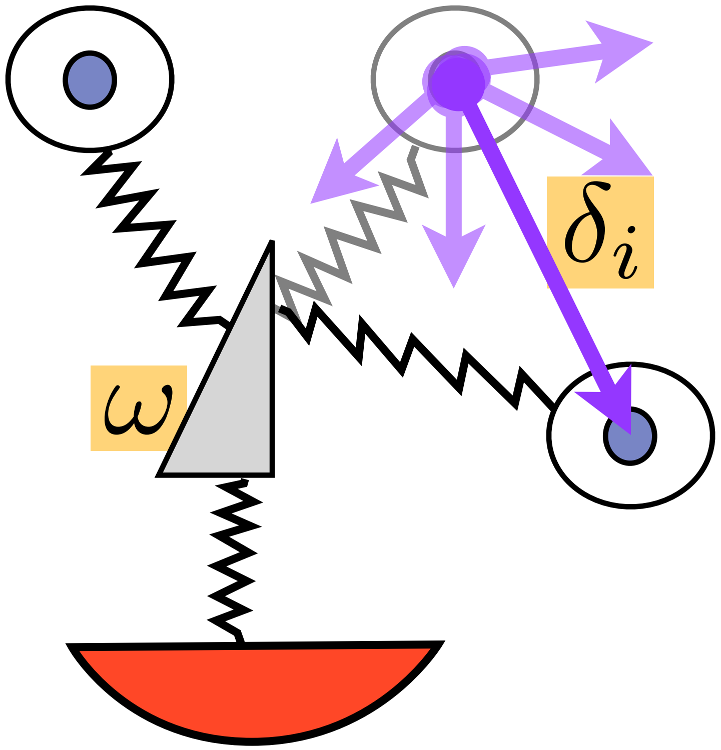
slide credit: Girshick et al

Object hypothesis 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

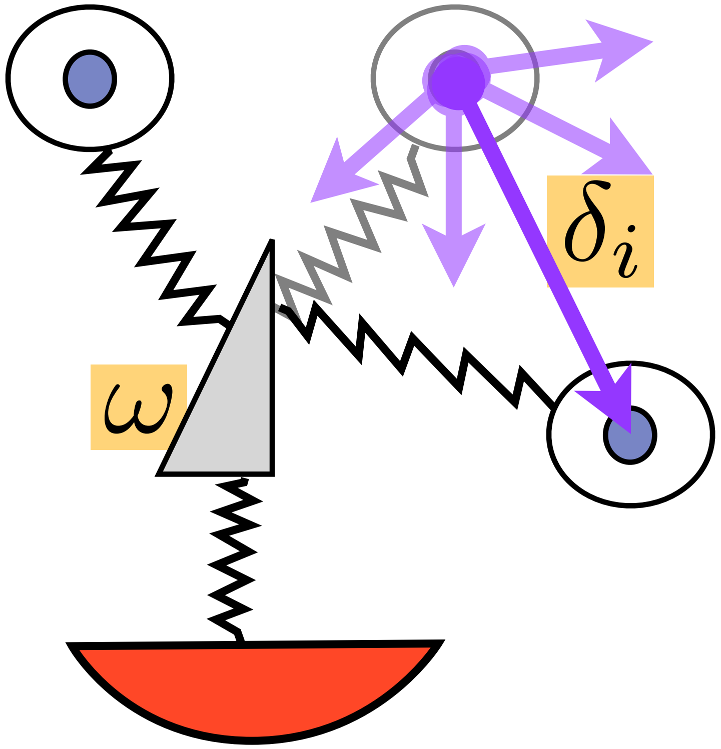
slide credit: Girshick et al

Object hypothesis 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

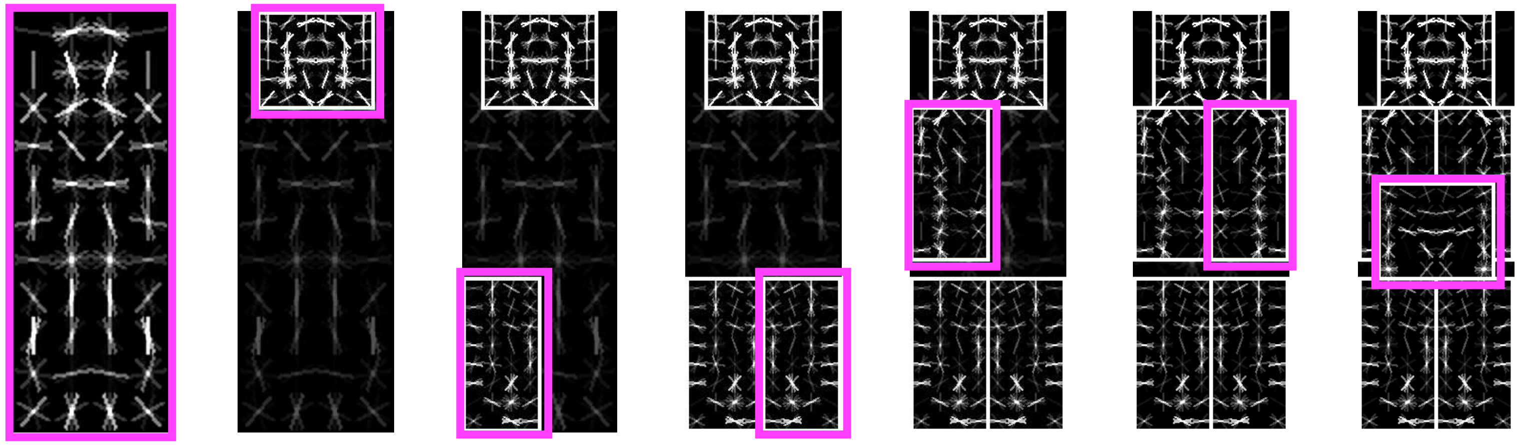
Contents

- Sliding window object detection
- Deformable part models
- Cascade DPM
- Sparselets

slide credit: Girshick et al

Star cascade ingredients

1. A hierarchy of models defined by a part ordering



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

$m_0(\omega) \overset{?}{\leq} t_1$ → prune ω

$\forall \delta_1 : \; m_0(\omega) - d_1(a_1(\omega) \oplus \delta_1) \overset{?}{\leq} t'_1$ → prune δ_1

$m_0(\omega) - d_1(a_1(\omega) \oplus \delta_1^*) + m_1(a_1(\omega) \oplus \delta_1^*) \overset{?}{\leq} t_2$ → prune ω

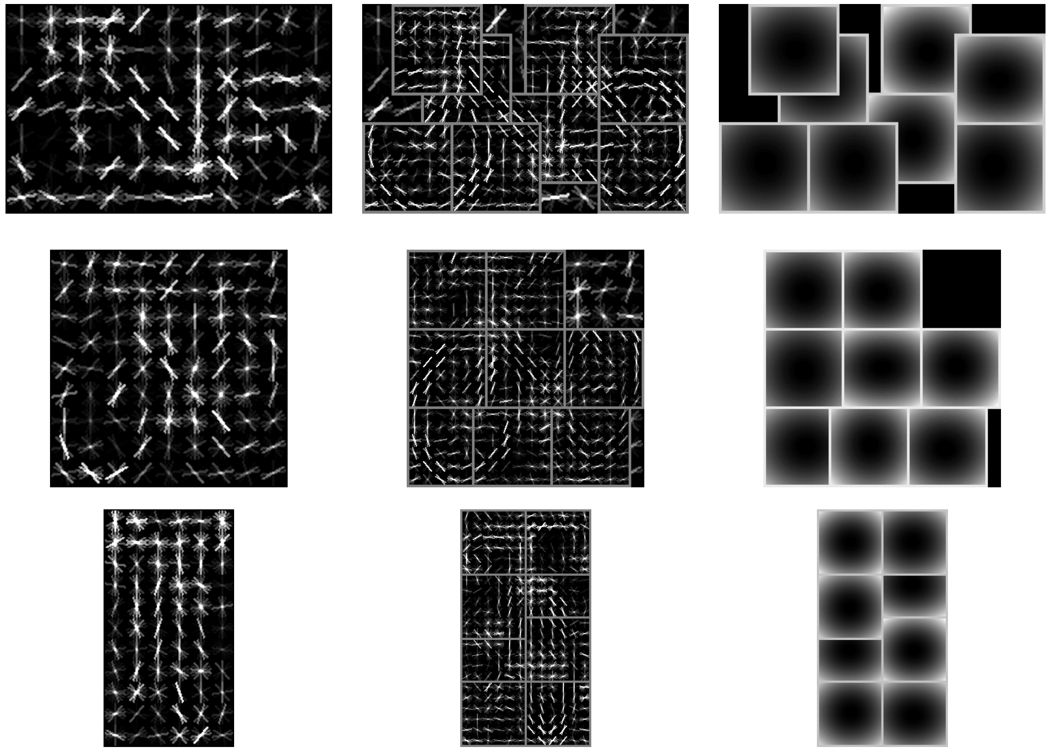
$\forall \delta_2 : \; m_0(\omega) - d_1(a_1(\omega) \oplus \delta_1^*) + m_1(a_1(\omega) \oplus \delta_1^*) - d_2(a_2(\omega) \oplus \delta_2) \overset{?}{\leq} t'_2$ → prune δ_2

$\vdots$

slide credit: Girshick et al

Star cascade algorithm

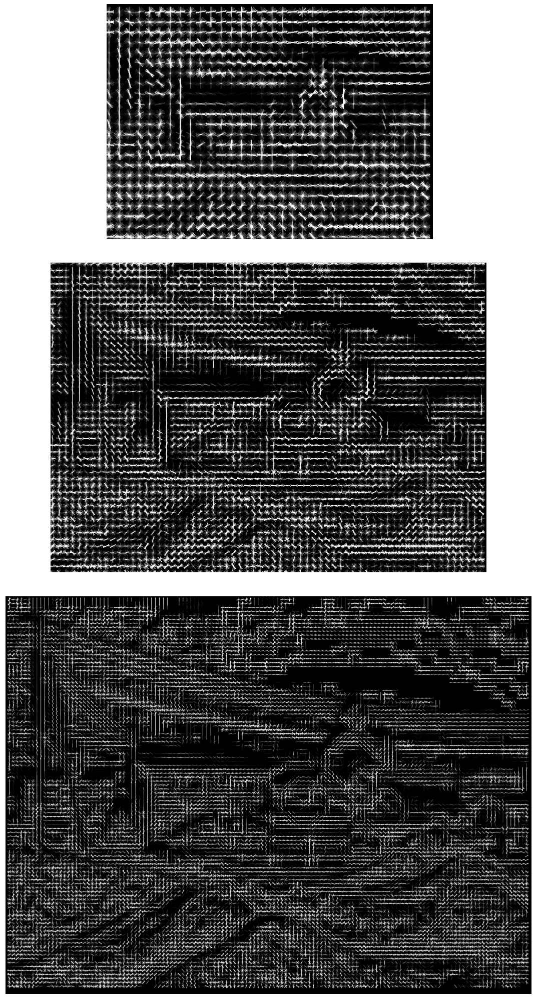
test image



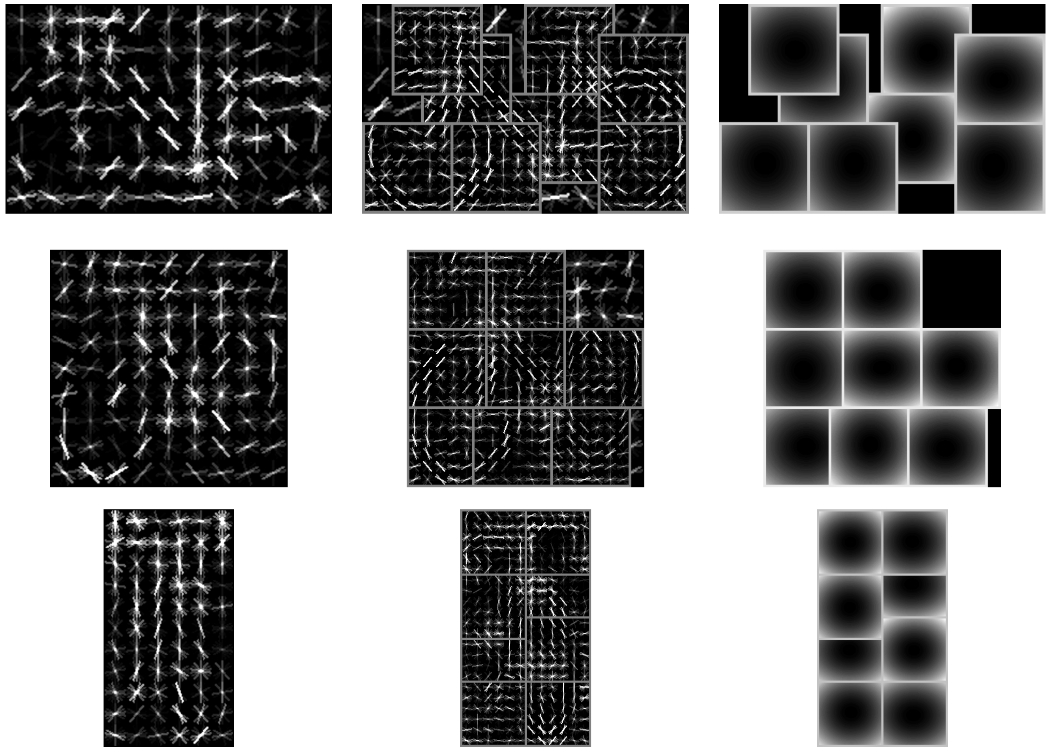
object model
+ part ordering
+ thresholds

slide credit: Girshick et al

Star cascade algorithm



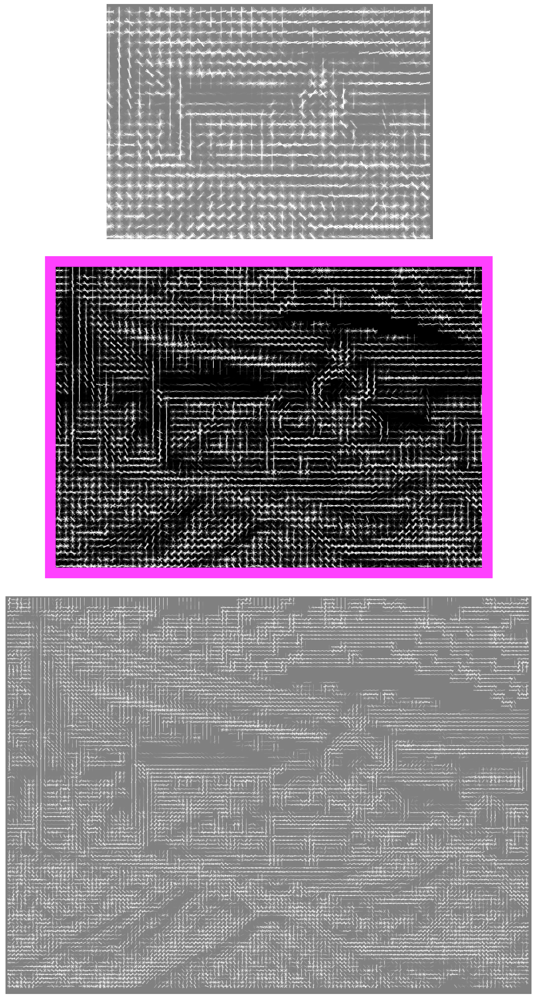
HOG pyramid
from test image



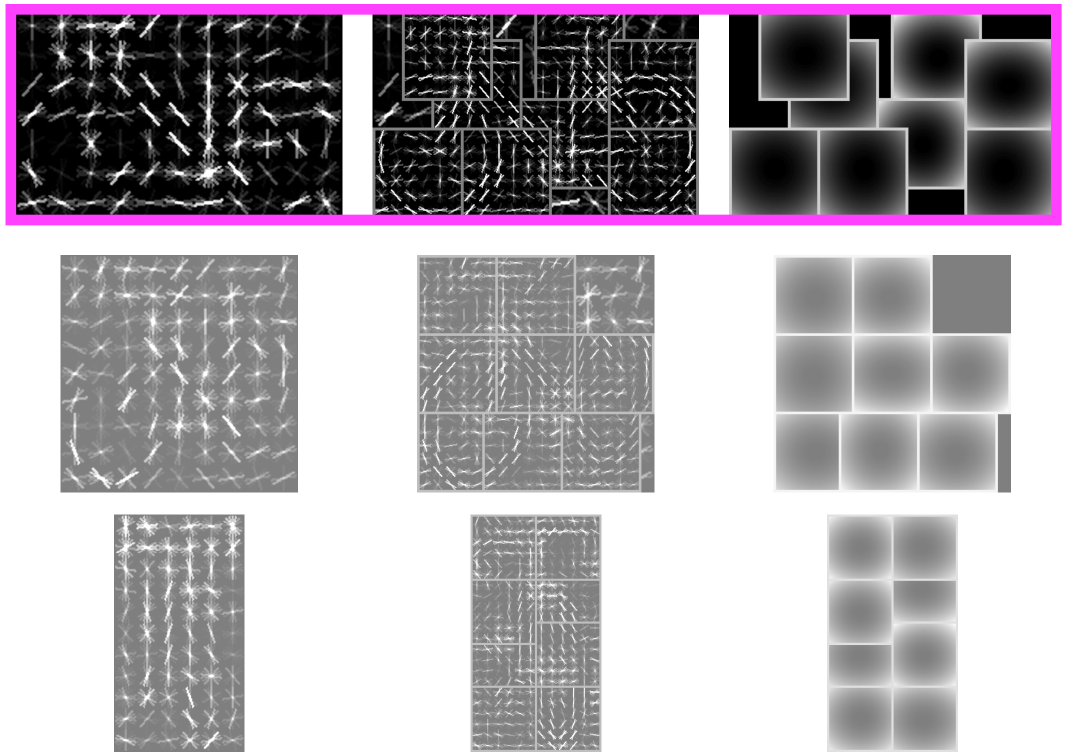
object model
+ part ordering
+ thresholds

slide credit: Girshick et al

Star cascade algorithm



HOG pyramid
from test image



object model
+ part order
+ thresholds

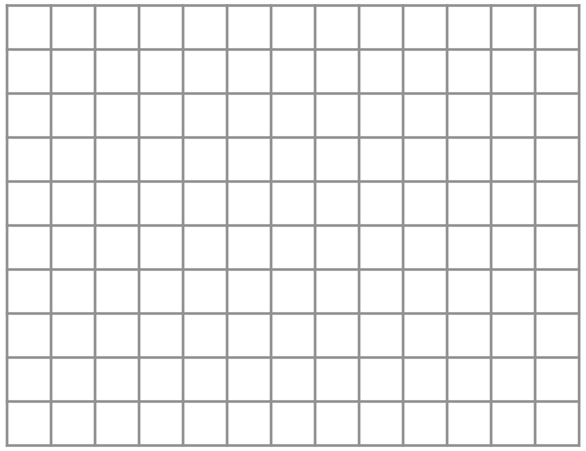
slide credit: Girshick et al

Star cascade algorithm

filter score tables

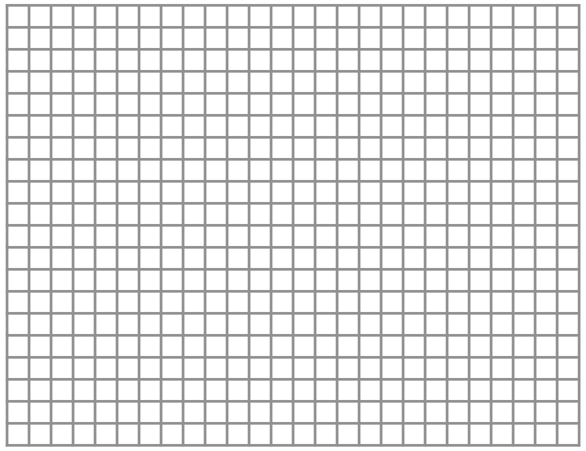
Root

$m_0(\omega)$



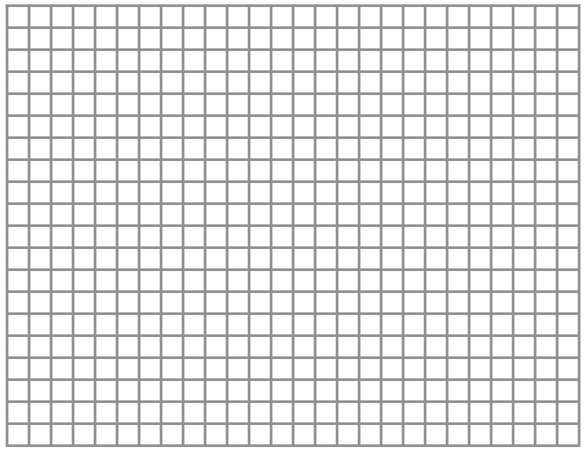
Part 1

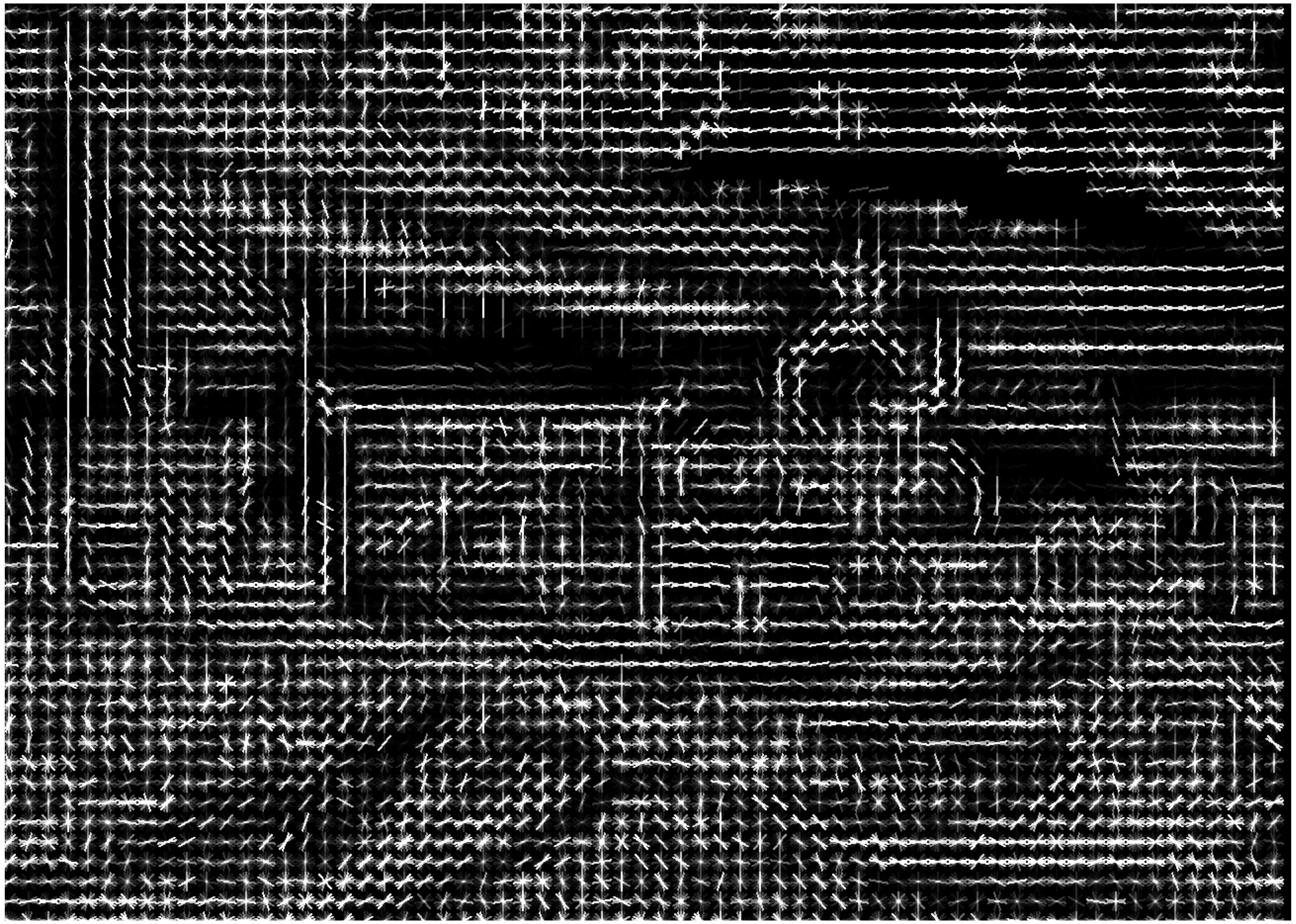
$m_1(\omega)$



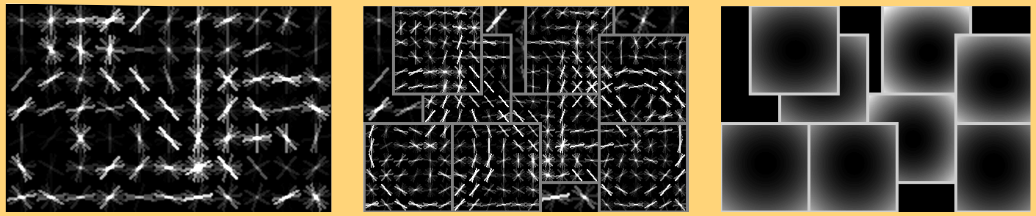
Part 2

$m_2(\omega)$





cascade test:

model: 

operation:

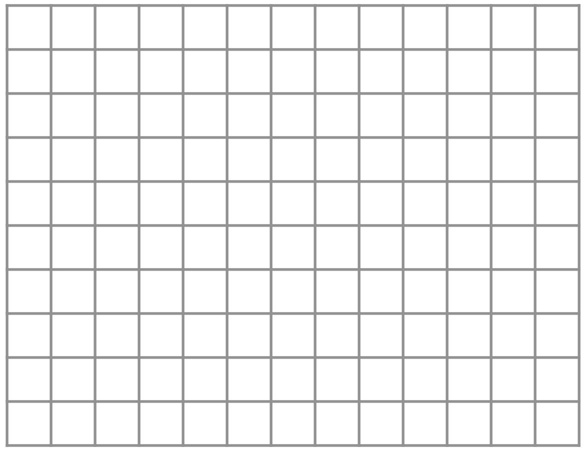
slide credit: Girshick et al

Star cascade algorithm

filter score tables

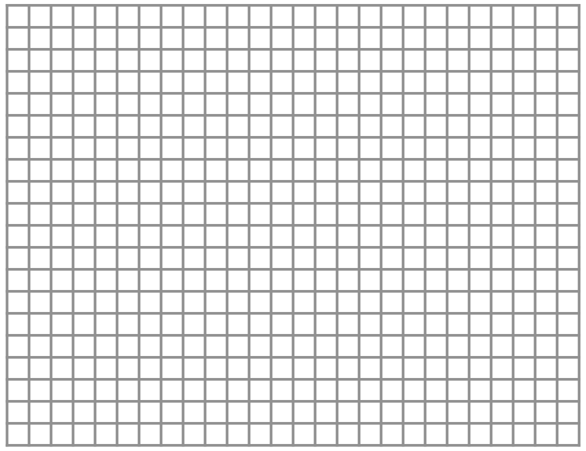
Root

$m_0(\omega)$



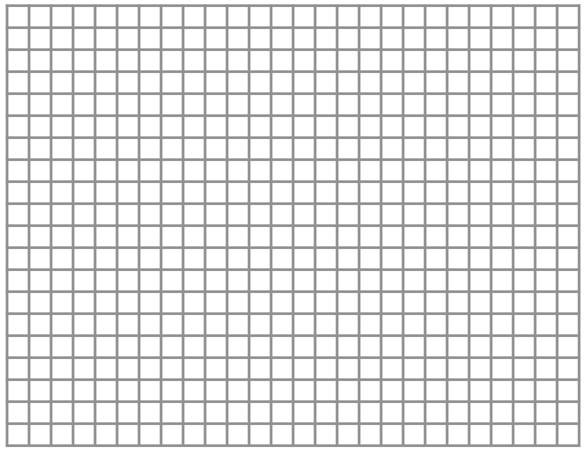
Part 1

$m_1(\omega)$



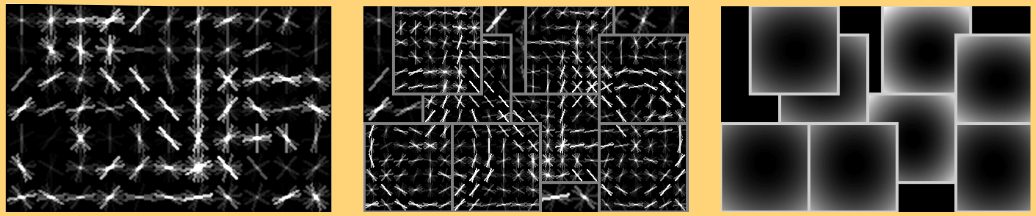
Part 2

$m_2(\omega)$



cascade test:

model:



operation:

slide credit: Girshick et al

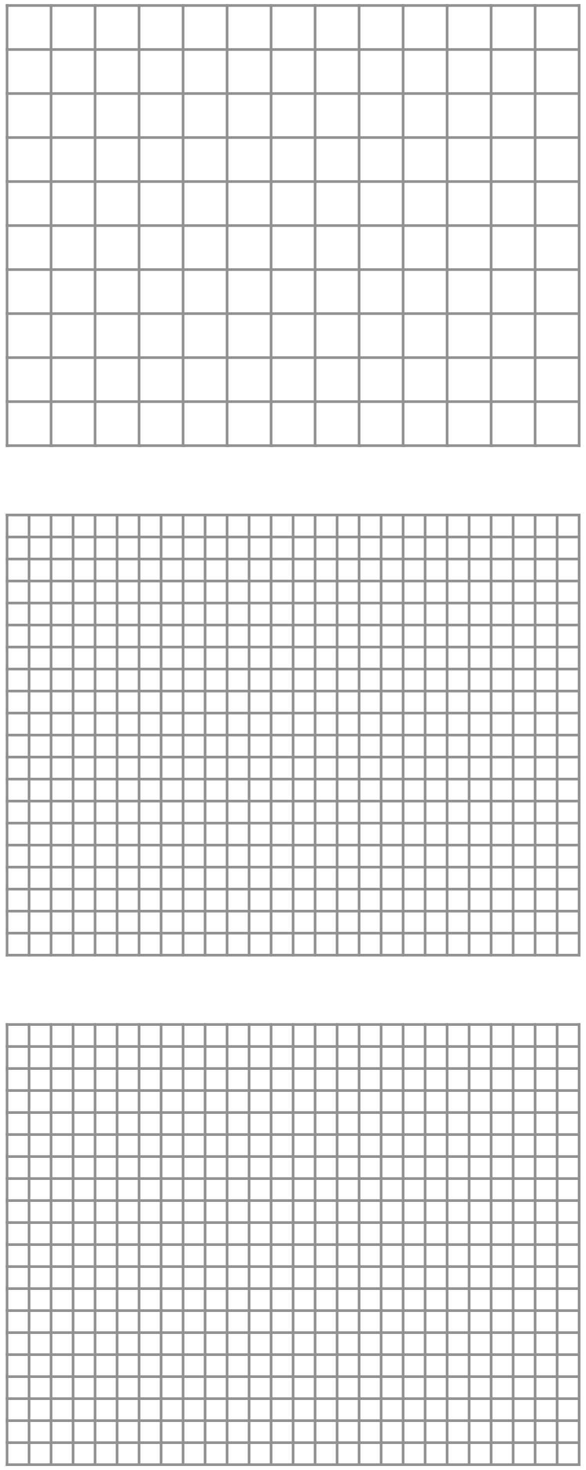
Star cascade algorithm

filter score tables

Root
$m_0(\omega)$

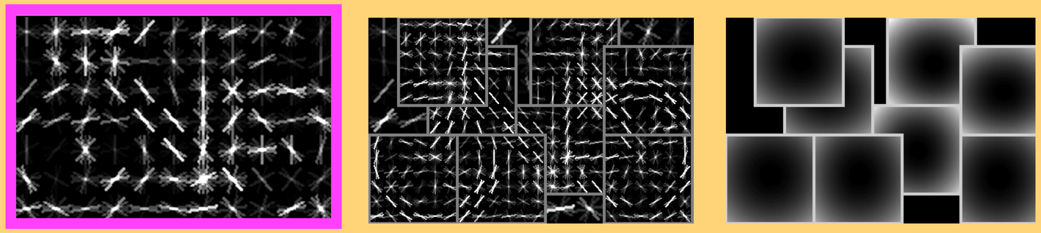
Part 1
$m_1(\omega)$

Part 2
$m_2(\omega)$



cascade test:

model:



operation: test root locations

slide credit: Girshick et al

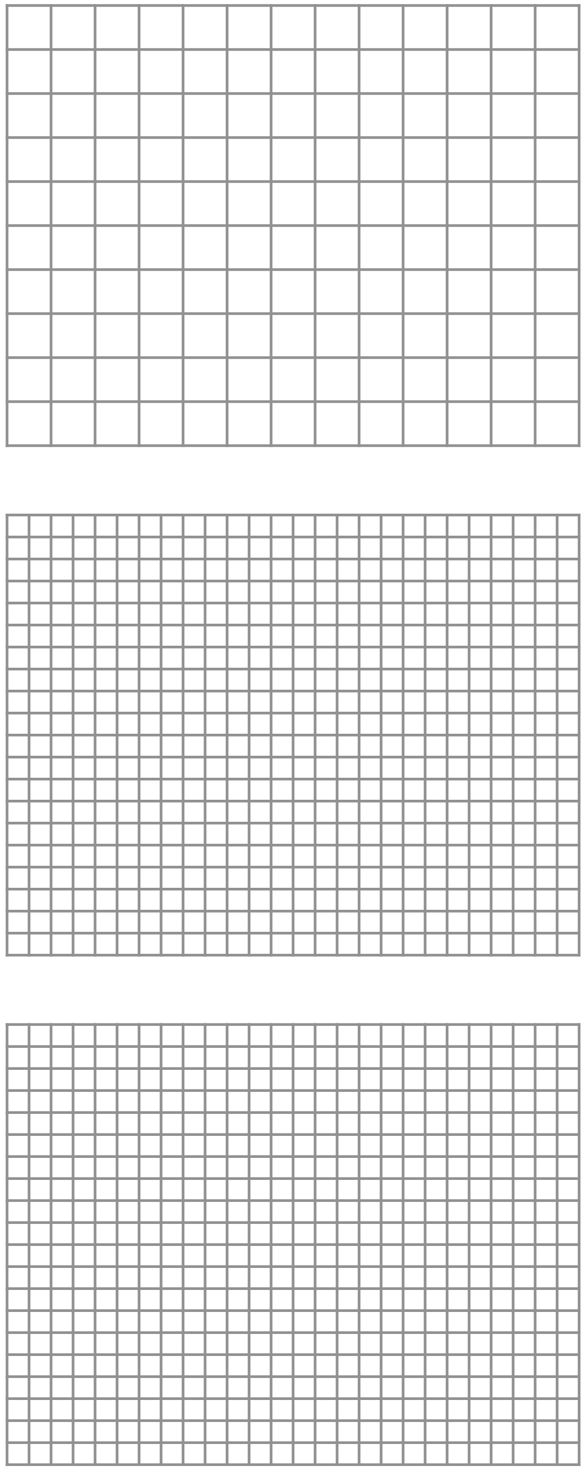
Star cascade algorithm

filter score tables

Root
$m_0(\omega)$

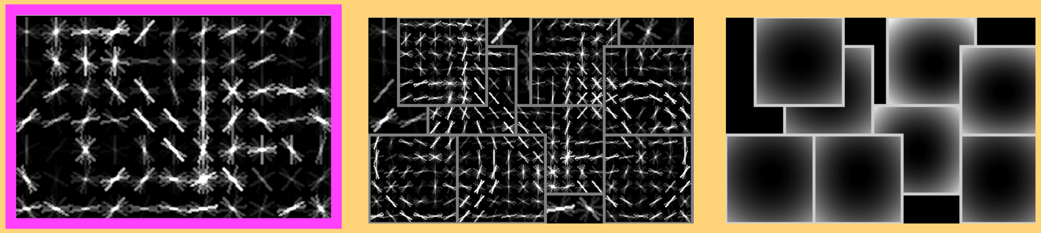
Part 1
$m_1(\omega)$

Part 2
$m_2(\omega)$



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

model:



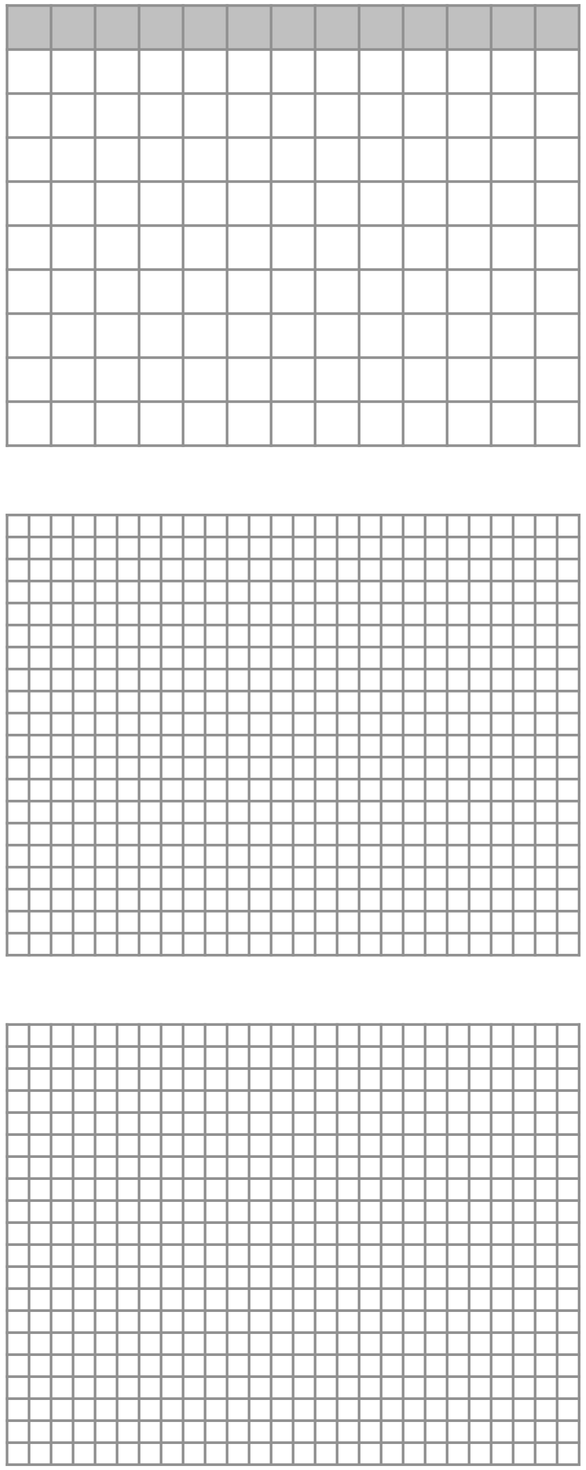
operation: test root locations

result: fail

slide credit: Girshick et al

Star cascade algorithm

filter score tables



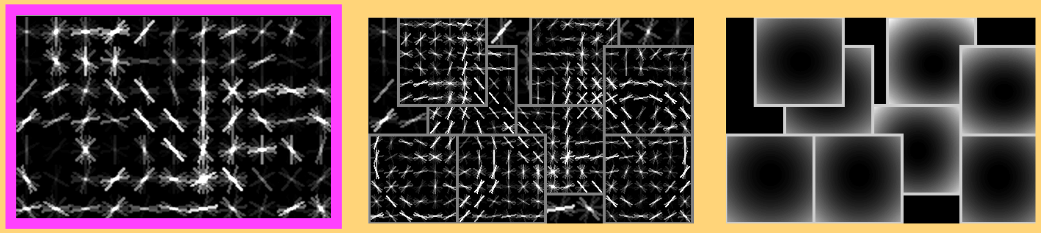
Root
$m_0(\omega)$

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

slide credit: Girshick et al

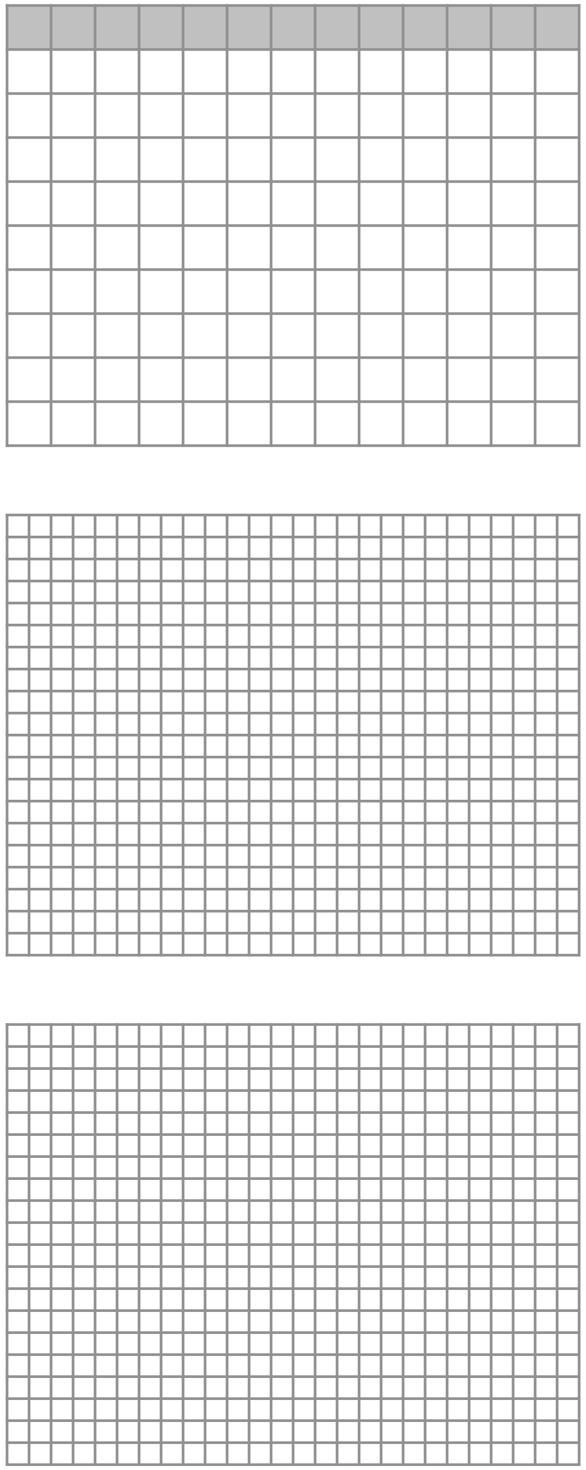
Star cascade algorithm

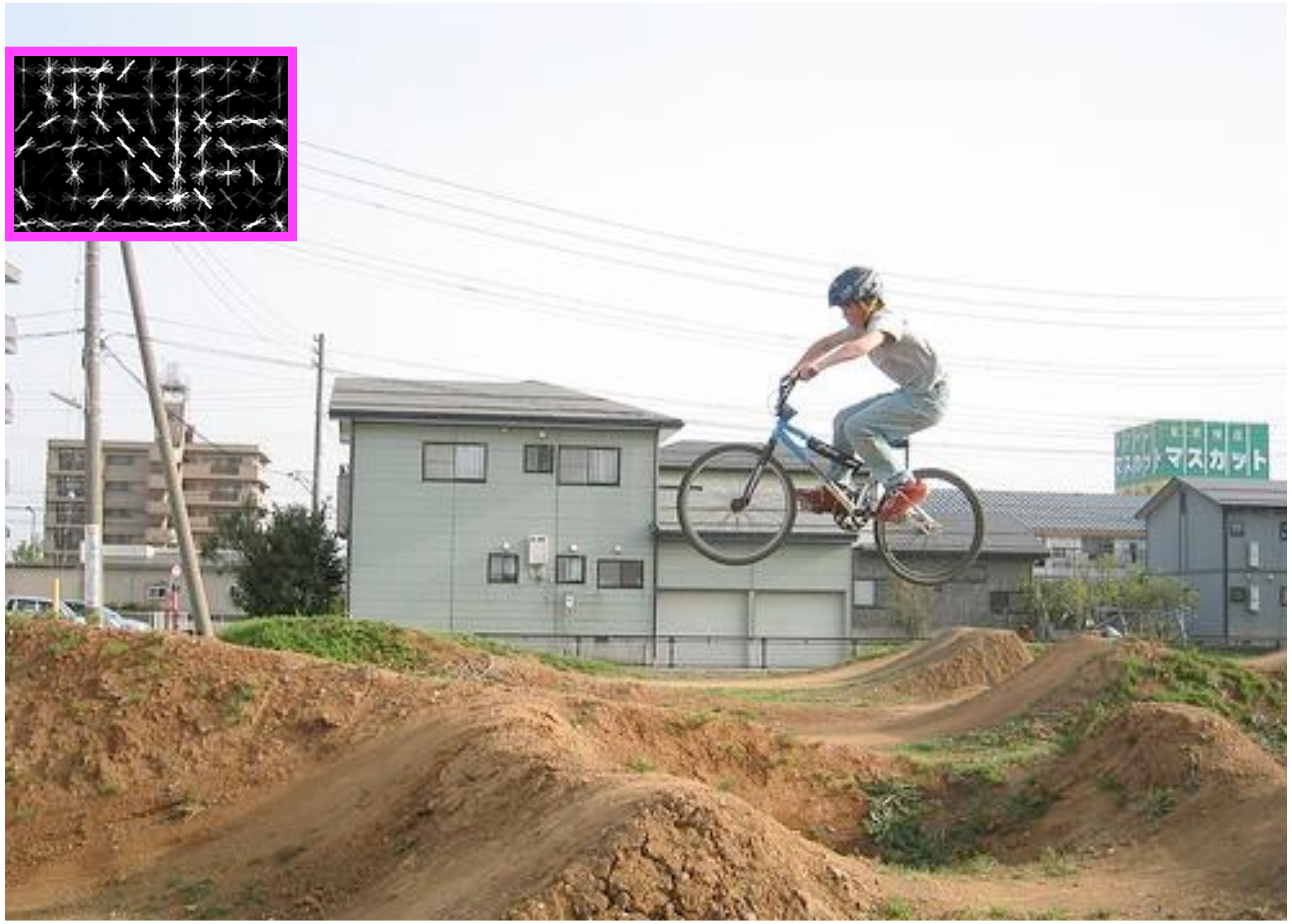
filter score tables

Root
$m_0(\omega)$

Part 1
$m_1(\omega)$

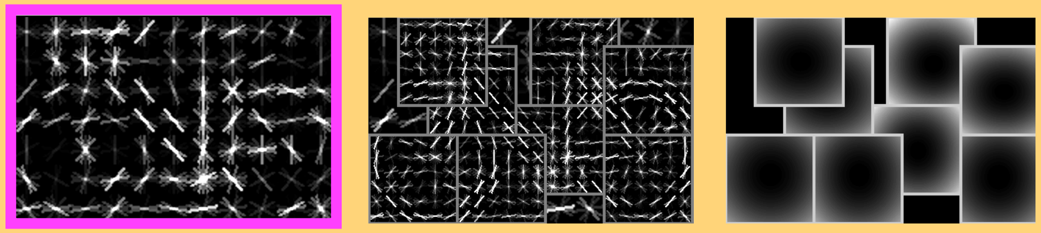
Part 2
$m_2(\omega)$





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

model:



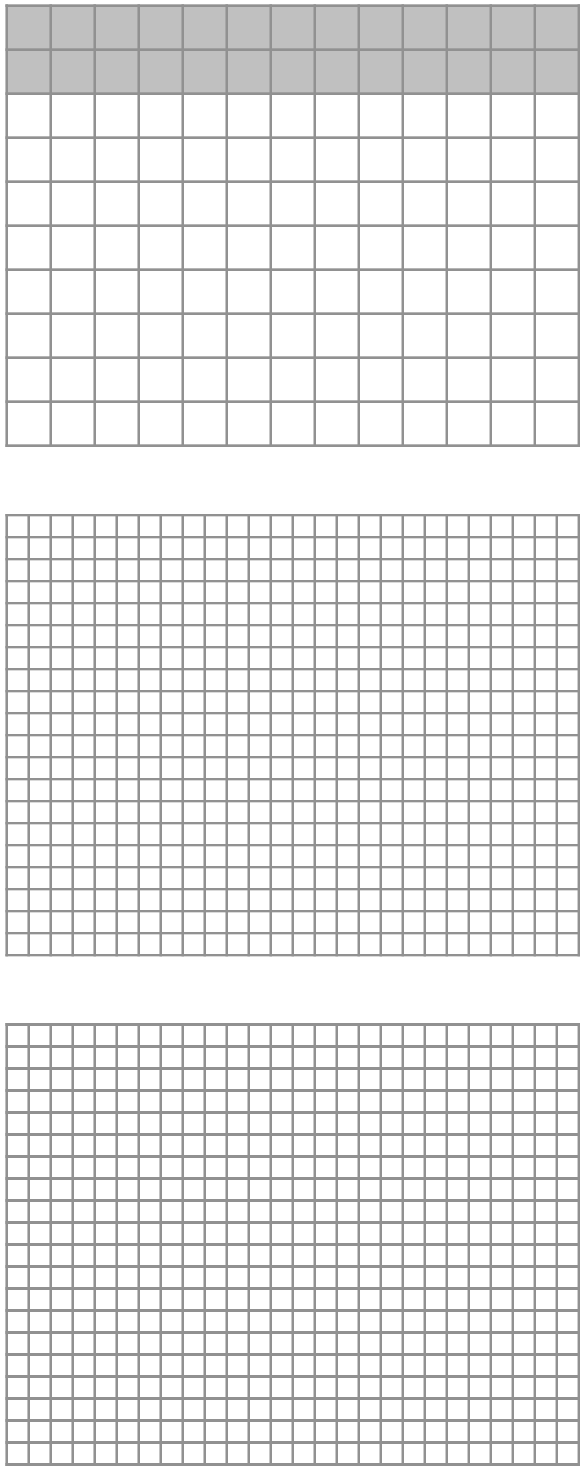
operation: test root locations

result: fail

slide credit: Girshick et al

Star cascade algorithm

filter score tables



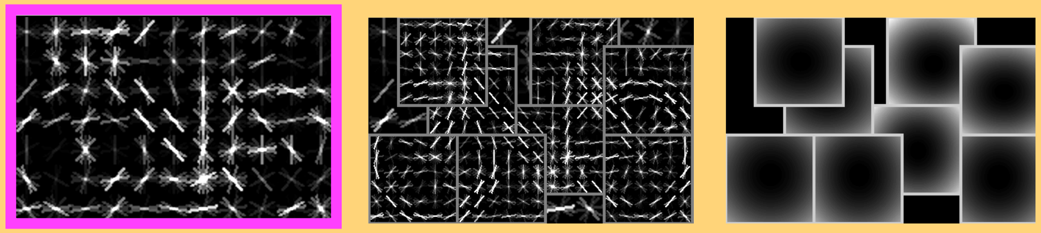
Root
$m_0(\omega)$

Part 1
$m_1(\omega)$

Part 2
$m_2(\omega)$

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

model:



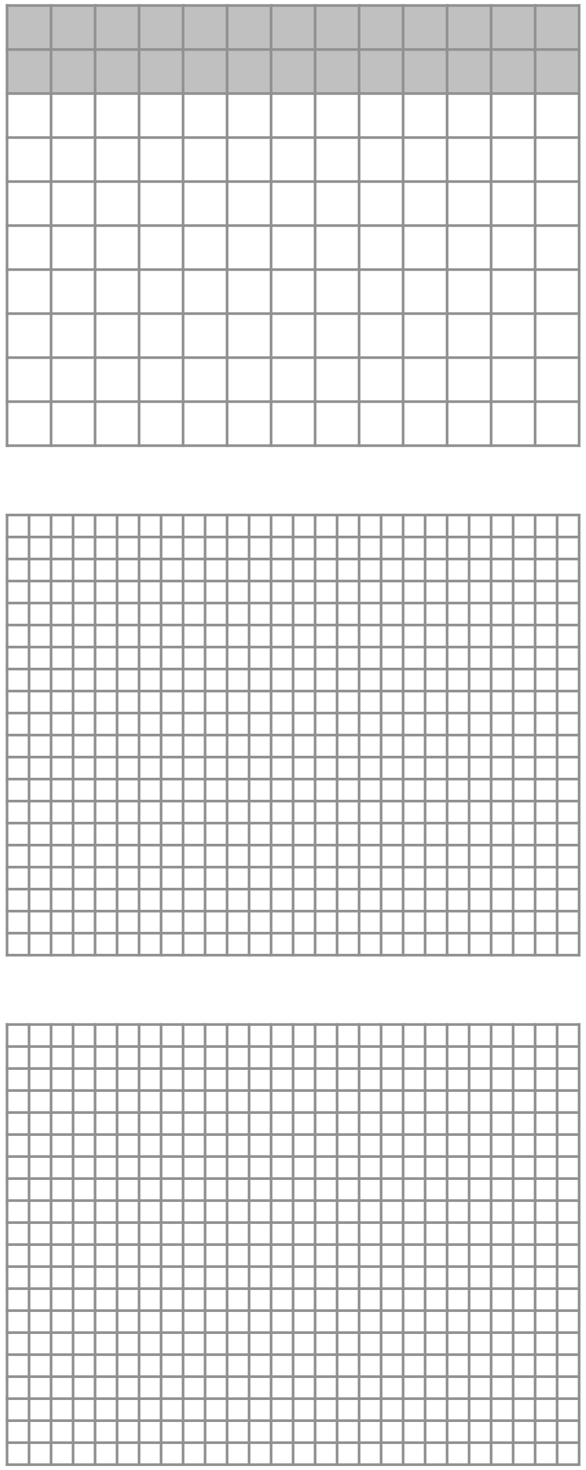
operation: test root locations

result: fail

slide credit: Girshick et al

Star cascade algorithm

filter score tables



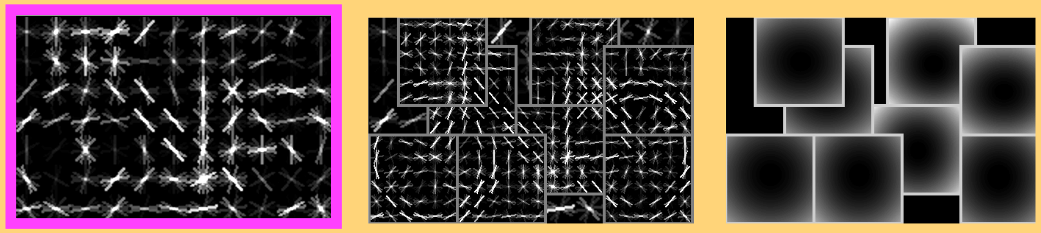
Root
$m_0(\omega)$

Part 1
$m_1(\omega)$

Part 2
$m_2(\omega)$

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

model:

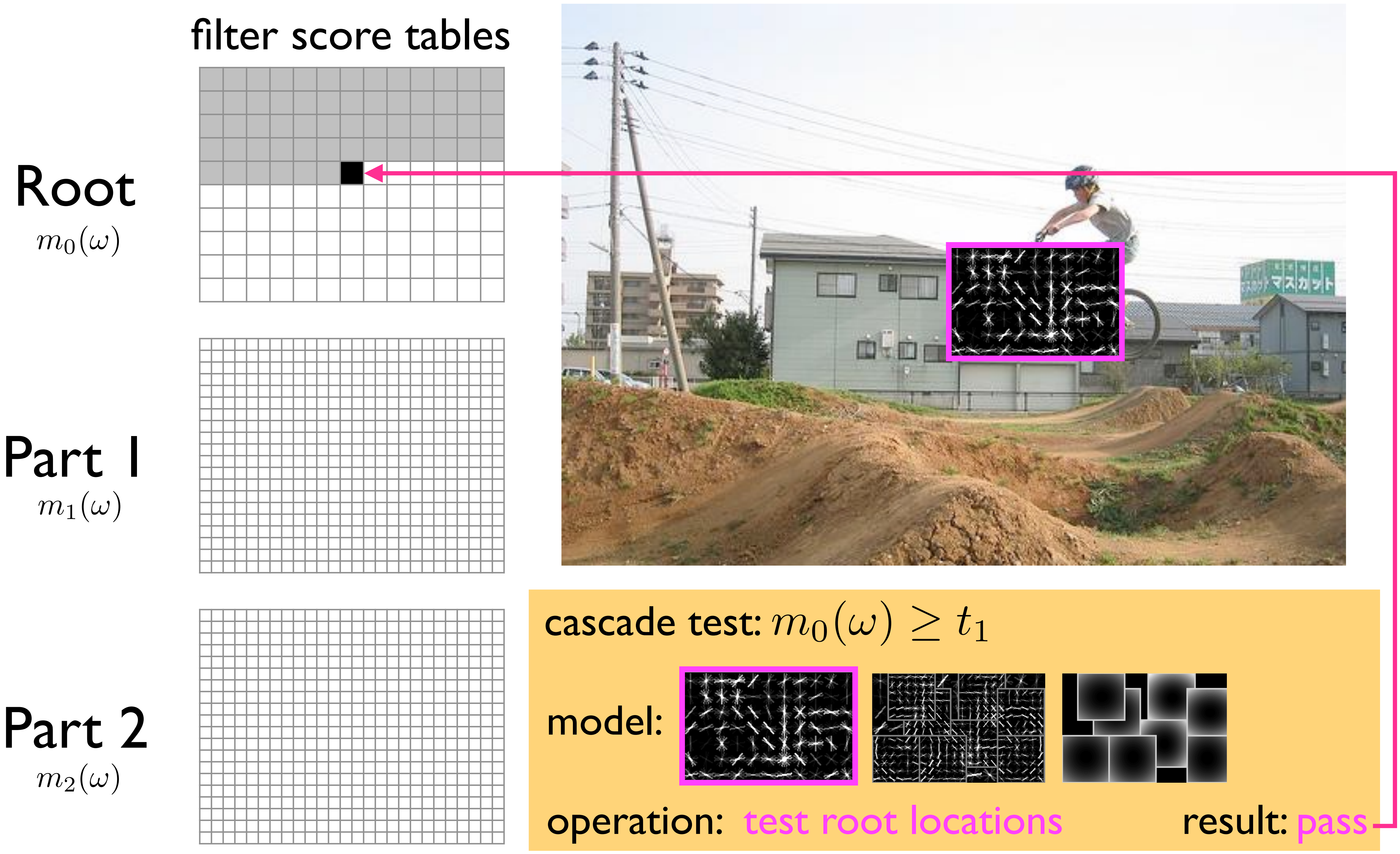


operation: test root locations

result: fail

slide credit: Girshick et al

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

slide credit: Girshick et al

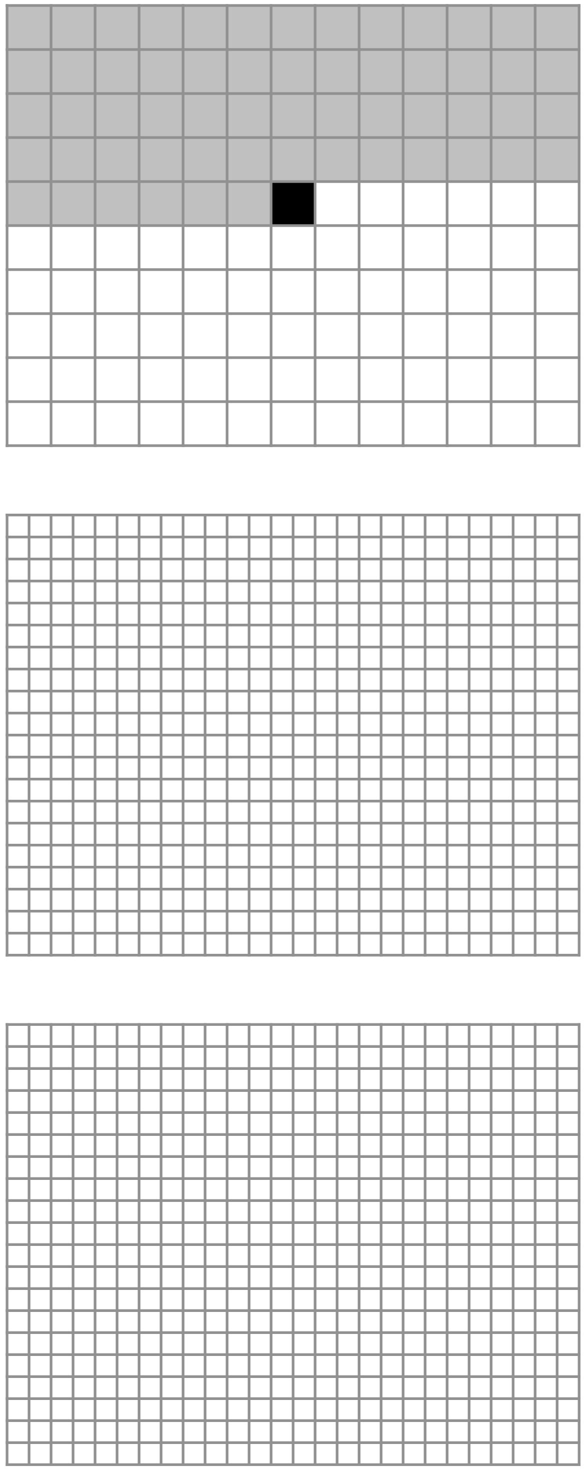
Star cascade algorithm

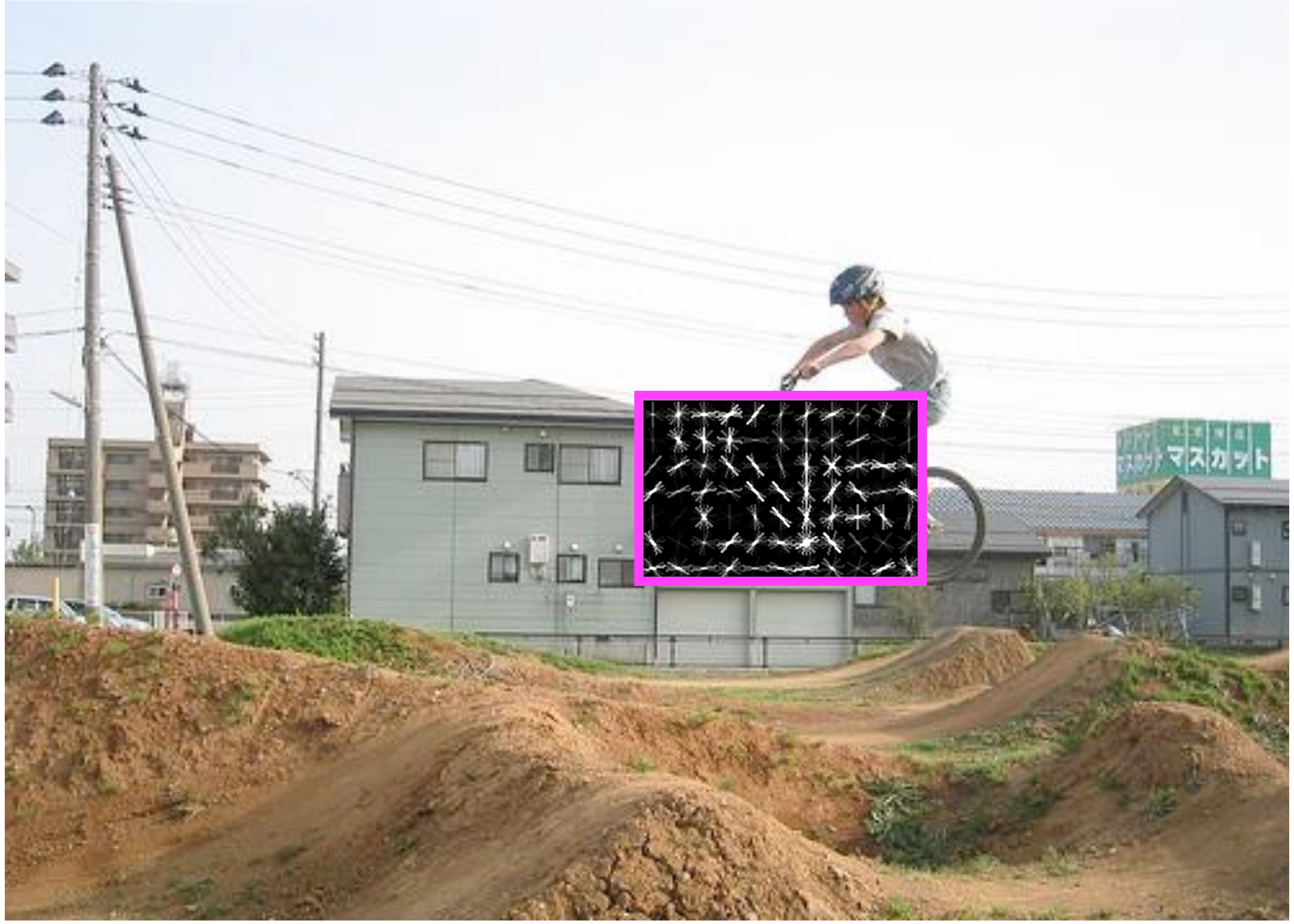
filter score tables

Root
$m_0(\omega)$

Part 1
$m_1(\omega)$

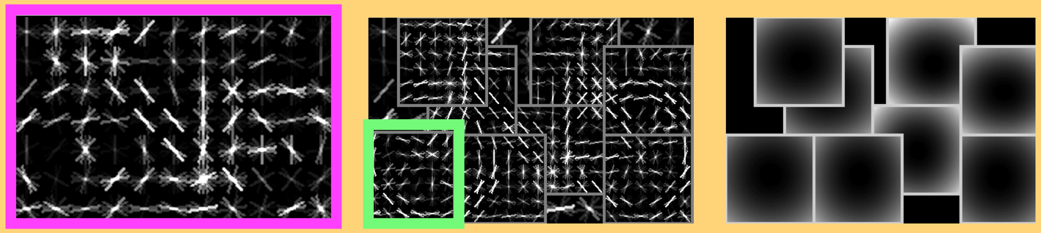
Part 2
$m_2(\omega)$





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

model:

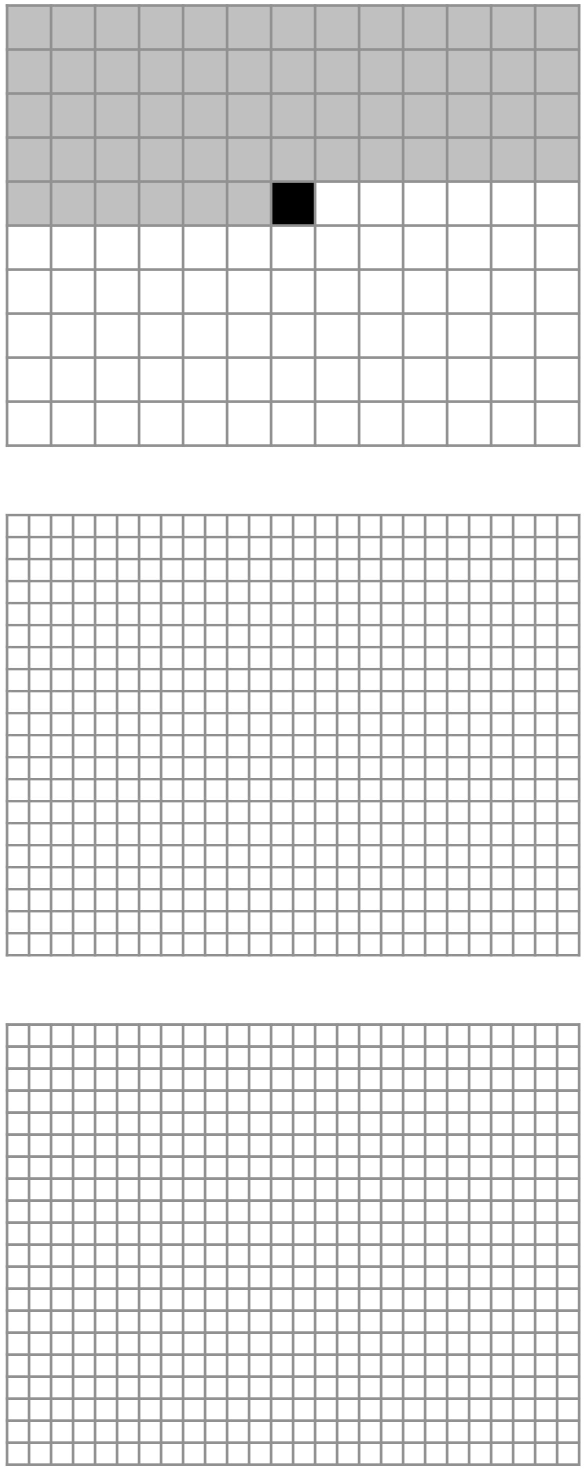


operation: displacement search

slide credit: Girshick et al

Star cascade algorithm

filter score tables

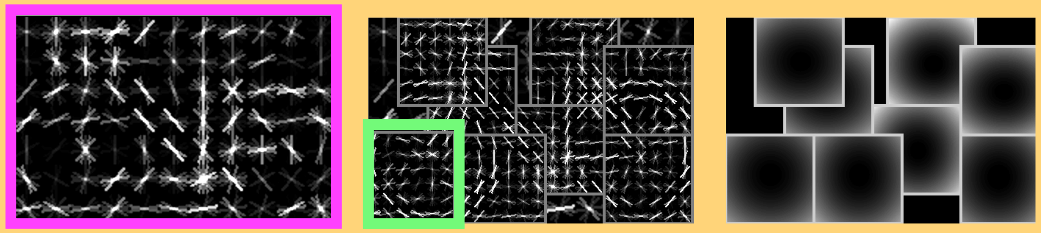


Root
$m_0(\omega)$

Part 1
$m_1(\omega)$

Part 2
$m_2(\omega)$

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

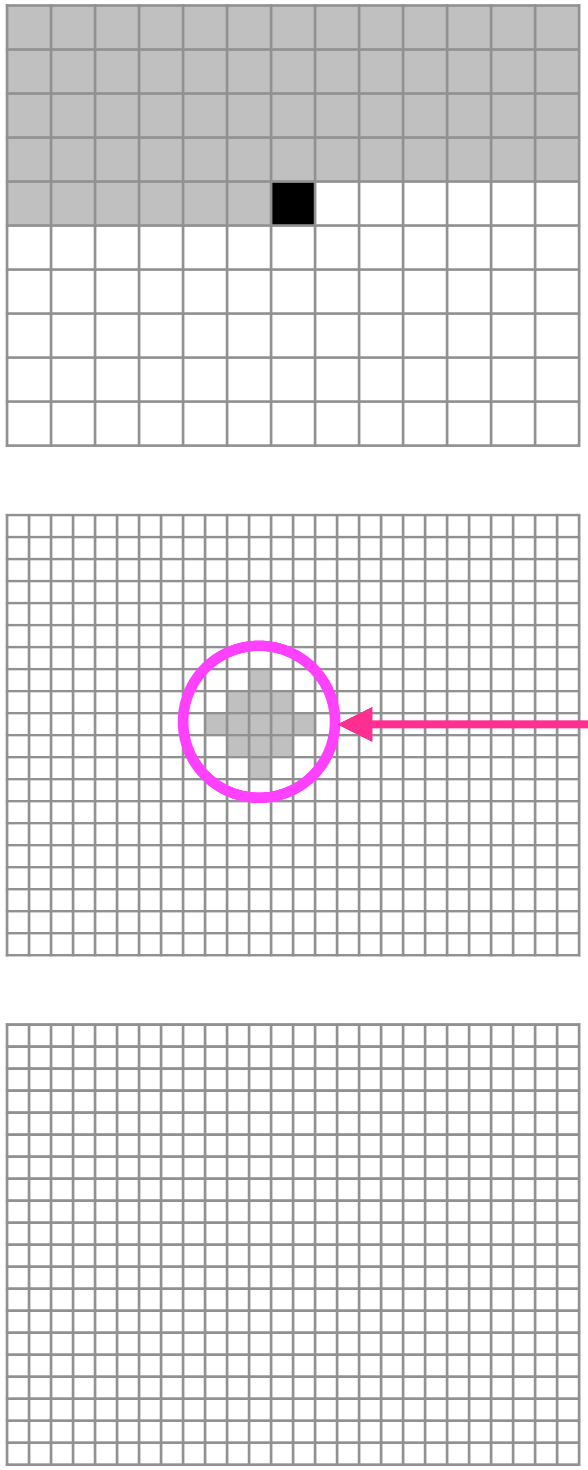
model: 

operation: displacement search

slide credit: Girshick et al

Star cascade algorithm

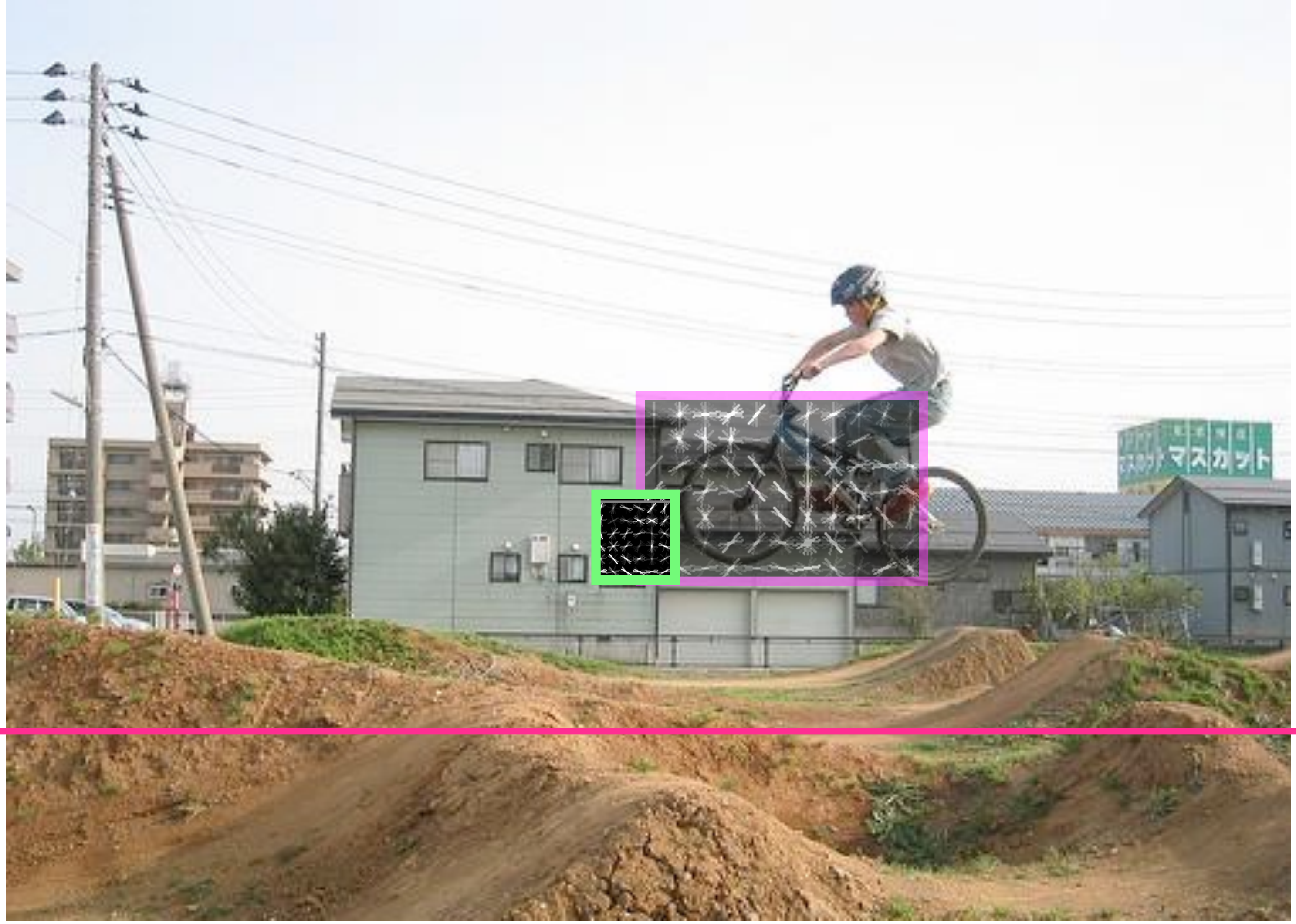
filter score tables



Root
$m_0(\omega)$

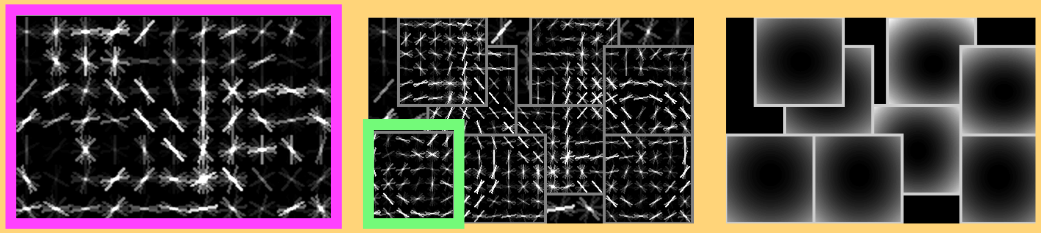
Part 1
$m_1(\omega)$

Part 2
$m_2(\omega)$



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

model:

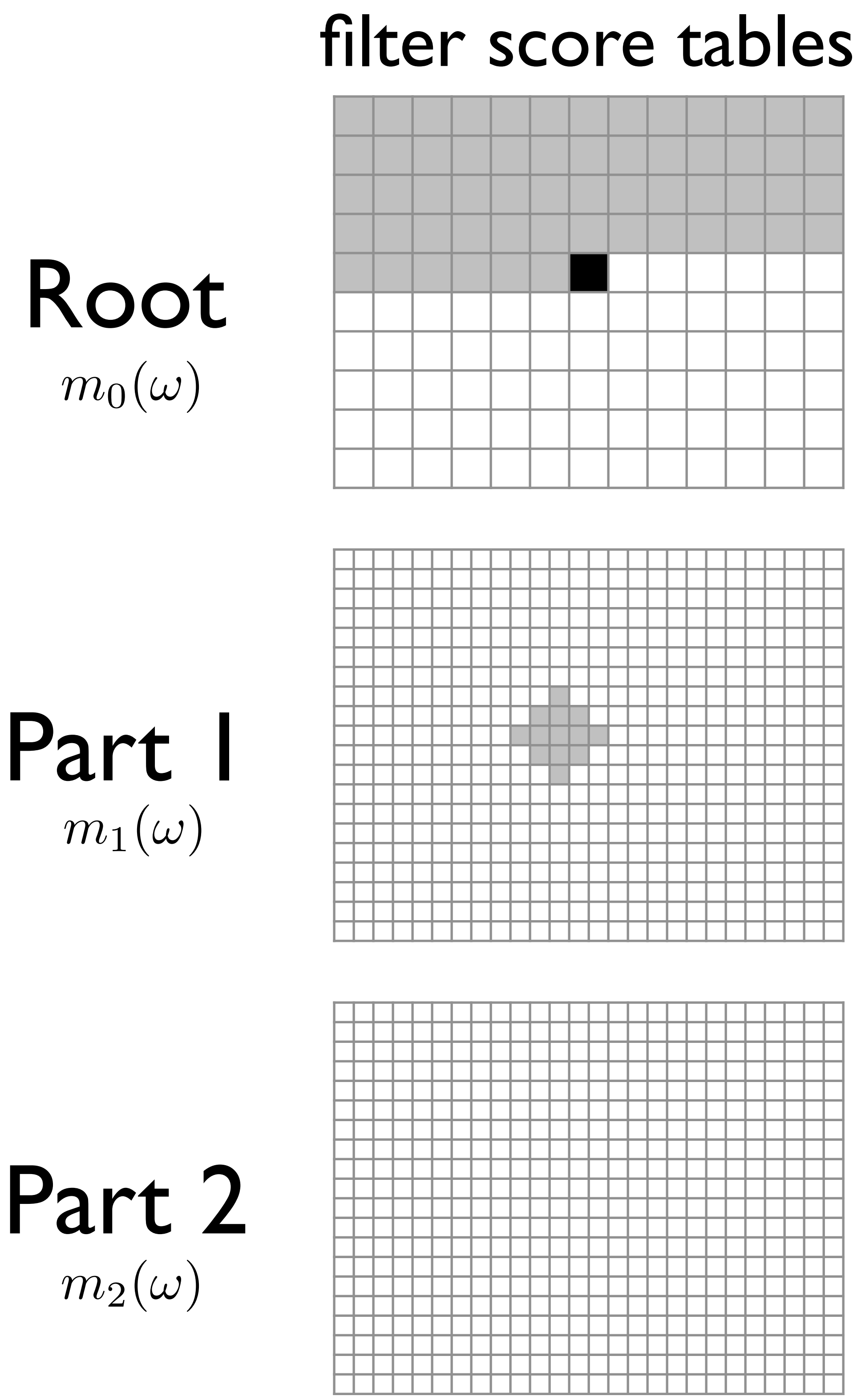


operation: displacement search

result: pass

slide credit: Girshick et al

Star cascade algorithm

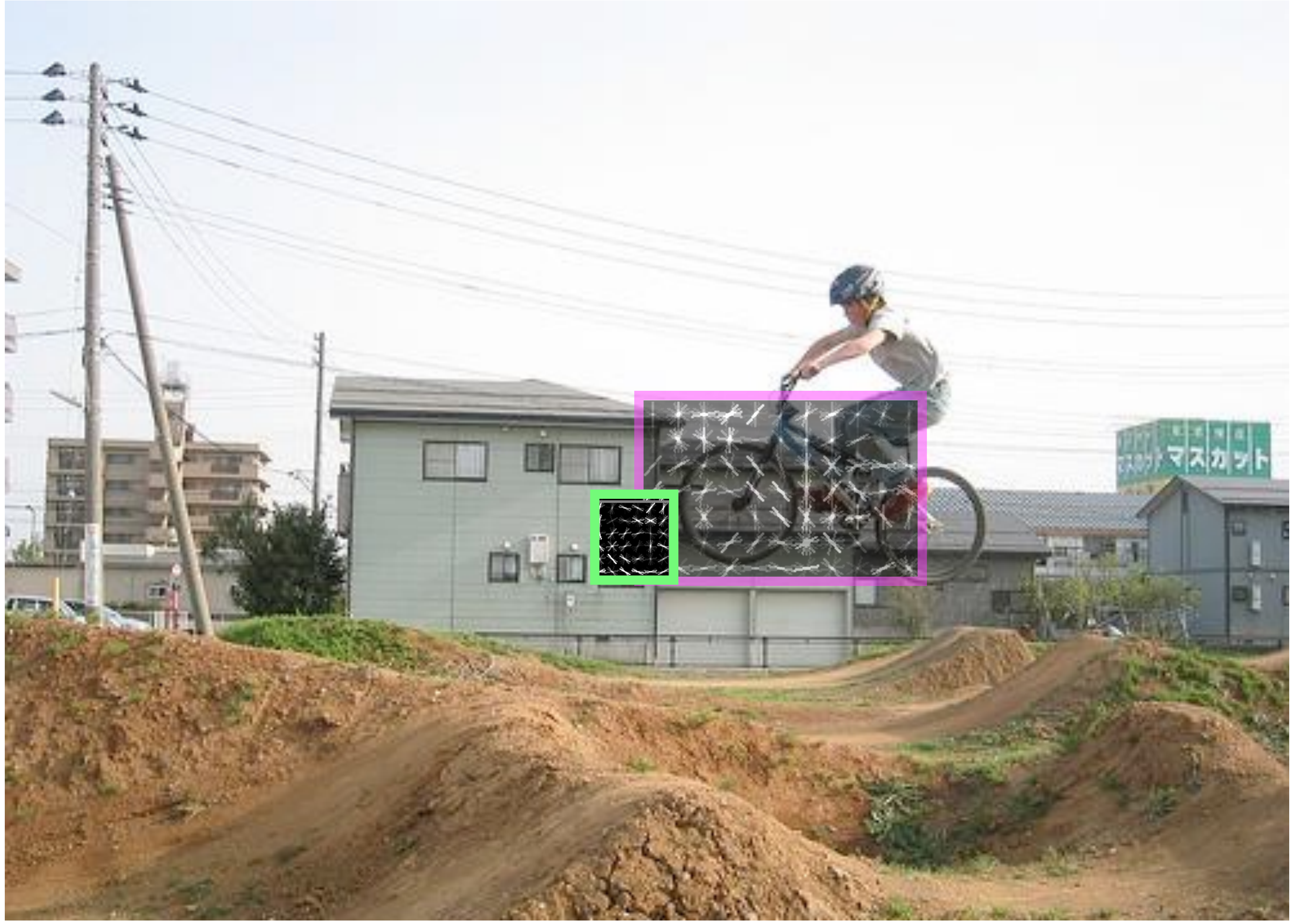


filter score tables

Root $m_0(\omega)$

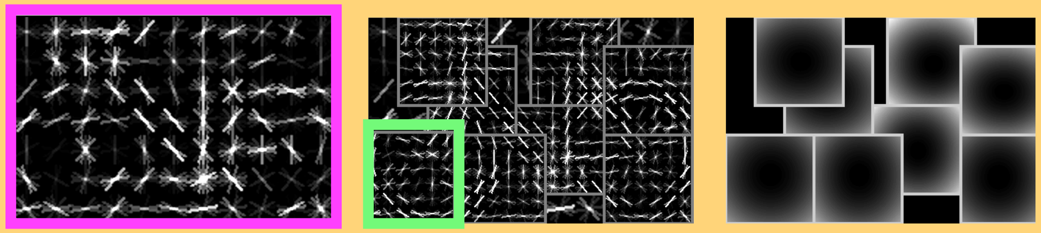
Part 1 $m_1(\omega)$

Part 2 $m_2(\omega)$



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

model:

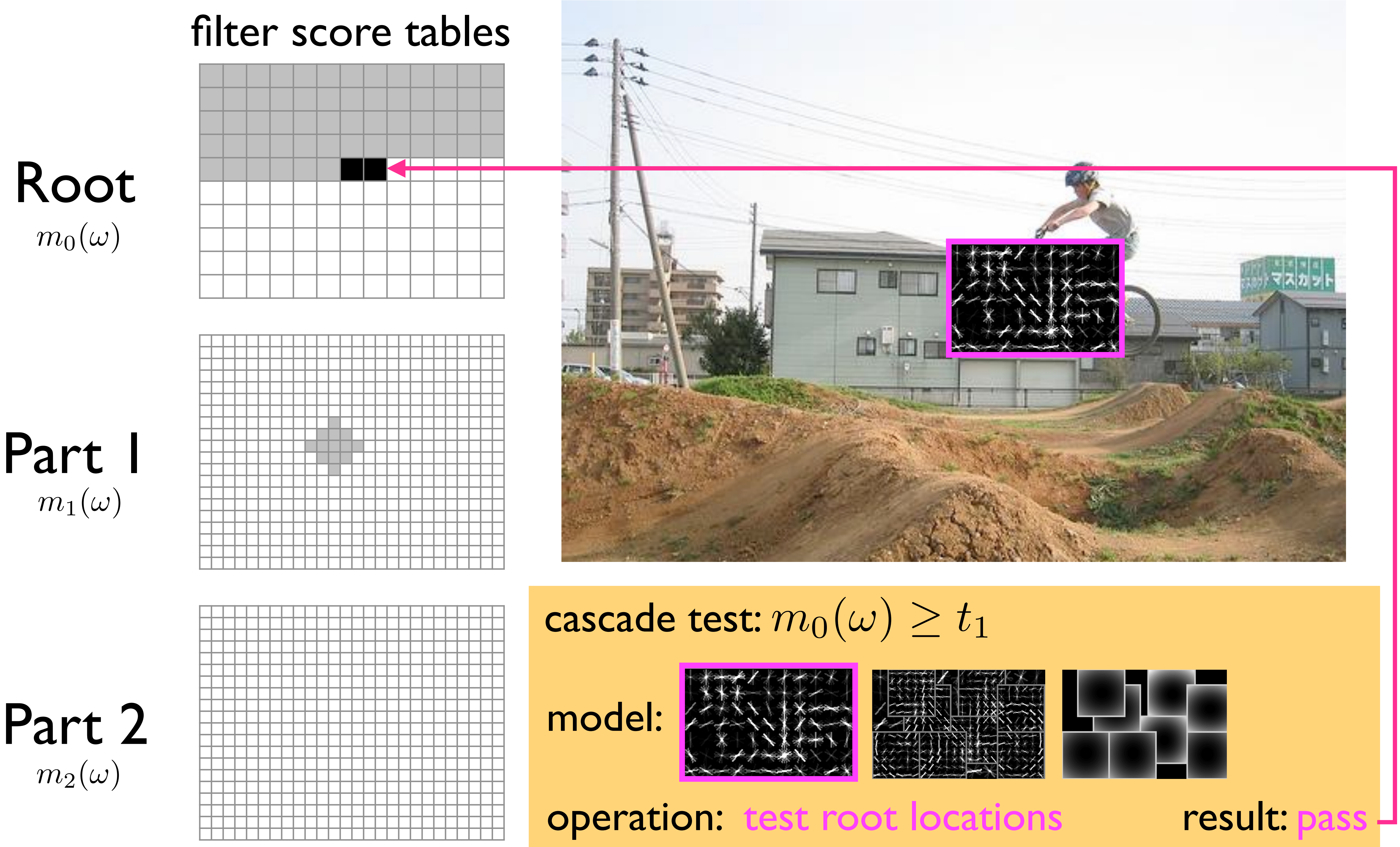


operation: test partial score

result: fail

slide credit: Girshick et al

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:

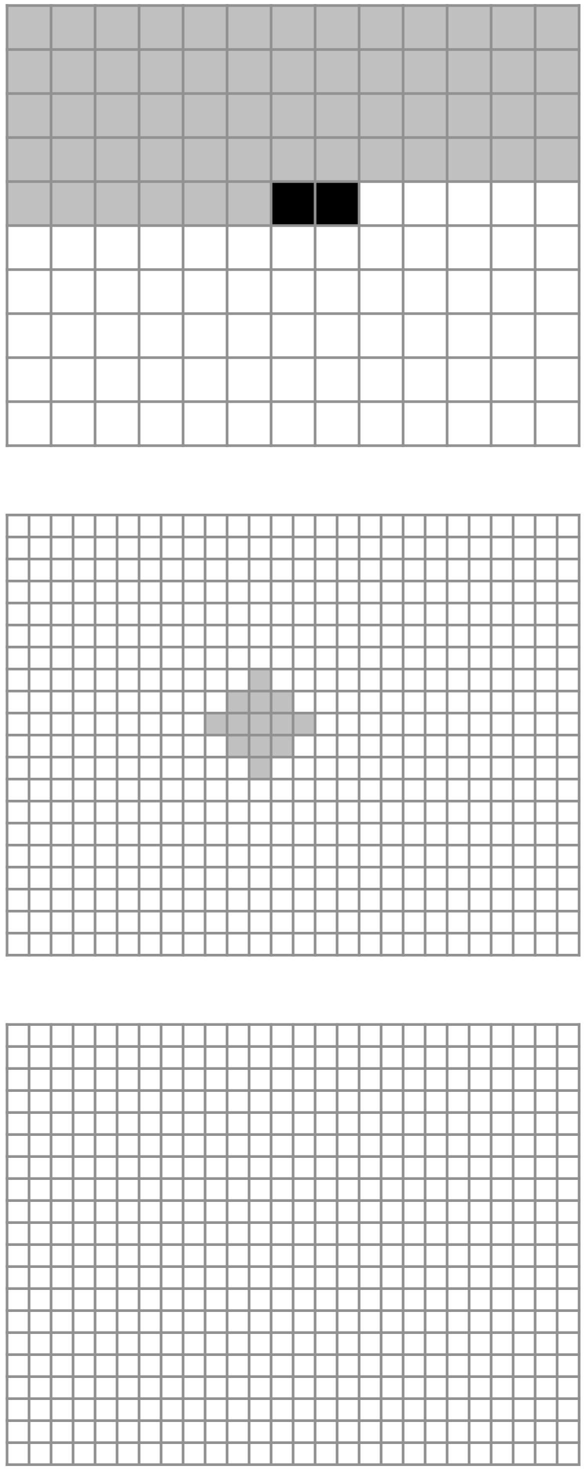
operation: test root locations

result: pass

slide credit: Girshick et al

Star cascade algorithm

filter score tables



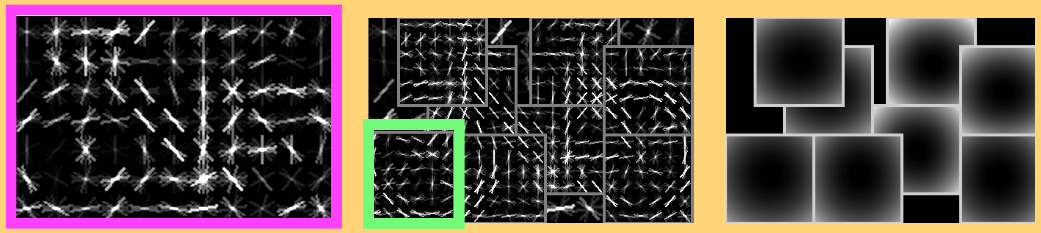
Root $m_0(\omega)$

Part 1 $m_1(\omega)$

Part 2 $m_2(\omega)$

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

model:

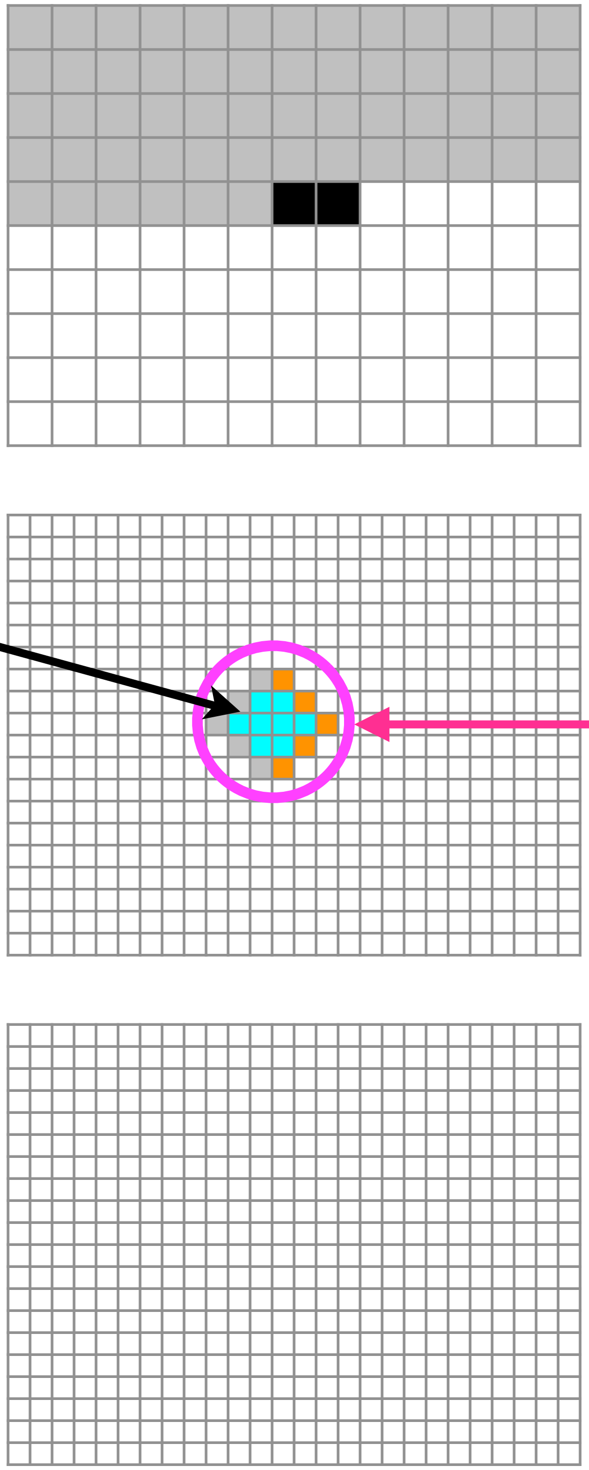


operation: displacement search

slide credit: Girshick et al

Star cascade algorithm

filter score tables



Root

$m_0(\omega)$

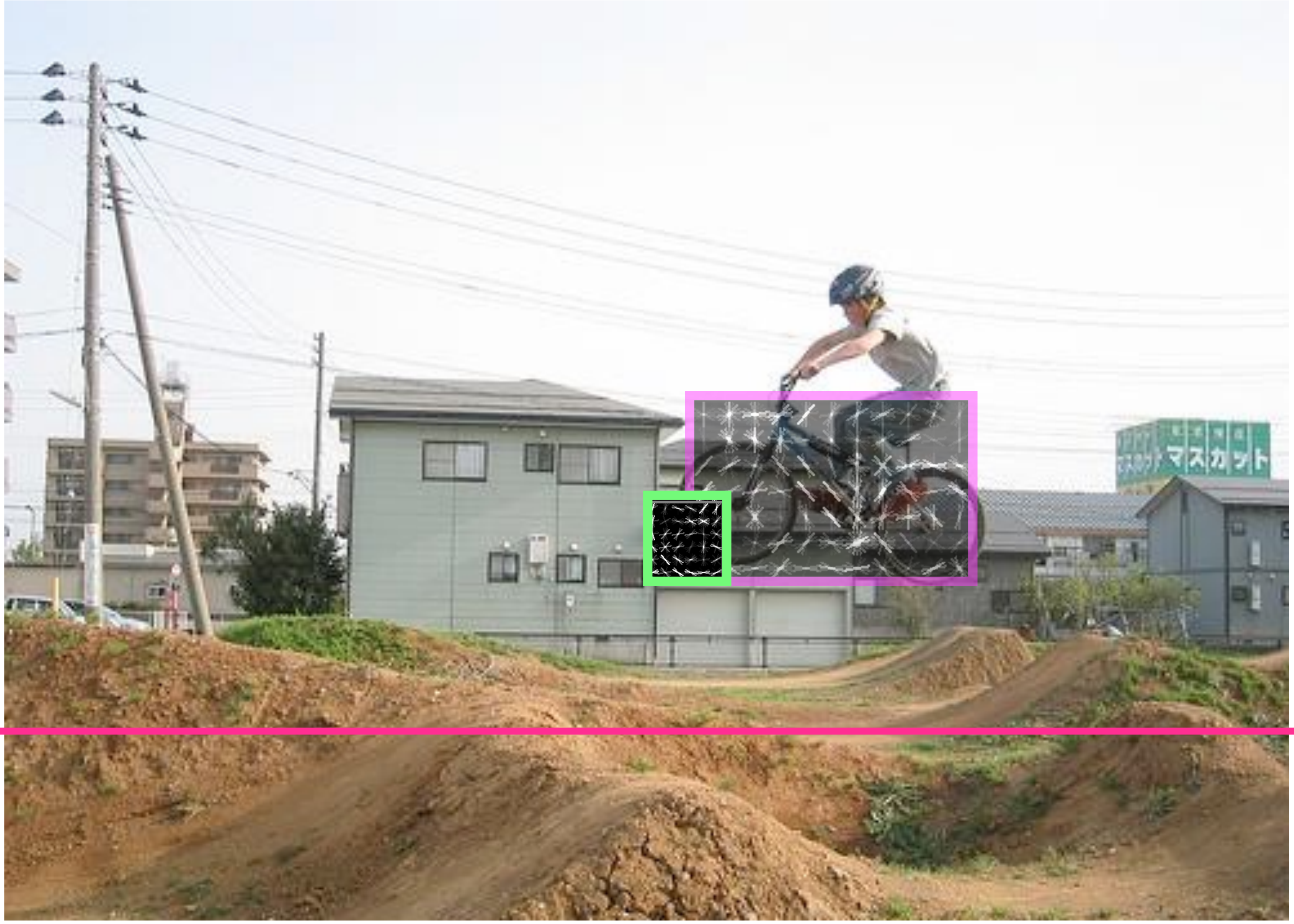
cached!

Part 1

$m_1(\omega)$

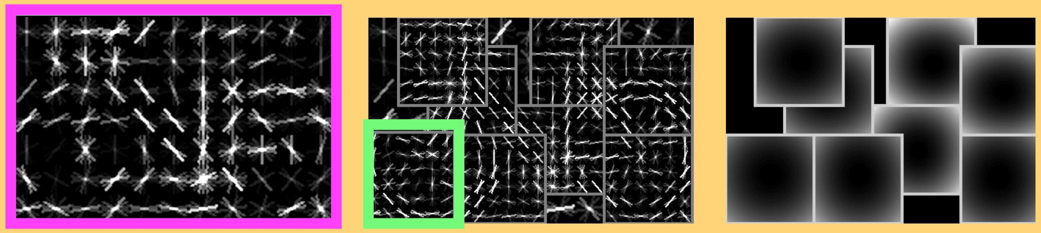
Part 2

$m_2(\omega)$



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

model:



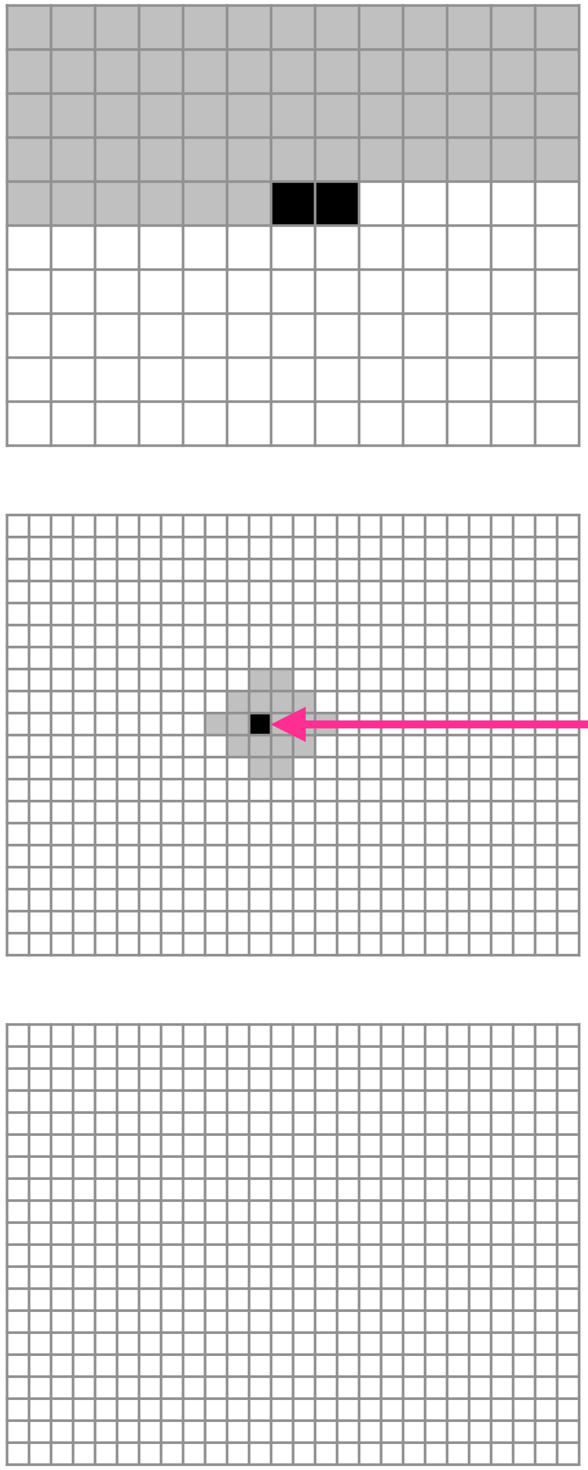
operation: displacement search

result: pass

slide credit: Girshick et al

Star cascade algorithm

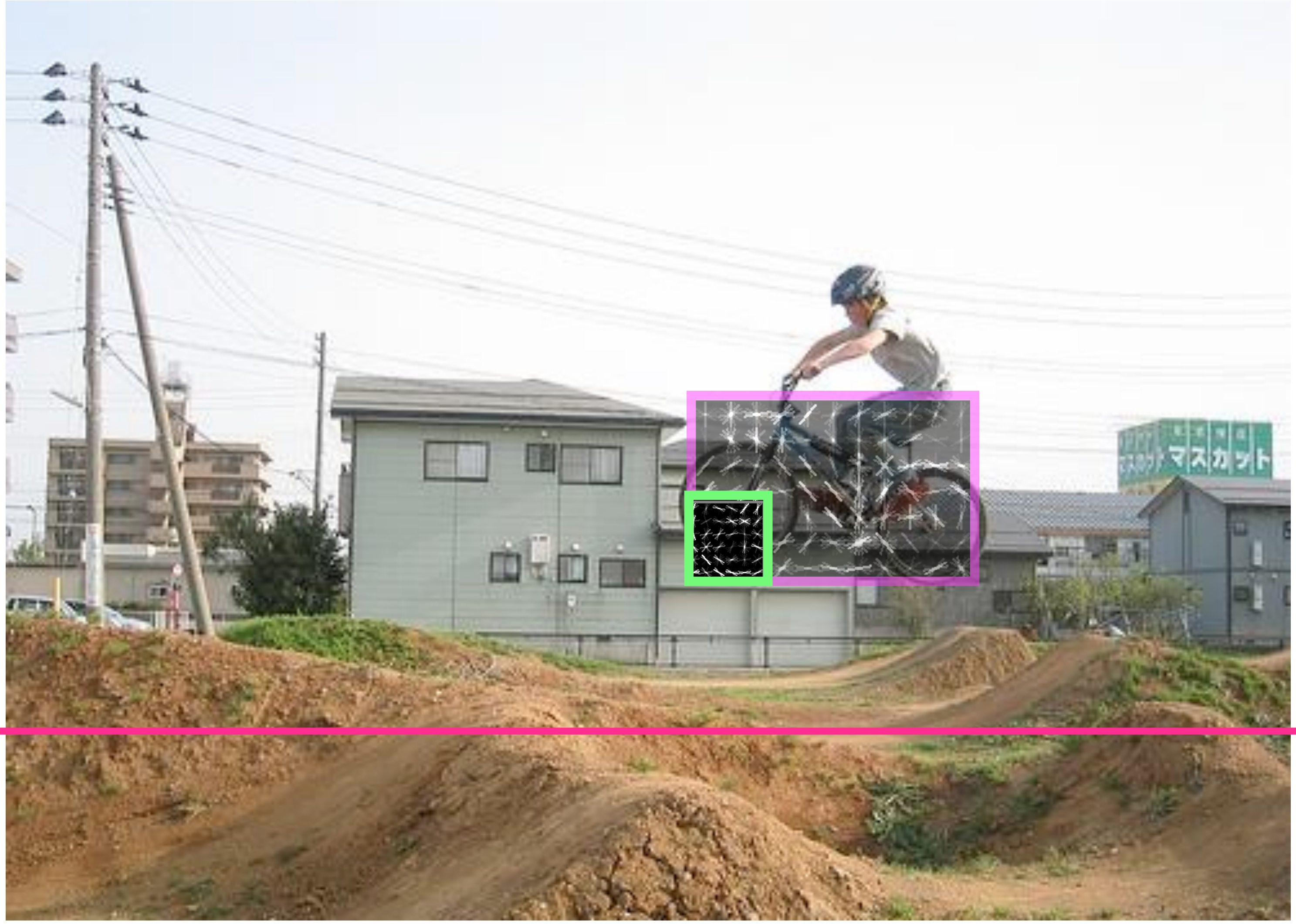
filter score tables



Root
$m_0(\omega)$

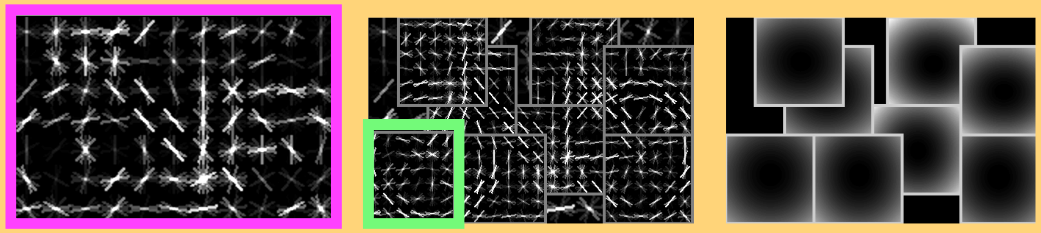
Part 1
$m_1(\omega)$

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

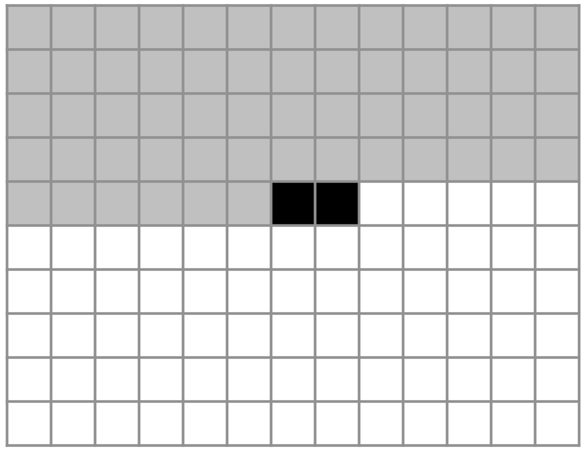
slide credit: Girshick et al

Star cascade algorithm

filter score tables

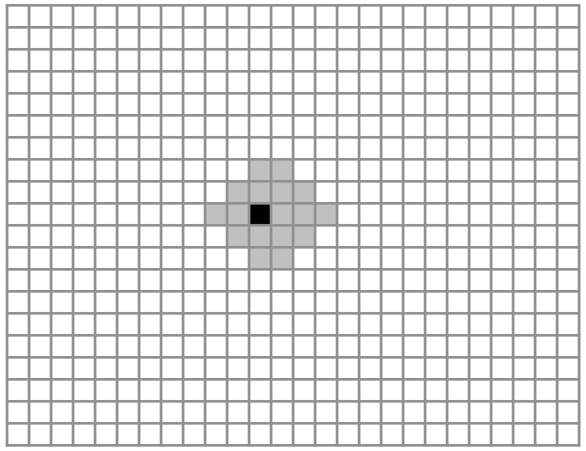
Root

$m_0(\omega)$



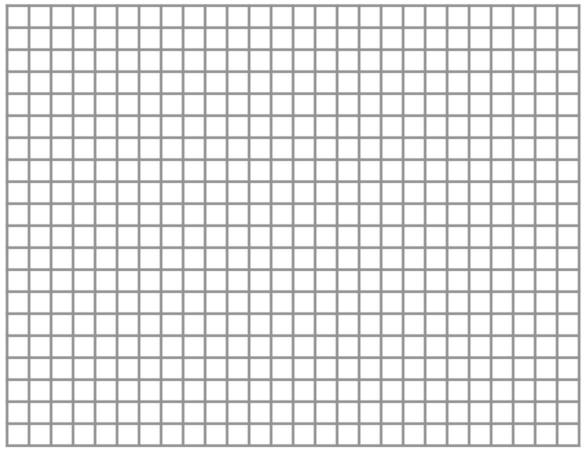
Part 1

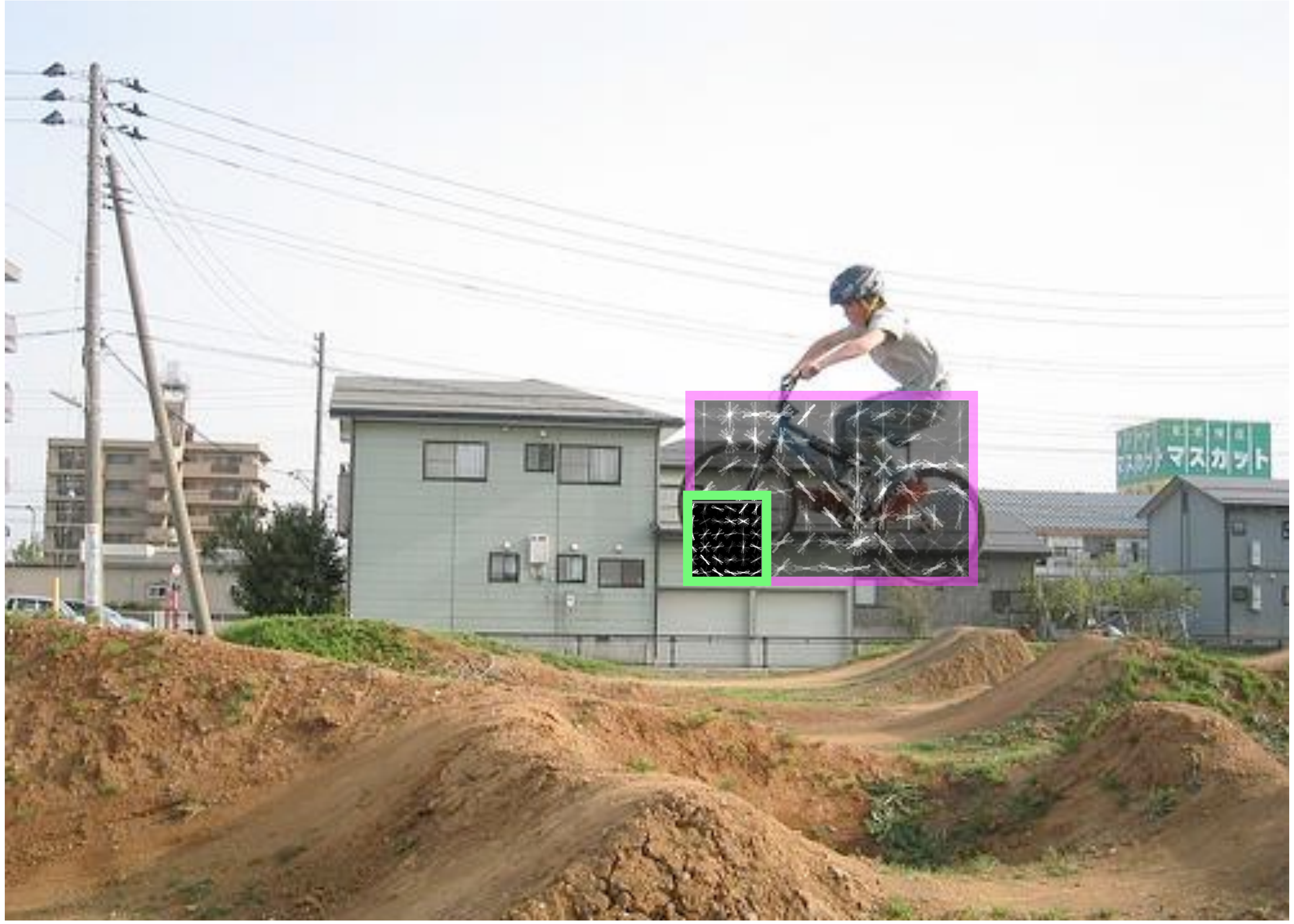
$m_1(\omega)$



Part 2

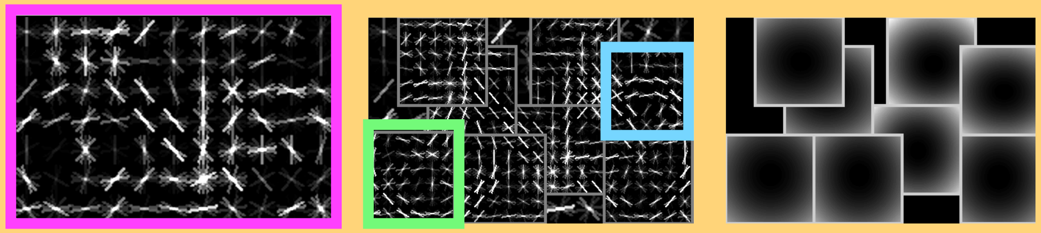
$m_2(\omega)$





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

model:

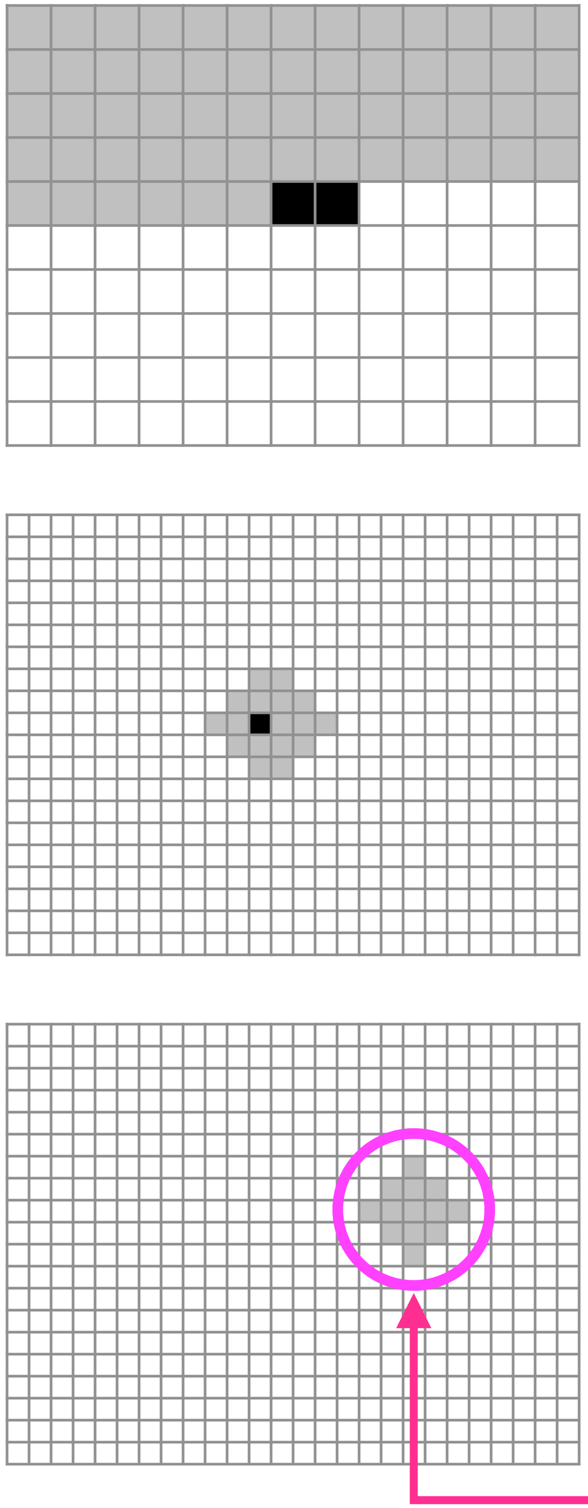


operation: displacement search

slide credit: Girshick et al

Star cascade algorithm

filter score tables



Root

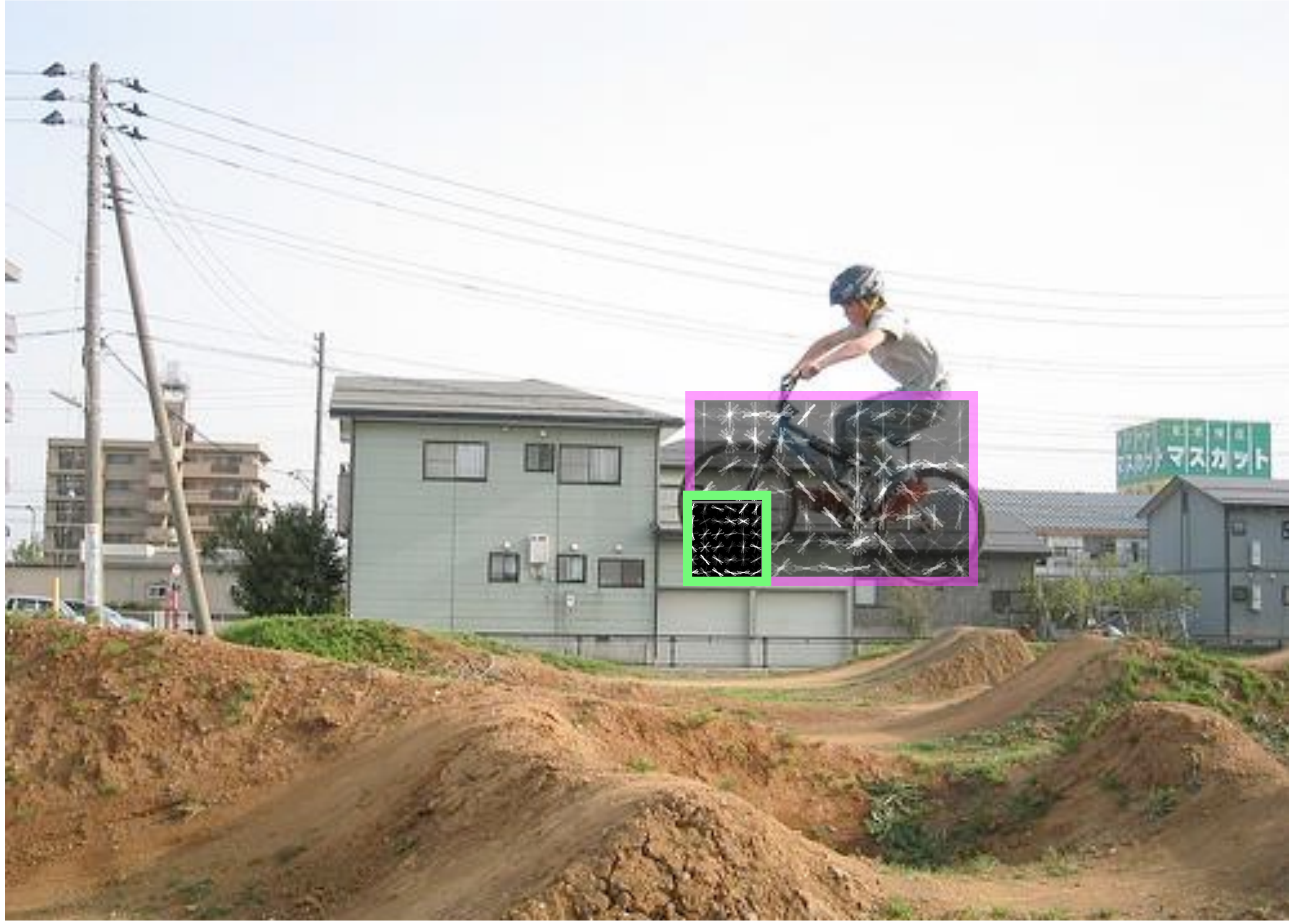
$m_0(\omega)$

Part 1

$m_1(\omega)$

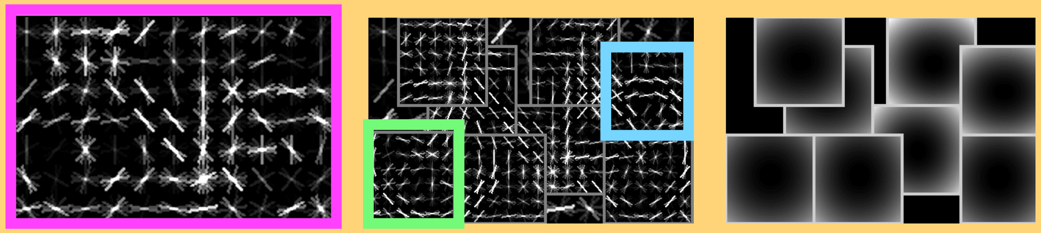
Part 2

$m_2(\omega)$



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

slide credit: Girshick et al

Star cascade algorithm

filter score tables

Root

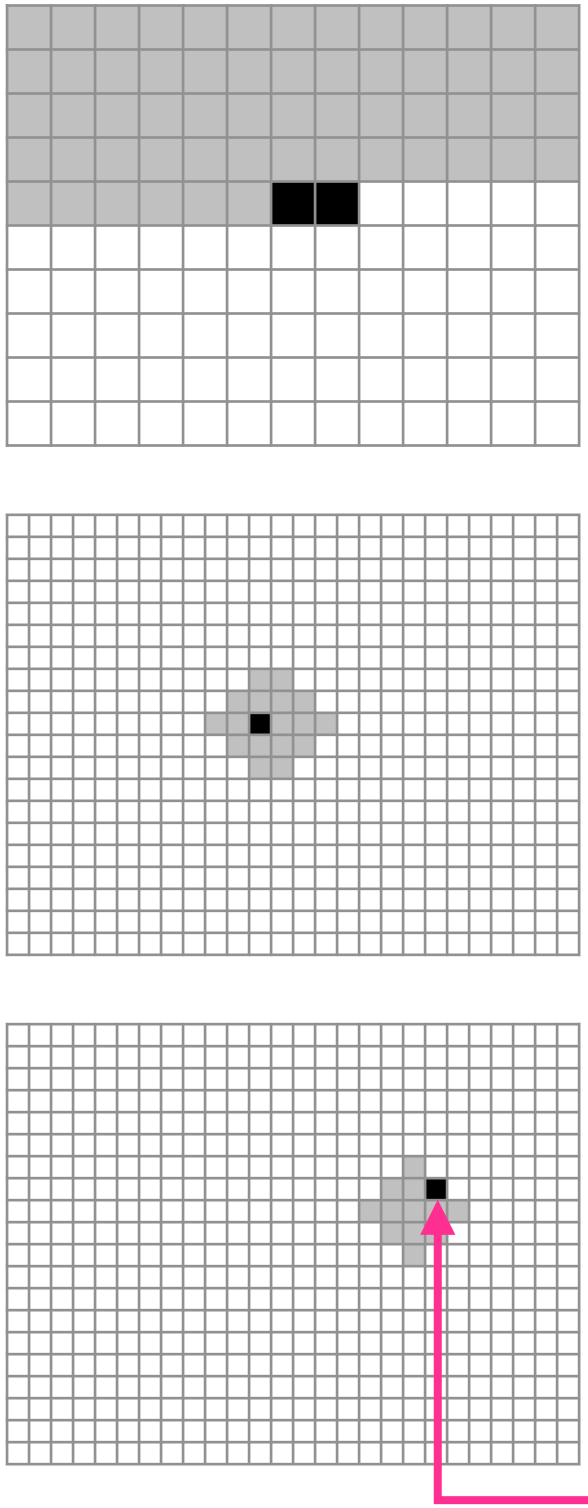
$m_0(\omega)$

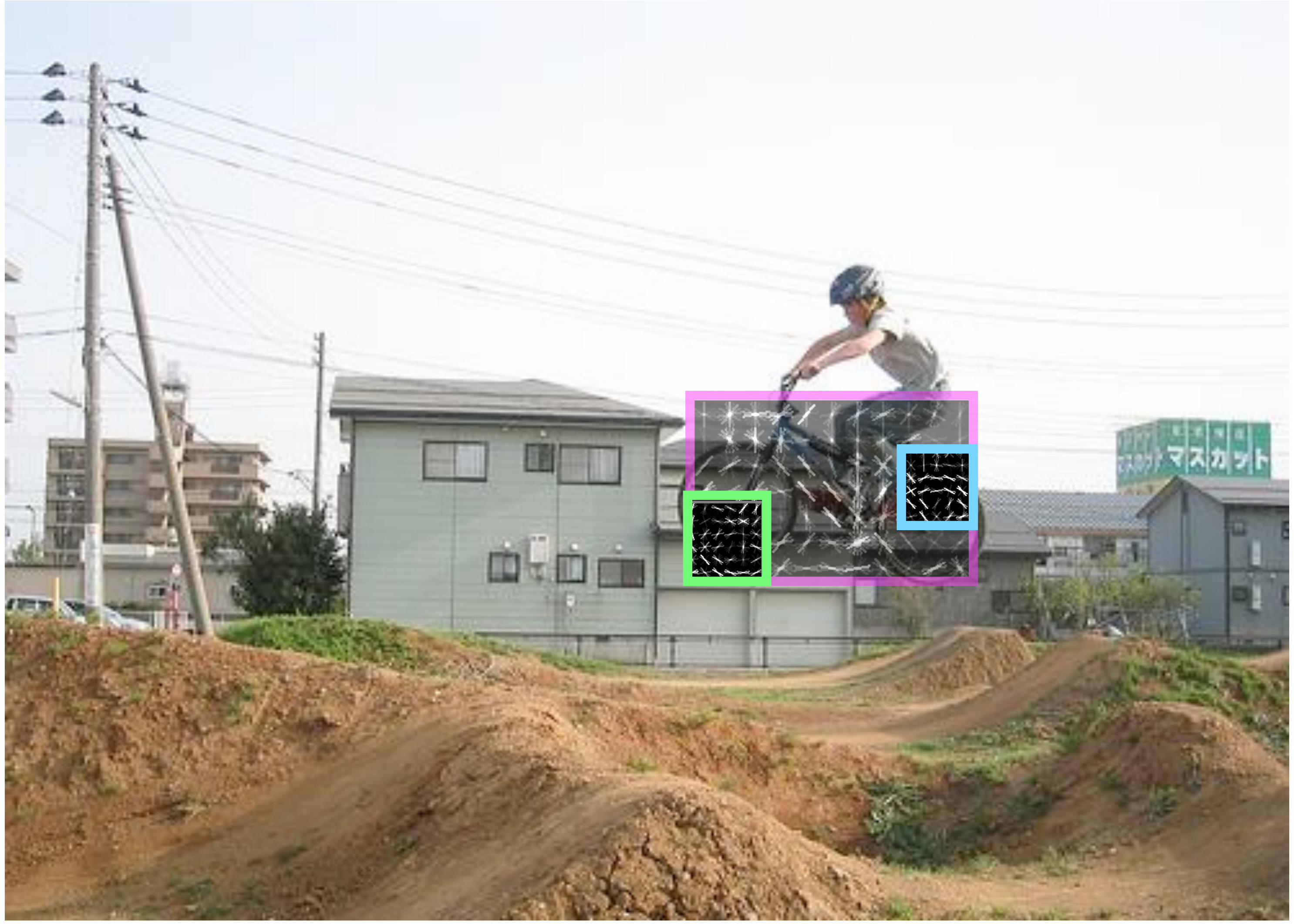
Part 1

$m_1(\omega)$

Part 2

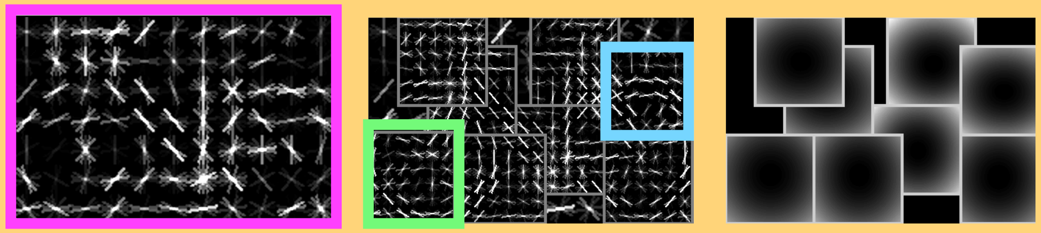
$m_2(\omega)$





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

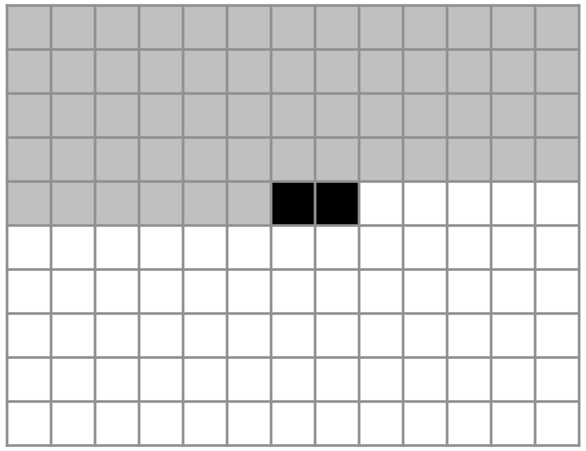
slide credit: Girshick et al

Star cascade algorithm

filter score tables

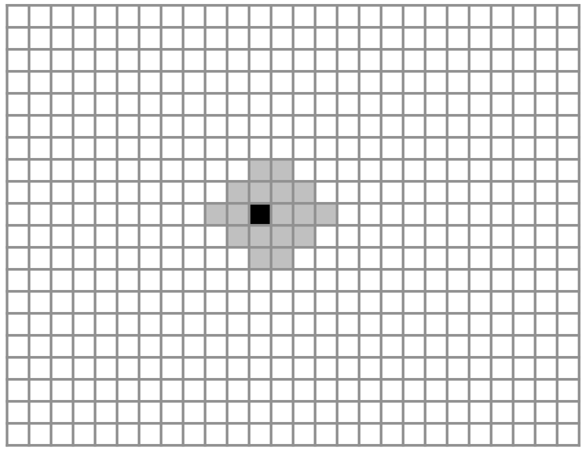
Root

$m_0(\omega)$



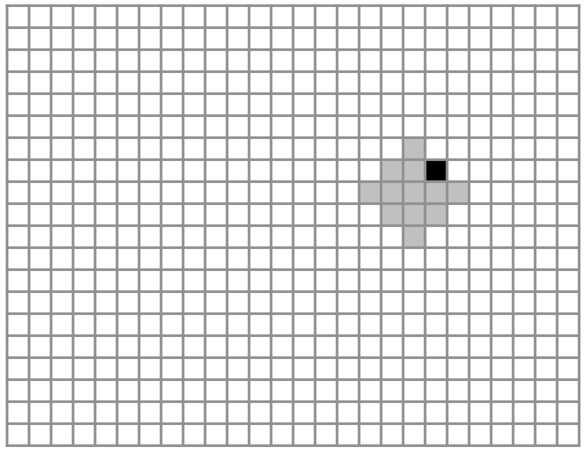
Part 1

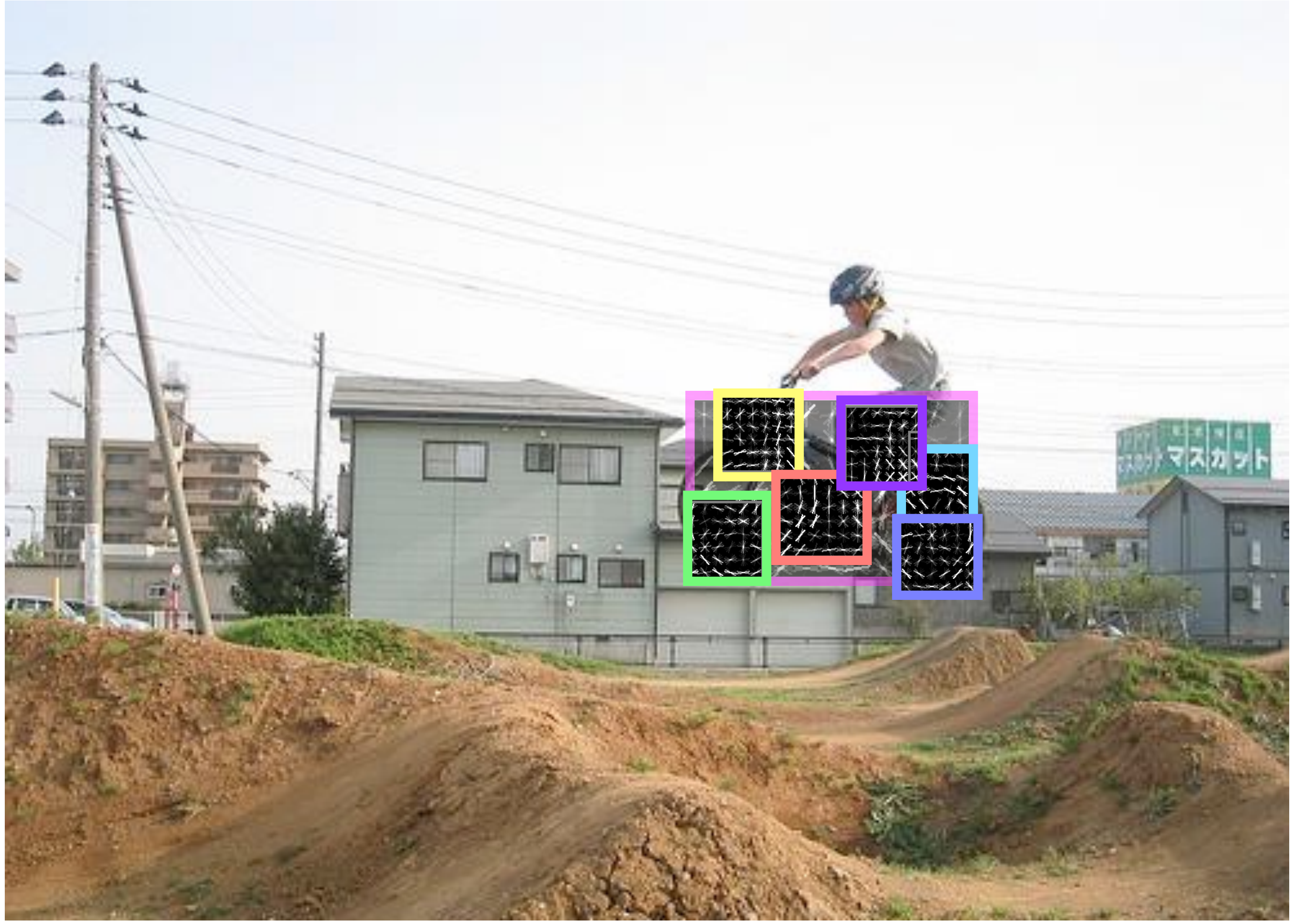
$m_1(\omega)$



Part 2

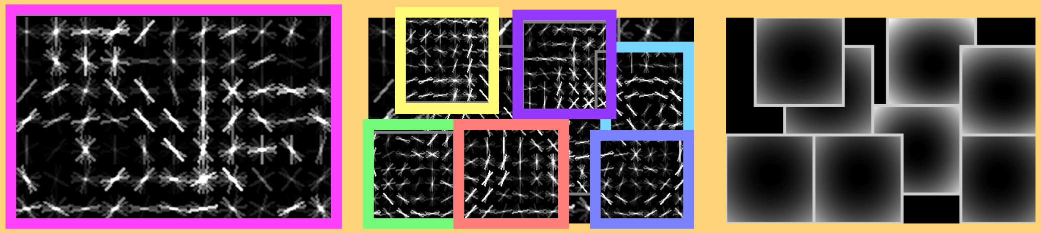
$m_2(\omega)$





cascade test: ...

model:

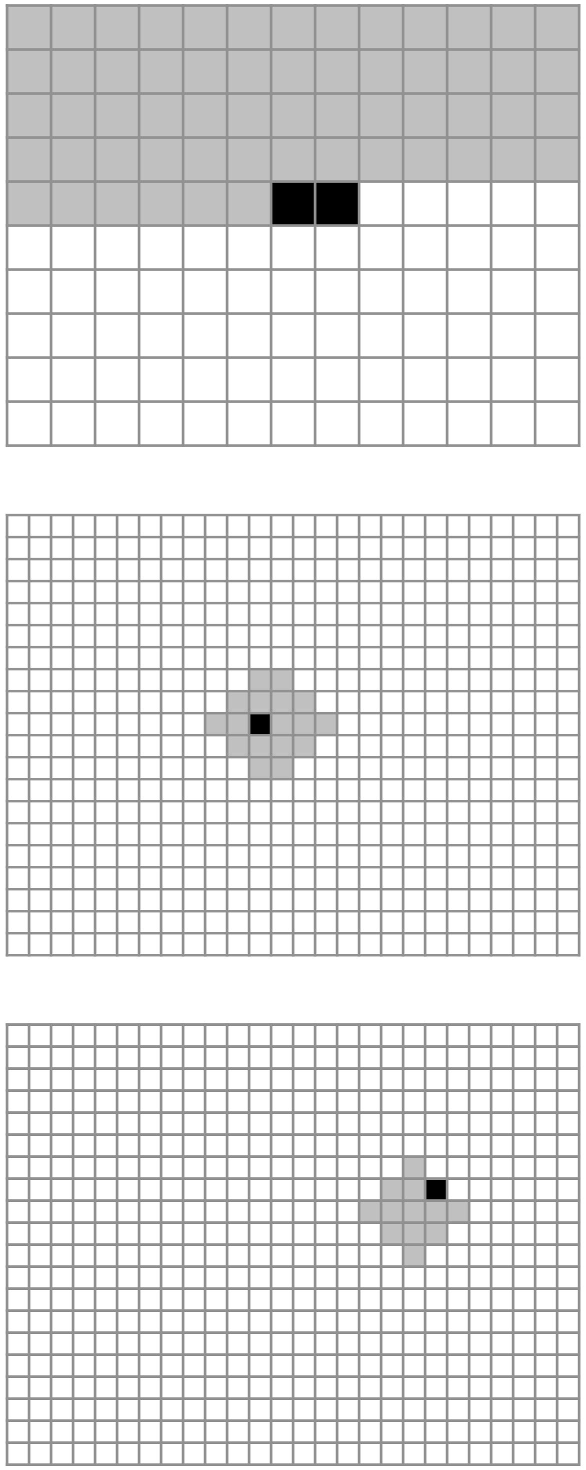


operation: continue testing remaining parts

slide credit: Girshick et al

Star cascade algorithm

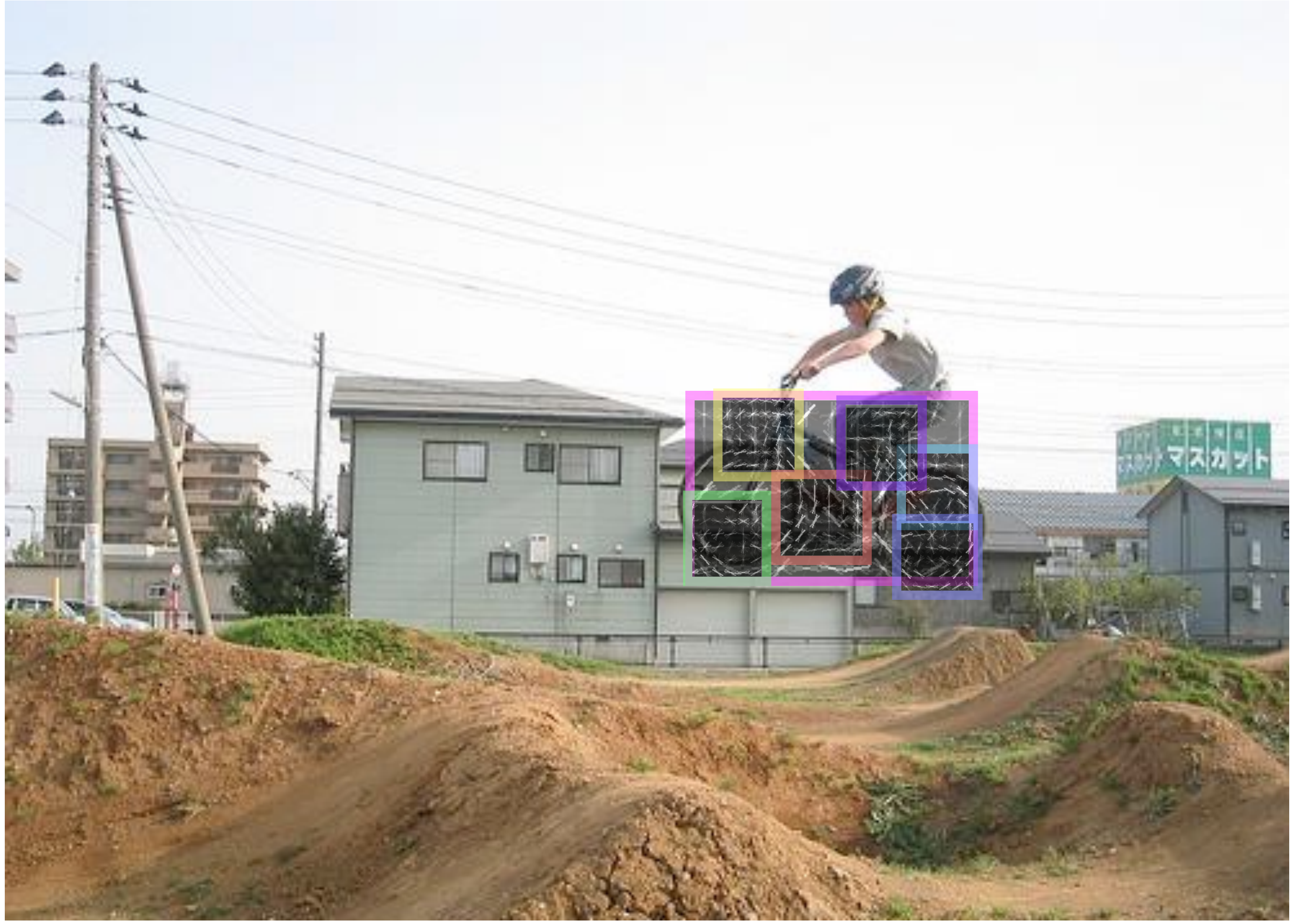
filter score tables



Root
$m_0(\omega)$

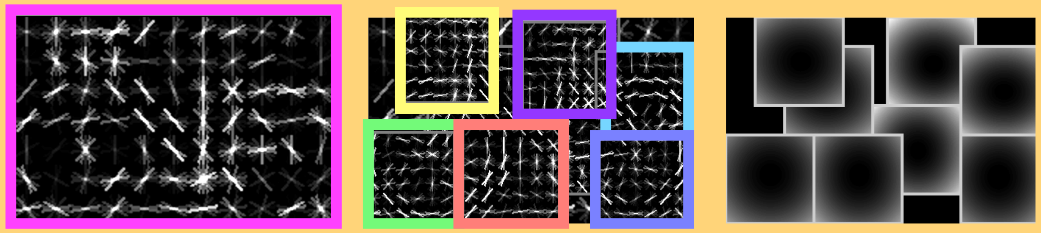
Part 1
$m_1(\omega)$

Part 2
$m_2(\omega)$



cascade test: all tests passed => detection!

model:

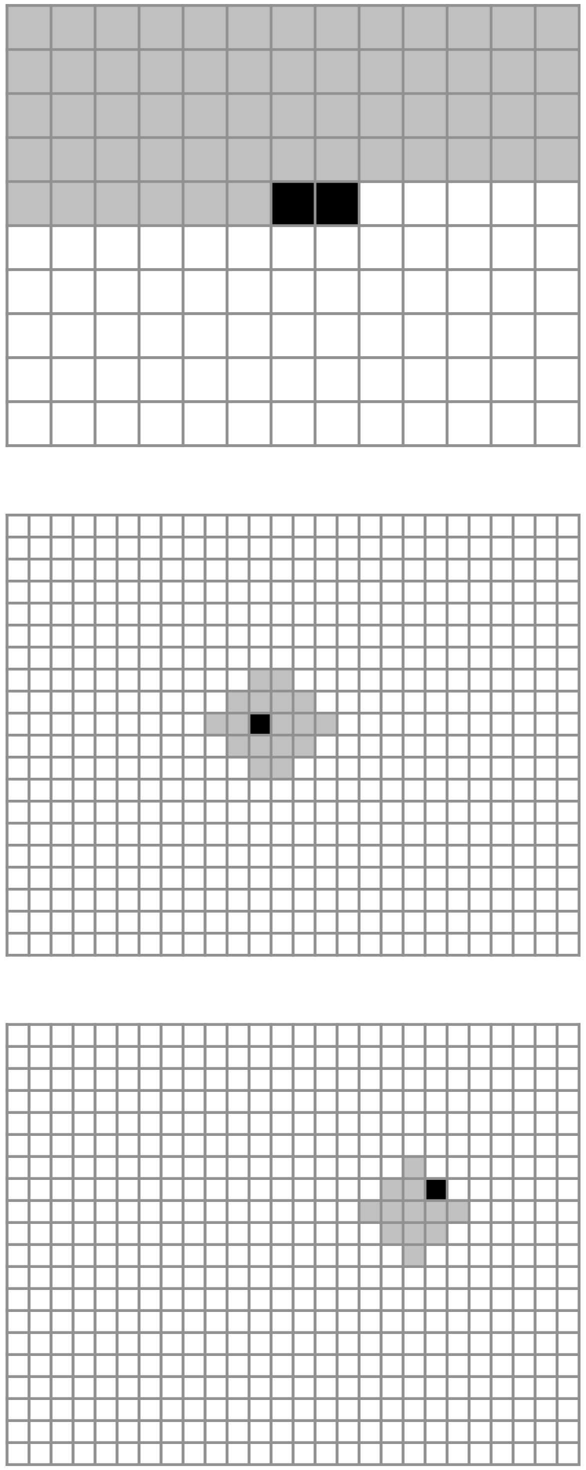


operation: report object hypothesis

slide credit: Girshick et al

Star cascade algorithm

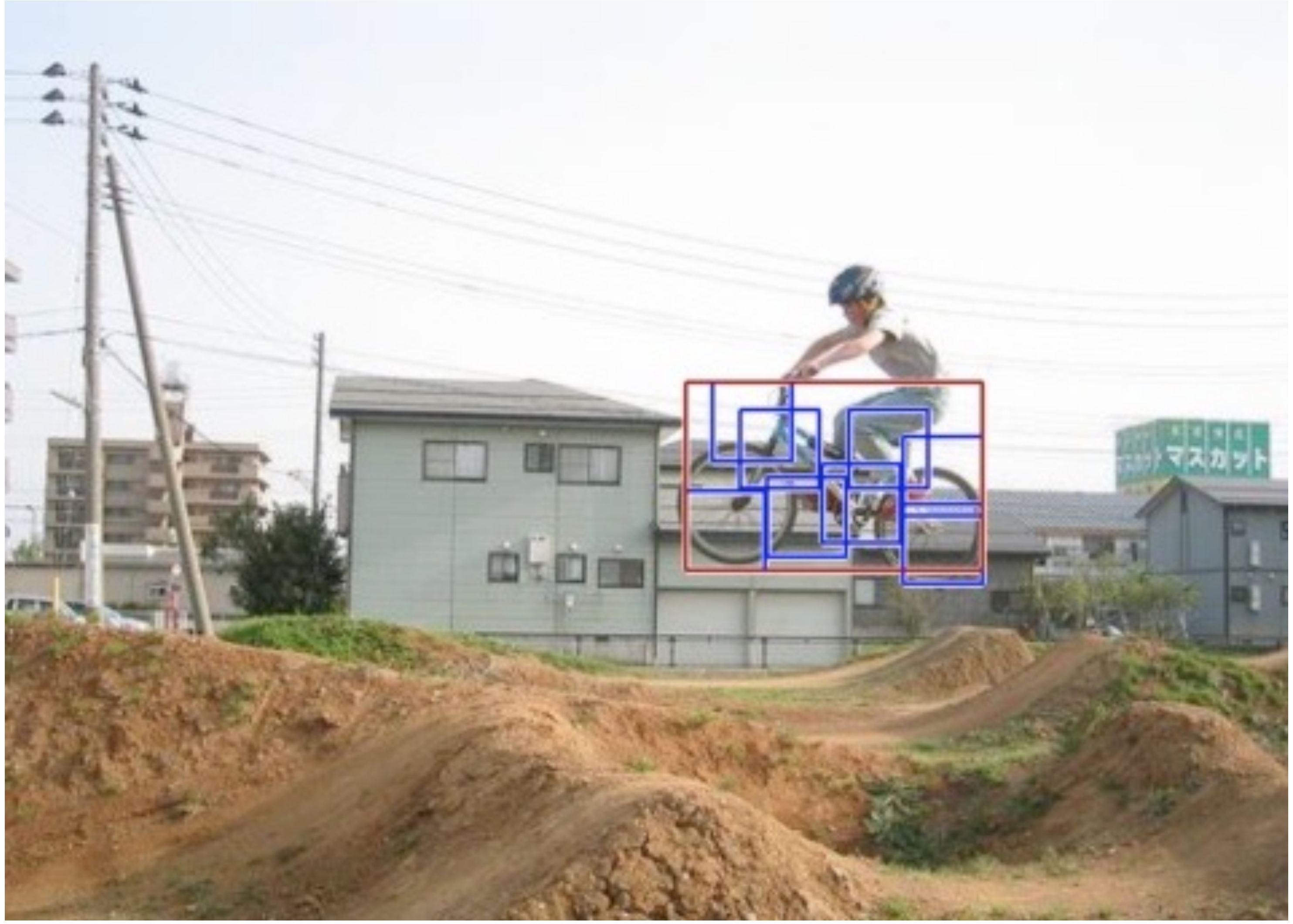
filter score tables



Root
$m_0(\omega)$

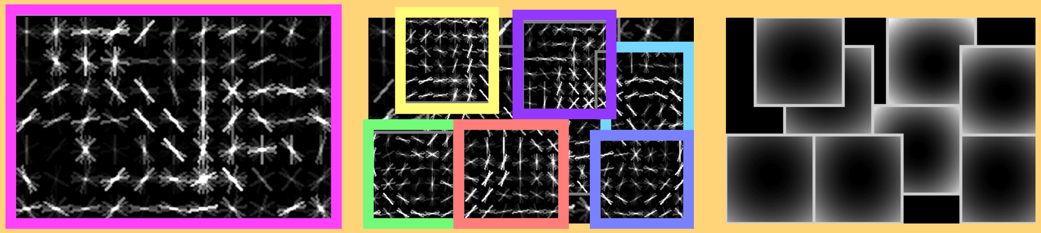
Part 1
$m_1(\omega)$

Part 2
$m_2(\omega)$



cascade test:

model:



operation: continue with root locations...

slide credit: Girshick et al

Threshold selection

don’t prune many true positives

We want *safe* and *effective* thresholds

but do prune lots of true negatives

slide credit: Girshick et al

PAA threshold

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

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

Probably Approximately Admissible thresholds

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

↓

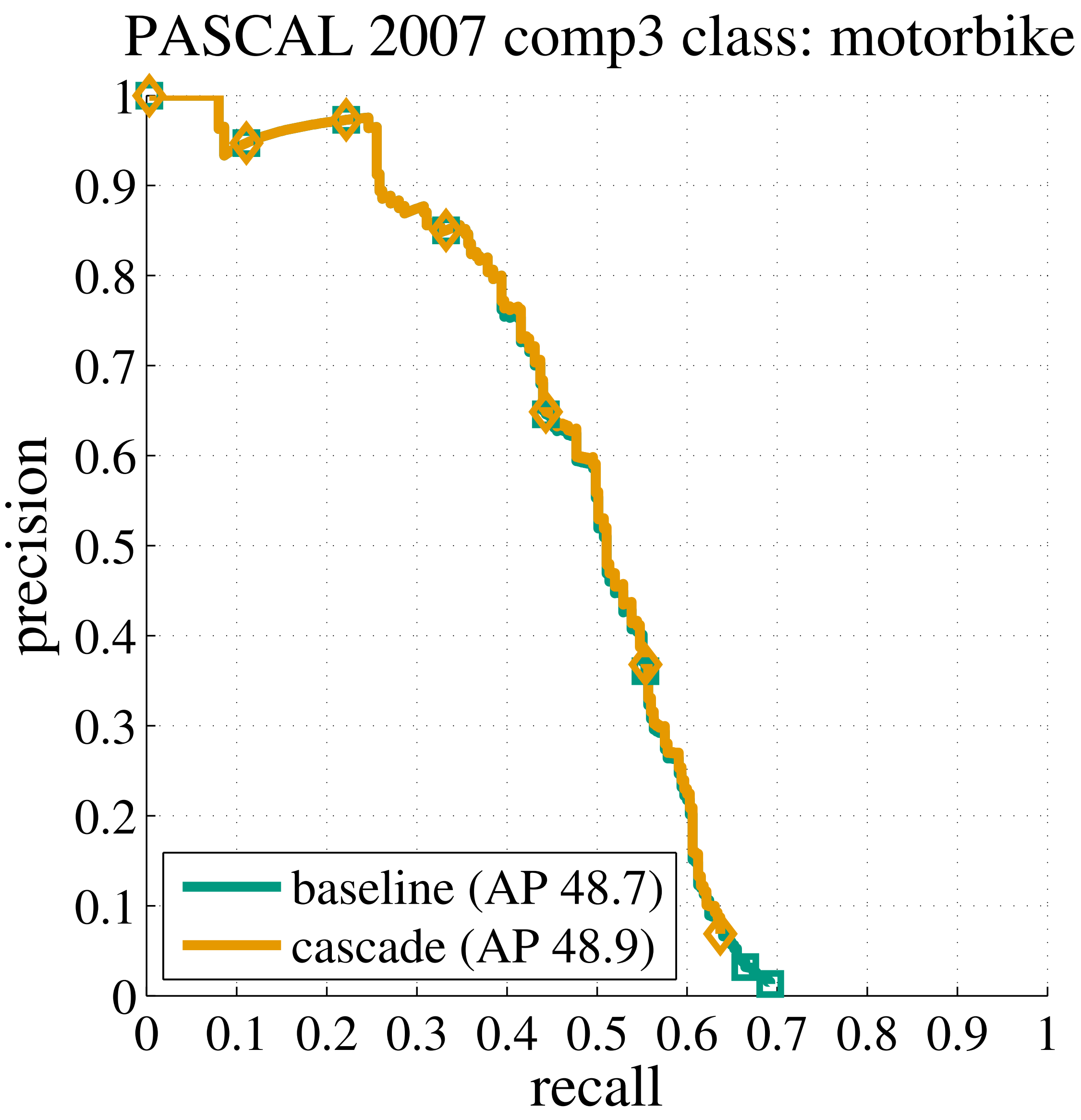
`min` of partial scores over examples in X

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

slide credit: Girshick et al

Example results

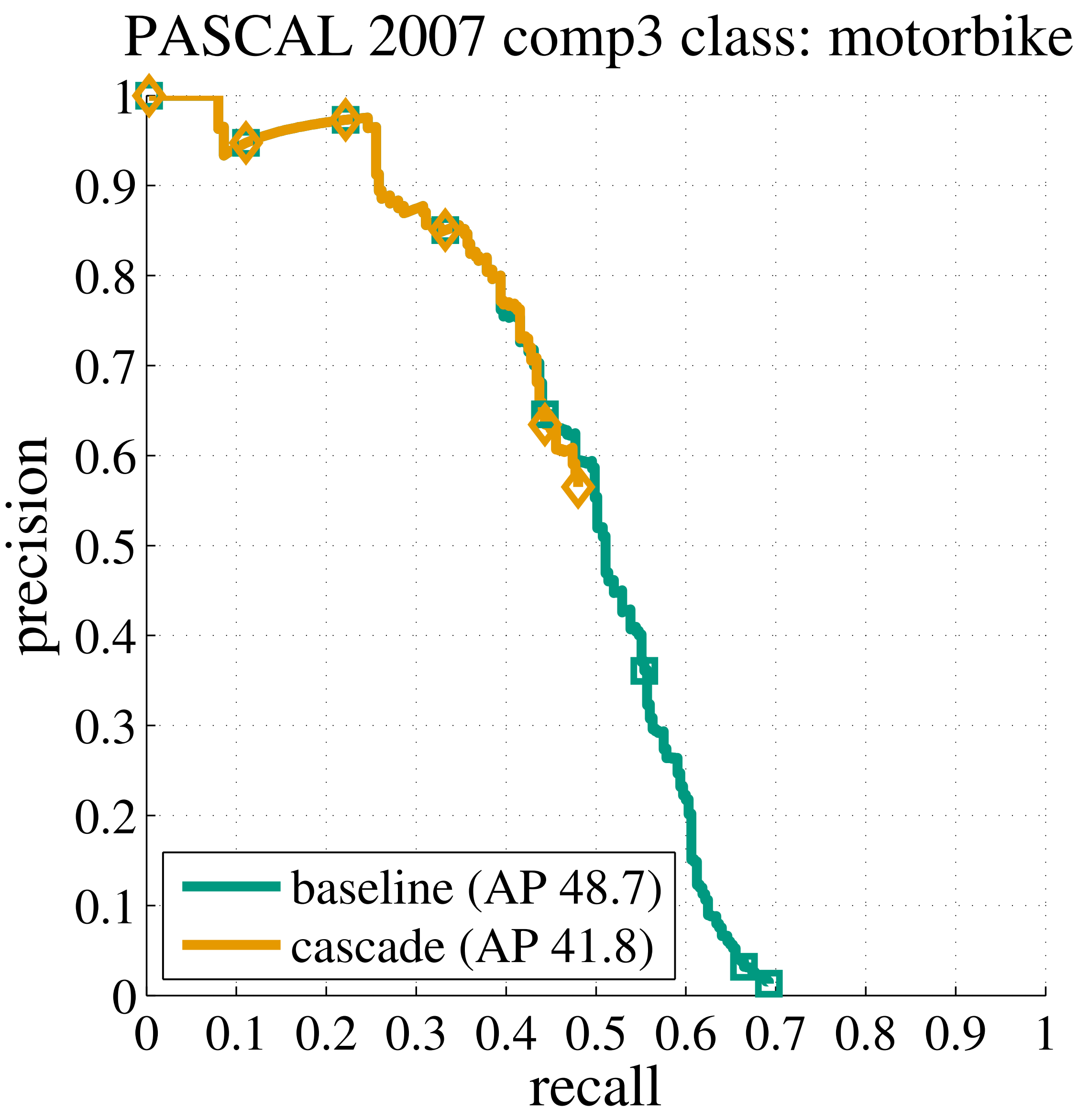
high recall



23.2x faster

(618ms per/image)

less recall ⇒ faster



31.6x faster

(454ms per/image)

Discussion

Contents

- Sliding window object detection
- Deformable part models
- Cascade DPM
- **Sparselets**
- Hashing based

Generalized Sparselet Models for Real-Time Multiclass Object Recognition

Hyun Oh Song, Ross Girshick, Stefan Zickler, Christopher Geyer,
Pedro Felzenszwalb, Trevor Darrell

ECCV12, ICML13, TPAMI14

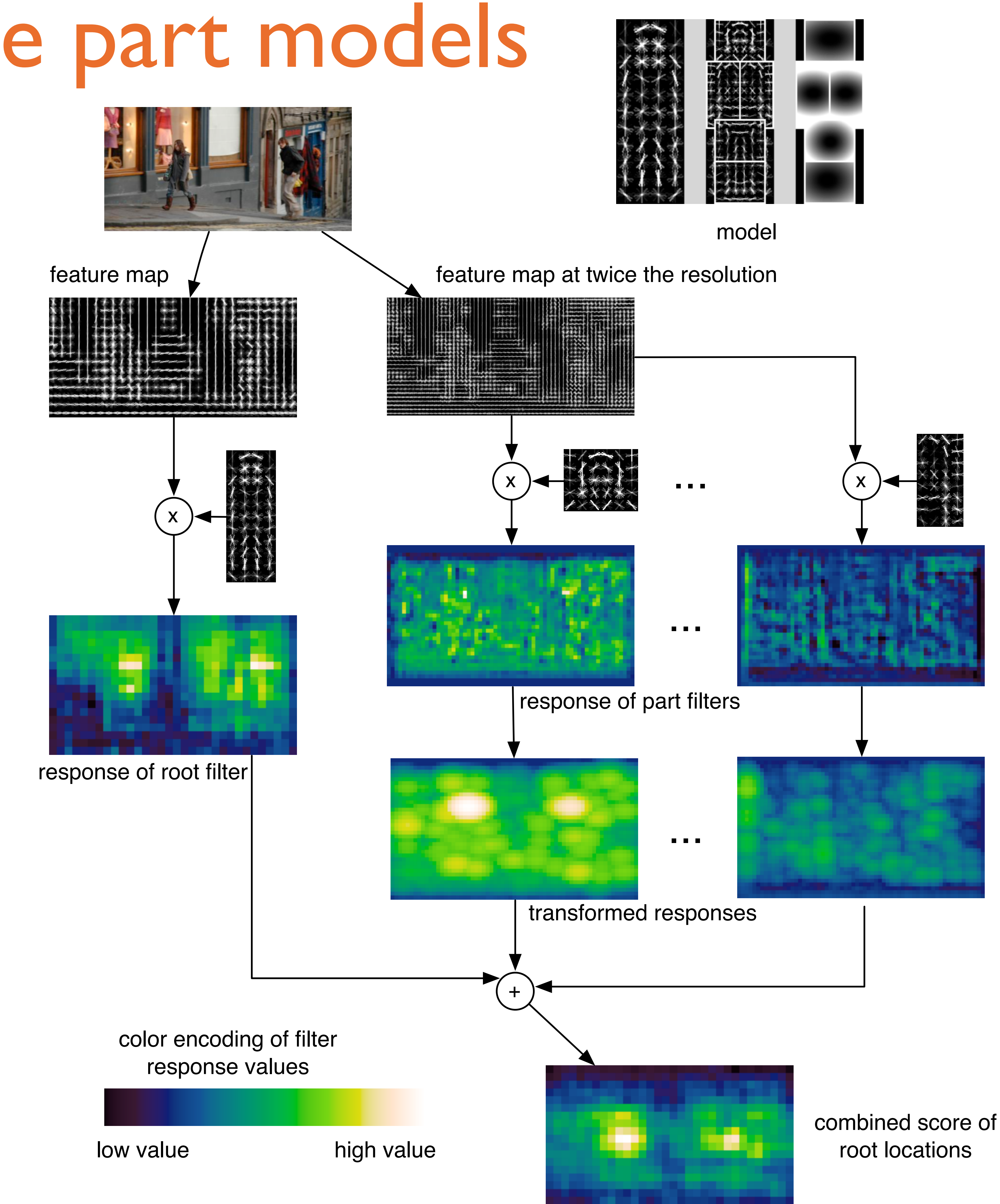
Goal

- Shared predictive model with sparse activation vectors
- Efficient inference for linear structured output predictors
- Example application: realtime object recognition in CV, faster retrieval in IR, etc.

Related works

- Learning shared low dimensional predictive structure (e.g., Ando and Zhang, JMLR05)
- Shared part models (Steerable part models, Pirsiavash et al)

Deformable part models



model

feature map

feature map at twice the resolution

x

x

...

x

response of root filter

...

response of part filters

...

transformed responses

+

color encoding of filter
response values

low value

high value

combined score of
root locations

Felzenszwalb et al,
PAMI 2010

Sparselet review

Set of model filters $\mathcal{W} = \{\mathbf{w_1}, ..., \mathbf{w_K}\}$

Set of sparselet filters $\mathcal{S} \ = \{\mathbf{s_1}, ..., \mathbf{s_d}\}$

$$\min_{\alpha_{ij}, s_j} \sum_{i=1}^{K} ||\mathbf{w}_i - \sum_{j=1}^{d} \alpha_{ij}\mathbf{s}_j||_2^2$$

$$\text{subject to} \quad ||\boldsymbol{\alpha_i}||_0 \leq \epsilon \qquad \forall i = 1, ..., K$$

$$||\mathbf{s}_j||_2^2 \leq 1 \qquad \forall j = 1, ..., d$$

Sparse reconstruction of filter response

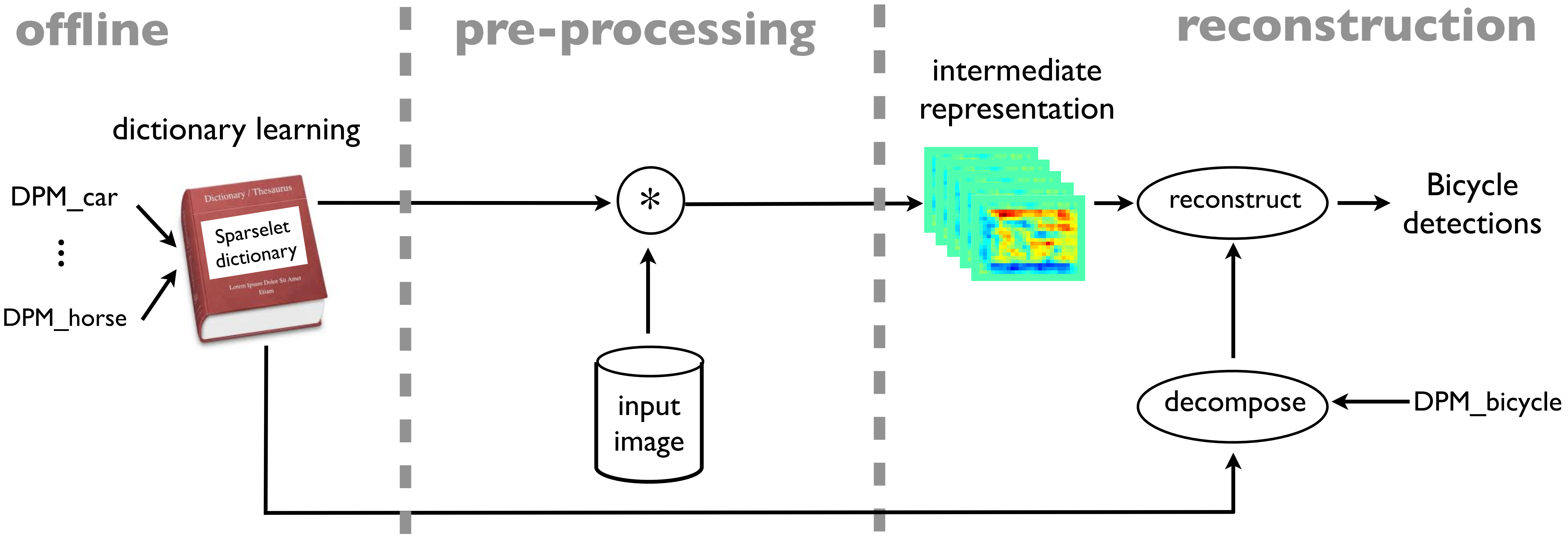
$$\Psi * \mathbf{w}_i \approx \Psi * \left(\sum_{\substack{j=1 \\ \forall \alpha_{ij} \neq 0}}^{d} \alpha_{ij} \mathbf{s}_j \right) = \sum_{\substack{j=1 \\ \boxed{\forall \alpha_{ij} \neq 0} \\ \text{Sparsity}}}^{d} \alpha_{ij} \underset{\text{Cached}}{\underline{\left(\Psi * \mathbf{s}_j \right)}}$$

Matrix factorization point of view

$$\begin{bmatrix} \text{---} \; \Psi * \mathbf{w}_1 \; \text{---} \\ \text{---} \; \Psi * \mathbf{w}_2 \; \text{---} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \text{---} \; \Psi * \mathbf{w}_K \; \text{---} \end{bmatrix} \approx \boxed{\begin{bmatrix} \text{---} \; \boldsymbol{\alpha}_1 \; \text{---} \\ \text{---} \; \boldsymbol{\alpha}_2 \; \text{---} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \text{---} \; \boldsymbol{\alpha}_K \; \text{---} \end{bmatrix}} \begin{bmatrix} \text{---} \; \Psi * \mathbf{s}_1 \; \text{---} \\ \text{---} \; \Psi * \mathbf{s}_2 \; \text{---} \\ \vdots \\ \text{---} \; \Psi * \mathbf{s}_d \; \text{---} \end{bmatrix}$$

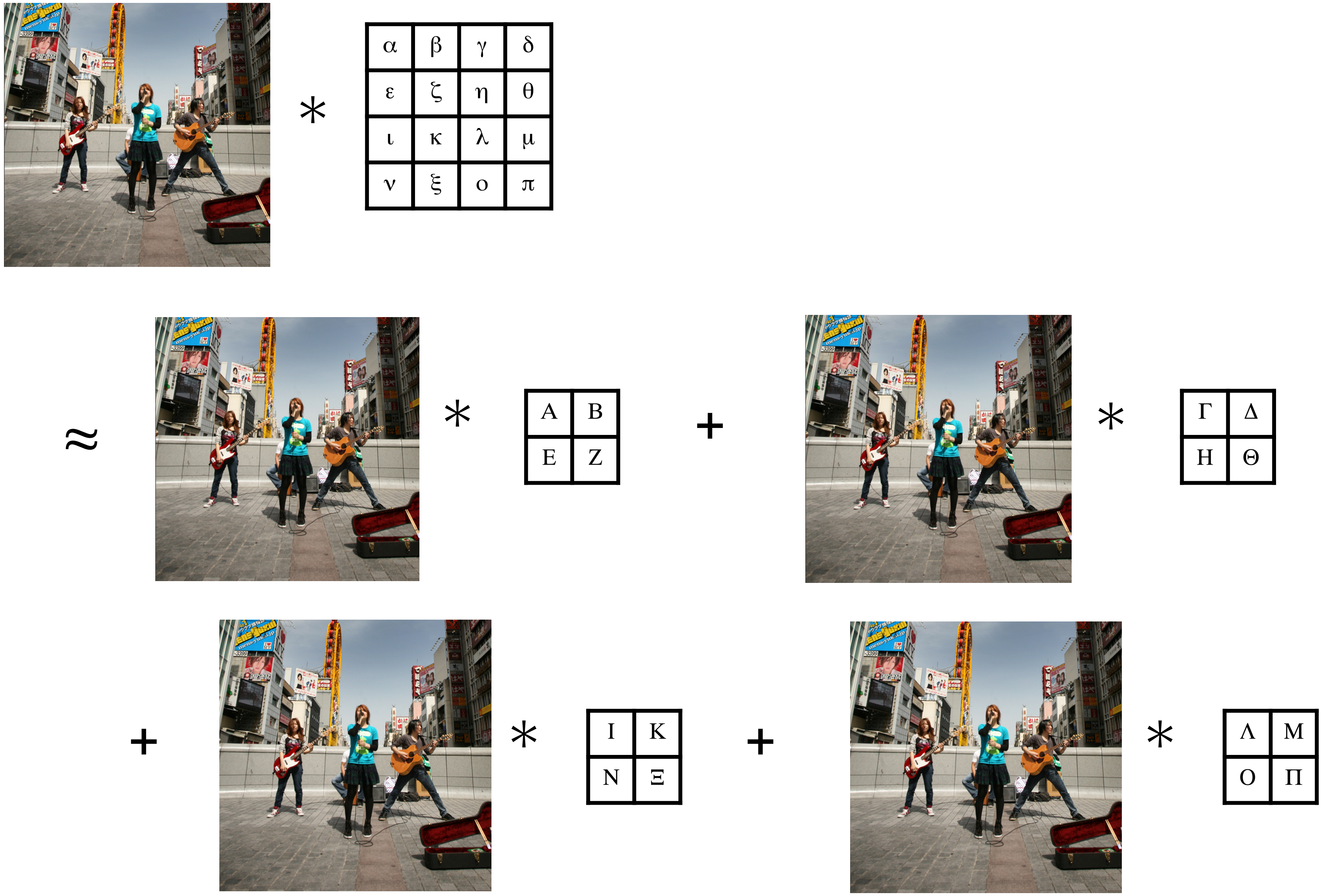
80 ~ 99 % Sparse

System concept



Blocked representation

- Intuition: model weights might be composed of shared building blocks/tiles

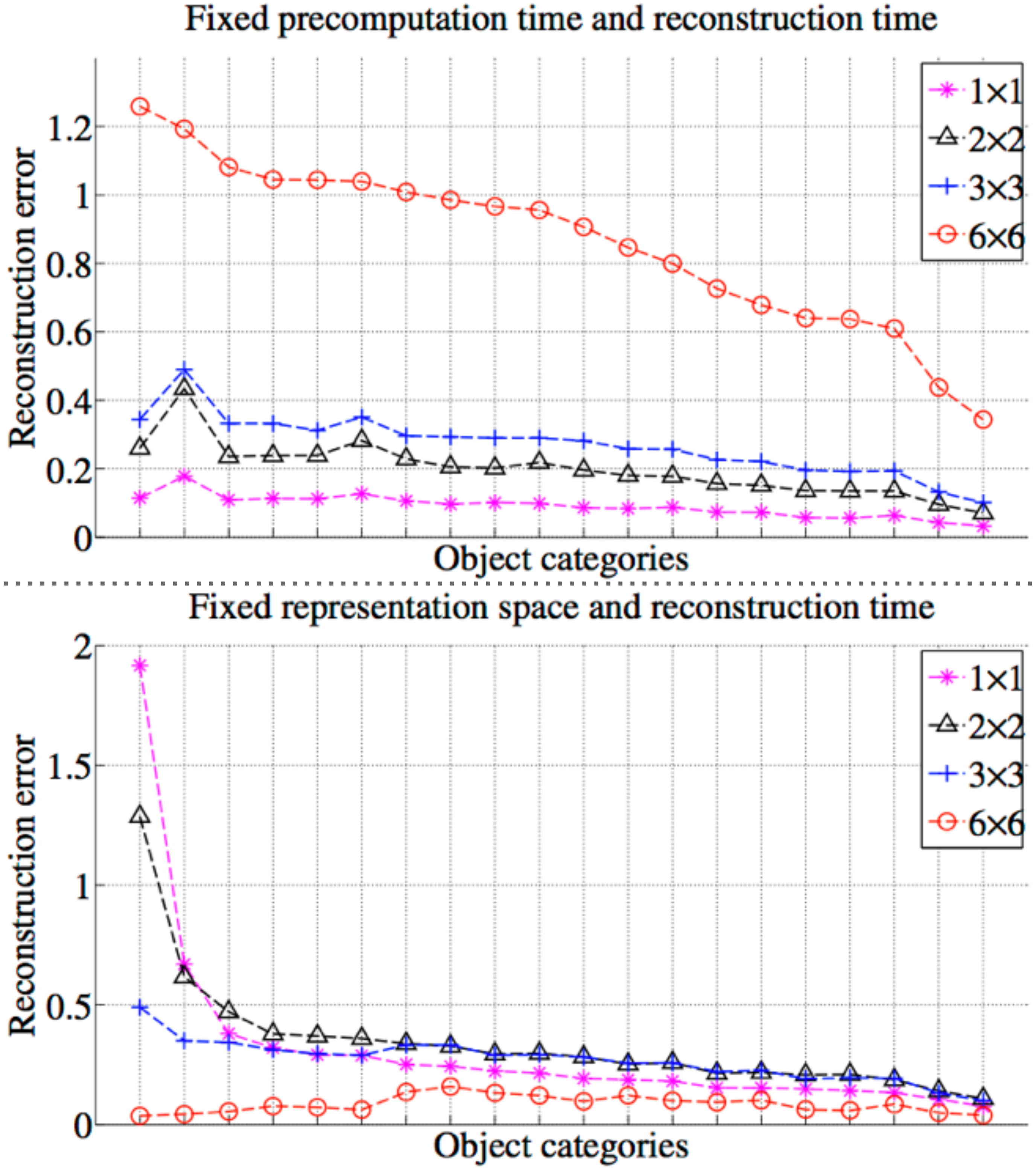


$$* \begin{bmatrix} \alpha & \beta & \gamma & \delta \\ \varepsilon & \zeta & \eta & \theta \\ \iota & \kappa & \lambda & \mu \\ \nu & \xi & o & \pi \end{bmatrix}$$

$$\approx \; * \begin{bmatrix} A & B \\ E & Z \end{bmatrix} + \; * \begin{bmatrix} \Gamma & \Delta \\ H & \Theta \end{bmatrix}$$

$$+ \; * \begin{bmatrix} I & K \\ N & \Xi \end{bmatrix} + \; * \begin{bmatrix} \Lambda & M \\ O & \Pi \end{bmatrix}$$

Blocked representation



Fixed precomputation time and reconstruction time

Fixed representation space and reconstruction time

Increase dictionary size

Fix dictionary size

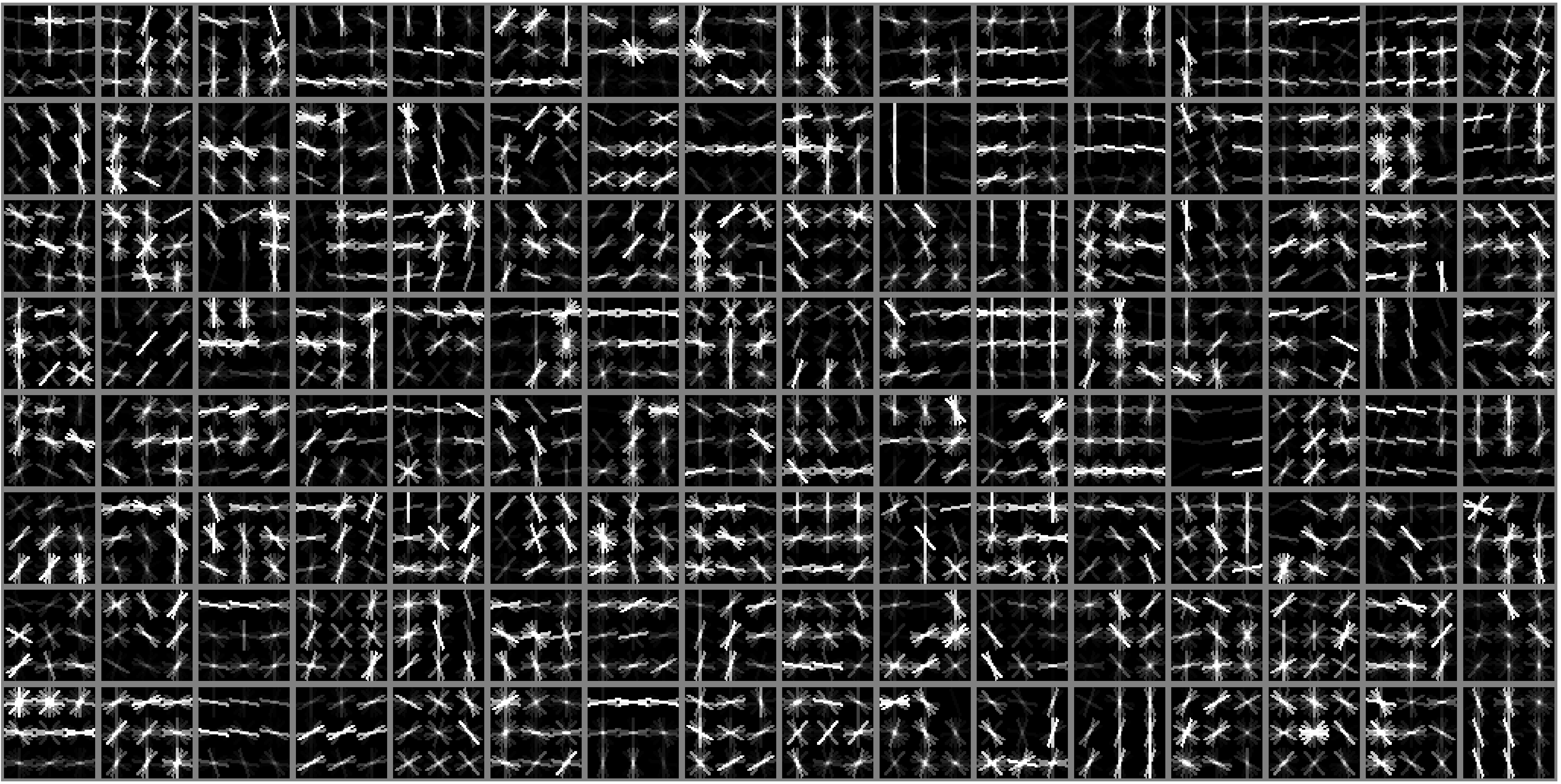
Reconstruction error for all 20 object categories from PASCAL 2007 dataset as sparselet parameters are varied. The precomputation time is fixed in the top figure and the representation space is fixed on the bottom. Object categories are sorted by the reconstruction error by 6×6 in the top figure and by 1×1 in the bottom figure.

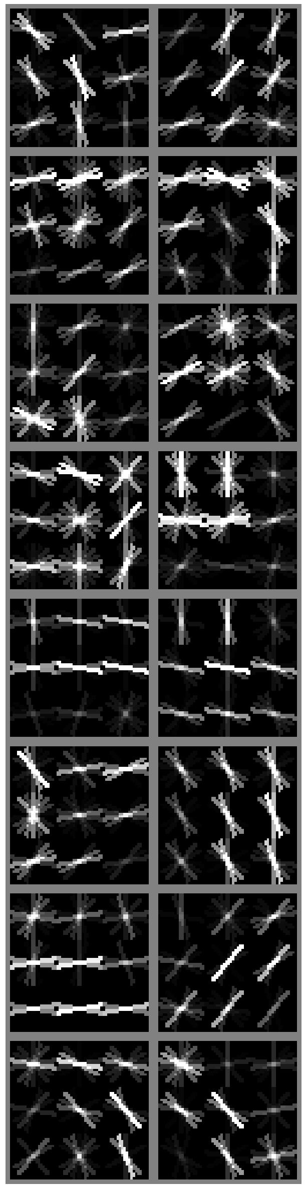
Blocked representation

- Empirically, filter reconstruction error always decreases as we decrease sparselet size (@ fixed computation time)

- However, the space required to store the intermediate representation is proportional to the sparselet dictionary size $|S|$. This means we have computation time VS memory bandwidth tradeoff.

Visualized sparselet blocks on HOG





(Left) Sparselet dictionary of size 128

(Right) Top 16 activated sparselets for PASCAL motorcycle class

Blocked representation

$$f_{\mathbf{w}}(\mathbf{x}) = \operatorname*{argmax}_{k \in \{1,\ldots,K\}} \mathbf{w}_k^\intercal \mathbf{x}$$

Model parameterization

$$\mathbf{w}_k = (\mathbf{b}_{k1}^\intercal, \ldots, \mathbf{b}_{kp}^\intercal)^\intercal$$

Data parameterization

$$\mathbf{x} = (\mathbf{c}_1^\intercal, \ldots, \mathbf{c}_p^\intercal)^\intercal$$

Sparselets approximation of model blocks

$$\mathbf{b} \approx \mathbf{S}\boldsymbol{\alpha} = \sum_{\substack{i=1 \\ \alpha_i \neq 0}}^{d} \alpha_i \mathbf{s}_i$$

Sparselets: $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_d]$

Sparselet Demo

Demo specifications

- Alienware laptop with NVIDIA GeForce GTX580 with 3GB memory

- Runs all 20 PASCAL category detection @ 5 Hz (frames per second)

- Full specs and quantitative average precision results in Song et al, *TPAMI15*

- *CPU version of the source code available at https://github.com/rksltnl/sparselet-release1*

Potential mobile implementation

- NVIDIA Shield supports CUDA with < 2GB memory

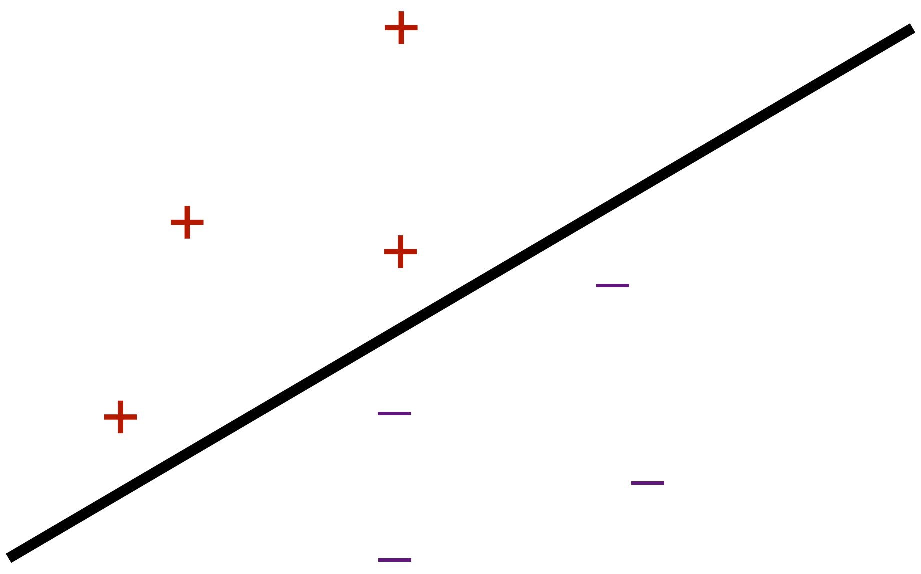
- ARM NEON optimizations on CPU side

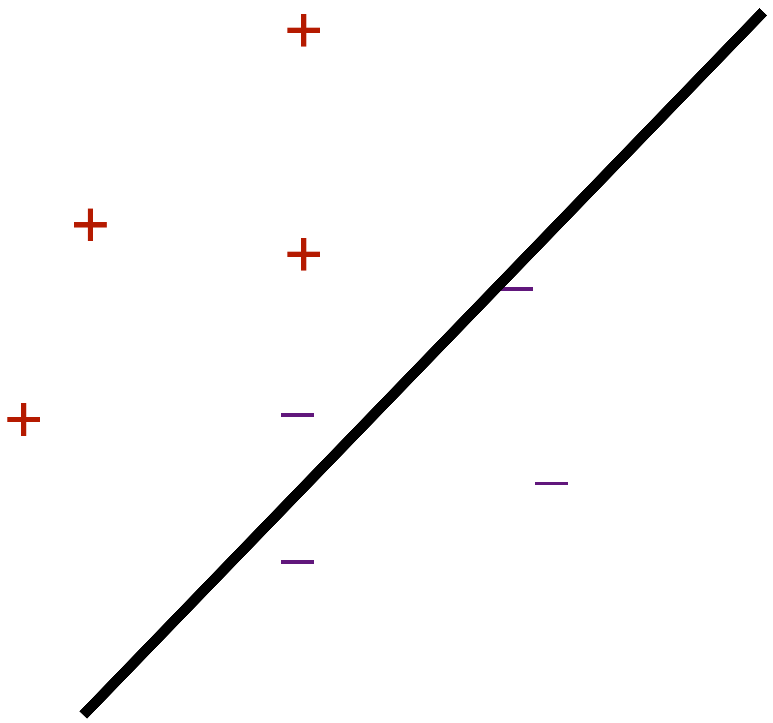
Discriminative sparselet activation

$$f_{\mathbf{w}}(\mathbf{x}) = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \; \mathbf{w}_k^\intercal \mathbf{x}$$

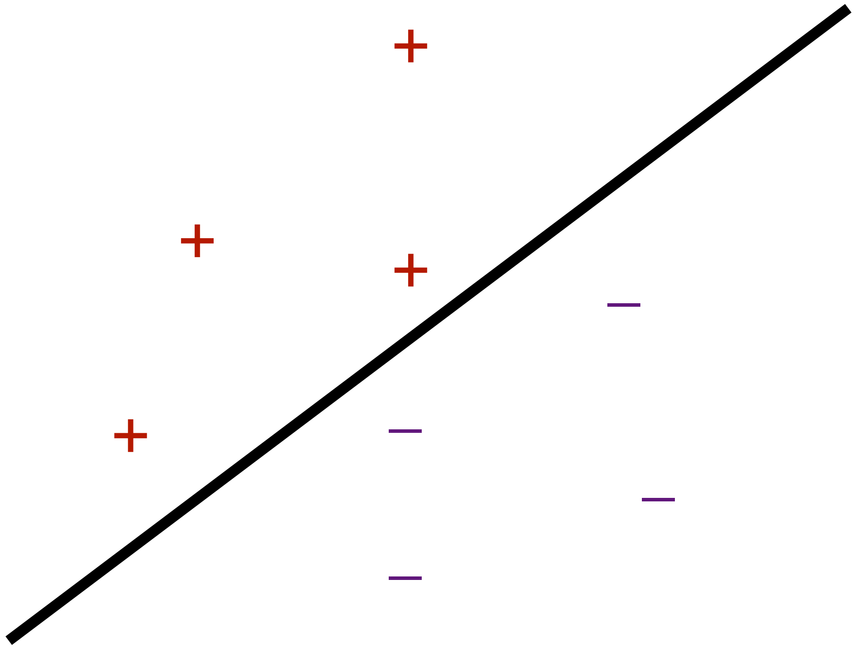
Original w$_k$

Sparselet approximation w$_k$





(i) Reconstructive
ECCV 12



(ii) Discriminative
ICML 13

Learning parameterization

$$
\begin{aligned}
\mathbf{w}_k^\top \mathbf{x} &= (\mathbf{b}_{k1}^\top, \ldots, \mathbf{b}_{kp}^\top)(\mathbf{c}_1^\top, \ldots, \mathbf{c}_p^\top)^\top \\
&= \sum_{i=1}^{p} \mathbf{b}_{ki}^\top \mathbf{c}_i \approx \sum_{i=1}^{p} (\mathbf{S}\boldsymbol{\alpha}_{ki})^\top \mathbf{c}_i = \sum_{i=1}^{p} \boldsymbol{\alpha}_{ki}^\top (\mathbf{S}^\top \mathbf{c}_i)
\end{aligned}
$$

Model parameter: sparse activation vector

Feature: sparselet response

Structural SVM for DAS

Parameter vector $\boldsymbol{\beta} = (\boldsymbol{\alpha}^\intercal, \tilde{\mathbf{w}}^\intercal)^\intercal$

Transformed features $\tilde{\boldsymbol{\Phi}}_k\,(x, y) = \left(\mathbf{c}_1^\intercal S, \ldots, \mathbf{c}_{p_k}^\intercal S\right)^\intercal$

Aggregate feature vector

$$\tilde{\boldsymbol{\Phi}}(x, y) = (\tilde{\boldsymbol{\Phi}}_1^\intercal(x, y), \ldots, \tilde{\boldsymbol{\Phi}}_s^\intercal(x, y), \underline{\boldsymbol{\Phi}_{s+1}^\intercal(x, y), \ldots, \boldsymbol{\Phi}_K^\intercal(x, y)})^\intercal$$

projected feature slot remainder feature slot

Training

Discriminative activation of sparselets

$$\boldsymbol{\beta}^* = \operatorname*{argmin}_{\boldsymbol{\beta}} \underline{R(\boldsymbol{\alpha})} + \frac{\lambda}{2}\|\tilde{\mathbf{w}}\|_2^2 + \frac{1}{M}\sum_{i=1}^{M} \max_{\hat{y}\in\mathcal{Y}} \Big(\boldsymbol{\beta}^\intercal \tilde{\boldsymbol{\Phi}}(x_i, \hat{y}) + \Delta(y_i, \hat{y})\Big) - \boldsymbol{\beta}^\intercal \tilde{\boldsymbol{\Phi}}(x_i, y_i)$$

Sparsity inducing norm

Sparsity enforcing norms

I. **Lasso penalty** $R_{\text{Lasso}}(\boldsymbol{\alpha}) = \lambda_1 \|\boldsymbol{\alpha}\|_1$

II. **Elastic net penalty** $R_{\text{EN}}(\boldsymbol{\alpha}) = \lambda_1 \|\boldsymbol{\alpha}\|_1 + \lambda_2 \|\boldsymbol{\alpha}\|_2^2$

III. **Combined ℓ_0 and ℓ_2 penalty** $R_{0,2}(\boldsymbol{\alpha}) = \lambda_2 \|\boldsymbol{\alpha}\|_2^2$ subject to $\|\boldsymbol{\alpha}\|_0 \leq \lambda_0$

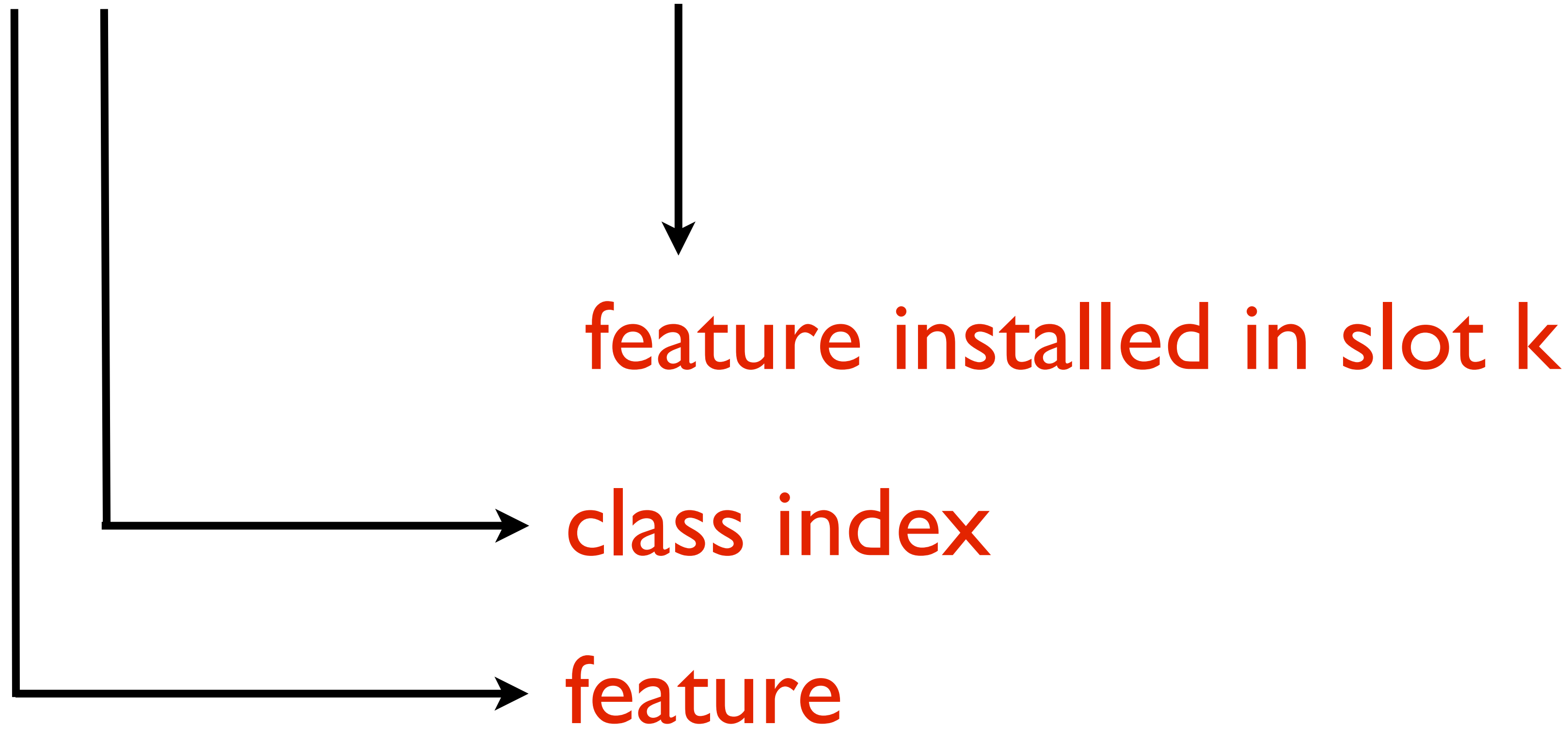
III-A. **Overshoot, rank, and threshold (ORT)**

III-B. **Orthogonal matching pursuit (OMP)**

Joint feature map: multiclass classification

$$\mathbf{w} = \left(\mathbf{w}_1^\intercal, \ldots, \mathbf{w}_K^\intercal\right)^\intercal$$

$$\boldsymbol{\Phi}\left(\mathbf{x}, k\right) = \left(0, \ldots, 0, \mathbf{x}^\intercal, 0, \ldots, 0\right)^\intercal$$



feature installed in slot k

class index

feature

Inference $$f_{\mathbf{w}}\left(\mathbf{x}\right) = \underset{k}{\operatorname{argmax}}\ \mathbf{w}^\intercal \boldsymbol{\Phi}\left(\mathbf{x}, k\right)$$

Joint feature map: multiclass classification with sparselets

$$\boldsymbol{\beta} = (\boldsymbol{\alpha}_1^\intercal, \ldots, \boldsymbol{\alpha}_K^\intercal, \underline{\tilde{\mathbf{w}}_1^\intercal, \ldots, \tilde{\mathbf{w}}_K^\intercal})^\intercal$$

$$\tilde{\boldsymbol{\Phi}}\left(\mathbf{x}, k\right) = \left(0, \ldots, 0, \left(\underline{\mathbf{c}_1^\intercal S, \ldots, \mathbf{c}_{p_k}^\intercal S}\right)^\intercal, 0, \ldots, 0, \underbrace{0, \ldots, 0, 1, 0, \ldots, 0}_{\text{per-class bias}}\right)^\intercal$$

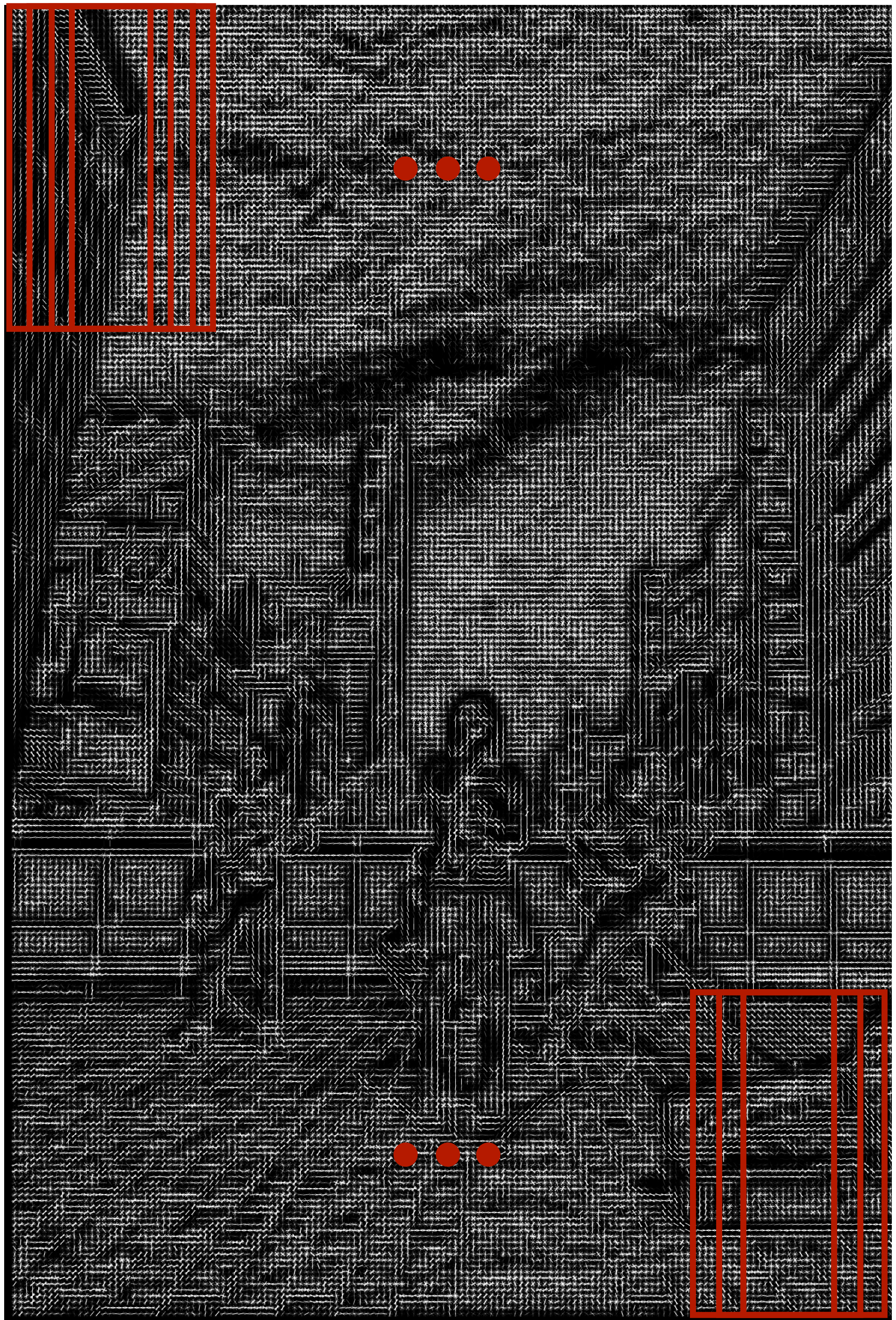
projected feature blocks installed in slot k

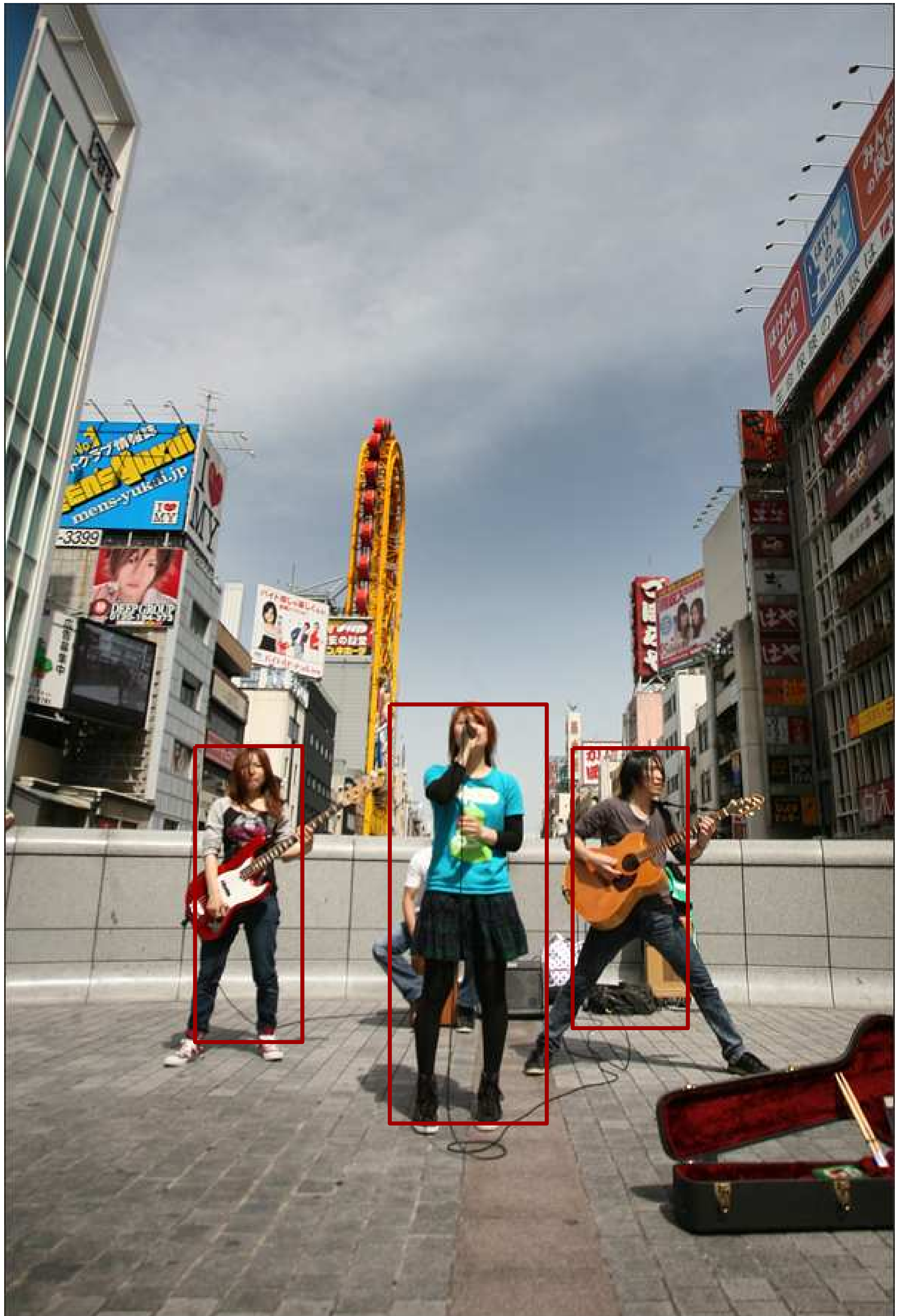
class index

feature

Inference $\quad f_{\boldsymbol{\beta}}\left(\mathbf{x}\right) = \underset{k}{\operatorname{argmax}} \ \boldsymbol{\beta}^\intercal \tilde{\boldsymbol{\Phi}}\left(\mathbf{x}, k\right)$

Object detection with HOG+SVM

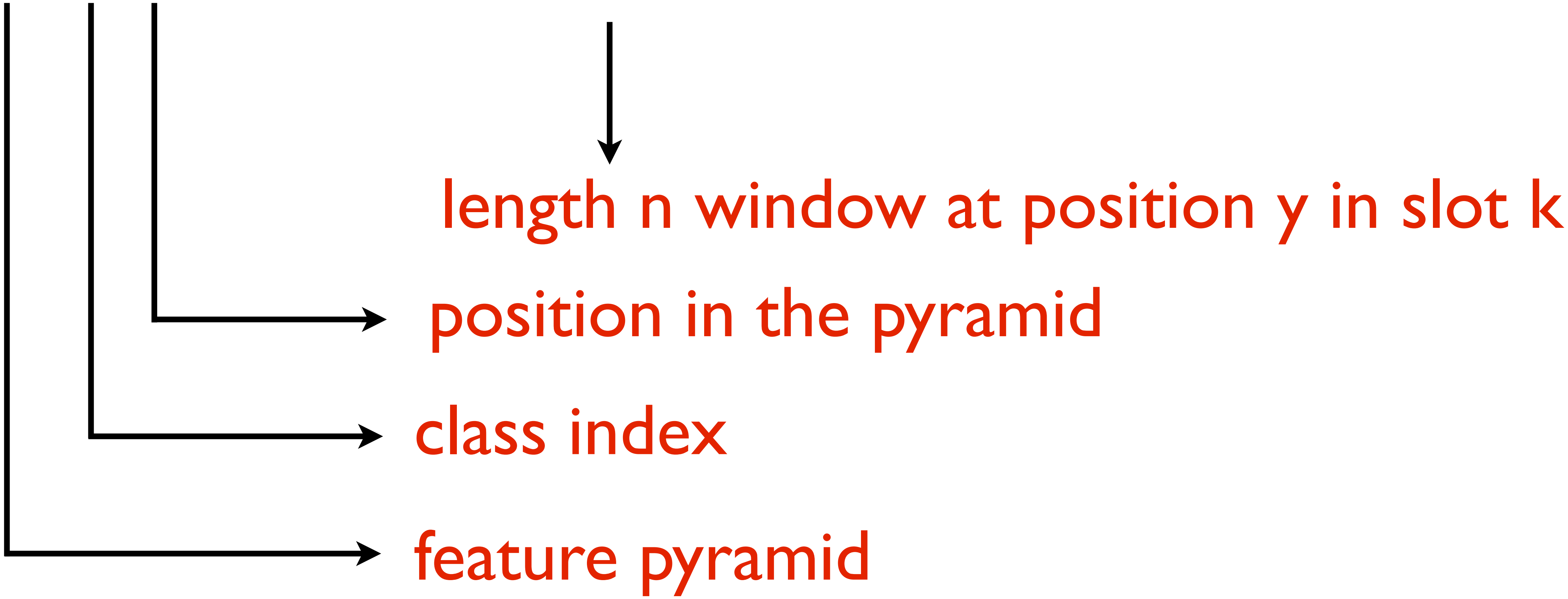




Joint feature map: object detection

$$\mathbf{w} = (\mathbf{w}_1^\top, \ldots, \mathbf{w}_K^\top)^\top$$

$$\mathbf{\Phi}(\mathbf{x}, (k, y)) = \left(0, \ldots, 0, \mathbf{x}_{y:n}^\top, 0, \ldots, 0\right)^\top$$



length n window at position y in slot k

position in the pyramid

class index

feature pyramid

Inference $\quad f_{\mathbf{w}}(\mathbf{x}) = \underset{k,y}{\operatorname{argmax}} \; \mathbf{w}^\top \mathbf{\Phi}(\mathbf{x}, (k, y))$

Joint feature map: object detection with sparselets

$$\boldsymbol{\beta} = \left(\boldsymbol{\alpha}_1^\intercal, \dots, \boldsymbol{\alpha}_K^\intercal, \underline{\tilde{\mathbf{w}}_1^\intercal, \dots, \tilde{\mathbf{w}}_K^\intercal}\right)^\intercal$$

$$\tilde{\boldsymbol{\Phi}}\left(\underbrace{\mathbf{x}}_{\text{feature pyramid}}, (\underbrace{k}_{\text{class index}}, \underbrace{y}_{\text{position in the pyramid}})\right) = \left(0, \dots, 0, \underbrace{\left(\mathbf{c}_{y,1}^\intercal S, \dots, \mathbf{c}_{y,p_k}^\intercal S\right)}_{\text{projected feature window from position y installed in slot k}}{}^\intercal, 0, \dots, 0, \underbrace{0, \dots, 0, \mathbf{1}, 0, \dots, 0}_{\text{per-class bias}}\right)^\intercal$$

Inference

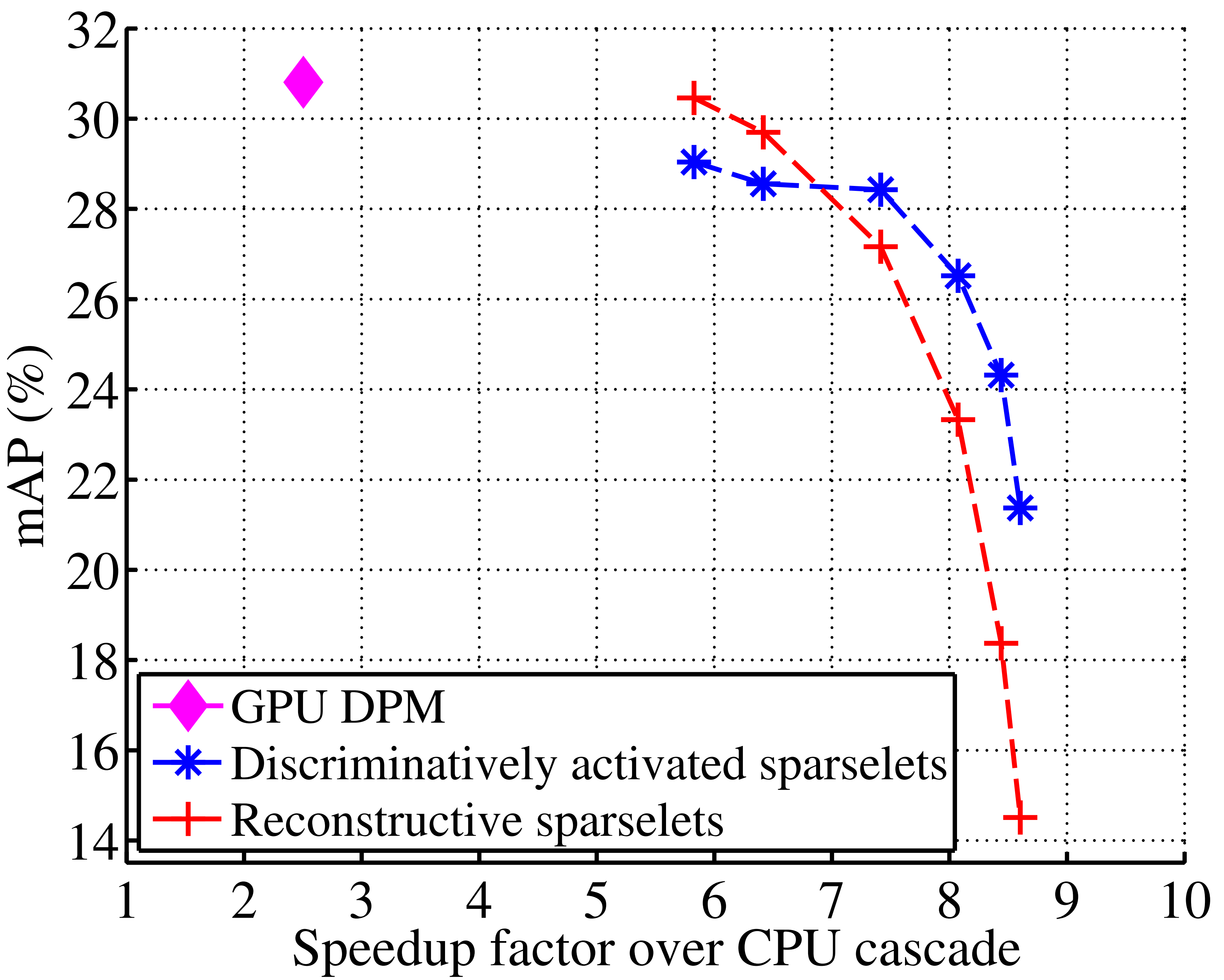
$$f_{\boldsymbol{\beta}}(\mathbf{x}) = \operatorname*{argmax}_{k,y} \; \boldsymbol{\beta}^\intercal \tilde{\boldsymbol{\Phi}}\left(\mathbf{x}, (k, y)\right)$$

Computational cost analysis

$$\text{Speedup} = \frac{\text{Original classifier cost}}{\text{Sparselet shared cost} + \text{Sparse reconstruction}} = \frac{Qm}{dm + Q\lambda_0}$$

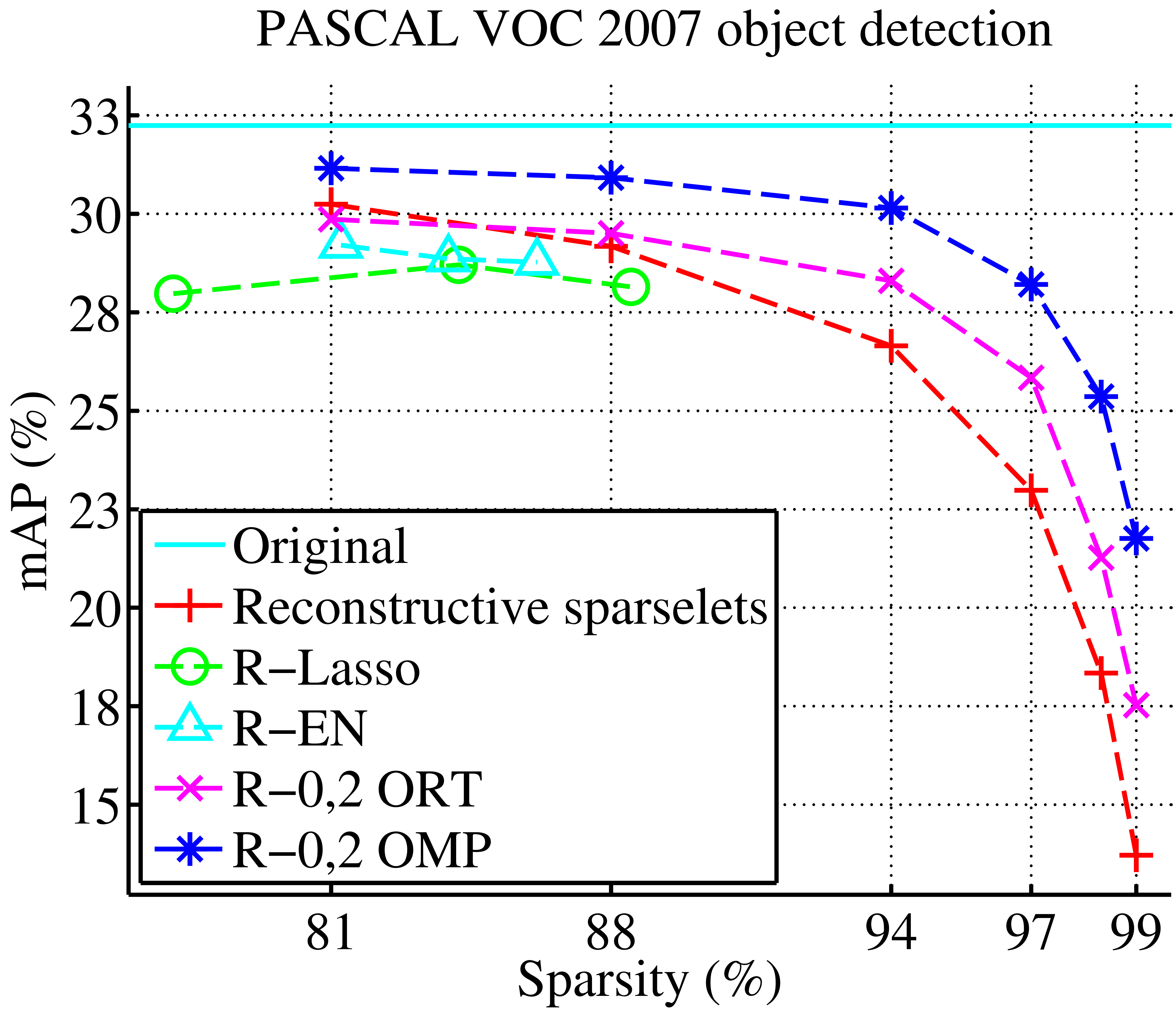
- To achieve speedup, number of sparselets should be small. $Q > d$
- Activation sparsity λ_0 dominates the speedup as Q grows.

Experiment 1 - Run Time

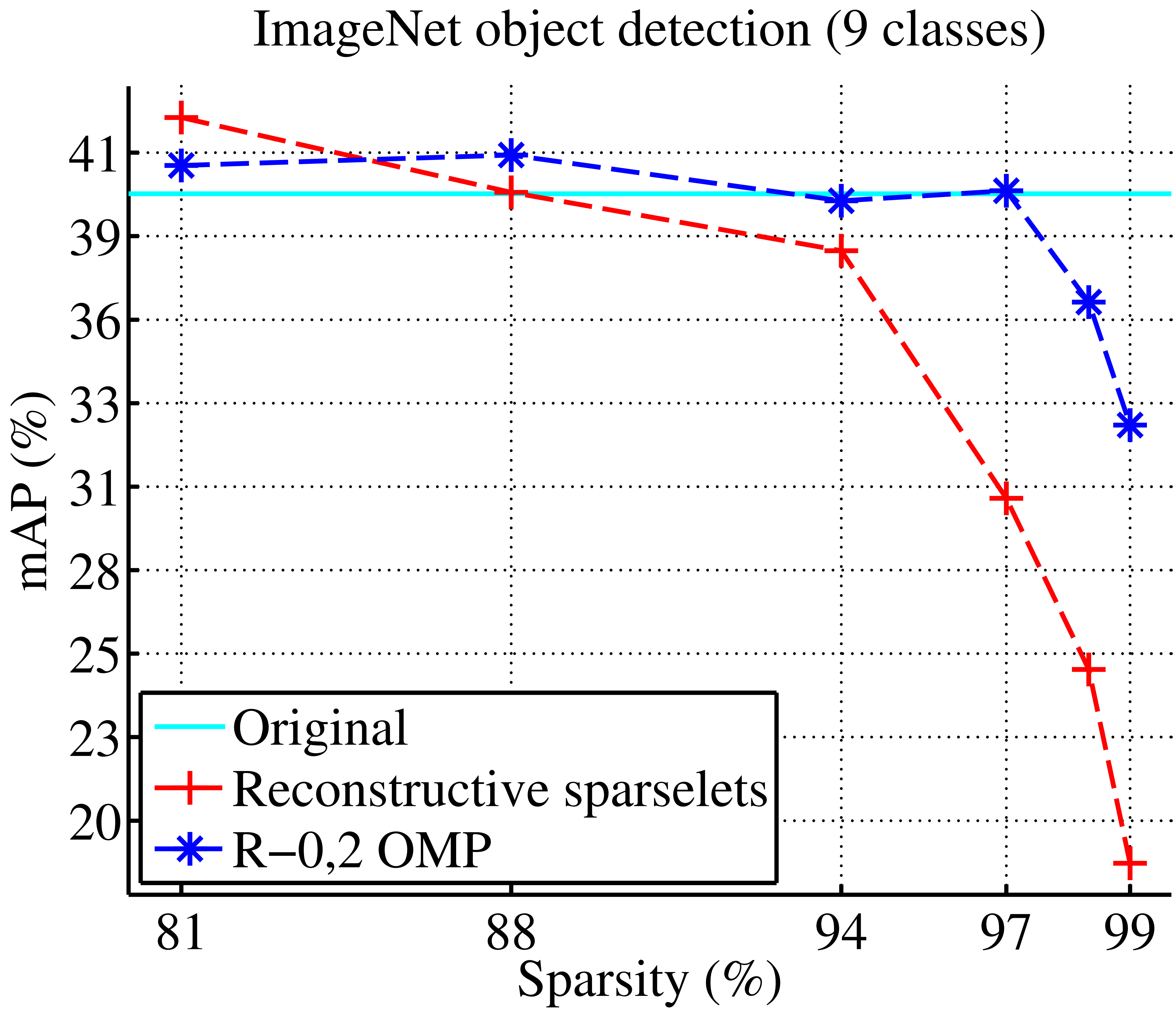


Run time comparison for DPM implementation on GPU, reconstructive sparselets and discriminatively activated sparselets in contrast to CPU cascade.

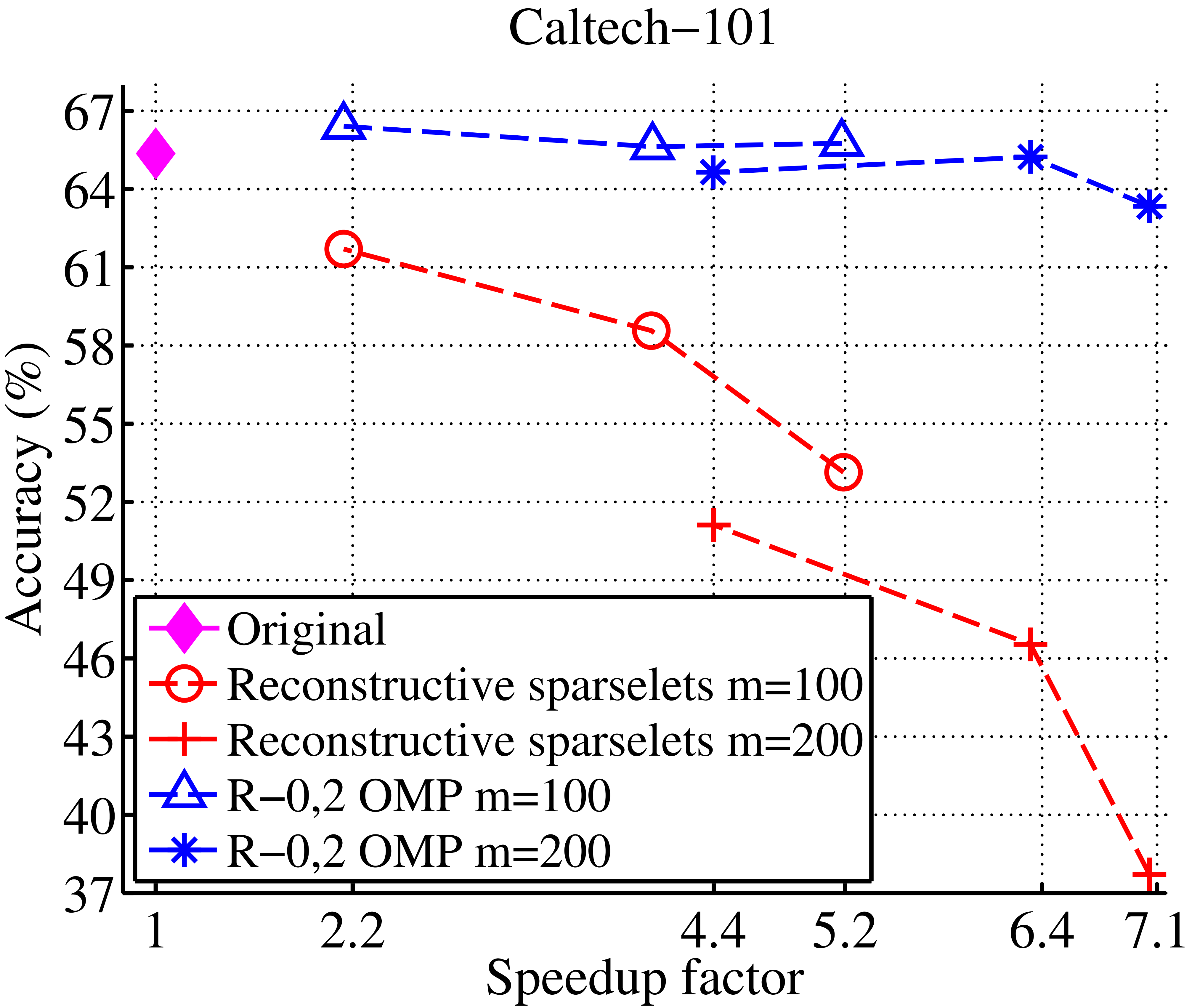
Experiment 2 - PASCAL detection



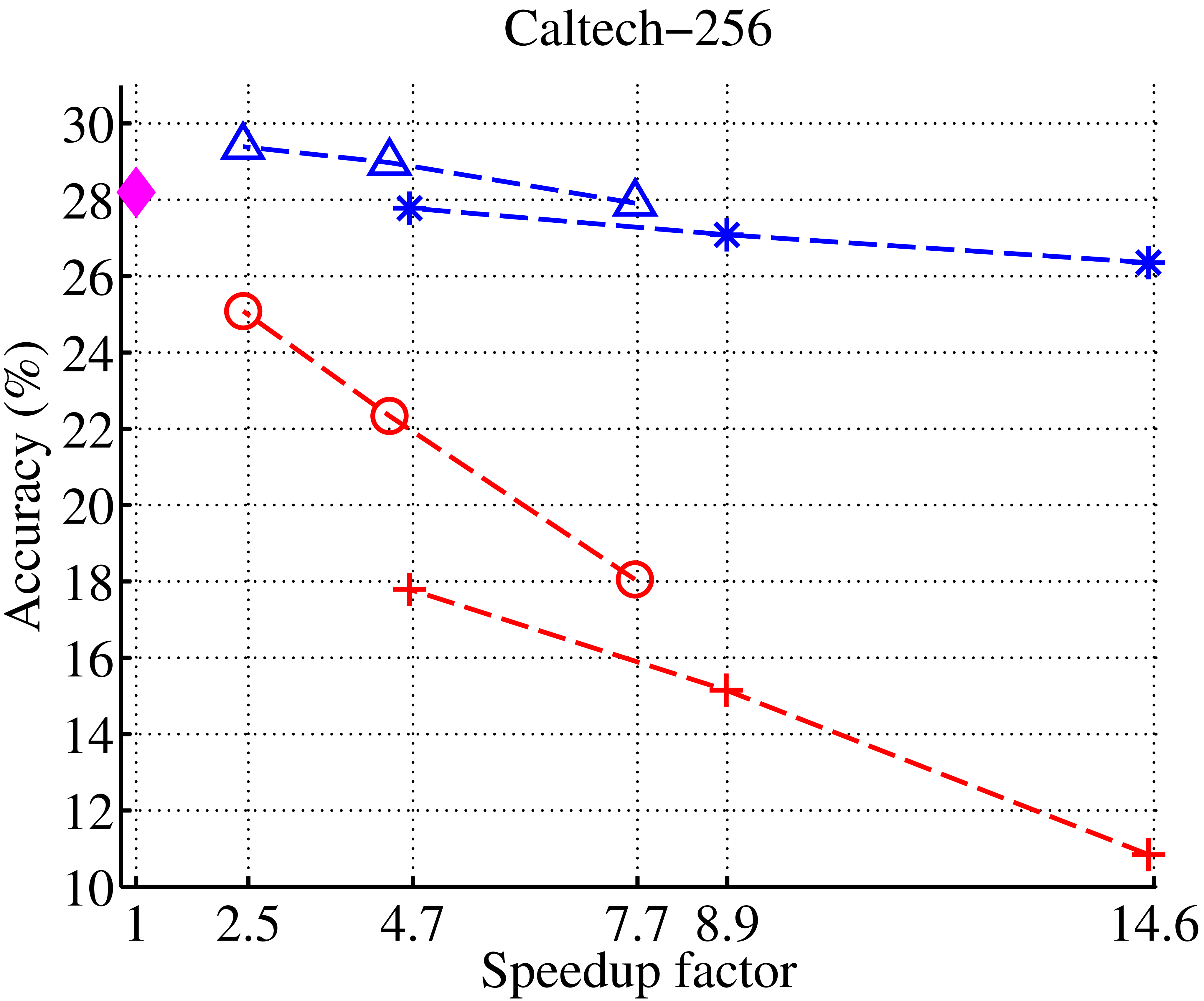
Experiment 3 - ImageNet detection



Experiment 4 - Caltech 101Classification



Experiment 5 - Caltech 256Classification

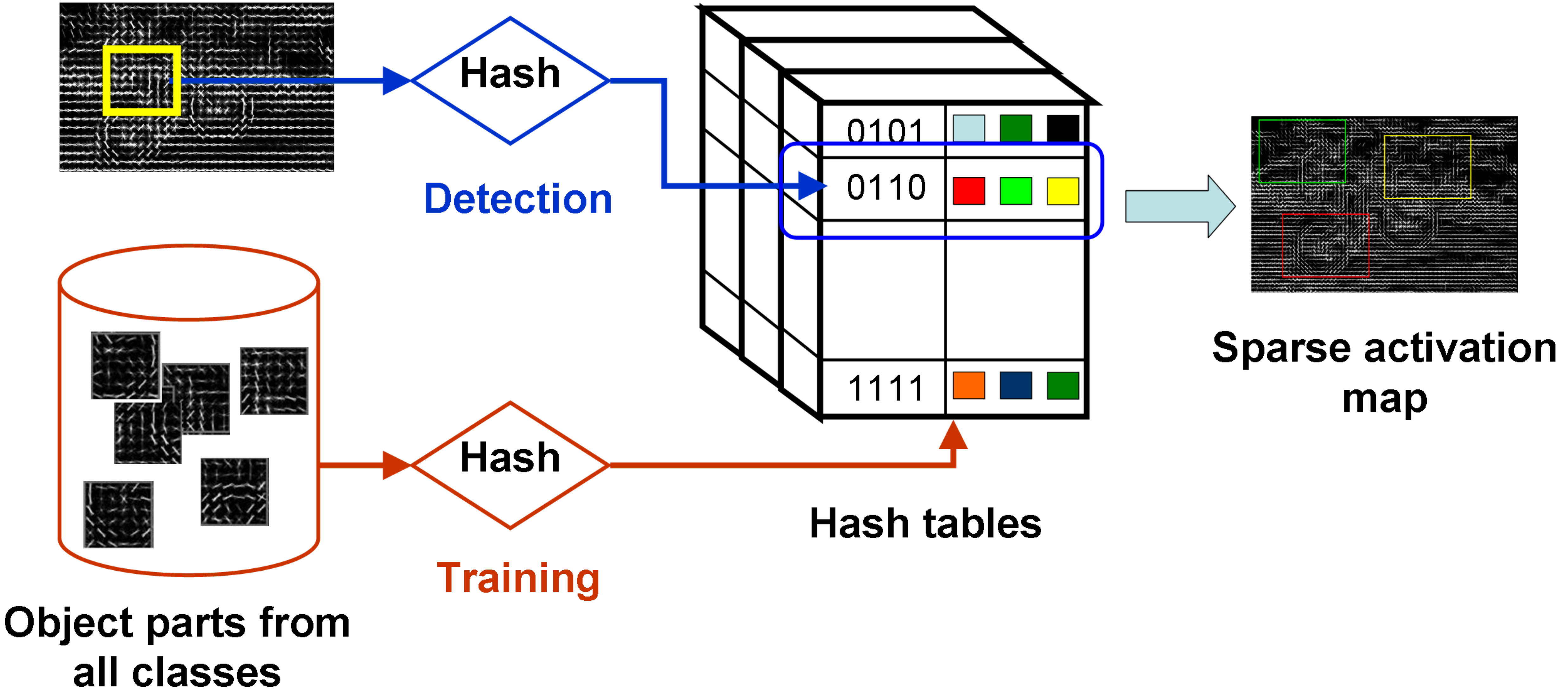


Discussion

Contents

- Sliding window object detection
- Deformable part models
- Cascade DPM
- Sparselets
- Hashing based

Hashing part filters



Hash

Detection

0101

0110

1111

Sparse activation map

Hash

Training

Hash tables

Object parts from all classes

Fast, Accurate Detection of 100k object classes on a single machine, Dean et al, *CVPR13*

Conclusion

- Surveyed sliding window object detection
- Various methods exist for speeding up the inference time (not training time)
- For fast training, LDA HOG (Hariharan, *ECCV12*) works well.