Learning an Alphabet of Shape and Appearance for Multi-Class Object Detection

Andreas Opelt, Axel Pinz and Andrew Zisserman

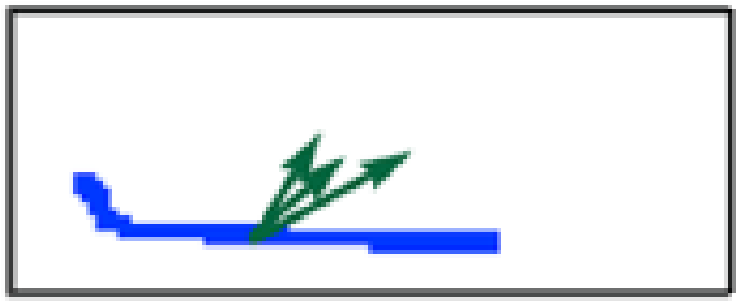
09-June-2009

Object class recognition

- Object class recognition is a key issue in computer vision.
- People use Shape and/or Appearance to categorize objects.
- In this paper they combine both shape and appearance.
- The alphabet is the basis for a codebook representation of object categories.
- The main focus of the paper is on representation and use of shape and geometry rather than appearance.

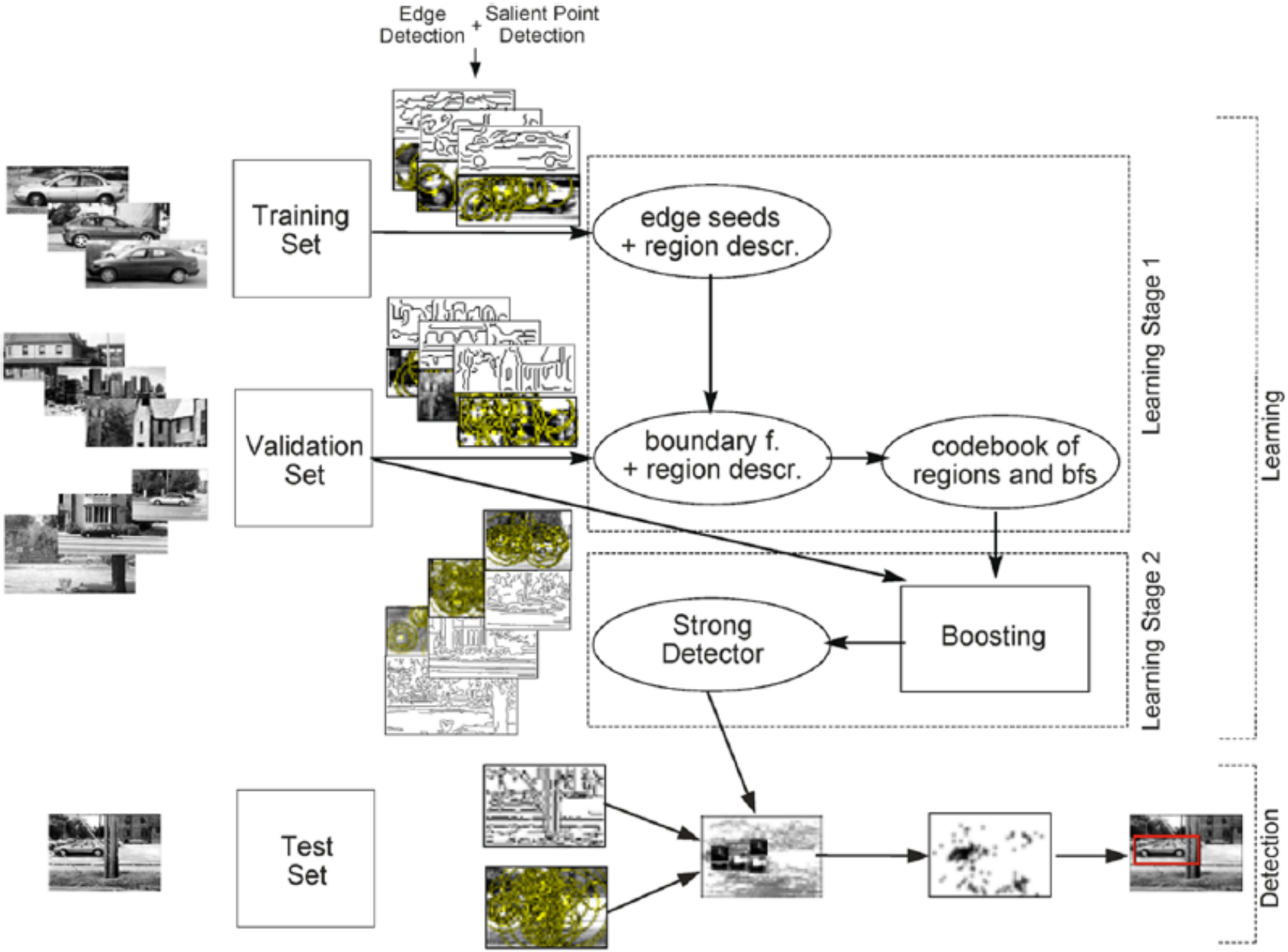
Object class recognition

Boundary-Fragment-Model(BFM)



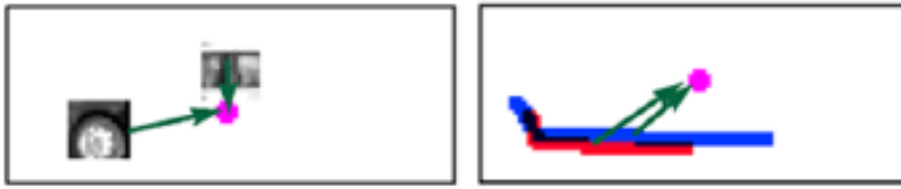
- A BFM is restricted to a codebook of boundary fragments and does not represent appearance at all.
- The boundary represents the shape of many object classes quite naturally without requiring the appearance (e.g. texture) to be learnt and thus we can learn models using less training data to achieve good generalization.

System Overview



System Overview

(a)



(b)

Figure: (a) Two alphabet entries (one region, one Boundary-Fragment). (b) Two weak detectors (one region-based, one Boundary-Fragment based).

Training Data

To train the model following data are required:

- A training image set with the object delineated by a bounding box.
- A validation image set with counter examples (the object is not present in these images), and further examples with the object's centroid (but the bounding box is not necessary).

Learning

Learning is performed in two stages.

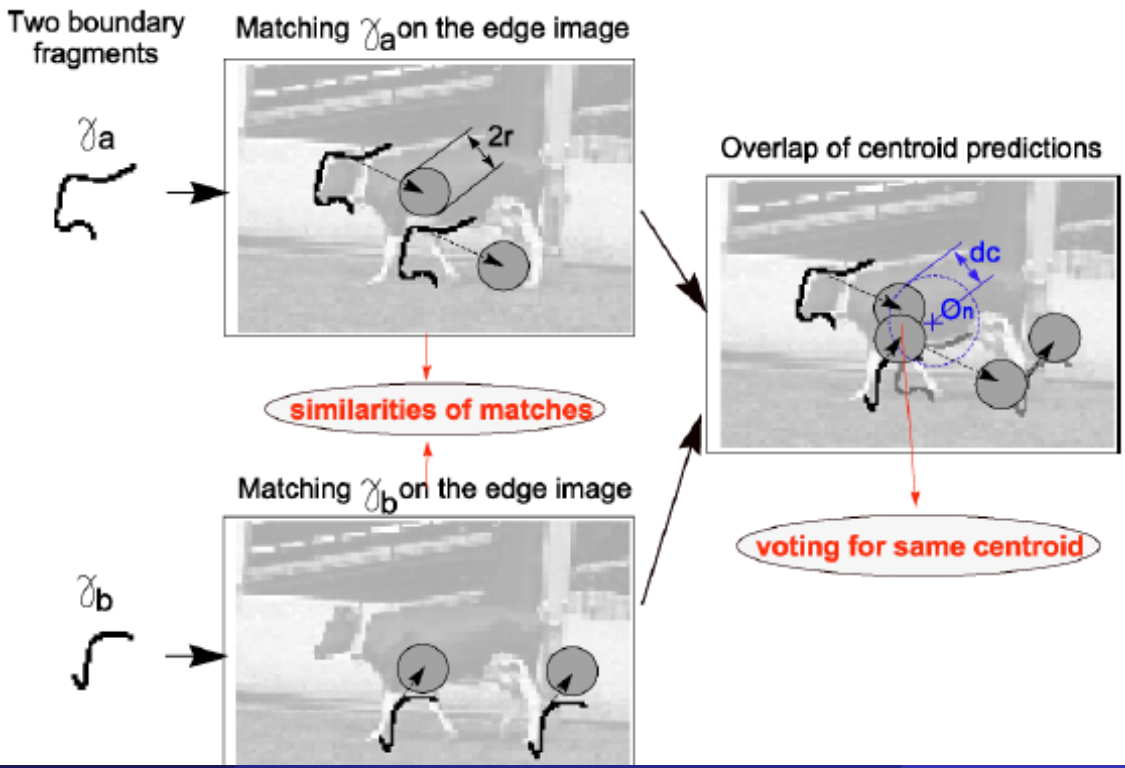
- Alphabet entries are added to a codebook.
  - An alphabet entry can either be a Boundary-Fragment (BF-a piece of linked edges), or a patch (salient region and its descriptor).
  - Each entry also casts at least one centroid vote, which is represented as a vector.
- Weak detectors are formed as pairs of two alphabet entries, and Boosting is used to select a strong detector.
  - A strong detector consists of many weak detectors.
  - This process selects the weak detectors which perform best on positive validation images and rejects the negative images (including a good centroid estimate).

Implementation Details

- Linked edges are obtained for each image in the training and in the validation set using a Canny edge detector.
- Training images provide the candidate boundary fragments γ_i by selecting random starting points on the edge map of each image.
- Then at each such point they grow a boundary fragment along the contour.
- Growing is performed from a certain fragment starting length L_{start} in steps of L_{step} pixels until a maximum length L_{stop} is reached.

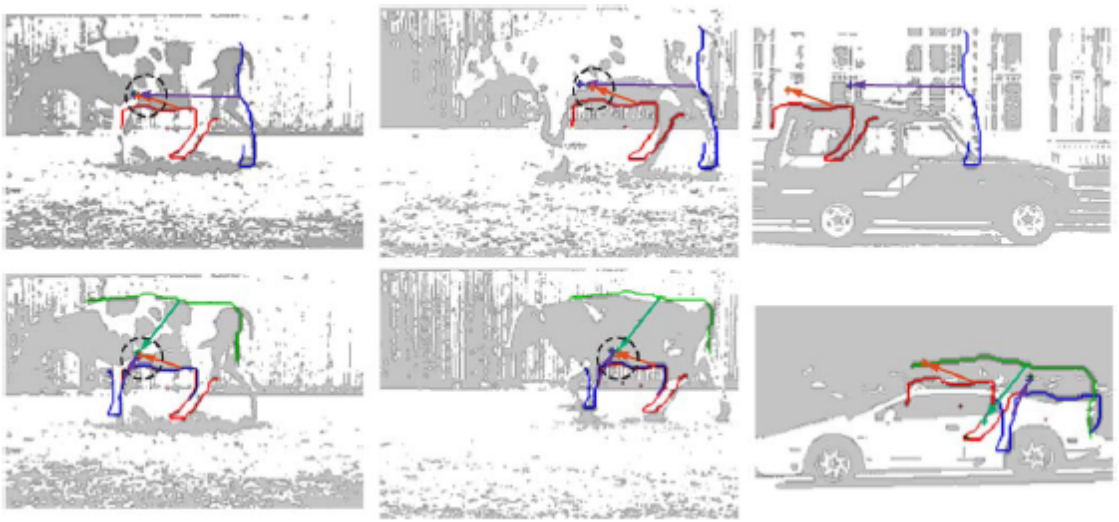
Weak detector

- The combination of boundary fragments to form a weak detector h_i. It fires on an image if the k boundary fragments (γ_a and γ_b) match image edge chains, the fragments agree in their centroid estimates (within an uncertainty of $2r$). In the case of positive images, the centroid estimate agrees with the true object centroid (O_n) within a distance of d_c



Matching weak detectors

- The top row shows a weak detector with $k = 2$, that fires on two positive validation image because of highly compact center votes close enough to the true object center (black circle). In the last column a negative validation image is shown. There the same weak detector does not fire (votings do not concur). Bottom row: the same as the top with $k = 3$.



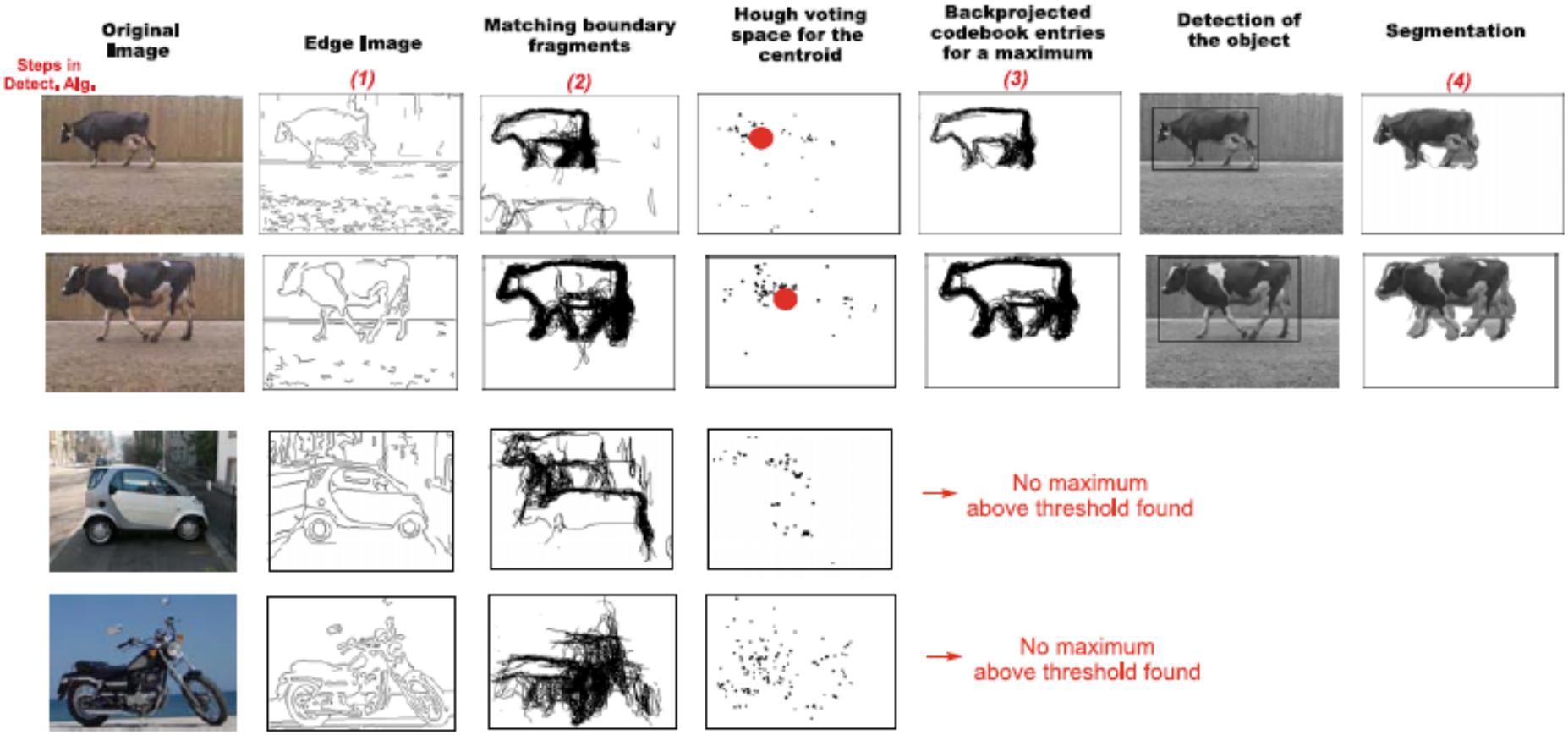
Learning a Strong Detector

- From a weak detector consisting of k boundary fragments and a threshold th_{hi} they learn this threshold and form a strong detector H out of T weak detectors h_i using AdaBoost.
- First they calculate the distances $D(h_i, I_j)$ of all combinations of boundary fragments (using k elements for one combination) on all (positive and negative) images of validation set $I_1, ..., I_v$.
- Then in each iteration $1, ..., T$ they search for the weak detector that obtains the best detection result on the current image weighting.

Detection and Segmentation

- First the edges are detected.
- The boundary fragments of the weak detectors are matched to this edge image.
- In order to detect (one or more) instances of the object (instead of classifying the whole image) each weak detector h_i votes with a weight w_{hi} in a Hough voting space.
- Votes are then accumulated as follows:
  - For all candidate points x_n found by the strong detector in the test image I_T they sum up the (probabilistic) voting of the weak detectors h_i in a $2D$ Hough voting space.

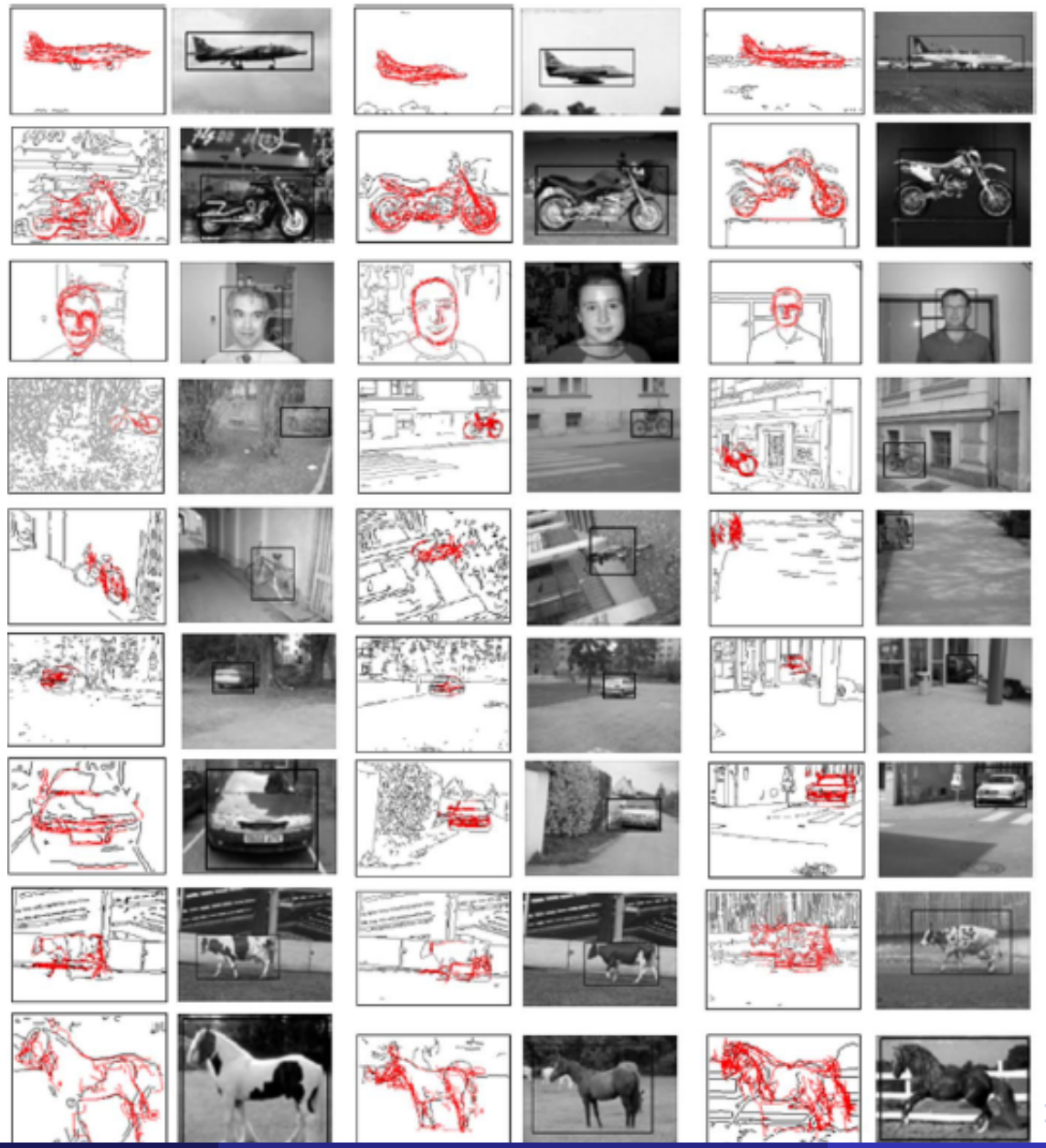
Detection and Segmentation



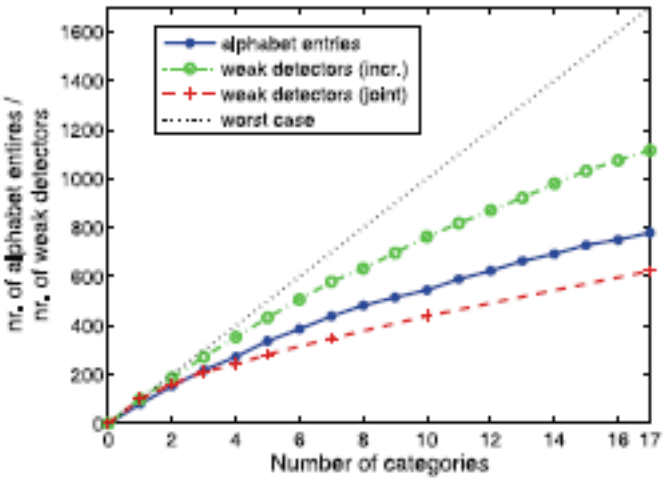
The BFM for Multiple Categories

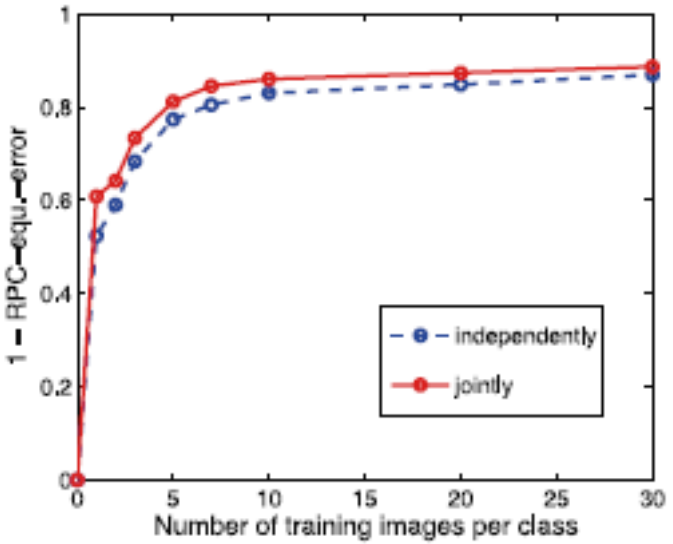
- Building the alphabet of shape for many categories is based on the process for the one-class BFM.
- They also search over other categories to see if a boundary fragment can be shared.
  - The boundary fragment matches on many positive validation images of another category and gives a roughly correct prediction of the object centroid. In this case they just update the alphabet entry with the new costs for this category and sharing is possible.
  - The boundary fragment matches well on many positive validation images, but the prediction of the object centroid is not correct, though often the predictions for each match are consistent with each other. In this case they add a new centroid vector to the alphabet entry.
  - The third obvious case is where the boundary fragment matches arbitrarily in validation images of a category in which case high costs emerge and sharing is not possible

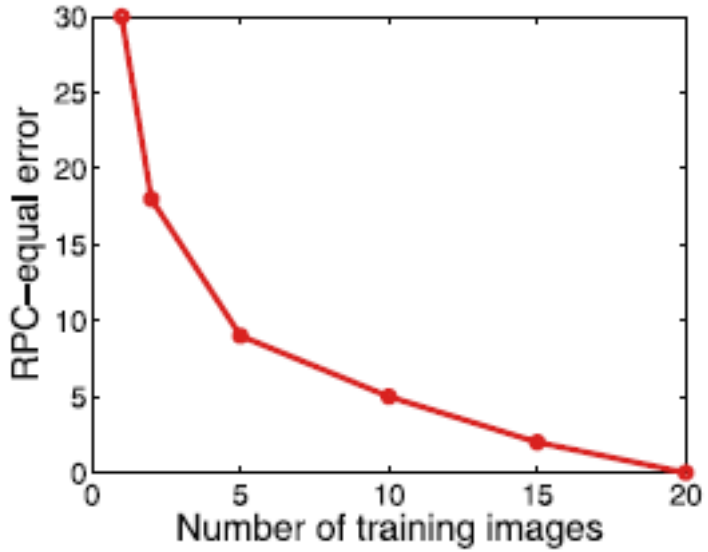
Results



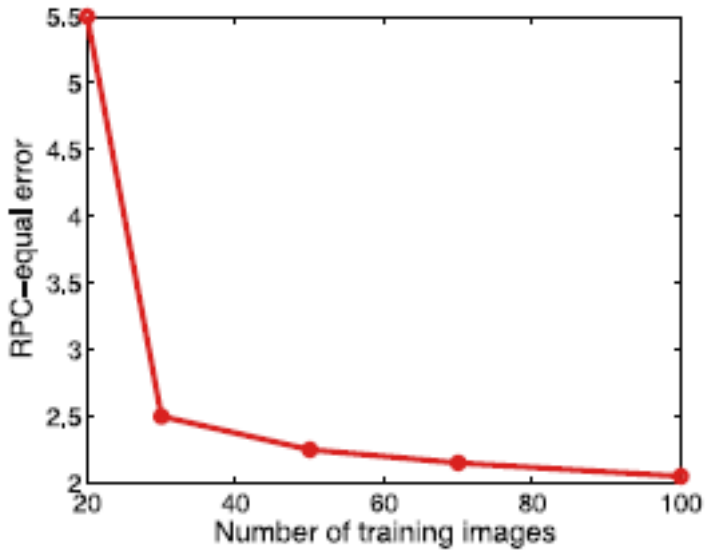
Results







Cow Data Set



Cars-Rear Data Set

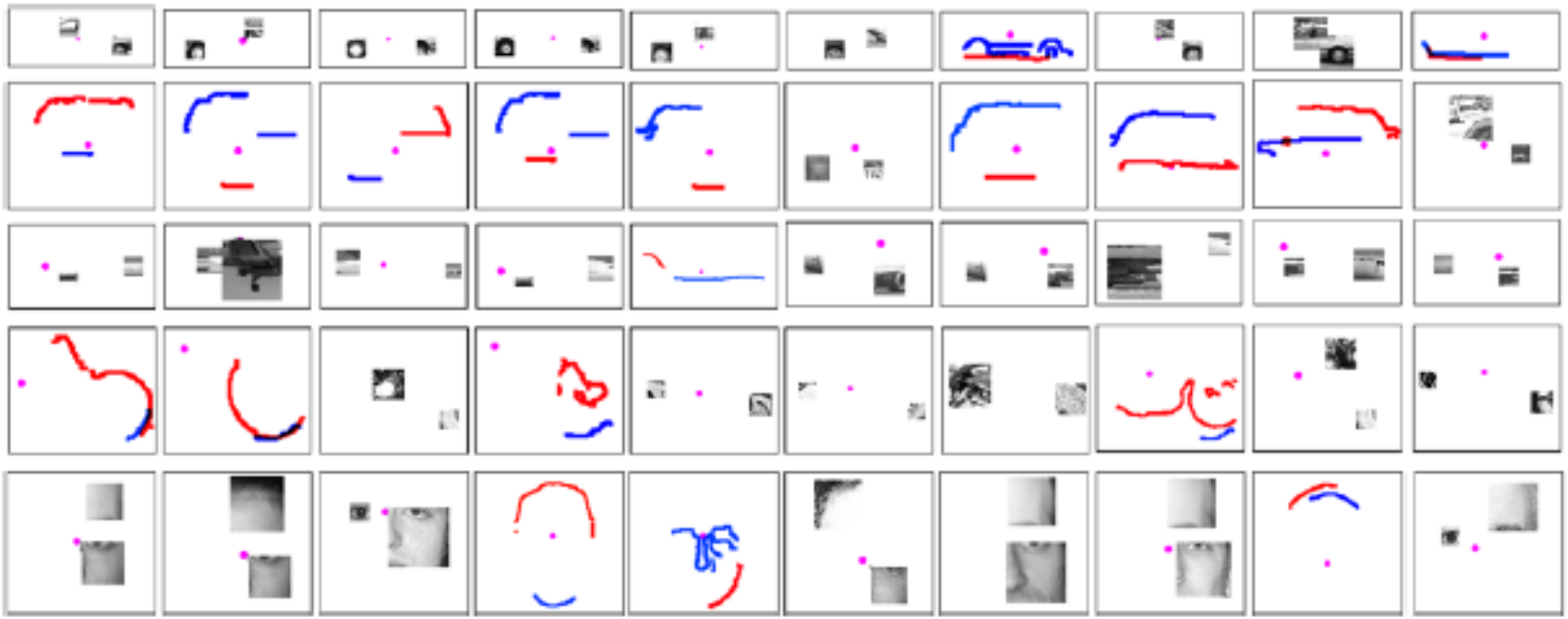
Results

Table 4 Confusing cows and horses: The first 3 rows show the failures made by the three different models (FP = false positive, FN = false negative, M = multiple detection). The last row shows the RPC-equal-error rate for each model

	Cow-BFM	Horse1-BFM	Horse2-BFM
FP	0	3	0
FN	0	13	12
M	0	1	2
RPC-eq.	0%	23%	19%

Results

- The first ten weak detectors learnt in the UM for the categories: Cars-side (UIUC), Cars-rear, Airplanes, Motorbikes and Faces (Caltech).



Conclusion and Discussion

- Less False positives.
- Less Training data.
- Processing Time??
- Scaling and Rotation.