A Hierarchical Part-Based Model for Visual Object Categorization

Guillaume Bouchard and Bill Triggs

LEAR, GRAVIR-INRIA-CNRS, Grenoble, France

http://lear.inrialpes.fr/people/{bouchard,triggs}

Work supported by European Union research project LAVA

Local Part Based Category Recognition

Extract ***distinctive local features***, link them by ***loose geometric constraints***

• Local features $\Rightarrow$ robustness to occlusions / appearance / variations outside feature support.

• Loose geometry $\Rightarrow$ flexible shape to allow for within-class variation, changes of viewpoint,

Families of Approaches

- ***Bag of features:*** no geometry, just vote using feature appearances.
- ***Feature based matching:*** rigid feature sets under similarity, affine, *etc*., transformations.
- ***Network / constellation:*** multiply interconnected networks over a few salient features/parts.
- ***Multiscale trees:*** coarse to fine networks with regular branching at each scale, dense image features (pixels, filters)
- ***Our model:*** many features (100's); tree structure linked to coherent object parts not scale; rapid training (100's of images per minute in Matlab).

Hierarchical Spatial Model

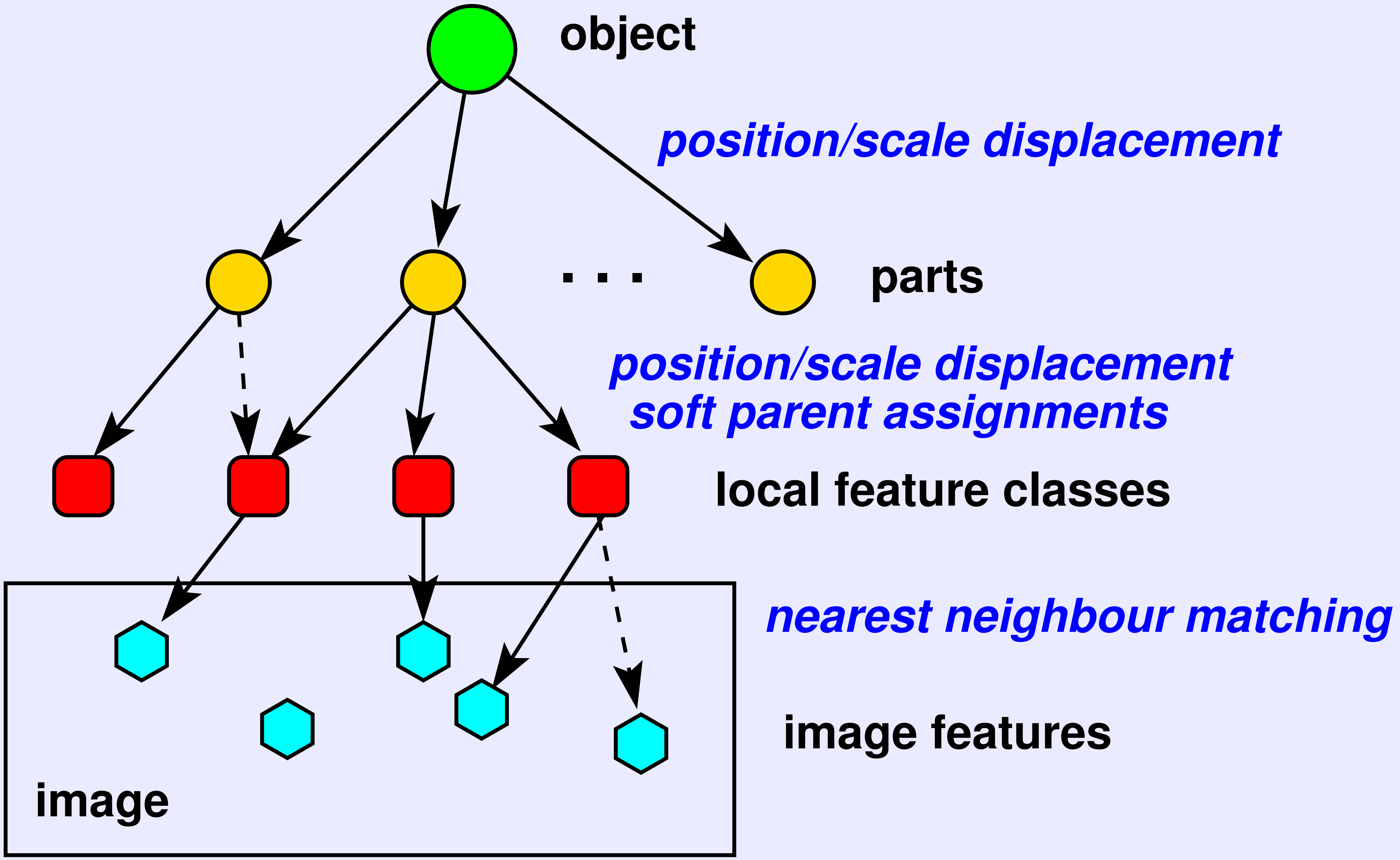
- Object / part / sub-part hierarchy, leaves are local image features

  — below we use just 3 layers: object / part / feature

- Each child has an uncertain relative position & scaling relative to its parent — in general a PDF over some class of geometric transformations

- Soft assignment of children to parents allows re-adoption during training, but most children have just one dominant parent

Graphical Structure



Flexibility under Viewpoint Changes

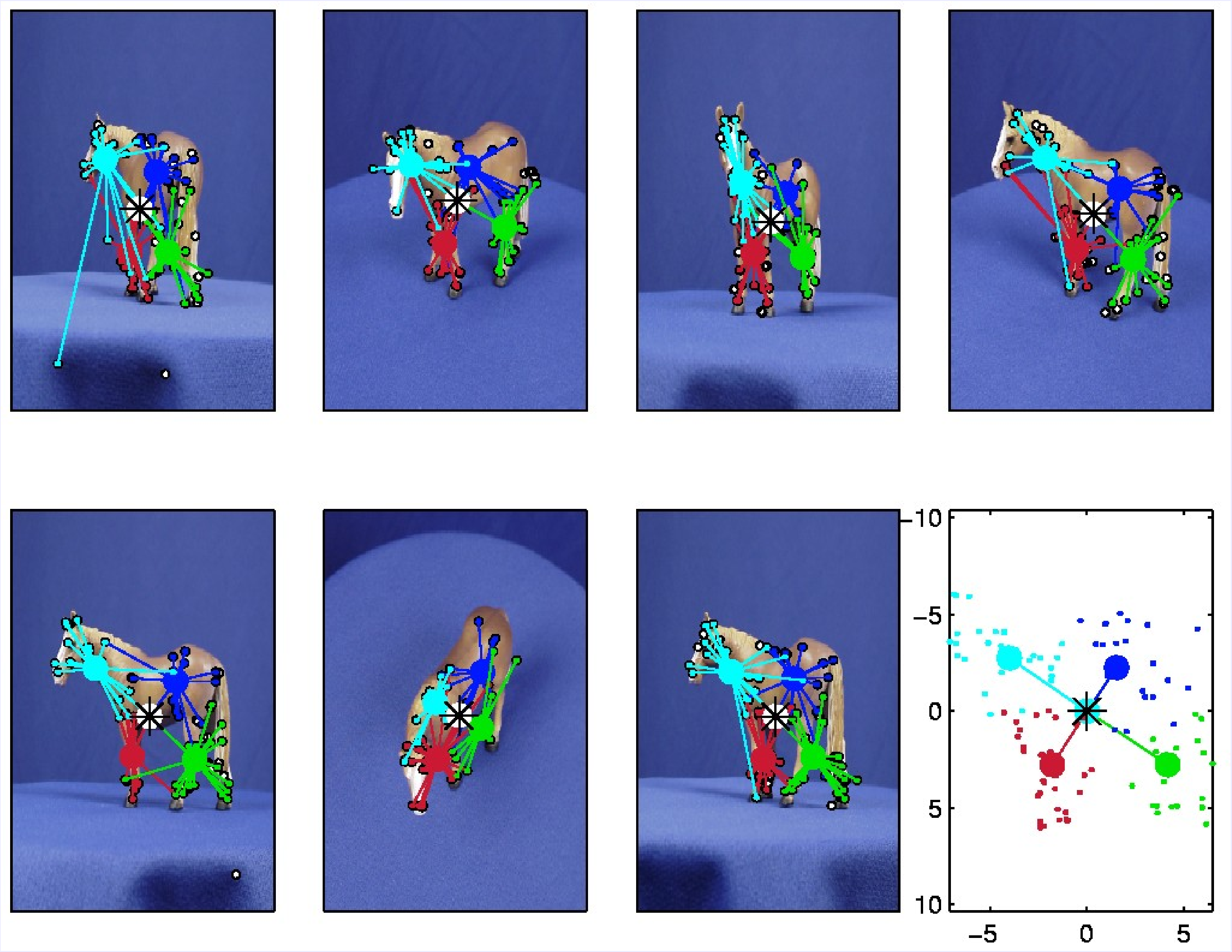


Image Features

- Correspondence is based on ***local invariant features***

— in the below experiments, SIFT over scale invariant Harris points

- Our image measurements are *tests for the presence of a local feature matching the given appearance class in the given image region*
- For each feature class k, we select the *single most probable image feature*, and use just these assigned features in the subsequent processing.
- A feature may be chosen by several classes (rare owing to locality).
- Robustness — poor matches are essentially ignored.

Density Model for Feature Class k

The probability density model for **features of class** k is:

$$p_k(\boldsymbol{x}, \boldsymbol{a} \,|\, \ldots) \;=\; (1-\pi_k)\, p_{\text{background}} + \pi_k\, \mathcal{N}(\boldsymbol{a}|\bar{\boldsymbol{a}}_k, \boldsymbol{\Sigma}_{\boldsymbol{a}_k}) \sum_{\text{parts } p} \tau_{pk}\, \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}_p + \bar{\mathbf{dx}}_{pk}, \boldsymbol{\Sigma}_{\mathbf{dx}_{pk}})$$

- $\boldsymbol{a}, \boldsymbol{x} \;=\;$ appearance, position/scale of feature
- $\pi_k \;=\;$ probability of feature being from object not background
- Each model part p has:

— a distribution for the feature's position/scale relative to the part centre $\boldsymbol{x}_p$

— a (sparse) prior probability $\tau_{kp} = p(p|k)$ for the feature to belong to the part.

Fitting a Model Instance to an Image

- Given the instantiated model and the above image correspondence method, fitting is standard Expectation-Maximization.
- In test images, we adjust just the image instance parameters (part positions,...)
- During training, we adjust both instance and model parameters

Instantiating a Model Instance

Instantiation uses a Hough-like voting method. (Some of the experiments use an earlier image-alignment based method).

$\boxed{1}$ For each part p, each image feature f votes into a position/scale pyramid for p's centre $\boldsymbol{x}_p$, using f's appearance probability w.r.t. p:

$$\boxed{\sum_{\text{feature class } k} \frac{\tau_{pk}}{w_k} \mathcal{N}(\boldsymbol{a}_f | \bar{\boldsymbol{a}}_k, \boldsymbol{\Sigma}_{\boldsymbol{a}_k}) \, \mathcal{N}(\boldsymbol{x}_f | \boldsymbol{x}_p + \bar{\boldsymbol{dx}}_{pk}, \boldsymbol{\Sigma}_{\boldsymbol{dx}_{pk}})}$$

- To suppress common background features and enhance rarer object ones, we divide the vote by the total number of features assigned to class k: $w_k = \sum_f \mathcal{N}(\boldsymbol{a}_f | \bar{\boldsymbol{a}}_k, \boldsymbol{\Sigma}_{\boldsymbol{a}_k})$.
- For speed, we actually use hard assignments $f \to k$ and $k \to p$.

$\boxed{2}$ Work up spatial tree combining part pyramids into superpart ones:

$$\boxed{\mathrm{smooth}\Big(\sum_{\text{subparts } p} f(\mathrm{spatial\_offset}_p(\mathrm{pyramid}_p))\Big)}$$

— $f(x) = \log(1 + x)$ makes it harder for high peaks in outlier subparts to dominate the valid contributions of the other parts.

$\boxed{3}$ Maxima in the top-level pyramid give potential object centres.

$\boxed{4}$ Work back down the tree assigning part positions. If there is a good part pyramid maximum near the expected part position, use it. Otherwise use the default part offset.

Training — Model Initialization

We heuristically rank the training images according to their expected quality as training examples (see below), and use just the best image to estimate the initial model parameters.

$\boxed{1}$ For each feature in the initial image, initialize a feature class k centred at its position and appearance.

$\boxed{2}$ Cluster the features into P spatial groups (initial "parts", but some will be background). Cluster centres $\longrightarrow$ part centres; median feature scale $\longrightarrow$ part scale; relative feature positions/scales $\longrightarrow$ feature offsets; feature-part assignment $\longrightarrow$ τ matrix.

$\boxed{3}$ Propagate the centres up tree by averaging subpart positions/scales (one vote per subpart).

Notes on Model Initialization

- Step 1 could be improved: some classes will be background junk and we will miss some informative features from other images
- Some feature classes may have the same appearance: this allows for (small numbers of) repeated features (wheels, eyes...)
- We could also try initializing from 2nd, 3rd, ...image, but so far this hasn't been necessary.
- Trying to initialize from averages of several images gives much worse results.

Ranking the Training Images

$\boxed{1}$ Use K-means to cluster the features from all (positive) training images into ~ 500 classes

— each image has a 500-D signature $\boldsymbol{S}$ (vector of class counts)

$\boxed{2}$ Select the ~ 30 "most informative" feature classes, and rank images by the number of these classes that they contain ($\boldsymbol{S}_c \neq 0$)

Feature Selection for Ranking

Supervised method

Train a logistic discriminant (RVM, LASSO, linear SVM. . .) to predict the $\pm$ class from the binarized signature vector $(\boldsymbol{S} \neq 0)$. Choose the features with the highest weights.

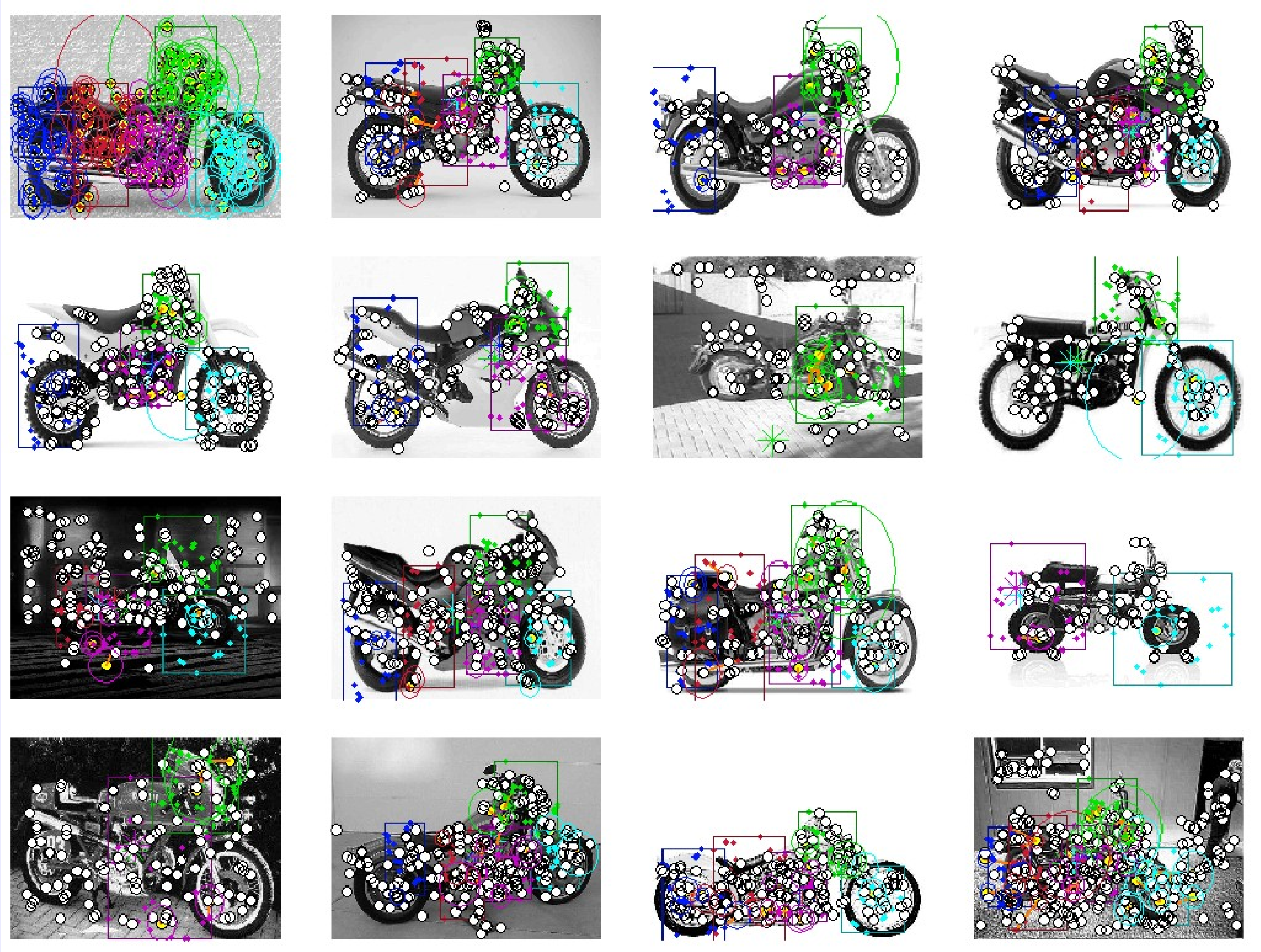
- This works fine but requires negative images. . .

Unsupervised Method

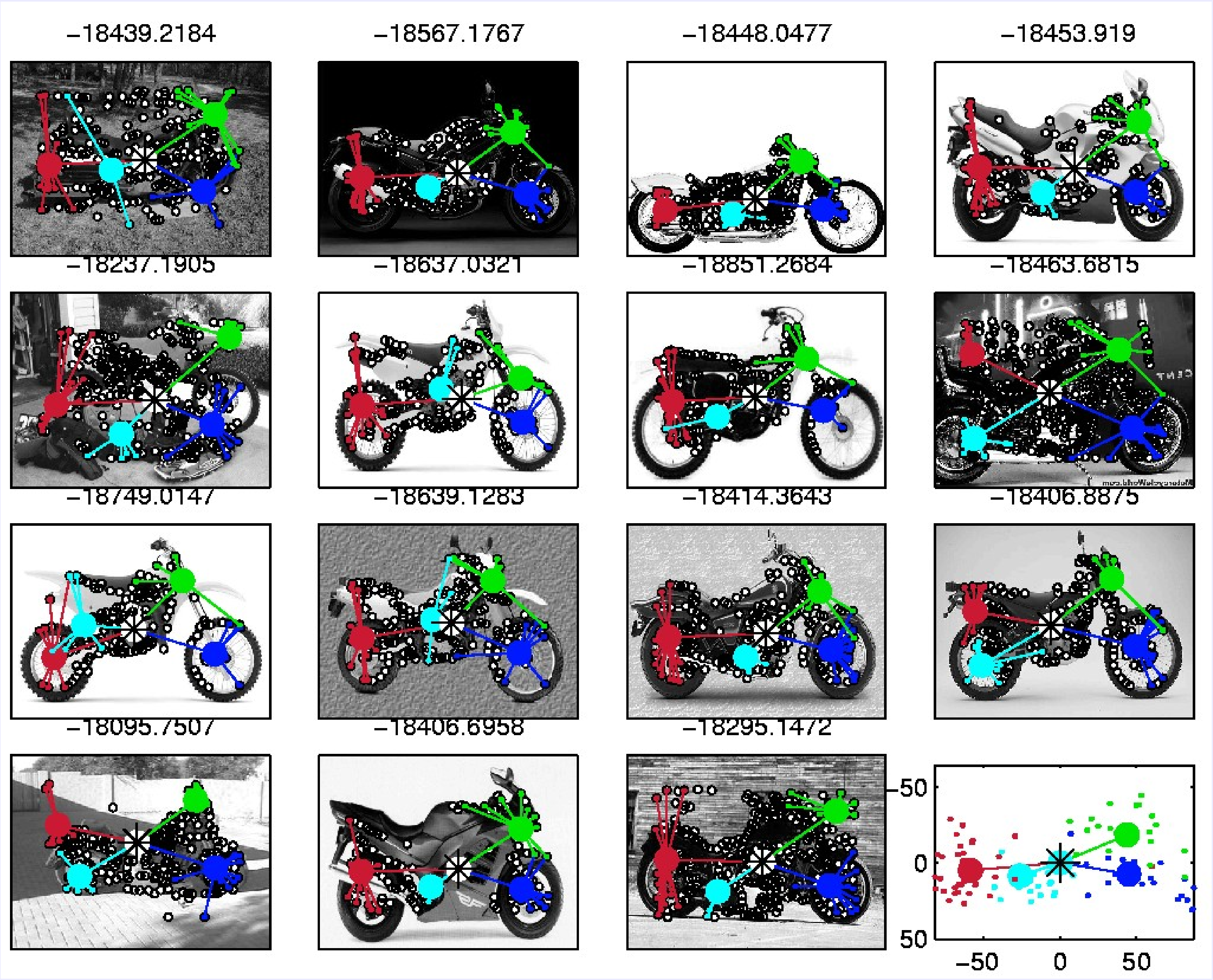
For each feature, count the number of images in which it occurs *exactly once* (alternatively, 1–2 times). Choose the ~ 30 features with the highest counts.

- This selects *distinctive features representing unique object parts* (background features seldom occur exactly once per image). It fails for objects dominated by repetitive texture.

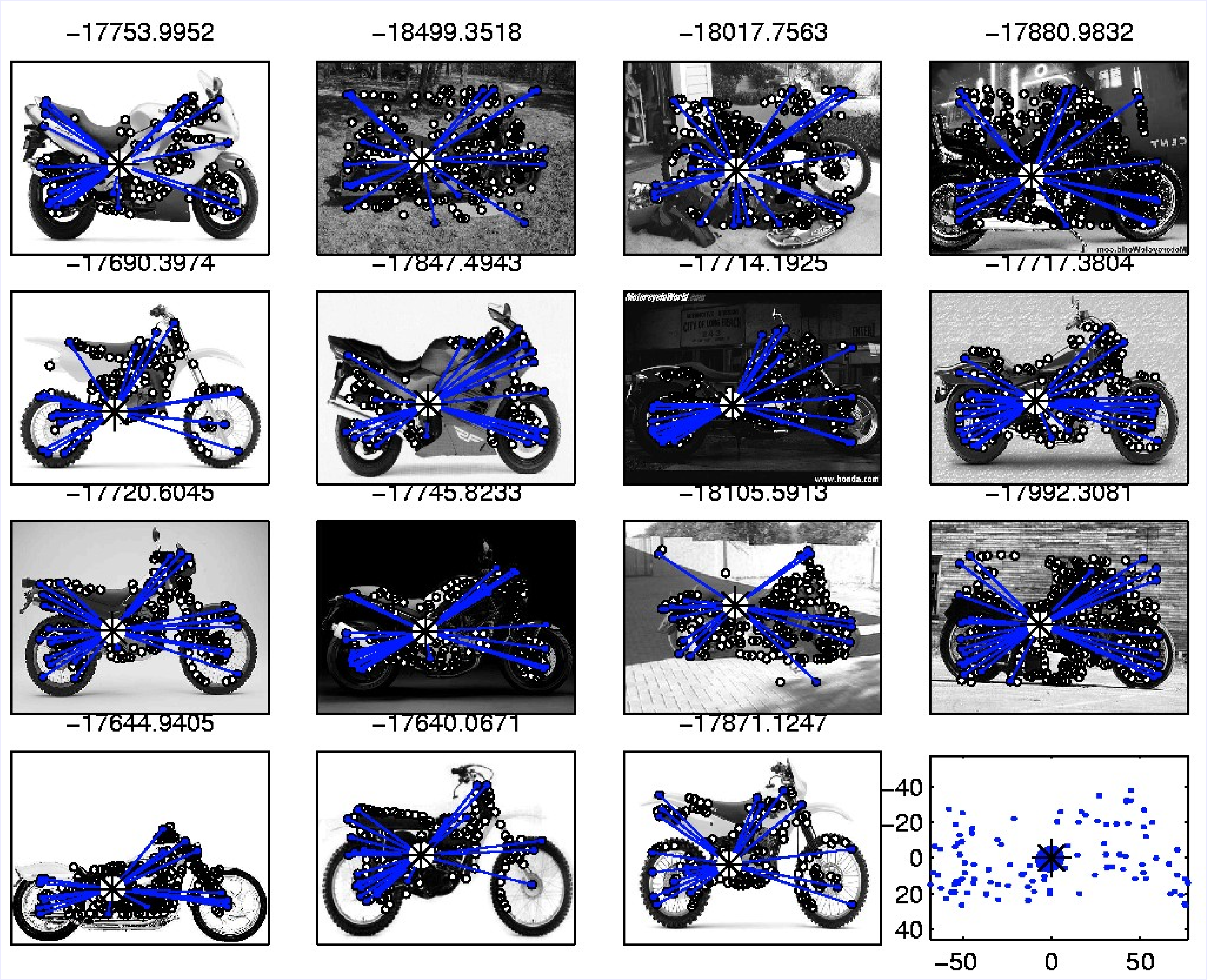
Training Initialization — Motorbikes



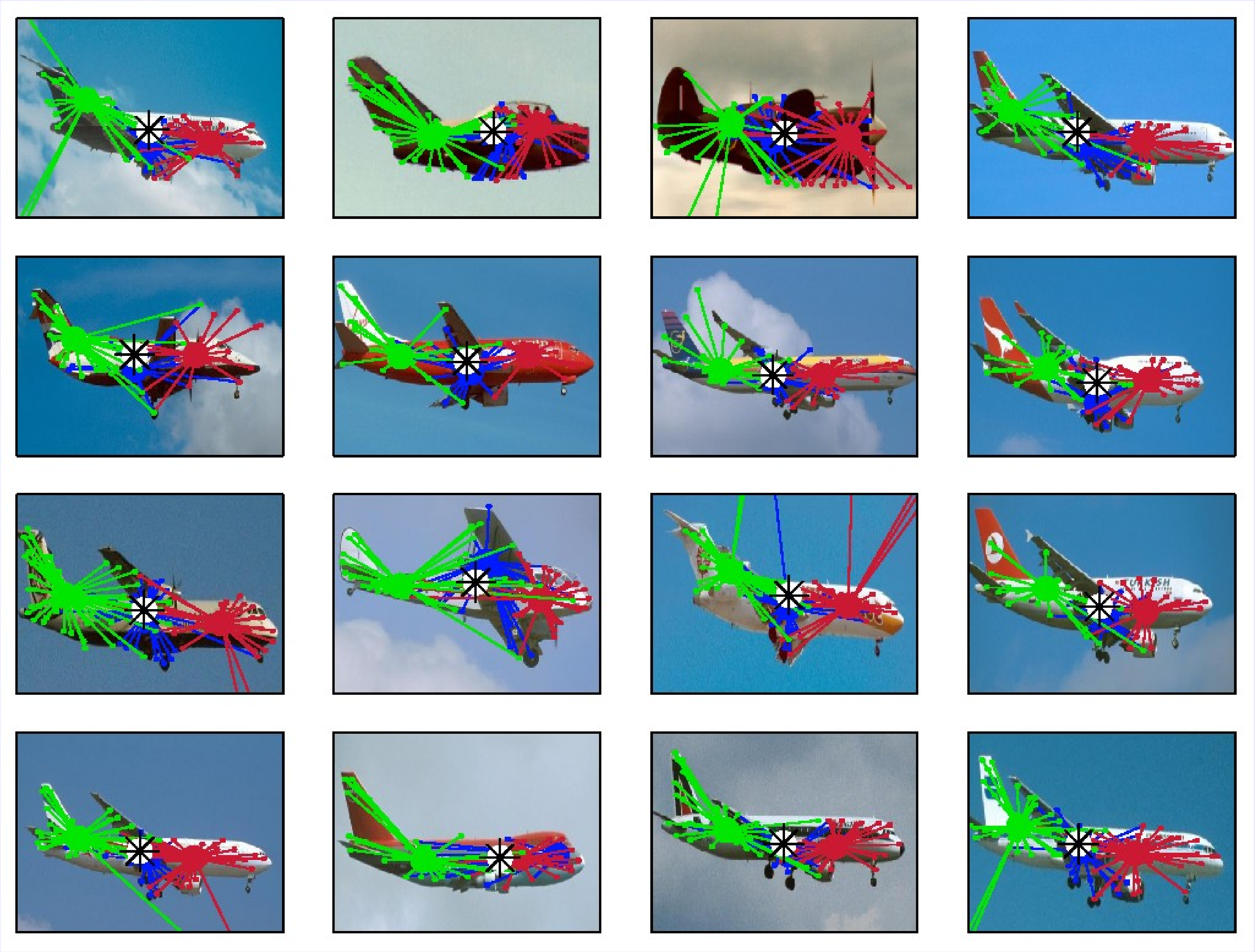
Test Set Fits — Motorbikes / 4 Parts



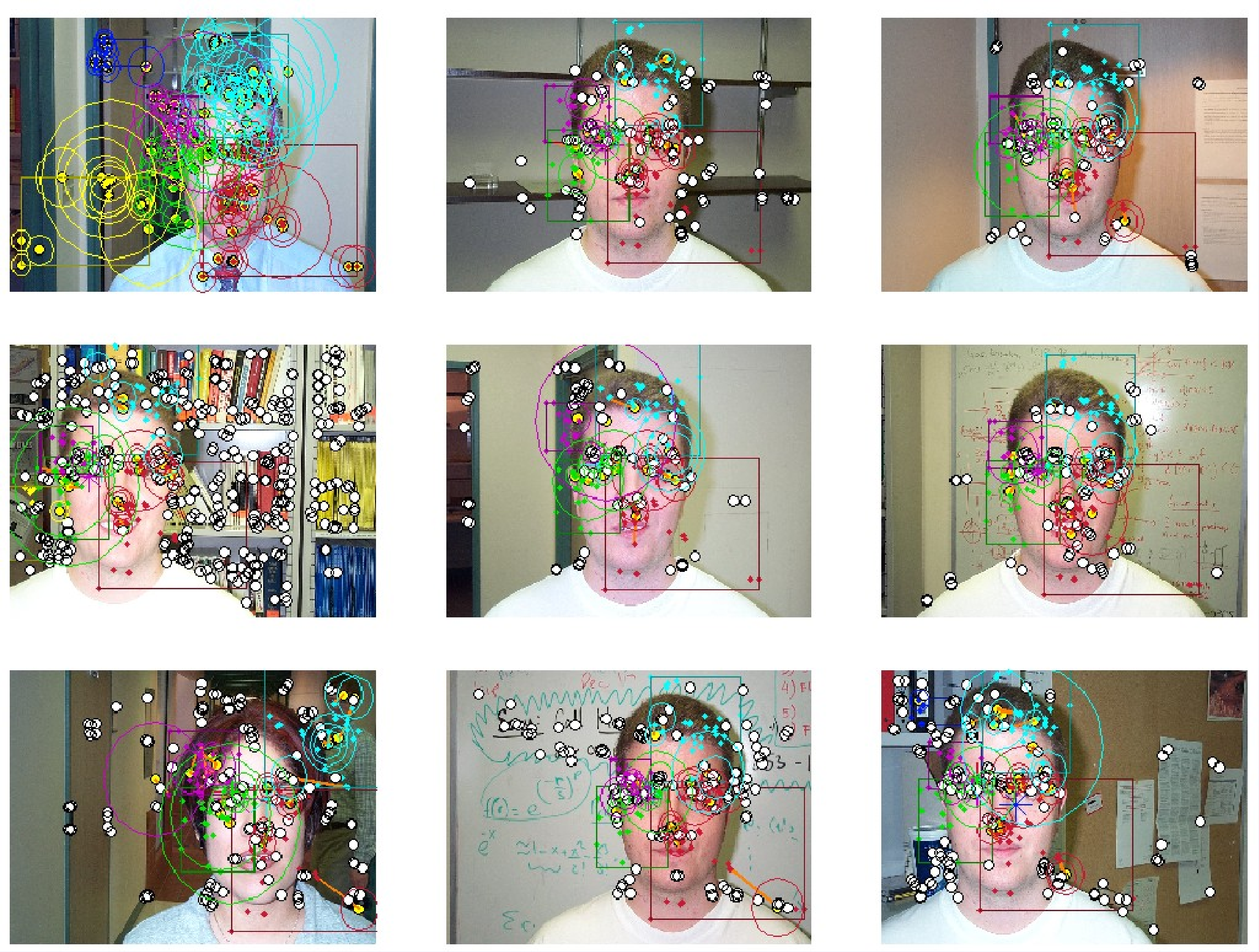
Test Set Fits — Motorbikes / 1 Part



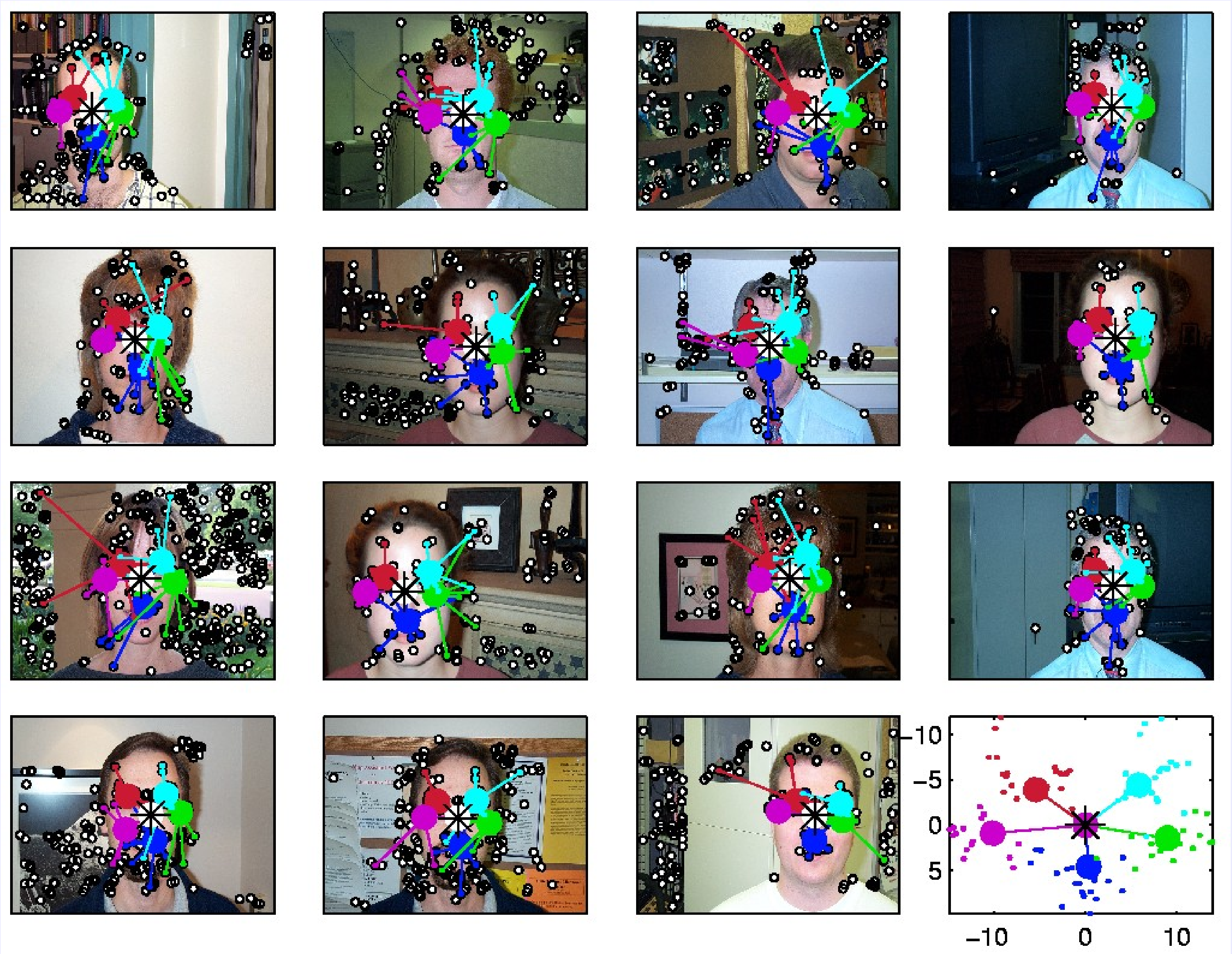
Test Set Fits — Aeroplanes / 4 Parts



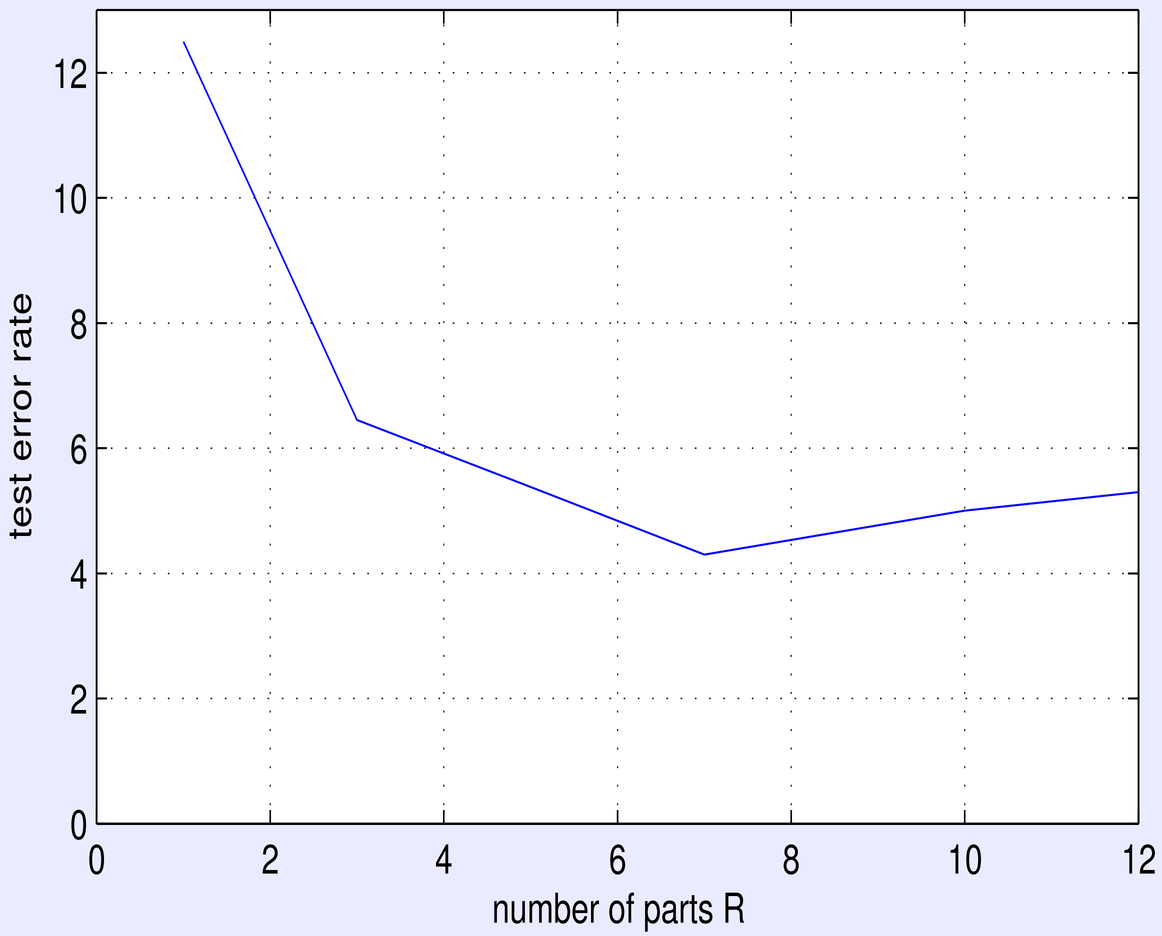
Training Initialization — Faces

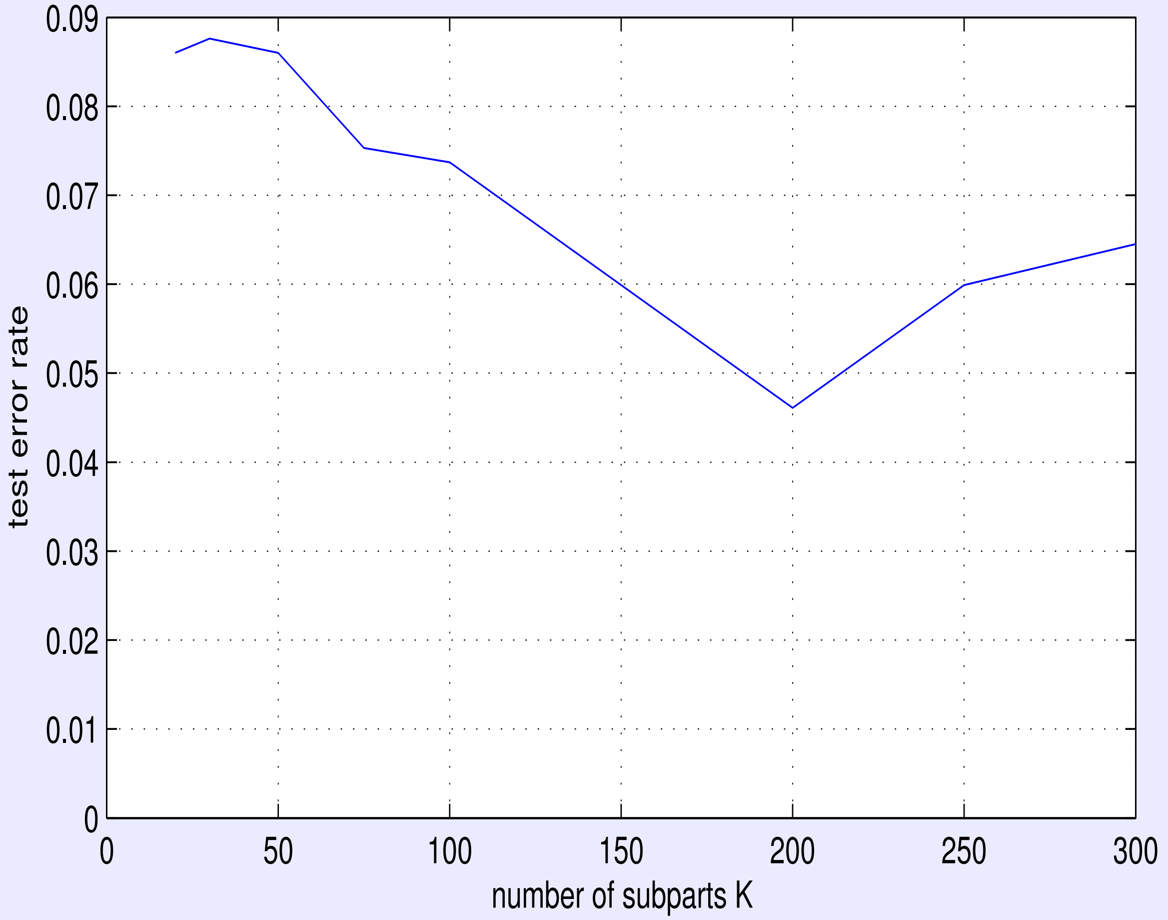


Training Set Fits — Faces / 4 Parts



Number of Parts and Feature Classes





Summary

- Loose part / sub-part hierarchy learns flexible part assemblies.
- Geometrically weaker than constellation model, but handles 100's of features.
- Image features are existence tests for invariant local features.
- Soft assignments allow subparts to be re-parented during learning.
- Training and testing are rapid (< 1 second/image).

In progress

- Poisson field model for feature assignment under texture.

Confusion Matrices — Likelihood based Classification

	3 parts models				one-part model			
err.rate	plane	bike	bkgd	leaves	plane	bike	bkgd	leaves
bike	1.66				2.0			
bkgd	1.4	0.0			2.79	0.0		
leaves	3.31	0.0	2.15		4.97	0.0	8.6	
faces	1.10	0.0	0.0	6.45	2.3	1.0	0.9	12.9