Structure-Perceptron Learning of a Hierarchical Log-Linear Model

Long (Leo) Zhu
Department of Statistics
University of California, Los Angeles
lzhu@stat.ucla.edu

Yuanhao Chen
University of Science and Technology of China
yhchen4@ustc.edu

Xingyao Ye
School of Software
Tsinghua University
yexy04@mails.thu.edu.cn

Alan Yuille
Department of Statistics, Psychology and Computer Science
University of California, Los Angeles
yuille@stat.ucla.edu

Abstract

In this paper, we address the problems of deformable object matching (alignment) and segmentation with cluttered background. We propose a novel hierarchical log-linear model (HLLM) which represents both shape and appearance features at multiple levels of a hierarchy. This model enables us to combine appearance cues at multiple scales directly into the hierarchy and to model shape deformations at short-range, medium range, and long-range. We introduce the structure-perceptron *algorithm to estimate the parameters of the HLLM in a discriminative way. The learning is able to estimate the appearance and shape parameters simultaneously in a global manner. Moreover, the structure-perceptron learning has a feature selection aspect (similar to AdaBoost) which enables us to specify a class of appearance/shape features and allow the algorithm to select which features to use and weight their importance. This method was applied to the tasks of deformable object localization, segmentation, matching (alignment), and parsing. We demonstrate that the algorithm achieves the state of the art performance by evaluation on public dataset (horse and multi-view face).*

1. Introduction

Deformable object matching (alignment) and segmentation are two fundamental tasks in computer vision. They have many applications including object recognition, pose estimation and tracking. These tasks are difficult due to four major reasons – shape deformation, appearance variation, cluttered backgrounds, and occlusion.

Traditional approaches to matching problems, such as deformable templates [11, 1, 14] usually focus only on representing shape deformations without attempting to learn an appearance model. This causes difficulties in situations where there is cluttered background and complex appearance cues are required. Recent works on object segmentation [18, 23, 20] do include complex appearance cues, but these methods do not have internal representations of the object and so they are not suitable for matching (alignment) or for parsing. We will review the literature in section 2.

In this paper, our goal is to learn a hierarchical log-linear object model (HLLM) that can be applied to both segmentation and matching (alignment). This HLLM is able to represent shape and appearance cues at different levels in the hierarchy. This multi-level modeling of appearance differs from most other hierarchical approaches [4, 17, 15, 30] where appearance is only modeled at the bottom level.

We learn the parameters of the HLLM by using the structure-perceptron algorithm. Structure-perceptron enables us to learn all the parameters globally in a consistent manner (i.e. at all levels of the hierarchy simultaneously). Moreover, it enables us to specify a large class of shape/appearance features and allow the algorithm to decide which features should be used and how to weight their importance (similar to AdaBoost). Structure-perceptron learning [7] is a discriminative approach which is computationally simpler than standard methods such as maximum likelihood estimation (as used, for example for learning Conditional Random Fields [19]). Moreover, there are advantages to discriminative learning because this strategy focusses attention on estimating the parameters values of the model most relevant to decision making (e.g. about segmentation or matching).

We demonstrate the success of the HLLM by applying it to segmentation and matching/alignment tasks on large datasets with groundtruth. We use the horse dataset (from Weizmann) and the face alignment dataset. In both cases,

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

we obtain state of the art results. This is perhaps particularly interesting because on the face dataset we are comparing to results obtained by methods such as Active Appearance Models [10] which are specialized for faces.

2. Background

2.1. Deformable Shape Matching

There has been a range of attempts to model deformable objects in order to detect, align, register, and recognize them. Coughlan *et al*. [12] provided a flat Markov Random Field to represent the boundary of object, and showed that dynamic programming (DP) could be used to detect the object. This type of work was extended by Felzenszwalb [14] and by Coughlan using a pruned version of Belief Propagation (BP) [11]. Shape context [1] is a model that has longer-range interaction between features. It is effective when there is a clean background with no clutter, but it is not best suited for cluttered backgrounds. Some algorithms are capable of detecting the boundary contour (e.g. [6, 27]). But these algorithms need initialization in position, orientation, and scale if they are applied to images with cluttered background.

Recent work has introduced hierarchical models to represent the structure of objects more accurately (and enable shape regularities at multiple scales). Shape-trees were presented [15] to model shape deformations at multiple levels. Chen *et al*. [5] propose an AND/OR graph representation (similar to [4, 17]), which is a multi-level mixture Markov Random Field, and provide a novel bottom-up and top-down based inference algorithm. But both of these models concentrate on modeling the shape deformation at different scales and use simple appearance models defined at leaf nodes only.

The main limitation of all above models is that there was no learning algorithm. The image features were manually designed (and hence comparatively simple), the geometry models were hand-specified, and the relative weights of appearance and shape had to be manually tuned. In this paper we define a hierarchy model which is a simplified version of Chen *et al*.'s [5] AND/OR graph. We extend this model to include appearance cues at all scales. We apply the structure-perceptron algorithm to select the most useful shape and appearance features from a vocabulary and simultaneously assign weights to them.

2.2. Object Segmentation

There have recently been a number of advances in object segmentation. In contrast to object matching, the task of object segmentation aims at segmenting the object from background, but not recovering the poses (position, orientation, etc.) of individual parts.

Borenstein and Ullman [3] provide a public horse dataset and study the problem of deformable object segmentation on this dataset. Torr and his colleagues [18] develop Object-Cut which locates the object via a pictorial model learnt from motion cues and use the min-cut algorithm to segment out the object of interest. Ren *et al*. [23] address the segmentation problem by combining low-, mid- and high-level cues in Conditional Random Field (CRF). Similarly, Levin and Weiss [20] utilize CRF to segment object but assuming that the position of the object is roughly given. In contrast to supervised learning, Locus [28] explores a unsupervised learning approach to learn a probabilistic object model. Recently, Cour and Shi [13] achieve the best performance on this standard dataset. It is important to note that none of these methods report performance on matching/alignment.

3. Hierarchical Log-Linear Model

3.1. The model

We represent the appearance and shape of the object by a hierarchical graph defined by parent-child relationships. The top node of the hierarchy represents the position of the center of the object. The leaf nodes represent points at the boundary of the object. The intermediate nodes represent different subparts of the object. The hierarchy is automatically constructed by a hierarchical clustering algorithm, i.e., Segmentation Weighted Aggregation (SWA) [25]. This is illustrated in figure (1). We use ν to index nodes of the hierarchy. The set of child nodes of ν is denoted by T_ν. Thus, T_ν encode all vertical edges of a hierarchy. (μ, ρ, τ), which refer to every triple child nodes of ν, define the horizontal dependencies (horizonal edges) of a hierarchy. $\{\nu, T_\nu, (\mu, \rho, \tau)\}$ fully determine the topology of the hierarchy.

A configuration of the hierarchy is an assignment of state variables $y = \{y_\nu\}$ with $y_\nu = (Px_\nu, Py_\nu, \theta_\nu, s_\nu)$ at each node ν, where (Px, Py), θ and s denote part position, orientation, and scale respectively. The state y of a node is an abstraction (center position, size and orientation) of the state variables of its children. A novel feature of this hierarchical representation is that the node variables y are of the same dimension at all levels of the hierarchy although they denote different subparts, thus have different semantic meanings.

The conditional distribution over all the states is given by a log-linear model:

$$P(y|x;\alpha) = \frac{1}{Z(x;\alpha)} \exp\{\Phi(x,y)\cdot\alpha\}, \qquad (1)$$

where x denotes the input image, y is the parse tree, α are the parameters to be estimated, $Z(x;\alpha)$ is the partition function, $\Phi(x,y)$ are potential functions and $\Phi(x,y)\cdot\alpha$ is

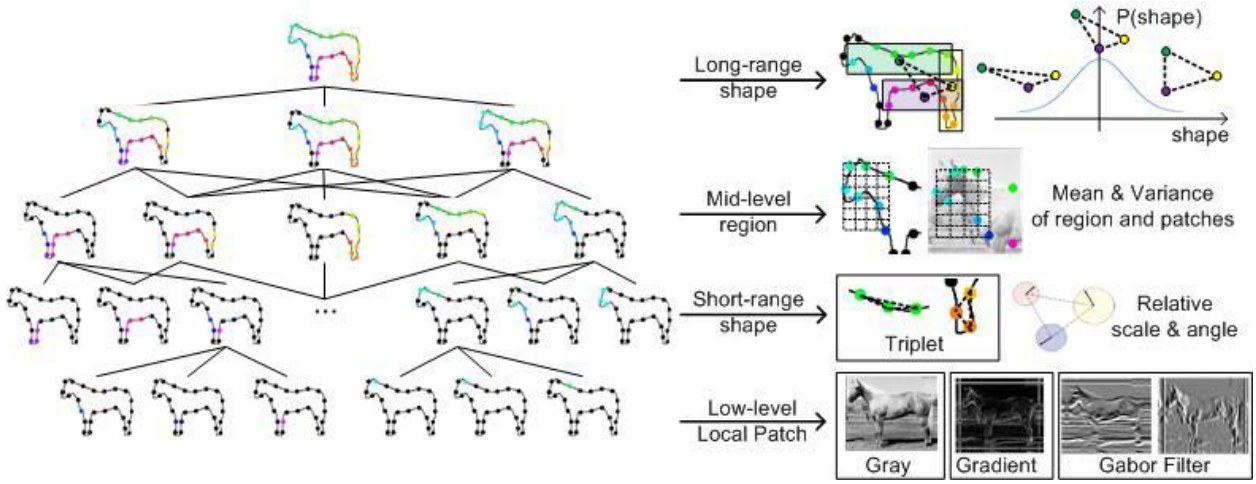

Figure 1. The hierarchy is constructed by Segmentation Weighted Aggregation (an hierarchical clustering algorithm). Black dots indicate the positions of the leaf nodes in the hierarchy. Color dots indicate the subparts. The appearance and shape deformation are modeled at multiple levels in the hierarchy

the inner product $\sum_{i \in Edge} \alpha_i \Phi_i(x, y)$ where the summation is calculated on potential functions defined over edges of the hierarchy.

In this paper, we propose a hierarchical log-linear model (HLLM) which uses potentials $\Phi(x, y)$, see equation (1), defined on the hierarchical graph. The structure-perceptron learning will not compute the partition function $Z(x; \alpha)$. Therefore we do not have a formal probabilistic interpretation. As we will show, this log-linear model is well suited to structure-perceptron learning.

The potential function $\Phi(x, y)$ of HLLM are defined over a hierarchical structure. More specifically, $\Phi(x, y)$ takes three forms: (i) the *data terms* $\Phi^D(x, y)$, (ii) the *horizontal terms* for spatial relations $\Phi^H(y)$, and (iii) the *vertical terms* $\Phi^V(y)$. These terms are defined as follows.

The *data terms* $\Phi^D(x, y)$ are defined over all levels of the hierarchy (see figure 1) and represents image features (of patches and regions). For leaf nodes, $\Phi^D(x, y)$ involves local image features like the grey intensity, gradient, Canny edge, the response of Difference of Offset Gaussian (DOOG) with different scales (13*13 and 22*22) and orientations $(0, \frac{1}{6}\pi, \frac{2}{6}\pi, ...)$, and so on (the bottom row of figure 1). For non-leaf nodes (the second row of the right panel of figure 1), $\Phi^D(x, y)$ represents regional features (e.g., the mean, variance, histogram of image features) whose ranges are determined by y. This differs from most other conditional models – e.g. Textonboost [26], Ren *et al*. [23] and Levin and Weiss [20]'s CRF models – since these contain no hidden variables defined at high levels. For example, y refer to the position of image lattice and their feature size is fixed locally. However, in our case, we are able to access different sizes of features by exploring different y's. This feature design is more flexible and accurate to deal with the geometrical deformations.

The *horizontal terms* and *vertical terms* encode the geometrical prior described in the hierarchy. The *horizontal terms* of the hierarchical shape prior (feature) impose the horizontal connections at a range of scales (see the top and third rows in figure 1). It is defined over all triples μ, ρ, τ formed by the child nodes of each parent. Its form is given by $\Phi^h(y) = g(y_\mu, y_\rho, y_\tau)$, where $g(.,.,.)$ is a logarithm of Gaussian distribution defined on the *invariant shape vector* $l(y_\mu, y_\rho, y_\tau)$ constructed from (y_μ, y_ρ, y_τ) [29]. This shape vector depends only on variables of the triple, such as the internal angles, which are invariant to the translation, rotation, and scaling of the triple. This ensures that the potential is also invariant to these transformations. The parameters of the Gaussian are estimated from training data. Note that the shape features are defined over different levels to allow both short-range and long-range interactions.

The *vertical terms* $\Phi^V(y)$ are used to hold the structure together by relating the state of the parent nodes to the state of their children. The state of the parent node is determined precisely by the states of the child nodes. This is defined by $\Phi^V(y) = h(y_\nu, \{y_\mu \text{ s.t.} \mu \in T_\nu\})$, where T_ν is the set of child nodes of node ν, $h(.,.) = 0$ if the average orientations and positions of the child nodes are equal to the orientation and position of the parent node. If they are not consistent, then $h(.,.) = \kappa$, where κ is a small negative number.

In summary, the hierarchical representation decomposes both the appearance and shape modeling into multiple levels. At low levels of the hierarchy (the third and fourth rows of figure 1), the short-range shape constraints between small parts are modeled together with the small-scale appearance cues. At higher levels (the top and second rows of figure 1), the long-range shape regularities between larger parts are imposed and large scale appearance cues are used.

Input: $\{MP^1_{\nu 1}\}$. Output:$\{MP^L_{\nu L}\}$
Loop : $l = 1$ to L, for each node ν at level l

1. Composition: $\{P^l_{\nu,b}\} = \oplus_{\rho \in T_\nu, a=1,...,M^{l-1}_\rho} MP^{l-1}_{\rho,a}$
2. Pruning: $\{P^l_{\nu,a}\} = \{P^l_{\nu,a} | \Phi_\nu(P^l_{\nu,a}) \cdot \alpha > Threshold_l\}$
3. Local Maximum: $\{(MP^l_{\nu,a}, CL^l_{\nu,a})\} = LocalMaximum(\{P^l_{\nu,a}\}, \epsilon_W)$ where ϵ_W is the size of the window W^l_ν defined in space, orientation, and scale.

Figure 2. The inference algorithm. $\oplus$ denotes the operation of combining two proposals.

3.2. The Parsing Algorithm

We modify the inference algorithm described in [5] to obtain the best parse tree y^* by computing $y^* = \arg\max \Phi(x, y) \cdot \alpha$. This algorithm runs (empirically) in polynomial time in terms of the number of levels of the hierarchy, which is needed to make structure-perceptron learning practical.

The inference algorithm [5] was designed for AND/OR graphs but it can be directly adapted to perform inference for hierarchical models (by treating a hierarchical graphs as a special case of an AND/OR graph which only has AND nodes). The algorithm has a bottom-up and top-down strategy but we only take the bottom-up procedure. The bottom-up stage makes proposals for the configuration of the hierarchy. This proceeds by combining proposals for sub-configurations to build proposals for larger configuration. To prevent a combinatorial explosion we prune out weak proposals which have low fitness score ($\Phi(x, y) \cdot \alpha$ evaluated for the configuration) and use clustering which selects a small set of *max-proposals* to represent each cluster.

We give the pseudo-code for the algorithm in figure 2. The input to a level l is a set of max-proposals $\{MP^{l-1}_{\nu,a}\}$ for each node ν at level $l-1$ (each max-proposal, or proposal, is a configuration $\{Z^{l-1}_{\nu,a}\}$ of the subtree with root node ν). The max-proposals generate proposals $\{P^l_{\mu,b}\}$ for nodes at level l by composition. We prune out this set of proposals by rejecting those with low fitness scores (i.e. $\Phi(x, y) \cdot \alpha$ evaluated for the configuration) and by clustering using *local maximum* to group the proposals into a set of clusters $\{CL^l_{\mu,a}\}$, each represented by a max-proposal $\{MP^l_{\mu,a}\}$ (the local maximum is taken with respect to spatial position, scale, and orientation). The output $\{MP^l_{\mu,a}\}$ is used as input to the next level $l+1$. See [5] for full details.

4. Structure-Perceptron Learning

4.1. Background on Perceptron and Structure-Perceptron Learning

Perceptron learning was traditionally applied to classification tasks [16]. The theoretical properties of convergence and generalization of perceptron learning have been clearly justified by Freund and Shapire *et al*. [16] for binary classification and multi-class classification.

More recently, Collins [7] developed the structure-perceptron algorithm which applies to situations where the output is a structure (e.g. a sequence or tree of states). He proved theoretical results for convergence properties, for both separable and non-separable cases, and for generalization. In addition Collins and his collaborators demonstrated the structure-perceptron for many successful applications in natural language processing, including tagging [8] (an example of a sequence/chain output), and parsing [9] (an example of tree output).

Structure-perceptron learning can be applied to learning log-linear models. The learning proceeds in a discriminative way. By contrast to maximum likelihood learning, which requires calculating the expectation of features, structure-perceptron learning only needs to calculate the most probable configurations (parses) of the model. Therefore, structure-perceptron learning is more flexible and computationally simpler (i.e. the max calculation is usually easier than the sum calculation).

To the best of our knowledge, structure-perceptron has never been exploited in computer vision (unlike the perceptron which has been applied to binary classification and multi-class classification tasks). Moreover, we are applying structure-perceptron to more complicated models (i.e. HLLMs) than those treated by Collins [8] (e.g. Hidden Markov Models for tagging).

4.2. Structure-Perceptron Learning

The goal of structure-perceptron learning is to learn a mapping from inputs $x \in X$ to output structure $y \in Y$. In our case, X is a set of images, with Y being a set of possible parse trees which specify the positions, orientations, scales of objects and their subparts in a hierarchical form. We use a set of training examples $\{(x_i, y_i) : i = 1...n\}$ and a set of functions Φ which map each $(x, y) \in X \times Y$ to a feature vector $\Phi(x, y) \in R^d$. The task is to estimate a parameter vector $\alpha \in R^d$ for the weights of the features. The feature vectors $\Phi(x, y)$ can include arbitrary features of parse trees, as we discussed in section 3.1.

The loss function used in structure-perceptron learning is usually of form:

$$Loss(\alpha) = \Phi(x, y) \cdot \alpha - \max_{\overline{y}} \Phi(x, \overline{y}) \cdot \alpha, \qquad (2)$$

where y is the correct structure for input x, and $\overline{y}$ is a dummy variable.

The basic structure-perceptron algorithm – *Algorithm I* – is designed to minimize the loss function. Its pseudo-code is given in figure 3. The algorithm proceeds in a simple way (similar to the perceptron algorithm for classification). The parameters are initialized to zero and the algorithm loops over the training examples. If the highest scoring parse tree

Input: A set of training images with ground truth (x^i, y^i) for $i = 1..N$. Initialize parameter vector $\alpha = 0$.

Algorithm I:

For $t = 1..T, i = 1..N$

- Use bottom-up inference to find the best state of the model on the i'th training image with current parameter setting, i.e., $y^* = \arg\max_y \Phi(x^i, y) \cdot \alpha$
- Update the parameters: $\alpha = \alpha + \Phi(x^i, y^i) - \Phi(x^i, y^*)$

Output: Parameters α

Figure 3. Algorithm I: a simple training algorithm of structure-perceptron learning

for input x is not correct, then the parameters α are updated by an additive term. The most difficult step of the method is finding $y^* = \arg\max_y \Phi(x^i, y) \cdot \alpha$. This is precisely the parsing (inference) problem. Hence the practicality of structure-perceptron learning, and its computational efficiency, depends on the inference algorithm. As discussed earlier, see section (3.2), the inference algorithm has polynomial computational complexity for an HLLM which makes structure-perceptron learning practical for HLLMs.

4.3. Averaging Parameters

There is a simple refinement to Algorithm I, called *"the averaged parameters"* method (Algorithm II), whose pseudo-code is given in figure 4. The averaged parameters are defined to be $\gamma = \sum_{t=1}^{T} \sum_{i=1}^{N} \alpha^{t,i}/NT$, where NT is the total number of iterations. It is straightforward to store these averaged parameters and output them. The theoretical analysis in [7] shows that Algorithm II (with averaging) gives better performance and convergence rate than Algorithm I (without averaging). We will empirically compare these two algorithms.

Algorithm II:

For $t = 1..T, i = 1..N$

- Parse: $y^* = \arg\max_y \Phi(x^i, y) \cdot \alpha$
- Store: $\alpha^{t,i} = \alpha$
- Update: $\alpha = \alpha + \Phi(x^i, y^i) - \Phi(x^i, y^*)$

Output: Parameters $\gamma = \sum_{t,i} \alpha^{t,i}/NT$

Figure 4. Algorithm II: a modification of algorithm I.

4.4. Feature Selection

We emphasize that structure-perceptron learning can be considered as a procedure of feature selection (similar to AdaBoost). The training algorithm sequentially gives more weight to features which are more distinct between the groundtruth and the parse returned by the model. This feature selection property allows us to specify a large dictionary of possible features and enable the algorithm to select those features which are most effective. This allows us to learn HLLMs for different objects *without* needing to specially design features for each object. Moreover, the same mechanism allows us to automatically select from features defined at multiple levels according to the unobserved (or hidden) hierarchical configuration y. Thus our approach is more flexible than existing conditional models (e.g., CRF [23, 20, 26]) which use multi-level features but with fixed relative positions (i.e. not adaptive). In section 5.3, we empirically study what features the structure-perceptron algorithm judges to be most important for a specific object like a horse. Section 5.4 also illustrates the advantage of feature selection by applying the same learning algorithm to the different task of face alignment without additional feature design.

5. Experimental Results

5.1. Dataset and Evaluation Criterions

We use a standard public dataset, the Weizmann Horse Dataset [3], to perform experimental evaluations for HLLMs. This dataset is designed to evaluate segmentation, so the groundtruth only gives the regions of the object and the background. To supplement this groundtruth, we required students to manually parse the images by locating the positions of leaf nodes of the hierarchy in the images. These parse trees are used as ground truth to evaluate the ability of the HLLM to parse the horses (i.e. to identify different parts of the horse). There are a total of 328 images which are divided into three subsets – 50 for training, 50 for validation, and 228 for testing. The parameters learnt from the training set, and with the best performance on validation set, are selected.

To show the generality of our approach, and its ability to deal with different objects without hand-tuning the appearance features, we apply it to the task of face alignment. The dataset [22] contains ground truth of standard 65 key points which lie along the boundaries of face components with semantic meaning, i.e, eyes, nose, mouth and cheek. We use part of this dataset for training (200 images) and part for testing (80 images).

For a given image x, the parsing results are obtained by estimating the configuration y of the HLLM. To evaluate the performance of parsing (for horses) and matching/alignment (for faces) we use the **average position error** measured in terms of pixels. This quantifies the average distance between the positions of leaf nodes of the ground truth and those estimated in the parse tree.

The HLLM does not directly output a full segmentation of the object. Instead the set of leaf nodes gives a sparse

Table 1. Performance of Detection and Parsing

Training	Validation	Detection	Parsing	Speed
50	50	99.1%	16.04	23.1s

Table 2. Comparisons of Segmentation Performance on Weizmann Horse Dataset

Methods	Testing	Seg. Accu.	Pre./Rec.
Our approach	228	94.7%	93.6% / 85.3%
Ren [23]	172	91.0%	86.2%/75.0%
Borenstein [2]	328	93.0%	
LOCUS [28]	200	93.1%	
Cour [13]	328	94.2%	
Levin [20]	N/A	95.0%	
OBJ CUT [18]	5	96.0%	

estimate for the segmentation. To enable HLLM to give full segmentation we modify it by a strategy inspired by grab-cut [24] and obj-cut [18]. We use a rough estimate of the boundary by sequentially connecting the leaf nodes of the HLLM, to initialize a grab-cut algorithm (recall that standard grab-cut [24] requires human initialization, while obj-cut needs motion cues). We use **segmentation accuracy** to quantify the proportion of the correct pixel labels (object or non-object). In addition, we also report precision/recall, see [23], which has the advantage that it does not depend on the relative size of the object and the boundary. (For example, you can get 80% segmentation accuracy on the weizmann horse dataset by simply labelling every pixel as background). We note that segmentation accuracy is commonly used in the computer vision community, while precision/recall is more standard in machine learning.

We rate *detection* to be successful if the area of intersection of the labeled object region (obtained by graph-cut initialized by the HLLM) and the true object region is greater than half the area of the union of these regions.

5.2. Experiment I: Deformable Object Detection, Segmentation and Parsing

The best parse tree is obtained by performing inference algorithm over HLLM learnt by structure-percepton learning. Figure 5 shows several parsing and segmentation results. The states of the leaf nodes of parse tree indicate the positions of the points along the boundary which are represented as colored dots. The points of same color in different images correspond to the same semantic part. One can see our model's ability to deal with shape variations, background noise, textured patterns, and changes in viewing angles. The performance of detection and parsing on this dataset is given in Table 1. The localization rate is around 99%. Our model performs well on the parsing task since the average position error is only 16 pixels (to give context, the radius of the color circle in figure 5 is 5 pixels). Note no other papers report parsing performance on this dataset since most (if not all) methods do not estimate the positions of different parts of the horse (or even represent them). The time of inference for image with typical size 320×240 is 23 seconds.

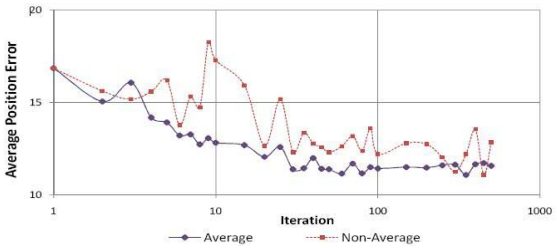

Figure 6. The average position errors (y-axis) across iterations (x-axis) are compared between Algorithm-II(average) and Algorithm-I (non-average).

In table 2, we compare the segmentation performance of our approach with other successful methods. Note that the object cut method [18] was reported on only 5 images. Levin and Weiss [20] make the strong assumption that the position of the object is given (other methods do not make this assumption) and do not report how many images they tested on. Overall, Cour and Shi's method [13] was the best one evaluated on large dataset. But their result is obtained by manually selecting the best among top 10 results (other methods output a single result). By contrast, our approach outputs a single parse only but yields a higher pixel accuracy of 94.7%. Hence we conclude that our approach outperforms those alternatives which have been evaluated on this dataset. As described above, we prefer the precision/recall criteria [23] because the segmentation accuracy is not very distinguishable (i.e. the baseline starts at 80% accuracy, obtained by simply classifying every image pixel as being background). Our algorithm outperforms the only other method evaluated in this way (i.e. Ren *et al.*'s [23]). For comparison, we translate Ren *et al.*'s performance (86.2%/75.0%) into segmentation accuracy of 91% (note that it is impossible to translate segmentation accuracy back into precesion/recall).

5.3. Diagnosis

In this section, we will conduct diagnosis experiments to study the behavior of structure-perceptron learning.

Convergence Analysis. Figure 6 shows the average position error on training set for both Algorithm II (averaged) and Algorithm I (non-averaged). It shows that the averaged algorithm converges much more stablely than non-averaged algorithm.

Generalization Analysis. Figure 7 shows average position error on training, validation and testing set over a num-

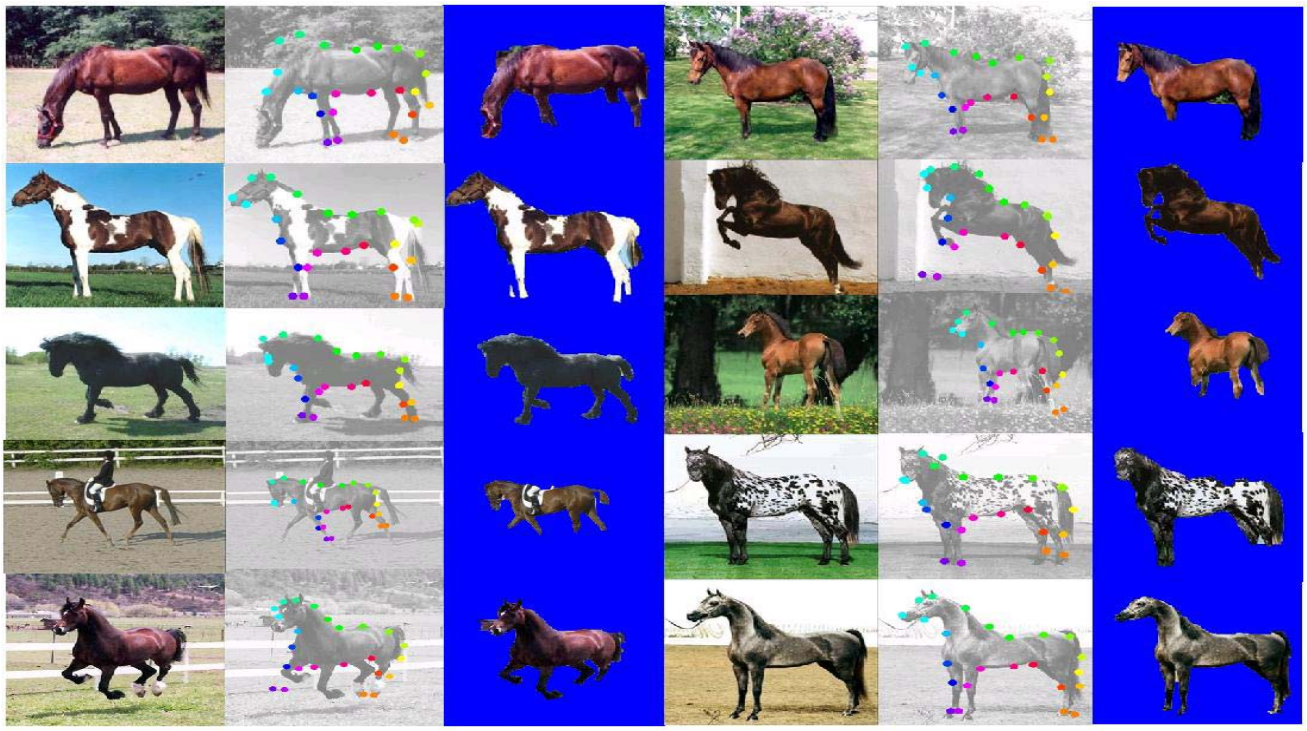

Figure 5. Examples of Parsing and Segmentation. Column 1 , 2 and 3 show the raw images, parsing and segmentation results respectively. Column 4 to 6 show extra examples. Parsing is illustrated by dotted points which indicate the positions of leaf nodes (object parts). Note that the points in different images with the same color correspond to the same semantical part.

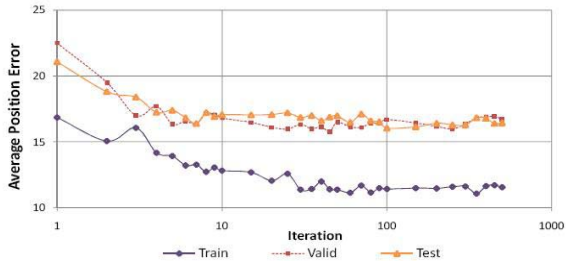

Figure 7. The average positions errors on training, validation and testing dataset are reported.

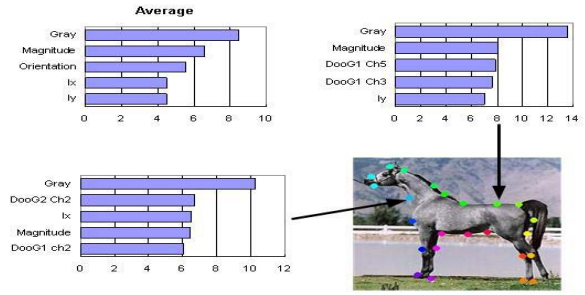

Figure 8. Weights of Features. The most useful features overall are gray value, magnitude and orientation of gradient, and difference of intensity along horizontal and vertical directions (Ix and Iy). DooG1 Ch5 means Difference of offset Gaussian (DooG) at scale 1 (13*13) and channel (orientation) 5 ($\frac{4}{6}\pi$).

ber of training iterations. Observe that the behavior on the validation set and the testing set are quite similar. This confirms that the selection of parameters decided by the validation set is reasonable.

Feature Selection. We show the features learnt from structure-perceptron learning in figure (8) (features are shown at the bottom level only for reasons of space). The top 5 features, ranked according to their weights, are listed. The top left, top right and bottom left panels show the top 5 features for all leaf nodes, the node at the back of horse and the node at the neck respectively. Recall that structure-perceptron learning performs feature selection by adjusting the weights of the features.

5.4. Experiment II: Multi-view Face Alignment

The task of multi-view face alignment has been much more thoroughly studied than horse parsing. Our HLLM approach, using identical settings for horse parsing, *achieves an average distance error of* 6.0 *pixels, comparable with the best result* 5.7 *pixels*, obtained by [21]. Their approach is based mainly on the Active Appearance Models [10] which were motivated specifically to model faces and which assume that the shape deformations are mostly rigid. By contrast, our HLLMs are suitable for both rigid and deformable objects and required no special training to apply to this problem. Figure 9 shows the typical parse results for face alignment.

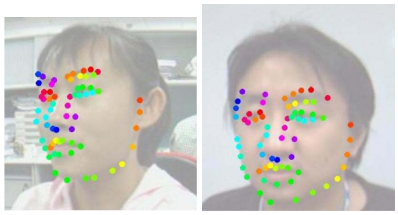

Figure 9. Multi-view Face Alignment.

6. Conclusion

We developed a hierarchical log-linear model (HLLM) for representing objects which can be learnt by adapting the structure-perceptron algorithm used in machine learning. Advantages of our approach include the ability to select shape and appearance features at a variety of scales in an automatic manner.

We demonstrated the effectiveness and versatility of our approach by applying it to very different problems, evaluating it on large datasets, and giving comparisons to the state of the art. Firstly, we showed that the HLLM outperformed other approaches when evaluated for segmentation on the weizmann horse dataset. It also gave good results for parsing horses (where we supplied the groundtruth), though there are no other parsing results reported for this dataset. Secondly, we applied HLLMs to the completely different task of multi-view face alignment (without any parameter tuning or selection of features) and obtained results very close to the state of the art.

7. Acknowledgments

This research was supported by NSF grant 0413214.

References

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002.

[2] E. Borenstein and J. Malik. Shape guided object segmentation. In *CVPR (1)*, pages 969–976, 2006.

[3] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *ECCV (2)*, pages 109–124, 2002.

[4] H. Chen, Z. Xu, Z. Liu, and S. C. Zhu. Composite templates for cloth modeling and sketching. In *CVPR (1)*, pages 943–950, 2006.

[5] Y. Chen, L. Zhu, C. Lin, A. L. Yuille, and H. Zhang. Rapid inference on a novel and/or graph for object detection, segmentation and parsing. In *NIPS*, 2007.

[6] H. Chui and A. Rangarajan. A new algorithm for non-rigid point matching. In *CVPR*, pages 2044–2051, 2000.

[7] M. Collins. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8, 2002.

[8] M. Collins and N. Duffy. New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron. In *ACL*, pages 263–270, 2001.

[9] M. Collins and B. Roark. Incremental parsing with the perceptron algorithm. In *ACL*, page 111, 2004.

[10] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *ECCV (2)*, pages 484–498, 1998.

[11] J. M. Coughlan and S. J. Ferreira. Finding deformable shapes using loopy belief propagation. In *ECCV (3)*, pages 453–468, 2002.

[12] J. M. Coughlan, A. L. Yuille, C. English, and D. Snow. Efficient deformable template detection and localization without user initialization. *Computer Vision and Image Understanding*, 78(3):303–319, 2000.

[13] T. Cour and J. Shi. Recognizing objects by piecing together the segmentation puzzle. In *CVPR*, 2007.

[14] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[15] P. F. Felzenszwalb and J. D. Schwartz. Hierarchical matching of deformable shapes. In *CVPR*, 2007.

[16] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

[17] Y. Jin and S. Geman. Context and hierarchy in a probabilistic image model. In *CVPR (2)*, pages 2145–2152, 2006.

[18] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Obj cut. In *CVPR (1)*, pages 18–25, 2005.

[19] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[20] A. Levin and Y. Weiss. Learning to combine bottom-up and top-down segmentation. In *ECCV (4)*, pages 581–594, 2006.

[21] H. Li, S.-C. Yan, and L.-Z. Peng. Robust non-frontal face alignment with edge based texture. *J. Comput. Sci. Technol.*, 20(6):849–854, 2005.

[22] S. Z. Li, H. Zhang, S. Yan, and Q. Cheng. Multi-view face alignment using direct appearance models. In *FGR*, pages 324–329, 2002.

[23] X. Ren, C. Fowlkes, and J. Malik. Cue integration for figure/ground labeling. In *NIPS*, 2005.

[24] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004.

[25] E. Sharon, A. Brandt, and R. Basri. Fast multiscale image segmentation. In *CVPR*, pages 1070–1077, 2000.

[26] J. Shotton, J. M. Winn, C. Rother, and A. Criminisi. *TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV (1)*, pages 1–15, 2006.

[27] Z. Tu and A. L. Yuille. Shape matching and recognition - using generative models and informative features. In *ECCV (3)*, pages 195–209, 2004.

[28] J. M. Winn and N. Jojic. Locus: Learning object classes with unsupervised segmentation. In *ICCV*, pages 756–763, 2005.

[29] L. Zhu, Y. Chen, and A. L. Yuille. Unsupervised learning of a probabilistic grammar for object detection and parsing. In *NIPS*, pages 1617–1624, 2006.

[30] L. Zhu and A. L. Yuille. A hierarchical compositional system for rapid object detection. In *NIPS*, 2005.